SG-Lib 5.5

Function Reference

Tim C. Lueth
Professor at Technical University of Munich (TUM), Germany
generated 14 July 2026

Function Reference — SG-Lib 5.5

© 2010–2026 Tim C. Lueth, Professor at Technical University of Munich (TUM), Germany.

SG-Lib Academic License (v1.0)

This license applies to the parts of SG-Lib authored by Tim C. Lueth. Third-party files (e.g. MATLAB Central / File Exchange contributions) retain their own licenses and are NOT covered by this text.

1. Intellectual property. The SG-Lib files authored by Tim C. Lueth are protected intellectual property.

2. Grant of use. Permission is granted, free of charge, to accredited universities and their members to use SG-Lib solely for non-commercial academic research and teaching. This includes calling, linking and embedding the UNMODIFIED functions in the user's own research code.

3. No modification, no derivatives. The source code may not be modified, and no derivative works may be created from it, without the author's prior written consent. (Using and integrating the unmodified code as described in clause 2 is expressly permitted.)

4. No redistribution. SG-Lib, or any part of it, may not be redistributed, sublicensed or published without the author's prior written consent.

5. No commercial use. Any commercial use, in whole or in part, requires a separate written license from the author. Requests: tim.lueth@tum.de

6. Attribution. This copyright notice and license must be retained in every copy and in the header of each covered file.

7. No warranty. Provided “as is”, without warranty of any kind, express or implied. The author is not liable for any damages arising from its use.

8. Governing law. German law applies. Place of jurisdiction is Munich, Germany.

9. Third-party components. SG-Lib uses third-party software, each under its own license: Manifold / manifold3d (mesh boolean and 2D projection) — Apache License 2.0, © The Manifold Authors (see LICENSE_thirdparty_Manifold_Apache2.txt). Any other third-party files retain their own licenses.

Contact for permissions and commercial licensing: Prof. Dr. Tim C. Lueth, TUM — tim.lueth@tum.de

---

Acknowledgments and Thanks

SG-Lib is the work of Tim C. Lueth. Over the years it has also come to rely on a small number of excellent utility functions written by others. They are used here within an academic, non-commercial research library, with gratitude.

All rights to the third-party functions listed below remain with their respective authors. Their own licenses apply to those files; the SG-Lib Academic License does NOT cover them. Sincere thanks to:

- select3d, SGisInterior, gcfSG, tangentRP, screencapture — contributions from the MATLAB Central community - ellipseofPL — Ohad Gal (BSD license, (c) 2003 Ohad Gal) - lastcommand — Adam Danz - marchingcubes — Peter Hammer - plothatch — Iram Weinstein - sphere_line_intersection Paul Bourke & Campbell Barton - userIP — Michael Kleder (and Stephen L.) - username — Daniel Shub (snippet found online, 2018)

If any author listed here would like a correction to their attribution, or does not wish their function to be included, please contact Tim C. Lueth (tim.lueth@tum.de) and it will be addressed promptly.

Warm thanks to the students, doctoral researchers and staff of Tim Lueth's group who contributed ideas and code to SG-Lib over the years. Named here with gratitude (alphabetical by surname):

Valentin Ameres, Sebastian Baumgartner, Martin Bossert, Samuel Detzel, Christian Dietz, Sebastian Heininger, Martin Helm, Franz Irlinger, Yannick Krieger, Michael Kruttschnitt, Simon Laudahn, Brian Li, Marie Lueth, Alexandra Mercader, Franz Moser, Nicholas Moser, Stephan Nowatschin, Korbinian Rzepka, Simon Schiele, Florian Schleich, Quirin Schmid, Robin Schregle, Andreas Schröffer, Christoph Stockhammer, Yilun Sun, Mattias Träger, Julia Zygar

Additional individual functions carry further author credits directly in their file headers; if a name is missing here or needs correction, please contact Tim C. Lueth.

Tim C. Lueth, Munich - tim.lueth@tum.de

Table of Contents

1 Data Structure and Manipulation **46**

1.1 Point List Manipulation . . . 46
1.2 Vertex List Manipulation . . . 56
1.3 Transformation Matrices . . . 59
1.4 CPL as NaN Separated List . . . 61
1.5 Edge List Manipulation (PL/VL) . . . 69
1.6 Facet List Manipulation (PL/VL) . . . 77
1.7 Tetrahedron List Manipulation . . . 83
1.8 VLFL Manipulation . . . 85
1.9 SG Manipulation . . . 98

2 Analytical Geometry **100**

2.1 Graphical User Interface . . . 100
2.2 Geometry Drawings . . . 113
2.3 Plot Annotation . . . 132
2.4 Crossing points and tangents . . . 133
2.5 Planar and Spatial Transformation . . . 144
2.6 Curves & Splines . . . 154
2.7 Normals & Vectors . . . 155
2.8 Distances & Metrics . . . 157
2.9 Convex Hull & Registration . . . 162

3 Polygons and Contours **164**

3.1 Single 2D Polygons (PL) . . . 164
3.2 Closed Nested 2D Polygons (CPL) . . . 165
3.3 Planar 3D Polygons (VL) . . . 189
3.4 Spatial 3D Polygons (CVL) . . . 189
3.5 General PL functions . . . 193
3.5.1 PL modification . . . 193
3.5.2 Crossing Points . . . 193
3.5.3 CPL Boolean Operations . . . 193
3.5.4 PL Comparison . . . 195
3.6 CPL to Surfaces . . . 195
3.7 CVL to Solid Geometries . . . 195

4 Surfaces **196**

4.1 2D Surfaces . . . 196
4.2 3D Surfaces . . . 200

4.3 Feature Surfaces . . . 200
4.4 Surfaces to Solid Geometries . . . 203

5 Solid Geometry Modeling 204

5.1 SG basic shapes . . . 204
5.2 SG Generation . . . 212
5.2.1 SG of CPL Extrusion . . . 212
5.2.2 SG of CPL Connections . . . 216
5.2.3 SG of Meshes . . . 217
5.2.4 SG from CVL, Surfaces & Floorplan . . . 219
5.3 Spatial Transformation . . . 223
5.4 SG Modification . . . 225
5.5 SG Boolean Operations . . . 231
5.6 SG Cutting & Slicing . . . 233
5.6.1 SG Cutting . . . 233
5.6.2 SG Slicing . . . 235
5.7 SG Connection & Joints . . . 238
5.8 SG Queries & Properties . . . 241
5.9 SG Selection & Grouping . . . 245
5.9.1 Selection . . . 245
5.9.2 Grouping & Collections . . . 246

6 Frames and Kinematics 248

6.1 Frame Definition . . . 248
6.2 Kinematics Chains . . . 249
6.3 Forward and Inverse Kinematic . . . 267
6.4 Symbolic Definitions . . . 268
6.5 Fourbar Linkage . . . 269
6.6 Linkages and Coupler Curves . . . 270
6.6.1 Coupler Curves and Joint Kinematics . . . 270
6.6.2 Pose Definition and Manipulation . . . 275
6.6.3 Four-bar Synthesis and Analysis . . . 282
6.6.4 Four-bar and Hinge Solids . . . 294

7 Automatic Design, Engineering & Simulation 296

7.1 Parameterized Shape Generation . . . 296
7.2 Parameterized Links and Joints . . . 308
7.3 Automatic Shape Generation . . . 310
7.4 FEM and Shape Optimization . . . 314
7.4.1 Mesh and Model Creation . . . 314
7.4.2 Load Definition and FS . . . 314
7.4.3 Stress and Displacement . . . 315

7.4.4 Shape Optimization . . . 315
7.5 Multi-Body Simulation . . . 316
7.5.1 Simscape Model and Block Assembly . . . 316
7.5.2 Simscape Library and Blocks . . . 322
7.5.3 Simulation Run and Output . . . 322
7.6 Engineering and CAD Standard Parts . . . 323
7.6.1 DIN/ISO Standards . . . 323
7.6.2 Standard Components (DIN/ISO parts) . . . 326
7.6.3 Automatic Part Designs . . . 329
7.6.4 Fits (interference / clearance) . . . 334
7.6.5 Dimensioning . . . 334
7.6.6 Completed CAD Designs . . . 336
7.7 Robotics, Hardware and Sensors . . . 339
7.7.1 Serial Robots (JACO, Raambo, FeeTech) . . . 339
7.7.2 FeeTech Motor Control . . . 342
7.7.3 USB and Serial Interfaces . . . 343
7.7.4 RealSense Cameras . . . 345
7.7.5 NDI Vicra Tracking . . . 345
7.8 Design Automation Concepts . . . 346
7.8.1 Script Language . . . 346
7.8.2 Jarvis Concept . . . 347
7.8.3 Claude Interface . . . 348
7.8.4 chatGPT Interface . . . 348

8 STL & 3D Printing & Voxel Models 351

8.1 STL Import/Export . . . 351
8.2 STL Packaging . . . 355
8.3 STL Surfaces Analysis . . . 358
8.4 Vector Export (SVG/DXF) . . . 359
8.5 Voxel Models . . . 359
8.5.1 VM Import & Scaling . . . 359
8.5.2 VM Visualization . . . 361
8.5.3 VM to VL, CPL . . . 364
8.5.4 VM to SG Surface Model . . . 364

9 Testing, Development, AI & Media 366

9.1 Sample Generation . . . 366
9.2 Test Generation . . . 369
9.3 SG Check . . . 369
9.4 SG Repair . . . 373
9.5 Software Development . . . 376
9.5.1 File & Folder Handling . . . 376

9.5.2 System, macOS & Hardware Tools . . . 381
9.5.3 Toolbox Build & Documentation . . . 383
9.5.4 Debugging . . . 385
9.5.5 String, Cell & Data Utilities . . . 386
9.5.6 Media & Document Files . . . 395
9.5.7 Web & Company Data . . . 395
9.5.8 General / Misc Helpers . . . 397
9.6 AI, Data and Integration . . . 413
9.6.1 LLM / ChatGPT (OpenAI) . . . 413
9.6.2 Expert Systems and STRIPS Planning . . . 418
9.6.3 AI Text, String and Cell Helpers . . . 420
9.6.4 Web and External Data Sources . . . 426
9.6.5 FileMaker Interface . . . 426
9.6.6 Encryption . . . 427
9.7 Video, Audio and Media . . . 427
9.7.1 Video . . . 427
9.7.2 Audio . . . 429
9.8 Tutorials and Worked Examples . . . 431
9.8.1 Guided Tutorials . . . 431
9.8.2 Worked Examples (VLFL_EXP) . . . 432

New and not yet assigned **438**

Part I

Guide

A guided tour of the library, chapter by chapter. Each chapter heading links to its detailed function section in Part II; function names in the text jump straight to their entry. Use it to understand what each category is for and how the pieces fit together.

About the SG-Lib

The Solid Geometry Library (SG-Lib) is a MATLAB toolbox for designing, simulating and shape-optimizing geometric bodies, mechanisms, kinematics and complete robots as multi-body systems. Unlike CSG modeling, it is built entirely on surface (mesh) data, and it can import STL files and medical image data so a design can be combined with real machine elements or patient-specific anatomy. It has been developed since 2010 with one goal: to design robots and mechanisms automatically, so that a task-specific device for medicine and care - manufactured by 3D printing and laser cutting - can be created in a fraction of the usual time and with a minimum number of drives.

There are two ways into the library. If you want to USE it for design, optimization and dynamic simulation, start with Solid Geometry Modeling (Chapter 5). If you want to UNDERSTAND and EXTEND it, start with the Data Structures (Chapter 1).

The reference follows the construction workflow. Chapter 1 (Data Structure and Manipulation) defines the core types. Chapter 2 (Analytical Geometry) provides drawing, transformation and measurement primitives. Chapter 3 (Polygons and Contours) and Chapter 4 (Surfaces) build the 2D and surface geometry that Chapter 5 (Solid Geometry Modeling) turns into solids and then assembles, cuts and checks. Chapter 6 (Frames and Kinematics) adds motion and mechanisms. Chapter 7 (Automatic Design, Engineering & Simulation) automates design including FEM, multi-body simulation and standard parts. Chapter 8 (STL & 3D Printing & Voxel Models) prepares parts for manufacturing. Chapter 9 (Testing, Development, AI & Media) collects checking, repair, developer and integration tools. A function's first one or two letters usually name its result type (e.g. VLof... returns a vertex list); the library is deliberately not object-oriented so that tab-completion of these typed prefixes stays fast.

1 Data Structure and Manipulation ›

SG-Lib is deliberately not object-oriented: the authors found that tab-completion of function-name prefixes — which barely works with class methods — is far more valuable for fast exploration than encapsulation. Instead, the library relies on a small set of plain data structures, and each function name signals its return type through its first one to three letters.

The core types are: PL (2D point list), VL (3D vertex list), R (rotation matrix, [2x2] in 2D or [3x3] in 3D), HT (homogeneous transformation matrix, [3x3] in 2D or [4x4] in 3D), CPL/CVL (closed 2D/3D point lists, individual contours separated by a [NaN NaN] row), EL (edge list), FL (facet list), TL (tetrahedron list; note that TL today more often denotes a frame/transformation list), CEL/CIL/SEL (sorting/index lists that group EL, FL or TL entries into contours or meshes), and SG (a struct of VL and FL, or a cell array of such structs, describing a solid body) together with its transform-carrying variant SGT.

This chapter collects the functions that build and manipulate these underlying lists, independent of any particular solid. Section 1.1 (Point List Manipulation) covers 2D point lists (PL). Section 1.2 (Vertex List Manipulation) covers 3D vertex lists (VL) and point clouds (PC). Section 1.3 (Transformation Matrices) covers R and HT matrices. Section 1.4 (CPL as NaN-Separated List) covers the closed-contour format used throughout the library. Section 1.5 (Edge List Manipulation), 1.6 (Facet List Manipulation) and 1.7 (Tetrahedron List Manipulation) cover EL, FL and TL respectively. Section 1.8 (VLFL Manipulation) covers the combined vertex/facet surface representation, and Section 1.9 (SG Manipulation) covers the solid-geometry struct itself.

1.1 Point List Manipulation ›

A point list (PL) is the simplest 2D structure in SG-Lib: a plain n-by-2 matrix of x/y rows, PL = [x1 y1; x2 y2; x3 y3; ...]. Example: PL = [0 0; 10 0; 10 10; 0 10]. Unlike a CPL, the first and last point are NOT identical — a PL is treated as an open sequence, not an automatically closed contour. Try PL = rand(10,2); PLplot(PL); view(0,90) to see one plotted.

Contour construction and repair form the largest cluster: PLcat concatenates point lists, PLsupplement adds missing points to a contour, PLconnectopenends closes an open line, PLoutercontour removes self-crossings, and PLreassign, PLsortC, PLminxy/PLminyx reorder or pick reference points.

Geometric primitives and shape generators cover circles, arcs and profiles: PLcirclearc, PLtangentcirc/PLtangentcirc3/PLtangentcirc4P, PLradialEdges, PLrhombus, PLtrapezoid, PLegg, PLhook, PLarrow, PLknurled, PLwrench/PLinbus, and thread/gear profiles such as PLthreadDIN, PLthreadtrapezoidalDIN, PLtoothevolvente and PLgearSIN.

Sampling, spacing and analysis functions resample or measure contours: PLofCPLequidistant, PLonCPLcontour, CPLlength, PLshortcut, PLgrow, PLcircleofPL, and the grid/potential-field helpers GPLdistancetoCPL, GPLparameters, GPLpotentialpointsCPS, matrixofCPL.

Conversion and correlation functions translate between representations or match two contours: PLofVL, PLofVLFL, PLofFloorPlan, PLofimcontourc, PLtransform/PLtransform_2018, PLcorrelateaux, PLcorrelatingsegments, and PLsetdiff. Rule of thumb: reach for CPL instead of PL as soon as a contour must be explicitly closed or may contain several separate loops.

1.2 Vertex List Manipulation ›

A vertex list (VL) is the 3D counterpart of a PL: a plain n-by-3 matrix of x/y/z rows, VL = [x1 y1 z1; x2 y2 z2; x3 y3 z3; ...]. Example: VL = [0 0 0; 10 0 0; 10 10 0; 0 10 0]. As with PL, the first and last point are NOT the same — that distinguishes a VL from a CVL. Try SGfigure; VL = rand(10,3); VLplot(VL); view(-30,30) to visualize one. A related but distinct type is the point cloud (PC): unlike an ordered VL that traces a path, a PC is an unordered set of points, typically from scans or sampling.

Basic list operations combine, clean and normalize vertex data: VLcat concatenates lists, VLremsuccident removes consecutive duplicate rows, VLaddz lifts a 2D PL into 3D by appending z = 0, VLnorm normalizes vectors, VLcorrelate finds matching indices between two lists, and VLdownsample thins a list.

Point-cloud functions handle unordered data: PCofVL / PCofSG convert a VL or an SG into a point cloud, PCplot displays one, PCremovenoise filters outliers, PCfindNearestNeighbors and VLfindorthoplanes / PCfindplanes analyze local structure, and PCwriteXYZ / VLreadPLY exchange data with external formats (XYZ, PLY).

Geometric analysis and generation functions compute derived quantities or new samples: VLeigenvect returns eigenvectors/eigenvalues of a point set, PLofVLcoplanar and PLofVLplaneprojection (see Section 1.1) flatten coplanar or projected data, and VLfibonacciSphere distributes points evenly on a sphere.

Mesh/grid-image helpers connect VLs to raster or depth data: VLmeshgrid, VLmeshgridselect, VLofimsegthres, and FLofVLonsphereprojection / VLFLselectcrossingfacets bridge to the facet-list world of Section 1.6.

1.3 Transformation Matrices ›

SG-Lib represents spatial relations with translation vectors, rotation matrices R ([2x2] in 2D or [3x3] in 3D), and homogeneous transformation matrices HT ([3x3] in 2D or [4x4] in 3D) whose last row is always [0 0 0 1]; scaling and shearing are not evaluated. Transformations from Peter Corke's Robotics Toolbox are supported alongside SG-Lib's own conventions. Example: TofR(rot(0,0,pi/4)); view(-30,30);

Construction from geometric input builds an HT from points, features or an existing R: TofR converts a rotation matrix to an HT, T3P builds an HT from three points (right-handed), TofDPhi builds a 2D link HT from distance and angle, HTofT pads a 2x2/3x3 matrix into a full 4x4 HT.

Construction from solids and surfaces derives frames from geometry already in a scene: TofVLUL / TofVLULez return the HT at the origin of a union area, TofFS returns the HT of a specified feature surface, TofSG2022 sets an HT relative to a (possibly nested) SG, and TofPCVL returns the HT for a planar contour vertex list.

Camera and orientation helpers cover special cases: Tofcam returns the HT of a camera and its target for a given camera position, and Tupright rotates an HT so its z-axis points upward. Rule of thumb: build the HT once from the most specific geometric context available (three points, a feature surface, or an SG frame), then propagate it with the standard 4x4 matrix product rather than recomputing rotation and translation separately.

1.4 CPL as NaN Separated List ›

MATLAB itself uses several incompatible ways to store multiple contours in one list — for instance, the built-in contourc function differs from the closed-polygon format of the Mapping Toolbox. SG-Lib adopts the latter as its CPL/CVL format: individual contours are separated by a [NaN NaN] row, and a contour's last point may or may not repeat its first point (this second convention is not always enforced consistently across SG-Lib). Example: CPLofPL([0 0; 10 0; 10 10; NaN NaN; 5 5; 7 5; 7 7]) builds a two-contour CPL from a flat point list. Because this is the format used throughout the library for anything with multiple loops, holes or islands, this section carries many conversion, sorting and cleanup functions.

Construction and conversion turn other representations into CPL/CVL: CPLofPL appends the closing point where required, CPLofcontourc / CVLofcontourc convert from MATLAB's contourc output, CPLofimage / CPLcontourc work directly from image contours, CPLoftext / CPLtextimage build contours from text strings, and PLofCPL / CILofCVL / VLofCVL convert back to plain lists or index lists.

Ordering, orientation and sorting functions fix winding direction and grouping: CPLisccw / CPLisccwcorrected / CPLisccwinout check or correct clockwise/counter-clockwise orientation, CPLinvert reverses direction, CPLorder / CPLsortC / CPLcircshift normalize starting point and order, and CPLsortinout separates inner from outer contours.

Spatial relations between contours answer inside/outside and crossing questions: CPLinsideCPL, iscrossingCPL, CPLdisttopoint, matrixdistanceofCPL, and CPLoutercontour extract the relevant subsets or distances.

Editing and pattern functions reshape or replicate contours: CPLgrow offsets a contour, CPLradialEdges rounds corners, CPLremnoise smooths, CPLaddauxpoints / CPLinsertcollinearpoints / CVLinsertcollinearpoints insert helper points, and CPLcopypattern / CPLcopypatternPL / CPLfillPattern / CPLfillHoneycomb tile a contour with copies. Utility helpers separateNaN, woNaN, selectNaN, replaceNaN and cellofNaN operate directly on the NaN-separated structure and are worth knowing before writing custom CPL code.

1.5 Edge List Manipulation (PL/VL) ›

Edge lists (EL), facet lists (FL, Section 1.6) and tetrahedron lists (TL, Section 1.7) all describe how a set of points is partitioned into boundary edges, boundary surfaces or boundary volumes — typically via Delaunay triangulation. In planar point lists, edges and their winding direction separate inside from outside; the enclosed contour can then be filled with triangles, and the boundary edges that constrain this filling are therefore called constraints. In spatial vertex lists, surfaces play the same role for separating inside from outside of a volume, and the interfaces that constrain tetrahedralization are likewise called constraints.

This section provides functions to convert CPL/CVL contours into an EL, to sort and separate an EL, and to find open ends, open edges or open surfaces. Because an EL can in principle be unsorted, its edges occasionally need to be grouped into index lists that record which consecutive points belong to which closed contour or mesh; since SG-Lib 4.2 this grouping is built on MATLAB's graph classes (see graphofEL).

Construction and conversion build an EL from other structures: ELofCVL / CELofPL derive edges from a (closed) vertex or point list, ELofFL / ELuniqueofFL extract edges from a facet list, ELofCIL / CILofOEL convert between edge lists and contour index lists, and ELofn builds the edge list for a single closed, sorted contour.

Sorting, direction and grouping functions turn a raw edge soup into usable contours: ELsort / ELsortFE / ELseparate / ELorder sort and separate an EL into individual contours, ELflip / ELswap / ELcorrect / ELdirectVLEL fix edge direction, ELconnectedgroups / ELofELmesh / treeNodesofEL / graphofEL group edges into connected meshes, and ELreconnect repairs a broken sorted list.

Query and cleanup functions inspect or trim an EL: isinsideEL / usedinEL / ELhistogram report how often edges or nodes are used, ELunique / ELremovedoubledconstraints remove duplicates, and ELomitVLEL / ELomitedgesVLEL / ELremove drop unwanted edges. Rule of thumb: build an EL from a CPL/CVL first, then use the grouping functions rather than re-deriving contours from raw coordinates.

1.6 Facet List Manipulation (PL/VL) ›

A facet list (FL) defines triangular surfaces by referencing vertex indices of a VL: FL = [vi1 vi2 vi3; vj1 vj2 vj3; vk1 vk2 vk3; ...]. The order of the three indices within a row simultaneously fixes the direction of the triangle's normal vector, so consistent winding matters for correct rendering and boolean operations. A planar point set can be tessellated in 2D as a PLFL using delaunayTriangulation; a spatial point set can be tessellated in 3D as a VLFL the same way. Note that MATLAB's delaunayTriangulation can fail when a planar surface is decomposed while embedded in 3D — in that case, transform the surface into a plane with TofVL first, tessellate there, and the result remains valid once transformed back.

Construction from other structures builds an FL where none existed: FLofVLEL / FLofVLEL2 / FLofVLELn tessellate a vertex list against a planar edge list, FLofEL converts an edge list into a facet list, FLofDelaunay / FLofVLdelaunay wrap MATLAB's Delaunay triangulation, FLofTetrahedron / FLofTL extract the four triangles of each tetrahedron, and FLcoverVLEL / FLcoatVLEL / FLclosecontoursVLCL / FLwallofCPL close contours with covers or walls.

Ordering and normalization functions fix winding and grouping: sortFL / FLorder / FLshift reorder and circularly shift rows, FLseparate splits an FL into individual solids, and FLunique / FLremdoublevertex remove duplicate or degenerate rows.

Query and selection functions extract subsets by geometric or topological criteria: FLselect returns all facets touching one vertex, FLofSGz / FLofSGcolor select facets by z-plane or color, opponentFL / FLofoppositefacesofVLFL find opposing facets, and FLofVLFLvi / FLofVLFLfi / FLofVLFLnearestneighborVL select facets relative to a known vertex or facet subset. Rule of thumb: prefer the VLFL-level functions of Section 1.8 whenever both the vertex and facet lists need to change together — the FL-only functions here assume the underlying VL stays fixed.

1.7 Tetrahedron List Manipulation ›

A tetrahedron list (TL) row [v1 v2 v3 v4] holds the four vertex indices of one tetrahedron; its four triangular faces are given by FL = [2 3 4; 2 4 1; 4 3 1; 2 1 3] applied to that row. MATLAB's DelaunayTriangulation, the alphaShape function, and the PDE Toolbox can all generate a tetrahedron model from a vertex/point cloud (VL); however, only the PDE Toolbox can mesh an existing surface model directly into a tetrahedron model that respects that surface. Note: the abbreviation TL is today used more often for frame (transformation) lists; its tetrahedron-list meaning shown in this chapter is largely historical and rarely used now.

Construction functions build a TL from vertices or facets: THLofVL / TRofCPLz generate a tetrahedron representation from a vertex list or a z-sliced CPL, TLofFL extrudes a planar facet list into a 2.5D tetrahedron

list, and SGofVLalphaShape wraps the alphaShape-based route.

Ordering and cleanup functions normalize a TL/TR for further use: TRorder / TLseparate group tetrahedra into individual solids, TRunique removes duplicate points, and TRcat concatenates tetrahedron solids without checks; TRselect reduces a triangulation to a chosen subset.

Surface extraction functions recover the boundary shell from a volumetric mesh: VLFLofTR / FLfreeboundaryTR4FL / TR3ofSG / TR3ofTR4 return the free-boundary or surface triangulation of a tetrahedron model, FLofTL returns all facets of a TL, and TLFLneighbors / TLFLneighbors2 find neighboring tetrahedra across shared facets. VLELofTR returns the edge constraints of a triangulation, and TRplot / VLTLplot visualize a TL either as one solid or as separated tetrahedra.

1.8 VLFL Manipulation ›

Edge lists, facet lists and tetrahedron lists all reference indices into a vertex list, so whenever the VL changes — points added, removed or renumbered — the corresponding EL/FL/TL must be kept in sync. This section provides that glue: functions to combine or shorten VL/FL pairs together, to append newly generated points back onto an existing vertex list, and to build, repair or query the combined mesh (VLFL). With 96 functions this is the largest cluster in the chapter, spanning generation, editing and repair.

Generation of common shapes produces ready-made VL/FL pairs: VLFLcylinder, VLFLcone, VLFLring, VLFLdome, VLFLtetra, VLFLbar2P, VLFLspring, VLFLthread / VLFLthreadnut, VLFLgear, and text/character solids VLFLchar / VLFLtext / VLFLcreateLogo.

Closing and covering functions turn open contours into closed surfaces: VLFLclose / VLFLcloseC / VLFLclose3DC close a single contour, FLcoverVLEL / FLcoatVLEL / VLFLcoatPCVL / VLFLconnect2VLEL cover or coat contours between planes, and VLFLclosure builds a surface a solid can be pushed through along z.

Combination and concatenation join separate solids: VLFLcat / VLFLcat2 append and optionally deduplicate vertices, SGof2facets builds a minimal two-face surface, and CLODofVLFL prepares a mesh for level-of-detail generation.

Repair, cleanup and quality functions keep a mesh consistent: VLFLfusedoubledvertices merges near-duplicate vertices, VLFLshort / VLFLshort2 / VLFLshortopti remove unused vertices, VLFLvertexongrid snaps vertices to a grid, VLFLcorrectopenfaces / VLFLbridge2boundaryEdges close accidental gaps, and VLFLerrordoubledvertex / VLFLerroropenfaces deliberately inject errors for testing detection routines.

Diagnostics and queries locate problems or boundaries: VLFLfreeboundary / VLFLfreeboundaryfacets find the open edge loop of a surface, VLFLshowopenfaces / VLFLclassifyopenfacets report non-manifold or open regions, VLof2VLnonmanifold finds non-manifold vertices, and VLFLfaceNormal / VLFLvertexNormal compute normals. Rule of thumb: after any manual edit to VL or FL, run VLFLfusedoubledvertices and VLFLshowopenfaces before trusting the mesh downstream.

1.9 SG Manipulation ›

In its simplest form, an SG is a struct with just two fields, VL and FL — a vertex list and its facet list. Optional fields extend it: SG.col stores facet colors, SG.Tname names attached frames, T holds a cell list of [4x4] HT matrices, TFiL lists the facets that define each T, and TFoL lists facets attached to each TFiL. An SG can also be a cell list of several SGs, SG = {SGA, SGB, SGC, ...}, which is why some functions in this section are written recursively while others connect a cell list into a single frame.

Assembly and containers combine solids or build cell-based collections: SGcat2 concatenates two or more solids or solid sets, SGCaddSG / SGCaddSGn / SGCaddVLFL add a solid (or n copies of one) to a container, and SGshort produces one merged vertex/facet list from a nested SG.

Query and measurement functions extract information without changing the solid: SGsize returns the min/max extent in x, y, z, and VLofSG returns the vertex list of a (possibly nested) SG.

Housekeeping functions keep the struct tidy: SGstripfields removes empty/unused fields, and SGorder reorders triangle facets so they group by individual solid. Rule of thumb: reach for the nested/cell-list form of SG only when parts genuinely need independent transforms (via T/TFiL/TFoL) — a flat VL/FL struct is sufficient for a single rigid body.

2 Analytical Geometry ›

Analytical geometry is the mathematical foundation underneath the whole SG-Lib construction chain: before a solid can be generated, modified, cut or checked, the underlying points, lines, planes, transformations and curves must be handled correctly. This chapter collects the low-level geometric toolbox that the higher-level SG/VLFL functions build on, plus the plotting infrastructure used to inspect results. Section 2.1 (Graphical User Interface) covers the modified figure, interactive selection and stand-alone app tooling. Section 2.2 (Geometry Drawings) plots points, vectors, frames, contours and solids in 2D/3D. Section 2.3 (Plot Annotation) labels vertices, facets, contours and coordinates in a figure. Section 2.4 (Crossing Points and Tangents) computes intersections, tangents and bounding boxes for lines, circles and surfaces. Section 2.5 (Planar and Spatial Transformation) creates and combines homogeneous transformation matrices (HT) in 2D and 3D. Section 2.6 (Curves & Splines) generates parametric curves such as Bezier curves, helices and evolvents. Section 2.7 (Normals & Vectors) computes and orients normal vectors for edges, facets and point clouds. Section 2.8 (Distances & Metrics) measures distances, angles, centers and volumes. Section 2.9 (Convex Hull & Registration) computes convex hulls and registers point sets by their hull.

2.1 Graphical User Interface ›

The SG-Lib extends MATLAB's figure with mouse-click 3D picking, custom menus and toolbars so that interactive user interfaces can be built and later compiled into a stand-alone EXE or App. Figure setup & windows create and manage plot windows (SGfigure, VLFLfigure, VLFLviewer, SGui, VLFLui, fullview, figsize, togglefig). Interactive picking & selection pick points, facets and solids by mouse (select3d, selectV3d, FLselect3Dginput, pofselect3D, fiofselect3d, VLui, ginputTL, ginput2Dcircle). Camera & view control set and store camera position, target and light (camset, selectCameraPosition, tcamera, camlightTL, sview/rview, rotateazimut). Buttons, menus & annotations add interactive UI elements and help text (SGfiguretopbuttons, SGfigureleftbuttons, plotannotation, SGhelptext, uipolygoncontextmenu). Image & deployment tools handle screenshots, image warping and app packaging (snapplot, screencapture, imwarpT, deployTL). A few experimental natural-language front ends (jarvis, jarvisgrammar, SGtextparseNL) build on this infrastructure to steer designs by spoken or typed commands. Rule of thumb: build the interface out of SGfigure/VLFLviewer plus the select*/pofselect3D pickers, and use deployTL only once the interface is meant to ship as a stand-alone tool.

2.2 Geometry Drawings ›

These functions draw points, vectors, contours, coordinate frames, surfaces and complete solids in 2D and 3D, so that the results of geometric procedures can be inspected comprehensibly on screen. Basic primitives plot points, lines and circle segments (pplot, lplot, slplot, PLplot, cplot). Solid & mesh plots render vertex/facet lists and complete solids, including multi-view layouts (VLFLplot, VLFLplots, SGplot, SGplot4, VLFLplotfacets). Contours & edges plot closed/open polygon and edge lists (CVLplot, VLELplot, plotCVL, FEplot, FSplot). Frames & motion visualize coordinate systems, transformations and vertex-list motion (plotT, tlplot, plotTP, VLplotmotion, VLFLfaceNormalplot). View, light & color management handle camera, lighting and color mapping (tcamera, VLFLcamlight, SGshadelight, rgb2char, char2rgb, VLcol). Rule of thumb: use VLFLplot/SGplot for solids, CVLplot/VLELplot for 2D contours, and plotT/tlplot to check frames and axes.

2.3 Plot Annotation ›

This small group labels geometric data directly in the plot: point lists, edge lists, facet lists and vertex lists as well as coordinates and feature surfaces. Point & vertex labels write descriptors next to individual points or vertices (textP, textVL). Frame labels annotate a coordinate system given as an HT matrix (textT). Contour & mesh labels tag whole contours, facets and feature surfaces (textCVL, textVLFL, textVLFS). Rule of thumb: pick the text* function that matches the data type you already plotted (P/T/VL/CVL/VLFL/VLFS) to keep labels consistent with the underlying geometry.

2.4 Crossing points and tangents ›

These functions compute intersections, tangents, circle segments and containment tests between lines, circles, planes, triangles and bounding boxes in 2D and 3D. Line & circle crossings find intersection points of lines, edges and circles (crossL, cross2L, cross2edges, cross2circ, crosscircline, cross4P). Tangents & arcs construct tangent points, tangent lines and circle segments (tangente, tangentRP, tangent2circ, circof2tangent, PLcircle). Plane & triangle intersections handle 3D planes and ray/triangle tests (crossT, cross2T, Tcross2T, intersectstriangle, intersectstriangle46). Containment & interior tests check whether points lie inside contours, triangles, surfaces or bounding boxes (SGisInterior, insidePLELdelaunay, isintriangle2, isonVLFLsurface, SGisInteriorofBB). Bounding boxes & angle utilities compute, grow and combine bounding boxes and normalize angle intervals (BBofBB, BBgrow, BBconcat, angleshift, diffanglew). Rule of thumb: for 2D line/circle geometry start with cross2L/cross2circ/tangentRP, for solids use the SGis*/BBof* family for containment and bounding-box logic.

2.5 Planar and Spatial Transformation ›

This group creates and manipulates homogeneous transformation (HT) matrices in 2D ([3x3]) and 3D ([4x4]), the common language SG-Lib uses to describe position and orientation of points, frames, vertex lists and solids. Frame construction build an HT matrix from a point, an ez-vector, Euler angles or three points (TofP, TofPez, TofPEul, ezof3P, Rofa, rotdeg). Applying transformations move, rotate or mirror point/vertex lists and solids (VLtrans, VLtransR, VLtransT, PLtrans, PLtransR, VLswapX/Y/Z). Relative & composite frames combine or express one frame relative to another (TofT, TtransT, RPofTrelT, Ttransrel, TofTrel). Axis swaps & mirroring reorient a VL by swapping axes or mirroring at a line/plane (VLswapXY, VLswapYZ, mirroringatline, mirroringatline3D). Barycentric & projection utilities convert between Cartesian and barycentric coordinates or project onto planes (carttobaryVL, barytocartUVL, VLplaneprojection, planeparamofT). Rule of thumb: build a target frame with TofP*/rotdeg, then move geometry into it with VLtransT/PLtransT rather than composing raw matrices by hand.

2.6 Curves & Splines ›

This chapter collects parametric curve generators that produce an ordered point/vertex list (PL/VL) describing a path, as opposed to an unordered point cloud (PC). Bezier & radius curves build smooth curves through or near control points (VLBezier4P, VLBezierC, VLRadius4P, RLcurvature). Helices & winding generate 3D helical paths, e.g. for springs or wound wires (VLhelix, VLofPLcylwind). Special/periodic curves cover evolvents and periodic test curves (PLevolvente, zigzag, circ2poly). Rule of thumb: pick VLBezierC when you need a curve through a general set of control points, VLhelix for constant-pitch helical paths, and PLevolvente for gear-style evolvent contours.

2.7 Normals & Vectors ›

These functions compute, orient and visualize normal vectors for edges, facets and point clouds, and derive orthogonal directions and planes from them — vector orientation is order-dependent, so an ordered contour (PL/VL) yields consistent edge normals while an unordered point cloud (PC) needs neighborhood-based estimation. Edge & contour normals compute normals along a contour or its boundary edges (NLcontourVL, edgeNormal, PLnorm). Facet & vertex normals derive normals for facets and interpolate them to vertices (PLFLfaceNormal, vertexNormalbyFN). Orientation & conformance make a set of normals consistently outward/inward facing (NLconformance). Point-cloud normals & frames build local frames from point-cloud normals (TplanesofPCnormals, TplanesofVLFLfacets). Vector & plane utilities compute orthogonal vectors, vector norms, signed angles and Hesse plane forms (vec2ortho, vecnormr, acos2, Tofhesseplane, hesseplaneofT, distE, distT). Rule of thumb: use vertexNormalbyFN for smooth shading normals on a mesh, NLconformance whenever normals from different sources need a common orientation, and TplanesofPCnormals when starting from an unordered PC rather than an ordered contour.

2.8 Distances & Metrics ›

This chapter gathers distance, angle, center and size measurements on points, contours, surfaces and solids. Point & pair distances measure distances between points or between two point/vertex lists, including nearest neighbors (VLdistance, VLmindist, nearestpair, VLnearestNeighborN, VLmostdistanceNeighborN). Angles measure angles of point lists, between vectors, or along a contour (PLangle, VLangle, diffangle, aplot). Centers & fits fit centers, axes and simple primitives through point sets (center3P, center4P, centerVL, ellipseofPL, SGgetCenterPoint). Bounding boxes & volume compute bounding-box size and solid volume (BBofVL, sofBB, SGvolume, SGmagnifyVL). Clustering & reduction reduce dense point sets or clusters, mostly for unordered point clouds (PC) (nearestcluster, PLconvexseg). Rule of thumb: use VLmindist/nearestpair for point-to-point distances, BBofVL/sofBB for quick bounding-box-based sizing, and nearestcluster when thinning an unordered PC rather than an ordered VL.

2.9 Convex Hull & Registration ›

This small chapter computes convex hulls of 2D point lists and uses them to register (align) two point sets onto each other. Convexity & hull compute whether a contour is convex and construct the convex hull of a point list (isconvexPL, VLGraham, VLGrahamPVL). Hull-based center & registration derive a robust center point and a registration transform from the convex hull (centerPL, TregisterConvexHull). Rule of thumb: prefer VLGrahamPVL over the older VLGraham, and use TregisterConvexHull when two point clouds (PC) need a first coarse alignment before fine registration.

3 Polygons and Contours ›

Polygons and Contours are the 2D and 3D point-list backbone that almost every other SG-Lib chapter builds on: a construction usually starts as a flat or spatial outline before it becomes a surface or a solid. In 2D a Polygon (PL) is simply a point list [x1 y1; x2 y2; …]; in 3D the same idea is called a Vertex List (VL). If the last point equals the first, the list is closed (CPL in 2D, CVL in 3D); several closed loops can be concatenated in one list by separating them with a [NaN NaN] row, which lets one CPL/CVL variable carry outer boundaries and holes together. Choose the subsection by what you already have and what you need. Section 3.1 (Single 2D Polygons, PL) generates common open or closed 2D shapes such as circles, stars or gears. Section 3.2 (Closed Nested 2D Polygons, CPL) is the large toolbox for creating, editing, measuring and combining closed 2D contours, including polyshape-based region handling. Section 3.3 (Planar 3D Polygons, VL) and Section 3.4 (Spatial 3D Polygons, CVL) lift the same ideas into 3D for flat and freely curved contours. Section 3.5 (General PL functions) bundles cross-cutting point-list modification, crossing-point and Boolean operations, and comparison. Sections 3.6 (CPL to Surfaces) and 3.7 (CVL to Solid Geometries) are the bridge onward: they turn a finished contour into a meshed surface (VLFL) or a full solid-geometry structure (SG), continuing the construction chain from contour towards Generation (chapter 5).

3.1 Single 2D Polygons (PL) ›

Single 2D Polygons (PL) collect generators for the plain 2D point lists that feed almost every other chapter: a PL is a list PL = [x1 y1; x2 y2; x3 y3; …], which need not be closed — if the first and last point coincide it is called a CPL instead. Basic shapes create simple outlines (PLsquare for a rectangle or trapeze, PLcircseg for a circular segment, PLrand for random points). Decorative and organic contours produce recognizable silhouettes (PLstar, PLkidney, PLcircleoval). Curves and paths generate smoothly varying point sequences (PLspiral, PLsinusofPL, PLtangentcirc2 for a tangential circular-arc fillet). Standardized shapes follow engineering norms (PLgearDIN builds a DIN gear contour from 3 to 48 teeth). Rule of thumb: use this section when you need a ready-made 2D point list from a formula or standard, then hand it on to Section 3.2 (CPL) once it is closed, or to Section 3.5 for further processing.

3.2 Closed Nested 2D Polygons (CPL) ›

Closed Nested 2D Polygons (CPL) is the largest toolbox in this chapter: a CPL is one or more closed 2D contours PL=[x1 y1; …; x1 y1] concatenated in a single [n x 2] list, with separate loops divided by a [NaN NaN] row so that outer boundaries and holes can travel together as one variable. Creation & extraction build CPLs from other data (CPLofPLalphashape, CPLofprocessedimage, CPLofdrawing, CPLofSGhull, CPLofSGprojection). Editing & simplification reshape or clean up a contour (CPLbufferEdges, CPLsimplify, CPLremovearea, CPLremstraightAmin, CPLfft/CPLremFFT, CPLinsertpoint). Boolean-adjacent region tools work through the polyshape/CPS structures (PSregions, PSanalyze, CPLregionsholes, CPLouterboundaries, CPSremovearea). Orientation & topology enforce consistent winding and enclosure (CPLsetallcw, CPL2ccw/CPL2cw, CPLisccwcorrectedCPS, CPLselectinout, closedCPL/openedCPL). Measurement & comparison quantify contours (CPLarea, CPLdistance, CPLdistancewall, CPLisidenticalcontour, CPLopendistance). Application-specific generators build ready-made mechanical contours (CPLgripperjaw, CPLfischertechnik, CPLmotorshaft, CPLhatch, CPLstent). Visualization helpers show contours and regions for debugging (CPLplotcolors, CPLplotregions, CPLplotinout, CPLplotcwccw). Rule of thumb: nearly every CPL name starting with CPL, CPS or PS operates on this same nested-contour format; use CPLbool-family functions (see also Section 3.5.3) for set operations between two CPLs.

3.3 Planar 3D Polygons (VL) ›

Planar 3D Polygons (VL) is the 3D counterpart of Section 3.1: a Vertex List VL = [x1 y1 z1; x2 y2 z2; …] is the 3D analogue of a PL, used whenever a 2D-style contour must live at an arbitrary position and orientation in space rather than only in the xy-plane. Typical uses are contours that were generated flat in 2D (PL/CPL) and then placed onto a plane in 3D via a frame transform (see the HT/R glossary and Section 3.4 for the fully spatial case), or contours read directly from 3D geometry that happen to be planar, such as a single cross-section of a solid. This section currently has no dedicated functions of its own beyond the shared VL/CVL toolbox; for the general case of a 3D contour that is not planar, and for the operations that actually manipulate 3D contours (sweeping, slicing, sampling), see Section 3.4 (Spatial 3D Polygons, CVL). Rule of thumb: if your 3D points all satisfy one plane equation, treat them as a planar VL and use CVLplanarofCVL (Section 3.4) to confirm and cap the plane; otherwise use the general CVL functions directly.

3.4 Spatial 3D Polygons (CVL) ›

Spatial 3D Polygons (CVL) generalize the closed-contour idea of Section 3.2 into full 3D: a CVL is a closed vertex list VL=[x1 y1 z1; …; x1 y1 z1] that need not lie in a single plane, with several loops concatenated via a [NaN NaN NaN] separator exactly as CPLs are in 2D. Extraction from solids reads contours off an existing SG (CVLofSGT, CVLofSGcutT, CVLofSGcutplanes, CVLofSGfreeboundary, CVLfreeBoundaryVLFL, CVLofconvexHull). Slicing & cross-sections cut a body along a frame or rotate the cut (CVLofSGcutTrot, CVLclipT, CVLinsertT). Shape generation & deformation build or bend a spatial contour (CVLofRL from a radius list, CVLofCPL to lift a 2D CPL into 3D, CVLofCPLbendsinus, CVLofCPLbendspherical, CVLzsweep). Editing & geometry queries clean up or analyze points on a CVL (CVLaddauxpoints, CVLsortlinestraight, CVLedgeNormal, CVLedgecenter, CVLpointdistancetoedge, CVLplanarofCVL). Rule of thumb: use CVL functions whenever a contour genuinely leaves the plane (e.g. a helical or draped path); if the contour is flat, the lighter-weight CPL tools in Section 3.2 are usually sufficient, and Section 3.7 shows how a finished CVL becomes a meshed or solid geometry.

3.5 General PL functions ›

General PL functions bundles the cross-cutting point-list operations that apply across both open and closed, 2D and 3D polygon representations, rather than being specific to one contour type. Section 3.5.1 (PL modification) provides basic reshaping such as the convex hull of a point list. Section 3.5.2 (Crossing Points) computes intersections between polygon courses, typically via Delaunay triangulation, complementing the analytical-geometry intersection functions found elsewhere. Section 3.5.3 (CPL Boolean Operations) performs set operations — union, subtraction, intersection, exclusive-or — between closed 2D contours. Section 3.5.4 (PL comparison) measures similarity or correlation between two point lists. Rule of thumb: come here for a generic point-list utility that is not tied to a specific shape generator, then return to Section 3.1/3.2 for shape-specific work.

3.5.1 PL modification ›

PL modification holds basic, shape-independent reshaping operations for plain 2D point lists. Convex hull computation reduces an arbitrary point list to its outer envelope (PLconvexhull, returning the convex hull of an [n x 2] point list). Rule of thumb: use this before more specialized CPL operations when you only need the outer envelope of scattered points, not a full contour pipeline.

3.5.2 Crossing Points ›

Crossing Points handles intersections of polygon courses rather than ideal analytical curves: while classical analytical geometry computes intersections of ideal circles or lines, the functions here process real point-list polygons of the kind also used in mapping-toolbox-style workflows, almost always relying on Delaunay triangulation internally. Splitting by intersection breaks a contour or edge list at the points where it crosses itself or another set (ELsplit, PLsplitpointsofCPL, which returns the split points found via Delaunay triangulation). Rule of thumb: reach for this section when you need actual crossing points of real polygon data, and for pure ideal-geometry intersections look at the analytical-geometry chapters instead.

3.5.3 CPL Boolean Operations ›

CPL Boolean Operations perform set-theoretic combination of closed 2D contours, built on MATLAB's mapping toolbox and the polyshape class; the two workhorse functions are CPLpolybool and isccw. Region-based set operations (add, subtract, intersect, XOR) work on the polyshape representation (CPLboolpshape, CPLbool, CPLadd, CPLsubtract, CPLunion, CPLintersect, CPLxor). Legacy PL/EL-pair Boolean operations offer the same set operations on point-list/edge-list pairs (PLELboolAuB for union, PLELboolAmB for A-minus-B, PLELboolAiB for intersection, plus the 'x' and PLELboolABcheck relationship-check variants). Merging & splitting combine or divide several contours at once (CPLmeltbool, CPLsplitbool, CPLunitecwccw, CPLunitesorted). Rule of thumb: prefer the polyshape-based CPLunion/CPLsubtract/CPLintersect/CPLxor family for new code; the PLELbool* functions remain for compatibility with older edge-list-based pipelines.

3.5.4 PL Comparison ›

PL comparison quantifies how similar two plain point lists are to each other. Correlation measurement returns a numeric similarity score between two lists (PLcorrelate, returning the correlation of two single point lists). Rule of thumb: use this lightweight comparison for quick similarity checks; for full contour comparison including orientation and area see the CPL-level comparison functions in Section 3.2 (e.g. CPLisidenticalcontour).

3.6 CPL to Surfaces ›

CPL to Surfaces converts a pair of closed contours into the wall facets connecting them, forming the bridge from flat contour data (Section 3.2) towards meshed surfaces (VLFL). Wall meshing between contours builds the connecting facet band between two matching edge lists (FLcontourwallELn, returning the wall facets between two contours given by an edge-list-based correspondence). Rule of thumb: use this after two contours have been correlated (see CLofFLW/CPLevalcorrelation-style tools in Section 3.2) and you need the actual triangulated wall between them, typically as a step towards a full SG in chapter 5.

3.7 CVL to Solid Geometries ›

CVL to Solid Geometries is the final step of this chapter: it triangulates a spatial closed contour into a meshed surface, the direct precursor of a full SG. Planar triangulation covers an almost-flat CVL with a 2D-style Delaunay mesh (VLFLofCVLdelaunay2D, returning a triangulation for an almost planar CVL). Capped 3D triangulation closes a single contour with both a top and bottom tessellation to form a thin solid skin (VLFLofVLELdelaunay3D, VLFLofCVLdelaunay3D — both return the top and bottom tessellation for one closed vertex list/CVL). Rule of thumb: use the 2D variant when the contour is (nearly) planar for a lighter, more regular mesh, and the 3D-capped variants when the contour is curved in space and needs a genuine top/bottom cover; the resulting VLFL feeds directly into the SG-of-mesh functions of chapter 5 (Generation).

4 Surfaces ›

Surfaces collects everything that lives one dimension below a full solid: open meshes, triangulated patches and the feature surfaces that make sense of a mesh's geometry. Many meshes imported from CAD or scan data are not "watertight" and contain open edges, so this chapter provides the tools to build, repair, interpret and finally close such surfaces into solids. The central data type is VLFL, a vertex list plus facet list describing a mesh. Section 4.1 (2D Surfaces) triangulates planar contours into a VLFL and offers a large toolbox for cleaning, querying and extending such meshes. Section 4.2 (3D Surfaces) extracts edge and contour information from feature surfaces on a solid. Section 4.3 (Feature Surfaces) groups facets into feature surfaces by normal-vector and edge-angle criteria, which is the basis for mounting-face and frame detection. Section 4.4 (Surfaces to Solid Geometries) turns an open, edge-bounded surface into a genuine three-dimensional SG body.

4.1 2D Surfaces ›

2D Surfaces builds and manipulates planar meshes: a 2D point list (PL) together with a facet list (FL), whose boundary is described by an edge list (EL). Triangulation from contours creates the mesh in the first place, using PLELDelaunay, PLFLofCPLpoly (wrapping poly2fv from the Mapping Toolbox), PLFLofCPLdelaunay, PLFLofCPLdelaunayGrid, PLFLofPLELdelaunay, TRofCPL, TR2ofimage (from a grayscale image) and VLFLofCPL. Mesh repair and cleanup fix bad or redundant geometry: VLFLremoveIntersectFacets, VLFLremsmallfacets, VLFLsmoothboundary, VLFLremovekinks, VLFLremoveFacet, VLFLdeleteFLofvi and VLFLcircFLedges. Queries and analysis inspect a mesh without changing it, such as VLofVLFLselfIntersection, SGintersectingtriangles, VLFLoppositefaces, viVLFLnearestneighborVL, VLFLfaceneighborhood, VLFLregions, VMdistancetoSG and VLisInteriorofBBlist. Extension and refinement add detail, e.g. VLFLaddfacetcenterpoints, VLFLaddfacetgridpoints, VLFLoptimizeFLWez and VLFLfreeboundarycircshift. A final group turns a 2D surface into a solid body for specific use cases, such as SGofCPLsheetmetal, SGofCVLzdelaunayGrid, SGcutandstretch and SGfillingsurfaceT. Rule of thumb: start from a closed 2D contour (CPL) with one of the PLFLofCPL... or VLFLofCPL functions, then clean and query the resulting VLFL before feeding it into a solid-generation step.

4.2 3D Surfaces ›

3D Surfaces currently holds a single, specialized function: VLELofSGfeatureSurface, which creates a vertex list, a contour index list and a feature-surface index list from an SG's feature surfaces. It bridges the mesh-level view of a solid (VLFL) with the higher-level feature-surface concept developed in Section 4.3 (Feature Surfaces), by turning a feature surface's boundary into an explicit VL/EL contour that other parts of the library can consume. Expect this section to grow as more 3D-specific surface-extraction functions are added alongside its 2D counterpart in Section 4.1.

4.3 Feature Surfaces ›

Feature Surfaces groups the facets of a mesh into meaningful surfaces separated by feature edges, i.e. edges whose transition angle exceeds an alpha threshold; feature surfaces are always open and always bounded by feature edges. Detecting feature edges and surfaces is the entry point: TR3featureEdges, TRfeatureEdgeFacets, FLfeatureEdgeSurface, FLfeatureEdgeSurface2, VLELofSGfeatureEdge, FEofSG and FEofVLFL. Selecting and filtering feature surfaces by criteria such as position or orientation uses FSofSG, FSofSGfacet, FSofSGT, FSofSGnormal, FSnormalofSG, FSfeature and the more general SGfeature. Mounting faces, a special case of coplanar feature surfaces sharing a normal vector, are found and shown with TR3mountingfaces, MLofSG and MLplot. Frames derived from a feature surface are produced by TofSGfil and TofSGFSselect, which describe a homogeneous transformation (T/HT) using spatial terms. A final pair of functions, SGinsideSurface and FLinsideSurfaceofSG, checks whether facets face into the interior of a solid. Rule of thumb: pick out a feature surface first with FSofSG or FSfeature, then derive frames, mounting faces or edge lists from it.

4.4 Surfaces to Solid Geometries ›

Surfaces to Solid Geometries addresses a common practical problem: one or more open, edge-bounded surfaces that need to be turned from a two-dimensional description into a genuine three-dimensional SG body. Typical applications include the processing of scanned surfaces such as people, garments or other digitized objects, where the raw scan is a mesh with holes rather than a closed solid. Conceptually this section closes the loop opened in Section 4.1 (2D Surfaces) and Section 4.3 (Feature Surfaces): once a surface's boundary and feature structure are known, the remaining step is to cap or thicken it into an SG that the rest of the library (Modification, Cutting, Boolean, Queries) can operate on. Consult this section whenever a mesh is watertight in intent but not in practice.

5 Solid Geometry Modeling ›

Solid Geometry Modeling is the top-level group for designing, arranging and checking three-dimensional solid bodies (SG) in SG-Lib. It bundles the full construction red thread from a first shape to a validated design: Generation → Modification → Connection & Joints → Cutting & Slicing → Boolean → Queries/Checks. Section 5.1 (SG basic shapes) offers ready-made primitives and part generators for quick starts. Section 5.2 (SG Generation) turns 2D contours, meshes and surfaces into the first solid. Section 5.3 (Spatial Transformation) positions and orients solids absolutely or relative to each other. Section 5.4 (SG Modification) reshapes an existing solid, Section 5.7 (SG Connection & Joints) links solids and builds movable joints, Section 5.6 (SG Cutting & Slicing) separates or slices a solid, and Section 5.5 (SG Boolean Operations) combines, subtracts or intersects solids. Section 5.8 (SG Queries & Properties) inspects and validates the result, while Section 5.9 (SG Selection & Grouping) selects and organizes solids within larger assemblies.

5.1 SG basic shapes ›

SG basic shapes collects functions that create a finished solid (SG) directly in one call, without going through the Generation pipeline of Section 5.2. Simple primitives and markers cover everyday needs: SGbox, SGsphere, SGcylinder, SGcone and SGconetruncated build the classic building blocks, while SGtext, SGtextstamp, SGtextTUM and SGtextbanner turn text into 3D letters or engravings for labelling parts and coordinate systems. Mechanism and joint parts provide reusable hardware: SGbearing, SGbearingaxial, SGjointball, SGinsertrevjoint, SGdummyRjoint and the Fischertechnik/inch-quarter families (SGfischertechnikaxle, SGiqpin, SGiqknob, SGiqlink, SGiqsocket, SGiqanglerect) model axles, bearings and pluggable joints. Motor and drive shapes cover common motors and their shafts, e.g. SGmotorN20, SGmotorN40, SGmotorFeeTech40, SGmotorshaft and SGmotorSolExpert. Project- and device-specific parts (SGrobot, SGbatteriepack, SGraspicam2housing, SGiPadFlip2stand, SGboxMacmini, SGcompass and many similarly named housings/holders) are concrete, reusable MIMED lab designs rather than generic primitives. As a rule of thumb: reach for this chapter when you need a complete, named shape immediately; use Section 5.2 (SG Generation) instead when you must build a shape from your own contours or meshes.

5.2 SG Generation ›

Generation is the first step of the construction chain: it turns 2D contours, meshes and surfaces into the first solid body (SG) that later steps position, modify, join, cut and check. Choose the subsection by what input you already have. Section 5.2.1 (SG of CPL Extrusion) extrudes a single closed 2D polygon by translation, rotation or along a path into a body. Section 5.2.2 (SG of CPL Connections) lofts or maps two or more, possibly nested, contours onto each other through an outer surface. Section 5.2.3 (SG of Meshes) builds or converts solids from an existing vertex/facet mesh, patch or point cloud. Section 5.2.4 (SG from CVL, Surfaces & Floorplan) collects generators that build solids from contour stacks (CVL), surfaces, bounding boxes, floorplans and other parametric descriptions.

5.2.1 SG of CPL Extrusion ›

SG of CPL Extrusion turns a single closed 2D polygon (CPL) into a solid by translation, rotation or sweeping along a path, taking coordinate-frame transforms into account. Straight and rotational extrusion form the core: SGofCPLz builds a 2½D extrusion along z, SGofCPLrot and SGofCPLrota rotate a contour around the z-axis or an arbitrary axis, and SGofCPLT applies a general final transformation matrix. Path- and frame-guided extrusion sweeps a contour along a moving frame or path: SGofCPLtransT, SGofCPLTL and SGofCPLextrudealongCPL extrude along a list of frames or a second guiding contour. Edge and profile shaping refines the extruded body: SGofCPLzchamfer, SGofCPLzradialedge and SGchamfer2 chamfer or round the extrusion edges, while SGofCPLztwist, SGofCPLzgrow and SGofCPLbendonadrum twist, taper or roll the contour as it is extruded. Segmented and grid-based variants handle multi-region or gridded input: SGofCPLzseparated creates one solid per closed contour, SGofCPLzfillgaphull merges several sub-regions, and SGofCPLzdelaunayGrid / SGofCPLzgrid extrude using grid resolutions or z-coordinate lists. As a rule of thumb: use this chapter whenever exactly one contour needs to become a body; use Section 5.2.2 (SG of CPL Connections) once two or more contours must be connected.

5.2.2 SG of CPL Connections ›

SG of CPL Connections handles the case where two, potentially nested, contours must be connected through an outer surface rather than simply extruded. The underlying tasks are to morph different shapes into each other, to assign inner and outer contours, and to map several contours to one or many others. Two-contour connections form the basic case: SGof2CPLz, SGof2CPLsz and SGof2CPLznew build a solid between two closed polygons and a height, while SGof2T and SGof2SGT connect two coordinate frames or two solids-with-frames directly. Vessel- and branch-style connections handle organic or tree-like geometry: SGof2CPLzcorrelate, SGof2CPLTLcorrelate and SGof2CPLzbranch correlate matching contours or create 1:n / n:1 branching solids. Surface and mesh variants close two contours or surfaces via SGof2CVL, SGof2Dfunc (from a 1D/2D function) and SGof2CPLzfillgaphull (filling gaps in sub-region-free extrusions). SGsimplehinge360RR and SGof2CPLtransition4FDM address special manufacturing needs: an interlocking 360° rotary hinge and a support-free FDM transition, respectively. Rule of thumb: reach for this chapter as soon as more than one contour is involved and a smooth transitional surface — not a straight extrusion — is required.

5.2.3 SG of Meshes ›

SG of Meshes builds or converts a solid from an already existing vertex/facet mesh rather than from a 2D contour. Direct mesh-to-solid conversion is the core task: SGofVLFL wraps a vertex list (VL) and facet list (FL) into a proper SG, SGofpatch converts a graphics patch object, and SGofgo / SGofgca pull the surface data straight out of a graphics object or the current figure. Convex construction generates a solid envelope from unordered points: SGofVLdelaunay computes a convex Delaunay triangulation and SGofVLtrans builds a convex solid from a transformed vertex list. Structural and container conversions bridge between representations: SGofSG converts a cell list of solids into one struct, SGofVLFLSGc extracts only the solid geometry from mixed structs/solids/vertex data, and SGofFS returns the solids attached to a set of feature surfaces. Specialized fitting functions round out the chapter: SGofSGCVLR separates an extruded two-plane surface and replaces the straight sections, SGofSGTsurface returns the flange surface for a named frame, and SGofVLFLz derives a simplified fitting solid from a surface.

5.2.4 SG from CVL, Surfaces & Floorplan ›

SG from CVL, Surfaces & Floorplan gathers generators that build a solid from contour stacks (CVL), surfaces, bounding boxes, floorplans and other parametric descriptions, filling the gaps left by plain extrusion or mesh conversion. Radius/z and rotational-surface generators build axially symmetric parts: VLFLrotsymRLn, VLFLrotsymRLCL and VLFLofCPLrot construct solids from radius-height lists, optionally combined with a contour, while VLFLshaft, VLFLbolt and VLFLhollowsnapaxle model shafts, bolts and slotted snap axles. Bounding-box and surface fitting derive solids that fit around or under existing geometry: SGofBB / VLFLofBB / SGBB build bounding-box solids, SGofSurface and SGplateunder fit a body to or under a surface, and the interactive SGofBBui builds a box from the current camera view. Slice-stack and floorplan reconstruction turn layered or 2D planning data into 3D bodies: SGofCVLz and SGofCVLzconvexhullfillgap reconstruct a solid from a CVLz stack, and SGofFloorplan builds a body from an architectural floor plan. Gears, boards and technical parts round out the set: SGspurgear and VLFLinternalgear generate involute gear teeth, SGboard returns a raster board, and SGdrawingtemplateofCPL / SGmodelSliderhole create printable drawing aids and key/keyhole geometry. External-data import completes the chapter: SGofGeoTIFF builds a solid from a depth image and ms3Dmodel2SG converts a Microsoft 3D model file.

5.3 Spatial Transformation ›

Spatial Transformation moves solids in space — absolutely, relative to each other, or relative to coordinate frames and feature surfaces — without changing their shape, sitting between Generation and the later Modification, Connection and Boolean steps of the construction chain. Absolute and origin placement move a solid to a fixed location: SGtrans0 and SGtrans1 place a solid at the coordinate origin or into quadrant 1, and SGtrans offers multiple transformation modes in one call. Translation and rotation primitives apply a single motion: SGtransP translates using a vector, SGtransR rotates using a rotation matrix, and SGrotate / SGTrotate rotate around the solid's own center or around a named frame. General and relative transforms combine motions or reference another solid: SGtransT and SGtransrelT apply a full homogeneous transformation matrix (directly or relative to a known solid), and SGtransrelSG repositions one solid relative to another by name. Arrangement and multiplication helpers place many solids at once: SGarrangeSGC arranges all parts of an SG container, SGstackn stacks solids, and SGcopyTL copies a solid across a cell list of frames. As a rule of thumb: use this chapter to place or orient a solid; use Section 5.7 (SG Connection & Joints) once solids must be structurally linked rather than merely positioned.

5.4 SG Modification ›

SG Modification is the second step of the construction chain: it reshapes an existing solid without changing its basic identity, preparing it for later connection, cutting or Boolean steps. Edges & corners round, chamfer or break sharp features (SGchamfer, SGbreakCorners, SGradialCorners, VLbreakEdges, VLradialEdges). Size & fit change overall dimensions to meet a target (SGgrow, SGscale2dim, SGshrinktofit, SGstretching, SGextend, SGstrechez). Walls, cavities & impressions create hollow sections, drilled holes, imprints or engravings (SGhollowsolid, SGTdrill, SGimpress, SGnegativeform, SGremimprint, SGdemoulding). Deformation & sweeping warp or sweep a body along a path or surface (SGbending, SGbendcone, SGshearing, SGsweepTproj, SGpunchTCPLthrough). Mesh housekeeping & duplication tidy vertices/facets or produce patterned copies (SGvertexongrid, SGfusedoubledvertices, SGflip, SGreduceVLFL, SGcopyVL, SGcopypatternXYZ). As a rule of thumb: use this chapter once a solid already exists and needs a shape change, then move on to Section 5.7 (SG Connection & Joints) or Section 5.6 (SG Cutting & Slicing) for the next step.

5.5 SG Boolean Operations ›

SG Boolean Operations combine two or more solids into one, forming a late step of the construction chain that typically follows Modification and Connection & Joints and precedes final Cutting or Queries/Checks. Core operators provide the four classic Boolean primitives: SGunion, SGsubtract, SGintersect and SGxor add, remove, keep-common or symmetric-difference two solids, all built on the more general SGbool3/SGbool4/SGbool5 engines that solve the operation with increasing robustness. Heuristic and analysis variants help when clean Boolean results are hard to reach: SGboolh applies a heuristic Boolean, SGboolanalyze and SGanalyzePenetration visualize or classify how two solids cross, and SGbooltest / SGboolDEBUG support testing and debugging problem geometries. Addition and replacement without full re-cutting cover common assembly patterns: SGaddrelSG arranges and combines a solid relative to another, SGsupplement adds only the parts of B outside A, and SGreplace / SGunionreplace / SGunioncomplete replace or complete a region of a solid. Supporting utilities round out the chapter: crossingfacets2VLFL finds crossing facets, SGsurfacemeltbool merges added surfaces, SGconcat performs a simple non-Boolean concatenation, and cross2Fplanar targets planar Boolean cases. Rule of thumb: reach for this chapter once two independently generated solids must become a single, watertight body.

5.6 SG Cutting & Slicing ›

SG Cutting & Slicing separates a solid into parts or extracts cross-sectional contours from it, typically applied after Connection & Joints and before or alongside the final Boolean and Queries/Checks steps of the construction chain. Section 5.6.1 (SG Cutting) divides a solid into pieces along a plane, box or blade, for example to make a part fit a build volume or to insert a joint. Section 5.6.2 (SG Slicing) instead extracts 2D cross-sectional contours through a solid, as needed for dimensioning, technical drawings, cutting patterns or layered-plate construction.

5.6.1 SG Cutting ›

SG Cutting divides a solid into two or more parts at one or several planes, boxes or blades — for instance to fit a print volume, remove material, or prepare a joint. Planar cuts split a solid at a defined level: SGcut cuts at a z-plane, SGcutT and SGcut2T cut below an arbitrary plane or between two frames, and SGcutend cuts relative to the bounding box. Box and recursive cuts handle bulk material removal: SGcutBB removes a bounding-box region and SGcutrecursivly repeatedly cuts a part into pieces until each fits a given maximum size, useful for splitting large prints (SGpuzzlecut3D) into printable segments. Vertex/facet-precise cutting inserts new geometry at the cut: SGcut2 cuts by inserting vertices and facets directly rather than through a Boolean pass. Joint-preparing cuts combine cutting with connector geometry: SGcutTcrossblade removes a cross-blade-shaped part, SGcutTinsertpins cuts and inserts connection pins in the same step, and SGcutTcylindricblade uses a cylindrical blade to create a joint surface. Rule of thumb: use SG Cutting to divide a body; use Section 5.6.2 (SG Slicing) instead when only a 2D contour of the cross-section is needed.

5.6.2 SG Slicing ›

SG Slicing creates cross-sectional contours of solid bodies, as needed for dimensioning and technical drawings, cutting patterns, or the construction of bodies from layered plates. Facet- and single-plane slicing extracts the raw geometry at a cut: VLFLofrangez and FILofrangez return a reduced facet/vertex list or a facet index for facets crossing a plane, and SGslicer computes the Delaunay triangulation of the sliced plane. Single- and multi-slice contour extraction returns usable 2D contours: CPLsingleSolidSlicer slices a single closed surface, CPLofSGslice / CPLofSGslice2 / CPLofSGslice3 return the CPL of a sliced SG (including separated or multi-directional variants), and CPLofSGsliceauto / CVLzofSGsliceauto pick slice heights automatically based on significant z-distance. Layered CVL-stack construction and repair builds and cleans up a full stack of z-sorted contours: CVLzofSG and CVLzofSGslices build the stack, CVLzsimplifyAmin, CVLzaddauxpoints, CVLzremovearea and CVLzisccwcorrected simplify, add points to, or correct orientation and gaps in the contours. Stack editing and gap-filling manage individual layers: CVLzseparate, CVLzreplace, CVLzremoveslices, CVLzremoveuncorrelatedCPL, CVLzcut and CVLzflipud select, replace, remove or flip specific slices, while CVLzconvexhullfillgap and CVLzoutlinecontourfillgap fill each layer with a single convex or outline contour. Rule of thumb: use this chapter whenever a solid must be turned into layered 2D data rather than another solid.

5.7 SG Connection & Joints ›

SG Connection & Joints links two or more solids together structurally or via movable joints, forming the step of the construction chain that typically follows Modification and precedes Cutting & Slicing or Boolean union. Straight and profiled links join two positions or solids: SGlinkstraight builds a straight link between two frames TA and TB, SGlink and SGconnect build structural or profiled-rod links between solid centers, and SGlinkwall attaches a plain wall to a solid. Curved and frame-based connections follow a smoother or transformed path: SGconnect2T and SGconnect2FS build Bezier-contour connections between two solids or feature surfaces, and SGspherelink / SG2SGspherical create spherical links or project a solid onto a sphere. Joints and inserted connectors build functional, often movable, interfaces: SGinsertJoint inserts joint structures inside a solid, SGinsertPeghole and SGinsertCut insert peg/hole connectors or a cut, SGbearing2012 returns a bearing for a joint, and SGfindjointDoFofSG tries to determine the degrees of freedom between two solids. Blade- and tube-based connectors provide specialized hardware: SGbladeofPL / SGbladeofCVL create cutting-blade-shaped solids and SGtube / SGplugofSGT create tube or plug connectors, while SGboreholemountingpost and SGpunchTCPLfirstwall build mounting posts and punched first-wall holes. Rule of thumb: use this chapter once separately generated solids must become one connected or articulated assembly.

5.8 SG Queries & Properties ›

SG Queries & Properties inspects, measures and validates solids, forming the checking step at the end of the construction chain after Generation, Modification, Connection, Cutting and Boolean operations. Type and validity checks confirm a solid is well-formed: isSG tests whether an object is a solid geometry, SGisempty and SGisconvex check for empty or convex solids, SGisidentical compares two solids for equality, and isplanarVLFL / isonVLFLedge check planarity or edge membership. Geometric measurements quantify size and shape: SGarea / VLFLarea return surface area, SGweight estimates weight, BBofSG returns the bounding box, SGfacenum / nofVLFL count facets, and VLFLfaceAngles / TR3neighborsAngle return facet or neighbor angles. Collision, containment and crossing analysis relate two or more solids to each other: VLcrossingSG, SGintersectFacetPoints and crossingfacets2VLFL find crossing points or facets, VLFLinpolyhedron tests whether points lie inside a solid, and SGanalyzePenetration / SGanalyzeGroupParts classify or color-group penetrating parts of an additive design. Hull, surface and coloring utilities support both analysis and visualization: SGconvexHull / SGconvexhullVL / SGconvexHullXYZ / SGconvexHullsliced build various convex hulls, SGuniquesurfaces and SGnearestpointsofSG extract unique surfaces or nearest points, and SGcolorfaces / SGcolorseparation / SGcolorcopy color or separate a solid by its facet colors. Rule of thumb: reach for this chapter to verify, measure or compare a finished solid rather than to change it.

5.9 SG Selection & Grouping ›

SG Selection & Grouping organizes solids within larger assemblies, complementing the construction chain by helping to pick out individual parts and manage collections of many solids at once. Section 5.9.1 (Selection) isolates a specific solid, surface or point from a cell list, a struct, or the current figure — for example by a click, a containing frame, or a small command language. Section 5.9.2 (Grouping & Collections) instead operates on many solids together: packing, converting between cell-list and struct-array representations, sorting by collision or containment, and separating surfaces or solids by position.

5.9.1 Selection ›

Selection isolates one specific solid, surface or point out of a cell list, a struct, or the currently displayed figure, so that later steps can act on exactly the intended part. Interactive picking retrieves geometry directly from the current view: SGsurfaceselect returns an isolated surface from a solid or from the current figure (SGofgca), and SGselect3dPL finds the nearest point and vertices similar to MATLAB's select3d but for a point list. Frame- and point-based selection uses geometric criteria instead of clicking: SGselectGroupofT returns the part of a solid that contains a given point or lies within a frame's region. Command-language selection filters larger sets: SGselect selects solids from a cell list using a small command language, useful when many parts must be filtered by name or property in one call. Rule of thumb: use this chapter to pick one part before editing it; use Section 5.9.2 (Grouping & Collections) to manage many parts together.

5.9.2 Grouping & Collections ›

Grouping & Collections manages many solids together as packed layouts, converted containers, or sorted/separated sets, once individual selection (Section 5.9.1) is no longer enough. Packing and plating arrange multiple solids for output: SGCpack packs several solids side by side and SGplatesofSGML converts a solid into a plated arrangement. Container conversion switches between representations: SGcell2SG converts a cell list into one solid geometry and SGcell2array converts a cell list into a struct array of solids. Surface separation and sorting split a solid or cell list by geometric criteria: SGsurfaces and SGseparatebyT return closed surfaces or separate a solid into parts below/above a frame, SGsurfacessort sorts independent surfaces, and SGseparatebyPez searches in front of and behind a given direction vector. Collision and containment sorting checks relationships across a whole cell list: SGsortcollBB, SGsortcollVLFLBB and SGsortcollVLFL return collision index lists at bounding-box or facet precision, while SGsortfitinBB and SGsortisinsideBB return fits-in / is-inside index lists. Rule of thumb: reach for this chapter whenever an operation must run on a whole set of solids rather than on one selected part.

6 Frames and Kinematics ›

Frames and Kinematics collects everything needed to place bodies in space relative to each other and to make them move. A frame is a 4x4 homogeneous-transformation matrix (HT, often called T) that fixes an origin plus an orthonormal ex/ey/ez triad; frames are frequently attached relative to feature surfaces with specific properties, a concept explained together with the frame tools here. Section 6.1 (Frame Definition) creates, names, stores and plots frames on a solid geometry (SGT). Section 6.2 (Kinematics Chains) builds and simulates kinematic chains of linked solids, including manipulator links, gear chains and animation. Section 6.3 (Forward and Inverse Kinematic) solves joint angles for simple planar arm robots. Section 6.4 (Symbolic Definitions) supports Denavit-Hartenberg and symbolic-equation based kinematics. Section 6.5 (Fourbar Linkage) offers a compact classic four-bar creation kit, while the larger Section 6.6 (Linkages and Coupler Curves) covers the full modern planar-linkage and coupler-curve workflow built on Pose and CPLR representations.

6.1 Frame Definition ›

Frame Definition provides the basic toolbox for creating, naming and managing HT frames stored inside a solid geometry's SGT structure. Frame construction builds a frame from geometric input: Tof3P returns a 4x4 HT matrix from three points, Tof2P/T3ofT2 convert simpler 2D or point-plus-axis descriptions into full 4x4 frames, and SGTui lets you pick a frame interactively on a union space. Frame storage and access on a solid keep named frames organized: SGTget and SGTset read or replace a named frame, SGT retrieves a frame from a solid's struct, and SGTremove deletes one. Feature-surface framing ties a frame to a selected face: SGTsetofFS creates a frame from a feature surface and a selector, and SGTframeselect selects a frame set. Visualization and layout round it out: SGTplot and SGTframeplot show a solid with its frames, SGTsetorigin repositions the solid so its base frame sits at a chosen origin, and SGTBB replaces the detailed surface geometry by a simplified bounding box for faster preview. As a rule of thumb, use the SGTget/SGTset pair whenever you need to read or overwrite a named frame programmatically, and the *ui functions when picking a frame by hand on-screen.

6.2 Kinematics Chains ›

Kinematics Chains is the large collection of functions for building, transforming, analyzing and animating chains of frames and linked solids, from a single joint up to full manipulators and gear trains. Chain construction and evaluation turn a sequence of frames or solids into a kinematic model: KMofSGs and KMchain build a kinematic structure from a chain of solids or frames, KMT returns an HT matrix from that model, and KMreset/KMplot reset or plot it; SGTchain returns the spatially transformed solids of a chain and SGTcalibchain recalibrates their base frames. Joint and frame analysis inspects existing geometry: SGanalyzeJointType and SGanalyzeJointFaces identify joint type, stator, mover and joint frame, SGisinternalframe/SGissurfaceframe classify a frame, and diffT/isequalT/uniqueT compare HT matrices. Frame algebra and conversion cover the low-level matrix work: T2ofT3/T3ofT2 convert between 3x3 and 4x4 forms, Trotate/Trelation/Tof2Tcrossez build derived frames, ezofT/pofT extract the ez-vector or translation part, and TLofPL/TofPLpath/PLofT move between point lists and frame lists. Manipulator and mechanism parts provide ready-made links: SGManipulatorLink/SGManipulatorBase and the numbered SGManipulator03…07 Base/Link families build robot links, while SGgearchainwindup, SGgearchainrotate, SGgearchainstraight and CPLofSGgears assemble and lay out gear chains. Animation and plotting bring chains to life: SGTchainanimation, SGTchainmovie, tplotangle, HTplot and TLplots visualize motion, angle and angular-velocity histories of a frame list. Use the T*/TL* low-level functions when working directly with matrices, and the SGT*/KM* functions when working through a solid's stored frame structure.

6.3 Forward and Inverse Kinematic ›

Forward and Inverse Kinematic bundles a small, focused set of solvers for planar two-link arm robots. Joint readout retrieves the current state: ROBgetJointPosition returns the position of a set of joints of a robot. Inverse kinematics computes joint angles from a desired end-effector position: invkinplan2 solves the classic planar 2-arm case, and invkinrplan3 returns the angles for a rotating planar two-arm kinematic. Forward kinematics does the reverse: forwkinrplan3 computes the end-effector pose of a planar rz two-arm mechanism rotating around an axis, given the joint angles. These functions are a lightweight complement to the general frame-chain machinery in Section 6.2 for the common two-link textbook case.

6.4 Symbolic Definitions ›

Symbolic Definitions applies MATLAB's Symbolic Math to kinematic chains, so joint angles and link lengths can be carried as symbols instead of numbers. Equation and symbol handling covers the generic machinery: symsofequation returns or defines the symbols of an equation system, subsvar substitutes several symbols in an equation, symfunchng changes the parameter of a given equation, symfun2sym converts a symfun into a plain sym formula, symfunparam returns individualized strings for symbolic parameters, and solveequations returns the symbolic solutions for a given set of equations. Symbolic kinematics applies this to chains: symbforwkine computes a forward-kinematics expression from a symbolic linkage description. Denavit-Hartenberg support builds solids directly from classic DH parameters: SGTofDenavitHartenberg returns a solid with frames from DH values, and SGTofDHset builds the solids for a whole kinematic chain defined by a set of DH parameters. Use this section when a closed-form or parametric expression of the kinematics is needed rather than a purely numeric evaluation.

6.5 Fourbar Linkage ›

Fourbar Linkage is a compact creation kit for the classic planar four-bar mechanism, independent of the larger Pose/CPLR-based workflow in Section 6.6. Design and synthesis start from desired poses: SGfourBarLinkageKit is the overall creation kit for four-bar linkages, while synth4Bar2Pose and synth4Bar3Pose return the four defining points of a linkage that satisfies two or three given poses. Angle calculation and visualization use those points: calc4BarAngle returns the four-bar-linkage points for the coupler link B at a given input angle, and plot4Bar plots a four-bar linkage. This section is the quick entry point for simple four-bar problems; for coupler-curve search, Pose-based synthesis, Gruebler degree-of-freedom checks or solid four-bar geometry, see Section 6.6.

6.6 Linkages and Coupler Curves ›

Linkages and Coupler Curves is the modern (post-2017) family of planar-linkage and coupler-curve functions, built on the CPLR contour representation, the Pose struct and the classic four-bar functions of Section 6.5. It is the natural continuation of Section 6.5 for anyone who needs pose-based synthesis, coupler-curve analysis or a printable solid mechanism rather than just the four defining points of a single four-bar. Section 6.6.1 (Coupler Curves and Joint Kinematics) represents linkages as CPLR contours, inserting hinges and computing joint sweeps and joint tracks. Section 6.6.2 (Pose Definition and Manipulation) creates, transforms, samples and plots the underlying Pose/LPS data independently of any specific synthesis method. Section 6.6.3 (Four-bar Synthesis and Analysis) covers pose synthesis, coupler-curve search, Gruebler degrees of freedom and inverse kinematics of planar linkages. Section 6.6.4 (Four-bar and Hinge Solids) turns the resulting linkages into printable solid geometry with piano hinges, simple hinges and linkage plates.

6.6.1 Coupler Curves and Joint Kinematics ›

Coupler Curves and Joint Kinematics represents linkages as CPLR contours (a cell list of 2D contours plus their linking joint frames), covering hinge insertion, joint sweeps and joint tracks along a motion. Hinge insertion turns a plain contour set into a jointed, extrudable one: CPLRinserthingelist, CPLRinserthingeclosed and CPLRinserthingelinkage return CPL regions linked by frames and prepared for extrusion, while CPLRofPLEL/CPLRofPLELplain build linkages from a point list, edge list and hinge radius. Contour and graph simplification clean up redundant structure: CPLRsimplifyEL, CPLRmeltbyIndex, CPLRmeltbyCycles and EEGmeltbyCycles merge or melt contours that share edges or form closed loops, and CPLRselectEL/EELofCPLR/ELTofCPLR/ELCofELT/CPLRofELC/CPLRofELT convert between the CPLR representation and its underlying (extended) edge lists. Kinematic analysis inspects the resulting mechanism: CPLRanalyzeTkinematics analyzes the contour-region structure and hinge frames, CPLRlinkageangles and CPLRfourbarhingeangles compute static or piano-hinge design angles, and wlim4CPLRinserthingeclosed returns absolute joint angle limits for a given pose. Motion, sweeps and Boolean combination move the linkage and combine the swept regions: CPLRanimate/CPLRmove animate the contours, CPLRjointtrack and CPLRsubtractjointtrack/CPLRsubtractjointsweep compute and subtract joint-blocked regions, and CPLRunionsweep/CPLRintersectsweep/CPLRsubtractsweep/CPLRsubtractTLbs combine swept contours with Boolean-style operations. CPLRposechange and CPLRposesetbycrank adjust a linkage's pose to a target configuration or a given crank angle, and CPLRofPose/CPLRsample provide CPLR contours from a Pose or as ready-made samples.

6.6.2 Pose Definition and Manipulation ›

Pose Definition and Manipulation creates, transforms, samples and plots poses (the Pose/LPS structures), independent of any specific four-bar synthesis method. Pose creation and interactive definition build the basic struct: Poseui returns a pose struct interactively, Poseofimageui defines three poses directly in a drawing image, and TofPose/PoseofT convert between pose vectors and 2D/3D HT matrices. Transformation and normalization adjust an existing pose: Posenorm rescales pose vectors to a common length, Poseflip/Poseorderchange/PosetransT/Posemagnify flip, reorder, transform or scale a pose, and PosetransrelCPLE/PosetransrelGPL/Posetrans4basepoints shift a pose relative to contours or grid points. Attachment points and grids connect a pose to concrete mechanism geometry: TofPoseattachmentpoints, TofPoseGPL, A1ofA1B1GPLPose and PoseaddGPL derive attachment-point frames and grid points, and center3Pose/center4Pose/center3Poseeffektorgrid find rotation centers for one or several poses. KDW analysis (K, D, W = crank, distance and swing parameters) is covered by KDWofPoseattachments, KDWPosevariation and KDWPoseinsideCPL. Sampling, storage and plotting round the section out: PosesampleWalk, PosesampleCover, PosesampleHook and related PosesampleXxx functions provide ready-made pose sets, PosewriteAPD/PosereadAPD/Posesampleupdate persist them to disk, and Poseplot/Poseplotspace/PoseplotA0B0/Poseplotsolution visualize a pose or a synthesis solution. The LPS (Linkage Polyshape) subgroup — LPSofgca, LPSplot, LPSjointmatrix, LPSconnect, LPSanimate — offers an alternative polyshape-based representation of the same linkages.

6.6.3 Four-bar Synthesis and Analysis ›

Four-bar Synthesis and Analysis is the computational core of the linkage family: pose synthesis, coupler-curve search, Gruebler degrees-of-freedom checks and inverse kinematics of planar linkages. Core four-bar computation covers the general-purpose entry points: fourBarLinkage and fourBarLinkageCPL are multi-purpose functions for a four-bar linkage, CPLfourbarlinkage/CPLfourbarlinkage2/PLfourbarlinkage return the coupler/crank/swing contours or point list, and fourBarLinkageplot/fourBarLinkageplotanim/fourBarLinkageplotanimui plot and animate the mechanism. Pose-based synthesis solves for a linkage that reaches 1, 2, 3 or N given poses: checkfourbar1Poseattachpermutation, checkfourbar2Poseattachpermutation, checkfourbar3Pose(permutation), checkfourbarNPose and fourBarposesyntheses perform and validate the synthesis, poletriangle3Pose and poseofPSanalyze support the classical pole-triangle method, and fourBarposelayering/fourBarposeplotsolution/fourBarposesortsolution lay out, plot and rank the resulting solutions. Coupler-curve tools search and evaluate the path traced by the coupler: PLofPosecouplercurve, PLofFourbarcouplercurve, Posefourbarplotcouplercurve, Posefourbarsearchcouplercurve/Posefourbarselectcouplercurve and FourBarLinkageofCouplercurve go from a linkage to its coupler curve and back. Angle, torque and limit analysis quantify the running mechanism: Posefourbarangles/Posefourbarinvkin return crank angles for given poses, Posefourbarposelimit/Posefourbaranglelimit/checkfourbarsegment find the reachable angle range, and Posefourbartorquetrans returns the torque-transmission factor from crank to swing link. Degrees of freedom and truss analysis check feasibility independent of any specific pose: DoFGruebler applies the Gruebler-Chebychev equation, DoFGrueblerJacobiRank cross-checks the DoF via the Jacobian rank, and jacobiantruss/findCollinearEdges/ELlongestpath support general truss and graph analysis. Solid-adjacent helpers such as SGpianohinge, CPSinserthinge/CPSinserthingelist and SGofCPLzpianohingelist bridge to the printable hinge geometry of Section 6.6.4, and fourbardesignbysiri is a fun dictation-driven shortcut for quick four-bar sketches.

6.6.4 Four-bar and Hinge Solids ›

Four-bar and Hinge Solids turns the planar linkages and poses of the preceding sections into printable solid geometry (SG): four-bar bodies, piano and simple hinges, and linkage plates. Basic four-bar solids build the physical mechanism: SGfourbarlinkage creates the solid parts for a given linkage, SGofPosefourbar and SGofPoseCPLrot build a four-bar solid from a Pose or from body-relative CPL rotations, and SGfourbarlayering designs the solid frames with respect to a feasible layering. Plates and support hardware provide the surrounding structure: SGfourbarplate returns a base plate for a four-bar experiment, SGfourbarhullplate returns the hull shapes of crank and swing, and SGpinwheel designs mating saw-tooth pin profiles for a rotationally indexed joint. Hinge generation from contours converts 2D CPLR data into 2.5D extruded, hinged solids: SGofCPLRTzpianohinge, SGofCPLRTzpianohingeclosed, SGof2CPLTzpianohinge and SGofCPLRz extrude linkage contours and insert piano hinges, SGofCPLRwlim gives a quick, 3D-printable linkage from link contours, and SGsimplehinge360 prepares a new joint able to rotate a full turn. General linkage solids close out the group: SGofLPS creates 3D solids from a linkage-polyshape structure with revolute joints, and SGcentralprojection converts a planar linkage into a spherical one. SGfourbarlinkage2021 remains as an experimental variant developed for a specific project.

7 Automatic Design, Engineering & Simulation ›

This group is the original heart of SG-Lib: moving from manual or geometrically parameterized design toward criteria-based automatic design and optimization. Not only geometries but also kinematics and mechanisms are meant to be designed and optimized automatically, so a target (a force, a stress limit, a gear ratio) drives the geometry rather than the other way round. Manual, interactive design steps remain necessary wherever no concept yet exists for an efficient formulation of the task.

Section 7.1 (Parameterized Shape Generation) collects parametric part generators that build a shape from given dimensions. Section 7.2 (Parameterized Links and Joints) turns solid bodies into moving bodies via parameterized joints and structures. Section 7.3 (Automatic Shape Generation) goes one step further and

derives the shape itself from a functional criterion. Section 7.4 (FEM and Shape Optimization) analyzes and improves shapes by stress-based simulation. Section 7.5 (Multi-Body Simulation) simulates the resulting mechanisms dynamically in Simscape. Section 7.6 (Engineering and CAD Standard Parts) supplies DIN/ISO standard tables, standard-compliant designs and automated part-design functions. Section 7.7 (Robotics, Hardware and Sensors) bridges the design side to real robots, motors and sensors that did not exist when SG-Lib was started in 2017.

7.1 Parameterized Shape Generation ›

Parameterized shape generation builds a solid directly from given dimensions and specifications, without deriving those dimensions from a functional criterion first. A useful mental model: instead of computing a screw diameter from a required tensile force (that is automatic design, Section 7.3), you already know the diameter and just need the DIN/ISO-correct geometry for it — M8, not M6.234, because that is what actually exists. These functions turn such a specification into a printable or assemblable body, and they try to keep the number of distinct basic elements small.

Gears & gear trains build involute gearing (SGgearSpur, SGgearBevel, SGgearInternal, SGgearRack, SGgearCrownwheel, SGgearplanet, SGgearpair, SGgearchain). Links, rods & rockers create kinematic-chain elements (SGrodlink, SGpinshaft, SGmodelLink4, SGsblink, SGsbbase, SGsbpin, SGwheelspoked, SGwheelspiral). Motion transmission & couplings connect shafts and motors (SGmotorshaftcoupling, SGmotorspindleclutch, SGtorquefuse, SGshaftaxis, SGsteepspindle). Fasteners, snaps & standard-style parts model catches and screws (SGiso4026, SGiso4762, SGsnaprivet, VLFLsnapfit, SGhandle). Enclosures, adapters & drag chains cover housings and cable management (SGcase, SGboxradial, SGdragchain, SGdragchainelement, SGplugintube, SGwebcamcover). Bending-beam helpers estimate spring/cantilever behavior (lofbendinggirder, sofbendinggirder). The remaining functions are largely one-off project parts (catheter tooling, LEMO connectors, furniture spacers, medal frames) that double as design-pattern examples; browse by name for a similar use case before writing a new generator from scratch.

7.2 Parameterized Links and Joints ›

These functions turn fixed solid bodies into moving ones by parameterizing the joints and structural links between them, so the resulting parts already carry the frames needed to model a kinematic chain. They are the moving-parts counterpart to Section 7.1's shape generators and feed directly into the Simscape assembly of Section 7.5.

Joint & node primitives build the fixed and rotating connection points of a mechanism (SGmodelJoint, SGmodelNode, SGsimpleHinge, SGmodelKeyhole). Link families provide interchangeable rod/link styles for the same mechanism (SGmodelLink1, SGmodelLink2, SGmodelLink3). Gear-rack geometry supports rack-and-pinion designs (PLgearrackDIN). Bending-beam estimators complement the ones in Section 7.1 with width and force (bofbendinggirder, Fofbendinggirder). The remaining entries (SGfailedPrintJobs, createvideos4ergosurg, createvideos4mimed) are workshop/lab utilities unrelated to joint design and are kept here mainly for historical reasons.

7.3 Automatic Shape Generation ›

In contrast to parameterized shape generation (Section 7.1), automatic or function-based shape generation makes the geometry depend on reaching a stated criterion. A typical example: the force needed to compress a snap hook is specified, and the shape of the catch is computed so that this force is actually reached — the geometry is the output, not the input.

Functional & topology-driven design compute a shape from a mechanical requirement (CPLTopCM_SUN for compliant-mechanism topology optimization, SGgearratio for automatic gear-train sizing, ashft for a fixed non-round angle constant). Connectors & cutting solids join or separate two bodies by a computed bridge or cut (TLconnecting2SG, SGTconnectingplug, SGTcuthinge, SGcut2fit, SGpinringtail). Housings & covers wrap functional envelopes around other components (SGmotorhousing, SGgearboxhousing, SGboxbolted, SGboxturnswitch, SGclosurecapFDM). Catheter & drive mechanisms implement the recurring

intravascular-robotics assemblies (SGcatheterroller, SGpaprottkaRobot4LinAx, SGpaprottkaCathRotator, SGpaprottkaCathFeeder, SGpaprottkaRobotDrive). Surface & mesh finishing add texture, imprints or growth to an existing surface (SGgrowinsideoutside, SGstamptextFDM, SGhollowrhombus). As a rule of thumb: if you find yourself picking a dimension to satisfy a force, stiffness or ratio, look here before hand-tuning parameters in Section 7.1.

7.4 FEM and Shape Optimization ›

FEM analysis and stress-based shape/topology optimization build on the automated generation and modification of tetrahedron models for the MATLAB PDE Toolbox, starting from SG-Lib surface models. The shape- and topology-optimization procedures follow Sigeki Hirai and Claus Mattheck and were implemented by Yilun Sun.

Section 7.4.1 (Mesh and Model Creation) turns a solid geometry into a tetrahedral mesh and a PDE model. Section 7.4.2 (Load Definition and FS) lets you select faces and define surface loads on that mesh. Section 7.4.3 (Stress and Displacement) solves the FEM problem and returns stress and displacement fields. Section 7.4.4 (Shape Optimization) feeds the resulting stress distribution back into the geometry to grow or shrink material where it is over- or under-stressed.

7.4.1 Mesh and Model Creation ›

Mesh and model creation converts an SG surface model into the volumetric tetrahedral representation that the MATLAB PDE Toolbox needs for FEM analysis.

Tetrahedral meshing generates the tetrahedron list from a solid (SGtetramesh, SGmeshtetrahedron — functionally the same, the latter is the newer name). PDE model assembly wraps that mesh into a solvable PDE/FE model (pdemodelofSG for the classic geometryFromMesh path, FEModelFromSG and the simplified FEDiscreteGeometryFromSG for more recent MATLAB releases). Rule of thumb: prefer FEModelFromSG/FEDiscreteGeometryFromSG on current MATLAB versions and fall back to pdemodelofSG only for legacy scripts.

7.4.2 Load Definition and FS ›

Load definition and face selection (FS) let you pick which surfaces of the meshed solid a load acts on and visualize that choice before solving. pdeplotfaces plots the mesh faces so you can pick face IDs interactively, and SGplotsurfaceload then visualizes the assigned surface load on the solid geometry — a quick sanity check before handing the model to the solver in Section 7.4.3.

7.4.3 Stress and Displacement ›

Stress and displacement solves the FEM problem set up in Sections 7.4.1–7.4.2 and returns the mechanical response. pdesolvesurfaceload runs the PDE-based FEM analysis for a meshed model with an applied surface load; pdestressstatic returns the resulting static stress field inside the solid geometry; pdeplotresult visualizes the computed displacement in a three-quadrant view for quick inspection.

7.4.4 Shape Optimization ›

Shape optimization takes the stress field from Section 7.4.3 and modifies the geometry to reduce stress concentrations or excess material, following two classic biomechanically inspired methods. SGshapeOptiCAO implements Computer Aided Optimization by locally growing the surface based on stress; SGshapeOptiSKO implements the Soft Kill Option, which instead removes material in low-stress regions. Both loop mesh → load → solve → reshape, iterating with the functions from Sections 7.4.1–7.4.3 until the shape converges.

7.5 Multi-Body Simulation ›

Dynamic simulation of multi-body models in SG-Lib is based on automatically generating Simulink/Simscape Multibody models through Simscape's programming interface. Because Simscape itself keeps changing between MATLAB releases, these functions are particularly sensitive to the MATLAB version in use.

Section 7.5.1 (Simscape Model and Block Assembly) provides the low-level building blocks — frames, joints, connections and subsystems — to assemble a model programmatically. Section 7.5.2 (Simscape Library and Blocks) creates reusable Simscape libraries and physical-signal blocks. Section 7.5.3 (Simulation Run and Output) runs the assembled model, logs output ports and produces simulation videos.

7.5.1 Simscape Model and Block Assembly ›

Rapid assembly of Simscape Multibody (SimMechanics) models: blocks, frames, joints and connections, built programmatically instead of by hand in the Simulink editor.

System & subsystem setup create and manage the model structure (smbNewSystem, smbCreateSubsystem, smbConvertSubsystem, smbExpandSubsystem, smbCreateSG for wrapping a solid geometry into a subsystem). Frames, transforms & positioning place blocks in space (smbAddFrame, smbSetRotationMatrix, smbSetTranslationVector, smbSetTransformationMatrix, smbSetPosition, smbSGtransP, smbNewGridPos). Joints & drives connect bodies with kinematic constraints (smbCreateJoint, smbCreateSGJoint, smbCreateDrive, smbDeleteDrive, smbCreateStopJointR, smbCreateStopJointC, smbSetJointInputTorque). Lines & connections wire block ports together (smbAddLine, smbCreateConnection, smbGetAllLines, smbGetConnectedBlocks, smbDeleteUnconnectedLines, smbCopyConnections, smbNewLineRouting). Ports & block introspection query the live model (smbGetPorts, smbGetAllPorts, smbGetPortofFrame, smbNameofPort, smbGetBlockInfo, smbGetLibraryPath, smbWhich, smbSysPortLineTable). Masses, springs & sensors add physical elements and instrumentation (SGmodelMass, smbCreateSGMass, smbCreateSpring, smbAddFrameSensor, smbCreateSGNode). Rule of thumb: build bottom-up — blocks and frames first, then joints, then lines — and use smbshow/smbDrawNow to inspect the diagram at each stage.

7.5.2 Simscape Library and Blocks ›

Creation of Simscape libraries and physical-signal blocks, so that custom SG-based components can be reused across models like any built-in Simscape block. smbPSLibname/smbPSLibCreateBlock/smbPSBlockname manage a library and write individual .ssc block definitions, smbPSLibCompile and smbPSLibInstall build and deploy a compiled library, and smlNewSystem/smlDeleteall manage plain Simulink sheets alongside it. smlArduinoexample provides a ready-made servo-motor model for stand-alone Arduino deployment, and smlhelp opens the relevant programmatic-model-editing documentation.

7.5.3 Simulation Run and Output ›

Running simulations, logging output ports and generating simulation videos from an assembled Simscape model. smbSimulate starts the simulation of the current Simscape Multibody root model, with smbSetSimulationtolerance to adjust solver tolerance beforehand. smbLogOutputPort activates data logging on a chosen block output, and smbTofSimOut extracts the resulting time vector and transformation matrix from the simulation output. smbFullModelSimulation reconstructs the full SG configuration at a given simulation time, and smbVideoSimulation renders a titled video of the run.

7.6 Engineering and CAD Standard Parts ›

This chapter collects everything needed to make a design DIN/ISO-compliant: standard tables, ready-made standard parts, automated part designs that combine them, fit calculation, engineering dimensioning, and a library of finished CAD designs. Naming reflects the layering: DIN*/ISO* functions return table values, SG*DIN/SG*ISO functions return standard-compliant solid geometries, and SGdesign* functions automate the combination of such parts into joints or connections.

Section 7.6.1 (DIN/ISO Standards) provides the numeric standard tables. Section 7.6.2 (Standard Components) turns those tables into solid geometries of screws, nuts and gears. Section 7.6.3 (Automatic Part Designs) generates the subtraction/supplement solids that combine standard parts with a host body. Section 7.6.4 (Fits) computes bore adaptation for interference, transition and clearance fits, including 3D-printer-specific calibration. Section 7.6.5 (Dimensioning) produces engineering-drawing-style dimensioned views. Section 7.6.6 (Completed CAD Designs) hosts finished, mostly parameterizable designs built from all of the above.

7.6.1 DIN/ISO Standards ›

DIN/ISO standard tables and values used for design: DIN*/ISO* functions return the raw numeric dimensions from a norm, to be consumed by the SG*DIN part generators in Section 7.6.2 or looked up directly.

Threads & screws provide metric thread and screw-head tables (DIN13, DIN912 for DIN912/ISO4762, DIN913 for DIN913/ISO4026, DIN934, DIN965 for DIN965/ISO7046, DIN985 for DIN985/ISO4762, DIN7991). Washers & knurled parts cover flat and knurled hardware (DIN433, DIN464). Drilling & fitting give hole sizing and additive-manufacturing-specific fit offsets (DIN20273, DIN4AMfitting, DIN4AMtemperature). Torque & drive geometry cover tightening and drive-tip standards (DINthreadtorque, DINtorx, DIN_S_BANCLOK_E). Lookup helpers search across tables (DINfindinTab, DINhelp opens the relevant DIN/ISO reference URLs). Rule of thumb: call the DIN*/ISO* table function first to get valid discrete values, then hand the result to the matching SG*DIN geometry function.

7.6.2 Standard Components (DIN/ISO parts) ›

Solid geometries of DIN/ISO parts: screws, nuts, gears, bits and torx drives. These SG*DIN functions consume the tables of Section 7.6.1 and return ready solid bodies, as opposed to Section 7.6.3's designs that only cut/add material for such a part.

Screws & retainers build fastener bodies (SGDIN912, SGDIN912head, SGDIN985, SGDIN7991head, SGDIN965, SGDIN965head, SGscrewDIN/SGDIN13 for a DIN13-related screw). Washers & knurled parts cover flat hardware (SGDIN433, SGDIN464). Gears & racks build involute gearing as standard parts (SGgearpairDIN, SGgearrackDIN, SGgearandrackDIN, SGgearwormDIN, SGgearDINrot). Bits, torx & bushings cover drive tips and kerbkonus hardware (SGbitISO1173, SGtorxDIN, SGDINbushingE). Locking & sliding assemblies combine several standard parts into a mechanism (SGlockingDIN912DIN985, SGfourbarlinkageA0B0DIN912985, SGManipulatorDIN912DIN985ballbearing, SGslidingcarriage, SGslidingcoverofCPLz).

7.6.3 Automatic Part Designs ›

Parametric SGdesign* functions that combine standard parts (screws, bearings, motors) with a host solid as joints or fixed connections. Unlike Section 7.6.2's finished part geometries, these mostly return the subtraction and/or supplement solids to be booleaned into your own design at a chosen frame.

Screw & bushing fixations create bores and countersinks for DIN912/DIN985/DIN7991 screw joints (SGdesignDIN912DIN985, SGdesignDIN7991, SGdesignDIN912BushingE, SGdesignDIN7991BushingE, SGcutT4design, SGdesignCutDIN912DIN985). Bearing connections seat ball or radial bearings (SGdesignBallbearing, SGdesignDIN912DIN985radialbearing, SGdesignDIN912DIN985axialbearing, SGdesignDIN912DIN985axradbearing, SGdesignAxlebearing). Motor & shaft integration cut mounting pockets for small geared motors and shafts (SGdesignMotorN20, SGdesignMotorN40, SGdesignMotorN40horiz, SGdesignDrivenshaft, SGdesignDriveSpindle). Plug, peg & tongue-groove connections join two arbitrary bodies (SGdesignPeginHole, SGdesignConeconnector, SGdesignPintle, SGdesignPluginFrame, SGdesignTongueGrooveRadial, SGdesignfaceconnectingrod). Covers, supplements & bridges fill or close volumes between parts (SGdesignBoltedLid, SGdesignCoverofSG, SGdesignSupplement, SGdesignSupplement2connect2SG, SGdesignSupplement2connectSG, SGdesignSupplementPlugin). Rule of thumb: SGdesign* functions return the cut/add solids, not the final part — combine the result with SGbool (Boolean chapter) against your own geometry.

7.6.4 Fits (interference / clearance) ›

Bore adaptation for interference, transition and clearance fits, including 3D-printer fit calibration — since a printed hole rarely comes out at its nominal diameter. slfit/islfit convert between a desired fit class and the actual slot/bore size to print for a given printer; SGwarnfittinguse is a developer helper to flag SG functions that rely on such a fit. SGfittinggauge and SGfitcalibration4Printer produce test solids to measure achievable fits and friction directly on your printer, while bambufit and anycubefit simply preselect known calibration profiles for a Bambu Lab or Anycubic Predator printer as the current default.

7.6.5 Dimensioning ›

Engineering-drawing dimensioning of contours and views of a solid geometry, turning a CPL/SG into an annotated technical drawing.

Contour & view classification analyze which drawing elements a contour needs (CVLdimclassifier, TofCVLdimclassifier for 3D-contour radii, viewdimensioning to set the view parameters). Ruler-style dimensioning place a simple measurement ruler under a 2D contour (PLdimensioning, SGprofiledepthdimension for interactive distance-between-surfaces measurement). Multi-quadrant solid views generate full engineering views of a solid (SGdimensioning, SGdimensioning4, SGTdimensioning for a frame-relative four-quadrant view). Single-view drawings isolate one projection (SGdimensioningTop, SGdimensioningFront, SGdimensioningSide). Surface-specific drawings dimension a picked surface (SGprofilesurfacedimension, SGprofilecrossdimension).

7.6.6 Completed CAD Designs ›

Finished, mostly parameterizable CAD designs for 3D printing or further processing — a library of complete, real-world parts built from the tools in the rest of this chapter, useful both as ready-to-print solutions and as worked examples.

Robotics & medical devices include a lecture robot and clinical hardware (AIM_Robot, FTclutch, SGmodelOR, SGservoshaft, SGvenacavainferiorring, SGstentPeterEwert01, SGstentVeitPhillipMohamadAbdelhafez, SGcatheterGauge). BAUHAUS-style furniture & interior objects form a recurring design series (SGBAUHAUSbox, SGBAUHAUSdeskcylinder, SGBAUHAUSflorence, SGBAUHAUSnotesbox, SGBAUHAUSchairscratch, SGBAUHAUStectaF51, SGBAUHAUSpenholdertilted, SGBAUHAUSsamsungG5C49, SGBAUHAUScupholdinsert). Clamps, hooks & wall fixtures attach objects to rails or walls (SGveribor120, SGveribor210, SGveribor120yannick, SGhook4rail, SGlightrailholder, SGlaundryrackholder, SGfurniturewedges, SGfurnitureWedge4beamer). Adapters & spare parts fix specific commercial products (SGjuicerAdapter, SGceilinglight, SGsocketcover, SGprinterstand). SGmakerworld simply opens grabcad.com in the system browser as an external resource. Rule of thumb: search this section by product/project keyword before designing a similar bracket or adapter from scratch.

7.7 Robotics, Hardware and Sensors ›

Hardware, robot and sensor interfaces that did not exist when SG-Lib was started in 2017, bridging the pure-geometry world of the earlier sections to real actuators, cameras and tracking systems.

Section 7.7.1 (Serial Robots) covers drivers, kinematic models and tool sets for the JACO, Raambo and FeeTech-based robots. Section 7.7.2 (FeeTech Motor Control) talks to FeeTech servo motors over their serial protocol. Section 7.7.3 (USB and Serial Interfaces) provides generic USB/serial plumbing for Arduino, Pololu, PEAK-CAN and laser devices. Section 7.7.4 (RealSense Cameras) captures images, depth and point clouds from Intel RealSense cameras. Section 7.7.5 (NDI Vicra Tracking) interfaces with the NDI Vicra stereo tracking camera for pose measurement.

7.7.1 Serial Robots (JACO, Raambo, FeeTech) ›

Drivers, models and tool sets for the JACO, Raambo and FeeTech-based robots, spanning both real-hardware communication and offline simulation/modeling.

JACO driver & simulation talk to and mimic the JACO arm (JACOget, JACOset, JACOsim, JACOstruct, JACOmodel, jaco05, SGjacohand). Raambo communication & control set up and drive the Raambo robot over serial (raamboserialconnect, raamboserialport, raambostartup, raamboSET, raamboGET, raambocommandlines, raamboscreen, raambopathrecord, raambopathmove). Raambo TCP tool design build the interchangeable tool-center-point attachments for the Raambo's sixth axis (SGraamboTCPMagnetM20, SGraamboTCPMagnetM22, SGraamboTCPgearwheeldisk, SGraamboTCPgearwheeldiskTL, SGraamboTCPknurledgearcover, SGraamboTCPknurledgearcoverTL, SGraamboTCPbagcutter, SGraamboTCPmagnetpush, SGraamboTCPmagnetpull, SGraamboTCPtoolM16ofCPL, SGraamboTCPtoolM16extension, SGraamboTCPstraightmoveM16, SGraamboTCProtatingrodM16, SGraamboTCPtoolstand, SGraamboTCPcreatefulltoolset, CPLraamboTCPgearnut). Robot experiments &

flanges prototype alternative arms and mounts (SGROBEXP5, SGmotorFeeTech40flange). MIMED_LCL_Control is an Arduino (.ino) sketch that provides a robot control shell for the LCL setup.

7.7.2 FeeTech Motor Control ›

FeeTech control interface and contours for driving FeeTech SCS-series servo motors over the URT-1 serial-to-RS485 board. FTSCScmd builds the command byte sequence for the FeeTech SCS RS485 protocol, and FTSCSdata converts the raw bytes read back from the URT-1 board into usable data. FTSCSmotorlist scans the bus for connected motor IDs, FTSCScmdfindserial searches for the URT-1 board itself across ports and baud rates, and FTSCScallback is the serial-read callback that ties the communication together.

7.7.3 USB and Serial Interfaces ›

Auxiliary methods for USB and serial connections, serial buffers and device initialization (Arduino/Pololu/PEAK/laser), mostly aimed at macOS.

Port discovery & testing find and check available serial devices on macOS (SCavailable, SCtest, USBsearch, serialportused, USBhelp). Struct-based buffers implement an async read/write byte buffer independent of any specific device (sbuffercreate, sbufferinfo, sbufferwrite, sbufferget). Device initialization bring up specific external hardware (USBpololu for a Pololu Maestro controller, PEAKinstall for a PEAK USB-to-CAN adapter, LaserInit for a laser distance/data device, smlArduinohelp for connecting an Arduino to MATLAB since R2018b). SGmicroscopeclamp is a 3D-printable spare part for a typical USB microscope camera, grouped here with its electronics context.

7.7.4 RealSense Cameras ›

Support for Intel RealSense SR300 cameras: images, depth, point clouds and VL/VLFL export for use as a 3D scanner input to SG-Lib.

Device lifecycle start and stop the camera session (SR300start, SR300stop, SR300install to fetch the TUM-MIMED driver package). Raw image capture grab color, depth and combined frames (SR300videoimage for a 1920x1080 color frame, SR300depthimage for a 640x480 depth frame, SR300allimages for both at once). Geometry export convert camera data into SG-Lib data types (SR300getVL for a vertex list, SR300getPointclouds for point clouds, SR300getVLFL for a full surface). KBCallback is the keyboard callback used during interactive capture sessions.

7.7.5 NDI Vicra Tracking ›

Interface for the NDI Vicra stereo tracking camera, used to measure the real-world pose of tracked tools or markers as homogeneous transformation matrices.

Setup & status initialize the tracking session (Vreset, eminit for an NDI electromagnetic tracking system, Vplotstats to verify the camera is running). Marker & tracker queries return raw or registered tracking data (Vstray for the list of visible stray markers, VTget/VTgetp for an immediate vs. precise pose of one registered tracker, VTget2/VTget2p for the relative transform A_T_B between two trackers). VICRA is the class library underlying the NDI stereo camera interface, used internally by the functions above.

7.8 Design Automation Concepts ›

7.8.1 Script Language ›

7.8.2 Jarvis Concept ›

7.8.3 Claude Interface ›

7.8.4 chatGPT Interface ›

8 STL & 3D Printing & Voxel Models ›

SG-Lib grew out of the automatic design of additive-manufactured, patient-specific medical robots, so this group bundles everything between the solid-geometry world (SG) and the outside world of files, printers and cutters. Section 8.1 (STL Import/Export) reads and writes STL and related formats (OBJ, PLY, 3MF, STEP, MAT) to exchange geometry with CAD systems like CATIA, Solidworks or AutoCAD and with CAD archives such as GrabCAD. Section 8.2 (STL Packaging) prepares print jobs: it arranges multiple solids into a container, boxes them for shipping, and manages G-code for the 3D printer. Section 8.3 (STL Surfaces Analysis) checks whether an imported mesh is a watertight body or a set of independent open/closed surfaces. Section 8.4 (Vector Export (SVG/DXF)) turns 2D contours into cut files for laser cutters and plotters. Section 8.5 (Voxel Models) covers the complementary voxel-based representation (from DICOM/CT stacks) and its conversion to and from point lists, contours and SG surface models.

8.1 STL Import/Export ›

STL import brings design data from CAD programs such as CATIA, Solidworks and AutoCAD, as well as parts libraries from online vendors (e.g. Conrad Electronics) or CAD archives (e.g. GrabCAD), into SG-Lib. File I/O covers the main mesh formats: stlsolid, VLFLreadSTL/VLFLwriteSTL and their binary counterparts VLFLreadSTLb/VLFLwriteSTLb, STLAsctoBin and STLreadfiletype handle STL at the vertex/facet (VLFL) level, while SGreadSTL/SGwriteSTL, SGreadOBJ/VLFLreadOBJ, SGreadPLY, SGread3MF, SGreadSTEP-related functions (SGreadSTEPstructure, SGwriteSTEP, STEPmanifoldSolidBrepofSG) and SGwriteVRML do the same at the SG level. Archiving & batches collect and reload whole solids or model families: SGwriteSTLarchive, SGarchivefilename, SGreadSTLdir, stldir, SGwriteMultipleSTL and SGwriteSeparatedSTL. Print-job support prepares STL for 3D printers, including SGwriteSTLshrinked (shrunk copies to compensate shrinkage) and SGreadSTLreducedFL (facet-reduced import). Auxiliary I/O rounds things out with SGwriteMAT/SGreadMAT for MATLAB structs, SGreadtable/SGwritetable for annotated Excel tables, SGreadstructfile/SGwritestructfile for plain-text struct dumps, SGwriteIcon for a quick JPG preview, SGmeshlab to open a model in an external STL viewer, and the application callbacks appureadSTL/appsavegcaSTL. As a rule of thumb, prefer the SG-level functions over the lower-level VLFL functions whenever an SG equivalent exists, since VLFL functions return vertex list and facet list as two separate outputs instead of one SG struct.

8.2 STL Packaging ›

STL Packaging prepares solids for the physical 3D-print run once their geometry is finished. Layout & container functions gather several solids into one print job and pack them onto minimal build volume: SGClayout, SGboxing, SGboxpacking, binpacking3D and VLFLboxing (which also encloses the set in an optionally labeled box). Duplication & arrangement create symmetric copies for batch printing with SGpatternXYZ, SGcopyrotZ and SGarrangeSG (arranging a cell list of solids in a single row). Print orientation is handled by SGrotate4print, which rotates a part to keep printed lines as long as possible. G-code handling covers the actual machine files: SGgcoderead, SGgcodesimulate (Marlin dialect), SGgcodegetinfo, SGgcodefilenamechange, SGgcodefilechangetemp, gcodecopytemp, gcode2sdcard, gcode3mf2sdcard, gcodecopy2folder, gcodecleanup and gcodecleanup2dir. Parts-list & ordering functions document a build and link to vendors: SGpartslist, SGpartsfuncparams, SGpartslistastext, SGpartslistfindproductcodes, productcode, amazoncode2link and amazonlink2code.

8.3 STL Surfaces Analysis ›

After importing an STL file created by a program like CATIA, Solidworks or AutoCAD, it is worth checking whether the mesh forms a single watertight ("waterproof") solid or several independent open or closed surfaces before continuing with Boolean operations or printing. Boundary & packaging checks find and group these surfaces: surfacesofSG returns all feature surfaces (open boundaries) with a common edge, and SGpacking returns a package of all solids in a cell list. Point-in-region tests decide whether points lie inside a contour or shape: insideCPS tests a point list against a CPL polyshape or CPL, and isInteriorofCPL tests a single point against a closed 2D contour. Triangle-level geometry is analyzed by VLFLtrianglegeometry, which returns key metrics (area, normal, quality, …) for every triangle of a facet list. Run these checks before boolean or printing steps whenever a mesh's origin or integrity is uncertain.

8.4 Vector Export (SVG/DXF) ›

Vector export turns 2D contours (CPL) into cut files for laser cutters and plotters, complementing the 3D STL path. Typical uses are cutting textiles or neoprene shaped to fit an exoskeleton, or slicing a solid geometry into layers for laminated-object manufacturing (LOM) and producing them layer by layer on a laser cutter. SVG output is provided by CPLwriteSVG, which writes an SVG cut file directly from a CPL. DXF output is covered by CPLwriteDXF, which writes a 2D DXF file from a CPL, and CPL2DXF, which returns the underlying LWPOLYLINE strings with increasing object numbers for embedding in a larger DXF document.

8.5 Voxel Models ›

Voxel Models cover the second geometry representation in SG-Lib besides the surface-based SG: a regular 3D grid of intensity values (VM), typically coming from medical imaging (CT/MRI/DICOM). Choose the subsection by what you need to do with the volume. Section 8.5.1 (VM Import & Scaling) reads image stacks (DICOM, NIfTI) and converts between voxel and cartesian coordinates. Section 8.5.2 (VM Visualization) displays a voxel model as cross-sections, montages, pseudo-3D renderings or transparent voxel cubes. Section 8.5.3 (VM to VL, CPL) extracts point lists and contours directly from the voxel grid. Section 8.5.4 (VM to SG Surface Model) reconstructs a closed surface mesh (SG/VLFL) from the volume, e.g. via marching cubes or Delaunay triangulation, so it can rejoin the regular SG construction chain (Generation → Modification → …).

8.5.1 VM Import & Scaling ›

VM Import & Scaling brings volumetric image data into SG-Lib and keeps its coordinate systems consistent. Reading image stacks is done by VMreaddicom, VMuidicom (with a file-open dialog), VMreaddicomdir, VMreadnifti and VMofimagestack (inverse radon reconstruction from a stack of projections). Directory handling supports messy clinical exports: dicomCollectionSeparation sorts matching files into a coherent stack/directory, and VMrenamedicomdir renames DICOM files by series and instance number. Coordinate conversion translates between grid indices and physical space with voxofcart and cartofvox. Resizing & reshaping adjust the matrix itself: VM returns a size-reduced (squeezed) matrix, and VMresize resizes a voxel image model to new dimensions.

8.5.2 VM Visualization ›

VM Visualization is the largest voxel subchapter: it displays, navigates and interrogates volumetric data before or instead of converting it to a surface. Cross-section & overview plots show the classic radiology views with VMplot/VMplot_2012 (2x2 view), VMimage (three cross-sectional views), VMmontage (all slices in one figure) and matrixplot/GPLofmatrix for 2D matrices as grid point lists. 3D & pseudo-3D rendering give a spatial impression without full meshing: VMpseudo3D, VMscatter (particle model), VLplotvoxel/VMplotvoxels (transparent voxel cubes) and marchingcubes (the classic iso-surface extraction algorithm). Interactive inspection reads back user input and window state: VMginput, VMgetSubplot, VMdoubleclick, VMcaxis/VMcaxis2/VMwindowplot (contrast/window adjustment) and VMaxis (min/max intensity). Image processing & analysis operate on the matrix values: VMintensityscale, VMnoisethreshold, VMneighborhood, MinMaxofmatrixneighbors, VMimrot90, VMof2Dprojections and the grid/edge helpers GPLnearestpoint, GPLdistancegridpointsofCPL, ELofmatrixneigbors and ELofmatrixneigborweight. Conversion helpers VMofVL and VMofSGweight build a voxel model from a vertex list or a weighted SG, bridging back to the surface-based functions in 8.5.4.

8.5.3 VM to VL, CPL ›

VM to VL, CPL provides the two lightest-weight extractions from a voxel model, useful when a full surface mesh is not needed. VLofVM returns the coordinates of all (occupied) voxels as a plain point list (VL) — note this list can become long for high-resolution volumes. CPLofVM returns a contour around the (transformed) voxel image, i.e. a 2D outline suitable for further CPL-based processing such as vector export (Section 8.4).

8.5.4 VM to SG Surface Model ›

VM to SG Surface Model reconstructs a closed mesh from a voxel model, rejoining the surface-based SG construction chain. Marching-cubes reconstruction is the standard approach: VLFLofVMmarchcube and SGofVMmarchcube extract an iso-surface at the VLFL and SG level respectively, and SGofVMisosurface does the same directly as an SG. Delaunay reconstruction offers an alternative, often smoother, path via VLFLofVMdelaunay and SGofVMdelaunay. Region/ray helpers include TR3ofVM (a 3D triangulation of the volume) and VLofimage, which converts a single 2D image into a contour or a thin solid. The inverse direction is covered by VMofSG, which rasterizes an existing SG back into a voxel model — useful for boolean-style operations that are easier on a regular grid than on a mesh.

9 Testing, Development, AI & Media ›

This top-level group bundles everything around testing, tooling and media that supports the construction chain rather than extending it. Section 9.1 (Sample Generation) creates defined test geometries such as point lists, contours and solids. Section 9.2 (Test Generation) builds and re-runs systematic function tests across the toolbox. Section 9.3 (SG Check) and Section 9.4 (SG Repair) detect and fix surface anomalies such as open edges, non-manifold topology and self-intersections. Section 9.5 (Software Development) collects file, system, debugging and general-purpose utilities used while writing and packaging SG-Lib itself. Section 9.6 (AI, Data and Integration) covers LLM access, expert systems and external data/FileMaker interfaces, and Section 9.7 (Video, Audio and Media) supports the production of tutorials and lecture recordings.

9.1 Sample Generation ›

Sample Generation provides ready-made test data with well-defined properties so that any new or changed function can be tried out without first modeling a real part. Basic geometry samples cover every data type in the type chain: PLsample and VLsample return plain 2D/3D point lists, CPLsample builds nested closed contours, VLFLsample creates open surfaces, and SGsample returns closed test solids; CSGsample supplies pairs of solids specifically for exercising Boolean operations. Specialised samples target particular subsystems: SGerrorsample deliberately injects surface problems for the checkers in Section 9.3, SGjointsample and SGCLsample provide contours and cutting frames for joint and CPL-command tests, and Posesample/SGfischertechniksample/SGservosample supply linkage poses and ready STL parts. Statistical and geometric edge cases are covered by VLsampletriangles/PLsampletriangles (extreme triangle shapes) and stdexplain2/ttestexplain (illustrate mean, standard-deviation and estimation behaviour). Rule of thumb: pick the sample function whose name matches the data type you need to feed into the function under test.

9.2 Test Generation ›

Test Generation supports building and re-running systematic tests across the whole toolbox, so that changes and improvements to existing functions can be verified with broad parameter coverage rather than by hand. VLFL_Toolbox_make assembles the toolbox directory that is later compiled into a shippable Matlab toolbox, and VLFL_Toolbox_test publishes all tutorials to exercise that compiled version end to end. testfunctTL runs a function against a generated list of varied parameter sets to catch regressions, while subfig is a small display helper that arranges multiple test-result figures on screen like subplot. Rule of thumb: use this group when you need repeatable, parameter-swept regression testing rather than a single hand-built example.

9.3 SG Check ›

SG Check examines surfaces and solids for correctness and anomalies — useful after importing an STL file that was not created with SG-Lib, while developing new toolbox functions, or after an already-tested function terminates unexpectedly. The group mirrors the checks a tool like MeshLab performs: open edges of bodies or surfaces, non-manifold edges (an edge shared by more than two facets), non-manifold vertices (a vertex shared by unrelated surface patches), and self-penetration of the mesh. Boundary and manifold checks are covered by VLFLcheckboundaryedges/SGcheckboundaryedges, VLFLchecknonmanifoldedges/SGchecknonmanifoldedges and VLFLchecknonmanifoldvertices/SGchecknonmanifoldvertices, with VLFLcheckmeshlab/SGcheckmeshlab bundling the most common tests into one call. Vertex and facet hygiene is handled by VLFLcorrectdoubledvertex, FLsort and VLFLcloseopenfacets, while VLcheckvertexaccuracy/SGcheckvertexaccuracy estimate how many digits of rounding a mesh actually needs. Intersection analysis functions such as cross2F, VLFLcrossF, intersectmatrix/intersectcontour/intersectfacets and iscollofSG compute crossing contours and collisions between triangles, facet lists and solids, and CLmissingedgesVLFL/CLdecomposecontoursVLCL trace missing edges around open facets. Rule of thumb: run SGcheckmeshlab first for a broad overview, then drill into the specific check that flagged a problem.

9.4 SG Repair ›

SG Repair collects heuristic strategies for correcting surface tessellation once SG Check (Section 9.3) has located a problem. Imperfections such as open surfaces, non-manifold edges, non-manifold vertices and self-penetrating facets often go unnoticed because the mesh still displays fine on screen; additive manufacturing based on slicing tolerates such flaws quite well, but Boolean operations and tetrahedral meshing (e.g. for the PDE toolbox) are very sensitive to them. Correction is not always possible or unambiguous, but classic cases from CAD export, shape optimisation (SGshapeOptimization) or voxel/surface-to-solid conversion recur often enough to warrant dedicated fixes. Closing and rebuilding facets is handled by FLrepair, VLFLrepair, ELrepair and SGrepairFLsimple, while SGrepairnonmanifoldvertices and the experimental SGrepairnonmanifoldedges remove shared-topology problems between one or more surfaces. SGwatertightofSG produces a watertight reconstruction within about 60 seconds, and SGcloseopenBoundary offers an imperfect but useful attempt at closing remaining open boundaries. Diagnostic companions VLFLcheckselfintersection/SGcheckselfintersection, VLFLchecktrianglegeometry/SGchecktrianglegeometry and VLcheckboundaryaccuracy quantify the remaining problems, while SGmakemissingvertex/SGmakemissingfacet/SGmakedoublevertex/SGmakedoublefaces deliberately introduce such errors for testing the repair functions themselves. Printing-oriented checks such as SGslicercheck, CVLzcheckprinting, SGcheck4cura and PLcorrelationcheck flag issues that specifically affect slicing and FDM output.

9.5 Software Development ›

Software Development bundles the infrastructure used to write, package and maintain SG-Lib itself, independent of any actual geometry work. Dependencies matter whenever functions are packed into toolboxes or renamed in the source, which is why depuse ("who uses this function") and depfun ("which functions does this one depend on") underpin much of this group. Section 9.5.1 (File & Folder Handling) manages folders, filenames, paths and zip archives; Section 9.5.2 (System, macOS & Hardware Tools) wraps OS processes, email, Java and Arduino access; Section 9.5.3 (Toolbox Build & Documentation) builds the toolbox, p-code and HTML/JSON documentation; Section 9.5.4 (Debugging) supplies breakpoint, print and workspace helpers; Section 9.5.5 (String, Cell & Data Utilities) offers general string/cell/array conversions; Section 9.5.6 (Media & Document Files) handles PDF/video/image file conversion; Section 9.5.7 (Web & Company Data) provides web and company lookups; and Section 9.5.8 (General / Misc Helpers) catches everything else that supports day-to-day development.

9.5.1 File & Folder Handling ›

This group provides access to folders, files, filenames, paths and zip archives used throughout SG-Lib development and daily file management. Path and directory discovery is handled by desktopdir, matlabdir, matlabtoolboxdir, SGtoolboxdir and dir2volume, which locate standard directories across machines and external volumes. Filename manipulation covers expname, filenameinc, filenameadddate/dateinfilename and filepartl, which build, increment or decode Tim-Lueth-style dated filenames, while renamefile, movefiles and copyfilebycond move or rename files with conflict handling, including wildcard support that plain movefile lacks. Date and metadata utilities such as fcreationdate/fcreationdate2, fcreationdateSet, ftouch/ftouchcreationdate and creationdatefile read or set file creation dates, and imagefileadjustfolder/imagefileadjustyearfolder/imagefileadjustcreationdate apply this recursively to whole photo archives. Archiving and housekeeping are covered by zipdir, desktopcleanup, desktopshadow, downloadcleanup and isfileinfolder, and plain text I/O is done with readtextfile/writetextfile, repairtextfileEOF and isUTF16file. Rule of thumb: for anything that touches file dates or Tim Lueth's dated-filename convention, check here before writing a new helper.

9.5.2 System, macOS & Hardware Tools ›

This group wraps operating-system, macOS, email, Java and Arduino/hardware functionality that Matlab does not expose directly. Process and system monitoring is covered by whoisPID, whoisTCP, isprocrunning, sudokill, spindump/spindumpkill/app_spindumpkill and checkmatlabfulldiskaccess/checkmatlabxcodetoolaccess, which diagnose and fix processes that can make Matlab unstable on macOS. Network and login utilities include whoisIP, pingIP, vpnstatus and whoisloggedin, and version checks are handled by verOSX, verJAVA, verXCODE and osx. Email access goes through the Java-based JAVAemailServerImap/JAVAemailTUMimap and JAVAemailreadtextmessage/JAVAemailmsg2cell, and Round-Robin cooperative-multitasking for simple realtime scenarios is provided by RRrun, RRshell, RRkeyboardCallback, RRhelp and RRcputime together with cputic/cputoc. Hardware and workflow helpers such as SGrobotarduinoIDE, arduinoTL, AVRarduinoMega, homebrewinstallonmac/homebrewtest, workdir/work and addenv round out the group by opening tools or preparing the working environment. Rule of thumb: reach for this group whenever a task needs to shell out to the OS rather than compute geometry.

9.5.3 Toolbox Build & Documentation ›

This group builds the SG-Lib toolbox itself and generates its HTML/JSON documentation and p-code. Dependency analysis is central: depuseTL/depuseToolbox/depuseString/depuseStringmult and depfunTL_2016b answer "who uses this function" and "what does this function depend on", which matters whenever functions are renamed or repackaged. Packaging and compilation are handled by pcodeTL/pcodeTLaddmfilehelp, SGcreateToolboxfromFolder, SGcreateToolboxMfilesfromPfiles and VLFL_Toolbox_revision, which turn m-files into distributable p-code toolboxes and minor-revision updates. Documentation generation covers SGcreateHTMLdocumentation4Matlab, SGcreateJSONfnctSignatures4Matlab, SGcreate_demos_xml_File, helptoc2htmllist/SGcreateToolboxHTMLfilesfromhelptoc and SGconverthtmldoc2UTF8, while SGcheckHTMLdocumentation4Matlab/SGmissingdocumentation/VLFL_Toolbox_missingdoc verify that every function actually has matching help and demo entries. SGlibpath, SGlibmovepathtotop and SGlibgithubupload manage where the toolbox lives and how it is published, and SGexplain/SGlibHelpTutorials/HELP_SGLIB_* provide wiki-style help texts and tutorials for end users. Rule of thumb: before hand-editing documentation files, check whether one of the SGcreate…/SGcheck… functions here already generates or validates them.

9.5.4 Debugging ›

Debugging collects small helpers for breakpoints, targeted output and workspace inspection that go beyond Matlab's built-in debugger. Variable inspection across function boundaries is handled by dbsave/dbload (write/read a variable to/from the global workspace) and dbcheck, which checks whether a variable is already stored there. Controlled output during a run uses dbprint3D, dbplot (plots SG/PL/CPL/CVL objects for visual debugging), dbprintftext/dbprintfwarning and dbprintflog, with dbprintmode/dbstacklevel switching on the ability to jump to specific code lines and controlling how deep the call stack is reported. dbpause works like pause but also shows the calling function, dbreturn offers an early-exit mechanism Matlab itself lacks, and dbopenfiles lists currently open functions/files for orientation during a debugging session. Rule of thumb: use dbsave/dbload to move a value out of a deeply nested call, and dbplot/dbprint3D to see a geometry object without stopping the whole computation.

9.5.5 String, Cell & Data Utilities ›

This group provides general-purpose string, cell-array and numeric-conversion helpers used throughout SG-Lib. String and cell handling covers strrepn, cellstrfind, numofstrtab, uniqueNaN/unsortNaN (for NaN-separated lists such as CPL), array2cell/cell2class/class2cell and stripfields/retainfields for struct field management. Matrix and grid utilities include matrixrand/matrixvalues/matrixneighbors, matrixdistancefield/matrixfilldistance/matrixfillborder/matrixfillseedpoints for potential-field style computations, matrixskeleton for skeleton paths, and rcofind/indofrc for converting between linear and row/column indices. Unit and value conversions are a strong cluster here: Charriere2mm/French2mm/Gauge2mm/AWG2mm convert catheter and wire gauges to millimetres, mm2inch/mm2French, Fahrenheit2Celsius and PS2KW/KW2PS handle everyday engineering unit conversions, and single2parts/code3Byte2single/parts2single decompose and rebuild floating-point numbers at the bit level. Statistics helpers meanGauss/erfGauss/erfinvGauss/roundgauss/randnorm support rounding and distribution-aware computations, and sequence helpers such as linearinterp, scalesequence, ismonotonously and risemonotonously operate on ordered numeric sequences. Rule of thumb: check this section before writing a new small conversion or array-shuffling helper — it very likely already exists here.

9.5.6 Media & Document Files ›

This group handles conversion and management of PDF, video and image document files. PDF housekeeping is the largest cluster: PDFadjustcreationdate/PDFadddate2filename apply Tim Lueth's yyyy-mm-dd filename convention to scans, PDFcheckExt/isPDFfile verify file extensions and type, PDFcheckOCR flags PDFs without a usable text layer, and PDFscanfoldercheck runs these checks as an hourly batch job on the scanner directory (see scannerdir). Video format conversion is covered by videoffmpeg, videoWMV2MP4/videoAVI2MP4, videoffmpg2gif and bambudownloadvideoIP, which use ffmpeg to convert or fetch timelapse videos, for example from a 3D printer running in parallel to a print job. Image conversion is handled by imagePICT2JPG and imageisformat, and Videoquickmovecursor moves a cursor overlay to a given point for tutorial recordings. Office-application helpers word/powerpoint simply open the corresponding application for daily reports, with powerpointwordexcelbugs documented as a workaround note for a known VBA bug. Rule of thumb: for anything date-stamped in Tim Lueth's PDF naming scheme, look here before writing new logic.

9.5.7 Web & Company Data ›

This group performs web and company lookups and reads external data sources. Company/vendor shortcuts form the largest cluster: functions such as conrad, igus, FESTO, Hoffmann, kugellagerexpress/ekugellager and amazonparts simply open the website of a parts supplier, saving repeated manual navigation, while bank/press shortcuts (bekb, vrba, coba, welt, nzz) do the same for personal login and news pages. Financial and statistical data access is covered by FINdataM2USAread/FINdataM2EURread (money supply from FRED/ECB), FINdataPOP (population data) and FINpythonpanda, which activates the Matlab-Python bridge for pandas-based analysis. General web/file utilities include loadweb/loadwebdir for fetching .mat files and paths from a URL, MIMEDwebdisk for TU Munich's webdisk, and finddoubledrows/finduniqueVL/find2logical for cleaning up row-based data read from such sources. DeepL opens the browser-based translation tool, and youtube/typo3 open Tim Lueth's channel and the SG-Lib homepage CMS respectively. Rule of thumb: this section is for external, internet-facing lookups — pure in-Matlab computation belongs elsewhere.

9.5.8 General / Misc Helpers ›

This is the general-purpose catch-all for helpers that do not fit any other Software Development subsection, spanning geometry primitives, engineering calculations, batch jobs and personal productivity shortcuts. Small geometry/index helpers include pgrid, sizeVL, lsegment, crosspointVLFL/crosspointVLFL2, epsofVL and imat/removeimat, which support bounding-box, grid and index bookkeeping used across the library. Engineering and physics calculations cover dimpowercable/wirecurrent2cs (cable sizing), VanDerWaalsRadius/atomiccovalentradius (chemistry constants), VofRsphere (ball volume for counterweights) and the MGTsystemIIR/MGTsystemFIR/MGTmotor family, which models filters, motors and control loops. Job and batch management is handled by SGbatchjob/SGbatchresult/SGbatchdelete and batchunique/batchjobsTL/batchlastcommand, which submit and track Matlab's cluster batch jobs. File/editor and lecture-recording shortcuts such as openinmatlabeditor, opentolineTL/editTL/docTL, clear4lectures/clcl/clcd and SGprint/SGprintPreForm round out daily development and 3D-printing workflows, while numerous single-purpose links (timlueth, robotYK1, Wegertseder, AM_3DPrinting) simply open a specific website or generate a specific report. Rule of thumb: search by keyword before assuming a helper does not exist — this section is large and often already has exactly what you need.

9.6 AI, Data and Integration ›

AI, Data and Integration covers everything that connects SG-Lib to large language models, classic expert systems, external data sources and outside databases. Section 9.6.1 (LLM / ChatGPT (OpenAI)) wraps the OpenAI API for chat, assistants, vector stores and generated images/video. Section 9.6.2 (Expert Systems and STRIPS Planning) implements classic rule-based planning such as the General Problem Solver on blocks-world and sliding-puzzle examples. Section 9.6.3 (AI Text, String and Cell Helpers) supplies the string and cell-array plumbing the AI functions need to build prompts and parse answers. Section 9.6.4 (Web and External Data Sources) queries Google Scholar/Patents and the German federal statistics office. Section 9.6.5 (FileMaker Interface) provides a JDBC/SQL bridge to FileMaker databases, and Section 9.6.6 (Encryption) offers simple routines for encrypting data and web access.

9.6.1 LLM / ChatGPT (OpenAI) ›

This group wraps the OpenAI/ChatGPT API for text, image and video generation, plus assistants and vector stores for retrieval-augmented workflows. Simple one-line chat access is provided by chat1/chat/chatm/chatas (single-line to persistent, thread-based conversations), finchat (forces a single numeric answer) and chatweb/chatGPT, which just open the browser interface. Content-generation functions cover chatimage/chatimage2video (DALL·E-based sketches and diagrams), chatvideo (Sora-based 12-second video clips), chatcode (returns and writes runnable Matlab code) and chatdiagramm (generates plotting code from a description). Text-processing helpers chatspellcheck/chatclipcorrect, chatfilename, chat2table and chatscienceclozetext apply the LLM to spelling/translation, filename generation, table extraction and cloze-text tasks respectively, while document-analysis functions OpenAIPDFtextclassification/PDFcreateOpenAIfilenames/OpenAIPDFemailanswers use OCR plus the LLM to classify and rename scans or draft email replies. Assistant/vector-store management forms a large CURL-based cluster: openAIcreateAssistantCurl/openAIcreateThreadCurl/openAIstartRunCurl/openAIwaitForRunCurl drive a full assistant run, and openAIcreateVectorstore/openAIuploadFile/openAIgetVectorStoreFilesCurl/openAIdeleteFileFromVectorStoreCurl manage the files an assistant can search. Rule of thumb: use chat/chatm for a quick single answer, and the openAI…Curl assistant/thread functions only when you need persistent, file-backed context.

9.6.2 Expert Systems and STRIPS Planning ›

This group implements STRIPS-style expert-system planning: rule bases, condition checking and the resulting plan sequences. XPSGPS, XPSblocksworld/XPSblocksworldequation and XPSslidingpuzzle are worked examples of the (first) General Problem Solver applied to classic AI planning domains such as blocks world and the sliding puzzle, with XPSEQS handling equation-based problem states. Rule management is supported by XPSrulesread/XPSruleswrite (read/write rules from/to an Excel sheet), XPSrulechecker (flags easy human errors in rule definitions) and XPSruleanalyzer (collects the unique terms used across a rule base), while ruleconclusion performs the actual single-step rule-checking inference and XPSconditionfindneval evaluates all combinations of a variable-based condition. json2matlab, jsonplansequence and json2TextMatlabcodeUsecase bridge this classic planning approach with modern LLM output by turning a JSON chat response into a Matlab plan or code sequence, and wiesoweshalbwarum applies the chatm function to structured “why/what/how” question templates. Rule of thumb: use this group for symbolic, rule-based planning tasks, and the json2… bridges when an LLM response needs to become an executable plan.

9.6.3 AI Text, String and Cell Helpers ›

This group supplies the string and cell-array plumbing that the AI functions in Section 9.6 rely on to build prompts and parse LLM answers. Line and paragraph formatting is handled by strbreaklines, str2codelines, sprintfvec/fprintfvec2/sprintfvecUP and cell2strlines, which turn long text or numeric vectors into readable, code-ready or engineering-formatted output. Pattern search and replace covers strfindafter/strfindtag, strfind2double, strrepalias and the stringpatterncompare/stringpatternreplace/stringpatternpermute/varsofstringpattern family, which match and substitute symbolic variables inside string equations. Hyperlink helpers strhyperlink/strhyperlink2str/strhyperlinkmatlab/sprintfhyperlinktab build clickable command-window links to files, code lines or table entries. Encoding and cleanup utilities isUTF8, latin1ofUTF, char2html/html2char and hexstr2str/str2hexsstr/hexstr2uint8 convert between character encodings, and siri2jarvis converts spoken or written natural-language sentences into a command form the library can act on. Rule of thumb: reach for this group whenever an AI function needs to assemble, clean or decode text rather than reason about it.

9.6.4 Web and External Data Sources ›

This small group reads external research and statistics data over the web. googleScholarSearch performs a Google Scholar search for given keywords and uses OpenAI to help interpret or filter the results, complementing the patent-search helpers found in Section 9.5.7. destatisJSONfindtabs and destatisJSONreadtab access the German federal statistics office's (Destatis) GENESIS database API to locate and download official data tables directly into Matlab. Rule of thumb: use this group when a task needs authoritative external research or government statistics rather than a general web page.

9.6.5 FileMaker Interface ›

This group provides a JDBC/SQL interface between Matlab and FileMaker databases, which Tim Lueth uses to document and manage the SG-Lib function catalog itself. FMopen and FMinitJDBC establish the connection, checking for and installing the FileMaker JDBC driver if needed, and FMclose closes it again. FMgetData and FMgetFieldTabs read all field values or the table/field structure of an open database, while FMsqlQuery runs an arbitrary SQL request for more targeted reads. FMselectSQLcmd and FMinsertSQLcmd build SQL SELECT and INSERT strings in a more intuitive, parameterised way than writing raw SQL by hand, and FMhelp documents the whole Matlab-FileMaker interface. Rule of thumb: this is the layer through which the chapter texts and function descriptions you are reading right now are ultimately stored and retrieved.

9.6.6 Encryption ›

This small group offers simple routines for encrypting data and securing web access. SGencrypt and SGdecrypt create and consume a 32-byte password/key derived from a string, forming the basic building block for the other functions here. strencoder/strdecoder implement a lightweight, random-key-based string encoder and matching decoder for cases where a full cryptographic library is unnecessary, and webdec works like the standard web function but reads its target link in encrypted form via strencoder/strdecoder, keeping URLs or credentials out of plain view in the source code. Rule of thumb: use these for lightweight obfuscation of scripts and links, not for security-critical data.

9.7 Video, Audio and Media ›

Video, Audio and Media bundles the tools used to produce tutorials and lecture recordings around SG-Lib, separate from any geometry construction. Section 9.7.1 (Video) covers frame extraction, title/end-title generation, clip assembly and format conversion between GIF/MP4/AVI/WMV. Section 9.7.2 (Audio) covers recording, trimming, resampling, vocoder effects and spoken-text editing for narration tracks. Together they cover the full pipeline from a raw screen or camera recording to a finished, narrated tutorial video.

9.7.1 Video ›

This group creates and edits video content for tutorials and lecture recordings, from raw frames to finished clips. Frame-level access is handled by imageVideoFrames (extracts still images from a video) and videoCopyFrames (copies frames between a Videoreader and Videowriter), while imageVideoWrite writes a single image repeatedly into an open video stream, for example to hold a static title. Title and framing functions imageVideoTitle/imageVideoEndtitle/imageVideoImagePage generate title, end-title and copyright image pages, and videoCopyCutMovies/videoWriteClipMovie/createvideoscript assemble cut sections, images and clips into a finished silent movie or full tutorial script. The Videoquick* family (Videoquickstart, Videoquickwritetextpage, Videoquickwritegcf, Videoquickrotate3D, Videoquickanimatego, Videoquickwritelastcommand, videoquickmovecursor) offers a simplified, script-friendly way to turn Matlab figures directly into video content, including recorded 3D-view rotations and on-screen cursor moves. Format conversion and audio handling round out the group via videoGIF2MP4, video2nosound and videosplitaudio/imageaudio2video, which extract or attach an audio track. Rule of thumb: start with the Videoquick* functions for a simple screen-capture-to-video workflow; use the imageVideo*/videoCopy* functions when you need frame-level control.

9.7.2 Audio ›

This group handles audio recording, editing, resampling and spoken-text processing for narrating tutorials and lectures. Recording is covered by audiorecorderTL/audiorecorderTLautostop (records from a specified device, optionally stopping automatically) and audiorecordspokentext, which records a list of sentences and compiles them into one longer narration track. Editing and analysis functions audiotrim, audioseparatebreaks, audioeditspokentext (keyboard-driven editing of spoken text) and audionoiselevel (detects the noise-level threshold) help clean up and structure a raw recording, while audioresample and audiovocoder change sample rate and pitch respectively without altering the other. audiowriteTL/audioreadTL wrap the standard write/read functions with a fixed 44100 Hz format and optional playback, soundTL extends sound with support for specific audio output devices, and audio4slide combines a still image and a text/audio track into a slide-style presentation segment. Rule of thumb: use audiorecordspokentext plus audioeditspokentext for narration built from short sentences, and audioresample/audiovocoder only when the sample rate or pitch itself needs correcting.

9.8 Tutorials and Worked Examples ›

9.8.1 Guided Tutorials ›

9.8.2 Worked Examples (VLFL_EXP) ›

New and not yet assigned ›

This chapter is a temporary holding area. It collects functions and experiments that have not yet been assigned to a regular chapter of the SG-Lib - newly added helpers, prototypes and dated experiment scripts (often named exp_YYYY_MM_DD). They stay here, outside the main reference flow, until they are reviewed and moved to the chapter where they belong. Treat everything listed here as unstable: names and behavior may still change, and some entries are one-off experiments rather than supported functions.

Part II

Function Reference

The detailed catalog. Each chapter lists its functions with a one-line summary, then a detailed description with inputs, outputs and an example image. The bracketed link next to each chapter title jumps back to its Guide entry in Part I.

1 Data Structure and Manipulation [guide]

1.1 Point List Manipulation [guide]

Functions in this section:

CPLinsertcollinearpoints — inserts collinear auxiliary points that a minimum point distance is guaranteed

CPLlength — returns the boundary length of a CPL

CVLinsertcollinearpoints — inserts collinear auxiliary points that a minimum point distance is guaranteed

GPLdistancetoCPL — returns for a grid point list the point distance insider or outside of a contour

GPLparameters — Auxiliary fnct to calculate xmin ymin dx and dy from a given meshrid or VL

isPL — PL has no NaN separator, 1st and last point are different

matrixdistanceofCPL — returns a simple matrix containing the distance to the contour CPL

matrixofCPL — simplified version of matrixdistanceofCPL

PLaddauxpoints — adds auxiliary points to an open point list

PLaddTetragridofCPL — returns a tetra grid point list for a bounding box or CPL

PLadjust — returns an adjusted PL related to a position grid and a angle grid

PLarrow — returns a point list for an arrow

PLbending — bends a CPL on a cylinder or spherical ellipsoid

PLBezierC — Bezier splines for open point list

PLcat — returns a concatenated point list of several point list

PLchamfer2side — two sided version of PLchamfer

PLcirclearc — returns an circle arc between two circle points in right hand rule

PLcircleofPL — returns minimal and maximal circle around a CPL

PLcircleovalline — creates staight lines between points extrudes them by using different radii at the points and round edges

PLcirclesinus — returns a circle-based-sinus shaped point-list

PLcircleui — returns a PL for a circle interactively defined

PLclampCTtable — returns the CPL of a clamp for an hybrid-OR CT-table

PLconnectopenends — closes an open line by a contour

PLcorrelatingsegments — A new implemenation of PLcorrelate

PLegg — returns the contour of an egg

PLfft — same as CPLfft but for open contours

PLgearSIN — Based on PLgearDIN, a purely sinusoidal tooth profile

PLgridhull — returns a convex hull around the given Point list

PLgrow — grows a contour line in a distance to an open or closed PL

PLhexalobular — returns a hexalobular Point list related to DIN ISO 10664:1999-09-01

PLhook — creates the shape of a hook

PLinbus — similar as PLcircle but uses the wrench size

PLinsertcollinearpoints — inserts collinear auxiliary points that a minimum point distance is guaranteed

PLinterlock — returns lines or closed polygons to create interlocking surfaces for kinematic DoF

PLjunkerhallpart — creates a profile of a pressed profile

PLknurled — returns a sinus contour on a circle

PLmeancluster — converts a point list into a mean values point list

PLmeshgrid — simple fnct for a better understand of meshgrid

PLminxy — returns the point with minimal x and y values

PLminyx — returns the point with minimal y and x value

PLmirroratline — mirrors a PL or CPL

PLofCPLequidistant — returns points on closed or open contours with equal distances

PLofFloorPlan — Returns a point list from a distance angle list

PLofimcontourc — returns the centers of pixel coordinates in an image

PLofVL — Converts a planer VL[nx3] into a PL [nx2]

PLofVLFL — returns the 2D representation for a planar VLFL

PLofVLplaneprojection — main axis projection of a vertex list along the largest dimension of a normal vector

PLofVLprojection — returns image point on a target screen

PLonCPLcontour — findet äquidistante Punkte entlang einer Kontour

PLoutercontour — returns a point list that has no crossings

PLpinring — returns the contour of a SGpinring

PLplotdiscretesample — plots a PL as a digital signal to avoid problem of misinterpretation

PLplotdiscretetime — plots a PL as a digital signal to avoid problem of misinterpretation

PLradialEdges — returns a point list (PL) with rounded edges

PLreassign — reassigns the coordinates of a new list to the indices of an original list

PLrhombus — Point list of a rhombus

PLrope — creates cross section of twisted (laid) ropes

PLroutefind — returns a right angle point list between start and end point

PLsetdiff — more tolerant than setdiff for vectors

PLshaftrad — creates a PL for a contour for SGofCPLrot

PLslottapering — CPL of a slot tapering for creating blades for axis etc

PLsortC — returns a point list that is sorted around its' center

PLsupplement — returns a given contour including supplemented points

PLtangentarc4P — returns center point and circle segment for the two tangents

PLtangentcirc — returns the a circ segment between p1 and p3 with radius R

PLtangentcirc3 — returns the a circ segment between p1 and p3 with radius R

PLtetragrid — returns a tetra grid point list for a bounding box or CPL

PLthreadDIN — Fnct to create the crosssection contour using DIN13 for a thread

PLtoothevolvente — returns the evolvent between base circle and outer circle

PLtransform — transforms ONE PL into another PL

PLtrapezoid — returns a trapezoid shape liek the stern of a slalom water ski

PLtriangle — returns a triangle from an agle

PLtrussoval — creates a symbolic 2D contour of a rod within a truss network to distinguish the animation of a truss from real linkages.

PLtubbing — returns a points list for a 2D tubbing

PLvaryfrom — simply adds small values in a determinstic way to a point list

PLwedgeoncircle — Design a wedge for a circular movement

PLwrench — exactly the same as PLinbus

VLremz — simply reduces the cols of a list to 2

Detailed description

CPLinsertcollinearpoints

inserts collinear auxiliary points that a minimum point distance is guaranteed

In: **CPL**: Point list without nan, open or closed **d**: maximum point distance to guarantee

Out: **CPLN**: Point list with collinear auxiliary points, open or closed as PL

CPLlength

returns the boundary length of a CPL

In: **CPL**: CPL

Out: **len**: len of contour line

CVLinsertcollinearpoints

inserts collinear auxiliary points that a minimum point distance is guaranteed

In: **CVL**: Point list without nan, open or closed **d**: maximum point distance to guarantee

Out: **CVLN**: Point list with collinear auxiliary points, open or closed as PL

GPLdistancetoCPL

returns for a grid point list the point distance insider or outside of a contour

In: **CPL**: Contour to use **GPL**: grid distance or Grid point list **k**: inside outside analysis if existing **x**: x mesh grid **y**: y mesh grid **cdown**: distance to wall or distance to center **cins**: inside or outside contour

Out: **GPL**: Grid point list **k**: distance list; nan for prohibited areas **x**: x mesh grid **y**: y mesh grid

GPLparameters

Auxiliary fnct to calculate xmin ymin dx and dy from a given meshrid or VL

In: **x**: GPL of x of meshgrid **y**: empty or y of meshgrid **z**: empty or y of meshgrid

Out: **orgp**: [minx miny minz] **delt**: [dx dy dz] **ux**: unique of x values usable for meshrid **uy**: unique of y values usable for meshrid **uz**: unique of z values usable for meshrid

isPL

PL has no NaN separator, 1st and last point are different

In: **PL**: Point List, (nx2) or (nxm)

Out: **b**: true if PL has no NaN separator, 1st and last point are different

matrixdistanceofCPL

returns a simple matrix containing the distance to the contour CPL

In: **CPL**: Closed Polygon Line **nrc**: [nx ny] or [dx dy] if [dx,dy]<10 mm

Out: **M**: Matrix containing distance values **GPL**: Grid point list for matrix values **insout**: inside outside values of matrix values **ax**: [xmin xmax ymin ymax]

matrixofCPL

simplified version of matrixdistanceofCPL

In: **CPL**: CPL **nrc**: [nx ny] or [dx dy] if [dx,dy]<10 mm

Out: **M**: Matrix **GPL**: Grid Point Coordinates **ax**: [xmin xmax ymin ymax]

PLaddauxpoints

adds auxiliary points to an open point list

In: **PL**: Open Point list **d**: distance of axuilary points

Out: **PLN**: New Point list

PLaddTetragridofCPL

returns a tetra grid point list for a bounding box or CPL

In: **CPL**: CPL **d**: distance of triangle grid points **w**: wall distance/ default is d/3

Out: **PL**: Point list

PLadjust

returns an adjusted PL related to a position grid and a angle grid

In: **PL**: Point list **pc**: option resolution for points coordinates in mm **pa**: option resolution for angle values in degree (0..180)

Out: **CPL**: Closed Polygon Line

PLarrow

returns a point list for an arrow

In: **l**: length (x=[0..l]) or [A0;B0] Start point and End point **h**: height in mm; default is 10% **s1**: tip size in mm; default is h

Out: **PL**: Point list (open) of arrow

PLbending

bends a CPL on a cylinder or spherical ellipsoid

In: **PL**: Point List **Rx**: Radius Rx **de**: distance/intrusion; default is -0.1 **Ry**: Optional radius Ry; default is 10000 **Rz**: Optional radius Rz; default is Rx **cent**: center solid before bending; default is true

Out: **VL**: Vertex list of the bended point list

PLBezierC

Bezier splines for open point list

In: **PL**: 2D Point list **k**: number of points of the Bezier curve; default is size(PL,1)

Out: **PLN**: 2D Point list

PLcat

returns a concatenated point list of several point list

In: **PL**: One or more points lists separated by comma

Out: **PL**: resulting concatenated point list

PLchamfer2side

two sided version of PLchamfer

In: **z**: z or [z_left z-right] **x**: x or [x_left x-right] **sh**: shape or {shape_left, shape_right} **d**: delta or [delta_left delta-right]

Out: **PL**: Point list for SGofCPLextrude

PLcirclearc

returns an circle arc between two circle points in right hand rule

In: **cp**: center point or radius (+/-) **A**: point A or angle w1 **B**: point B or angle w2 **lh**:

Out: **PLR**: Right hand arc starting at A (red) **PLL**: Left hand arc starting at B (green)

PLcircleofPL

returns minimal and maximal circle around a CPL

In: **CPL**: CPL

Out: **PLCmax**: PLcircle(Rmax) **PLCmin**: PLcircle(Rmin) **rmax**: rmax **rmin**: rmin **cp**: cp

PLcircleovalline

creates staight lines between points extrudes them by using different radii at the points and round edges

In: **RR**: rows of radii for each point of PL **PL**: point list defining an open **rr**: optiona radius for round edges

Out: **CPL**: Final contour

PLcirclesinus

returns a circle-based-sinus shaped point-list

In: **R**: Radius or [Rout Rin] **n**: optional number of points for half circle **L**: Length; default is 100; or [A0 B0] **ns**: optional number of waves; default is 1

Out: **PL**: Point List

PLcircleui

returns a PL for a circle interactively defined

Out: **PL**: Point List

PLclampCTtable

returns the CPL of a clamp for an hybrid-OR CT-table

In: **wal**: [width wallsize radius lowerwallsize]; default is [40 3 0.5 3]

Out: **CPLC**: Contour of the clamp **CPLT**: Contour of the table; not centered **walorg**: the input parameter vector wal as used

PLconnectopenends

closes an open line by a contour

In: **PL**: single open line

Out: **CPL**: Closed contour without doubled entries **PLE**: Externsion may contain start or end point

PLcorrelatingsegments

A new implemenation of PLcorrelate

In: **CPLA**: Contour A, No NAN allowed **CPLB**: Contour B, No NAN allowed **ww**: angle resolution for PLdecompose default is pi/10

Out: **CL**: Ccrrelation list **VL**: combined vertex list of both contours, CPLA at z=0 and CPLB at z=100

PLegg

returns the contour of an egg

In: **r**: radius of evolvente **a**: angle of evolvente

Out: **PL**: Point list

PLfft

same as CPLfft but for open contours

In: **PL**: Open Point List

Out: **PLN**: Frequency limited Contour

PLgearSIN

Based on PLgearDIN, a purely sinusoidal tooth profile

In: **m**: modul **z**: number of teeth **tig**: 0/1 for tott or gap at angle 0 **n**: number of auxiliary points **shr**: shringkage; default is 0.05 on both sides

Out: **PLG**: Point list fo the gear

PLgridhull

returns a convex hull around the given Point list

In: **PCL**: Point cloud / point list

Out: **PL**: Convex hull point list around the points

PLgrow

grows a contour line in a distance to an open or closed PL

In: **PL**: Polygon line **w**: distance for growing **edge**: true call CPLgrowEdge; default is false;

Out: **RPL**: Recalculated CPL **ENL**: Edge normal vector list **PNL**: Point normal vector lit **PDL**: Point direction vector list

PLhexalobular

returns a hexalobular Point list related to DIN ISO 10664:1999-09-01

In: **D**: Outer Diameter **Amin**: optional contour accuracy limit; default is 0.005mm^2 **ncalc**: optional number of points to cal contour; default is at least 100

Out: **PLC**: Point List of Torx/hexalobular shape

PLhook

creates the shape of a hook

In: **h**: height **w**: width **o**: overhang **a**: angle; default is 45 degree **r**: radial edges

Out: **PL**: PL

PLinbus

similar as PLcircle but uses the wrench size

In: **D**: DIAMETER NOT RADIUS! default is 1/4 Inch **n**: number of faces; default is 6

Out: **PL**: Point list **Rm**: Radius of circle for PLcircle

PLinsertcollinearpoints

inserts collinear auxiliary points that a minimum point distance is guaranteed

In: **PL**: Point list without nan, open or closed **d**: maximum point distance to guarantee

Out: **PLN**: Point list with collinear auxiliary points, open or closed as PL

PLinterlock

returns lines or closed polygons to create interlocking surfaces for kinematic DoF

In: **L**: length of the contour in x (if R==0) else in y **R**: Outer and inner Radius for Joints; default is [0 Ro-6] **Pal**: Pin length, angle, nozzel size; default is [1.5 pi/3 0.4] **"debug"**: if used, the individual design steps are shown

Out: **PL1**: repeated contour for length L **PLA**: contour with y below than 0 | or CPLA **PLB**: contour with y above than 0 | or CPLB **CPLA**: basic inner for Radial joints **CPLB**: basic outer for Radial joints **UPLA**: basic lower planar joints **UPLB**: basic upper planar joints

PLjunkerhallpart

creates a profile of a pressed profile

In: **h**: list of sections; default is [5 10 10 5] **t**: radius of impression; default is [0 4] **a**: alpha for impressipm; default i pi/10 **w**: wall thickness; default is r(2)/4 **r**: radius for breaking edges; default is 0.5

Out: **CPL**: CPL of profile to extrude **CPL2**: CPL of profile at the tips

PLknurled

returns a sinus contour on a circle

In: **R**: Radius os sequence of Radius (SGLib 5.4) **n**: nofrd(R) **k**: number of waves **m**: offset to R; default is R/20

Out: **PLK**: CPLof of knureld contour

PLmeancluster

converts a point list into a mean values point list

In: **PL**: point or verteex list [n x m] **res**: resolution four rounding; default is 0.5 (will be changed in future)

Out: **PLM**: Point list with mean values **ia**: index of short list in original list **ib**: index of original list in short list **hi**: Point list with grid rounded points **PLR**: Point list with grid rounded points **nn**: 8-neighbor index matrix of grid points (dead code, not returned in normal use) **EL**: edge list of the neighbor digraph (dead code, not returned in normal use)

PLmeshgrid

simple fnct for a better understand of meshgrid

In: **xx**: list of x values **yy**: list of y values **zz**: list of z values; default is 0

Out: **PL**: Point list of all mesh coordinates

PLminxy

returns the point with minimal x and y values

In: **PL**: Point Liste **maxX**: if true; we search for maximum y and maximum x

Out: **P**: Point with minimal y and minimal x **k**: find index in PL

PLminyx

returns the point with minimal y and x value

In: **PL**: Point Liste **maxY**: if true; we search for maximum y and maximum x

Out: **P**: Point with minimal y and minimal x **k**: find index in PL

PLmirroratline

mirrors a PL or CPL

In: **PL**: PL or CPL **p1**: point 1 on line **p2**: point 2 on line

Out: **MPL**: Mirrored PL

PLofCPLequidistant

returns points on closed or open contours with equal distances

In: **CPL**: SINGLE CLOSED CONTOUR **nr**: bumber of points; default is 4; negative number means minimal distance **isc**: false= open CPL; true=closed CPL; default is true

Out: **PLN**: POINT LIST on Contour with equal distances

PLofFloorPlan

Returns a point list from a distance angle list

In: **FP**: Distance Angle List

Out: **PL**: Point List

PLofimcontourc

returns the centers of pixel coordinates in an image

In: **I**: Image **cent**: true= image origin is in center; default is false **pixs**: size of a pixel; default is [1 1]

Out: **PL**: Coordinate list of contour centers

PLofVL

Converts a planer VL[nx3] into a PL [nx2]

In: **VL**: Vertex list [nx3]

Out: **PL**: Point list [nx2] **T**: Transformation matrix for transformation PL->VL

PLofVLFL

returns the 2D representation for a planar VLFL

In: **VL**: Vertex list [nx3] **FL**: Facet list of a planar surface [nx3] **fi**: selected indices; default is all

Out: **PL**: Point list [nx2] **T**: Transformation matrix **c**: index list to transform PL indices into VL indices **d**: index list to transform VL indices into PL indices

PLofVLplaneprojection

main axis projection of a vertex list along the largest dimension of a normal vector

In: **VL**: Vertex list **nv**: normal vector; default is [0 0 1] ==> xy projection

Out: **PL**: 2D Point list as main axis projection (not true to scale) **nvv**: unit normal vector of the projection plane's main axis

PLofVLprojection

returns image point on a target screen

In: **VL**: Vertex list **pc**: camera center **pt**: target image origin **cup**: camera up center; default is [0 0 1] **sc**: image scaling (pixel size); default is [1 1] **isize**: image size; default is 512 512

Out: **PL**: Point List in image coordinate system **T**: T matrix of image coordinate system **VLT**: Position of the projektion points in VL's system **Tc**: T matrix of camera coordinate system

PLonCPLcontour

findet äquidistante Punkte entlang einer Kontour

In: **PL**: CPL **dist**: distance **remo**: remove overlap; 0== NO REDUCTION; 1 DEFAULT; >1 FACTOR FOR DISTANCE

Out: **PL**: Point List **PLX**: Missing Point list

PLoutercontour

returns a point list that has no crossings

In: **CPL**: Point List **out**: returns the FIRST outer contour if true; if false the first inner; default is true

Out: **PLN**: Single Point List of the FIRST outer contour **PLX**: Singe Point List of the FIRST outer contour

PLpinring

returns the contour of a SGpinring

In: **H**: List of height segments [h1 h2 h3] **R**: [Ri Ro] default is Ro=Ri+slfit(c) **a**: angle; default is pi/180*60 **o**: buffer, oversize for blade shapes; default is 0;

Out: **PL**: Point list to be rotated

PLplotdiscretesample

plots a PL as a digital signal to avoid problem of misinterpretation

In: **PL**: Signal [k x()]

Out: **h**: handle to plot

PLplotdiscretetime

plots a PL as a digital signal to avoid problem of misinterpretation

In: **PL**: Signal [k x()] **col**: color similar to **wth**: with of line **w2**: line width of the stepped (double-sampled) signal; default is 1 **cps**: plot style for the filled area under the signal (CPSplot); default is '' (none)

Out: **h**: handle to plot **PLD**: double-sampled stepped point list of the signal

PLradialEdges

returns a point list (PL) with rounded edges

In: **PL**: Point list nx2 **R**: Radius; default is 1 **cvk**: 0==all; -1=concave, +1=convex **wmin**: minimal abs angle to handle; default is 0 **wmax**: maximum abs angle to handle; default is pi **KL**: nr of sections of a line; 2 are required for radius at 90degree **m**: number of points; default is nofrd

Out: **NPL**: New point list

PLreassign

reassigns the coordinates of a new list to the indices of an original list

In: **PLorg**: Original point list **PLnew**: Final point list **thr**: default is 1e-12;

Out: **b**: index list to convert a new index list into original indices

PLrhombus

Point list of a rhombus

In: **d**: diameter in x; default is [6 6 5 2]); **h**: y value for each diameter; default is [0 3 4 15] **a**: turning angle after design; default is 0 rad

Out: **PL**: Point list of the rhombus

PLrope

creates cross section of twisted (laid) ropes

In: **D**: [Outer Diameter inner Diameter] **v2**: second vector such as [1 0] **fuse**: if true, there are no single cardeel anymore; default is false

Out: **CPL2**: Final Contour

PLroutefind

returns a right angle point list between start and end point

In: **PL**: Original Point List

Out: **NPL**: New Point List

PLsetdiff

more tolerant than setdiff for vectors

In: **CPL**: Point list A **PL**: Point list B **tol**: Tolerance **rem**: if true, NaN separators are removed from the result; default is false

Out: **PLN**: Returns a Point list CPL without PL

PLshaftrad

creates a PL for a contour for SGofCPLrot

In: **r**: [r1 r2 r3 ....] radius **d**: distances between steps **b**: [radius breakrule] breaking the edges; default is [0.5 -1] **ramp**: if true use ramps instead of corners; default is false

Out: **PL**: Point list of a shaft

PLslottapering

CPL of a slot tapering for creating blades for axis etc

In: **D**: Diameter of the axis **H**: Height of the Channel / Blade **sl**: slot size; default is slfit'c' **rm**: [tapering height-block height-ramp];m default is [2*sl 1.5 2] **al**: optional rotation angle

Out: **CPLX**: Contour to cout out

PLsortC

returns a point list that is sorted around its' center

In: **PL**: Original point list **d**: order same angle: 'outer' | 'inner'; default is 'inner'

Out: **NPL**: Sorted point list **WL**: Angle list **DL**: Distance list

PLsupplement

returns a given contour including supplemented points

In: **PL**: Point list with missing points given as [inf inf inf]

Out: **PL**: Point list with interpolated points **nsup**: Number of interpolated / supplemented points

PLtangentarc4P

returns center point and circle segment for the two tangents

In: **PLA**: Start Point and Endpoint of Line A [xs ys; xe ye] **PLB**: Start Point and Endpoint of Line B [xs ys; xe ye]

Out: **PLall**: Radius of the 4 circles (1st is left of PLA(2,:) **PLX**: center points of the 4 circles (1st is left of PLA(2,:) **R**: cross points of B of 4 circles (1st is left of PLA(2,:)

PLtangentcirc

returns the a circ segment between p1 and p3 with radius R

In: **p1**: 1st point; alternatively PL **p2**: 2nd point; alternatively desired radius **p3**: 3rd point;; alternatively maximize radius; **R**: Desired Radius **M**: maximize radius; default is false

Out: **PLS**: Point List to replace 2nd point of the list **pc**: Center Points of the Radius **R**: Radius used in PLtangentcirc

PLtangentcirc3

returns the a circ segment between p1 and p3 with radius R

In: **PL**: three points [p1; p2; p3] **R**: Desired Radius **maxR**: option to maximize the radius

Out: **PLE**: Point List to replace 2nd point of the list **p2x**: Center Points of the Radius **R**: Radius used in PLtangentcirc

PLtetragrid

returns a tetra grid point list for a bounding box or CPL

In: **d**: distance of **bb**: length, b h, bounding box or CPL **w**: wall distance/ default is d/3

Out: **PL**: Point list

PLthreadDIN

Fnct to create the crosssection contour using DIN13 for a thread

In: **D**: DIN diameter; negative values for nuts **P**: optional pitch, otherwise taken from DIN13 **w**: thread flank half-angle in degrees; default is 30 **"nonDIN"**: not used in this functri

Out: **PLB**: CPL for the cross section if the thread **P**: Pitch of the tread **KD**: Core diameter of the core of bold or nut

PLtoothevolvente

returns the evolvent between base circle and outer circle

In: **rb**: base circle = d*cos(20/180*pi) **z**: number of teeth **tig**: outer circle = (z+2)*m **n**: foot circle = df=(z-2.5)*m

Out: **PLT**: Point list of on etooth **PLE**: first flank starting at base circle **PL2**: second flank starting at base circle **PLG**: Complete Contour of the Gear contour

PLtransform

transforms ONE PL into another PL

In: **PLA**: Point list A **PLB**: Point list B **sl**: percentage between 0==PLA and 1==PLB; list of values **CL**: Correlation List, see PLcorrelate

Out: **PL**: CPL ==> or cell list of CPLs if numel (sl)>1 **CL**: Correlation list

PLtrapezoid

returns a trapezoid shape liek the stern of a slalom water ski

In: **b**: base and top default is [60 90] **h**: height of trapezoid; default is 210

Out: **PL**: Point list

PLtriangle

returns a triangle from an agle

In: **a1**: angle; default 1e-2 **a2**: angle; default is 0 **h**: height of the triangle

Out: **PL**: Point list of a single triangle

PLtrussoval

creates a symbolic 2D contour of a rod within a truss network to distinguish the animation of a truss from real linkages.

In: **Roi**: Radius [Rout Rin Rrad D/2]; default is [5 5/3 5*4 sqrt(L)/10] **PAB**: Point List; [Start Point; End Point]

Out: **CPL**: CPLof a simple rod with a joint opening to distinguish mechanisms link contours from truss edge contours

PLtubbing

returns a points list for a 2D tubbing

In: **RI**: Inner radius **RA**: Outer radius **n**: optionl n **phiA**: starting angle **phiB**: ending angle

Out: **PL**: Point list **EL**: Edge list

PLvaryfrom

simply adds small values in a determinstic way to a point list

In: **VL**: point list **dd**: [dx dy dz]; default is [1 1 1]/1234 **rr**: [wx wy wz]; default is [1 1 1]*34/14333

Out: **VLN**: New point list

PLwedgeoncircle

Design a wedge for a circular movement

In: **R**: Radius; default is 20 **h**: [wedge-height-above wedge-below-circle] **w**: angle segment of wedge; default is 2*pi / 16

Out: **PL**: Wedge contour **dx**: x-offset of the wedge flank; equals sin(w/2)*R **dy**: y-offset of the wedge flank; equals cos(w/2)*R

PLwrench

exactly the same as PLinbus

In: **D**: DIAMETER NOT RADIUS! default is 1/4 Inch **n**: number of faces; default is 6

Out: **PL**: Point list **Rm**: Radius of circle for PLcircle

VLremz

simply reduces the cols of a list to 2

In: **VL**: Vertex list with n x 3 rows

Out: **PL**: Point list with n x 2 rows

1.2 Vertex List Manipulation [guide]

Functions in this section:

FLofVLonsphereprojection — returns a facet list that is based on alphashape triangulation of sphere projected vertices

PCfindNearestNeighbors — returns for a pointcloud a list of direct neighbors, distances and areas

PCfindplanes — EXPERIMENT TO SHOW THE PLANES IN AN IMAGE

PCofSG — returns a point cloud from a SG with defined number of points and noise

PCofVL — returns a points cloud including NL from Vertex List

PCplot — plots a point cloud or a cell list of point clouds without any registration

PCwriteXYZ — writes an XYZ File of Vertices and Normals that is used by Meshlab

VLaddz — returns a vertex list 3D (z=0) for a point list (2D)

VLBeziernoose — returns a Point list of a noose as link between 2 points with a desired spring length

VLcat — Time consuming adding two vertex list

VLcorrelate — returns the indices of a list in another list

VLdownsample — returns a downsampled vertex list

VLeigenvect — Calculates Eigenvectors and Eigenvalues of Vertex list

VLexpand — returns outer and inner contour for a given convex hull WORK IN PROGRESS

VLfibonacciSphere — returns n vertices distributed on a sphere with radius r

VLFLselectcrossingfacets — selects the crossing facets of a solid that is crossed by a second one

VLlink2P — returns links of special shapes between two points in 2D.

VLmeshgrid — simple fnct for a better understand of meshgrid

VLnorm — returns the list of all normalized vectors and the norm length for a given vector list

VLofCVL — creates a Contour index list for a CPL/CVL

VLofimsegthres — returns a VL of segmented points within an image

VLreadPLY — reads in a Point Cloud in PLY Format

VLremsuccident — Removes successive identical lines in a list AND opens a PL.

VLsortC — returns a point list that is sorted around its' center

VLunidirect — returns a unidirected vector ist

VLunite2C — returns for 2 contour lines from inside2C the united vertex list

VLvaryfrom — simply adds small values in a determinstic way to a vertex list

Detailed description

FLofVLonsphereprojection

returns a facet list that is based on alphashape triangulation of sphere projected vertices

In: **VL**: Vertex list / Point cloud

Out: **FL**: Facet list **PL**: Point list [AZ EL]

PCfindNearestNeighbors

returns for a pointcloud a list of direct neighbors, distances and areas

In: **pc**: point cloud or vertex list **ind**: indices' default is all **nr**: number of neighbors; default is 3 **al**: minimal angle for triangles; default is zero

Out: **NIL**: Neighbor index list [n x nr] = Facet list for nr==3 **DIL**: Neighbor distance list [n x nr] **OA**: point cloud **AL**: Area list **pc**: point cloud

PCfindplanes

EXPERIMENT TO SHOW THE PLANES IN AN IMAGE

In: **PC**: point cloud object or nx3 vertex list **pdepth**:

PCofSG

returns a point cloud from a SG with defined number of points and noise

In: **SG**: Solid Geometry **nmax**: expected number of points; default is 30000 **vgrid**: grid size; defaults is 1/100 of sofBB; quick response with 0 **vnoise**: vertex noise in percent! 1 == 1/100 of sofBB; default is 0;

Out: **pc**: point cloud **NL**: normal vector list of point cloud

PCofVL

returns a points cloud including NL from Vertex List

In: **VL**: Vertex list

Out: **pc**: point cloud including normals

PCplot

plots a point cloud or a cell list of point clouds without any registration

In: **PC**: cell list of point clouds

Out: **h**: vector of handles to the plotted point clouds

PCwriteXYZ

writes an XYZ File of Vertices and Normals that is used by Meshlab

In: **PC**: Point Cloud or Vertex list **FNAME**: Output file name **ONAME**: Object name

VLaddz

returns a vertex list 3D (z=0) for a point list (2D)

In: **PL**: Point list (2 x n) **t**: z value to append; the last element is used as the z height; default is 0

Out: **VL**: Vertex list (n x 3)

VLBeziernoose

returns a Point list of a noose as link between 2 points with a desired spring length

In: **d**: length in x **h**: half of the height in y **b**: upper neck of the noose **a**: lower neck of the noose **n**: number of points

Out: **VL**: Point list as vertex list **PL**: control point list used to generate the Bezier curve

VLcat

Time consuming adding two vertex list

In: **V1**: Vertex List 1 **V2**: Vertex List 2

Out: **VL**: Combined Vertex list **j**: Number of added vertices **j1**: index in the combined list of the 1st vertex of V2 **j2**: index in the combined list of the 2nd vertex of V2 **j3**: index in the combined list of the 3rd vertex of V2

VLcorrelate

returns the indices of a list in another list

In: **NVL**: Long vertex list to search in; must not be unique, first entry is returned **VL**: Second vertex list to find in NVL **thr**: optional index list to search for

Out: **vi**: index in NVL

VLdownsample

returns a downsampled vertex list

In: **VL**: Vertex list **gs**: grid size; default is 1 percent (100x100x100) = 10000 **ongrid**: 'grid','mean','stable','first','last'

Out: **NVL**: New vertex list **a**: selection index of VL **b**: mapping index of VL

VLeigenvect

Calculates Eigenvectors and Eigenvalues of Vertex list

In: **VL**: Vertex List

Out: **EVL**: row list of eigenvecors ordered in norm of eigenvalue **EWL**: row of Eigenvalue **T**: Transformation matrix

VLexpand

returns outer and inner contour for a given convex hull WORK IN PROGRESS

In: **VL**: Vertex list of a convex hull **s**: Slot distance between convex hull and inner contour **d**: Thickness/Distance between inner and outer contour **nb**: Thickness of the bock **bdx**: size of the outer box (negative = relative) in x **bdy**: size of the outer box (negative = relative) in y

Out: **IL**: Inner contour as vertex list **OL**: Outer contour as vertex list **NL**: Normal vector of the edges **VNL**: Normal vector of the vertices

VLfibonacciSphere

returns n vertices distributed on a sphere with radius r

In: **R**: Radius of the sphere **n**: numbre of points on the sphere

Out: **VL**: Vertext list

VLFLselectcrossingfacets

selects the crossing facets of a solid that is crossed by a second one

In: **SG**: Solid A **T**: point or HT matrix; default==''; manual select **XSG**: cube length, rectangle size, box size or Solid Geoemetry

Out: **CFL**: crossing facets of [SG XSG] **vs**: size vector of the crossing region bounding box **T**: HT matrix of the selected point/frame

VLlink2P

returns links of special shapes between two points in 2D.

In: **p1**: Start point or vertex **p2**: End point or vertex **shape**: 'straight' or 'zigzag' **excent**: size of the eccentric part

Out: **VL**: Vertex list of the shaped line

VLmeshgrid

simple fnct for a better understand of meshgrid

In: **xx**: list of x values **yy**: list of y values **zz**: list of y values;

Out: **VL**: Point list of all mesh coordinates

VLnorm

returns the list of all normalized vectors and the norm length for a given vector list

In: **VL**: Vector list

Out: **NL**: Normal vector list **DL**: Norm length list (Distance)

VLofCVL

creates a Contour index list for a CPL/CVL

In: **CVL**: CLosed Polygon list CPL/CVL

Out: **VL**: Vertex list without doubles and NaNs **CIL**: Contour index list related to VL **EL**: Edge list

VLofimsegthres

returns a VL of segmented points within an image

In: **I**: Logical Image

Out: **VLC**: Vertex List of centers **VLs**: size of features **VL**: All feature poits **S**: index list for features

VLreadPLY

reads in a Point Cloud in PLY Format

In: **FName**: File Name or Empty for interactive selection **mag**: magnification; default is 1000

Out: **VL**: Vertex List; double prec **ptCloud**: Point Cloud; single prec

VLremsuccident

Removes successive identical lines in a list AND opens a PL.

In: **VL**: list of numeric rows

Out: **SVL**: decimated list **ii**: removed indices in original list

VLsortC

returns a point list that is sorted around its' center

In: **VL**: Original point list **d**: order same angle: 'outer' | 'inner'; default is 'inner'

Out: **VL**: Sorted point list **WL**: Angle list **DL**: Distance list

VLunidirect

returns a unidirected vector ist

In: **VL**: Vector list **thr**: number of relevant digits for round; default is 5

Out: **NL**: Reduced Vector list **ri**: removal index list

VLunite2C

returns for 2 contour lines from inside2C the united vertex list

In: **VL**: Vertex contour list 1 as used in inside2C **PL**: Vertex contour list 1 as used in inside2C **CCL**: Crossed Countour list as generated by inside2C **EPL**: Edge processed list as generated by inside2C

Out: **UVL**: United vertex list

VLvaryfrom

simply adds small values in a determinstic way to a vertex list

In: **VL**: Vertex list

Out: **VLN**: New vertex list

1.3 Transformation Matrices [guide]

Functions in this section:

HTofT — returns a 4x4 homogenous transformation matrix even for 2x2

Tofcam — returns HT matrix of camera and target for a given camera point, target point, and camera up vector

TofDPhi — returns a 3x3 HT matrix for 2D link

TofFS — returns a HT matrix for a speficied feature surface

TofPCVL — returns the HT matrix for a planar contour vertex list

TofR — returns a HT matrix for an R matrix

TofVLUL — returns the HT matrix of the origin of an union area!

TofVLULez — returns an HT Matrix of the specified Union

Tupright — rotates an HT matrix to have the z-dimension upright.

Detailed description

HTofT

returns a 4x4 homogenous transformation matrix even for 2x2

In: **T**: transformation matrix (2x2, 3x3, 4x4)

Out: **HT**: transformation matrix (4x4)

Tofcam

returns HT matrix of camera and target for a given camera point, target point, and camera up vector

In: **pc**: point of camera; default is campos **pt**: point of target; default is camtarget **cup**: cup direction ; rough; default is camup **va**: camera view angle in degrees; default is the current axis CameraViewAngle

Out: **T**: Transformation of camera **Tt**: Transformation of target **va**: current camera angle **ax**: current axis values

TofDPhi

returns a 3x3 HT matrix for 2D link

In: **D**: Distance in X **Phi**: rotation around Z

Out: **T**: 3x3 HT matrix (2D)

TofFS

returns a HT matrix for a speficied feature surface

In: **TR**: Solid Geometry **Nr**: Number list or ez vector list of the selected surfaces **fe**: feature edge angle; default 1; use 0.08 for planar surfaces ~ 6 degree **FS**: Optional string selector for center points such as 'R1' **Rz**: Optional Rz rotation angle **dz**: Optional dz translation in

Out: **T**: Frame Matrix **fsi**: feature surface index; scalar if unique or selected **FN**: full feature name list **NN**: full frame list for feature names **Ti**: CVL for found radial contours; sorted **RCVL**: CVL for found radial contours; sorted **FIL**: surface index list from surfacesofSG

TofPCVL

returns the HT matrix for a planar contour vertex list

In: **PCVL**: Planar contour vertex list

Out: **T**: Transformation matrix

TofR

returns a HT matrix for an R matrix

In: **R**: Rotation matrix 2x3 or 3x3 **t**: optional translation vector (x y) or [x y z] **w**: optional torsion angle at the end

Out: **T**: 3x3 or 4x4 transformation matrix

TofVLUL

returns the HT matrix of the origin of an union area!

In: **VL**: Vertex list **UL**: Union list **i**: Size index of union (1st, 2nd, etc)

Out: **T**: HT Matrix, centered **UVL**: Union vertex list **UFL**: Union facet list

TofVLULez

returns an HT Matrix of the specified Union

In: **VL**: Vertex list **UL**: Union list **ez**: desired ez vector **oout**: optional flag to consider only outside facets; default is false

Out: **T**: Transformation matrix **ui**: selected union index **UL**: sorted union list, by size

Tupright

rotates an HT matrix to have the z-dimension upright.

In: **T**: original HT matrix

Out: **T**: correct HT matrix

1.4 CPL as NaN Separated List [guide]

Functions in this section:

BBofCPL — returns the bounding box of a CPL

cellofNaN — converts a CVL/CPL into a cell list

centerCPL — returns the center of all contours of a CPL

CILofCVL — creates a Contour index list for a CPL/CVL

circshiftCPL — returns a circular shifted CPL

CPLaddauxpoints — add supporting points to embedded CPL to guarantee a specified points distance

CPLcircshift — shifts & closes a PL. First point is [minx miny]

CPLcontourc — returns the CPL of matlab's contourc command

CPLconvexseg — returns the segments of convex and concave contours a CPL.

CPLcopypattern — creates patterns by copying CPLs

CPLcopypatternPL — copies a CPL at positions of a PL

CPLcorrelate — correlates the contours from two CPLs

CPLdisttopoint — returns the minimal distance between a contour and a given point

CPLedgeNormal — returns edge normals and point normals for a CPL

CPLfaceplot — plots the faces of a CPL

CPLfillHoneycomb — fills a contour with honeycomb

CPLfillin — creates a contour that fills a another or several contours

CPLfillPattern — fills a contour with copies of a pattern

CPLgrow — grows a contour line in a distance

CPLinsideCPL — returns the contours of a CPL that inside another CPL

CPLinvert — Change the direction of all contours in a CPL

CPLisccw — returns index list for CPL direction

CPLisccwcorrected — returns a CPL with all CPLs in correct orientation cw/ccw order is unchanged !

CPLisccwinout — returns which contour has the right orientation wrt shell and orientation

CPLnonmanifold — true if a point is used twice in the CPL or PL

CPLofBB — returns a CPL around a 2D Bounding Contour BB

CPLofcontourc — converts the contourc format into the CPL format

CPLofimage — returns a point list related to matlab;s contour fnct

CPLofmeshgridrods — returns two grid line templates

CPLofPL — appends the first point to a point list of required

CPLofplateSG — returns for a flat plate solid its contour

CPLofPLEL — returns a polybool contour from PL and EL

CPLofPLFL — returns a polybool contour from PL and each edge of FL

CPLoftext — returns a CPL of one or more textlines separated by \n

CPLofVLFL — creates a CPL and T of a planar VL,FL

CPLorder — returns an xy ordered and shifted CPL

CPLoutercontour — returns for a CPL the outer contour wrt cw/ccw-orientation

CPLplot — plots a 2D polybool contour polygon in a 3D figure

CPLradialEdges — returns a vertex list (VL) with rounded edges (USE CPLisccwcorrected before!)

CPLrecontour — returns a contour list for a sliced/cutted object

CPLremnoise — returns a smoother CPL

CPLsortC — returns a contour that start with angle -pi

CPLsortinout — returns a inside outside sorted CPL (There is a changed behavior

CPLspiral — returns a closed polygon in spiral shape

CPLtemplateofCPL — returns a template for a given CPL

CPLtextimage — returns contour of text string

csgofPL — returns MATLAB's Constructed Solid Geometry (CSG) dg matrix for a list of polygons

CVLofcontourc — converts the contourc format into the CVL format

CVLofVL — converts a CVL in a CVL whose contours are closed

CVLofVLCIL — returns a CPL/CVL from a PL or VL using a contour index list

CVLofVLEL — returns a closed polygon vertex list from a vertex list and an edge list

CVLremovez — removes contours with a defined z value from a CVL

CVLremstraightAmin — removes points with a minal distance/area to a predecessor or successor

CVLseparatez — returns a sliced CVL/CPL for a given z value

delaunayofCPL — more sharp delaunay triangulation in 2D (planar case)

FLofCPL — misleading fnct - use PLFLofCPLdelaunay, PLFLofCPLpoly instead

iscrossingCPL — checks crossing/touching of two CPLs

PLcrossCPLline — return the crossing point of a single CPL and a line

PLofCPL — converts a nested CPL into a nested PL

replaceNaN — replaces or removes/ deletes a contour within a CPL

selectNaN — connects subsets of closed polygons lines (2D/3D) wrt an index list

separateNaN — separates List by NaN rows entries

VLedgeNormal — returs normal and plane vector for free boundary contour

VLremstraightAmin — removes points with a minal distance/area to a predecessor or successor

woNaN — returns a row list without all rows that have a NaN in first row

Detailed description

BBofCPL

returns the bounding box of a CPL

In: **CPL**: Closed Polygon Line **zmax**: z value used when the input is 2D; default is 0 **d**: reserved parameter; default is 0

Out: **bb**: bb [xmin xmax ymin ymax] **cp**: center point of the bounding box

cellofNaN

converts a CVL/CPL into a cell list

In: **CVL**: CVL/CPL separated by NaN

Out: **VLcell**: Cell list of separated PL/CVL

centerCPL

returns the center of all contours of a CPL

In: **CPL**: Closed Polygon list with several PL's

Out: **PL**: List of center of CPL using centerPL **RL**: List of distance to most distant point vom PL

CILofCVL

creates a Contour index list for a CPL/CVL

In: **CPL**: CLosed Polygon list CPL/CVL

Out: **CILC**: Contour index list related to CPL **VL**: Vertex list without doubles and NaNs **CIL**: Contour index list related to VL

circshiftCPL

returns a circular shifted CPL

In: **CPL**: Closed Polygon List **n**: number of circshifts polygon; default is random **pl**: if true; the individul contours are shifted; default is false

Out: **CPL**: resulting CPL

CPLaddauxpoints

add supporting points to embedded CPL to guarantee a specified points distance

In: **CPL**: Closed Polygon List **d**: maximum distance between two points

Out: **CPLN**: New Closed Polygon List

CPLcircshift

shifts & closes a PL. First point is [minx miny]

In: **PL**: Original PL or CPL **cl**: constant length; default is false

Out: **NPL**: Shifted and CPL **k**: shiftindex

CPLcontourc

returns the CPL of matlab's contourc command

In: **Inputparameter**: exactly the parameter same as contourc

Out: **CPL**: Closed polygon list

CPLconvexseg

returns the segments of convex and concave contours a CPL.

In: **CPL**: Single Point List

Out: **con**: array of convex and concave

CPLcopypattern

creates patterns by copying CPLs

In: **CPL**: Original Polygon **number**: copy vector [nx ny] **delta**: distance vector [dx dy]

Out: **CPLN**: New closed polygon line

CPLcopypatternPL

copies a CPL at positions of a PL

In: **CPL**: CPL to copy (nx2) **PL**: PL of relative Positions (nx2) **dw**: optional turing angle or turning array [nx1]; default is 0

Out: **CPLN**: Final CPL [nx2]

CPLcorrelate

correlates the contours from two CPLs

In: **CPLA**: Contour A **CPLB**: Contour B **corall**: if true all contour are paired; default is false

Out: **CLL**: Correlation list [ACi BCi Level PAi PBi] **NCA**: Contours of A not existing in contour B **NCB**: Contours of B not existing in contour A **a**: complete list of used parameter of a [row parent level cx cy] **b**: complete list of used parameter of b [row parent level cx cy]

CPLdisttopoint

returns the minimal distance between a contour and a given point

In: **CPL**: Closed polygon line **p**: point of interest; default is [0 0]

Out: **dmin**: minimal circle radius that touches the contour. negative if outside **ei**: Index of the nearest contour edge **pcc**: closest point on the contour to point p

CPLedgeNormal

returns edge normals and point normals for a CPL

In: **CPL**: Closed Polygon List separated by NaN

Out: **ENL**: edge normal list **PNL**: Point normal list **PDL**: Point distance list (vector to next point) **an**: normalized edge direction unit vectors

CPLfaceplot

plots the faces of a CPL

In: **PL**: Point List or CPL **c**: color **w**: countour width **a**: alpha; default is 0.5

Out: **h**: handle

CPLfillHoneycomb

fills a contour with honeycomb

In: **PLs**: [x y] or closed contour of the border **w**: width of the wall; default is 0.5 **d**: distance between the walls **ww**: width of the border wall; if ww<0 ; ww=distance to pattern **n**: number of walls per cell; default is 6

Out: **CPLc**: Closed contour that can be extruded by SGofCPLz

CPLfillin

creates a contour that fills a another or several contours

In: **CPL**: Closed Polygon List **Amin**: Minimum surface to be interested **Asam**: percentage of maximum area ; default a=90%, 0.9

Out: **iCPL**: inverse polygon **A**: VL and FL of the open surface

CPLfillPattern

fills a contour with copies of a pattern

In: **CPL**: Contour to be filled **CPLA**: Filling pattern to be copied **w**: border thickness of CPL; default is 1 **d**: distance between the pattern; default is w **cut**: cut instead of removal of overlapping pattern; default is false **pc**: 'c' moves the contour into the center; is 'c' on by default

Out: **CPLN**: CPL of resulting contour with punched patterns **XPL**: CPL of punching patterns

CPLgrow

grows a contour line in a distance

In: **CPL**: Closed Polygon line **w**: distance for growing **edge**: true call CPLgrowEdge; default is false; **unid**: "norm", "outw", "hole" , "fitt", for handling growing

Out: **RPL**: Recalculated CPL **ENL**: Edge normal vector list **PNL**: Point normal vector lit **PDL**: Point direction vector list

CPLinsideCPL

returns the contours of a CPL that inside another CPL

In: **CPL**: One outer CPL that defines the border line **CPLs**: CPL with several contours separated by NAN

Out: **ind**: index which contour is inside of CPLO **CPLN**: CPL of all internal CPLs

CPLinvert

Change the direction of all contours in a CPL

In: **CPL**: Closed Polygon List separated by NaN **add**: frame around with distance add; default=0; no frame

Out: **NCPL**: Closed Polygon List reversed direction

CPLisccw

returns index list for CPL direction

In: **CPL**: CPL

Out: **cwi**: counter-clockwise index **nani**: NAN-indices

CPLisccwcorrected

returns a CPL with all CPLs in correct orientation cw/ccw order is unchanged !

In: **CPL**: Original CPL

Out: **CPL**: Corrected CPL **cio**: inside outside index

CPLisccwinout

returns which contour has the right orientation wrt shell and orientation

In: **CPL**: Contour polygon list **chng**: flag; default is false; currently unused

Out: **corr**: true if correct **cio**: level of contour (0=most outwards) **ccw**: ccw direction

CPLnonmanifold

true if a point is used twice in the CPL or PL

In: **CPL**: closed polygon list

Out: **nm**: true if non manifold points

CPLofBB

returns a CPL around a 2D Bounding Contour BB

In: **BB**: Bounding Box or CPL **d**: distance around the bounding box; default is 1

Out: **CPL**: Closed Polygon Liste

CPLofcontourc

converts the contourc format into the CPL format

In: **C**: result of the contourc function **rem**: removes straight lines if true; default is true

Out: **CPL**: Closed polygon list

CPLofimage

returns a point list related to matlab;s contour fnct

In: **I**: image **n**: number of contours **f**: if true create a zero frame around; default is false **bin**: if true, convert the image via imofimsketch first; default is false

Out: **CPL**: Closed Polygon List

CPLofmeshgridrods

returns two grid line templates

In: **X**: List of X values **Y**: List of Y values **d**: optional thickness of CPL; default is 0

Out: **CPL**: Complete list of lines **PLX**: Only X lines **PLY**: Only Y lines **PL**: Point List (unique)

CPLofPL

appends the first point to a point list of required

In: **PL**: point list

Out: **CPL**: closed polygon list

CPLofplateSG

returns for a flat plate solid its contour

In: **SG**: Solid, i.e. a flat plate

Out: **CPL**: Contour of the plate **z**: height of the plate **zmin**: zmin used for SG

CPLofPLEL

returns a polybool contour from PL and EL

In: **PL**: Point list **EL**: Edge list

Out: **CPL**: Contour point list **EL**: reconnected edge list produced by ELofELmesh **CEL**: contour edge index list [start end] for each closed contour **ELu**: list of the open (unclosed) edges that could not be assigned to a contour

CPLofPLFL

returns a polybool contour from PL and each edge of FL

In: **PL**: Point list **FL**: Facet list **bound**: if true; the boundaries are

Out: **CPL**: Contour point list

CPLoftext

returns a CPL of one or more textlines separated by \n

In: **str**: one or more textlines separated by \n **siz**: size in pt **FW**: FontWidth; see text; default is 'normal' **FN**: FontName; see text; default is 'Helevetica'

Out: **CPL**: Closed Polygon Line **I**: image frame captured from the rendered text

CPLofVLFL

creates a CPL and T of a planar VL,FL

In: **VL**: Vertex list **FL**: Facet List **fi**: index of facets; default is 1:end

Out: **CPL**: Closed Contour List **T**: Transformation Matrix **ez**: ez vector

CPLorder

returns an xy ordered and shifted CPL

In: **CPL**: Closed Polygon List (including NaN) **smax**: max if true, min if false; default is false

Out: **NCPL**: Ordered closed polygon list

CPLoutercontour

returns for a CPL the outer contour wrt cw/ccw-orientation

In: **CPL**: Closed Polygon Line; separated using NaN NaN **out**: returns the FIRST outer contour if true; if false the first inner; default is true

Out: **NCPL**: New CPL

CPLplot

plots a 2D polybool contour polygon in a 3D figure

In: **CPL**: Point list of x/y coordinates nx2 **c**: Color and symbol of points **w**: Width of the used line **t**: Size of a cone shaped arrow tip, default is 0 **b**: Size of a cube showing the start point of a contour; default is 0 **f**: Size of a ring showing the end point of a contour; default is 0

Out: **h**: Handle to figure

CPLradialEdges

returns a vertex list (VL) with rounded edges (USE CPLisccwcorrected before!)

In: **CPL**: Vertex list nx3 **R**: Radius; default is none **cvk**: 0==all; +1=concave, -1=convex **wmin**: minimal abs angle to handle; default is 0 **wmax**: maximum abs angle to handle; default is pi **KL**: nr of sections of a line; 2 are required for radius at 90degree; **m**: number of points; default is nofrd

Out: **CPLN**: New closed Polygon list; All points are doubled

CPLrecontour

returns a contour list for a sliced/cutted object

In: **CPL**: Closed polygon list, contains NaN **frm**: true=frame, false=no frame; default is true;

Out: **NPL**: resulting closed polygon list (with our without a frame)

CPLremnoise

returns a smoother CPL

In: **CPL**: CPL **r**: minimal allowed distance between 2 points; 0.05; **maxw**: noise angle; default is pi/2 -1e-5;

Out: **CPLN**: CPLwithout jitter

CPLsortC

returns a contour that start with angle -pi

In: **CPL**: Original CPL **ctype**: 'zero','min','max', 'czero', 'cmin'; default is 'czero' **CA**: Optional Center Point; do not use it

Out: **CPS**: Resorted CPL **WA**: corresponding angle **CA**: Center of CPL **s**: circshift value to return

CPLsortinout

returns a inside outside sorted CPL (There is a changed behavior

In: **CPL**: Original CPL **in1st**: true==inside first; false=outside first; default is true

Out: **cind**: level index related to separatedNaN (CPL) **CC**: Corresponance matrix **NCPL**: New contour list sorted by enclosure order **pi**: parent index list [parent level]

CPLspiral

returns a closed polygon in spiral shape

In: **RI**: Inner radius **RO**: Outer radius **ws**: turning angle; default is 2pi **d**: wall thickness; default is (RO-RI)/2/(wm/(2*pi)+1 **n**: number of points; default is nofrd **m**: number of spokes **chk**: minimum allowed slot width between spiral spokes; errors if smaller; default 0

Out: **PL**: Point list [nx2]

CPLtemplateofCPL

returns a template for a given CPL

In: **CPL**: CPL separated by NaN NaN **pen**: diameter of template **un**: if true; contours are united; default is false;

Out: **CPN**: Resulting drawing-template contour of the CPL

CPLtextimage

returns contour of text string

In: **text**: Text String / Latex **dt**: Pixel size; Default is 0.25 **f**: Font size; Default is 72 **sx**: Size in x; default is 0, i.e. no scaling **sy**: Size in y; default is 0, i.e. no scaling **sf**: Size in z; default is 0, i.e. no scaling **ha**: horizontal alignment: 'l' or 'c' or 'r'

Out: **CPL**: Closed Polygon List **VL**: VertexLlist **EL**: Edge List **FL**: Facet List

csgofPL

returns MATLAB's Constructed Solid Geometry (CSG) dg matrix for a list of polygons

In: **PL**: CPL of cell list of CPLs **sf**: Formula wrt. to A polygon sequence 'A' 'B' 'C' etc.

Out: **csg**: or dg (Geometry description matrix) **bt**: Name-space (boundary) matrix returned by decsg, relating minimal regions to the input segments **gd**: geometry description matrix (columns per polygon)

CVLofcontourc

converts the contourc format into the CVL format

In: **C**: result of the contourc function **rem**: removes straight lines if true; default is false

Out: **CVL**: Closed polygon list [nx3] with the z values for the contour

CVLofVL

converts a CVL in a CVL whose contours are closed

In: **CVL**: CVL with potential open contours

Out: **CVL**: CVL with ensured closed contours

CVLofVLCIL

returns a CPL/CVL from a PL or VL using a contour index list

In: **VL**: Point or Vertex list **CIL**: [startindex endindex; startindex endindex;]

Out: **CVL**: CLosed Polygon List or Closed Vertex List

CVLofVLEL

returns a closed polygon vertex list from a vertex list and an edge list

In: **VL**: Vertex List **EL**: Edge list **rem**: if true; remeshing using CPLofPLEL / ELofELmesh; default is true

Out: **CVL**: closed polygon vertex list

CVLremovez

removes contours with a defined z value from a CVL

In: **CVL**: Contour vertex list **z**: list of z values to remove

Out: **XVL**: New Contour vertex list

CVLremstraightAmin

removes points with a minal distance/area to a predecessor or successor

In: **CVL**: Closed Polygon Line in 2D or 3D **Amin**: Minimal Area between two succeeding points; default is 1e-3;

Out: **NVL**: New Vertex List

CVLseparatez

returns a sliced CVL/CPL for a given z value

In: **CVL**: Slices CVL **z**: desired z value; currently only scalar **thr**: tolerance for finding a z value

Out: **CVLz**: Contours that fulfill the z-value condition; separated by NaN OR zl if isempty(z) **CIL**: List of z Values

delaunayofCPL

more sharp delaunay triangulation in 2D (planar case)

In: **CPLB**: Closed Contour List

Out: **TR3**: delaunayTriangulation **oi**: removed facets from outside **nfi**: outside facets **fi**: fi=isInterior(TR3) **EL**: Edge List of the **CPL**: closed polygon list of the resulting contour boundary

FLofCPL

misleading fnct - use PLFLofCPLdelaunay, PLFLofCPLpoly instead

In: **CPL**: Closed Polygon List

Out: **FL**: Facet List

iscrossingCPL

checks crossing/touching of two CPLs

In: **CPLA**: Closed polygon A **CPLB**: Closed polygon B **touch**: check touch instead of cross; default is true

Out: **c**: true/false **pc**: first detected crossing point

PLcrossCPLline

return the crossing point of a single CPL and a line

In: **CPL**: Closed Polygon List **pa**: line start point **pb**: line end point **touch**: check on touch instead of cross; default is false **full**: return a full crossing point list; default is false

Out: **CP**: Crossing Point **k**: index in PL **k1**: subsequent index in PL **kb**: distance from pa **ka**: distance from pb

PLofCPL

converts a nested CPL into a nested PL

In: **CPL**: Closed polygon list with unclear closing condition

Out: **NPL**: Closed polygon list with open polygons

replaceNaN

replaces or removes/ deletes a contour within a CPL

In: **CPL**: CPL **ci**: index of contour **CPLi**: new contour for contour ci

Out: **CPL**: Resulting CPL

selectNaN

connects subsets of closed polygons lines (2D/3D) wrt an index list

In: **CPL**: CPL / CVL **ci**: index list [

Out: **CPLN**: Subset of CPL or new ordered CPL

separateNaN

separates List by NaN rows entries

In: **VL**: List of nan separated sublists **i**: optional number of sublist

Out: **x**: List i or number of lists if nargin==1 **a**: list of [0 nans size(VL,2)+1]

VLedgeNormal

returs normal and plane vector for free boundary contour

In: **VL**: Vertex list of succeeding vertices **ez**: optional first normal vector; default is normal of facet [1 2 3] **Amin**: Minimal area to ignore; default is 1e-3; 1-7 makes sense also

Out: **NL**: Normal Vector List (red) **OL**: Orthogonal Plane Directed Vector list (blue)

VLremstraightAmin

removes points with a minal distance/area to a predecessor or successor

In: **CVL**: Closed Polygon Line in 2D or 3D **Amin**: Minimal Area between two succeeding points; default is 1e-3;

Out: **NVL**: New Vertex List **ci**: mapping index from original to new

woNaN

returns a row list without all rows that have a NaN in first row

In: **PL**: vector row list containing NaN

Out: **NPL**: vector row list without NaN **ii**: logical index of the kept rows (non-NaN first column)

1.5 Edge List Manipulation (PL/VL) [guide]

Functions in this section:

CELflip — Flips orientation of contour edge list

CELofEL — returns a corrected succeding EL for a single contour

CELofPL — returns single contour edge list for a points list

CILofOEL — returns contour index list for a SORTED edge list of several consecutive open contours

convexEL — returns a sorted list of convex edge/contour segments

ELconnectedgroups — returns edge lists in groups of isolate indices

ELcorrect — finds edges with a wrong direction and turns them

ELdirectVLEL — returns a edge list, in which all contours have the same direction than the first contour in the edge list

ELflip — flips orientation of edge list

ELhistogram — returns a list of nodes in an edge list and how often it was used

ELofCIL — Converts a Contour Index List into an Edge list

ELofcrosspoints2F — returns the Edge Constraint List from a crosspoint List

ELofCVL — returns a simple edge list for a vertex list

ELofELmesh — returns a separated EL consisting of independent meshes and a rest of points and lines

ELofFL — converts a facet list into an edge list (simple)

ELofFLborder — returns the border edges of a open 3D or closed 2D surface

ELofFLborder2 — returns the border edges of a surface

ELofFLpatch — converts a patch (more than triangles) list into an edge list

ELofjointFL — returns a list of joint edges of two facet list

ELofn — returns an edge list for a closed contour (sorted point list) of n points

ELomitedgesVLEL — returns an edge list that omits edges of same direction

ELomitVLEL — returns an edge list that omits edges of same direction of MULTI CONTOUR EDGE LIST

ELorder — reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

ELplot — plots the graph of the given edge list

ELreconnect — resorts a valid but broken and recompiled edge list

ELremove — removes an index from an unsorted edge list

ELremovedoubledconstraints — removes doubles from a constraint edge list

ELselect — return all edges from an edge list start start or end with ONE SINGLE index

ELseparate — reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

ELsort — returns a increasing sorted edge list separated by NaN

ELsortFE — groups and sorts unsorted edge lists

ELswap — flips orientation of edge list

ELunique — returns a unique list of all edges

ELuniqueofFL — returns a unique list of all edges from a facet list

ELunsort — TEST and Development fnct for unsorting edge or facet lists

graphofEL — creates an undirected graph model from an edge list

isinsideEL — returns an index list of all edges that exist in two direction

ismemberFL — returns ismember for all three possibilities of a facet list

minspanedges — RETURNS the minimal spanning tree required for Kirchhoff loops (Mesh equations)

PLELaddauxpoints — add supporting points to a point list and a contour edge list

PLELconcat — concatenates two different PL and EL and connects them

PLELhsort — returns a hierarchy index list for a PL,EL,SIL

PLELmeshpattern — returns mesh pattern for truss structures or pantograph/scissor mechanisms

PLELnorm — returns norm vectors of edges and points of contours

PLELofCPL — converts a polybool contour point list to point list and edge list

PLELofCPLplain — returns a PL and EL and a modified CPL based on the explicit qish of "open" or "closed"

PLELofFeatureEdges — returns sorted closed polygon edge lists of the feature edges of a solid

PLELofFeatureEdges2 — returns sorted closed polygon edge lists of the feature edges of a solid

PLELofPLpair — converts Point Pair List into a point list and edge list

PLELplot — plots a 2D contour by PL and EL

PLELpushin — pushes points of contour B into contour A

PLELradialEdgesofCPL — Creates 2D/3D radial edges in closed contours and open paths

PLELselect — returns only the required points for a given edge list. The edge list is reindexed

PLELtube — returns a 2D point list and a edge list for a cylinder or a tube cross-section

textVLEL — plots descriptors for each edge of an edge list

treeNodesofEL — returns the tree structure from a given edge list

usedinEL — returns a list of unique numbers and the number of use in the list

VLELcat — returns the connection of several VLs and ELs

VLELcat2 — Simple concatenation of two vertex lists and edge lists.

VLELofCVL — converts nan separated succeeding vertex lists into a vertex list and an edge list

VLELselect — returns only the required vertices for a given edge list. The edge list is reindexed

VLELshort — returns a vertex list without double and without entries that are not used by EL

VLELvertexfusion — merges the vertex list to a grid of a defined resolution

VLofcorruptedFL — returns the vertex list indices of corrupted facets

Detailed description

CELflip

Flips orientation of contour edge list

In: **CEL**: Contour edge list

Out: **CEL**: Contour edge list flipped

CELofEL

returns a corrected succeding EL for a single contour

In: **EL**: unsorted but correct edge list

Out: **CEL**: Sorted Edge list of Contour or Line of Nodes **CHN**: Chain [n x 1] with nodes

CELofPL

returns single contour edge list for a points list

In: **PL**: Points list (2D), works also with VL (3D)

Out: **CEL**: Contour edge list

CILofOEL

returns contour index list for a SORTED edge list of several consecutive open contours

In: **EL**: Edge list

Out: **CIL**: Contour index list **ci**: real closed contour index list (without points) **pi**: single point contour index list

convexEL

returns a sorted list of convex edge/contour segments

In: **p1**: p1 of the corresponding convex triangle **p2**: p2 of the corresponding convex triangle **p3**: p3 of the corresponding convex triangle **VL**: Vertex list **EL**: Unsorted edge list

Out: **SEL**: sorted edge list **I**: index list for each new contour segment **o**: number of contour segments

ELconnectedgroups

returns edge lists in groups of isolate indices

In: **EL**: Edge list **vi**: Optional list of interested vertices; default is ''

Out: **NEL**: New Edge list; NOT SORTED ONLY GROUPED **CIL**: Group Index List

ELcorrect

finds edges with a wrong direction and turns them

In: **EL**: Edge list with edges in wrong direction

Out: **EL**: EL **CL**: List of turned edges in the original direction

ELdirectVLEL

returns a edge list, in which all contours have the same direction than the first contour in the edge list

In: **VL**: Vertex list **EL**: Facet list

Out: **ELD**: Directed edge list

ELflip

flips orientation of edge list

In: **EL**: Edge list

Out: **EL**: Edge list flipped

ELhistogram

returns a list of nodes in an edge list and how often it was used

In: **EL**: Edge list

Out: **[nh]**: list of [nodenumber usage]

ELofCIL

Converts a Contour Index List into an Edge list

In: **CIL**: Contour index list [start index end index; start end ,...] **cl**: closing; default is true

Out: **EL**: Edge List **PL**: list of start point indices (first column of the edge list)

ELofcrosspoints2F

returns the Edge Constraint List from a crosspoint List

In: **XL**: List of cross points of triangle A and triangle B (may include PA and PB) **PA**: Three points of triangle A **PB**: Three points of triangle B **PinA**: index list which points of B are inside A **PinB**: index list which points of A are inside B

Out: **EL1**: edge constraint list for triangle A **EL2**: edge constraint list for triangle B

ELofCVL

returns a simple edge list for a vertex list

In: **VL**: Vertex list **close**: 'c' if closed, default is 'c'

Out: **EL**: Edge list of certex list

ELofELmesh

returns a separated EL consisting of independent meshes and a rest of points and lines

In: **EL**: Original edge list **"corr"**:

Out: **NEL**: New edge list (blue) **CIL**: CIL of meshes **ELu**: list of restpoints that are not part of a mesh (red) **corr**: true if the remaining open edge list was corrected

ELofFL

converts a facet list into an edge list (simple)

In: **FL**: Facet list (n x 3)

Out: **EL**: Edge list (n x 2)

ELofFLborder

returns the border edges of a open 3D or closed 2D surface

In: **FL**: Facet list

Out: **EL**: Edge list **CIL**: contour index list **ELF**: list of the two-directional (interior) edges that were removed

ELofFLborder2

returns the border edges of a surface

In: **FL**: Facet list

Out: **EL**: Edge list **SIL**: Surface index list

ELofFLpatch

converts a patch (more than triangles) list into an edge list

In: **FL**: Facet list (n x m) % m>=3

Out: **EL**: Edge list (n x 2)

ELofjointFL

returns a list of joint edges of two facet list

In: **FL1**: Facet List 1 **FL2**: Facet List 2

Out: **[ELiL]**: Edge list plus Facet Index in FL1 and FL2

ELofn

returns an edge list for a closed contour (sorted point list) of n points

In: **n**: number of points

Out: **EL**: Edge list

ELomitedgesVLEL

returns an edge list that omits edges of same direction

In: **VL**: Vertex list **EL**: Edge List, must be sorted

Out: **EL**: New Edge list, maybe shorter

ELomitVLEL

returns an edge list that omits edges of same direction of MULTI CONTOUR EDGE LIST

In: **VL**: Vertex list **EL**: Edge List, must be sorted

Out: **ELN**: New Edge list, maybe shorter

ELorder

reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

In: **EL**: edge list/facet list/tetrahedron list consisting of several solids **jv**: common vertices that define a connection; default is size(EL,2)-1

Out: **ELN**: ordered edge list/facet list/tetrahedron list of several solids **SIL**: Solid index list (start and end-row) in returned list

ELplot

plots the graph of the given edge list

In: **EL**: Edge List

Out: **h**: handle to plot **G**: digraaph of the edge list

ELreconnect

resorts a valid but broken and recompiled edge list

In: **EL**: broken edge list **GPL**: Only for graphical output required if nargout==0

Out: **EL**: Edge list **CIL**: Contour index list **k**: number of changes required

ELremove

removes an index from an unsorted edge list

In: **EL**: original unsorted edge list **indI**: index to be removed, or index list

Out: **ELN**: new unsorted edge list without the index

ELremovedoubledconstraints

removes doubles from a constraint edge list

In: **CL**: Constraint edge list

Out: **CL**: Constraint edge list with no doubles

ELselect

return all edges from an edge list start start or end with ONE SINGLE index

In: **EL**: Edge list **i**: vertex index

Out: **ELi**: Edge list from index. ELi(:,1) is always i

ELseparate

reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

In: **EL**: edge list/facet list/tetrahedron list consisting of several solids **jv**: optional number of common vertices that define a connection; default is size(EL,2)-1

Out: **ELN**: Reordered Edge List **SIL**: Sorted Index List

ELsort

returns a increasing sorted edge list separated by NaN

In: **EIL**: Edge List

Out: **EIL**: Edge List

ELsortFE

groups and sorts unsorted edge lists

In: **FEL**: Unsorted edge list for instance a feature edge list

Out: **EL**: Edge list **CEL**: Contour index list **mi**: mesh index; false = contour, true is mesh

ELswap

flips orientation of edge list

In: **EL**: Edge list

Out: **EL**: Edge list flipped

ELunique

returns a unique list of all edges

In: **ELo**: Edge List or Facet List **remeq**: removes degenerated edges [i i]; default is false;

Out: **EL**: Edge list sorted by first vertex index

ELuniqueofFL

returns a unique list of all edges from a facet list

In: **FL**: Facet list

Out: **EL**: Edge list sorted by first vertex index

ELunsort

TEST and Development fnct for unsorting edge or facet lists

In: **EL**: Original edge list

Out: **EL**: Unsorted edge list;

graphofEL

creates an undirected graph model from an edge list

In: **EL**: Edge list; must have unique entries! **n**: optional number of vertices

Out: **G**: MATLAB's Graph struct using Edges and Nodes

isinsideEL

returns an index list of all edges that exist in two direction

In: **EL**: Edge list

Out: **ind**: index of the inside edges

ismemberFL

returns ismember for all three possibilities of a facet list

In: **FLA**: Facet List A **FLB**: Facet List B

Out: **im**: ismember result

minspanedges

RETURNS the minimal spanning tree required for Kirchhoff loops (Mesh equations)

In: **G**: Graph or Digraph

Out: **EL**: Edge list **T**: Tree **KL**: Chord edge list - edges not contained in the spanning tree, each closing one Kirchhoff loop

PLELaddauxpoints

add supporting points to a point list and a contour edge list

In: **PL**: Point list in 2D (but would work also with a 3D vertex list) **CEL**: Contour edge list **d**: maximum distance between two points

Out: **PL**: Point list in 2D (but would work also with a 3D vertex list) **CEL**: Contour edge list

PLELconcat

concatenates two different PL and EL and connects them

In: **PL1**: Point list 1 **EL1**: Edge list 1 **PL2**: Point list 2 **EL2**: Edge list 2 **lnk**: if true; an additional edge is created to connect PL1 and PL2; default is false **CPL**: to connecting edges should be collision free inside a CPL

Out: **PL**: Point List **EL**: Edge List

PLELhsort

returns a hierarchy index list for a PL,EL,SIL

In: **PL**: Point List **EL**: Edge List **SIL**: Surface Index List

Out: **HIL**: hierarchy index list

PLELmeshpattern

returns mesh pattern for truss structures or pantograph/scissor mechanisms

In: **nxy**: [nx ny]; default is [3 2] **dxy**: [dx dy]; distance between the points, default is [20 25]; **pat**: 'cross' or 'square'; default is 'cross' **dbl**: double edges; true or false; default is false

Out: **PLN**: Point list **ELN**: Edge list

PLELnorm

returns norm vectors of edges and points of contours

In: **PL**: Point list **EL**: Edge list

Out: **PLN**: Point list norm vectors **ELN**: edge list norm vectos

PLELofCPL

converts a polybool contour point list to point list and edge list

In: **CPL**: Polybool contour point list

Out: **PL**: Point list (without NaN) **EL**: Edge list **SIL**: start/end index list of each closed contour within PL/EL

PLELofCPLplain

returns a PL and EL and a modified CPL based on the explicit qish of "open" or "closed"

In: **CPL**: Closed Polygon list including nan separated contours **"open"**: if used, the results describe an open contour **"closed"**: if use, the results describe an closed contour

Out: **NPL**: New Point List **NEL**: New Edge List **NCPL**: New CPL, open or closed as required;

PLELofFeatureEdges

returns sorted closed polygon edge lists of the feature edges of a solid

In: **TR**: surface triangulation for feature edges **alpha**: feature edge angle

Out: **ELFL**: Edge-facet list of the feature edges **SIL**: Selected Index List for

PLELofFeatureEdges2

returns sorted closed polygon edge lists of the feature edges of a solid

In: **TR**: surface triangulation for feature edges **alpha**: feature edge angle

Out: **ELFL**: Edge-facet list of the feature edges **SIL**: Selected Index List for

PLELofPLpair

converts Point Pair List into a point list and edge list

In: **PLpair**: PLpair [n x 4]

Out: **PLN**: Point list - unique and stable **ELN**: Edge list

PLELplot

plots a 2D contour by PL and EL

In: **PL**: Point list of x/y coordinates nx2 **EL**: Edge list **col**: Color and symbol of points **wid**: Width of the used line

Out: **h**: Handles to Lines

PLELpushin

pushes points of contour B into contour A

In: **PLA**: Point list A **ELA**: Edge list of A (right hand order) **PLB**: Point list B to test and shift **thr**: shift value; default = 1e-12

Out: **PLB**: Shifted point list

PLELradialEdgesofCPL

Creates 2D/3D radial edges in closed contours and open paths

In: **CPL**: open or closed point list in 2D or 3D **R**: maximum radius >0 ; default is 1 **"open"**: if used, the result will be an open path not a contour **"closed"**: if used, the result will a closed contour; this is default.

Out: **NPL**: New Point list; with modified start point **NEL**: New edge list; **NCPL**: NEW CPL list; open or closed as defined

PLELselect

returns only the required points for a given edge list. The edge list is reindexed

In: **PL**: Original point list **ELG**: Original edge list

Out: **PL**: Shorter point list **SI**: Sorting index **EL**: Reindexed edge list

PLELtube

returns a 2D point list and a edge list for a cylinder or a tube cross-section

In: **R**: Radius **nf**: number of facets; default is nofrd **wall**: wall thickness, default is 0 **RY**: Radius in Y, default is R

Out: **PL**: Point List (n x 2) **EL**: Edge list (n x 2)

textVLEL

plots descriptors for each edge of an edge list

In: **VL**: Vertex list **EL**: Edge list **c**: color **s**: thickness **nt**: optional selector **lb**: descirptor letter; default is 'E'

Out: **h**: vector of handles to the edge text labels

treeNodesofEL

returns the tree structure from a given edge list

In: **EL**: Edge list

Out: **vi**: list of vertices [vi n] **vc**: vertes use counter [bi ti li] **bi**: branch index **ti**: terminal index **li**: link index

usedinEL

returns a list of unique numbers and the number of use in the list

In: **NEL**: Any kind of list of integers, such as EL, FL

Out: **PIL**: Entry index list [Entry number]

VLELcat

returns the connection of several VLs and ELs

In: **VL1**: Vertex list 1 **EL1**: Vertex list 2

Out: **VL1**: Concatenated vertex list **ELn**: Concatenated edge list **EL1**: corrected edge list EL1 **varargout**: corrected edge list EL2 ... ELn

VLELcat2

Simple concatenation of two vertex lists and edge lists.

In: **V1**: Vertex list 1 **E1**: Edge list 1 **V2**: Vertex list 2 **E2**: Edge list 2

Out: **VL**: [V1;V2] **EL**: [E1;E2+size(V1,1)] **E2**: E2+size(V1,1)

VLELofCVL

converts nan separated succeeding vertex lists into a vertex list and an edge list

In: **CVL**: NAN Separated Vertex List

Out: **VL**: Vertex List (without NaN) **EL**: Edge list

VLELselect

returns only the required vertices for a given edge list. The edge list is reindexed

In: **VL**: Original vertex list **ELG**: Original edge list

Out: **VL**: Shorter vertex list **SI**: Sorting index **EL**: Reindexed edge list **b**: index vector into the original edge list such that SI equals ELG at those positions

VLELshort

returns a vertex list without double and without entries that are not used by EL

In: **VL**: Vertex list **EL**: Edge list

Out: **SVL**: Shortened vertex list **SEL**: Resorted edge list

VLELvertexfusion

merges the vertex list to a grid of a defined resolution

In: **VL**: Vertex list **EL**: Edge list **delta**: Grid distance

Out: **VL**: Shorten vertex list **EL**: Edge list **n**: number of vertices **dn**: number of reduced vertices

VLofcorruptedFL

returns the vertex list indices of corrupted facets

In: **FL**: Facet list

Out: **vi**: list of involved vertex points **fi**: list of corrupted facets **d**: index list [n x 3] describing which vertex of a list is doubled **v1**: vertices at position [1 2] **v2**: vertices at position [2 3 **v3**: vertices at position [3 1]

1.6 Facet List Manipulation (PL/VL) [guide]

Functions in this section:

FLclosecontoursVLCL — returns for a contour list the facet list

FLcoatVLEL — returns the facets to coat contours in different plane to generate a coating surface

FLcontourwallFLn — returns a simple wall contour between the boundaries of 2 identical surfaces

FLcoverVLEL — returns the facets to close contours in a plane to generate a cover surface

FLofcrossingSG — separates the faces of a solid that are relevant for boolean operation with a second body

FLofCVL — returns the faces for a tube-style vertex list

FLofCVLneigborpoints — creating faces between three succeeding points of a contour

FLofDelaunay — returns the convex hull of a vertex list

FLofEL — converts an edge list into a facet list (tricky) EXPERIMENTAL

FLofELinFL — finds one or more edges in a facet list

FLofn — DO NOT USE (MACRO=>SINGLE LINE) Identical to FLofVL (VL) TESSELATION

FLofoppositefacesofVLFL — returns the list of faces that have opposite facets within a bounding box

FLofPLcorrelation — returns a facet list for a given point correspondance list

FLofSGcolor — finds the faces of a specified color

FLofSGz — returns the planar faces for a z value and the faces that have edges for this z value

FLofTetrahedron — returns the triangles of a tetrahedron list

FLofUL — returns a facet list of a union list

FLofVLdelaunay — creates a delaunay tessellation in 2D of a VL with respect to a ez vector;

FLofVLEL — returns the facet list for a vertex list and a planar edge list in 3D

FLofVLEL2 — returns the facet list for a vertex list and a planar edge list in 3D

FLofVLELn — returns the facet list for a vertex list and a planar edge list in 3D

FLofVLFLfi — returns the facets that are attached to the known facets

FLofVLFLficurved — returns the facets that are attached to the known facets

FLofVLFLnearestneighborVL — Facet List that contains vertices that are neighbors of a vertex list

FLofVLFLvi — Facet List that contains entries of a vertex list

FLofVLmeshgrid — creates a facet list for a vertex list that was created from a depth image

FLofVLPLofVLplaneprojection — returns a FL for 2D could also be FLofVLELplanarDelaunay

FLorder — returns a shifted and ordered facet list

FLplaneofVLCiL — returns facets to close an inner and an out contour

FLremdoublevertex — returns the indices of a FL that a double entry vertex index

FLremove2Dh — removes facets from a solid generated with VLFLclose2Dh

FLselect — returns all neighbor facets of ONE SINGLE vi

FLseparate — reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

FLshift — returns a shifted and ordered facet list

FLtest2 — returns a Vertex and Facet list from an Vertex list and Edge list

FLunique — returns a FL without copies of circshifted rows

opponentFL — returns the opponents indices for a facet list

sizePLFL — returns the index of facets with area > 0

sortFL — returns a sorted and circular shifted facet list (see FLsort)

Detailed description

FLclosecontoursVLCL

returns for a contour list the facet list

In: **VL**: Vertex list **AMELS**: All missing edge list sorted **CIL**: Contour index list

Out: **FL**: Facet list

FLcoatVLEL

returns the facets to coat contours in different plane to generate a coating surface

In: **VL**: Vertex list **ELA**: Edge list the bottom contours (z-Vector) **ELB**: Edge list the top contours

Out: **FL**: Facet list of a cover surface related to the direction of ELO **FLA**: Covers the discontinued vessels of A **FLB**: Covers the discontinued vessels of B

FLcontourwallFLn

returns a simple wall contour between the boundaries of 2 identical surfaces

In: **FLA**: Facet list describing opening A **FLB**: Facet list describing opening B **n**: difference number between vertices

Out: **FLW**: Facet List of the wall contour

FLcoverVLEL

returns the facets to close contours in a plane to generate a cover surface

In: **VL**: Vertex list **ELO**: Edge list of an outer contour; the first contour is the outer one **ELI**: Optional inner contours, could also be part of ELO

Out: **FL**: Facet list of a cover surface related to the direction of ELO

FLofcrossingSG

separates the faces of a solid that are relevant for boolean operation with a second body

In: **SG**: Solid **B**: Solid to check interference

Out: **fii**: facet index of SG that intersects B **EL**: free-boundary edge list of the intersecting surface **G**: graph built from the boundary edge list EL

FLofCVL

returns the faces for a tube-style vertex list

In: **VL**: Vertex list **ni**: Optional size list **at**: optional assignment type; default is 'number' **tt**: optional turning type; default is 'none'

Out: **FLB**: Facet list of the walls **FLW**: Facet list of the bottom **FLT**: Facet list of the cover **CiL**: CiL **ni**: size list

FLofCVLneigborpoints

creating faces between three succeeding points of a contour

In: **CVL**: 3D Point list

Out: **FL**: Facet list combining three neigborpoint [i-1 i i+1]

FLofDelaunay

returns the convex hull of a vertex list

In: **VL**: Vertex list (nx2) or (nx3) **vni**: optional index of VL **EL**: optional EL as constraints **force2D**: if true; a 2D tessellation is forced; default is false

Out: **FL**: Facet list to close VL(vni,:) **NVL**: New vertices (Steiner points) added by the triangulation beyond the input VL

FLofEL

converts an edge list into a facet list (tricky) EXPERIMENTAL

In: **EL**: Edge list

Out: **FL**: Facet list **s**: number of separated surfaces found in the edge list

FLofELinFL

finds one or more edges in a facet list

In: **FL**: Facet list **EL**: Edge List

Out: **FL**: Faces that contain the edges **ci**: selection index

FLofn

DO NOT USE (MACRO=>SINGLE LINE) Identical to FLofVL (VL) TESSELATION

In: **n**: number of points

Out: **FL**: Facet list

FLofoppositefacesofVLFL

returns the list of faces that have opposite facets within a bounding box

In: **VL**: Vertex list **FL**: Facet list **dmax**: maximal distance **redf**: number of reduced facet for first opposit faces check. default is 1000

Out: **si**: logical index of facets

FLofPLcorrelation

returns a facet list for a given point correspondance list

In: **CL**: Correspondance list **nA**: number of A points within VL; default is max(CL(:,1) **VL**: optional vertex list for a drawing

Out: **FL**: Facet list created by the correspondance list

FLofSGcolor

finds the faces of a specified color

In: **SG**: Solid Geometry **col**: search color

Out: **fi**: facet index in SG

FLofSGz

returns the planar faces for a z value and the faces that have edges for this z value

In: **SG**: Solid Geometry **z**: z scalar value for slicing; or 'max' or 'min' **tol**: tolerance for z difference; default is 1e-3

Out: **FLi**: index list of planar faces **z**: index list of faces with edges in the z plane **ELi**: Edge list index with 2 in the plane **TN**: HT matrix (frame) of the planar faces in the z plane

FLofTetrahedron

returns the triangles of a tetrahedron list

In: **FL4**: Tetrahedron list from DelaunayTri **allf**: true = all facets, false = outside facets

Out: **FL**: Outer facet list **FLA**: All facets of the tetrahedrons

FLofUL

returns a facet list of a union list

In: **UL**: Union list of ULgenerate **n**: number of selected union

Out: **FL**: Facet list of this union **v**: Normal vector **A**: Overall facet area **IL**: logical index list of the facets belonging to union n

FLofVLdelaunay

creates a delaunay tessellation in 2D of a VL with respect to a ez vector;

In: **VL**: Vertex list **md**: expected distance between vertices; default is mean value **al**: accepted angle deviation from central projection; default is 0.5/pi*180 ~ 22 degree **ez**: default is [0 0 1]; if ez is []; T of VL is used to find the orientation

Out: **FL**: Facet list **md**: expected/mean vertex distance actually used

FLofVLEL

returns the facet list for a vertex list and a planar edge list in 3D

In: **PL**: Vertex list **EL**: Edge list

Out: **FL**: Facet list

FLofVLEL2

returns the facet list for a vertex list and a planar edge list in 3D

In: **VL**: Vertex list **ELU**: Edge list

Out: **FL**: Facet list

FLofVLELn

returns the facet list for a vertex list and a planar edge list in 3D

In: **VL**: Vertex list **EL**: Edge list **nv**: normal vector, has to be correct

Out: **FL**: Facet list

FLofVLFLfi

returns the facets that are attached to the known facets

In: **VL**: Vertex list **FL**: Facet list **fi**: facet indices that are know to belong to the surface

Out: **cfi**: connected facet index

FLofVLFLficurved

returns the facets that are attached to the known facets

In: **VL**: Vertex list **FL**: Facet list **fi**: facet indices that are know to belong to the surface **al**: limiting angle **ms**: monotonic increasing; default is true;

Out: **cfi**: connected facet index

FLofVLFLnearestneighborVL

Facet List that contains vertices that are neighbors of a vertex list

In: **VL**: Vertex list **FL**: Facet List **VLR**: vertex list to find the nearest neighbor for

Out: **NFL**: FL(fi3,:); (GREEN) **fi3**: all faces that have three points of VL(vi,:) **fi2**: all faces that have 2 points of VL(vi,:) **fi1**: all faces that have 1 points of VL(vi,:) **fi0**: all faces that have 0 points of VL(vi,:)

FLofVLFLvi

Facet List that contains entries of a vertex list

In: **VL**: Vertex list **FL**: Facet List **vi**: vertex index list OR logical vi or vertex list for triangulation.nearestNeighbor

Out: **FL**: FL(fi3,:); (GREEN) **fi3**: all faces that have three points of VL(vi,:) - green **fi2**: all faces that have 2 points of VL(vi,:) - yellow **fi1**: all faces that have 1 points of VL(vi,:) - magenta **fi0**: all faces that have 0 points of VL(vi,:) - red

FLofVLmeshgrid

creates a facet list for a vertex list that was created from a depth image

In: **VL**: Vertex list of a surface that was created from a depth image **ss**: size of the Image used to create VL **D**: if s closed surface model is request, D is the socket height

Out: **FL**: Facet list created by connecting 4 neighbar points **VL**: Modified VL, if D is used

FLofVLPLofVLplaneprojection

returns a FL for 2D could also be FLofVLELplanarDelaunay

In: **VL**: Vertex List of a 2D Surface inb 3D Space **EL**: Optional Edge list

Out: **FL**: Facet List **PL**: Projection Point list; Not a real Transofmation

FLorder

returns a shifted and ordered facet list

In: **FL**: Original facet list

Out: **FL**: Final facet list

FLplaneofVLCiL

returns facets to close an inner and an out contour

In: **VL**: Vertex list **indO**: Vertex contour index list that describes the outer contour **indI**: Vertex contour index list that describes the inner contour

Out: **FLc**: Facet list to close the contour

FLremdoublevertex

returns the indices of a FL that a double entry vertex index

In: **FL**: Facet list

Out: **fi**: removal facet index

FLremove2Dh

removes facets from a solid generated with VLFLclose2Dh

In: **FL**: Original facet list **nc**: number of factes of the caps **rml**: list of facets to remove

Out: **[FL]**: New facet list

FLselect

returns all neighbor facets of ONE SINGLE vi

In: **FL**: Facet list **vi**: index of vertex

Out: **FLv**: reduced facet list **fi**: facet index of FL **FLs**: FL with sortet rows

FLseparate

reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

In: **FL**: edge list/facet list/tetrahedron list consisting of several solids **jv**: optional number of common vertices that define a connection; default is size(EL,2)-1

Out: **FLN**: Reordered Facet List **SIL**: Sorted Index List

FLshift

returns a shifted and ordered facet list

In: **FL**: Original facet list

Out: **FL**: Final facet list

FLtest2

returns a Vertex and Facet list from an Vertex list and Edge list

In: **p1**: Point 1 spanning the reference plane **p2**: Point 2 spanning the reference plane **p3**: Point 3 spanning the reference plane **OVL**: Outer contour vertex list **OEL**: Outer contour edge list **IVL**: Inner contour vertex list **IEL**: Inner contour edge list

Out: **VLA**: Vertex list of the outer surface A **FLA**: Facet list of the outer surface A **CLA**: Vertex list of the inner surface B **VLB**: Facet list of the inner surface B **FLB**: facet list of the region outside the constrained edges

FLunique

returns a FL without copies of circshifted rows

In: **FL**: Facet list

Out: **FL**: Facet list with unique rows;

opponentFL

returns the opponents indices for a facet list

In: **FL**: Facet list **fii**: optional index of facet

Out: **OiL**: opponent of vertex i in facet fii

sizePLFL

returns the index of facets with area > 0

In: **PL**: 2D Point list **FL**: Facet list

Out: **si**: index of area > 0 **sf**: facet area (+1= upward, -1 = downward)

sortFL

returns a sorted and circular shifted facet list (see FLsort)

In: **XFL**: Facet list

Out: **XFL**: Sorted facet list

1.7 Tetrahedron List Manipulation [guide]

Functions in this section:

FLfreeboundaryTR4FL — returns the freeBoundary facets of a tetrahedron

FLofTL — returns all facets of a tetrahedron list

SGofVLalphaShape — returns the alphashape of a vertex list

THLofVL — returns the tetrahedron list of a vertex list

TLFLneighbors — returns the vertex indices of a tetrahedron list that are the attachments to th facet list

TLFLneighbors2 — returns the vertex indices of a tetrahedron list that are the attachments to th facet list

TLofFL — converts a planar facet list into a 2.5 tetrahedron list

TLseparate — reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

TR3ofSG — returns the triangulation for a solid gemeotry

TR3ofTR4 — returns the surface triangulation of a tetrahedron

TRcat — returns a concatenation of tetrahedron solids without any checks

TriRepofVLFL — returns a DelaunayTri representation of a solid

TRofCPLz — returns a valid tetrahedron representation of a solid

TRorder — reorders the tetrahedrons to individual solids

TRplot — plots a tetrahedron triangulation similar to SGplot

TRselect — returns a reduced tetrahedron triangulation for a given tetrahedron list

TRunique — returns a tetrahedron triangulation with unique Points

VLELofTR — returns the edge constraints of a triangulation

VLFLofTR — returns the freeboundary surface of a tetrahedron or the surface of a triangulation

VLTLplot — plots a tetrahedron list as one volume or as separated tetrahedrons

Detailed description

FLfreeboundaryTR4FL

returns the freeBoundary facets of a tetrahedron

In: **TR4FL**: Facet List of a tetrahedron triangulation (FLofTL)

Out: **fb**: outside, free boundary facet index list

FLofTL

returns all facets of a tetrahedron list

In: **TL**: Tetrahedron list

Out: **FL**: Facet list (all normal vectors outside)

SGofVLalphaShape

returns the alphashape of a vertex list

In: **VL**: Vertex List **al**: optional alpha radius

Out: **SG**: Solid Geometry of the Alpha Shape **ca**: critical alpha value of the alpha shape

THLofVL

returns the tetrahedron list of a vertex list

In: **VL**: Vertex list

Out: **THL**: Tetrahedron list

TLFLneighbors

returns the vertex indices of a tetrahedron list that are the attachments to th facet list

In: **TL**: tetrahedron list nx4 **FL**: facet list nx3 **fi**: optional selection list for FL; default is 1:end

Out: **VLi**: cell array of the vertex indices attached to the facets **TLi**: Tetrahedron indices attached to the facets

TLFLneighbors2

returns the vertex indices of a tetrahedron list that are the attachments to th facet list

In: **TL**: tetrahedron list nx4 **FL**: facet list nx3 **fi**: optional selection list for FL; default is 1:end

Out: **VLi**: cell array of the vertex indices attached to the facets **TLi**: Tetrahedron indices attached to the facets **NL**: Neighbors list for the tetrahedron list

TLofFL

converts a planar facet list into a 2.5 tetrahedron list

In: **FL**: Facet list **n**: number of points in vertex list

Out: **TL**: Tetrahedron list

TLseparate

reorders the edge list (triangle, tetrahedron) to individual polygons, surfaces, solids

In: **TL**: edge list/facet list/tetrahedron list consisting of several solids **jv**: optional number of common elements that define a connection; default is size(EL,2)-1

Out: **TLN**: Reordered Edge List **SIL**: Sorted Index List

TR3ofSG

returns the triangulation for a solid gemeotry

In: **SG**: SG.VL, SG.FL

Out: **TR**: Triangulation nx3

TR3ofTR4

returns the surface triangulation of a tetrahedron

In: **TR4**: tetrahedron triangulation **rem**: remove unused (internal) vertices; default is false

Out: **TR3**: surface triangulation **VL**: Vertex list **FL**: Facet list **SI**: FL=SI(FL) => maps to TR4.Points

TRcat

returns a concatenation of tetrahedron solids without any checks

In: **TRA**: Triangulation A **TRB**: Triangulation B

Out: **TR**: Concatenation of Triangulation of A and B

TriRepofVLFL

returns a DelaunayTri representation of a solid

In: **VL**: Vertex list **FL**: Facet list **dw**: Feature angle; default pi/6

Out: **dt**: DelaunayTri class object **fe**: Features edges, Optional

TRofCPLz

returns a valid tetrahedron representation of a solid

In: **CPL**: Closed polygon list for base plate **z**: solid height in z-direction

Out: **TR**: tetrahedron representation (triangulation) of the extruded solid

TRorder

reorders the tetrahedrons to individual solids

In: **TR**: tetrahedron triangulation consisting of several solids

Out: **TR**: tetrahedron triangulation consisting of several separated solids **SIL**: Solid index list (start and end-tetrahedron) in ConnectivityList

TRplot

plots a tetrahedron triangulation similar to SGplot

In: **TR**: Tetrahedron **c**: color **fb**: true=freeBoundary [default], false=tetrahedrons,

Out: **varargout**: handles

TRselect

returns a reduced tetrahedron triangulation for a given tetrahedron list

In: **TR**: Tetrahedron triangulation **TL**: Subset of TR.ConnectivityList

Out: **TR**: New tetrahedron triangulation consisting of minimal points for the ConnectivityList subset **SI**: Sorting index **TL**: Reindexed subset of TR.ConnectivityList

TRunique

returns a tetrahedron triangulation with unique Points

In: **TR**: Tetrahedron triangulation

Out: **TR**: Tetrahedron triangulation with unique Points

VLELofTR

returns the edge constraints of a triangulation

In: **TR**: Triangulation (nx4 or nx3 or nx2)

Out: **VL**: Vertex List (Point List) **EL**: Facet List (Constraints) **FL**: Facet List (ConnectivityList or freeBoundary)

VLFLofTR

returns the freeboundary surface of a tetrahedron or the surface of a triangulation

In: **TR**: Triangulation (nx4 or nx3)

Out: **VL**: Vertex List (Point List) **FL**: Facet List (ConnectivityList or freeBoundary)

VLTLplot

plots a tetrahedron list as one volume or as separated tetrahedrons

In: **VL**: Vertex list **TL**: Tetrahedrons list **c**: color **so**: true = single handle, false = separated handles

Out: **h**: handles to graphics object

1.8 VLFL Manipulation [guide]

Functions in this section:

CLODofVLFL — resorts facet list and vertex list prepared for a CLOD generator.

delaunayTriangulation2D — returns a 2D delaunay triangulation for planar surfaces in 3D

delaunayTriangulationONE2D — creates a delaunayTriangulation of a single triangle avoiding inaccuracy problems

FLofPLEL — returns tessellation of a 2D points list and contour edge list

neighborsAngleSurface — returns connected facets with respect to a feature angle

PLFLofCPLpshape — returns a delaunay triangulation based on the 2018th concept of polyshape

PLFLofPLalphashape — returns a facet list from a point list using alphashape

PLFLofVLELdelaunay — closes a planar surface of VLEL by delaunay

SGof2facets — creates a surface consisting of two faces

SGseparate — plots or returns sorted surfaces or one surface of a solid geometry

VLDLBBofVLFL — Fundamental fnct to return for a facet list the vertex coordinates, the edge vectors, and the bounding box of the facets

VLELcorticalis — generates vertex list and edge list to connect two contours

VLELcorticalis2 — generates vertex list and edge list to connect two contours including additional shells

VLFLbar2P — calucates a bar between two points, considering an optional 3rd point to define the x-axis

VLFLboxT — returns a box at a predefined position

VLFLcap — returns a tube covered with a cap

VLFLcat — concatenates two solids and removes double vertexes

VLFLcat2 — appends without any check a vertex list after another and also a facet list

VLFLchar — returns vertex/facet list for some characters

VLFLclose — Closes an contour right hand with an optional second contour

VLFLclose3DC — Solid procedure to generate facets for ONE closed contour

VLFLcloseC — Solid and fast procedure to generate facets for ONE closed contour

VLFLclosure — returns a surface in which the solid can be pushed in from z-direction

VLFLcoatPCVL — returns coat and covers for vessel contours

VLFLcone — returns vertex list and facet list for a cone solid

VLFLconnect2VLEL — returns the facets of the outside wall of a tube between to Contours

VLFLcontourtube — returns VL, FL of a contour profiled tube along a path in space

VLFLcreateblock — returns a solid block model in the x/y plane

VLFLcreatecross — returns a solid cross model in the x/y plane

VLFLcreateframe — returns a solid frame model in the x/y plane

VLFLcreateLogo — creates a 3D solid volume logo with a text string

VLFLcreateshaft — returns a solid shaft model in the x/y plane

VLFLcylinder — calculates vertex list and facet list for a cylinder

VLFLDelaunayVL — Converts or Displays the Tetrahedrons of a Vertex List

VLFLDelaunayVLEL — closes a surface by tessellation of triangles using Delaunay

VLFLdome — return VL and FL of a dome

VLFLdovetail — generates groove and tongue of a dove tail

VLFLellipse1 — returns a vertex list and a facet list for a defined tube geometry

VLFLellipse3 — returns light weight structure of a closed sphere or ellipsoid

VLFLerrordoubledvertex — creates an doubled vertex error by adding an existing vertex of a random facet

VLFLerroropenfaces — creates an open faces error by adding an existing vertex of a random facet

VLFLexpboard — returns vertex/facet list of a test board with textural description

VLFLfaceNormal — returns the normal vector (or NAN) of the facet list and cutted (1e-4) length

VLFLfreeboundary — returns the freeboundary of a solid

VLFLfreeboundaryfacets — returns the facets attached to the freeboundary edges

VLFLfusedoubledvertices — fuses vertices with similar positions, shortens the VL, but the number of facets remains unchanged.

VLFLgear — returns the vertex/facet list for the gear teeth of an internal or external gear

VLFLgearhub4shaft — returns the vertex/facet list of a gear hub for a specific gear shaft according to DIN 5480-1:2006-03)

VLFLgrid8P — returns a vertex/facet list of a grid structure connecting a list of 8 Points

VLFLgrid8P2 — returns a vertex/facet list of a grid structure connecting a list of 8 Points

VLFLgridP — generates a grid object given by the size and grid thickness

VLFLinsertFacetPoints — This fnct retesselates all surfaces of a solid by inserting points into existing facets

VLFLjointR1 — returns vertices/facets of a revolving joint

VLFLmeshdensity — decreases the edge length of the facets for deformation use

VLFLmotorBodySocket — returns the vertex/facet list for a body socket of a servo motor

VLFLmotorRodSocket — returns the vertex/facet list for a rod socket of a servo motor

VLFLofCPLz — returns a 2½D extruded Solid volume from a contour polygon list (CPL)

VLFLofSGT — returns the VL and FL of the surface related to a frame

VLFLpost — returns VL and FL of a hollow post

VLFLpprojectPL — returns the surface of a parallel projection of a contour

VLFLpprojectPLEL — returns the surface of a parallel projection of a contour

VLFLremsurfedgepoints — returns a surface model without edge points and surface points that are inside of a surface

VLFLremsurfpoints — returns a surface model without surface points that are inside of a surface - boundary/edge points are unchanged

VLFLreorder — returns VL and FL with reordered vertices

VLFLring — returns vertex list and facet list of a ring

VLFLrotsymRLELn — returns VL and FL for a torus style solid

VLFLselect — returns only the required vertices for a given facet list. The facet list is reindexed

VLFLseparateEL — separates edge list after two solid intersection

VLFLshort — shortens a vertex list to the vertices that are really used in the facet list

VLFLshort2 — shortens a vertex list to the vertices that are really used in the facet list

VLFLshortopti — returns the vertex list with optimal number of loss less reduced vertices

VLFLspherecurvedCL — returns VL and FL for a sphere curved surface given by a 2D contour point list

VLFLspherecurvedGPL — returns a sphere curved (3D) contoured surface grid (2D)

VLFLspring — returns vertex/facet list for a mechanical spring

VLFLstar — returns a structure of several independent objects that can combine an inner and a outer structure

VLFLstarlink — returns a structure of several independent objects that can combine an inner and a outer structure

VLFLtetra — returns a simple tetrahedron geometry

VLFLtext — returns vertex/facet list for a text string

VLFLtextattachVLUL — returns a string as VLFL list positioned at the center of a specified union

VLFLtextimage — returns vertex/facet list for a text string

VLFLthread — generates a screw's thread

VLFLthreadnut — generates a screw nut's thread

VLFLvertexNormal — returns the vertex normal for the vertices of a solid

VLFLvertexongrid — returns a solid geometry with all vertices set to a grid

VLFLweave — returns the vertex list XVL and facet list XFL of a weaved rectangle

VLFLyarn — calculates vertex list and facet list for a wire-yarn

VLof2VLnonmanifold — returns vertex list and indices of non manifold vertices on TWO DIFFERENT vertex lists

Detailed description

CLODofVLFL

resorts facet list and vertex list prepared for a CLOD generator.

In: **VL**: Original vertex list **FL**: Original facet list

Out: **VL**: Resorted vertex list **FL**: Resorted facet list

delaunayTriangulation2D

returns a 2D delaunay triangulation for planar surfaces in 3D

In: **VL**: Vertex list, Point list in 3D **CL**: Edge List, Constraints **dig**: number of digits for rounding; default is 1e-12

Out: **TR3**: triangulation **CL**: Edge List, constraints **NVL**: true if points list is extended or consolidated (doubled vertex removal) **chng**: Full Triangulation **k**: inInteriorflag **FL**: Flag whether triangles intersect **isInter**: inInteriorflag

delaunayTriangulationONE2D

creates a delaunayTriangulation of a single triangle avoiding inaccuracy problems

In: **PLO**: 2D Point list in order [p1; p2; p3; .....] **CL**: Constraintlist **thr**: threshold for testing; default is 1e-7

Out: **FLN**: Facet List = Connectivity List **CLN**: Edge List - Constraint List **outside**: points of PLO that are outside of edges of the triangle **nearside**: points of PLO that are inside near the edges of the triangle **acc**: barycentric distances of points to edges that fall below the threshold

FLofPLEL

returns tessellation of a 2D points list and contour edge list

In: **PL**: 2D Point list **EL**: Contour edge list

Out: **FLin**: Facet list inside **VL**: Vertex list could be longer **EL**: Edge List all **FLout**: Facet list outside

neighborsAngleSurface

returns connected facets with respect to a feature angle

In: **NAL**: =TR3neighborsAngle(TR3) **NBL**: =neighbors(TR3), also delivered by [NAL,NBL]=TR3neighborsAngle(TR3) **alpha**: feature angle **fi**: starting facet index (can be several)

Out: **fa**: facet index with respect to NAL,NBL of (TR3) **FCL**: Facet connection list; optional

PLFLofCPLpshape

returns a delaunay triangulation based on the 2018th concept of polyshape

In: **CPL**: CPL with enclosed CPLs

Out: **PL**: Point list [n x 2] **FL**: facet lis [n x 3]

PLFLofPLalphashape

returns a facet list from a point list using alphashape

In: **PL**: Original Point cloud **r**: optional radius for alphashape triangulation

Out: **PL**: Point list (doubled entries removed) **FL**: Facet List **CPL**: Closed Polygon List of contour (endpoint=startpoint) **EL**: Edge list of free boundary sorted **CEL**: Contour index list

PLFLofVLELdelaunay

closes a planar surface of VLEL by delaunay

In: **VL**: Vertex list [nx3] **EL**: Edge consraints

Out: **PL**: Point List **FL**: Tesselated facet list **EL**: Same Edge List

SGof2facets

creates a surface consisting of two faces

In: **w1**: [angle 1 length 1] **w2**: [angle 2 length 2] **w3**: angle rotation agains axis

Out: **SG**: solid geometry (VL/FL) of the two facets

SGseparate

plots or returns sorted surfaces or one surface of a solid geometry

In: **SG**: Solid Geometry; if empty **i**: index list of single/selected surface

Out: **SG**: Solid Geometry **SIL**: Surface Index List **n**: number of closed surfaces **BBi**: Cell list of Bounding Boxes of the Solids

VLDLBBofVLFL

Fundamental fnct to return for a facet list the vertex coordinates, the edge vectors, and the bounding box of the facets

In: **VL**: Vertex list **FL**: Facet List **thr**: supplement to bounding boxes; default is 1e-12

Out: **v1**: List of first vertex of facet list: v1=VL(FL(:,1),:); **v2**: List of second vertex of facet list: v2=VL(FL(:,2),:); **v3**: List of thried vertex of facet list: v3=VL(FL(:,3),:); **d1**: d1=v2-v1; **d2**: d2=v3-v2; **d3**: d3=v1-v3; **bb**: bounding box of all facets;

VLELcorticalis

generates vertex list and edge list to connect two contours

In: **IVL**: Inner contour vertex list; It has to be sorted (!) **OVL**: Outer contour vertex list; It has to be sorted (!)

Out: **CPVL**: Crossing point vertex list **CLEL**: Crossing link edge list to connect inner and outer contour

VLELcorticalis2

generates vertex list and edge list to connect two contours including additional shells

In: **IVL**: Inner contour vertex list; It has to be sorted (!) **OVL**: Outer contour vertex list; It has to be sorted (!)

Out: **CPVL**: Crossing point vertex list **CLEL**: Crossing link edge list to connect inner and outer contour

VLFLbar2P

calucates a bar between two points, considering an optional 3rd point to define the x-axis

In: **PL**: List of 2 or 3 Points. 1st point is start point, 2nd point is end point, 3rd point is optional **bx**: desired size in x direction **by**: desired size in y direction **bz**: desired add-on or sub-on before start and after end point.

Out: **VL**: Vertex List **FL**: Facet List

VLFLboxT

returns a box at a predefined position

In: **T**: Transformation **sz**: size in z, default is 1 **sx**: size in x, default is 10 **sy**: size in y, default is sx **dz**: position change in -z, default is 0.01 **R**: radius of hole in the plate to remove SLS-Material

Out: **VL**: Vertex list **FL**: Facet list

VLFLcap

returns a tube covered with a cap

In: **ZP**: Size in Z of the post **ZD**: Size in Z of the dome **RX**: Radius in X **RY**: Radius in Z **wall**: thickness of the wall **nfo**: number of facets of the outer contour **nfi**: number of facets of the inner contour

Out: **VL**: Vertex list of the object **FL**: Facet list of the post plus dome without floor **FLf**: Facet list of the floor

VLFLcat

concatenates two solids and removes double vertexes

In: **VL1**: Vertex list 1 **FL1**: Facet list 1

Out: **VL**: concatenated vertex List **FL**: concatenated facet List

VLFLcat2

appends without any check a vertex list after another and also a facet list

In: **V1**: Vertex list 1 **F1**: Facet list 1 **V2**: Vertex list 2 **F2**: Facet list 2

Out: **VL**: concatenated vertex List **FL**: concatenated facet List

VLFLchar

returns vertex/facet list for some characters

In: **c**: character

Out: **VL**: Vertex list **FL**: Facet list **dx**: Size in x, i.e. [0..x] **dy**: Size in y, i.e. [0..y] **dz**: Size in z, i.e. [-z/2..z/2]

VLFLclose

Closes an contour right hand with an optional second contour

In: **OVL**: Outer contour vertex list (outer, bottom) (3D) **IVL**: Inner contour certex list (inner, top) (3D)

Out: **VL**: Vertex list **FL**: Facet list

VLFLclose3DC

Solid procedure to generate facets for ONE closed contour

In: **VL**: Vertex list or point list with 4 ore more points **h**: optional height or displacement vector; default is 0;

Out: **VL**: Vertex list (resorted) **FL**: Facet list

VLFLcloseC

Solid and fast procedure to generate facets for ONE closed contour

In: **VL**: Vertex list or point list with 4 ore more points

Out: **VL**: Vertex list (resorted) **FL**: Facet list

VLFLclosure

returns a surface in which the solid can be pushed in from z-direction

In: **VL**: Vertex list **FL**: Facet list **mindz**: minimal required distance between to slices; default is 0 **maxdz**: maximal allowed distance between to slices; default is 1 **zint**: Slicing intervall 2 x 1; default is [-inf +inf]

Out: **VLLA**: Vertex list **FLC**: Facet list of the closure surface **ELB**: Edge list of the bottom contour **ELT**: Edge list of the top contour

VLFLcoatPCVL

returns coat and covers for vessel contours

In: **VLi**: Planar Contour Vertex List, separated: VL1, VL2, VL3 etc.

Out: **VLN**: Vertex **FLcoat**: Facet list of the coat without the covers for VLBA, etc. **varargout**: Facet list to cover planar contour VL1, VL2, VL3 etc.

VLFLcone

returns vertex list and facet list for a cone solid

In: **H**: Height **R**: Radius; default is H/2 **nf**: Number of facets, default is 16

Out: **VL**: Vertex list **FL**: Facet list of the cone **FLb**: Facet list of the bottom

VLFLconnect2VLEL

returns the facets of the outside wall of a tube between to Contours

In: **VL1**: Vertex list of contour 1 **EL1**: Edge list of contour 1 **VL2**: Vertex list of contour 2 **EL2**: Edge list of contour 2

Out: **VL**: Vertex list of the connector **FL**: Facet list of the connector **FLCap1**: Cap on side 1 **FLCap2**: Cap on side 2

VLFLcontourtube

returns VL, FL of a contour profiled tube along a path in space

In: **CL**: Contour profile list (n x 2) or VALUE Radius **ELU**: Edge list of CL or VALUE Wall thickness **PL**: Path list (np x 3) for spatial tube movement **WL**: Value list (np x 1) for expanding/narrowing along the path **nf**: number of facets used when CL is given as a radius value; default is 8

Out: **VL**: Vertex list of tube **FL**: Facet list of tube **nv**: Number of vertices of one Contour, Required of CL is generated **EL**: Edge list of the Contour (if not given as input parameter) **FLCapS**: Facet list of start cap **FLCapE**: Facet list of end cap

VLFLcreateblock

returns a solid block model in the x/y plane

In: **X**: Size in X [0..X] **Y**: Size in Y [0..Y] **H**: Size in Z [0..H]

Out: **VL**: Vertex list **FL**: Facet list **CL**: Contour list

VLFLcreatecross

returns a solid cross model in the x/y plane

In: **X**: Size in X [0..X] **Y**: Size in Y [0..Y] **H**: Size in Z [0..H] **b**: bar width

Out: **VL**: Vertex list **FL**: Facet list **CL**: Contour list

VLFLcreateframe

returns a solid frame model in the x/y plane

In: **X**: Size in X [0..X] **Y**: Size in Y [0..Y] **H**: Size in Z [0..H] **b**: bar width

Out: **VL**: Vertex list **FL**: Facet list **CLO**: Contour list **CLI**: Inner contour list of the frame

VLFLcreateLogo

creates a 3D solid volume logo with a text string

In: **VStr**: Text String

Out: **XVL**: Vertex List **XFL**: Face List **I**: Image

VLFLcreateshaft

returns a solid shaft model in the x/y plane

In: **R**: Radius **H**: Size in Z [0..H]

Out: **VL**: Vertex list **FL**: Facet list **CL**: Contour list **f**: number of facets

VLFLcylinder

calculates vertex list and facet list for a cylinder

In: **Z**: Height of the cylinder **R**: Radius of the cylinder **f**: Number of facets of the cylinder **o**: direction (1=outward, -1=inwards) of the facets

Out: **VL**: Vertex list of the cylinder **FL**: Facet list of the cylinder **CL**: contour point list of the base circle [x y 0] of the cylinder **f**: number of facets actually used (input f incremented by 1)

VLFLDelaunayVL

Converts or Displays the Tetrahedrons of a Vertex List

In: **VL**: Original Vertex list

Out: **VL**: Vertex list **FL**: Facet List

VLFLDelaunayVLEL

closes a surface by tessellation of triangles using Delaunay

In: **p1**: Point 1 of outer triangle **p2**: Point 2 of outer triangle **p3**: Point 3 of outer triangle **OVL**: Vertex list **OEL**: Edge list of contour and constraints

Out: **VLA**: Vertex list of closed surface **FLA**: Facet list of closed surface **CLA**: Constraint list **VLB**: Vertex list of removed surface, wrt convex hull **FLB**: Vertex list of removed surface, wrt convex hull

VLFLdome

return VL and FL of a dome

In: **RZ**: Radius in Z (Height) **RX**: Radius in X **RY**: Radius in Y **wall**: Thickness of the dome **nfo**: Number of facets of the outer sphere **nfi**: Number of facets of the inner sphere; default is nfo

Out: **VL**: Vertex list of inner and outer wall **FL**: Facet list of inner and outer wall **FLf**: Facet list of the dome's floor

VLFLdovetail

generates groove and tongue of a dove tail

In: **h**: height or vector **gw**: groove width **slot**: tongue length

Out: **GVL**: Groove's vertex list **GFL**: Groove's facet list **TVL**: Tongue's vertex list **TFL**: Tongue's facet list **PL2**: point list of the groove contour **PL3**: point list of the added tongue base contour

VLFLellipse1

returns a vertex list and a facet list for a defined tube geometry

In: **Z**: Height; from -Z/2 to +Z/2; center is [0;0;0] **Rx**: Radius in x direction **Ry**: Radius in y direction **dx**: Wall thickness **f**: Number of facets of the polygon, f=4 would be mean a cube shape

Out: **VL**: Vertex list of the tube **FL**: Facet list of the tube

VLFLellipse3

returns light weight structure of a closed sphere or ellipsoid

In: **rx**: Radius in X of the sphere **ry**: Radius in Y of the sphere **rz**: Radius in Z of the sphere **dr**: Thickness of the sphere **nc**: Number of Pattern, default is **na**: Number of closed planes **nb**: Number of open planes

Out: **VL**: Vertex list of the overall object **FL**: Facet list of the overall object

VLFLerrordoubledvertex

creates an doubled vertex error by adding an existing vertex of a random facet

In: **VL**: Vertex List **FL**: Facet List

Out: **VL**: vertex list with an additional/doubled entry **FL**: facet list indexing the doubled entry

VLFLerroropenfaces

creates an open faces error by adding an existing vertex of a random facet

In: **VL**: Vertex List **FL**: Facet List

Out: **VL**: Unchanged Vertex List **FL**: Facet list with a surface missing

VLFLexpboard

returns vertex/facet list of a test board with textural description

In: **text**: **dx**: size of the board in x [0..x]; default dx=50 **dy**: size of the board in y [0..y]; default dy=dx **dz**: size of the board in z [0..-z]; default z=2; minimum is 1 **nx**: distance between bars in x; default ist 0 **ny**: distance between bars in y; default ist nx

Out: **VL**: Vertex list **FL**: Facet list

VLFLfaceNormal

returns the normal vector (or NAN) of the facet list and cutted (1e-4) length

In: **VL**: Vertex list n x 3 **FL**: Facet list **fi**: optional value; facet index (list) **th**: Threshold below which a facet is treated as degenerate (normal returned as NaN)

Out: **NL**: Normal vector list **AL**: Area list;

VLFLfreeboundary

returns the freeboundary of a solid

In: **VL**: Vertex list or Solid Geometry **FL**: Facet List or empty

Out: **EL**: Edge list

VLFLfreeboundaryfacets

returns the facets attached to the freeboundary edges

In: **VL**: Vertex list **FL**: Facet list **thr**: threshold for VLFLshort2; default is 0;

Out: **FLb**: facets attached to the freeboundary edges **EL**: freeboundary edges **FLx**: Faces not attached to boundary edges **fbf**: face index of Flb

VLFLfusedoubledvertices

fuses vertices with similar positions, shortens the VL, but the number of facets remains unchanged.

In: **VL**: Vertex list **FL**: Facet list **thr**: Threshold für round(VL,thr); default is 1e-17 DO NOT USE DIGITS

Out: **NVL**: New vertex list **NFL**: New facet list

VLFLgear

returns the vertex/facet list for the gear teeth of an internal or external gear

In: **H**: Height of the gear, default is 3 **R**: Radius of the gear (Outer/internal, and Inner/external gears) **n**: Number of teeth, default is 20 **tl**: Tooth length (negative/internal gear, positive/external gear) **fw**: Tooth foot width **tw**: Tooth tip width, default is fw

Out: **VL**: Vertex list of the gear teeth **FL**: Facet list of the gear teeth

VLFLgearhub4shaft

returns the vertex/facet list of a gear hub for a specific gear shaft according to DIN 5480-1:2006-03)

In: **hub**: **norm**: Desired outer diameter of the gear hub cylinder; a negative value is interpreted as the wall thickness

Out: **VL**: Vertex list **FL**: Facet list **CPL**: closed polygon list of the gear hub cross section

VLFLgrid8P

returns a vertex/facet list of a grid structure connecting a list of 8 Points

In: **PL**: List of 8 Points of a cuboid. **b**: Thickness of the solid grid structure

Out: **VL**: Vertex list of the grid **FL**: Facet list of the grid

VLFLgrid8P2

returns a vertex/facet list of a grid structure connecting a list of 8 Points

In: **PL**: List of 8 Points of a cuboid. **b**: Thickness of the solid grid structure

Out: **VL**: Vertex list of the grid **FL**: Facet list of the grid

VLFLgridP

generates a grid object given by the size and grid thickness

In: **P**: Point to define maximum values: P=[X;Y;Z] **b**: thickness of the grid bars (negative means in addition)

Out: **VL**: Vertex list of the grid object **FL**: Facet list of the grid object

VLFLinsertFacetPoints

This fnct retesselates all surfaces of a solid by inserting points into existing facets

In: **VLA**: Solid A **FLA**: Solid B **NPL**: New Point List created by SGintersectFacetPoints

Out: **XVL**: Original point list plus additional points **XFIL**: Facet index list of original facets that were changed **NFL**: Facets that belong completely to crossing points (red) **SFL**: Facets that connect untouched points and crossing points (yellow) **UFL**: Facets that are not touched (green)

VLFLjointR1

returns vertices/facets of a revolving joint

In: **R1H**: Height of the overall link **R1A**: Outer Diameter of the inner axis **R1I**: Inner Diameter of the inner axis **s**: distance between inner and outer axis **R2H**: Height of the outer axis; Default is R1H/3 **hd**: Height of the block; default is 1 **sd**: distance between block and outer axis; default is s **n**: optional flag passed to VLFLellipse controlling the ring caps; default is 0

Out: **VL**: Vertex list of object **FL**: Facet list of object

VLFLmeshdensity

decreases the edge length of the facets for deformation use

In: **VL**: Vertex list **FL**: facet list **maxd**: Maximal edge length

Out: **NVL**: New vertex list, contains doubles **UFL**: New facet list **NFL**: is always empty until procedure is not interrupted

VLFLmotorBodySocket

returns the vertex/facet list for a body socket of a servo motor

In: **sl**: length of the motor body in x; default is 32 (FUTUBA S3156) **sb**: width of the motor body in y; default is 11 (FUTUBA S3156) **ax**: position of the motor axis in x; default is 10 (FUTUBA S3156) **bb**: width of the socket blocks; default is 4 mm **bh**: height of the socket blocks; default is 2 mm **wh**: height of the motor body socket housing; default is 10 mm **wb**: width of the motor body sockethousing; default is 1 mm

Out: **VL**: Vertex list of the body socket of a servomotor **FL**: Facet list of the body socket of a servomotor

VLFLmotorRodSocket

returns the vertex/facet list for a rod socket of a servo motor

In: **ll**: length of the rod; default is 10 (FUTUBA S3156) **bl**: width of the rob; default is 4 (FUTUBA S3156) **az**: height of the rod top in z; default is 12 (FUTUBA S3156) **bb**: width of the socket walls; default is 1 mm **bh**: height of the socket walls; default is bl/2 **ml**: distance between center and start of socket walls; default is bl **ch**: height of the socket cover; default is 1 mm **cb**: width of the socket cover; default is 2*bb

Out: **VL**: Vertex list of the rod socket of a servomotor **FL**: Facet list of the rod socket of a servomotor

VLFLofCPLz

returns a 2½D extruded Solid volume from a contour polygon list (CPL)

In: **CPL**: Contour polygon list (CPL) **z**: New in SG-Lib Rel 4.7: height z can be scalar or [zmin zmax]

Out: **VL**: Vertex list **FL**: Facet list **PL**: Point list of the base contour **EL**: Edge list of the base contour **FLW**: Facet list of the wall **FLU**: Facet list of the floor **FLO**: Facet list of the ceiling

VLFLofSGT

returns the VL and FL of the surface related to a frame

In: **SGN**: Solid with Frames **N**: Name of Frames

Out: **VL**: Vertex list; shortend **FL**: FAcet list; shortend

VLFLpost

returns VL and FL of a hollow post

In: **h**: Height in Z **RX**: Radius in X **RY**: Radius in Y **wall**: wall thickness **nfo**: number of facets of the outer ring; default is nofrd(max(RX,RY)) **nfi**: number of facets of the inner ring; default is nfo

Out: **VL**: Vertex list of the post **FL**: Facet list of the post **FLf**: Facet list of the floor **FLc**: Facet list of the ceiling

VLFLpprojectPL

returns the surface of a parallel projection of a contour

In: **PL**: Point list to describe a contour **ez**: direction of the parallel projection **VL**: Vertex list of a solid **FL**: Facet list of a solid **diffuse**: optional offset added along the projection direction; default is 0

Out: **NVL**: Vertex list of the surface (PLE; PL) **NFL**: Facet list of the surface **PLP**: projected contour points on the solid; inf where no intersection was found **PLE**: supplemented projected contour points on the solid

VLFLpprojectPLEL

returns the surface of a parallel projection of a contour

In: **PL**: Point list to describe a contour **ELU**: Edge list of the projected contour **ez**: direction of the parallel projection **VL**: Vertex list of a solid **FL**: Facet list of a solid **diffuse**: length to intrude into the body

Out: **VL**: Vertex list of the **FL**: Facet list **PLP**: Supplemented contour **PLE**: Supplemented and recalculated contour **FLCapO**: Start cap to close the FL on the start contour side **FLCapA**: End cap to close the FL on the end contour side

VLFLremsurfedgepoints

returns a surface model without edge points and surface points that are inside of a surface

In: **VL**: Vertex list of surface model **FL**: Facet list of surface model **alpha**: alpha angle for feature surface; default is 0.01 **minf**: minimum facets per face; default is 4

Out: **VLN**: New vertex list surface model with reduced vertices **FLN**: New facet list surface model with reduced vertices **FLNorg**: Reduced Facet list based on the original vertex list **FSi**: Feature surface index list

VLFLremsurfpoints

returns a surface model without surface points that are inside of a surface - boundary/edge points are unchanged

In: **VL**: Vertex list of surface model **FL**: Facet list of surface model

Out: **VLN**: New vertex list surface model with reduced vertices **FLN**: New facet list surface model with reduced faces **FLNorg**: Reduced Facet list based on the original vertex list **FSi**: Feature surface index list

VLFLreorder

returns VL and FL with reordered vertices

In: **VL**: Vertex list [n x 3] **FL**: Facet list (mx3) **nvi**: new vertex index list (same length as VL) (nx1)

Out: **NVL**: Resulting vertex list [nx3] **NFL**: Facet list using the new numbers

VLFLring

returns vertex list and facet list of a ring

In: **r**: Radius of the ring; default is 10 **n**: number of facets; default is nofrd **h**: height of the ring **w**: width of the ring; default is r/2

Out: **VL**: Vertex list of the ring **FL**: Facet list of the ring

VLFLrotsymRLELn

returns VL and FL for a torus style solid

In: **RL**: Point list 2D **CEL**: Contour edge list **n**: number of point on the polygon; default is nofrd

Out: **VL**: Vertex list of torus solid **FL**: Facet list of torus solid

VLFLselect

returns only the required vertices for a given facet list. The facet list is reindexed

In: **VL**: Original vertex list **FLG**: Original facet list

Out: **VL**: Shorter vertex list **SI**: Sorting index **FL**: Reindexed facet list **b**: index b **c**: third output of unique on the facet list (indices mapping original entries to unique vertices)

VLFLseparateEL

separates edge list after two solid intersection

In: **AVL**: Vertex list of outside contour of solid A **AEL**: Edge list of outside contour of solid A **XVL**: Vertex list of outside contour of intersecting solid B **XEL**: Vertex list of outside contour of intersecting solid B

Out: **AVL**: Complete vertex list of Solid A Facet **BEL**: Cutted outside contour of Solid A Facet

VLFLshort

shortens a vertex list to the vertices that are really used in the facet list

In: **VL**: Original vertex list **FL**: Original facet list

Out: **SVL**: Shortend vertex list **SFL**: Shortend and sorted facet list

VLFLshort2

shortens a vertex list to the vertices that are really used in the facet list

In: **OVL**: Original vertex list **FL**: Original facet list **thr**: optional threshold, default is 1e-12

Out: **VL**: Shortend vertex list **FL**: Shortend and sorted facet list **n**: Number of removed (fused) vertices

VLFLshortopti

returns the vertex list with optimal number of loss less reduced vertices

In: **VL**: Vertex List **FL**: Facet List **otyp**: optimization type: 'vertex' or 'boundary'

Out: **NVL**: SG.VL, SG.FL with optimal number of loss less reduced vertices **NFL**: New Facet List

VLFLspherecurvedCL

returns VL and FL for a sphere curved surface given by a 2D contour point list

In: **CL**: Points list of 2D contour **Rx**: Radius [-R 0 0] **Ry**: sphere radius in the y direction; default equals Rx

Out: **VL**: Vertex list of surface **FL**: Facet list of surface **CVL**: Contour Vertex list **CL**: Contour list (incl. auxiliary points)

VLFLspherecurvedGPL

returns a sphere curved (3D) contoured surface grid (2D)

In: **GPL**: Grid point list in 2D (x z) **R**: Radius (-x) **CEL**: Contour edge list rel. to GPL

Out: **VL**: Vertex list **FL**: Facet list **CL**: Contour vertex list

VLFLspring

returns vertex/facet list for a mechanical spring

In: **r**: Radius of the spring center; default is 5 **h**: Height of the spring (0-max); default is 50 **n**: number of full turns; default is h/r **m**: number of springs integrated; default is 3 **dx**: width of the spring cable; default is 2 **dy**: height of the spring cable; default is dx **l**: height of the base and top plate; default is dy

Out: **VL**: Vertex list of the spring **FL**: Facet list of the spring **sx**: Radius in x-direction **sy**: Radius in y-direction **sz**: Size in z--direction

VLFLstar

returns a structure of several independent objects that can combine an inner and a outer structure

In: **Ro**: Outer Diameter **Ri**: Innerer Diameter **bx**: Thickness in x-direction **by**: Thickness in y-direction **f**: Number of facets

Out: **VL**: Vertex list of the strutures **FL**: Facet list of the strutures

VLFLstarlink

returns a structure of several independent objects that can combine an inner and a outer structure

In: **Ro**: Maximum outer radius **Ri**: Innerer Diameter **bx**: Thickness in x-direction **f**: Number of links

Out: **VL**: Vertex list of the strutures **FL**: Facet list of the strutures

VLFLtetra

returns a simple tetrahedron geometry

In: **a**: side length of the tetrahedron base

Out: **VL**: Vertex list of the tetrahedron **FL**: Facet list of the tetrahedron

VLFLtext

returns vertex/facet list for a text string

In: **text**: String consisting of characters 'TL:-MXVI' **dt**: Distance between two chars; default is 0.5 **t**: Thickness of the base plate [0..-t]; default is 1 **sx**: Size in x; default is 0, i.e. no scaling **sy**: Size in y; default is 0, i.e. no scaling **sz**: Size in z; default is 0, i.e. no scaling

Out: **VL**: Vertex list **FL**: Facet List **dx**: Size in x [0..x] **dy**: Size in y [0..y] **dz**: Size in x [0..z]

VLFLtextattachVLUL

returns a string as VLFL list positioned at the center of a specified union

In: **VL**: Vertex list **UL**: Union list **ez**: desired ez vector/rotation [x y z w] optional w for rotation **atext**: desired text string **sx**: desired size in x **sy**: desired size in y **ha**: Horizontal alignment 'l', 'c', 'r' for multi line **pos**: optional override position [x y z] for the text center; default is [nan nan nan]

Out: **TVL**: Text vertex list **TFL**: text facet list **ui**: union index of the selected union **T**: Center Frame for Text position

VLFLtextimage

returns vertex/facet list for a text string

In: **text**: String consisting of characters 'TL:-MXVI' **dt**: Pixel size **t**: Thickness of the base plate [0..-t]; default is 1 **sx**: Size in x; default is 0, i.e. no scaling **sy**: Size in y; default is 0, i.e. no scaling **sz**: Size in z; default is 0, i.e. no scaling **ha**: horizontal alignment 'l', 'c', or 'r'; default is 'l' **FN**: font name; default is 'default'

Out: **VL**: Vertex list **FL**: Facet List **dx**: Size in x [0..x] **dy**: Size in y [0..y] **dz**: Size in x [0..z] **CPL**: contour point list of the text from VLofimage

VLFLthread

generates a screw's thread

In: **H**: Height of the thread **D**: Outer diameter **d**: Core diameter **di**: Inner diameter **h**: Thread pitch **alpha**: Thread angle; default is pi/3 (60 Degree)

Out: **VL**: Vertex list **FL**: Facet list

VLFLthreadnut

generates a screw nut's thread

In: **H**: Height of the thread **D**: Outer diameter **d**: Core diameter **di**: Inner diameter **h**: Thread pitch **alpha**: Thread angle; default is pi/3 (60 Degree)

Out: **VL**: Vertex list **FL**: Facet list

VLFLvertexNormal

returns the vertex normal for the vertices of a solid

In: **VL**: Vertex list **FL**: Facet list **vi**: optional vertex index; default is all

Out: **NVL**: Normal Vector List [nf x 3]

VLFLvertexongrid

returns a solid geometry with all vertices set to a grid

In: **VL**: Vertex List or Solid Geometry **FL**: Facet List, or empty if VL=SG **grid1**: default is 1e-3 **grid2**: default is 0.25*grid 1

Out: **VLN**: Final Vertex List or Final Solid if nargout==1 **FLN**: Final Facet List

VLFLweave

returns the vertex list XVL and facet list XFL of a weaved rectangle

In: **x**: Size of the Tissue in x-direction (-x/2 .. x/2) **y**: Size of the Tissue in y-direction (-y/2 .. y/2) **n**: Number if notches **m**: Grid Ratio (1) **h**: Thickness in y-direction (VLFLyarn) **b**: Thickness in z-direction (VLFLyarn) **s**: Slot distance (VLFLyarn) **f**: Thickness of the enclosing frame if f>0

Out: **XVL**: Vertex list of a textile-tissue structure as a combination of crossed wire-yarns **XFL**: Facet list of a textile-tissue structure as a combination of crossed wire-yarns

VLFLyarn

calculates vertex list and facet list for a wire-yarn

In: **l**: length of the yarn in x-direction **n**: number of notches (1 is up, 2 us up and down, if u=1) **ld**: length of the notches (half at the begining and end) **b**: Thickness of the wire-yarn in z-Axis **h**: Thickness of the wire-yarn in y-Axis **s**: Additional Slot Distance beside h/2 (required depending on the crossing wires) can be negative **u**: direction of notches (1 starts in +y-direction, -1 starts in -y direction)

Out: **VL**: Vertex list of the wire-yarn **FL**: Facet list of the wire-yarn

VLof2VLnonmanifold

returns vertex list and indices of non manifold vertices on TWO DIFFERENT vertex lists

In: **VLA**: Vertex list A **VLB**: Vertex list B

Out: **NML**: Non manifold vertices; identical points used in both solids **nma**: Indices of the non-manifold vertices within VLA **nmb**: Indices of the non-manifold vertices within VLB

1.9 SG Manipulation [guide]

Functions in this section:

SGCaddSG — adds a solid geometry to a container

SGCaddSGn — returns a SG container with n copies of a solid

SGCaddVLFL — adds a solid geometry to a container

SGcat2 — returns concatenation of two or more solids or sets of solids

SGorder — reorders the triangle facets to individual solids

SGshort — returns one vertex list and one faces list for nested SG

SGsize — returns the minimal and maximal value in x, y, z of a SG

SGstripfields — remove fields in SG structs if they are empty, i.e. unused

VLofSG — returns for nested cell solids the vertex list

Detailed description

SGCaddSG

adds a solid geometry to a container

In: **C**: Solid geometry container

Out: **C**: Solid geometry container

SGCaddSGn

returns a SG container with n copies of a solid

In: **C**: Solid geometry container **SG**: Solid Geometry (.VL, .FL) **n**: number of copies

Out: **C**: Solid geometry container

SGCaddVLFL

adds a solid geometry to a container

In: **C**: Solid geometry container **VL**: Vertex list of solid geometry **FL**: Facet list of solid geometry

Out: **SGC**: Solid geometry container

SGcat2

returns concatenation of two or more solids or sets of solids

In: **FT**: Solid A consisting of VL and FL and FC

Out: **FT**: Solid of consisting of VL and FL of Solid A and B

SGorder

reorders the triangle facets to individual solids

In: **SG**: Solid geometry whose facet list consisting of several solids

Out: **SG**: Solid geometry whose facet list consisting of several solids **SIL**: Solid index list (start and end-facet) in facet list

SGshort

returns one vertex list and one faces list for nested SG

In: **SG**: Solid geometry potentially nested **thr**: rounding threshold **sil**: false=show result; true=no comments

Out: **SG**: Vertex list **n**: Facet list

SGsize

returns the minimal and maximal value in x, y, z of a SG

In: **SG**: Solid Geometry, or Cell list of SGs, or separated by comma

Out: **m**: 1 x 6 array (xmin xmax ymin ymax zmin zmax]

SGstripfields

remove fields in SG structs if they are empty, i.e. unused

In: **SG**: Solid Geometry

Out: **SG**: Solid Geometry

VLofSG

returns for nested cell solids the vertex list

In: **SG**: NEsted Solid Geometry; cells of cells of solids

Out: **VL**: Vertex list

2 Analytical Geometry [guide]

2.1 Graphical User Interface [guide]

Functions in this section:

app_2012_11_09 — Application for opening and viewing STL Files

apptogglelighttrans — toggles the VLFLplotligh status from facets to light

axis4math — sets axis for mathematical fncts instead of solid geometries or CPLs

BUYballbearing — returns a list of buyable and used standard parts for ball bearings

camset — sets the camera target to inspect a specific target point

CATIAbackground — just switches to colors similar to CATIA

copyfig — copies a axis diagramm into a new MATLAB figure

copygo — copies graphics objects of a type list from an axis similar to findobj and copyobj

copyplot — copies the content of one figure into a random one

CurrentPointofgcaBB — returns for a given coordinate and the current camera position the CurrentPoint result of a virtual mouse click

deployTL — Procedure that explains how to install the application deployment tools

editorwindow — opens the last file modified in the editor window

findingca — select graphics objects of a specific type

fiofselect3d — Select interactively FS, surface, and complete solids

FLselect3Dginput — selects a point in the gca and finds the facets in a surface

fontsize — sets the font size of all text to a specified figure

fullview — maximize current figure to maximal height of the screen

gcahandleofSG — returns the handle to all patches of gca

gcoofT — returns the crossing points in the current set of graphics objects

getgcapixelsize — returns the position of the gca in the current figure

getkeyboardcharpause — fnct getkey for figures if rotate3d is on - and the callbacks are blocked

getkeyboardchartimer — fnct getkey for figures if rotate3d is on - and the callbacks are blocked

ginput2Dcircle — returns by user interaction center and radius of a circle

ginputTL — procedure similar to ginput

google — just searches a string at google.com

hideingca — simply hide (set visibiliy to off) or delete graphics objects of a specific type

iconfilesearch — searches and shows in the original matlab directories for icon files that contain a string term

iconlibTL — returns an icon for the use in userinterfaces

immeasureui — user interface to measure distances within an image using mouse clicks

imreadui — reads an image file and opens dialogs to clip and scale the image

imwarpT — warps an image related to a given transformation matrix

keyboardlineeditor — implements a text editor for interactive programming

lastcommand — lastcommand searches in the command History

matlabopen — open the matlab application on mac

mimedserver — Access to the TUM MIMED Server through samba and mwn webdisc

mimedserver2webdisk — Creates a mwn webdisc link for a file or folder

nofcolmap — returns a number instead of a color string and support more colors than colofn

papermode — removes picture title and increases font size for publications

pastego — inserts graphics objects into the current axis gca

patchofgca — returns the handle to all patches of the current graphics axis

patchofSG — creates a patch struct of a solid geometry

pixelofaxis — returns the axis position of the current axis in the figure

pixelpositionofP — returns the pixelposition of a point coordinate

PLofhandle — returns the point list of different graphics objects in gca

plotannotatecolors — returns an annotation string from plotannotation

plotannotation — plots a annotation box (per default into the right upper corner) of a plot axis

plotannotationdelete — delete plot annotation

plotannotationtopleft — shows an annotation in the left side of the axis

PLplotlineseg — plots a line using PLplot in distance separated segments

pofselect3D — returns touch point, solid, handle normal vector etc.

readkeyboardchar — reads a single keyboard key or returns empty

Rofview090 — returns the rotation matrix to rotate the object into a view direction

rotateazimut — callback func for figure-azimut-rotating by wheel

screencapture — returns the full screen as frame, i.e. also confidential inforamtion in other applications

screencaputureofcommandwindow — returns the screenshopt of the command window

select2d — interactive drawing of a square and return of the overlapping elements

select3d — Determines the selected point in 3-D data space.

selectCameraPosition — selects and sets the camera position

selectsubplot — interactively select a subplot

selectV3d — plots a SG in a figure and shows selected normal vectors of a selected corner

setgcapixelsize — sets the current gca to a default pixelsize

setplotlight — changes the alpha value of a given graphics object handle

SGButtonDownFcn — Callback Fnct for current axis and all polygons inside

SGfigtrans0 — moves all parts of the current axis into the first octant

SGfigure — Application for opening and viewing STL Files

SGfigureannotation — creates a simple help text annotation into the current figure

SGfigureeval — executes a command line and echos the command in SGfigure

SGfigureInteractiveMove — Interactive moving of patches on the screen

SGfigurepapermode — removes picture title and increases font size for publications

SGfiguretopbuttons — creates a button row in the current figure

SGgethandle — returns a handle if the object is already drawn in gca

SGhelptext — Writes a text annotation box on top of the current axis

SGkeyboardCallback — used as KeyPressFcn-Callback within SG-Library

SGkeyboardgetkey — returns the last key pressed

SGplotalpha — plots a solid support directly color and alpha value

SGplotsurfaces — plots closed surface of the solids seperatly in different colors

SGraspberrystarterkit — this file creates documentation for an raspberry Pi 4 startup kit

SGselect3Dsurface — EXPERIMENT how to select interactively FS, surface, and complete solids

SGtitle — draws the name of the calling fnct as figure title

SGTplotalpha — plots a solid including all frames and supports directly color and alpha value

SGui — Opens a dialog, reads and plots an STL-File

SGWindowButtonMotionFcn — implements the concept of get(gca,'CurrentPoint') also for rotate3d on

snapplot — saves the current figure as JPG-File on the user desktop

sofgca — returns the pixel resolution of gca

stringparsebackspace — processes a char array similar to an input scan process

textadjust2gca — adjust text width to fit in gca

textBB — writes a text at specified position relative to bounding box

textSG — creates descriptors for cells of solid geometries

titleappend — appends a string to the gca title

titleofcaller — returns the name of the calling fnct plus date & time as string

Tofselect3d — Select interactively FS, surface, and complete solids

Tofview — returns the Transformation matrix of the camera for a specific view with a change of view

uipolygoncontextmenu — create the ui context menu for polygon

uipolygonmoveWindowButtonDownFcn — callback fnct and interactive fnct to move and rotate polygons and to attach ui context menu

uipolygonSelectWindowButtonUp — MAIN CALLBACK FUNCT for a figure to make polygons movable if rotate3d is off

uitogglefigstyle — Application callback procedure for toggling grid an colors

viewVLinspect — inspects one by one points of the vertex list

VLFLfigure — Application for opening and viewing STL Files

VLFLofgca — returns the vertex list and facet list of the current figure

VLFLplotalpha — plots a solid support directly color and alpha value

VLFLplotfreeboundary — plots the freeboundary of a solid with grid forced vertices (1e-6)

VLFLplothide — selects a patch (VLFL) by ginput and sets it as invisible

VLFLplotlight — switches the light and renderer on for VLFLplots and VLFLplots4

VLFLui — Opens a dialog, reads and plots an STL-File

VLFLviewer — Application for opening and viewing STL Files

VLofgca — returns all vertices of the current figure patches

VLplotGraphics — more generic version of VL scatter

VLui — returns a vertex list drawn by the user in a plot window

VLwarpgrid — creates a meshgrid for warping an image

xyzlabel — sets or removes the xyz axis labels in 3D plots

zoomVL — simply zooms into a bounding box of the vertex list

Detailed description

app_2012_11_09

Application for opening and viewing STL Files

In: **SG**: Vertex list of solid geometry or Solid Geometry **SG.FL**: Facet list of solid geometry **xcol**: color for plotting the solid; default is 'm'

Out: **hf**: handle to figure **hm**: handle to menu **ht**: handle to toolbar **htt**: handle to button 1

apptogglelighttrans

toggles the VLFLplotligh status from facets to light

axis4math

sets axis for mathematical fncts instead of solid geometries or CPLs

In: **zoomf**: fractional zoom-out margin added around the tight axis; default is 0.1

Out: **ax**: axis ratio after call

BUYballbearing

returns a list of buyable and used standard parts for ball bearings

In: **Ri**: Inner Diameter

Out: **TL**: Table list

camset

sets the camera target to inspect a specific target point

In: **p**: point; default is ''; **d**: diameter of axis; default is 1/10 of the current view **ez**: direction to camera position;

CATIAbackground

just switches to colors similar to CATIA

In: **iscat**: true or false; toggles if not used

copyfig

copies a axis diagramm into a new MATLAB figure

In: **cf**: copy figure handle; default is gcf/gca **nf**: optional number/handle to figure to copy to.

Out: **nf**: new figure handle

copygo

copies graphics objects of a type list from an axis similar to findobj and copyobj

In: **ca**: axis to copy from

Out: **h**: handle to graphics objects

copyplot

copies the content of one figure into a random one

CurrentPointofgcaBB

returns for a given coordinate and the current camera position the CurrentPoint result of a virtual mouse click

In: **p**: point in 3D that is inspected from the current camera position

Out: **pc**: [cp1; cp2] entry point of BB and exit point of bb from camera beam through p

deployTL

Procedure that explains how to install the application deployment tools

editorwindow

opens the last file modified in the editor window

findingca

select graphics objects of a specific type

In: **gotype**: type such as 'point', 'line', 'text','polygon' , 'ptach' etc.

Out: **h**: handle to object

fiofselect3d

Select interactively FS, surface, and complete solids

In: **SG**: Solid Geometry; if empty SGofgca is used **pfi**: optional already preselected facet index **alp**: angle to select faces **mbf**: force button mode 1, 2, 3 **Text**: Optional text for the header of the windows

Out: **fi**: touched facet index **fil**: Facet index list that belongs to touched graphics object **T**: Frame of FS **pp**: Picked 3D point

FLselect3Dginput

selects a point in the gca and finds the facets in a surface

In: **VL**: Vertex List **FL**: Facet list **sh**: if true; only one facet is returned instead of all; default is true

Out: **fi**: list of facets that belong to the mouse click **FL**: facet list used for selection **VL**: vertex list used for selection

fontsize

sets the font size of all text to a specified figure

In: **FS**: default is 16pt on a printout **ds**: if true, the font size is adjusted based on the figure size

Out: **h**: handle to all text **FS**: applied font size in points (after dynamic scaling)

fullview

maximize current figure to maximal height of the screen

In: **wid**: optional width in pixels or percent or [mincol maxcol]

gcahandleofSG

returns the handle to all patches of gca

Out: **h**: handle of all patches in gca

gcoofT

returns the crossing points in the current set of graphics objects

In: **cax**: current axes; default is gca **T**: Frame of View direction; default is Tofcam **allh**: if true all crossed objects are returned

Out: **gcoT**: gra[hics object handle, sorted by distance from T **cpa**: crossing point (list) **dta**: distance point (list)

getgcapixelsize

returns the position of the gca in the current figure

Out: **rect**: [left bottom width height] in pixels

getkeyboardcharpause

fnct getkey for figures if rotate3d is on - and the callbacks are blocked

In: **chrset**: set of allowed character; default is 1:255

Out: **c**: uint8 of char pressed during pause command and rotate3d was on

getkeyboardchartimer

fnct getkey for figures if rotate3d is on - and the callbacks are blocked

In: **chrset**: set of allowed character; default is 1:255 **maxtime**: maximum of waiting time; default is 3 seconds

Out: **c**: uint8 of char pressed during pause command and rotate3d was on

ginput2Dcircle

returns by user interaction center and radius of a circle

In: **gr**: grid resolution; default is 1 **ax**: axis of the screen; default is

Out: **cc**: center of circle **cr**: radius of circle **h**: handle to graphic

ginputTL

procedure similar to ginput

In: **maxnr**: number of points to collect

Out: **VL**: Vertex list

google

just searches a string at google.com

hideingca

simply hide (set visibiliy to off) or delete graphics objects of a specific type

In: **gotype**: type such as 'line', 'text','polygon' etc. **hide**: 'hide', 'delete', 'clear' 'invisible', visible' etc.

Out: **h**: handle to object

iconfilesearch

searches and shows in the original matlab directories for icon files that contain a string term

In: **sterm**: search term of cell list of icon files

Out: **fn**: list of files within the matlab directory **iconnams**: unique list of found icon file names **icondirs**: unique list of directories containing the icon files

iconlibTL

returns an icon for the use in userinterfaces

In: **name**: name of icaon

Out: **a**: icon matrix

immeasureui

user interface to measure distances within an image using mouse clicks

In: **imf**: Image or image file name or empty

Out: **h**: handle to graphics objects to be deleted

imreadui

reads an image file and opens dialogs to clip and scale the image

In: **fname**: filename optional; default is user request **clip**: clipping frame; default is user selection **newx**: target size of the x-axis in units for scaling; default is empty, then requested via dialog

Out: **I**: Clipped Image **ax**: Axes of the clipped and scaled image

imwarpT

warps an image related to a given transformation matrix

In: **I**: image or image size [r c] **T**: Transformation matrix **scal**: scaling factor **cent**: true if centered; default is false **fc**: frame color such as 'b-'; default is ''; **al**: alpha value if the image should be transparent % in SG-Lib 5.2 **"size"**: if used, the scaling factor is the size

Out: **h**: handle to image

keyboardlineeditor

implements a text editor for interactive programming

In: **h**: handle to figure the keyboard is attached to **CLINE**: Textbuffer, Text line or empty **NLINE**: Textbuffer behind current line; or empty **RLINE**: Textbuffer, same line but right hand of cursor **CRNUM**: uint8 character list of escape characters; default is 13 (CR)

Out: **c**: last pressed key **LR**: Text row left of cursor **CLINE**: Complete text buffer including current line **NLINE**: Rest of text buffer after current line **RLINE**: Current text line rifht hand of cursor

lastcommand

lastcommand searches in the command History

Out: **res**: search results

matlabopen

open the matlab application on mac

mimedserver

Access to the TUM MIMED Server through samba and mwn webdisc

In: **text**: abbreviation or keyword for the call

mimedserver2webdisk

Creates a mwn webdisc link for a file or folder

In: **url**: abbreviation or keyword for the call

Out: **wdb**: webdisk link to open MWN weblink; if nargout==0 webdisk is accessed

nofcolmap

returns a number instead of a color string and support more colors than colofn

In: **col**: color integer od string **rgb**: if true, return the RGB triplet instead of the color index; default is false

Out: **col**: number of color **cstr**: color string abbreviation for the color

papermode

removes picture title and increases font size for publications

pastego

inserts graphics objects into the current axis gca

In: **h**: object handles to insert **hgca**: axis to insert and list of tyes

patchofgca

returns the handle to all patches of the current graphics axis

In: **gh**: gca or gcf; default gca

Out: **h**: handle / handle array

patchofSG

creates a patch struct of a solid geometry

In: **SG**: Soldid Geometry; **realp**: true=real patch; default is false

Out: **h**: handle to patch struct or patch class

pixelofaxis

returns the axis position of the current axis in the figure

Out: **sp**: [left bottom size-x size-y] **p0**: axis origin [x0 y0] in data coordinates **ff**: scaling factors [fx fy] from data units to pixels

pixelpositionofP

returns the pixelposition of a point coordinate

In: **p**: point coordinate [x y]

Out: **pf**: pixel coordinate in figure (uicontrol) **ppf**: pixel coordinate in percentage of figure (uipanel) **ppa**: pixel coordinate in percentage of axis

PLofhandle

returns the point list of different graphics objects in gca

In: **h**: handle or string such as 'point', 'line', 'text', 'polygon', 'patch' etc. **ascell**: result as separeted list; default is false

Out: **PLs**: Point lists **hp**: handle to objects if input handle was a string

plotannotatecolors

returns an annotation string from plotannotation

In: **sgcolord**: colors for colofn **varnames**: list of variables or strings to be used to create an annotation

Out: **ptxt**: annotation string

plotannotation

plots a annotation box (per default into the right upper corner) of a plot axis

In: **str**: string or cell with separated strings **position**: Vtop, Vmiddle, Vdown, Hright, Hmiddle, Hleft, left, right, center

Out: **t**: handle to annotation box

plotannotationdelete

delete plot annotation

Out: **h**: handle to deleted textboxes

plotannotationtopleft

shows an annotation in the left side of the axis

In: **str**: string **vargs**: list of optional parameters for annotation

Out: **t**: handle to textbox

PLplotlineseg

plots a line using PLplot in distance separated segments

In: **PL**: Point list **s**: segment number or distances **c**: color **w**: line **vec**: vector tip size **a**: transparancy; default is 0;

Out: **h**: handle

pofselect3D

returns touch point, solid, handle normal vector etc.

In: **obj**: graphics object; default is gca

Out: **p**: touched point **SG**: touched graphics object **ef**: normal vector of facet **ec**: normal vector to camera **fi**: facet index of the touched surface (from select3d) **hp**: handle of the touched graphics/patch object

readkeyboardchar

reads a single keyboard key or returns empty

In: **cmdarray**: optional keyboard buffer to extend OR false for delete keyboard buffer

Out: **cc**: last char or empty **cmdarray**: keyboard including last char

Rofview090

returns the rotation matrix to rotate the object into a view direction

In: **ew**: elevation angle as in view **aw**: azimut angle as in view

Out: **R**: Matrix to rotate a solid that it look in view(0,90) like in (ew,aw);

rotateazimut

callback func for figure-azimut-rotating by wheel

In: **src**: handle to figure **callbackdata**: scroll-wheel callback data; VerticalScrollCount controls the azimuth rotation direction

screencapture

returns the full screen as frame, i.e. also confidential inforamtion in other applications

In: **scsize**: [start end cols rows]; default is get(0,'ScreenSize');

Out: **imgData**: rows x cols x 3 as uint8

screencaputureofcommandwindow

returns the screenshopt of the command window

In: **"save"**:

Out: **imgData**: image data of the command window

select2d

interactive drawing of a square and return of the overlapping elements

In: **styp**: 'polygon' or 'line'

Out: **CPL**: CPL of the square **hpol**: handle to overlapped and selected polygons **hsel**: handle to overlapped and selected lines **PL**: empty; not assigned by the fnct

select3d

Determines the selected point in 3-D data space.

In: **obj**: Graphics Object: default is gco **ppp**: optional point in picture

Out: **pout**: point **vout**: next vertex **viout**: vertex list index **facevout**: touched facet in coordinates **faceiout**: facet list index **axchild**: touched child handle

selectCameraPosition

selects and sets the camera position

In: **p**: point to set; default is empty

selectsubplot

interactively select a subplot

Out: **hi**: selected subplot by last click cursor position **cfsp**: list of all subplots **hk**: list of all subplots except the selected one

selectV3d

plots a SG in a figure and shows selected normal vectors of a selected corner

In: **SG**: Solid Geometry to plot **FS**: font size for vertex labels; default is 16

setgcapixelsize

sets the current gca to a default pixelsize

In: **siz**: [x y], i.e. [c r]

Out: **ns**: [x y], i.e. [c r]

setplotlight

changes the alpha value of a given graphics object handle

In: **hi**: patch handle **col**: color value or letter; optional **alf**: alpha index

SGButtonDownFcn

Callback Fnct for current axis and all polygons inside

In: **~**: callback arg 1 **~**: callback arg 2

SGfigtrans0

moves all parts of the current axis into the first octant

In: **cax**: default is gca

SGfigure

Application for opening and viewing STL Files

In: **VL**: Vertex list of solid geometry; or SG or az of view **FL**: Facet list of solid geometry or el of viee

Out: **h**: handle to figure

SGfigureannotation

creates a simple help text annotation into the current figure

In: **str**: string **bc**: backgoundcolor; default is unchanged **sz**: text size; default is 10 **sl**: starting position from bottom; default is 0.97% **hl**: height of annotation; default is 1-sl **bl**: relative width factor of the annotation box; default is 1 (full width)

Out: **h**: handle to annotation

SGfigureeval

executes a command line and echos the command in SGfigure

In: **codestr**: code to execute **sd**: if true, show date as title; else clear title; default is false

SGfigureInteractiveMove

Interactive moving of patches on the screen

In: **SGfigureInteractiveMovebutton**: Task Order 'VIEW ROTATION' 'SOLID MOVE' 'Solid Rotate'; default is [2 1 3]

SGfigurepapermode

removes picture title and increases font size for publications

In: **FS**: font size; default is 16 **ds**: dynamic size flag scaling font by figure height; default is true

SGfiguretopbuttons

creates a button row in the current figure

In: **nrx**: Nr of the **callback**: desired function to call if pressed; default is 'beep' **bstyle**: button style; default is 'pushbutton' **icfile**: icon file or text **bsize**: button size **tiptext**: description text at tool tip

Out: **[hf]**: Handle to the figure

SGgethandle

returns a handle if the object is already drawn in gca

In: **SG**: Solid Geometry

Out: **h**: handle

SGhelptext

Writes a text annotation box on top of the current axis

In: **helpstr**: help string (yellow) **cname**: caller string (red) **waitp**: if true, turn on rotate3d and pause after showing text; default is false

Out: **h**: handle to text annotation

SGkeyboardCallback

used as KeyPressFcn-Callback within SG-Library

In: **src**: src of callback events **event**: result of callback events

SGkeyboardgetkey

returns the last key pressed

In: **hf**: optional window handle; default is gcf

Out: **key**: key pressed **lastkey**: last key pressed

SGplotalpha

plots a solid support directly color and alpha value

In: **SG**: SOlid Geometry **c**: color; default is []; **a**: alpha value; default is 1; **e**: edge color; default is []; **relatSG**: optional Solid to use as spatial reference **spatrel**: list of spatial relations such as of SGtransrelSG

Out: **h**: handle to grhics object patch

SGplotsurfaces

plots closed surface of the solids seperatly in different colors

In: **SG**: Solid Geometry **col**: color force; default is ''; **Fs**: Font size for descriptors; default is 0 **si**: surface selectors

Out: **h**: handle to drawing

SGraspberrystarterkit

this file creates documentation for an raspberry Pi 4 startup kit

In: **keyword**: keyword for explainMfileheader

Out: **textname**: file name of the created or used documentation file

SGselect3Dsurface

EXPERIMENT how to select interactively FS, surface, and complete solids

In: **SG**: Solid Geometry; if empty SGofgca is used **pfi**: optional already preselected facet index **alp**: angle to select faces **mbf**: force button mode 1, 2, 3 - only in interactive mode **Text**: Optional text label

Out: **fil**: Vertex list that belongs to touched graphics object **VL**: Facet list that belongs to touched graphics object **FL**: logical facet index list of selection **hp**: touched graphics object from select3d **fi**: touched facet index **pp**: Picked 3D point

SGtitle

draws the name of the calling fnct as figure title

In: **n**: hierarchy of calling functions; default is 0; **w**: 'window' renames the window instead of a gca title

Out: **h**: returns the handle since SG-Lib 5.6

SGTplotalpha

plots a solid including all frames and supports directly color and alpha value

In: **SG**: SOlid Geometry **c**: color; default is []; **a**: alpha value; default is 1; **e**: edge color; default is []; **relSG**: Solid as Position reference **relpos**: relative positioning like SGtransrelSG

Out: **h**: handle to graphics object patch

SGui

Opens a dialog, reads and plots an STL-File

In: **mag**: magnification; default is 1 **c**: Color of Object, Plots only if color is defined

Out: **SG**: Solid Geometry of STL File **filename**: filename of selected file **pathname**: pathname of selected file

SGWindowButtonMotionFcn

implements the concept of get(gca,'CurrentPoint') also for rotate3d on

In: **srch**: callback source handle (unused); passed by MATLAB callback **cbdat**: callback event data (unused); passed by MATLAB callback

snapplot

saves the current figure as JPG-File on the user desktop

In: **pname**: name of shown procedure, default is 'Tim C. Lueth' **fname**: desired filename **fonts**: font size for the paper mode figure; default is 20

Out: **fname**: resulting filename

sofgca

returns the pixel resolution of gca

Out: **s**: largest inaccuracy **ss**: inaccuracy (pixel size in units)

stringparsebackspace

processes a char array similar to an input scan process

In: **inpstr**: input string including all control characters

Out: **outstr**: some control characters are processed

textadjust2gca

adjust text width to fit in gca

In: **hh**: handle to text

Out: **hh**: handle to the resized text

textBB

writes a text at specified position relative to bounding box

In: **VLO**: Vertex List or SG or graphics handle **STR**: String **loc**: location "+cc" or "c+-" **c**: color; default is red **s**: size; default is 12 point

Out: **h**: handle **p**: point of text

textSG

creates descriptors for cells of solid geometries

In: **SG**: Solid **c**: color of descriptor **s**: size of descriptor **nt**: selector for cells **lb**: Letter for Desciption **"loc"**: location, default is 'ccc', see also textBB, PofBB

Out: **h**: handle to text

titleappend

appends a string to the gca title

In: **txt**: text to append

Out: **h**: handle to title **old**: old title before change

titleofcaller

returns the name of the calling fnct plus date & time as string

In: **n**: hierarchy of calling functions; default is 0; **dt**: adds date and time if true; default is true **hl**: if true; fname is an hyperlink with line of caller; default is false **cl**: if true, append the caller's code line to the name; default is false

Out: **t**: string with caller and date and clock **fname**: calling function name **lines**: caller line number, optionally with the source code line

Tofselect3d

Select interactively FS, surface, and complete solids

In: **SG**: Solid Geometry; if empty SGofgca is used **pfi**: optional already preselected facet index **alp**: angle to select faces **mbf**: force button mode 1, 2, 3 **Text**: Optional text for the header of the windows

Out: **T**: touched facet index **fi**: Facet index list that belongs to touched graphics object **fil**: Frame of FS **pp**: Picked 3D point

Tofview

returns the Transformation matrix of the camera for a specific view with a change of view

In: **ew**: elevation angle as in view **aw**: azimut angle as in view

Out: **T**: Transformation matrix

uipolygoncontextmenu

create the ui context menu for polygon

Out: **cm**: context menu

uipolygonmoveWindowButtonDownFcn

callback fnct and interactive fnct to move and rotate polygons and to attach ui context menu

In: **h**: calling function **st**: calling event

Out: **h**: handle to polygon **hp**: gandle to all polygons

uipolygonSelectWindowButtonUp

MAIN CALLBACK FUNCT for a figure to make polygons movable if rotate3d is off

In: **srch**: source handle - not used yet **cbdat**: call back data - not used yet

uitogglefigstyle

Application callback procedure for toggling grid an colors

viewVLinspect

inspects one by one points of the vertex list

In: **VL**: Vertex list **vi**: optional index list to VL

VLFLfigure

Application for opening and viewing STL Files

Out: **hf**: handle to figure **hm**: handle to menu **ht**: handle to toolbar **htt**: handle to button 1

VLFLofgca

returns the vertex list and facet list of the current figure

In: **hgca**: handle to graphics object

Out: **VL**: Vertex list of the patches/facets of the current figure **FL**: Facet list of the patches/facets of the current figure **FEL**: Feature edge list for pi/6 **FC**: Facet color list **FA**: Face Alpha; also used as SG.alpha

VLFLplotalpha

plots a solid support directly color and alpha value

In: **VL**: Vertex List **FL**: Facet List **c**: color; default is []; **a**: alpha value; default is []; **e**: edge color; default is [];

Out: **h**: handle to graphics object patch

VLFLplotfreeboundary

plots the freeboundary of a solid with grid forced vertices (1e-6)

In: **VL**: Vertex list **FL**: Facet list **c**: color; default is 'r-' **w**: width of lines

Out: **h**: handle to the drawings

VLFLplothide

selects a patch (VLFL) by ginput and sets it as invisible

VLFLplotlight

switches the light and renderer on for VLFLplots and VLFLplots4

In: **mode**: Rendering mode; **al**: Transpanencecy **goh**: graphic object handle; default is gcf **c**: color of the object only it is a patch

VLFLui

Opens a dialog, reads and plots an STL-File

In: **mag**: magnification; default is 1 **c**: Color of Object, Plots only if color is defined

Out: **VL**: Vertex List of STL File **FL**: Facet List of STL File **filename**: filename of selected file **pathname**: pathname of selected file

VLFLviewer

Application for opening and viewing STL Files

In: **SG**: Vertex list of solid geometry **SG.FL**: Facet list of solid geometry **xcol**: color for the displayed solid; default 'm'

Out: **hf**: handle to figure **hm**: handle to menu **ht**: handle to toolbar **htt**: handle to button 1

VLofgca

returns all vertices of the current figure patches

In: **allpoints**: if false; VL=VLP; if true VL=[VLP;VLM]

Out: **VL**: Vertex list **VLM**: Vertices of lines **VLP**: Vertices of patches **VLS**: vertices of polyshapes/polygons

VLplotGraphics

more generic version of VL scatter

In: **VL**: Vertex list **obj**: radius, CPL, CPS, Solid **col**: color of the plot; either scalar or **alp**: alpha value for solids or polyshapes; not implemented yet

Out: **h**: handle to graphics

VLui

returns a vertex list drawn by the user in a plot window

In: **col**: number of points; default = inf | or existing VERTEX LIST [n x 3] **res**: resolution; default = 0.1 **ax**: axis [xmin xmax ymin ymax]; default=[-10 110 -10 110] **cplline**: line drawing color; default is '' **psdraw**: polyshape color; default is '' **linw**: contour line width; default is 1 **opl**: closed lines: true==> CPL; false => PL;

Out: **PL**: Vertex list (z=0); **h**: handle to delete the drawing afterwards;

VLwarpgrid

creates a meshgrid for warping an image

In: **siz**: image or image size [r c] **T**: Transformation matrix **scal**: scaling factor **cent**: if true; image is centered; default is false **isgeo**: if true color is height in z; default is false

Out: **VL**: Vertex list **XL**: warp grid X **YL**: warp grid Y **ZL**: warp grid Z

xyzlabel

sets or removes the xyz axis labels in 3D plots

In: **c**: 'on' or 'none' or 'off'; default is 'on'

zoomVL

simply zooms into a bounding box of the vertex list

In: **VL**: Vertex List **eq**: growing factor; default is 1.1

Out: **ax**: axis

2.2 Geometry Drawings [guide]

Functions in this section:

animatego — rotates one or several objects in a loop

axesofgcf — returns the handles to the axis

axis4all — applies the commands for axis to all subplots applies the commands for axis to all subplots

axis4animation — automatically increases but never decreases the axis area

axis4datenum — sets the xaxis to datenum ticks, for datenum diagrams wich is now different to datetime ticks

axisAIM — Just sets some figure values for the AIM Book von Tim Lueth and Mattias Traeger

axisratio — returns the axis ratio x y z

axisscale — scales the axis to a defined size

camlightdelete — just removes all camlighs in a gca

camlightTL — deletes all camlights and set three new

camofT — set camera to look on a given HT matrix using current zoom

camofVLFLfi — shows to a specific facet for inspecting a facet of interest

camplot — plots a line of the current camera position in the axis

char2rgb — returns a RGB value from a color character

col2RGB — generates a color vector from the colorchar

color — generates a color vector from the colorchar

colorname — returns a name for a color string or a cell list of colstrings

cplot — Plots a list of circle segments

CPLplotasPSplain — Plots each individual embedded CPL as a separate contour.

CPLplothatched — plots a CPL including hatch

CVLplot — plots a 2D polybool contour polygon in a 3D figure

CVLplots — similar to CVLplot but shows optionally the direction of the line segments

CVLplotsurface — plots a planar CVL as surface

CVLzplotasCPS — similar to CVLplot but separates the z values just in case somebody forgot nan nan

CVLzplotui — interactive editor to scroll through a stack of slices

deleteQuiver — deletes all Quivers (coordinate graphics objects) in the named axes

drawnowvid — this fnct is an auxiliary fnct to automate the creation of videos

ELTplot — plots the edge list of a truss including text and optional also the CPLR

exportgraphicsSG — creates figure exports for the AIM Book or other Books of Tim Lueth

FEplot — plots the featureEdges of TR, SG or VLFL

figcopyright — draws a copyright marker inside of the current figure

figsize — sets the figure size

figureisopen — returns whether there is already a graphics window open

figurerotate — rotates the current figure to show 3D effects

fontsize4rowsfullHD — font size for lectures and video recording on 25 lines

FSplot — plots the featureEdges of TR, SG or VLFL

gcasnapshot — saves the current figure as JPG-File on the user desktop

gccp — returns cameraposition, cameradirection and cameratarget

gcfSG — in contrast to gcf, this fnct returns empty if there is no open window

GeoTIFFBayern — load GeoTIFF Data from the Bayern cloud

getframeaxis — same as frameaxis but also stores the axis information

getprojectionimage — returns a single projection image of gca

Graphplot — plot a graph with coodinates

iconofCPL — creates an grayscale image/icon of a contour

im2stereo — returns corresponding points within 2 images taken from different view points

im2vec — simply converts/reshapes an image into a 1 x n vector

imadjustwhite — adjusts the intensity values from both side that

image2strbase64 — creates an base64 string to integrate images directly into html files

imageFolderContent — finds all image files within a folder

imageofgca — returns the image snapshot of the currect gca

imageofipcam — returns an image of the ipcam

imageoftext — returns a binar matrix and an image matrix for a text string

imageofVLprojection — returns an image of a VL projection using the matlab camera view concept

imageVideoTextPage — creates a textpage image for a video

imofimsketch — uses an imbinarize filter to improve images before contourfind

implot — plots an image as scaled texture

imreadlastchanged — reads in the last modified image of a directory

imreadraw — reads an image file as raw data

imresizep — resizes an image proportional relative to a mimum number of pixels

imstackofSGprojection — creates projection images similar to a CT from a solid

imstackread — reads in an image stack as default uint16

imstackwrite — writes a gray scale intesity matrix n x n x m into raw files (used for radon)

isSGfigureStop — will be called during animations

lplot — plots a straight line between 2 points

lplots — plots a straight lines between 2 point lists

patchupdateSG — replaces an existing patch handle by updated SG

plotannotationSG — plots a solid into the upper right corner of the current graphics axis

plotannotationtext — create quick an efficient text to be used in combination with plotannotation

plotCVL — plots facet that is enclosed by the vertex contour list

plotL — plots a HT-matrix as straight line

plotruler4SG — creates dimensions on the outside of a geometric body

plotrulerline — draws a simple ruler between two points in 2D

plotT2P — plots a HT matrix as straight line vector

plotTP — plots the basic planes of a coordinate system

PLplot — plots a 2D polygon in a 3D figure

PLplotcorrelation — simply plots a point list correlation

popgcf — remains only the remembered graphics object in the current figure

pplot — pplot shows a 3D-plot figure of a point or a list of points

projectionimage — returns a central projection image by using Matlabs view commands

pushgcf — remembers all graphics object in the current figure;

rgb2bgr16 — converts a RGB 3 x 8 Bit color into a BGR 3 x 5 + 1 Bit = 16 Bit Value for STL attribute by color values

rgb2char — returns a color character from an RGB value

rgbofall — return all different methods of matlab into a rgb set as type double

rotate3dlight — Switches the automatic camera light conditions on and off

rulerCPL — plots ruler for a closed polygon list

rview — Restores parameters of the current figure/View

select3dofcam — creates all selectin information for the

setcamangle — set the camera angle is set

setcamdirection — sets the camera position without changing the camera target

setcamdistance — The distance of the camera position relative to the target position is fixed

setcamtarget — set the camera target interactively by a mouse click or a point or a frame

setcamzoom — zoom by changing the camera view angle;

SGanimateSLLL — Animates a set of solids using the SLLL format

SGarrow — returns a SG for an arrow

SGcastplot — creates plots in a cell list wrt to the SGcast type

SGcellofgcapatch — returns the plotted patches of gca as separated solids as cell array

SGcolorerrors — idea simply to color solids instead of showing the errors

SGfigurefooter — plots a footer into the window

SGfigureInteractiveSort — fnct to sort surfaces into two independent solids interactevely

SGimaging — creates a simulated transparent surface view or slice view

SGofgca2subplots — takes the current gca and creates a cell list of solids/patches

SGofgcfsubplots — takes the current gcf, collects all patches in the subplots and creates a cell list of solids/patches

SGofT — generates an infinitely small cube (1e-17) at T

SGplot — plots a solid geometry

SGplot4 — plots a solid in 4 views

SGplotcellmultiple — Creates a SGfigure and plots all cell elements in different subplots

SGplotcellsplitanimation — plots the solids of an cell array depending on their minimal z value one after another

SGplotgears — plot a gear i predefined colors from blue(1st) to orange (last)

SGplotmultiple — creates a figure and simply plots all solids in a cell aray

SGplotparts — plots the overlapping parts of a solid separately

SGshadelight — sets shading and lightinhg

SGshowcolors — shows different colors to understand how to set rgb values

SGsurfacehistogram — plots a surface area histogram of selected surfaces

showrf — returns or shows only the reverse facets from the current view point

slplot — straight line plot

spreadmax — spreads the values of a matrix for a desired intervall min to max

subplotannotation — places a text annotation below the current subplot

subplotn — Creates the i-th subplot in an automatically sized grid for n subplots

subplotnumber — simply returns the number of subplots

surfofgca — returns the surfaces (not the patches) from the graphics handel

sview — Stores parameters of the current figure/View

tcamera — set camera to look on a given HT matrix using current zoom

textCPL — plots descriptors for each contour into a figure using text command

textCPS — writes text descriptors to CLPS

titleprintftext — creates a title based on the parameterlist given by the calling fnct

tlplot — plots a line given as HT matrix using t+k*ez

togglefig — toggels the position of the current SGfigure

toggleviewprojection — toggle view projection between "orthographic" and "perspective"

tplotaxis — plots the elongation of the z-axis of a frame or a frame list

tplotruler — plots a 3D ruler as arrow sequence in a figure

updategco — updates an graphics object with new coordinates

video2animatedGIF — creates an animated GIF from a video file

video2video — creates an AVI in different size and speed from a video file

view4all — sets the view angles for all subplots the same

viewsmooth — slowly rotates to a new viwe

VLcol — generates a simple colormap related to the basic Matlab colors

VLELCILplots — plots contours given as vertex list and edge list

VLELplot — plots lines given as vertex list CVL and edge list CLL

VLELplots — plots lines given as vertex list and edge list

VLFLcamlight — set the current light source on the position of the viewpoint

VLFLELplot — plots an edge list and all attached facets of the edges

VLFLfaceNormalplot — plots normal vector in appropriate length and writes the name of the facet at the tip

VLFLofPatch — returns vertex list and facet list of current figure or the current object

VLFLplot — plots a vertexlist and facet list by fill3 in a figure

VLFLplot4 — plots a vertex/facet list by fill3 in a figure with 4 Subplots

VLFLplotfacets — plot a facet list as separate independent facets

VLFLplotfi — plots the scenery around some vertices in unequal scale

VLFLplotlightoff — switches the camlights on or off

VLFLplotmanifold — plots a the non manifold points, edges, and facets of a solid

VLFLplots — plots an object/solid as shaded or faceted object

VLFLplots4 — plots an object/solid as shaded or faceted object in 4 views

VLFLshow — simply brings the current figure in the front

VLofgcalines — creates a list of vertex points used in

VLOLplot — prints one or all surfaces of an object list

VLplot — Plots different views of a vertex list (nx3) as contour lines or individual points.

VLplotdelaunay — simple inspection fnct

VLplotmotion — draws position, speed and acceleration of a VL

VLscatter — plots a vertex list with a corresponding image intensity list

VLULplot — Plots the union list of an object, union face by union face

zoompatch — zooms to see all patches of the current figure axis

zoomsubplot — opens a figure and zooms the subplot image

Detailed description

animatego

rotates one or several objects in a loop

In: **h**: handle **T**: Frame 4x4 **w**: angle difference in ez **style**: 'rotz' or 'linz'; default is 'rotz'

Out: **h**: handle to new graphics object

axesofgcf

returns the handles to the axis

In: **h**: handlo to graphics objects; default is gcf

Out: **h**: handles as array (sorted descending)

axis4all

applies the commands for axis to all subplots applies the commands for axis to all subplots

In: **fig**: default is gcf; no need to use it as first paramter **axisparams**: list of parameters same as for axis

axis4animation

automatically increases but never decreases the axis area

In: **defax**: initial values to set

Out: **nx**: current axis values

axis4datenum

sets the xaxis to datenum ticks, for datenum diagrams wich is now different to datetime ticks

axisAIM

Just sets some figure values for the AIM Book von Tim Lueth and Mattias Traeger

axisratio

returns the axis ratio x y z

Out: **ar**: [1 y/x z/x]

axisscale

scales the axis to a defined size

In: **ss**: [dx dy dz]

camlightdelete

just removes all camlighs in a gca

camlightTL

deletes all camlights and set three new

In: **mod**: 'on' to set the camlights, otherwise only delete existing ones; default is 'on'

Out: **h**: handle to three camlights

camofT

set camera to look on a given HT matrix using current zoom

In: **T**: HT matrix, the figure axes camera should look on

camofVLFLfi

shows to a specific facet for inspecting a facet of interest

In: **VL**: Vertex list **FL**: Facet list **fi**: facet **gf**: grow arund factor; default is 10

camplot

plots a line of the current camera position in the axis

Out: **p0**: CameraPosition **p1**: CameraTarget **v**: CameraViewAngle **a**: axis

char2rgb

returns a RGB value from a color character

In: **ca**: color character

Out: **rgb**: [R G B] color

col2RGB

generates a color vector from the colorchar

In: **cm**: color 0:7

Out: **RGB**: index to colormap VLcol

color

generates a color vector from the colorchar

In: **c**: color'' r y m c g b w k'

Out: **cm**: index to colormap VLcol

colorname

returns a name for a color string or a cell list of colstrings

In: **cs**: color such such as 'r--', or {'r--','b.-'} **num**: length of format; 0 is automatic

Out: **cname**: string or cell list of strings

cplot

Plots a list of circle segments

In: **A0**: Center and Radius of circle [x y R]; default is [0 0 1] **wins**: [min max]; angle **fc**: optional FaceColor **fa**: optional FaceAlpha **ec**: optional EdgeColor **lw**: optional Edge Line Width

Out: **h**: handle to go

CPLplotasPSplain

Plots each individual embedded CPL as a separate contour.

In: **CPL**: CPL with nan separated CPLi

Out: **h**: handle

CPLplothatched

plots a CPL including hatch

In: **CPL**: CPL **c**: color **al**: [alpha; width, distance]

Out: **h**: handle to go

CVLplot

plots a 2D polybool contour polygon in a 3D figure

In: **CVL**: Point list of x/y coordinates nx2 **c**: Color and symbol of points **w**: Width of the used line

Out: **h**: Handle to figure

CVLplots

similar to CVLplot but shows optionally the direction of the line segments

In: **CVL**: Several closed vertex list separated by nan **c**: Optional color of desired edges; default changing colors **w**: Optional line width; default is 1 **t**: Size of a cone shaped arrow tip, default is '=invisible **b**: Size of a cube showing the start point; default is '=invisible **f**: Size of a ring showing the end point; default is '=invisible

Out: **h**: handle to drawing elements

CVLplotsurface

plots a planar CVL as surface

In: **CVL**: CVL **c**: color **a**: transparenct **e**: edge color **planar**: true for planar CVL's; default is true

Out: **h**: handle to graphics

CVLzplotasCPS

similar to CVLplot but separates the z values just in case somebody forgot nan nan

In: **CVLz**: Several closed vertex list separated by nan **c**: Optional color of CPS; default changing colors **a**: Optional facete alpha **e**: Optional line color

Out: **h**: handle to drawing elements

CVLzplotui

interactive editor to scroll through a stack of slices

In: **CVLz**: Stack of CVLs **SG**: Optional Solid Geometry to plot in white

deleteQuiver

deletes all Quivers (coordinate graphics objects) in the named axes

In: **ax**: axes, default is gca

drawnowvid

this fnct is an auxiliary fnct to automate the creation of videos

In: **n**: number of video frames to write (or seconds if non-integer); default is 1

ELTplot

plots the edge list of a truss including text and optional also the CPLR

In: **ELT**: Edge list fo complete truss as generated by **PLA**: Point coordinate list of all used joint indices in ELT **CPLR**: optional CPLR if should be plotted **Rauto**: auto radius for drawing; default is sofBB(PLA)/40

Out: **h**: handle to graphics

exportgraphicsSG

creates figure exports for the AIM Book or other Books of Tim Lueth

In: **go**: default is gca **fname**: default is desktopdir('SNAPSHOT')

FEplot

plots the featureEdges of TR, SG or VLFL

In: **SG**: surface (Tetrahedron, Triangulation, Solid) **ang**: feature edge angle threshold; default 1

Out: **h**: handle to the plotted feature edges

figcopyright

draws a copyright marker inside of the current figure

In: **cname**: copyright owner; default is user name **fonts**: font size of the copyright text; default is 20

Out: **h**: handle to the text **lc**: position if text in gca

figsize

sets the figure size

In: **sz**: desired size; default is [960 540] **resz**:

Out: **[nsz]**: new size

figureisopen

returns whether there is already a graphics window open

Out: **io**: is open => true

figurerotate

rotates the current figure to show 3D effects

In: **mtime**: minimal rotation time; default is 1 **angle**: rotation angle; default is 5 degree

fontsize4rowsfullHD

font size for lectures and video recording on 25 lines

In: **rows**: 25 or 40 rows; default is 25

Out: **fs**: fontsize

FSplot

plots the featureEdges of TR, SG or VLFL

In: **TR**: surface (Tetrahedron, Triangulation, Solid) **alp**: alpha value for neighbor facets **fsi**: selected FS; depend on alpha; default is all **col**: color of FS; default is colofn **siz**: size of descriptors; default is 16 **v**:

Out: **h**: handle to graphics object consisting of text and patches

gcasnapshot

saves the current figure as JPG-File on the user desktop

In: **cax**: axis to snap, default is gca **fname**: optional file name; default is desktopdir(expname) .png

Out: **fname**: file name

gccp

returns cameraposition, cameradirection and cameratarget

Out: **cp**: camera position of current gca **cd**: camera direction of current gca **ct**: camera target of current gca

gcfSG

in contrast to gcf, this fnct returns empty if there is no open window

Out: **h**: handle or empty

GeoTIFFBayern

load GeoTIFF Data from the Bayern cloud

In: **L**: Longitude; default is 737 (Niederaudorf) **B**: Latitude; default is 5284 (Niederaudorf) **Res**: Resolution default is 1

Out: **I**: Image of the selected **txt**: complete command used **L**: Longitude **B**: Latitude

getframeaxis

same as frameaxis but also stores the axis information

In: **ax**: default is gca

Out: **I**: Image struct as of getframe

getprojectionimage

returns a single projection image of gca

In: **pc**: camera position [-x y z] with direction through [0 0 0] **vdist**: distance of the screen behind [0 0 0] **vwidth**: length of the screen in y and z **pc**: camera up vector; default is [0 -1 0]

Out: **I**: Image calculated by getframe and getgcapixelsize

Graphplot

plot a graph with coodinates

In: **G**: Graph **GPL**: optional Node Coordinates **M**: optional Color Values

Out: **h**: handle to graphics object

iconofCPL

creates an grayscale image/icon of a contour

In: **CPL**: Closed Polygon line **graysc**: true= grayscale; false=binary **m**: size of icon; default is 16 **turn**: true == transform tp eigenvectors; default is true

Out: **I**: Image of size [m x m]; maximum intensity [0..1] **PL**: PL optional point list for all values of I==0 (binary)

im2stereo

returns corresponding points within 2 images taken from different view points

In: **I1**: RGB image 1 **I2**: RGB image 1

im2vec

simply converts/reshapes an image into a 1 x n vector

In: **Iset**: Image

Out: **Vec**: Vector

imadjustwhite

adjusts the intensity values from both side that

In: **IS**: original intensity matrix

Out: **IS**: modified intensity matrix

image2strbase64

creates an base64 string to integrate images directly into html files

In: **fname**: filename, weblink oder image array **width**: desired with for rescale/resize

Out: **str**: base64 string **htmlstr**: HTML img src attribute string with the base64-encoded image data and width style

imageFolderContent

finds all image files within a folder

In: **DirName**: folder name

Out: **ImFileList**: cell list of image files

imageofgca

returns the image snapshot of the currect gca

In: **h**: optional different handle than gca such as gcf **rs**: resize parameter, see imresize

Out: **I**: Image of gca

imageofipcam

returns an image of the ipcam

In: **u**: url - used at least for the first call

Out: **I**: image read from the ip camera; empty on failure

imageoftext

returns a binar matrix and an image matrix for a text string

In: **line**: text string **siz**: Size, default is 10 point **col**: Gray scale, default is 204 **ha**: 'l','c','r'; default is 'l' **FN**: font name; default is 'default'

Out: **I**: Binary matrix **C**: Color image **n**: number of lines found

imageofVLprojection

returns an image of a VL projection using the matlab camera view concept

In: **VL**: Vertex list; default is a square for testing **isize**: image size [x y] centered; this defines the view angle **pc**: position of camera; default is [0 +100 0]; **pt**: position of screen; default is [0 -100 0]; **pixnum**: pixels of image; default is isize; **cup**: cup vector; default is [1 0 0]

Out: **I**: Gray scale matrix [0..1] **Ic**: Color Matrix struct I.cdata **pixs**: pixel size in mm; scaling **Tc**: HT matrix of camera position **Tt**: HT matrix of target position **C**: SG of the perspective cone in space

imageVideoTextPage

creates a textpage image for a video

In: **xy**: [x y] or filenamepath of video **ETitles**: EndTitle single row! **cols**: ['w','k'] is default **time**: number of seconds to show; default is 5 **WName**: Name of Videopage **librk**: line break; default is true **ipret**: optional latex interpreter **FSI**: optional font size; default is 18

Out: **I**: Image to write **h**: if handle is used as outpunt; the handle is not closed afterwards

imofimsketch

uses an imbinarize filter to improve images before contourfind

In: **I**: Image of a scan or sketch of drawings or letters

Out: **IN**: improved image

implot

plots an image as scaled texture

In: **I**: Image (just nxmx colors) or struct containing cdata and axis **newx**: length of the xaxis **newp**: origin; default is [0 0 0]; **scf**: if true, rescales the image to the current axis ratio (imresize); default false

Out: **h**: handle

imreadlastchanged

reads in the last modified image of a directory

In: **iext**: image extensionsl default is {'*.png' '*.jpg' '*.tif'} **idir**: image directory; default is desktopdir

Out: **im**: image read in by image read **fullnam**: full filename

imreadraw

reads an image file as raw data

In: **filname**: raw data projection image file **sz**: rows and cols of the image;default is automatic detection of a square matrix **bit**: bits per pixel; default is 16

Out: **IMG**: single image as real matrix **sz**: size or detected size

imresizep

resizes an image proportional relative to a mimum number of pixels

In: **I**: Images **maxrc**: maximum number of rows or cols; or [rw nan] or [nan col]; **defcol**: color for borders to extend; default is 255 **cut**: if used [cutx cuty], the image is not streched to maxrc but scaled proportinally and later cut at [cutx cutv]

Out: **I**: resized image smaller than [maxrc x maxrc]

imstackofSGprojection

creates projection images similar to a CT from a solid

In: **SG**: Solid Geometry; default is SGofgca **sz**: size; default is 300 **nw**: number of images; default is 180 **tr**: transparency level; default is 0.4 **"video"**: if used; a video is created for educationsal purposes

Out: **IS**: image stack **dirnam**: name of directory on desktop **dbeam**: distance from camera to the object center for the projection

imstackread

reads in an image stack as default uint16

In: **dirname**: folder name or file name **r90**: if true, imageas are rotated; default is TRUE **siz**: size; default is taken from file size **scl**: scaling; default is 1 **"ext"**: default is 'raw'

Out: **IS**: image stack **seldir**: selected directory

imstackwrite

writes a gray scale intesity matrix n x n x m into raw files (used for radon)

In: **IS**: Image stack (n x n x m) **dirnam**: directory name **spr**: if true; the images are spreaded to achieve maximum of intensity; default is false **inv**: if true; the images are inverted change b/w to w/b; default is false

Out: **dirnam**: name of directory created

isSGfigureStop

will be called during animations

In: **SGfigureStop**: if given, sets the persistent stop flag to this value

Out: **fact**: true if stopped

lplot

plots a straight line between 2 points

In: **p1**: Starting point **p2**: End point **c**: Color and symbol default is "r*-" **w**: line width; default is 1 **ts**: tip size; 0==auto **ss**: start block size; 0==auto **str**: Text string at tip **fs**: font size of text

Out: **hh**: handle to all line elements **pt**: point between p1 and p2 for text

lplots

plots a straight lines between 2 point lists

In: **P1L**: Starting point **P2L**: End point **c**: Color and symbol default is "r*-" **w**: line width; default is 1 **ts**: tip size; 0==auto **ss**: start block size; 0==auto **str**: Text string at tip **fs**: font size of text

Out: **h**: handle to all line elements

patchupdateSG

replaces an existing patch handle by updated SG

In: **h**: existing patch handle **SG**: Solid Geometry or VL **FL**: optional Faces List

plotannotationSG

plots a solid into the upper right corner of the current graphics axis

In: **SG**: Solid Geometry **"time"**:

Out: **h**: handle

plotannotationtext

create quick an efficient text to be used in combination with plotannotation

In: **NAME**: 1st and each second are the names of variables **VALU**: 2nd and each second are values of variables

Out: **txt**: char array

plotCVL

plots facet that is enclosed by the vertex contour list

In: **VL**: Vertex list **c**: optional color, default is 'r'

plotL

plots a HT-matrix as straight line

In: **T**: HT matrix **c**: color, default = 'g-.' **s**: size, default = 10 **w**: width, default = 2 **N**: Text string

plotruler4SG

creates dimensions on the outside of a geometric body

In: **SG**: Solid geometry for displaying scales

Out: **h**: handles to ruler

plotrulerline

draws a simple ruler between two points in 2D

In: **P1**: Start Point or [P1;P2] **P2**: End Point **nam**: Name, for instace "D="

Out: **h**: handle to the ruler graphics

plotT2P

plots a HT matrix as straight line vector

In: **p**: Point or **c**: color **s**: width **w**: String for name **N**: Name of Frame

plotTP

plots the basic planes of a coordinate system

In: **T**: HT matrix for coordinate system **f**: size of the plane **a**: transparency value **c1**: color of plane x==0 ; default is 'r' **c2**: color of plane y==0 ; default is 'b' **c3**: color of plane z==0 ; default is 'g'

Out: **h**: handles to the plotted planes

PLplot

plots a 2D polygon in a 3D figure

In: **PL**: Point list of x/y coordinates nx2 **c**: Color and symbol of points **w**: Width of the used line **vec**: if v>0, vectors are plotted from the origin **a**: transparancy; default is 0;

Out: **h**: Handle to figure

PLplotcorrelation

simply plots a point list correlation

In: **CL**: CL see PLcorrelate **CPLA**: CPLA **CPLB**: CPLB **col**: color; default is 'r-' **wdt**: width; default is 2 **ss**: selection of rows in CL **z**: specific z difference

popgcf

remains only the remembered graphics object in the current figure

pplot

pplot shows a 3D-plot figure of a point or a list of points

In: **p**: Point or list of points to be plotted 2xn or 3xn **c**: Color and symbol of points **w**: width of the used line

Out: **h**: Handle to figure

projectionimage

returns a central projection image by using Matlabs view commands

In: **pc**: camera view point **vdist**: camera target point **vwidth**: virtual screen width **pixel**: pixel size of virtual screen **w**: single angle value or list an angles; default is 0

Out: **I**: Image of pixel size or image stack **cax**: color axis [min max] **pc**: center point or list of center points

pushgcf

remembers all graphics object in the current figure;

rgb2bgr16

converts a RGB 3 x 8 Bit color into a BGR 3 x 5 + 1 Bit = 16 Bit Value for STL attribute by color values

In: **RGB**: [0..255 0..255 0.255] or 'r', 'y', 'b' etc.

Out: **bgr16**: Attribute by Count Value 16 Bit

rgb2char

returns a color character from an RGB value

In: **RGBA**: [R G B] color

Out: **ci**: color character

rgbofall

return all different methods of matlab into a rgb set as type double

In: **x**: input character 'b' or [255 0 0] etc.

Out: **rgb**: rgb value between 0 and 1

rotate3dlight

Switches the automatic camera light conditions on and off

In: **tog**: 'on', 'off', or toggle by no parameter

rulerCPL

plots ruler for a closed polygon list

In: **CPL**: CPL **ORG**: Position of ruler; default is center of CPL **OEZ**: Second Point for direction **SIZ**: size of tip; default is sofBB/20

rview

Restores parameters of the current figure/View

In: **varargout**: optional argument; if given, suppresses the warning when no view was stored

select3dofcam

creates all selectin information for the

Out: **pout**: point **vout**: next vertex **viout**: vertex list index **facevout**: touched facet **faceiout**: facet list index **handle**: handle to graphics object

setcamangle

set the camera angle is set

In: **na**: empty or 3D point or 4x4 HT Matrix **nd**: new camera distance to the target; if empty the distance is unchanged

setcamdirection

sets the camera position without changing the camera target

In: **T**: ez or HT matrix; only ez is used

setcamdistance

The distance of the camera position relative to the target position is fixed

In: **nd**: new distance; default is unchanged **zf**: zoom factor at this positionl default is 1

setcamtarget

set the camera target interactively by a mouse click or a point or a frame

In: **pp**: empty or 3D point or 4x4 HT Matrix

Out: **T**: frame at the camera target pointing along the target-to-camera direction

setcamzoom

zoom by changing the camera view angle;

In: **zf**: zoom factor; default is 1

SGanimateSLLL

Animates a set of solids using the SLLL format

In: **SLLL**: {Namestr, SG, TL-Framelist, color, [minlay maxlay]} **tmax**: time period to animate the movement; default is 10 sec

Out: **h**: handle to solids and annotation

SGarrow

returns a SG for an arrow

In: **l**: length or [start point; end point] **h**: diameter **s1**: tip size **w**: true or Radius for bending

Out: **SG**: Solid geometry of an arrow in z-direction

SGcastplot

creates plots in a cell list wrt to the SGcast type

In: **SGO**: Cell list of objects **TTO**: type of cell entry; if empty SGcast is used **Name**: optinal name as subplot title **cols**: color string

SGcellofgcapatch

returns the plotted patches of gca as separated solids as cell array

In: **hgca**: default is gca

Out: **SG**: cell array of the individual plotted solids

SGcolorerrors

idea simply to color solids instead of showing the errors

In: **SG**: Solid Geometry

Out: **SG**: Colored Solid Geometry

SGfigurefooter

plots a footer into the window

In: **textstr**: textstring for footer

Out: **h**: handle to text

SGfigureInteractiveSort

fnct to sort surfaces into two independent solids interactevely

In: **SG**: Original Solid

Out: **SGR**: Right selection **SGL**: Left selection

SGimaging

creates a simulated transparent surface view or slice view

In: **SG**: Solid Geometry **T**: View direction of target to camera **d**: thickness of volume **al**: alpha value of solid during imaging **pix**: pixel size of image; default is 512 x 512 **dims**: dimensions of the image; default is maximum

Out: **I**: Image struct including cdata, 'TI', and 'sc' as fields **TI**: Frame of upper left pixel **sc**: scale of image **SGA**: solid geometry of the imaging bounding box (volume slice) placed at frame T

SGofgca2subplots

takes the current gca and creates a cell list of solids/patches

In: **hgca**: axis to use; default is gca

Out: **SG**: cell array of solids **l**: patches in the given axis

SGofgcfsubplots

takes the current gcf, collects all patches in the subplots and creates a cell list of solids/patches

In: **ind**: indices of the axis that should be considered

Out: **SG**: cell array of solids in all subplots **ii**: indices of the axes actually processed

SGofT

generates an infinitely small cube (1e-17) at T

In: **T**: HT matrix 4x4

Out: **SG**: infinitely small cube (1e-17) at T

SGplot

plots a solid geometry

In: **SG**: Solid Geometry Structure **c**: Color; Option for VLFLplot **w**: Width; Option for VLFLplot **T**: optional HT matrix **N**: optional Name of Coordinate System

Out: **h**: handle to figure

SGplot4

plots a solid in 4 views

In: **SG**: Solid Geometry

SGplotcellmultiple

Creates a SGfigure and plots all cell elements in different subplots

In: **SGc**: Cell list of Solids that should be printed **"scale"**: if used, all figures have the same scale **"frames"**: if used, all figures show the frames too **"unique"**: 'frames' flag to also plot the frames **"color"**: 'unique' flag to group and plot only unique surfaces

Out: **h**: handle to graphics objects

SGplotcellsplitanimation

plots the solids of an cell array depending on their minimal z value one after another

In: **SGall**: cell array of solids

SGplotgears

plot a gear i predefined colors from blue(1st) to orange (last)

In: **SG**: cell list of gear solids **cols**: colorstring or color cells; default is 'bcgmo' **al**: transparency default is 1 **f**: gear ratio f calculated by SGgearchainwindup

Out: **hh**: handle as cell list to be used for Videoquickanimatego

SGplotmultiple

creates a figure and simply plots all solids in a cell aray

In: **SGc**: Cell array of Solids **frame**: if true; frames are plotted also; default is false **isunique**: if true, unique surfaces are grouped via SGuniquesurfaces; default is false

SGplotparts

plots the overlapping parts of a solid separately

In: **A**: Solid or Group Part Solid (A.SG) **c**: color or colorstring such as 'rbwcr' **Fs**: Font size for Descriptor; default is 16 **si**: selector for group parts

Out: **h**: handle to graphic objects

SGshadelight

sets shading and lightinhg

In: **nr**: 1..9 for different shadings

Out: **nr**: use less parameter; used only for creating an example result

SGshowcolors

shows different colors to understand how to set rgb values

In: **n**: Number of colors to display **c**: Optional color(s) to display

SGsurfacehistogram

plots a surface area histogram of selected surfaces

In: **B**: Solid Geometry **si**: surface index wrt SGsurfaces; default '' **be**: histogram bin edges; default is be=[0 0.1 0.2 0.3 0.4 0.5 0.6 1.0];

showrf

returns or shows only the reverse facets from the current view point

In: **VL**: Vertex list of a solid; default is VLFLofgca **FL**: Facet list of a solid; default is VLFLofgca

Out: **varargout**: index list of facet normal vectors for the current view point

slplot

straight line plot

In: **p**: starting point **ez**: direction if straight line **c**: line color **w**: line width; default is 1 **l**: line length; default is 10 **d**: color of the starting point

spreadmax

spreads the values of a matrix for a desired intervall min to max

In: **IS**: matrix **mm**: [min max]; defualt is [0 255]

Out: **ISN**: new matrix

subplotannotation

places a text annotation below the current subplot

In: **txt**: annotation text; default is empty (no annotation)

Out: **h**: handle of the created annotation textbox

subplotn

Creates the i-th subplot in an automatically sized grid for n subplots

In: **n**: total number of subplots; used to compute a near-square grid layout. **i**: index of the subplot to create/activate within the grid.

Out: **h**: handle of the created subplot axes.

subplotnumber

simply returns the number of subplots

In: **fig**: default is gcf

Out: **n**: number of subplots **subs**: list of handles

surfofgca

returns the surfaces (not the patches) from the graphics handel

In: **gh**: axis handle; default is gca

Out: **h**: handle to all surfaces of hf **VL**: Vertex list of surface **FL**: Triangulated Facet List of Surfaces using surf2patch

sview

Stores parameters of the current figure/View

Out: **varargout**: string with the current view angles and figure position

tcamera

set camera to look on a given HT matrix using current zoom

In: **T**: HT matrix, the figure axes camera should look on **d**: optional distance between camera and target; default is the current distance

textCPL

plots descriptors for each contour into a figure using text command

In: **CPL**: Contour list separated by Nan (CPL/CVL) **c**: Color, default is black **s**: Size, default is 16 **nt**: optional list of selected contours **lb**: optional string for text; default is 'C'

Out: **h**: handle to text

textCPS

writes text descriptors to CLPS

In: **CPS**: CPS **c**: color **s**: size **nt**: selector **lb**: text

Out: **h**: handle to text

titleprintftext

creates a title based on the parameterlist given by the calling fnct

Out: **h**: handle to title

tlplot

plots a line given as HT matrix using t+k*ez

In: **T**: Transformation matrix **c**: color **l**: length of line

togglefig

toggels the position of the current SGfigure

toggleviewprojection

toggle view projection between "orthographic" and "perspective"

tplotaxis

plots the elongation of the z-axis of a frame or a frame list

In: **TL**: Frame list **c**: color **a**: alpha value **dl**: radius and z starting at T **FN**: frame name label; default is 'ez'

Out: **h**: handle to graphics

tplotruler

plots a 3D ruler as arrow sequence in a figure

In: **T**: Frame to start the ruler along ez **z**: sequence of z values **s**: size; default is 15 **st**: z values; default is "delta", other is "coord"

updategco

updates an graphics object with new coordinates

In: **h**: handle to graphics object **PL**: new point list

video2animatedGIF

creates an animated GIF from a video file

In: **fname**: Video file; default is ''=> selection by browser **WName**: Name for animated GIF file; default is same as Video file name **sizecr**: size columns by rows; default is unchanged; if scalar<10 => ratio if scalar>10 =>rows **cutt**: [start time end time]; default is [0 inf] **frames**: fps; default is unchanged; 10 is a useful value; **accel**: integer acceleration factor > 1; default is 1

video2video

creates an AVI in different size and speed from a video file

In: **fname**: Video file; default is ''=> selection by browser **WName**: Name for animated GIF file; default is same as Video file name **sizecr**: size columns by rows; default is unchanged; if scalar<10 => ratio if scalar>10 =>rows **cutt**: [start time end time]; default is [0 inf] **frames**: fps; default is unchanged; 10 is a useful value; **accel**: integer acceleration factor > 1; default is 1

Out: **WName**: Name of written File

view4all

sets the view angles for all subplots the same

In: **AZ**: azimut angle; default is AZ of gca **EL**: elevation angle; default is EL of gca

viewsmooth

slowly rotates to a new viwe

In: **az**: azimut similar to view **el**: elevantion similar to view **n**: number of steps; default is 15 **alelorg**: [az el] original view to restore; used only when recording video, else returns

VLcol

generates a simple colormap related to the basic Matlab colors

Out: **cm**: Colormap

VLELCILplots

plots contours given as vertex list and edge list

In: **VL**: Vertex list **EL**: Edge list **CIL**: Contour index list **c**: Optional color of desired edges; default os 'r*-' **w**: Optional line width; default is 1 **t**: Size of a cone shaped arrow tip, default is 0 **b**: Size of a cube showing the start point of a contour; default is 0 **f**: Size of a ring showing the end point of a contour; default is 0

VLELplot

plots lines given as vertex list CVL and edge list CLL

In: **CVL**: Vertex list **CEL**: Edge list **c**: Optional color of desired edges; default os 'r*-' **NL**: Optional edge normal vector list; default is [] **cn**: Optional color of normal vectors; default is 'b-' **VNL**: Optional vertex normal vector list; default is [] **cv**: Optional color of vertex normal vectors; default is 'm-'

VLELplots

plots lines given as vertex list and edge list

In: **VL**: Vertex list **EL**: Edge list **c**: Optional color of desired edges; default os 'r*-' **w**: Optional line width; default is 1 **t**: Size of a cone tip, default is ''; 0 == automatic size **b**: Size of a start point cube; default is ''; 0 == automatic size **f**: Size of a end point ring; default is ''; 0 == automatic size

Out: **h**: vector of graphics handles of the plotted lines and markers

VLFLcamlight

set the current light source on the position of the viewpoint

VLFLELplot

plots an edge list and all attached facets of the edges

In: **VL**: Vertex list **FL**: Facet list **PEL**: Edge list

VLFLfaceNormalplot

plots normal vector in appropriate length and writes the name of the facet at the tip

In: **VL**: Vertex list **FL**: Facet List **c**: Color, default is magenta **t**: Thickness; default is 2 **sl**: optional list of selected points

Out: **h**: handle to modify or delete text if necessary **FN**: list of facet normal vectors **IC**: list of facet incenter points

VLFLofPatch

returns vertex list and facet list of current figure or the current object

In: **h**: current object; default is gca's patches

Out: **VL**: Vertex list **FL**: Facet list **h**: current object

VLFLplot

plots a vertexlist and facet list by fill3 in a figure

In: **VL**: is the vertex list **FL**: is the facet list **c**: is the desired color **fe**: Thickness of feature edges **nl**: length of normal vector; default is 0;

Out: **h**: handle

VLFLplot4

plots a vertex/facet list by fill3 in a figure with 4 Subplots

In: **VL**: is the vertex list **FL**: is the facet list **c**: is the desired color

VLFLplotfacets

plot a facet list as separate independent facets

In: **VL**: Vertex list or SG **FL**: Facet list

Out: **h**: handle list of all facets

VLFLplotfi

plots the scenery around some vertices in unequal scale

In: **VL**: Vertex list **FL**: Facet list **fi**: facet index

VLFLplotlightoff

switches the camlights on or off

VLFLplotmanifold

plots a the non manifold points, edges, and facets of a solid

In: **VL**: Vertex list **FL**: Facet list

VLFLplots

plots an object/solid as shaded or faceted object

In: **VL**: Vertex list **FL**: Facet list **c**: Color; default is 'r' **fe**: feature edge list

Out: **X**: handle to trisurf

VLFLplots4

plots an object/solid as shaded or faceted object in 4 views

In: **VL**: Vertex list **FL**: Facet list **c**: Color

VLFLshow

simply brings the current figure in the front

VLofgcalines

creates a list of vertex points used in

Out: **VL**: Vertex list of the line points in the current axes

VLOLplot

prints one or all surfaces of an object list

In: **VL**: Vertex list **OL**: Object list containing different objects **i**: Number of object to print; default=0=all

VLplot

Plots different views of a vertex list (nx3) as contour lines or individual points.

In: **VL**: Vertex list **c**: optional color; default is 'r.-' **w**: plot size **vec**: if v>0, vectors are plotted from the origin

Out: **h**: handle

VLplotdelaunay

simple inspection fnct

In: **VL**: Vertex list **c**: color of solid; default is 'y' **al**: alpha of solid; default is 1

VLplotmotion

draws position, speed and acceleration of a VL

In: **VL**: Vertex list **i**: optional angle list

VLscatter

plots a vertex list with a corresponding image intensity list

In: **VL**: Vertex list **C**: Intensity List (Hounsfield) **n**: step size for vertex selection **cmap**: optional colormap; such as 'gray'

Out: **h**: handle to plot

VLULplot

Plots the union list of an object, union face by union face

In: **VL**: Vertex list of the object **UL**: Union list of the object **c**: color of unions **Amin**: Minimal size to plot **p**: delay between unions; -1 == step by keyboard

zoompatch

zooms to see all patches of the current figure axis

In: **VL**: Scalar or Vertex list or Solid Geometry; default is gca **fac**: surrounding area; default is 0.5, relative to Bounding Box **md**: minimal distance; default is 1; distance in mm

zoomsubplot

opens a figure and zooms the subplot image

2.3 Plot Annotation [guide]

Functions in this section:

textCVL — plots descriptors for each contour into a figure using text command

textP — draws a texlabel text at the position

textT — drawn a text at the position of a HT matrix

textVL — plots descriptors for each vertex in the figure using the text command

textVLFL — plots descriptors for each facet in the figure using the text command

textVLFS — plots descriptors for each feature surface and the corresponding feature edges

Detailed description

textCVL

plots descriptors for each contour into a figure using text command

In: **CVL**: Contour list separated by Nan (CPL/CVL) **c**: Color, default is black **s**: Size, default is 16 **nt**: optional list of selected contours **lb**: optional string for text; default is 'C'

Out: **h**: handle

textP

draws a texlabel text at the position

In: **p**: point **str**: texlabel **delx**: displacement x; default is 0.02 (2%) **dely**: displacement y; default is 0.02 (2%) **Fnts**: Fontsize; default is 16 **col**: text color; default is 'k'

Out: **h**: handle to text

textT

drawn a text at the position of a HT matrix

In: **T**: T matrix, or array (:,:,3) or cell array **TStr**: Text string **c**: color **s**: font size

Out: **ha**: handle to decriptors

textVL

plots descriptors for each vertex in the figure using the text command

In: **VL**: Vertex list **c**: Color, default is black **s**: Size, default is 10 **nt**: optional list of selected points **lb**: optional string for text; default is 'F'

Out: **h**: handle to graphics object

textVLFL

plots descriptors for each facet in the figure using the text command

In: **VL**: Vertex list **FL**: Facet List **c**: Color, default is black **s**: Size, default is 10 **nt**: optional list of selected points **lb**: optional string for text; default is 'F'

Out: **h**: handle to modify or delete text if necessary

textVLFS

plots descriptors for each feature surface and the corresponding feature edges

In: **VL**: Vertex list **FL**: Facet List **FSi**: Feature surface index list; FSi = surfacesofSG(SG,fea); **c**: Color, default is black **s**: Size, default is 10 **nt**: optional list of selected points **lb**: optional string for text; default is 'F'

Out: **h**: handle to modify or delete text if necessary

2.4 Crossing points and tangents [guide]

Functions in this section:

angleintersect — intersect two angle areas

angleinterv — shifts an angle intervall table vainto a fixed interval such as [-pi..+pi] or [0 2*pi]

angleshift — shifts an angle value into a fixed interval such as [-pi..+pi] or [0 2*pi]

BBaddtolerance — increases the size of a Boundary Box

BBconcat — concatenate two bounding boxes

BBgrow — increases the size of a Boundary Box

BBofBB — scales an axis or bounding box

BBofCPS — returns the bounding box for a polyshape or CPS

BBofPose — returns the bounding box of a Pose

BBofPS — returns the bounding box of a polyshape

BBofSGcutT — returns the bounding box for a cutting frame

BBofT — returns the BB for a list of matrices

circlesegmentofPL — returns the angle intervall of a contour PL from a view point A0.

corner2rad — returns the radius based on the distance to the two tangents

CPLisclosedCPL — return whether the contours are open or closed in a CPL

CPLisstraightline — returns whether a CPL is a straight line with area zero

cross2circ — returns the crossing points of two circles

cross2edges — calculates in 2D the crossing point of two straight lines

cross2edges2 — calculates in 2D the crossing point of two straight lines

cross2L — returns the crossing point of two straight lines

cross2T — returns 3 crossing points of plane T1 with the plane T2

cross4P — return whether two lines cross each other (2D fnct)

crosscircline — returns the cross points of a circle and a straight line

crossL — returns the crossing points of a straight line with the main planes where z=0, x=0 or y=0

crosspoints2F — This Fnct designed for SGbool6 returns all crossing points of 2 triangles

crosspointsVLFLofT — returns the crossing points of the axis of a coordinate system

crosspointVLFLfi — returns the self crosspoints of a surface normal vector

crossT — returns 3 crossing points of a plane TE with the x/y-plane (z=0)

crosstest — return whether a straight line right from point a crosses a line between pa and pb

crosstest2 — return whether a straight line right from point a crosses a line between pa and pb

crossz — calculates in 2D only the z-part of the cross product and handles values near zero

diffanglew — returns the shortest distance from angle 1 to angle 2

dofn — returns the radius relation between a radius and a n-polygon

ELangleofLATAP — returns the bending angle for a catheter based on LAT and AP angles

findCollinearLine — returns the indices of the extrem points of a straight line

findfacetsinSG — creates a correlation index list of the facets in FLA and FLB

findfacetsinVLFL — creates a correlation index list of the facets in FLA and FLB

insidePLELdelaunay — returns whether a point is inside/outside a contour

intersectstriangle — fundamental fnct for the intersection between a ray and a triangle

intersectstriangle46 — fundamental fnct for the intersection between a ray and a triangle

isincirclesegment — returns whether an angle or a angle list is inside an angle interval

isintriangle2 — returns whether a point is in a triangle

isonVLFLplanes — returns whether a point is on a plane of a facet not necessarily inside of the facet

isonVLFLsurface — returns whether a point is on a surface

issameside — return whether 2 points p1,p2 are on the same side of a line through A to B

PCfindradiusangle — returns vertices and facets of surface near a point of a point cloud

phiofsr — returns a angle difference to achieve a slot size

PLchordof2PR — returns a radial segment for a given chord / arc tendon and a radius

PLcircle — returns a 2D point list of a circle or circle segment

PLcross2Lines — returns the crossing point of 2 lines in [x y]

PLcrossstraightnline — returns the crossing point of 2 lines in [x y]

PLlength — simply the cummulated distance between points of an open list

PLof3dist — creates a 3P Tracker geometry using the given distances

PLshortestpathinCPLcost — returns a path inside of a CPL using A* Zelinzky, Lozano-Perez, Dikstra

PLsquareui — allows to draw interactively a square in the current graphics area

rofcircbend — returns bending radius and angle for a compressed line

rofcirclearoundCPL — returns radius and center point of the surrounding and the most inner circle

rofcircleinsideCPL — returns radius and center point of maximum inner circle and the of the minimum surrounding circle

rofCPLsmalldist — returns the smallest distance between two contours in CPL

rofCPLsweep — returns the radius of a fully circled CPL

SGisInterior — returns the isInside Flag for a SG and a VL

SGisInteriorofBB — returns all facets of a solid that are within a bounding box

SGisIntersectingBB — returns all facets of a solid that are crossed/intersected by a bounding box

SGisOnsurface — returns whether a point is on a surface

sofrd — returns the minimal auxiliary point distance on a circular curved contour

tangent2circ — returns outer and inner tangents of 2 circles

tangente — returns the point for a tangente

tangentRP — returns the two tangential points of a circle

Tcross2T — returns the crossing line HT of two defined planes T1 and T2

ticksofBB — sets the ticks using ticks of minmax of a BB

TofBB — returns frames relative to bounding boxes

triangleheight — returns the heights of a triangle - slow but correct

VLcross2Lines3D — returns the crossing point of 2 lines in [x y z]

VLcrossstraightnline3D — returns the crossing point of 2 lines in [x y z]

VLFLfaceneighborangle — returns a signed (convex/concave) normal vector difference

VLFLfaceneighborswapangle — returns and explains swapping decisions

VLisInteriorofBB — returns the index point list for isinterior

VLisonCVL — returns whether a point is on a segment of a CVL or CPL

VLseparate — wish to solve boolean opeartion problems

VLtangentcirc — creates a vertex list for a tangential circle segment

volofBB — simply returns the volume of a bounding box or a cell list of bounding boxes

wofcross2circ — just returns the angles of the two crosspoints

zofBB — simple fnct to create a height of a bounding box

Detailed description

angleintersect

intersect two angle areas

In: **WINA**: angle interval A **WINB**: angle interval B

Out: **WINC**: Intersection of angle interval

angleinterv

shifts an angle intervall table vainto a fixed interval such as [-pi..+pi] or [0 2*pi]

In: **wins**: List of angle delta values [wmin wmax] **ival**: [-pi..+pi] or [0 2*pi]

Out: **wins**: Values of wins(:,1) in the allowed ival; no delta > 2*pi

angleshift

shifts an angle value into a fixed interval such as [-pi..+pi] or [0 2*pi]

In: **AL**: List of angle values **ival**: [-pi..+pi] or [0 2*pi]; or [ival ival+2*pi]; default is [0 2*pi]

Out: **AL**: Alle values inside the allowed interval

BBaddtolerance

increases the size of a Boundary Box

In: **BBL**: bounding box or axis **tol**: growing size

Out: **BBG**: new bounding box

BBconcat

concatenate two bounding boxes

In: **BBA**: Bounding box B or solid A **BBB**: Bounding box B or solid A

Out: **BBC**: New boundin box [xmin xmax ymin ymax zmin zmax]

BBgrow

increases the size of a Boundary Box

In: **bb**: bounding box or axis **m**: growing factor

Out: **nbb**: new bounding box

BBofBB

scales an axis or bounding box

In: **ax**: Bounding box **d**: scaling vector

Out: **nx**: new exis

BBofCPS

returns the bounding box for a polyshape or CPS

In: **CPS**: Colored polyshape struct or polyshape

Out: **bb**: bounding box

BBofPose

returns the bounding box of a Pose

In: **PS**: Pose (frame) or list of poses

Out: **bb**: bb [xmin xmax ymin ymax]

BBofPS

returns the bounding box of a polyshape

In: **ps**: polyshape

Out: **bb**: bounding box

BBofSGcutT

returns the bounding box for a cutting frame

In: **SGx**: Solid Geometry **Tx**: Cutting frame (cuts above z==0) **z**: size [-z/2 +z/2] or [zmin zmax] **d**: displacement to add to x and y values of BB; z unchanged **ct**: default is '-a' in interactive mode

Out: **BB**: Bounding box size **Ts**: Cutting frame (if interactively used) on surface **Tc**: Cutting Frame in center point **SGC**: Cutted Solid Geometry of [zmin zmax] **aew**: view angle in interactive mode

BBofT

returns the BB for a list of matrices

In: **T**: List of 3x3 or 4x4 HT matrices

Out: **BB**: [xmin xmax ymin y max zmin zmax]

circlesegmentofPL

returns the angle intervall of a contour PL from a view point A0.

In: **A0**: Center or view Point **PL**: Contour to be analyzed

Out: **wlim**: angle interval

corner2rad

returns the radius based on the distance to the two tangents

In: **d**: distance

Out: **r**: radius

CPLisclosedCPL

return whether the contours are open or closed in a CPL

In: **CPL**: CPL format contours

Out: **fot**: overall result **ol**: index list of open contour

CPLisstraightline

returns whether a CPL is a straight line with area zero

In: **CPL**: SINGLE CPL to analyze

Out: **log**: logical result; true if straight line

cross2circ

returns the crossing points of two circles

In: **A0**: Center of circle A **B0**: Center of circle B **ra**: Radius of circle A **rb**: Radius of circle B

Out: **Q1**: Solution right hand to A0B0 for positive x **Q2**: Solution left hand to A0B0 for positive x **x**: distance from A0 to B0 **y**: orthogonal from A0+e*x **e**: unit direction vector **o**: unit orthogonal vector

cross2edges

calculates in 2D the crossing point of two straight lines

In: **pa**: point a **na**: direction vector a **pb**: point b **nb**: direction vector b

Out: **cp**: crossing point **ka**: factor of direction vector a **kb**: factor of direction vector a

cross2edges2

calculates in 2D the crossing point of two straight lines

In: **pa**: point a **na**: direction vector a **pb**: point b **nb**: direction vector b

Out: **cp**: crossing point **ka**: factor of direction vector a **kb**: factor of direction vector a **kd**: +1 == infinity solutions, -1; no solution; 0 there is a solution

cross2L

returns the crossing point of two straight lines

In: **TG1**: HT-Matrix of line 1, the z axis is crossed **TG2**: HT-Matrix of line 2, the z axis is crossed **thr**: distance threshold below which the two lines are treated as crossing; default is 1e-3

Out: **p**: crossing point/minimal distance point (1x3) of the two lines **d**: minimal distance between the two line if they don't cross

cross2T

returns 3 crossing points of plane T1 with the plane T2

In: **T1**: HT matrix 1 (plane), that describes a plane by an ez-Vector and a position p **T2**: HT matrix 2 (plane), at describes a plane by an ez-Vector and a position p

Out: **PL**: is a list of 3 points PL=[px py pz] describing the crossing points of ex, ey, and ez vectors with the x/y plane (z=0).

cross4P

return whether two lines cross each other (2D fnct)

In: **pa**: start point of first line **pb**: end point of first line **pc**: start point of second line **pd**: end point of second line

Out: **kc**: =0, i.e. there is no crossing. =1; i.e. there is a crossing **ka**: between 0 and 1 if there is a crossing of line 1 **kb**: between 0 and 1 if there is a crossing of line 2

crosscircline

returns the cross points of a circle and a straight line

In: **C**: Center of circle [x y] **R**: Radius of circle **p**: point on straight line **ez**: direction of straight line

Out: **P**: List of 2 Points **K**: List of 2 K values **W**: List of two angle

crossL

returns the crossing points of a straight line with the main planes where z=0, x=0 or y=0

In: **T**: HT-Matrix of a straight line

Out: **PL**: List of 3 Points [px py pz] of the line thorugh the main planes **k**: vector [kx ky kz] of line parameters for the crossing points with the y/z, x/z, and x/y planes

crosspoints2F

This Fnct designed for SGbool6 returns all crossing points of 2 triangles

In: **p1**: Vertex 1 of A; or a VL=[p1;p2;p3;pa;pb;pc] **p2**: Vertex 2 of A **p3**: Vertex 3 of A **pa**: Vertex 1 of B **pb**: Vertex 2 of B **pc**: Vertex 3 of B **SHW**: show additional information if true

Out: **XP**: Crosspoints of Edges [edgeA edgeB X Y Z dir] similar to cross2F but handles also co-planar points **PinA**: Additional Points in A if both traingles co-planar (may be not in XP) **PinB**: Additional Points in B if both traingles co-planar (may be not in XP) **EL1**: Edges between additional points in A (MAGENTA) **EL2**: Edges between additional points in B (CYAN)

crosspointsVLFLofT

returns the crossing points of the axis of a coordinate system

In: **VL**: Vertex List of Solid **FL**: Facet List of Solid **T**: Coordinate Frame [4x4] **d**: distance to shift the frame along its z axis; default is 0; 'center' keeps 0

Out: **TVL**: Touch vertex list [6 x 3] **td**: distances between touch points [dx dy dz] **np**: center of crossing lines **TN**: Optional moved Frame matrix **FLi**: Facet index list corresponding to TVL

crosspointVLFLfi

returns the self crosspoints of a surface normal vector

In: **VL**: Vertrex list or SG **FL**: Facet List **fi**: facet index **back**: if true, the surface normal direction is reversed for the search; default is false

Out: **cp**: cross point on surface, empty if there is no crossing point **t**: distance (t * ez) **fic**: facet index of 1st crossed surface **fid**: full list of facet index and distance [fi, d]

crossT

returns 3 crossing points of a plane TE with the x/y-plane (z=0)

In: **TE**: is the HT matrix that describes a plane by an ez-Vector and a position p

Out: **PL**: is a list of 3 points PL=[px py pz] describing the crossing points of ex, ey, and ez vectors with the x/y plane (z=0). **dk**: vector [kx ky kz] of line parameters where the ex, ey, and ez vectors cross the x/y plane; NaN if an axis is parallel to the plane

crosstest

return whether a straight line right from point a crosses a line between pa and pb

In: **pa**: point to test **pb**: start point of line **pc**: end point of line

Out: **t**: -1==cross; 0== on the line; +1 nothing known

crosstest2

return whether a straight line right from point a crosses a line between pa and pb

In: **p**: point to test **pa**: start point of line **pb**: end point of line

Out: **t**: -1==cross; 0== on the line; +1 nothing known

crossz

calculates in 2D only the z-part of the cross product and handles values near zero

In: **pa**: vector a **pb**: vector b

Out: **z**: z value of the cross product **s**: sign of (z)

diffanglew

returns the shortest distance from angle 1 to angle 2

In: **w1**: angle w1 **w2**: angle w2

Out: **dw**: shortes turning angle from w1 to w2

dofn

returns the radius relation between a radius and a n-polygon

In: **n**: number of edges

Out: **m**: magnification of the point distance **id**: 1/d reduction of the edge distance

ELangleofLATAP

returns the bending angle for a catheter based on LAT and AP angles

In: **Wlat**: LAT angle in degree **Wap**: AP angle in degree

Out: **EL**: Elevation bending angle

findCollinearLine

returns the indices of the extrem points of a straight line

In: **PL**: Point list **tol**: default is 1e-6 (cross product)

Out: **iscoll**: true or false **si**: start index **ei**: end index **ii**: sorting index from start to end point to sort PL

findfacetsinSG

creates a correlation index list of the facets in FLA and FLB

In: **SGA**: Solid A **SGB**: Solid B **order**: if true, match facets in any vertex order; default is false **flips**: if true, match both front- and back-facing orientations; default is false

Out: **cc**: facet correlation index list [ai bi]

findfacetsinVLFL

creates a correlation index list of the facets in FLA and FLB

In: **VLA**: Vertex list Solid A **FLA**: Facet list Solid A **VLB**: Vertex list Solid B **FLB**: Facet list Solid B **order**: if true the vertex order is ignored; default is false **flips**: if true the facet direction is ignored; default is false

Out: **cc**: facet correlation index list [ai bi]

insidePLELdelaunay

returns whether a point is inside/outside a contour

In: **PL**: 2D Point list **EL**: Contour edge list **GPL**: Grid point list or Point list **csplit**: false=no split points allowed; true=split points allowed; default is false

Out: **ki**: +1==inside, -1==outside, 0== on the contour **XVL**: Point List **XFL**: Face List **XEL**: Edge list

intersectstriangle

fundamental fnct for the intersection between a ray and a triangle

In: **p0**: Point of line **d**: Direction of line **p1**: Vertex 0 **p2**: Vertex 1 **p3**: Vertex 2

Out: **b**: if t>0 and intersecting point exists **t**: intersecting point is p+t*d

intersectstriangle46

fundamental fnct for the intersection between a ray and a triangle

In: **p0**: Point of line **d**: Direction of line **p1**: Vertex 0 **p2**: Vertex 1 **p3**: Vertex 2

Out: **b**: if t>0 and intersecting point exists **t**: intersecting point is p+t*d

isincirclesegment

returns whether an angle or a angle list is inside an angle interval

In: **cseg**: angle interval; second values must be larger than first value **WL**: angle list

Out: **ins**: logical table which angle is inside

isintriangle2

returns whether a point is in a triangle

In: **p**: point to test **A**: Point A of triangle **B**: Point B of triangle **C**: Point C of triangle

Out: **b**: True if the point lies inside the triangle

isonVLFLplanes

returns whether a point is on a plane of a facet not necessarily inside of the facet

In: **VLA**: Vertex list **FLA**: Facet list **VL**: Vertex list to test **tol**: default is 1e-5

Out: **vi**: vertex index list true/false **il**: cell list of facets for each vertex

isonVLFLsurface

returns whether a point is on a surface

In: **VLA**: Vertex list **FLA**: Facet list **VL**: Vertex list to test

Out: **vi**: On Surface Index of VLT

issameside

return whether 2 points p1,p2 are on the same side of a line through A to B

In: **p1**: point to test **p2**: point to test **A**: Point A of line **B**: Point B of line

Out: **b**: result

PCfindradiusangle

returns vertices and facets of surface near a point of a point cloud

In: **pc**: point cloud **i**: index in point cloud **amax**: allowed angle deviation of vertex in radiant; default is pi/4 **rmax**: radius to consider around vertex i; default is 10000

Out: **VL**: Vertex list **FL**: Facet list

phiofsr

returns a angle difference to achieve a slot size

In: **s**: slot size **R**: minimal radius

Out: **phi**: angle to add

PLchordof2PR

returns a radial segment for a given chord / arc tendon and a radius

In: **PL**: [starting point of chord; end point of chord] **R**: Radius **rhand**: true = right hand; false = left hand; default is true **nseg**: number of segments for the arc; default is empty

Out: **PL1**: short angle (red) **PL2**: long angle (green) **m1**: center on left side **m2**: center on right side **a1**: angle of the short arc in radians **a2**: angle of the long arc in radians (pi-a1)

PLcircle

returns a 2D point list of a circle or circle segment

In: **R**: Radius or list of radii of concentric circles (Rev. 5.0) **nf**: number of facets (default is nofrd (R,0.05)) **dw**: circle segment]0..2*pi] (default is 2*pi) **Ry**: Radius of ellipse (default is R) **ow**: Offset angle

Out: **PL**: Point / Vertex list **w**: vector of angles in radians used to generate the point list

PLcross2Lines

returns the crossing point of 2 lines in [x y]

In: **A**: starting point A [x y] or PL[4 x 2 **B**: ending point A [x y] **C**: starting point B [x y] **D**: ending point B [x y]

Out: **CP**: Crossing Point **ka**: factor from A to B **kb**: factor from C to D **kc**: reason for removal

PLcrossstraightnline

returns the crossing point of 2 lines in [x y]

In: **A**: starting point A [x y] or PL[4 x 2 **B**: ending point A [x y] **C**: starting point B [x y] **D**: Direction from B to [x y]

Out: **CP**: Crossing Point **ka**: factor from A to B **kb**: factor from C into direction D; always > 0 **kc**: reason for removal

PLlength

simply the cummulated distance between points of an open list

In: **b**: Point list

Out: **ll**: length of open list

PLof3dist

creates a 3P Tracker geometry using the given distances

In: **d1**: distance in x **d2**: radius at second point **d3**: radius at third/first point point; **srt**: sort by length; default is false

Out: **PL**: Point list **n**: area of triangle **d3**: d3

PLshortestpathinCPLcost

returns a path inside of a CPL using A* Zelinzky, Lozano-Perez, Dikstra

In: **CPL**: Outer contour with holes as forbidden areas **A**: Starting Point A inside the CPL **B**: End Point B inside the CPL **shr**: shrinkage value; default is sofBB/n **n**: number of grid points; default is 100 **smo**: 'A*', 'Bezier', 'fft' for smoothening **CPLG**: No cost area; predefined rack area; default is empty **cost**: predfined cost for entering GPL; default is 0 therefor GPL is preferred

Out: **PLB**: Bezier smoothed path **PL**: Grid bases path **k**: grid numbers **shr**: dynamic shrinkage value

PLsquareui

allows to draw interactively a square in the current graphics area

Out: **CPL**: PL of a square

rofcircbend

returns bending radius and angle for a compressed line

In: **s**: original distance **l**: compressed distance

Out: **r**: bending radius **a**: bending angle **FPL**: Point list describing the shape

rofcirclearoundCPL

returns radius and center point of the surrounding and the most inner circle

In: **CPLZ**: CPL **"sep"**:

Out: **ro**: Outer radius **cp**: Center point **ri**: Inner radius **cpi**: Center point of the inner circle

rofcircleinsideCPL

returns radius and center point of maximum inner circle and the of the minimum surrounding circle

In: **CPLZ**: CPL to analyze **"sep"**: if used, the CPLs is separated and the r values are listst / array

Out: **ri**: Maximum inner radius (not unique) **cpi**: Center point of the most inner radius (not unique) **ro**: Smallest outer radius **cp**: Center point of the minimal outetr radius

rofCPLsmalldist

returns the smallest distance between two contours in CPL

In: **CPL**: Contour

Out: **rmin**: smallest distance between contours in CPL

rofCPLsweep

returns the radius of a fully circled CPL

In: **CPL**: Original contour **rp**: center point for rotation; default is [0 0] **cp**: optional center point if CPLC rotates independently

Out: **Rmax**: Maximal Radius **Rmin**: Minimum Radius **CPLC**: Sweeped Contour

SGisInterior

returns the isInside Flag for a SG and a VL

In: **SG**: Solid Geometry or cell list of SG **VL**: Vertex list to test or 4x4 Frame

Out: **VIL**: Vertex index list **Vin**: Vertex inside not on surface **Von**: Vertex on surface not inside

SGisInteriorofBB

returns all facets of a solid that are within a bounding box

In: **SG**: Solid Geometry **bb**: bound bog or second solid **T**: Transformation matrix for bb

Out: **SGi**: Solid that has vertices inside of the BB **si**: facets of SG (not SGi) with all vertices inside of BB **ci**: facets of SG (not SGi) that belong to the same surfaces of si **SIL**: Sort Selection and Index list used to create SGi using VLFLselect

SGisIntersectingBB

returns all facets of a solid that are crossed/intersected by a bounding box

In: **SG**: Solid Geometry **bb**: bound bog or second solid **T**: Transformation matrix for bb

Out: **SGi**: Solid that has faces that are crossed/intersected by the BB **si**: facets of SG (not SGi) that are crossed/intersected by BB **ci**: facets of SG (not SGi) that belong to the same surfaces of si **SIL**: Sort Selection and Index list used to create SGi using VLFLselect

SGisOnsurface

returns whether a point is on a surface

In: **SG**: Solid Geometry **VL**: Vertex list to test

Out: **vi**: On Surface Index of VL

sofrd

returns the minimal auxiliary point distance on a circular curved contour

In: **r**: Radius

Out: **s**: maximial step size

tangent2circ

returns outer and inner tangents of 2 circles

In: **A0**: Center of circle A **B0**: Center of circle B **ra**: radius of circle A **rb**: radius of circle B

Out: **A**: Positions outer 1 outer 2 inner 1 inner 2 of circle A [4 x 2] **B**: Positions outer 1 outer 2 inner 1 inner 2 of circle B [4 x 2] **exey**: [ex;ey vector starting at A0 towards B0 **w**: angle solutions relative to ex and ey

tangente

returns the point for a tangente

In: **x0**: x of circle center **y0**: y of x of circle center **x1**: x of interest point **y1**: x of interest point **R**: Radius of circle

Out: **x**: of a circle touch **y**: of a circle touch

tangentRP

returns the two tangential points of a circle

In: **R**: Radius of the circle **p**: point to match **pc**: optional center of the circle; default is [0 0]

Out: **q1**: point 1 (green) **q2**: point 1 (magenta) **w1**: angle of q1 in world coordinates relative to pc **w2**: angle of q2 in world coordinates relative to pc

Tcross2T

returns the crossing line HT of two defined planes T1 and T2

In: **T1**: HT matrix 1 (plane), that describes a plane by an ez-Vector and a position p **T2**: HT matrix 2 (plane), at describes a plane by an ez-Vector and a position p

Out: **TX**: HT matrix of the resulting straight line

ticksofBB

sets the ticks using ticks of minmax of a BB

In: **bb**: bb of Solid or SG **dis**: distance between ticks **res**: resolution default is .1

TofBB

returns frames relative to bounding boxes

In: **BB**: Bounding Box or cell list of Bounding boxes **listofrel**: list of relation such as 'under', {disp, ez, rotz}

Out: **T**: Frame or cell list of frames

triangleheight

returns the heights of a triangle - slow but correct

In: **v1**: vertex 1 or Vertex list **v2**: vertex 2 **v3**: vertex 3

Out: **h**: [h1 h2 h3] **cp**: foot points of the three heights on the opposite edges

VLcross2Lines3D

returns the crossing point of 2 lines in [x y z]

In: **A**: starting point A [x y] or PL[4 x 2 **B**: ending point A [x y] **C**: starting point B [x y] **D**: ending point B [x y]

Out: **CP**: Crossing Point **ka**: factor from A to B **kb**: factor from C into direction D; always > 0 **kc**: reason for removal

VLcrossstraightnline3D

returns the crossing point of 2 lines in [x y z]

In: **A**: starting point A [x y z] or PL[4 x 2 **B**: ending point A [x y z] **C**: starting point B [x y z] **D**: Direction from C to [x y z]

Out: **CP**: Crossing Point **ka**: factor from A to B **kb**: factor from C into direction D; always > 0 **kc**: reason for removal

VLFLfaceneighborangle

returns a signed (convex/concave) normal vector difference

In: **VL**: Vertex list **FLi**: single face row **FLn**: up to 3 rows for neighbor facets

Out: **w**: angles related to the FLn

VLFLfaceneighborswapangle

returns and explains swapping decisions

In: **VL**: Vertex List **FL**: Facet List **fi**: face index to swap **k**: index (1..3) to swap **NN**: seeVLFLtrianglegeometry **w**: see VLFLtrianglegeometry **A**: see VLFLtrianglegeometry

Out: **www**: angle information **ci**: used neighbor to fi == NN(fi,k) **i4**: vertex index for new triangles

VLisInteriorofBB

returns the index point list for isinterior

In: **BB**: Solid Geometry or BB **VL**: Vertex List

Out: **vi**: index list for isInterior: true = isinterior

VLisonCVL

returns whether a point is on a segment of a CVL or CPL

In: **CVL**: CPL or CVL **PL**: PL or VL **tol**: if tol > 0; function distanceVLtoVLEL is used; default is 0; much faster with 0

Out: **ci**: index of PL **tp**: matrix of PL and segment list of CVL

VLseparate

wish to solve boolean opeartion problems

In: **VL**: Vertex list **rr**: random increasement **dim**: default is [1 2 3]

Out: **VL**: Vertex list with unique vertices

VLtangentcirc

creates a vertex list for a tangential circle segment

In: **v1**: vector 1 from the origin **v2**: vector 2 from the origin **R**: desired Radius; automatically reduced; default is 1 **M**: maximize radius if possible; default is false **ty**: edge type 'rad', 'rma', or 'tan'; default is 'rad'

Out: **VL**: Vertex list describing a circle segment **Vc**: Circle center of circle segment **Rmin**: Radius that was really possible

volofBB

simply returns the volume of a bounding box or a cell list of bounding boxes

In: **BB**: Bounding Boxes

Out: **V**: Volume

wofcross2circ

just returns the angles of the two crosspoints

In: **A0**: Center of circle A **B0**: Center of circle B **r1**: radius of circle A **r2**: radius of circle B

Out: **w1**: absolute angle of Q1 angle relative to A0 **w2**: absolute angle of Q2 angle relative to B0 **dw**: angle of direction B0-A0 **Q1**: cross point 1 **Q2**: cross point 1

zofBB

simple fnct to create a height of a bounding box

In: **BB**: BB or solid **inf**: dimension request string (x,y,z,bbx,xmax,zmin,wtop,...); default is 'z'

Out: **z**: height of a bounding box

2.5 Planar and Spatial Transformation [guide]

Functions in this section:

barycartNT — returns the vertex list (3D) of a point list (2D) relative to one facet

barytocartUVL — converts a 2d Barycentric [u,v] vertex list into a 3D Cartesian coordinate list [x, y, z]

barytocartUVLdelaunay — converts a 2d Barycentric [u,v] vertex list into a 3D Cartesian coordinate list [x, y, z]

cartbaryNT — returns the 2D barycentric coordinates of a VL relative to one facet

carttobaryVL — converts a 3D planar vertex list into a 2D barycentric coordinate list [u,v]

CPLtransform — finds contours inbetween two sets of CPLs => CVLof2CPLzcorrelate

CPLtransrelCPL — relative transformation of CPLs

CPLtransrelT — transform the pose of a CPL from one coordinate system into another

diffTincz — returns for Frames the relative incliniation of the ez vector of B wrt to A

diffTrotz — returns for Frames the relative rotation of the ez vector of B wrt to A

dist2line — mirrors a point at a straight line in 2D

ezof3P — returns ez and R for 3 Point Rotation Matrix

mirroringatline — mirrors a point at a straight line in 2D

mirroringatline3D — mirrors a point at a straight line in 3D

planedistanceofT — returns the distance of a vertex list to a plane in HT coordinate form

planeparamofT — returns the plane parameters from a HT matrix (x/y plane) - ax+by+cz+d=0

plotT — plots a coordinate system given as HT matrix

PLtrans — multi transformation modes for a point list

PLtrans0 — returns an origin centered point list

PLtrans1 — returns a point list moved into the first quadrant

PLtransC — rotates a point list around a center

PLtransP — returns a translated point list

PLtransR — rotates all point of a point list

PLtransT — rotates and moves a point list nx2

PrelCPL — returns a point position given by a spatial description relative to a CPL

RL2exl — returns X vectors for a list of euler angles

RLofEulerInterpolation — Returns euler rotations as linear interpolation

RLplot — plots along the z axis the systems of a Euler Rotation List

Rofez — returns a rotation matrix based on a list of ez vector

Rofrdintrusion — provides the minimum intrusion distance to fully insert a cylinder into a sphere

Rofxy — Rotation matrix by the ez vektor and and optional rotation angle

rot — returns rotation matrix in 3x3 as Rx*Ry*Rz or 2x2 Rz

rotdeg — returns 3x3 rotation matrix for Deg angles in x, y, z

RPofTrelT — returns the frame T2 in relation to frame T1

screw2R — definition of a screw coordinate system

SGtransrel — returns a spatial transformed solid using SGtransrelSG

SGTtrans0 — move the center of the solid into the origin of the coordinate system

TflipR — turns the current right hand system into an inverted left hand system

tfplot — plots an HT matrix including the x/y plane

THLof2F — returns Tetrahedron list of two facets

TLofCVL — returns a list of frames that connect a start frame and an end frame along a spatial contour

Tof2vec — returns transformation matrix based on the 2 point movements

TofcamVLPL — returns the coordinate system and calibration matrix of a camera

TofDPhiH — returns a 4x4 transformation matrix

Tofgca — returns the HT matrix of the current camera position

TofP — returns a T matrix from a point in 2D or 3D

TofPcam — returns a transformation matrix using a vertex and a the current camera view

TofPEul — return from euler angle and position an HT matrix

TofPez — creates a T matrix from point and ez-Vector

TofSG — sets a HT matrix relativ to nested solid

TofSGML — returns HT for a mounting face with known number

TofSGMLez — returns HT for a mounting face with known ez

TofT — returns a T matrix defined relative to another T matrix

TofTrel — creates a HT matrix relative to a frame

TofVL — returns the main axis of a vertex list or SG

TPL — return for vertices of a patch the HT Matrix of the patch center

tplot — plots a coordinate system given as HT matrix

Ttransrel — language based fnct for spatial arrangement of frames

TtransT — returns a matrix multiplication of a (list of) homogenous transformation matrix

Txy — returns a matrix for a point p and pitch-yaw values of ez

VLinsertEulerSteps — inserts vertex points on straight lines if the twist angle is to large

VLplaneprojection — returns a PL as projection of a vertex list to a orthogal plane defined by an ez vector

VLprojection2SG — projects a Vertex list in parallel projection to a solid

VLswapX — returns a 2D or 3D point/vertex list, mirrored at the x-Axis

VLswapXY — x-axis becomes y-axis of a VL (z-rotation by 90 degree)

VLswapXZ — x-axis becomes z-axis of a VL (y-rotation by +90 degree)

VLswapY — returns a 2D or 3D point/vertex list, mirrored at the y-Axis

VLswapYX — y-axis becomes x-axis of a VL (z-rotation by -90 degree)

VLswapYZ — y-axis becomes z-axis of a VL (x-rotation by +90 degree)

VLswapZ — returns a 2D or 3D point/vertex list, mirrored at the z-Axis

VLswapZX — z-axis becomes x-axis of a VL (y-rotation by -90 degree)

VLswapZY — z-axis becomes y-axis of a VL (x-rotation by +90 degree)

VLtrans — return similar to PLtrans or SGtrans a transformed VL

VLtrans0 — moves an object into the origin of the coordinate system

VLtrans1 — moves an object into the first quadrant

VLtransN — returns a relative to normal vectors transformed vertex list

VLtransP — Moves all points of a vertex list using a translation vector p

VLtransR — transforms all points of a vertex list using a rotation matrix

VLtransT — returns a transformed Vertex list

wofT — returns simply the squeezed atan2 of the rotation matrix of T

Detailed description

barycartNT

returns the vertex list (3D) of a point list (2D) relative to one facet

In: **dt**: DelaunayTri object **Fi**: Facet index to relate to **XPL**: Vertex list, the 2D barycentric coordianates are requested

Out: **[XVL]**: Point list (2D) of barycentric coordinates

barytocartUVL

converts a 2d Barycentric [u,v] vertex list into a 3D Cartesian coordinate list [x, y, z]

In: **p1**: Point 1 of the triangle **p2**: Point 2 of the triangle **p3**: Point 3 of the triangle **UVL**: Barycentric vertex list, all points in plane

Out: **VL**: n x 3 list with [x y z]

barytocartUVLdelaunay

converts a 2d Barycentric [u,v] vertex list into a 3D Cartesian coordinate list [x, y, z]

In: **p1**: Point 1 of the triangle **p2**: Point 2 of the triangle **p3**: Point 3 of the triangle **UVL**: Barycentric vertex list, all points in plane

Out: **VL**: n x 3 list with [x y z]

cartbaryNT

returns the 2D barycentric coordinates of a VL relative to one facet

In: **dt**: DelaunayTri object **Fi**: Facet index to relate to **VL**: Vertex list, the 2D barycentric coordianates are requested

Out: **PL**: Point list (2D) of barycentric coordinates **inside**: inside of facet **onedge**: on edge of facet (not in corner) **bottop**: below(+1) or top (1) or on plane (0) **onplane**: onplane

carttobaryVL

converts a 3D planar vertex list into a 2D barycentric coordinate list [u,v]

In: **p1**: Point 1 of the triangle **p2**: Point 2 of the triangle **p3**: Point 3 of the triangle **VL**: Vertex list, all points in plane

Out: **UVL**: n x 2 list with [u v]

CPLtransform

finds contours inbetween two sets of CPLs => CVLof2CPLzcorrelate

In: **CPLA**: Contour A **CPLB**: Contour B **corall**: if true all contours are correlated not only pairs **zmm**: desired percentage of transformation 0=CPLA 1= CPLB

Out: **CPLt**: Final set of contours wrt to CCL **CLL**: CCL = Result of CPLcorrelate **MCA**: Missing contours in A - No possibility to use **MCB**: Missing contours in B - No possibility to use

CPLtransrelCPL

relative transformation of CPLs

In: **CPLA**: Original contour to manipulate **CPLB**: Contour that is considered as relation **de**: relation term such as 'ontop' or 'alignleft' **do**: optional numerical parameter

Out: **CPL**: Resulting CPL

CPLtransrelT

transform the pose of a CPL from one coordinate system into another

In: **CPL**: Contour for instance a crank **TA1**: 2D Frame or Frame list for example the path of the crank point A1 **TA0**: 2D Frame or Frame list for example the path of the crank origin point A0

Out: **CPLN**: Contour to be move using TA0 **T**: Transformation matrix to convert CPL to CPLN

diffTincz

returns for Frames the relative incliniation of the ez vector of B wrt to A

In: **TA**: Frame A **TB**: Frame b

Out: **w**: relative rotation of Bx relative to Ax

diffTrotz

returns for Frames the relative rotation of the ez vector of B wrt to A

In: **TA**: Frame A **TB**: Frame b

Out: **w**: relative rotation of Bx relative to Ax

dist2line

mirrors a point at a straight line in 2D

In: **P1**: Point on line **ev**: direction vector **p**: point to mirror

Out: **mp**: mirror point **do**: distance to straight line (wrt to ev) **cp**: crossing point on line **ov**: orthogonal vector to straight line from p to cp

ezof3P

returns ez and R for 3 Point Rotation Matrix

In: **VL**: Vertex List

Out: **ez**: ez vector [x y z] **R**: Rotation Matrix [ex ey ez] **T**: Transformation matrix [ex ey ez p1]

mirroringatline

mirrors a point at a straight line in 2D

In: **P1**: Point on line **ev**: direction vector **p**: point to mirror

Out: **mp**: mirror point **do**: distance to straight line (wrt to ev) **cp**: crossing point on line **ov**: orthogonal vector to straight line from p to cp

mirroringatline3D

mirrors a point at a straight line in 3D

In: **P1**: Point on line **ev**: direction vector **p**: point to mirror

Out: **mp**: mirror point **do**: distance to straight line (wrt to ev) **cp**: crossing point on line **ov**: orthogonal vector to straight line from p to cp **dz**: distance of cp from P1

planedistanceofT

returns the distance of a vertex list to a plane in HT coordinate form

In: **T**: Transformationmatrix 4x4 **VL**: Vertex list

Out: **D**: Distance vector

planeparamofT

returns the plane parameters from a HT matrix (x/y plane) - ax+by+cz+d=0

In: **T**: T **d**: option

Out: **pp**: plane parameter [a b c d]; ax+by+cz+d=0 **ez**: normal vector

plotT

plots a coordinate system given as HT matrix

In: **T**: HT matrix of the coordinate system **s**: size of the unit vectors, default is 1 **w**: width of the matlab lines, default is 1 **N**: Name of coordinate system

Out: **h**: Handle to the plotted graphics object

PLtrans

multi transformation modes for a point list

In: **PLin**: Point list nx2 **T**: Transformation Value, Vector, or Matrix

Out: **PLout**: Point list nx2 **varargout**: not used yet

PLtrans0

returns an origin centered point list

In: **PL**: Point list nx2

Out: **PL**: Point list nx2

PLtrans1

returns a point list moved into the first quadrant

In: **PL**: Point list nx2

Out: **PL**: Point list nx2

PLtransC

rotates a point list around a center

In: **PL**: Point List [nx2] **C**: Center of rotation default is zero **w**: angle of rotation default is pi/2

Out: **PL**: Resulting Point list

PLtransP

returns a translated point list

In: **PL**: Point list nx2 **p**: translation vector 1x2 **a**: optional axis for TofP; if given, translation uses a homogeneous transform TofP(p,a) via PLtransT; default empty

Out: **PL**: Translated point list

PLtransR

rotates all point of a point list

In: **PL**: Point List (nx2) **R**: Rotation matrix 2x2

Out: **PL**: Transformed Point list

PLtransT

rotates and moves a point list nx2

In: **PL**: Point list nx2 **T**: Transformation matrix 2x3 or 3x3

Out: **PL**: Point list nx2

PrelCPL

returns a point position given by a spatial description relative to a CPL

In: **CPLB**: Contour used for a relative spatial description **rel**: spatial relation **gap**: optional gap like in CPLtransrelCPL

Out: **PN**: Position of the point

RL2exl

returns X vectors for a list of euler angles

In: **RL**: Rotation List of Euler Angles

Out: **EXL**: List of ex vectors

RLofEulerInterpolation

Returns euler rotations as linear interpolation

In: **VL**: Vertex list (nx3) or scalar n **e1**: ZYX-Euler angle for first Vertex VL(1,:) or matrix T1,R1 **e2**: ZYX-Euler angle for last Vertex VL(end,:) or matrix T2,R2

Out: **RL**: Rotation list of euler angles (n x 3) **L**: Length of path, scalar **DL**: Distance vector; last one closes (n x 3) **NL**: Norm of distance vectors (n x 1) **EXL**: Optional List of

RLplot

plots along the z axis the systems of a Euler Rotation List

In: **RL**: Euler Angle Rotation List **VL**: Vertex list or distance between points

Rofez

returns a rotation matrix based on a list of ez vector

In: **ezz**: ez vector or list of ez vectors **ystab**: true== fix y-axis; false== fix x-axis; default is true;

Out: **R**: Rotation matrix or vector of rotation matrices

Rofrdintrusion

provides the minimum intrusion distance to fully insert a cylinder into a sphere

In: **d**: distance between chord and Radius R **r**: radius of the chord

Out: **R**: Radius of a circle with chordlength 2*r and d between chord and R

Rofxy

Rotation matrix by the ez vektor and and optional rotation angle

In: **x**: x value of the ez vector **y**: y value of the ez vector **wz**: rotation angle, default is 0

Out: **R**: Rotation matrix 3x3

rot

returns rotation matrix in 3x3 as Rx*Ry*Rz or 2x2 Rz

In: **wx**: x rotation angle (optional) **wy**: y rotation angle (optional) **wz**: z rotation angle

Out: **R**: Rotation matrix 3x3

rotdeg

returns 3x3 rotation matrix for Deg angles in x, y, z

In: **wx**: angle for x-rotation in Deg

Out: **R**: Rotation matrix 3x3

RPofTrelT

returns the frame T2 in relation to frame T1

In: **TA**: Frame A **TB**: Frame B

Out: **R**: Rotation angles (euler) XYZ **P**: Distance **A_T_B**: Frame B relative to Frame A

screw2R

definition of a screw coordinate system

In: **scr**: [x y w]

Out: **R**: Rotation matrix

SGtransrel

returns a spatial transformed solid using SGtransrelSG

In: **SG**: Solid Geometry

Out: **SGN**: Spatially transformed solid

SGTtrans0

move the center of the solid into the origin of the coordinate system

In: **SG**: Solid Geometry **z0**: if used, the minimal z value is this parameter z0

Out: **SGN**: Resulting Geometry with all frames modified

TflipR

turns the current right hand system into an inverted left hand system

In: **T**: Right hand frame

Out: **T**: Left hand frame

tfplot

plots an HT matrix including the x/y plane

In: **T**: HT Matrix **f**: size of plane; default is 1 **a**: Transparency alpha, default is 0.5 **c1**: Color; default is 'r' **c2**: Color; default is 'r' **c3**: Color; default is 'g'

Out: **h**: Handle to the plotted graphics object

THLof2F

returns Tetrahedron list of two facets

In: **F1**: 1x3 Facet **F2**: 1x3 Facet

Out: **THL**: Single tetrahedron

TLofCVL

returns a list of frames that connect a start frame and an end frame along a spatial contour

In: **XVL**: Open Path to follow in 3D **R**: Radius; default is 2 **T1**: Start Frame **T2**: End Frame **typ**: type 'rad', 'rmax', 'tan'; default is 'rad'; see VLradialEdges2T **tor**: true=>torsion; false=>no torsion; default is true **aux**: distance of axiliary points; default is R/3

Out: **Ti**: List of HT-Matrixes (4 x 4 x n] for n vertices

Tof2vec

returns transformation matrix based on the 2 point movements

In: **p1**: 1st point before movement **p2**: 1st point after movement **q1**: 2nd point before movement **q2**: 2nd point before movement

Out: **T12**: 2_T_1 Transformation from T1 into T2 **T21**: 1_T_2 Transformation from T2 into T1

TofcamVLPL

returns the coordinate system and calibration matrix of a camera

In: **VL**: Vertex List in world coordinates **PL**: Point List in the image coordinate system **fs**: focus scaling factor applied to the image point list; default is 1

Out: **T**: Transformation matrix of the camera position **K**: Intrinsic camera parameter **s**: scaling factor **azxz**: rotation angle zxz

TofDPhiH

returns a 4x4 transformation matrix

In: **D**: Distance in X **Phi**: Rotation around Z **H**: Height in Z

Out: **T**: Transformation matrix 4x4

Tofgca

returns the HT matrix of the current camera position

Out: **T**: HT matrix of camera coordinate system **Tt**: HT of target coordinate system **va**: current camera angle **ax**: current camera axis

TofP

returns a T matrix from a point in 2D or 3D

In: **p**: point [x y] or [x y z] **R**: optional rotation matrix; default is empty (no rotation)

Out: **T**: Transformation matrix 3x3 or 4x4

TofPcam

returns a transformation matrix using a vertex and a the current camera view

In: **p**: 3D coordinate in the current view OR handle of a graphics object; default is gca **r**: 'a', 'c', 's' 'm' as axial coral or sagittal point; default is 'm'ouse **cp**: if true; cross section center; if false or numeric; surface plus value

Out: **T**: HT matrix with ez = view direction and ey = cup of view **vi**: index of nearest vertex **fi**: index of touched facet **aew**: [view(1) view(2)] to restore camera view later **hg**: handle of the graphics object touched by select3d **r**: orientation char actually used ('a','c','s') after mouse selection

TofPEul

return from euler angle and position an HT matrix

In: **p**: Point in 3D **eul**: Euler angle [phiz phiy phix] - not x y z **seq**: default is 'ZYX'

Out: **T**: Homogenous Transformation Matrix

TofPez

creates a T matrix from point and ez-Vector

In: **p**: point; if empty [0 0 0] or [0 0] is used **ez**: ez-vector length is not relevant **a**: optional rotation angle

Out: **T**: Transformation matrix; ex is set by default

TofSG

sets a HT matrix relativ to nested solid

In: **SG**: Solid Geometry **rel**: relations or ez-vector **"5.1"**: flag string '5.1' to keep SG-Lib 5.1 behavior (ez always up) via TofSG2021

Out: **T**: Frame

TofSGML

returns HT for a mounting face with known number

In: **SG**: Solid Geometry with defined mounting faces ML, MA **mi**: mounting face number **fe**: feature edge angle; default is 0.5; used only if ML is not defined

Out: **T**: Transformation matrix [4x4] **SG**: Solid geometry with defined mounting faces ML, MA

TofSGMLez

returns HT for a mounting face with known ez

In: **SG**: Solid Geometry with defined mounting faces ML, MA **ez**: desired ez vector **ai**: order is size; default is 1 **fe**: feature edge angle; default is 0.5; used only if ML is not defined

Out: **T**: Transformation matrix [4x4] **TFiL**: facet index list of the selected mounting face **ss**: size (dimensions) of the selected mounting face from sofBB

TofT

returns a T matrix defined relative to another T matrix

In: **T0**: Original Matrix **R**: Rotation matrix relative to the T axis; scalar=phiz **P**: Translation vector relative to the T axis; scalar=z

Out: **T1**: resulting new T Matrix

TofTrel

creates a HT matrix relative to a frame

In: **T**: Original frame **rel**: Relative operation such as 'transz','roty','rotz' ... **gap**: Value / distance for the relative operation

Out: **T**: New Frame

TofVL

returns the main axis of a vertex list or SG

In: **VL**: Vertex list or Solid Geometry **ez**: ez vector for turning; default is empty **force**: if true ez is forced as ez vector; default is false

Out: **T**: HT Matrix of the main axis **SGC**: convex hull solid geometry of the vertex list; currently returned empty **D**: largest eigenvalue of the covariance matrix of the vertices

TPL

return for vertices of a patch the HT Matrix of the patch center

In: **PL**: Point list or vertex list

Out: **T**: Transformation for Backtransformation **X**: Vertex list of transformed vertices **n**: length of vertex list

tplot

plots a coordinate system given as HT matrix

In: **T**: HT matrix of the coordinate system **s**: size of the unit vectors, default is 1 **w**: width of the matlab lines, default is 1 **N**: Name of coordinate system **ts**: if empty, no surfaces for tips and origin

Out: **h**: Handle to the plotted graphics object

Ttransrel

language based fnct for spatial arrangement of frames

In: **T**: Frame to translate/rotate **comsequ**: commandsequence such as 'rotz',pi/3,'tranz',5 **"debug"**: if used; the indiviual sequences are plotted

Out: **T**: resulting Frame **k**: number of transformations made

TtransT

returns a matrix multiplication of a (list of) homogenous transformation matrix

In: **T1**: Transformation matrix to transform or matrix list **T0**: Original transformation matrix or matrix list **tinv**: if true; T0 will be interverted first

Out: **TN**: T1*T0 (!!)

Txy

returns a matrix for a point p and pitch-yaw values of ez

In: **p**: Origin of coordinate system **x**: x value of z-axis **y**: y value of z-axis

Out: **T**: is the resulting homogenous transformation matrix

VLinsertEulerSteps

inserts vertex points on straight lines if the twist angle is to large

In: **VL**: Vertex list (nx3) **e1**: ZYX-Euler angle for first Vertex VL(1,:) **e2**: ZYX-Euler angle for last Vertex VL(end,:) **gw**: limiting angle; default is pi/16

Out: **NVL**: New Vertex list **NRL**: new ZYX-Euler angle rotation list for the inserted vertices

VLplaneprojection

returns a PL as projection of a vertex list to a orthogal plane defined by an ez vector

In: **VL**: Vertex list **nv**: normal ez vector;

Out: **PL**: Point list **nvv**: adjusted ez vector ([0 0 1], [1 0 0]. or [0 1 0]

VLprojection2SG

projects a Vertex list in parallel projection to a solid

In: **SG**: Solid to project to **VL**: Vertex list **ez**: normal vector

Out: **VLN**: Corresponding projection points; nan if there is no projection point

VLswapX

returns a 2D or 3D point/vertex list, mirrored at the x-Axis

In: **VL**: Vertex list

Out: **RVL**: Resulting vertex list

VLswapXY

x-axis becomes y-axis of a VL (z-rotation by 90 degree)

In: **VL**: Original vertex list **r**: number of successive 90 degree z-rotations to apply; default is 1

Out: **RVL**: Rotated vertex list.

VLswapXZ

x-axis becomes z-axis of a VL (y-rotation by +90 degree)

In: **VL**: Original vertex list

Out: **RVL**: Rotated vertex list.

VLswapY

returns a 2D or 3D point/vertex list, mirrored at the y-Axis

In: **VL**: Vertex list

Out: **RVL**: Resulting vertex list

VLswapYX

y-axis becomes x-axis of a VL (z-rotation by -90 degree)

In: **VL**: Original vertex list

Out: **RVL**: Rotated vertex list.

VLswapYZ

y-axis becomes z-axis of a VL (x-rotation by +90 degree)

In: **VL**: Original vertex list

Out: **RVL**: Rotated vertex list.

VLswapZ

returns a 2D or 3D point/vertex list, mirrored at the z-Axis

In: **VL**: Vertex list

Out: **RVL**: Resulting vertex list

VLswapZX

z-axis becomes x-axis of a VL (y-rotation by -90 degree)

In: **VL**: Original vertex list

Out: **RVL**: Rotated vertex list.

VLswapZY

z-axis becomes y-axis of a VL (x-rotation by +90 degree)

In: **VL**: Original vertex list **n**: optional nr of repeated cycles, default is 1

Out: **RVL**: Rotated vertex list.

VLtrans

return similar to PLtrans or SGtrans a transformed VL

In: **VLA**: Vertex list **T**: Transformation vector

Out: **VL**: transformed vertex list

VLtrans0

moves an object into the origin of the coordinate system

In: **VL**: Original vertex list **z0**: optional flag; if true the object is shifted so its minimum z is 0; default is false

Out: **VLN**: Vertex list **sx**: Size in x (also maximum value in x) **sy**: Size in y (also maximum value in y) **sz**: Size in z (also maximum value in z)

VLtrans1

moves an object into the first quadrant

In: **VL**: Original vertex list

Out: **VLN**: Vertex list; no value is smaller than 0 **sx**: Size in x (also maximum value in x) **sy**: Size in y (also maximum value in y) **sz**: Size in z (also maximum value in z)

VLtransN

returns a relative to normal vectors transformed vertex list

In: **VLO**: Vertex list of a planar surface **FLO**: Facet list of a planar surface **s**: shrink factor **d**: distance factor **T**: optional homogeneous transformation matrix applied to the result; default eye(4)

Out: **VL**: Vertex list [nx3] **EL**: Edge list [nx2] **EOL**: Edge orthogonal list [nx3] **FL**: Facet list [nx3] **VNL**: Vertex normal list [nx3]

VLtransP

Moves all points of a vertex list using a translation vector p

In: **VL**: Original Vertex list **P**: translation vector

Out: **VL**: Transformed vertex list

VLtransR

transforms all points of a vertex list using a rotation matrix

In: **VL**: Original vertex list **R**: 3x3 Rotation matrix

Out: **VL**: Transformed vertex list

VLtransT

returns a transformed Vertex list

In: **VL1**: Vertex list n x 3 **T**: Transformation matrix 4 x 4

Out: **VL**: Transformed vertex list

wofT

returns simply the squeezed atan2 of the rotation matrix of T

In: **T2**: HT Matrix in 2D = 3x3xM

Out: **w**: angle list

2.6 Curves & Splines [guide]

Functions in this section:

circ2poly — ratio of polygon radius between if turned pi/n radiands

PLevolvente — returns the point list of an evolvente curve

RLcurvature — returns a circular bended closed contour

VLBezier4P — simple fnct to calculate a 3D Bezier curve

VLBezierC — returns a Bezier curve for a given list of points

VLhelix — generates a vertex list for a specified helix

VLRadius4P — returns a contour based on 4 points similar to VLBezier4P

zigzag — returns a zig zag curve in the interval [0 2*pi]

Detailed description

circ2poly

ratio of polygon radius between if turned pi/n radiands

In: **n**: number of polygon edges/corners **r**: optional radius

Out: **m**: magnification >1

PLevolvente

returns the point list of an evolvente curve

In: **r**: radius **w**: maximal angle; default is 2*pi **n**: length of PL; default is 12; **w0**: length of PL; default is 12;

Out: **PL**: Point list (2D) of the evolvente/involute function nox [n x 2]

RLcurvature

returns a circular bended closed contour

In: **RL**: Radius List (r z) **C**: [R z] of circle center **n0**: 0=neutral fiber at 0; else Rmin

Out: **CL**: circular bended radius list **alpha**: maximum bending angle

VLBezier4P

simple fnct to calculate a 3D Bezier curve

In: **p1**: 1st point **p2**: 2nd point **p3**: 3rd point **p4**: 4th point **n**: number of supporting points

Out: **VL**: Vertex list of the bezier spline

VLBezierC

returns a Bezier curve for a given list of points

In: **VL**: Vertex list **k**: number of points of the Bezier curve

Out: **BVL**: Vertex list of the Bezier curve

VLhelix

generates a vertex list for a specified helix

In: **R**: Radius (centered) of the helix **d**: Height (centered) of the helix **w2**: Optional turn angle, default is 2pi **w1**: Optional starting angle, default is 0 **n**: Optional maximum geometry error, default is 0.05 **ccw**: true=> ccw:math pos; default is true

Out: **VL**: Vertex list [nx3] of the helix in z-direction **n**: size(VL,1)

VLRadius4P

returns a contour based on 4 points similar to VLBezier4P

In: **p0**: 1st point **p1**: 2nd point **p2**: 3rd point **p3**: 4th point **wmax**: Maximal direction change angle **R**: Minimal Radius of the curve **nmax**: maximum length of a radius field; default is 40

Out: **VL**: Vertex list of the specified contour **NVL**: Vertex list with more supporting lines

zigzag

returns a zig zag curve in the interval [0 2*pi]

In: **x**: x value **per**: periode; default is 2*pi

Out: **y**: zigzag value

2.7 Normals & Vectors [guide]

Functions in this section:

acos2 — 2D fnct: returns the angle between two vectors including sign

distE — returns shortest distance to x/y plane and normal vector to p

distT — returns orthogonal vector and normal vector to T-plane

edgeNormal — returns the edge normals of surface boundaries

hesseplaneofT — Hessennormalform of T's xy plane

NLconformance — changes the orientation of a list of normal vectors

NLcontourVL — return the list of normal vector for the edges between the vertices and the vertices between the edges

PLFLfaceNormal — returns the rounded (1e-5) length of the area size/length of normal vector

PLnorm — returns a normalize vector list nx2

Tofhesseplane — returns an HT matrix from a hesse normal form of a plane

TplanesofPCnormals — returns the HT matrices for the points and normals

TplanesofVLFLfacets — returns the HT matrices for the facets from center and normal

vec2ortho — creates the orthogonal vectors for a 2D vector list

vecnormr — returns for a vector list (rows) the vectnorm result

vertexNormalbyFN — calculates the vertex normals based on the facet normals

VLNLplot — plots a normal vector list at a vertex list

Detailed description

acos2

2D fnct: returns the angle between two vectors including sign

In: **NL1**: Vector list 1 **NL2**: Vector list 1 **normfirst**: default is true;

Out: **w**: result from -pi .. + pi **a**: area between both vectors

distE

returns shortest distance to x/y plane and normal vector to p

In: **p**: 1x3 or 3x1 vector

Out: **o**: orthogonal vector: 1x3 **n**: normal vector: 1x3

distT

returns orthogonal vector and normal vector to T-plane

In: **T**: 4x4 Homogenous Transformation Matrix **p**: 1x3 or 3x1 vector

Out: **o**: orthogonal vector: 1x3 **n**: normal vector: 1x3

edgeNormal

returns the edge normals of surface boundaries

In: **VL**: Vertex list (nx3) **VNL**: Vertex normal list (nx3) (i.e. vertexNormal) **EL**: Edge list (i.e. freeBoundary)

Out: **VOL**: Vertex orthogonal vector of the starting vertex in the edge list **c**: starting vertex index list = EL(:,1)

hesseplaneofT

Hessennormalform of T's xy plane

In: **T**: HT Matrix **T0**: reference plane default is eye(4)

Out: **dh**: hesse distance to plane **ez**: ez vector of T **XL**: PL that defined the cutting edge

NLconformance

changes the orientation of a list of normal vectors

In: **NVL**: Normal vector list **ez**: reference direction vector.

Out: **NVL**: Correct normal vector list

NLcontourVL

return the list of normal vector for the edges between the vertices and the vertices between the edges

In: **VL**: Vertex list of a sorted(!) contour: Use VLGraham before or VLsortC

Out: **ENL**: List of normal vector of the edges **VNL**: List of normal vector of the edges

PLFLfaceNormal

returns the rounded (1e-5) length of the area size/length of normal vector

In: **PL**: Point list n x 2 **FL**: Facet list **fi**: optional value; facet index (list)

Out: **NL**: n x [0 0 1] **AL**: Area list;

PLnorm

returns a normalize vector list nx2

In: **PL**: Vector list

Out: **NL**: Normalize vector list **DL**: Length list

Tofhesseplane

returns an HT matrix from a hesse normal form of a plane

In: **d**: distance to plane **ez**: ez vector of plane **T0**: optional relative plane that is used for d and ez; default is eye(4)

Out: **T**: HT matrix of plane

TplanesofPCnormals

returns the HT matrices for the points and normals

In: **pc**: point cloud or vertex list

Out: **T**: cell list of transformation matrices **DVL**: Triangle vertex list **FL**: facet list

TplanesofVLFLfacets

returns the HT matrices for the facets from center and normal

In: **VL**: Vertex list or Solid Geometry **FL**: Facet list

Out: **T**: cell list of transformation matrices **DVL**: Triangle vertex list **DFL**: fTriangle vertex listacet list

vec2ortho

creates the orthogonal vectors for a 2D vector list

In: **v**: 2D vector list [n x 2]

Out: **o**: orthogonal vector

vecnormr

returns for a vector list (rows) the vectnorm result

In: **VL**: vector [n x 2] or [n x 3] as rows

Out: **LL**: vector norm of the rows

vertexNormalbyFN

calculates the vertex normals based on the facet normals

In: **TR3**: triangulation

Out: **NNL**: Normal vector list **NL**: vector sum of the face norms attached to the vertex

VLNLplot

plots a normal vector list at a vertex list

In: **VL**: Vertex list **NL**: Normal vector list **vi**: optional index selector **c**: optional color and line **w**: optional line thickness

Out: **h**: handle to plot

2.8 Distances & Metrics [guide]

Functions in this section:

aplot — draws an angle relative to a center and two vectors

BBofVL — returns the limits of the bound box

center3P — Center line of a cylinder given by 3 points

center4P — Center and diameter of a sphere given by 4 points

centerVL — returns the center and distance indices of a vertex list

collofBBs — Fundamental fnct to return from two bounding box list the colliding index lists

CVLremstraight — returns straight points on a line depending on distance and agle

diffangle — 3D fnct: calculates the angle between two normal vector lists

dofRintrusion — provides the minimum intrusion distance to fully insert a cylinder into a sphere

ellipseofPL — returns the parameter for an ellipse of at least 5 points

KDWofXY — returns two line distances of points and an angle and a displacement of a straigt line line from two points

lengthC — returns length and maximum of changing angle plus

nearestcluster — returns a reduced list of points and a cluster list

nearestpair — returns the nearest point pairs of two vector list

nofrd — returns the minimum number of edges to guarante a slot distance between a peg and a hole

outsideplaneVL — returns whether a vertex/vertex list is below or above a 3P plane

outsidetriangleVL — returns whether a vertex/vertex list is below or above or outside a 3P triangle

PLangle — returns the angles of a points list

PLangle2 — returns the angles of a points list

PLconvexseg — returns the segments of convex and concave contours within ONE CPL

PLminangle — returns the point with minimal angle value wrt to the center of the contour

rofRintrusion — provides the minimum intrusion distance to fully insert a cylinder into a sphere

SGcutTbladeangle — returns the mean center angle of the cutting blades

SGgetCenterPoint — returns a point in the center of all vertices

SGmagnifyVL — magnifies/shrinks the dimensions of the vertex list

SGmirrorD — Mirrors a solid in a distance to the maximal dimension in the specified direction

SGsurfacedistance — calculates the shift distance from a solid to the surface of another solid

SGvolume — returns the estimated volume of a solid

sofBB — returns the maximal dimension of the bounding box

VLangle — returns angle and distance between contour list in 3D

VLdistance — returns a distance list - but is different from VLnorm

VLmindist — returns the minimal distance larger than zero

VLmindxyz — returns the minimal distance larger than zero

VLmostdistanceNeighborN — return a list of n points ou of a points list that have the maximum distance sum

VLnearestNeighborN — returns the N nearest Neighbors between two point lists

VLtransSphere — projects the flat surface onto the xy plane onto the curved surface of the ellipsoid, and the vertical planes become curved surfaces that all meet at the center of the ellipsoid

Detailed description

aplot

draws an angle relative to a center and two vectors

In: **pc**: center of angle **la**: 1st vector **lb**: 2nd vector or angle or rotation matrix **c**: color; default is 'r' **w**: width of lines; default is 1; **r**: radius; default is **alpha**: transparency **"360"**: if used, angle over 180 degree are drawn correctly using TLofProt

Out: **h**: handle to all elements of the angle drawing **pt**: point for text desciption **a**: angle that was used if lb was a vector

BBofVL

returns the limits of the bound box

In: **VL**: Vertex list, Point List, Closed Polygon List **zmax**: z value assigned to the last vertex when the input is a 2D point or polygon list; default 0 **d**: distance added around the bounding box in every direction; default 0

Out: **bb**: [xmin xmax ymin ymax zmin zmax]

center3P

Center line of a cylinder given by 3 points

In: **p1**: Point 1 of the circle or cylinder, or Vertex List **p2**: Point 2 of the circle or cylinder **p3**: Point 3 of the circle or cylinder

Out: **T**: Matrix of a cylinder center line **r**: Radius of a cylinder **cp**: center point = T(1:3,4)

center4P

Center and diameter of a sphere given by 4 points

In: **p1**: Point 1 of the sphere **p2**: Point 2 of the sphere **p3**: Point 3 of the sphere **p4**: Point 4 of the sphere

Out: **px**: position of the sphere's origin **rx**: radius of the sphere

centerVL

returns the center and distance indices of a vertex list

In: **VL**: Vertex list (nx3)

Out: **c**: center **ci**: index list of nearest vertices **d**: distance vector to center (nx3) **n**: distance norm to center (nx1)

collofBBs

Fundamental fnct to return from two bounding box list the colliding index lists

In: **bba**: bounding box list A **bbb**: bounding box list B **first**: if true; function stops after first hit; default is false **unidir**: if true and bba==bbb the test saves half the time

Out: **CL**: Crossing list [n x 2] = [ia ib] **ca**: optional list of crossing boxes in A; ca=unique(CL(:,1),'stable'); **cb**: optional list of crossing boxes in B; cb=unique(CL(:,2),'stable');

CVLremstraight

returns straight points on a line depending on distance and agle

In: **CVL**: Contour verteex list **de**: default is 1e-6; **al**: default is 1 degree; will be changed in future to eps2

Out: **NCVL**: contour vertex list with straight-line and near points removed

diffangle

3D fnct: calculates the angle between two normal vector lists

In: **NL1**: normal vector lists 1 **NL2**: normal vector lists 2 **normfirst**: if true; VLnorm is sued to norm the vectors first;

Out: **AL**: real(acos(dot(NL1(i,:),NL2(i,1)))) **DA**: DA=abs(abs(pi/2-AL)-pi/2)

dofRintrusion

provides the minimum intrusion distance to fully insert a cylinder into a sphere

In: **R**: Radius of sphere **r**: radius of cylinder

Out: **d**: distance to insert

ellipseofPL

returns the parameter for an ellipse of at least 5 points

In: **PL**: Point List

Out: **cp**: center **a**: long axis **b**: short axis **ex**: ex vector **ey**: ey vector **phi**: turning angle

KDWofXY

returns two line distances of points and an angle and a displacement of a straigt line line from two points

In: **A1**: Point of attachment point A1 wrt to origin **B1**: Point of attachment point B1 wrt to origin **TP**: Optional Pose Coordinate System wrt to Origin; default is origin; eye(4);

Out: **K**: distance of A1 and B1 along the origin of a straight line **D**: displacement of the line wrt to the origin **W**: turning angle of the line relative to x-axis

lengthC

returns length and maximum of changing angle plus

In: **VL**: Vertex list of the contour

Out: **l**: length of the contour **wmax**: maximum segment angle of the contour **dwmax**: maximum bending force factor of the contour

nearestcluster

returns a reduced list of points and a cluster list

In: **RL**: Point list **lim**: maximum distance for clustering

Out: **RLn**: Points list with replaced mean values **Rn**: Cluster matrix related to RL

nearestpair

returns the nearest point pairs of two vector list

In: **PLA**: Vector list A **PLB**: Vector list B **tp**: 'min', 'mean' ; default is 'min'

Out: **p**: [Ai Bi] index list for the exclusive pairs **x**: Distance between the pairs **D**: Full Distance matrix **pn**: [Ai Bi] index list for the unpaired points

nofrd

returns the minimum number of edges to guarante a slot distance between a peg and a hole

In: **r**: radius of a circle, cylinder etc. **d**: slot size or maximum roughness; default is 0.025 **w**: optional angle; default is 2*pi

Out: **n**: minimum number of edges for a given radius; returns d for r<0 **dw**: angular offset step 2*pi/n/2 divided by a random factor

outsideplaneVL

returns whether a vertex/vertex list is below or above a 3P plane

In: **VL**: Vertex list **p1**: Point 1 of the plane **p2**: Point 2 of the plane **p3**: Point 3 of the plane

Out: **OL**: Outside List

outsidetriangleVL

returns whether a vertex/vertex list is below or above or outside a 3P triangle

In: **VL**: Vertex list **p1**: Point 1 of the plane **p2**: Point 2 of the plane **p3**: Point 3 of the plane

Out: **OL**: Outside List

PLangle

returns the angles of a points list

In: **PL**: vector list

Out: **w**: angle list **d**: distance list

PLangle2

returns the angles of a points list

In: **PL**: vector list

Out: **w**: angle list **d**: distance list **nw**: increment of angle list **wi**: number of full rotations

PLconvexseg

returns the segments of convex and concave contours within ONE CPL

In: **PL**: Single Point List

Out: **con**: array of convex and concave **AL**: Angle List **CIL**: Change index list plus status [n x 3]

PLminangle

returns the point with minimal angle value wrt to the center of the contour

In: **PL**: Point Liste **in0**: false=[-pi..+pi]; true=[0..2*pi]; default is false **C**: optional center point **at**: angle threshold; default is pi/100

Out: **P**: Point with minimal y and minimal x **pin**: find index in PL **AL**: Angle list **DL**: Distance from Center List **C**: Center Point

rofRintrusion

provides the minimum intrusion distance to fully insert a cylinder into a sphere

In: **R**: Radius of sphere **w**: wall thickness of sphere

Out: **r**: maximum radius to insert

SGcutTbladeangle

returns the mean center angle of the cutting blades

In: **SG**: Solid Geometry **T**: Cutting Frame **ct**: Cutting orientation; default is 'a' **R**: Radius to be used

Out: **w**: mean angle of cylindric blade (Starting angle would be w-pi/2) **R**: radius used for the cutting blade contour

SGgetCenterPoint

returns a point in the center of all vertices

In: **A**: Solid Geometry

Out: **p**: point

SGmagnifyVL

magnifies/shrinks the dimensions of the vertex list

In: **oSG**: Original Solid Geometry **m**: magnification value or [mx my mz] **lim**: optional boundaries for magnification **prop**: proportional scaling; default is true

Out: **SG**: resulting Solid Geometry **m**: used scaling factor [mx my mz]

SGmirrorD

Mirrors a solid in a distance to the maximal dimension in the specified direction

In: **SG**: Solid Geometry **T**: 'x','y','z' or T matrix **D**: distance in direction **centr**: true if result is centered; default is true

Out: **SGN**: mirrored Solid **C**: Both solids combined (and centered)

SGsurfacedistance

calculates the shift distance from a solid to the surface of another solid

In: **A**: Solid to move **B**: Solid to be touched **ez**: orientation i.e. shift vector

Out: **d**: distance **NA**: Shifted Solid A

SGvolume

returns the estimated volume of a solid

In: **SG**: Solid Geometry **sz**: step size in z; default is 1

Out: **VSUM**: Volume in cm^3 **A**: Area of the individual slices

sofBB

returns the maximal dimension of the bounding box

In: **SG**: VL oder SG or BB **minv**: optional minimal value; default is 0 **maxc**: if true and SG is a cell, use the largest-volume solid; default is false **dims**: dimensions to include as logical [x y z] or string like 'xy'; default is [true true true]

Out: **ms**: max dimension **s**: [sx,sy,sz] **cp**: center point of BB **bb**: BB of SG

VLangle

returns angle and distance between contour list in 3D

In: **VL**: vector list

Out: **w**: angle list **d**: distance to next point (circshift) **ov**: orthogonal vector (blue) **oa**: orthogonal area **isp**: is planar

VLdistance

returns a distance list - but is different from VLnorm

In: **VL**: Vertex list

Out: **DL**: Distance list **DVL**: Distance vector list

VLmindist

returns the minimal distance larger than zero

In: **VL**: vector list **epsr**: optional rounding before; default is 0, i.e. no rounding

Out: **dmm**: absolute minimal distance **dmin**: minimal distance per column

VLmindxyz

returns the minimal distance larger than zero

In: **VL**: vector list **Name**: Optional Name string for warnings and verbose print **FL**: Optional Facet list to get information on Area and collapsed facets **grid**: Optional Grid size to estimate the problems by grid rounding

Out: **dmin**: absolute minimal distance **dmm**: abolute maximum dimension

VLmostdistanceNeighborN

return a list of n points ou of a points list that have the maximum distance sum

In: **VL**: Vertex list **dim**: number of points required

Out: **nn**: full list of combiations **ll**: full list of distances

VLnearestNeighborN

returns the N nearest Neighbors between two point lists

In: **VL1**: Point list 1 **VL2**: Point list 2 **N**: selection of neighbors; default is [2:3] **D**: "ascend' (default) or 'descend'

Out: **nn**: rows of neighbors for VL1 in List VL2 **nd**: distance to neighbors from VL1 to VL2

VLtransSphere

projects the flat surface onto the xy plane onto the curved surface of the ellipsoid, and the vertical planes become curved surfaces that all meet at the center of the ellipsoid

In: **VL**: Vertex list **Radii**: [Rx Ry Rz] of an ellipsoid **Tcp**: HT matrix with enter an main axis of the ellipsoid

Out: **VLN**: Vertex list projected onto the ellipsoid surface

2.9 Convex Hull & Registration [guide]

Functions in this section:

centerPL — returns the mean of the convex hull of the point list

isconvexPL — return S>0 if PL (n x 2) is convex.

TregisterConvexHull — returns the registration matrix based on the convex hull of the vertices

VLGraham — returns the convex hull of a point list by Graham (outdated)

VLGrahamPVL — returns the convex hull for a given set of vertices that are in a plane

Detailed description

centerPL

returns the mean of the convex hull of the point list

In: **PL**: Point list (n x 2)

Out: **pc**: point list center **Rm**: maximum distance from the center to the points (enclosing radius) **NL**: list of normalized vectors from the center to each point

isconvexPL

return S>0 if PL (n x 2) is convex.

In: **PL**: Planar contour point list (n x 2)

Out: **S**: z-Value of the cross product **i**: index of the first element that is not convex

TregisterConvexHull

returns the registration matrix based on the convex hull of the vertices

In: **VLA**: Vertex list A or Solid A **VLB**: Vertex list B or Solid B

Out: **T**: Transformation from B into A

VLGraham

returns the convex hull of a point list by Graham (outdated)

In: **VL**: Unsorted point list

Out: **CHVL**: Sorted point list of the convex hull **n**: length of the convex hull

VLGrahamPVL

returns the convex hull for a given set of vertices that are in a plane

In: **PCVL**: planar contour vertex list

Out: **[VL]**: vertex list of the convex hull

3 Polygons and Contours [guide]

3.1 Single 2D Polygons (PL) [guide]

Functions in this section:

PLcircleoval — returns a circle-based-oval point-list

PLcircseg — returns a PL for a circle segment

PLgearDIN — returns a gear contour wrt DIN from 3 to 48 teeth (replaces PLgear of year 2012)

PLkidney — returns a kindey shaped point list

PLrand — returns a point list with random points

PLspiral — return the PL of a spiral

PLsquare — returns a 4 point list describing a rectangle or trapeze in 2D

PLstar — returns a 2D point list of a star or star segment

Detailed description

PLcircleoval

returns a circle-based-oval point-list

In: **R**: Radius **n**: optional number of points for half circle **L**: Length; default is 100; or [A0 B0] **Ri**: optional inner Radius for links; default is 0

Out: **PL**: Point List

PLcircseg

returns a PL for a circle segment

In: **R**: Radius **n**: number of auxiliary points **phiA**: start angle **phiB**: end angle **Ry**: Radius in y, default is R

Out: **PL**: Point list n * 2 [x y]

PLgearDIN

returns a gear contour wrt DIN from 3 to 48 teeth (replaces PLgear of year 2012)

In: **m**: modul; negative means internal gear **z**: Teeth number **tig**: starting orientation (=gap, 1=tooth) **n**: number of nodes per flank; default 5 **shr**: slot between inside and outside contours taken into both directions

Out: **PLG**: Point list (2D) **d**: rolling circle **da**: head diameter **df**: foot diameter **db**: radius of 2nd contour

PLkidney

returns a kindey shaped point list

In: **RI**: Inner radius **RO**: Outer Radius **w**: angle of circular segment **n**: number of points per element;default is '' = auto

Out: **PL**: Point List

PLrand

returns a point list with random points

In: **n**: number of points; default is 10 **r**: Radius; default is 10 **c**: Center; default is [0 0]

Out: **PL**: Point list nx2 (unsorted) **CPL**: Point list nx2 (sorted as contour), not closed

PLspiral

return the PL of a spiral

In: **RI**: Inner Diameter **RO**: Outer Diameter **wm**: Desired angle; default is 2pi **n**: number of points; default is nofrd((RO+RI)/2)/2pi*wm

Out: **PL**: Point list [nx2]

PLsquare

returns a 4 point list describing a rectangle or trapeze in 2D

In: **X**: Dimension in X or [x y] **Y**: Dimension in X, default Y=X or X(2) **dx**: optional displacement to create a trapeze or conus cross section

Out: **PL**: 2D Point list

PLstar

returns a 2D point list of a star or star segment

In: **R**: Radius **nf**: number of facets (default is nofrd (R,0.05)) **dw**: circle segment]0..2*pi] (default is 2*pi) **Ry**: Radius of ellipse (default is R) **ow**: Offset angle **sk**: shortening factor; default 0.0

Out: **PL**: Point / Vertex list

3.2 Closed Nested 2D Polygons (CPL) [guide]

Functions in this section:

centerinsideCPL — HEURISTIC APPROACH: Finds in arbitrary contours the center point within a region

CListype — returns the type of correlation link

CLofFLW — returns a correlation list from a facet list between 2 free boundaries

closedCPL — closes all contours in CPL(2D) and CVL(3D) without any other change beside removal of all consecutive identical point before

closedCVL — closes all contours in CPL(2D) and CVL(3D) after removal of all consecutive identical point before

CLplot — plots a correlation for debugging (same as plotcorrelation)

CLsortrows — Sorts a corrleation list

colorsofCPLofimage — returns the original colors of the contours in an image

CPL2ccw — changes the orientation of all nested CPLs to ccw

CPL2cw — changes the orientation of all nested CPLs to cw

CPL2line — converts a line type contour into a line by considering only the half of the contour

CPLangle — creates an angle piece contour

CPLAnycubicPredator — Creates the CPL for the Anycubic Z-Level testing solid

CPLarea — returns the area of the surfaces (VL/PL)

CPLarrange — Arranges a set of CPLS to be as widw as another one in x dimension

CPLBezier — returns Bezier Curves from CPLs

CPLbox — create a CPL of a square including text

CPLbridgePL — Creates a brige between a point or frame to a CPL/CPS

CPLbuffer — grows or shrinks a CPL

CPLbufferCPLlines — converts a set of lines into a closed polygon concept

CPLbufferEdges — returns contours with broken our rounded edges;

CPLbufferlines — returns all buffer contours with an repeated reduction value

CPLcell2NaN — concatenates a cell list of CPLs into one single CPL

CPLcircbend — Provides a contour for a circular arc chord length and a circular arc chord height

CPLcirclesofPL — create a list of circles for a list of points

CPLcirclesofShaft — 2D fnct returning a circular contour that encloses a rotating contour

CPLconcat — returns a simple concatenation of CPls

CPLconnect — Creates a connecting contour between two contour

CPLcontourslots — creates slot contours to connect two solid by simple plug sockets

CPLconvexhull — returns the convex hull for a CPL

CPLconvexhulldelaunay — returns the convex hull for a CPL

CPLconvexhullfillgap — returns the convex hull for a CPL

CPLconvexhulloutline — creates a ONE region CPL from several regions of a CPL by using the convex hull

CPLcopyBravaislattice — copies a contour to a normed Bravais lattice

CPLcopypatternTL — copies a CPL at positions of a PL

CPLcopyradial — copies a CPL radial with radius and number of copies

CPLcopyunionradial — creates radial copies and unifies

CPLcrosscircseg — fnct to detect crossing segments of a circ line with a CPL

CPLcrossline — calculates the intersection of a CPL with a line

CPLcrosslineauxpoints — creates crossing points of a line a a contour with a grid size

CPLcutPez — cuts a CPL along a border line

CPLcutPshortest — finds from a point the shortest distance that cuts the contour into two pieces

CPLdecompose — returns a list of contour segments that belong to the same angle segment

CPLdistance — calculates the distance between two contours after supplementing colinear points

CPLdistancewall — returns the minimal wall thickness between 2 CPLs or a region

CPLdulledges — returns a CPL with dulled edges

CPLextendbyPL — extends a CPL outline contour by some points

CPLfeatureedge — selects corners with an angle larger a limit

CPLfft — returns a fourier frequency reduced contour shape

CPLfillgap — creates a contour that fills a another or several contours

CPLfillinside — creates a contour that fills a another or several contours

CPLfillonCPLcontour — finds positions for circles within a contour

CPLfilmhinge — creates a simple bar contour with rounded edges

CPLfindsmalldist — returns areas of a CPL that have a small wall distance to the opposite contour

CPLfischertechnik — returns the most common CPL for the design of fischertechnik compatible connections

CPLfit — fits a contour or a point list into a polygon

CPLfitinCPLrot — shifts a CPL into the center of another and rotates for optimal fit

CPLfitinCPLscale — Scales and moves contour to fit into another

CPLflipud — flips one by one the CPL up and down to change the order cw / ccw

CPLgripperjaw — creates a contour for a gripper design

CPLharbour — returns the inner contour of an area limited by line type areas

CPLhatch — creates a hatch for a CPL

CPLhatchgridofCPL — creates filling meshs similar to a slicer

CPLinsertPL — inserts removed points if they are on still exisiting lines

CPLinsertpoint — inserts a point at an appropriate condition into a CPL

CPLisccwcorrectedCPS — returns a CPL with all CPLs in correct orientation cw/ccw

CPLisidenticalcontour — returns whether two contours overlap the same area

CPLisidenticalregistered — Compares two CPLS in the same orientation and size

CPLlinkofPL — returns a convex hull including buffer for a list of points

CPLlinspace — sets the number of points of each embedded contour of a CPL to n linspace points

CPLmotorshaft — returns the CPL for a motorshaft, for instance Fischertechnik

CPLobjects2cells — returns CPL as cell list of CPLs separated by not-overlapping outer-border CPLs

CPLof2CPLzheuristic — returns the projection of a solid the would be created by SGof2CPLzheurist

CPLofCPS — returns a CPL from a CPS or an array of polyshapes

CPLofFS — returns a CPL of a Feature Surface

CPLofgcaborehole090 — returns the contour of the shadow core in view (0,90) from above

CPLofouterregionboundary — returns the outer boundary of regions of CPL

CPLofPLalphashape — returns a closed polygon contour around a point list using alphashape

CPLofPLcontour — Converts a PL into a CPL, if the distance of last point and first point is below a threshold

CPLofPLcrossCPL — cuts a line by crossings using a CPL

CPLofPLgrow — creates CPL circles around points of a point list

CPLofPLgrowline — converts a single open polygon into a grown contour

CPLofPLseparatedlines — converts a list of 2D points paris into a CPLs that separates the each pair by nan nan

CPLofPolygon — returns the CPL of a graphics object type polygon

CPLofpolyshape — creates a CPL from an array of polyshapes

CPLofprocessedimage — returns contours inside a processed image

CPLofSGboreholes — returns the contour of the shadow core in view (0,90) from above

CPLofSGcutT — separates a solid similar to SGcutT but inserts connection points

CPLofSGcutTBB — returns a CPL based on the Bounding Box of a solid

CPLofSGFSselect — returns the CPL of a SG by FS selection

CPLofSGhull — returns the outer contour of a solid

CPLofSGprojection — returns a projection contour of a solid using the graphics board

CPLofSGTez — returns the CPL and the frames plus a frame list using TofCPLT

CPLofSGThull — Creates the CPLofSGhull after an transformation relative to a frame

CPLofspokedPL — converts a point list into a spokes between the most distance points

CPLoftesthinge — returns a shape and a Geometry for a flexible hinge with 180 degree radius

CPLopendistance — returns the maximal distance between neighbor points of a contour related to the contour size

CPLouterboundaries — All overlapping outer boundary contour of a CPL

CPLoutlinecontour — just returns ALL OUTLINE contours

CPLpinholesinCPL — returns the CPLs of the pins that would fit into bore holes of servos or PCB

CPLplotasPS — plots a CPL as polyshape using the region plot of polyshapes command

CPLplotbufferlines — plot buffer lines for a contour

CPLplotcolors — plots all contours or indexed contours in colors

CPLplotcwccw — plots a CPL in different colors depending on the orientation

CPLplotinout — plots the CPLs in colors depending in the enclosure level

CPLplotregions — plots the regions of a CPL / polyshape in different colors

CPLrack4PL — returns a contour for rack that can be used as rack (ground link for linkages)

CPLrackPLdelaunay — returns a minimal CPL bar structure that contains all points of PL

CPLrackPLdelaunaypath — same as CPLrackPLdelaunaypath, but allow forbidden areas

CPLradialconcaveEdges — concave edges in CPL based on polyshape and polybuffer

CPLradialhatch — create radial hatches between contours 1 and 2

CPLradialslots — creates rotating circle segment contour for socket connectors

CPLregionbufferintersect — detects the overlapping area of increased regions

CPLregions — sorts a CPL with respect to regions

CPLregions2cell — returns CPL as cell list of CPLs separated by not-overlapping outer-border CPLs

CPLregionsholes — separates CPLs into Surfaces and Holes

CPLregionshrink — returns the smallest shrinked contour with same dimensions, regions and holes

CPLremauxpoints — removes points from a CPL or a PL that

CPLremconvexcorner — remove single convex corners from the contour

CPLremdensity — removes points from a CPL or a PL that

CPLremFFT — removes points from a CPL by removing high frequencies

CPLremovearea — removes CPLs of small area using the polyshape class

CPLremovebuffer — just removes thin walls from CPLS (same as CPLdulledges)

CPLremovecollinearpoints — removes 2D/3D collinear point with different results for opened and closed CPL/CVL

CPLremoveoverlap — For each contour of CPL the polyshape is simplified

CPLremoveregionsholes — removes CPLs of small areas and small holes using the polyshape class

CPLremovethinwall — just removes thin walls from CPLS same as CPLdulledges and CPLremovebuffer

CPLremstraightAmin — removes points with a minal distance/area to a predecessor or successor

CPLrepair — returns a CPL for extrusion - finding, duplicating and separating split points (required for SGLib 5.4)

CPLRofCPL — uses the nan separated point list to create contours using PLgrowline

CPLselectbynearestP — returns an index list for selectNaN to select by the nearest Points/CPLs

CPLselectbyP — returns an index list to select enclosing CPLs

CPLselectinout — returns selected CPLs based on their enclosure index

CPLselectregionbyP — select a polyshape region (given als CPL) by an interior point

CPLselectspatial — selects parts of CPL based on spatial or other relations

CPLsetallcw — sets all contours of a CPL to cw

CPLsimplify — simplifies a CPL using polyshape simplyfy true

CPLskeleton — returns a skeleton with equal distances to all wall sizes

CPLskeletonsearch — creates a point list and an edge list

CPLsocketscrewed — Contour for an anti-rotation bushing

CPLsplitui — splits a CPL interactively into two CPLs

CPLspringSpiral — creates just the pathe and basic contour for a windup motor spring spiral

CPLsquareline — creates a sequence of boxes of different x-dimensions

CPLstent — creates a CPL pattern for a laser cutted stent

CPLstentline — creates a sequence of boxes of different x-dimensions

CPLsweep — returns a swept contour

CPLsweep_simple — 1st sweep fnct

CPLtextbanner — Returs a CPL for an size adjusted Textbanner or Contour

CPLui — interactive fnct to create and modify CPLS (Only one Contour)

CPLunionpatternPL — creates copied and united contours

CPLunionreplace — countour union after a subtraction of a bounding box or convex hull

CPLunionsorted — For 2.5D Design in Mechanical Engineering, there is a different CPLunion required

CPLunsort — changes the order of nan separated CPL using unsortNaN

CPSdiffarea — return the norm difference between two polyshape areas by xor

CPSlineui — interactive fnct to create and modify CPS lines (Only one Contour)

CPSofgca — returns a struct called Colored PolyShape (CPS) similar to a polygon array object

CPSofgcalines — converts all or selected lines graphcis objects into CPS structs using a line thickness

CPSofpolygon — returns a struct called Colored PolyShape (CPS) similar to a polygon array object

CPSouterboundary — returns the outer boundary as CPS or polyshape

CPSplot — plots a colored polygon from a CPS struct, a polyshape or a CPL (NO SIMPLIFICATION)

CPSremovearea — removes CPS/polyshape regions of small area

CPSsimilaritymap — converts an array of CPS or polyshapes into a cell list of similar CPS

CPSui — interactive fnct to create and modify CPLS (Only one Contour)

ELofPL — creates an EL for a PL of succeeding points

GPLauxgridpointsCPS2 — creates a defined number of Gridpoints

isCVP — returns whether a sequence of vertices is a contour vertex path

openedCPL — opens all contours in CPL(2D) and CVL(3D) without any other change beside removal of all consecutive identical points.

openedCVL — closes all contours in CPL(2D) and CVL(3D) after removal of all consecutive identical points.

PLcirctrapezoid — creates a trapezoid contour with a predefied angle

PLconnectopenradial — closes an open line by a contour

PLdecompose — returns a list of contour segments that belong to the same angle segment

PLdiamond — returns a diamond shape optonal als convex hull for a CPL

PLgrowline — converts a single open polygon into a grown contour

PLofCL — returns an arbitrary planar point list to vizualize the correspondance list

plotcorrelation — plots a correlation for debugging

PLshortestpathinCPL — returns a path inside of a CPL using A* Zelinzky, Lozano-Perez, Dikstra

PLui — interactive fnct to create and modify CPLS (Only one Contour)

PSanalyze — classifies the polyshapes regions outer boundaries into equal shapes

PSofgca — return the polyshapes of the polygons of current axis

PSregions — returns the regions of a polyshape

R12ofCPL — return standardized link and joint dimensions

RLofCVL — Returns Radii of a contour, same as CVLdimclassifier

SGofCPLflat — returns a SG struct of an open flat surface from a polyshape or closed polygon

TofCVL — returns for curved contours the mean transformation matrix

TofPL — returns a 2D HT matrix for a point list

VLNLOLplot — plot an open or closed CVL including Edge Normal Vector and Edge Orthogonal Vector

VLpathdist — distance between two points on a CVP PAth

Detailed description

centerinsideCPL

HEURISTIC APPROACH: Finds in arbitrary contours the center point within a region

In: **CPL**: Contour to analyze **cc**: default is 0.5 for middle of skeleton line **MS**: suggested center points, such as mean of CPL; default is false **"sep"**: if used, the contours are treated separetedly and M has a 3rd coordinate as index of selectNAN(CPL)

Out: **M**: Center point **PL**: Point list of last line

CListype

returns the type of correlation link

In: **CL**: Correlation list **ind**: [ai bi], [nan bi], [ai nan]

Out: **res**: Type of link **ind**: Correlation request as [ai bi], [nan bi] or [ai nan]

CLofFLW

returns a correlation list from a facet list between 2 free boundaries

In: **FLW**: Facet list of a wall between two free boundary contours

Out: **CL**: Correspondance list

closedCPL

closes all contours in CPL(2D) and CVL(3D) without any other change beside removal of all consecutive identical point before

In: **CPL**: Contour 2D or 3D with NaN embedded contours **lim**: default is 1e-12 for removal of succeding points

Out: **NCPL**: New contours 2D/3D with all closed contours without any consecutive identical point

closedCVL

closes all contours in CPL(2D) and CVL(3D) after removal of all consecutive identical point before

In: **CVL**: Contour 2D or 3D with NaN embedded contours **lim**: default is 1e-12 for removal of succeding points

Out: **NCVL**: New contours 2D/3D with all closed contours openedCPL, closedCPL, without any consecutive identical point

CLplot

plots a correlation for debugging (same as plotcorrelation)

In: **CL**: Corrleation list (or FL) **w**: Width of lines

CLsortrows

Sorts a corrleation list

In: **CL**: Correlation list

Out: **CL**: Sorted Correlation list

colorsofCPLofimage

returns the original colors of the contours in an image

In: **CPL**: Contour created by CPLofimage(I) **I**: Image

Out: **coli**: list of colors wrt to separateNaN **cols**: color list of each point of CPL

CPL2ccw

changes the orientation of all nested CPLs to ccw

In: **CPL**: CPL with nested CPLS

Out: **CPLN**: CPL with nested CPLS all in ccw

CPL2cw

changes the orientation of all nested CPLs to cw

In: **CPL**: CPL with nested CPLS

Out: **CPLN**: CPL with nested CPLS all in cw

CPL2line

converts a line type contour into a line by considering only the half of the contour

In: **CPLo**: Original CPL **dx**: expected distance between both boundaries; default is 0.1

Out: **CPLN**: PL of contour **b**: estimated width of the contour (area divided by length) **a**: area of the contour **l**: perimeter length of the contour

CPLangle

creates an angle piece contour

In: **P0**: Center point **L**: length of the legs **wlim**: minimum angle and maximum angle

Out: **CPLW**: Triangle cut out **PLN**: point list of the arc between the two legs

CPLAnycubicPredator

Creates the CPL for the Anycubic Z-Level testing solid

Out: **CPL**: CPL of the contour of SGsample4APredator

CPLarea

returns the area of the surfaces (VL/PL)

In: **CPL**: Closed Polygon List

Out: **ASUM**: Area sum of the surface boundaries **A**: Area list for facet list

CPLarrange

Arranges a set of CPLS to be as widw as another one in x dimension

In: **CPLC**: is a cell list of CPLs **CPL2**: is the dimension defining contour

Out: **CPLC**: Is an NAN separated CPL

CPLBezier

returns Bezier Curves from CPLs

In: **CPL**: CPL **ns**: number of Beziercoefficients

Out: **CPLB**: Contours of CPL as Bezier Curves

CPLbox

create a CPL of a square including text

In: **sze**: [x y]; default is 10 **txt**: chars for labeling

Out: **CPL**: CPL of Contour

CPLbridgePL

Creates a brige between a point or frame to a CPL/CPS

In: **CPL**: Contour to attach to **PL**: Point, 2 points or 3x3 HT Matrix | x direction is used if more than a single point **R**: Radius of bridge or contour

Out: **CPLN**: Complete new contour including the pridge to the frame/point **pcc**: contact point(s) on the contour nearest to the frame/point

CPLbuffer

grows or shrinks a CPL

In: **CPL**: Closed Polygon line **d**: Shifting distance **typ**: 'round'—Round out, 'square'—Square off, 'miter'—Preserve angle **simpl**: if true, simplify the polyshape (may split contours); default is true **keep**: if true, keep collinear points; default is true (ignored by polybuffer)

Out: **CPLN**: Closed Polygon line

CPLbufferCPLlines

converts a set of lines into a closed polygon concept

In: **CPL**: CPL consisting of open lines without meshes **d**: buffer width

Out: **CPLN**: New CPL or CPS

CPLbufferEdges

returns contours with broken our rounded edges;

In: **CPL**: Closed Polygon Line **r**: radius; default is 0.3 **t**: 'square', 'round'

Out: **CPLN**: Resulting buffered closed polygon list

CPLbufferlines

returns all buffer contours with an repeated reduction value

In: **CPL**: Closed Polygon Line **w**: reduction value

Out: **CPLC**: Cell list with {reduction value and CPL} for all repeated values

CPLcell2NaN

concatenates a cell list of CPLs into one single CPL

In: **CPL**: Cell list of CPLs

Out: **CPLN**: CPL that is a simple concatenation of all contours in CPL

CPLcircbend

Provides a contour for a circular arc chord length and a circular arc chord height

In: **s**: circular arc chord length **h**: circular arc chord height/width **tW**: turning angle

Out: **PL**: open CPL

CPLcirclesofPL

create a list of circles for a list of points

In: **PL**: Center point list **r**: radius of the circles

Out: **CPL**: Closed Polygon line

CPLcirclesofShaft

2D fnct returning a circular contour that encloses a rotating contour

In: **CPL**: Contour of a rotating shaft **slt**: optional slot size; default is 0;

Out: **CPLC**: CPL of enclsoing circl **r**: radius **cp**: centerpoint

CPLconcat

returns a simple concatenation of CPls

In: **A**: CPL A **B**: CPL B

Out: **CPL**: Resulting Boolean

CPLconnect

Creates a connecting contour between two contour

In: **CPLA**: Contour A **CPLB**: Contour A **ma**: minimal area for connection **tp**: buffer type passed to CPLbuffer; default is 'round'

Out: **Y**: connecting contour (CPL) linking contour A and B

CPLcontourslots

creates slot contours to connect two solid by simple plug sockets

In: **CPL**: Outer Contour **dlw**: length width distance **nsl**: number of slots

Out: **CPLs**: Final slot contour

CPLconvexhull

returns the convex hull for a CPL

In: **CPL**: CPL **buf**: buffer distance applied to the hull via CPLbuffer; default is 0

Out: **CPLN**: convex hull

CPLconvexhulldelaunay

returns the convex hull for a CPL

In: **CPL**: CPL

Out: **CPLD**: convex hull of the delaunay triangulation

CPLconvexhullfillgap

returns the convex hull for a CPL

In: **CPL**: CPL **d**: distance to fill the gap; default is 0.1 **ftyp**: edge type 'default is 'square' **keep**: colinear points; default is false

Out: **CPLD**: convex hull of the delaunay triangulation **dn**: gap distance actually used, doubled iteratively until a single contour is found

CPLconvexhulloutline

creates a ONE region CPL from several regions of a CPL by using the convex hull

In: **CPL**: Original CPL with more than one region **b**: buffer distance; default is 0 **r**: radius of edges; default is 0 **t**: type of convex hull: 'default', 'delaunay', 'fillgap' **keep**:

Out: **CPLN**: Final CPL

CPLcopyBravaislattice

copies a contour to a normed Bravais lattice

In: **CPL**: Contour to copy **A**: row vectors if Bravais lattice vectors **CPLS**: Optional enclosing contour **d**: additional distance between the pattern

Out: **CPLB**: Contours copied to the Bravais lattice nodes

CPLcopypatternTL

copies a CPL at positions of a PL

In: **CPL**: CPL to copy (nx2) **TL**: array of transformation matrices such from TLofPL **dw**: optional turing angle or turning array [nx1]; default is 0

Out: **CPLN**: Final CPL [nx2]

CPLcopyradial

copies a CPL radial with radius and number of copies

In: **CPL**: Original Contour **R**: Radius **n**: number of copies **ow**: offest angle

Out: **CPL**: copied contours **PL**: point list of the copy center positions on the radius

CPLcopyunionradial

creates radial copies and unifies

In: **CPL**: Contour to copy; default is PLsquare (2 10) **ofs**: Radius; offset in [x and y] default is [0 8] **n**: radius segments; default is 16

Out: **CPLN**: Final contour

CPLcrosscircseg

fnct to detect crossing segments of a circ line with a CPL

In: **CPLA**: CPL of restricted Area **A0**: Center of circular movement **wlim**: angle interval of circular movement; default is [0 2*pi]

Out: **wins**: list of crossed angle segments (right hand) **wout**: inverse angle segments of wlim (right hand)

CPLcrossline

calculates the intersection of a CPL with a line

In: **CPL**: CLosed polygon region **PL**: Point list **acc**: accuracy, line thickness; default is 0.01

Out: **CPLX**: Crossing Area

CPLcrosslineauxpoints

creates crossing points of a line a a contour with a grid size

In: **CPL**: Contour **p1**: Start Point **p2**: End Point **md**: distance; default is 1mm

Out: **GPL**: Gridpointlist

CPLcutPez

cuts a CPL along a border line

In: **CPL**: Closed Polygon List **cp**: center Point **ev**: unit vector to erase from

Out: **CPR**: cutted contour

CPLcutPshortest

finds from a point the shortest distance that cuts the contour into two pieces

In: **CPL**: Contour ro cut **p**: point to cut or empty for testing **tol**: tolerance for starting points; default is [1] **d**: distance for cut; default is 0.01; **Radd**: Radius for adding a circle at the cutting point to alle cutted contours **"sort"**: can be 'ascend' or 'decend'; default is 'decend'

Out: **[CPLR]**: cell array of cutted contours optionally sorted

CPLdecompose

returns a list of contour segments that belong to the same angle segment

In: **CPL**: CPL **da**: angle; default si pi/6 **mix**: if true; minxy is used to start with segment 1

Out: **SIL**: Separation Index List

CPLdistance

calculates the distance between two contours after supplementing colinear points

In: **CPLA**: Contour A **CPLB**: Contour B **d**: sample distance **lim**: if true d is limit not sample distance

Out: **dp**: minimal distance **PLA**: List of Points on CPLA (including auxiliary points) **PLB**: List of Points on CPLB (including auxiliary points)

CPLdistancewall

returns the minimal wall thickness between 2 CPLs or a region

In: **CPLo**: Outer CPL **CPLi**: Innver CPL **np**: number of equidistant sample points per contour; default is 100

Out: **wmin**: maximal wall thickness **wmax**: maximal wall thickness

CPLdulledges

returns a CPL with dulled edges

In: **CPL**: CPL **d**: distance between edges **typ**: miter (sharp edges) or square (alle egdes) or round (all edges)l default is miter

Out: **CPLN**: New CPL

CPLextendbyPL

extends a CPL outline contour by some points

In: **CPL**: Exisintg CPL **PL**: New Points **lim**: limit until extension default is s **buf**: buffer arount PL; default is s/10

Out: **CPLC**: New Contour

CPLfeatureedge

selects corners with an angle larger a limit

In: **CPL**: Closed Polygon list **al**: alpha; default is pi/4 (45 degree)

Out: **FE**: List of Points **FEi**: Index of FE within CPL

CPLfft

returns a fourier frequency reduced contour shape

In: **CPL**: Contour **nf**: Number of used frequencies **np**: number of points of the CPL; default is 100;

Out: **CPLA**: Resulting Contour

CPLfillgap

creates a contour that fills a another or several contours

In: **CPL**: Closed Polygon List **d**: gap distance to bridge; by default 1% of maximum size length **ftyp**: buffer joint type 'round','square', or 'miter'; default is 'square' **keep**: if true, keep collinear points; default is false

Out: **CPLN**: new polygon

CPLfillinside

creates a contour that fills a another or several contours

In: **CPL**: Closed Polygon List **Amin**: Minimum surface to be interested **Asam**: percentage of maximum area ; default a=90%, 0.9

Out: **CPLN**: inverse polygon

CPLfillonCPLcontour

finds positions for circles within a contour

In: **CPL**: Closed Polygon Line **CPLA**: circle radius or if negative number of points

Out: **PL**: point list of circle center positions inside the contour **ri**: circle radius used

CPLfilmhinge

creates a simple bar contour with rounded edges

In: **L**: Length **B**: Width/Height must be smaller than L **R**: Radius at the end

Out: **PL**: final Point list

CPLfindsmalldist

returns areas of a CPL that have a small wall distance to the opposite contour

In: **CPL**: Closed Contour **z**: minimalcontour thickness; default is 0.5mm

Out: **CPLF**: Lost Contours; if empty, the contour has no problems **CPLX**: Remaining Contour;

CPLfischertechnik

returns the most common CPL for the design of fischertechnik compatible connections

In: **tol**: [tolerance between the lines CPLL and CPLI[a **gb**: length of contour; default is 15 mm

Out: **CPLL**: Line to create a blade or concave contour (rail, blade) **CPLI**: Line to create a convex contour (knob) **CPLC**: Line to create the concave contour for the axle **PL3**: Line to create the concave contour for 2 axles **PL4**: Line to create the concave contour for 4 axles

CPLfit

fits a contour or a point list into a polygon

In: **CPLA**: Outside shape of a contour **CPLB**: Contour to fit in or Point list for circles **buff**: buffer for contour or buffer or point list **points**: if true CPLB is used as point list for circles

Out: **CPLN**: Contour or Point list that fits **mb**: relative shift **f**: scaling factor

CPLfitinCPLrot

shifts a CPL into the center of another and rotates for optimal fit

In: **CPLA**: CPL to fit CPLB in **CPLB**: CPL to be rotated to fit into CPLA **n**: number of angeles to try for best fit; default is **rec**: if true, the angle search is refined recursively; default is false

Out: **CPLN**: New Conotur **wf**: rotation angle

CPLfitinCPLscale

Scales and moves contour to fit into another

In: **CPLA**: Contour to fit in **CPLB**: Contour that has to be scaled to fit in CPLA **bs**: border size; default is 0

Out: **CPLB**: final scaled contour

CPLflipud

flips one by one the CPL up and down to change the order cw / ccw

In: **CPL**: Closed Polygon Line; contours separated by nan nan

Out: **CPLN**: CPL with each contour reversed (cw/ccw flipped) keeping contour order

CPLgripperjaw

creates a contour for a gripper design

In: **CPL**: Object contour to be grasped (Orange) **P0**: Position of TCP **CPL0**: Contour around TCP (green) **hg**: depth of gripper jaw into the object **convh**: if true, use only the convex hull of the object; default is false

Out: **CPLF**: Final Gripper Jaw Contour (red contour) **CPLF0**: Final Gripper Jaw Contour (red contour) with origin in P0

CPLharbour

returns the inner contour of an area limited by line type areas

In: **CPL**: CPL or Polyshape **inside**: true = inside; false = outside area **d**: distance to bridge

Out: **CPLN**: CPLN **ps**: polyshape

CPLhatch

creates a hatch for a CPL

In: **CPL**: CPL **d**: thickness and distance as multiple **al**: angle in degree; default is 60 degree **co**: contours to select

Out: **CPLI**: CPL of hatch by intersection => Thin lines=> Thin hatch **CPLS**: CPL of hatch by subtraction => Thin lines=> Thick hatch **mm**: center point **d**: thickness and distance as multiple

CPLhatchgridofCPL

creates filling meshs similar to a slicer

In: **CPLOrg**: Original contour of a solid cross section/projection or a solid **dim**: Wall and bar width and distance default is [2 20] **"fill"**: if used, the contour is filled solid via CPLfillinside before hatching

Out: **CPLF**: Contoor with a simple squared mesh

CPLinsertPL

inserts removed points if they are on still exisiting lines

In: **CPL**: arbitrary CPL **PL**: unosrted list of points that may be on lines of CPL **tol**: maximum distance

Out: **CPLN**: New CPL with inserted Points if possible

CPLinsertpoint

inserts a point at an appropriate condition into a CPL

In: **CPL**: CPL **p**: point **tol**: tolerance for inserting a new point

Out: **CPLN**: New CPL with integrated Point **ea**: information on contour the point was inserted **eaa**: information on all contours ov CPL **ok**: false if the point was to near

CPLisccwcorrectedCPS

returns a CPL with all CPLs in correct orientation cw/ccw

In: **CPL**: Original CPL **keepcolinear**: if true; the cpl is simplified before **simplify**: UNCHANGABLE TRUE

Out: **CPL**: Corrected CPL

CPLisidenticalcontour

returns whether two contours overlap the same area

In: **CPL1**: CPL of contour 1 **CPL2**: CPL of contour 2 **thr**: threshold; default is 1e-3

Out: **b**: logical value **a**: area difference **CPLD**: Difference contour between the two CPLs

CPLisidenticalregistered

Compares two CPLS in the same orientation and size

In: **CPLA**: 1st CPL **CPLB**: 2nd CPL **tol**: tolerance in relative difference area; default is 0.1

Out: **b**: logical information **A**: difference in area

CPLlinkofPL

returns a convex hull including buffer for a list of points

In: **PL**: Point list **s**: outer radius and inner radius [ro ri]; or default values

Out: **CPL**: Closed contour

CPLlinspace

sets the number of points of each embedded contour of a CPL to n linspace points

In: **CPL**: CPL with enclosed CPLs **n**: number of points per contour; default is n **all**: if true; also contours with less than n points are processed; default is false

Out: **CPLN**: New CPL with n points per contour

CPLmotorshaft

returns the CPL for a motorshaft, for instance Fischertechnik

In: **Ra**: Radius of the largest Diameter; default is 2.5 (5mm axis) **Rc**: Radius distance from the center to the plate; default is 1.5 **tW**: turning offset in degree **"diam"**: option to used values comming from a caliper

Out: **PL**: Point List for CPL

CPLobjects2cells

returns CPL as cell list of CPLs separated by not-overlapping outer-border CPLs

In: **CPL**: NaN separated CPL

Out: **CPLN**: cell list of CPLs separated by not-overlapping outer-border CPLs

CPLof2CPLzheuristic

returns the projection of a solid the would be created by SGof2CPLzheurist

In: **CPLA**: Base Contour **CPLB**: Top contour **z**: extrusion height (has an effect)

Out: **CPL**: proection contour

CPLofCPS

returns a CPL from a CPS or an array of polyshapes

In: **CPS**: CPS or polyshape

Out: **CPL**: CPL

CPLofFS

returns a CPL of a Feature Surface

In: **SG**: Solid Geometry **fil**: facet index list; or ez vector;

Out: **CPL**: CPL **T**: Transformation matrix for CVL ORIENTATION IS NOT CONSIDERED ! **ez**: ez vector used

CPLofgcaborehole090

returns the contour of the shadow core in view (0,90) from above

Out: **CPL**: Contour of shadow

CPLofouterregionboundary

returns the outer boundary of regions of CPL

In: **CPL**: Closed polygon line **sline**: additional line thickness; default is 1% of BB of CPL

Out: **CPLN**: outer boundary of regions **CPLO**: inner offset contour by sline (empty unless sline>0)

CPLofPLalphashape

returns a closed polygon contour around a point list using alphashape

In: **PL**: Original Point cloud

Out: **CPL**: Closed Polygon List of contour (endpoint=startpoint) **PL**: Point list (doubled entries removed) **FL**: Facet List **EL**: Edge list of free boundary sorted **CEL**: Contour index list

CPLofPLcontour

Converts a PL into a CPL, if the distance of last point and first point is below a threshold

In: **CPL**: Closed Polygon line or Point List (separation by nan is possible) **s**: threshold in percent if <.5 or absolut if s>0.5 **edge**: treu=round edges; default is false **caps**: treu=round caps; default is false

Out: **CPL**: Resulting contour (CPL)

CPLofPLcrossCPL

cuts a line by crossings using a CPL

In: **CPL**: Closed Polygon line **PL**: Point List **LL**: Optional Length or angle list related to the points of PL

Out: **NPL**: Point list separated by the crossing **FFL**: Interpolated Value List

CPLofPLgrow

creates CPL circles around points of a point list

In: **PL**: point list [n x 2] **R**: Radius

Out: **CPL**: Resulting closed polygon list of grown circles

CPLofPLgrowline

converts a single open polygon into a grown contour

In: **PL**: Open point list line **s**: thickness **edge**: true= rounded edges; false = simple; default false; **caps**: true= rounded caps; false = straight; default is false;

Out: **CPL**: Closed Polygon line

CPLofPLseparatedlines

converts a list of 2D points paris into a CPLs that separates the each pair by nan nan

In: **PL**: List of point [nx2] n is an even integer

Out: **CPL**: Pairs separated by nan nan

CPLofPolygon

returns the CPL of a graphics object type polygon

In: **h**: handle to polygon **ascell**: if return as cell list or as array

Out: **CPL**: Closed Polygon line

CPLofpolyshape

creates a CPL from an array of polyshapes

In: **ps**: polyshape **ascell**: returns polyshape array elements as cell list

Out: **CPL**: CPL

CPLofprocessedimage

returns contours inside a processed image

In: **I**: Image **ms**: optional resize value; default is 1024; 0==unchanged **fspecial**: optional filter values - see at fspecial

Out: **CPL**: Resulting contour (CPL) **I**: Processed image

CPLofSGboreholes

returns the contour of the shadow core in view (0,90) from above

In: **SG**: Solid Geometry **ez**: direction vector; default is [0 0 1] **ms**: minimal diameter of hole; default is 4; neg => holes only

Out: **CPL**: Contour of shadow **a**: area of contours

CPLofSGcutT

separates a solid similar to SGcutT but inserts connection points

In: **SG**: Solid Geometry **T**: Cutting frame; Z-Achsis spearates **pp**: point relative to T(!) that define the selected contours of the crossing CPL for example [0 0] **rh**: radius and height of pins, used only for result plot and as educationb

Out: **CPL**: Crossing Contour, The pins are ignored **T**: HT Matrix **r**: maximal radius of contour relative to T

CPLofSGcutTBB

returns a CPL based on the Bounding Box of a solid

In: **SG**: Solid Geometry **zz**: [zmin zmax]; default is [-inf +inf]);

Out: **CPL**: CPLofBB of the considered Solid shape between zmin and zmax

CPLofSGFSselect

returns the CPL of a SG by FS selection

In: **SG**: Solid Geometry **FSselectors**: FS selector sequence

Out: **CPLC**: Contour **T**: Koordinate System to move the CPL in space - Different to T in SGLib 5.3

CPLofSGhull

returns the outer contour of a solid

In: **SG**: Solid Geometry **T**: Optional Frame Name or HT for separating the solid

Out: **CPLXY**: Contour blue **CPLXZ**: Contour green **CPLYZ**: Contour red **T**: HT Matrix of the Frame **NPLXY**: Outline point list projected onto the XY plane **NPLXZ**: Outline point list projected onto the XZ plane **NPLYZ**: Outline point list projected onto the YZ plane

CPLofSGprojection

returns a projection contour of a solid using the graphics board

In: **SG**: Solid Geometry **T**: Projektion Orientation or just ez vctor; default is eye(4)

Out: **CPL**: CPL to be projected using T **CVL**: CVL already projected CPL **T**: projection frame used

CPLofSGTez

returns the CPL and the frames plus a frame list using TofCPLT

In: **SG**: Solid Geomtry **T**: Frame of Surface or ez vector **nr**: [number-of-points, radius, border]; default is [0 1 0] **"style"**: 'contour', 'grid', 'xgrid'; default is 'contour'

Out: **CPLorg**: Contour of surface **TF**: Frame to create CVL from CPL **TL**: List of Frames **ACVL**: cell list of CVLi;that defined the contours

CPLofSGThull

Creates the CPLofSGhull after an transformation relative to a frame

In: **SG**: Solid Geometry **T**: Frame char or HT matrix

Out: **CPL**: CPL of contour (XY after transformation) **T**: HT matrix of transformation **CVL**: CVL of the spatial transformed CPL

CPLofspokedPL

converts a point list into a spokes between the most distance points

In: **PL**: Point list **wt**: Radius = half spoke thickness **rd**: Radius to avoid notch effect inside; default is 0.5 **CPLA**: optional contour of a shaft or axis

Out: **CPL**: resulting CPL

CPLoftesthinge

returns a shape and a Geometry for a flexible hinge with 180 degree radius

In: **rmin**: minimal radius; default is 0.5 **delta**: Thickness of circle **sx**: size in x ; default is 10 mm **sy**: size in y ; default is 5 mm **txt**: text; default is rminand delta **h**: height of the solid; default is 1 **alpha**: opening angle of the hinge in radians; default is pi/4

Out: **CPL**: Closed Polygon line **SG**: Solid Geometry

CPLopendistance

returns the maximal distance between neighbor points of a contour related to the contour size

In: **CPL**: CLosed Polygon line

Out: **DP**: maximum distance in relation to bb of CPL **DN**: maximum distance between two points

CPLouterboundaries

All overlapping outer boundary contour of a CPL

In: **CPL**: Original Contour

Out: **CPLN**: Outer Boundary

CPLoutlinecontour

just returns ALL OUTLINE contours

In: **CPL**: CPL consisting of severalcontours

Out: **CPLN**: New CPL

CPLpinholesinCPL

returns the CPLs of the pins that would fit into bore holes of servos or PCB

In: **CPL**: CPL created by CPLofSGboreholes **rmin**: minimal radius to consider; default 2 mm **bord**: border of the outer contour that should not be touched by the pins; default 1mm

Out: **CPLH**: CPL of pins that fit into the bore holes

CPLplotasPS

plots a CPL as polyshape using the region plot of polyshapes command

In: **CPL**: closed polygon line **simp**: true for simplify; false=unmodified; default is false;

Out: **h**: handle

CPLplotbufferlines

plot buffer lines for a contour

In: **CPL**: CLosed Polygon Line **w**: distance between the buffer lines, default is maximal 15 lines

Out: **h**: handle to graphics

CPLplotcolors

plots all contours or indexed contours in colors

In: **CPL**: Closed Polygon line **ind**: indizes; default is all **fs**: optional font size **cl**: line color/style; default is 'r-'

Out: **h**: handle to graphics objects

CPLplotcwccw

plots a CPL in different colors depending on the orientation

In: **CPL**: Closed Polygon Line **anno**: if true; there is a plotannotation **wdth**: line width

Out: **h**: handle to drawing and annotation

CPLplotinout

plots the CPLs in colors depending in the enclosure level

In: **CPL**: Closed Polygon line **col**: line type; first char (color) is ignored

Out: **h**: handle to plot

CPLplotregions

plots the regions of a CPL / polyshape in different colors

In: **CPL**: Closed polygon **ind**: index of regions; default is all **simp**: flag to simplify the polyshape; default is true

Out: **h**: handle to graphics objects

CPLrack4PL

returns a contour for rack that can be used as rack (ground link for linkages)

In: **PL**: Point list such as [A0;B0] **b**: buffer; distance required to implement a revolute joint **t**: 'minimal', 'circ' or 'box', 'delaunay', 'spokes'; 'starr default is 'min' **d**: if used; the points are cut out with this radius

Out: **CPLF**: Contour to carry revolute joints for linkages

CPLrackPLdelaunay

returns a minimal CPL bar structure that contains all points of PL

In: **PL**: Point list **b**: buffer size around PL **c**: if true, a closed triangle is created in the special case of only three points

Out: **CPL**: final contour

CPLrackPLdelaunaypath

same as CPLrackPLdelaunaypath, but allow forbidded areas

In: **PL**: Point list **b**: buffer size around PL **CPLW**: Forbidded Area; ; default is empty **CPLG**: No cost area; predefined rack area; default is empty **slc**: slot size or fitting letter applied to the forbidden area; default is 0

Out: **CPL**: final contour

CPLradialconcaveEdges

concave edges in CPL based on polyshape and polybuffer

In: **CPL**: CPL contour **rrr**: radius for concave edges

Out: **CPLN**: CPL Contour with radial concave edges

CPLradialhatch

create radial hatches between contours 1 and 2

In: **CPL**: Tube style contour to fill **wl**: wall thickness; default is 1.2mm **cs**: Contour/region to fill 0 ist outside; ; default is [1 2] **n**: number of strutts

Out: **CPL**: Final Contour **CPLN**: Stutted Contour

CPLradialslots

creates rotating circle segment contour for socket connectors

In: **Riw**: [Ri Ro] **ls**: length of the slot **nmax**: maximum number of slots **off**: offset angle

Out: **CPLN**: Complete Conotur **CPL**: 1st Segment **nn**: Number of slots used

CPLregionbufferintersect

detects the overlapping area of increased regions

In: **CPL**: Original CPL **dx**: buffered

Out: **cc**: cell list if overlapping polyshapes **vv**: list of comparision [i k] **reg**: cell list of regions

CPLregions

sorts a CPL with respect to regions

In: **CPL**: Closed Polygon Line

Out: **CPLN**: Sorten to regions **PiL**: [parent-index of CPLN, region leve, 0 = outsidel]

CPLregions2cell

returns CPL as cell list of CPLs separated by not-overlapping outer-border CPLs

In: **ps**: NaN separated CPL **issort**: if 'ascend', or 'descend' the regions are sorted **slct**: sequences or scalar indices or non integer area threshold

Out: **CPL**: cell list of CPLs separated by not-overlapping outer-border CPLs **A**: Aread of the regions

CPLregionsholes

separates CPLs into Surfaces and Holes

In: **CPL**: Original CPL

Out: **CPLo**: Outside Conturs to add (RED) **CPLi**: Hole Contours to substact (GREEN)

CPLregionshrink

returns the smallest shrinked contour with same dimensions, regions and holes

In: **CPL**: Contour to shrink **dlim**: accuracy; default is 0.01 **rsep**: if true; the limit is individual to each function; default is true;

Out: **CPLo**: shinked contour with same number of regions and holes **dx**: limit of last shinkage **CPLf**: contour at the last shrinkage step before breaking

CPLremauxpoints

removes points from a CPL or a PL that

Out: **PL**: CPL with minimal distances or PL in case of a cloud

CPLremconvexcorner

remove single convex corners from the contour

In: **CPL**: Contour

Out: **CPLN**: contour with removed corners

CPLremdensity

removes points from a CPL or a PL that

In: **CPL**: CPL or PL **dist**: minimal distance between original points **cloud**: if true; All points are considered as point cloud not as CPL

Out: **PL**: CPL with minimal distances or PL in case of a cloud

CPLremFFT

removes points from a CPL by removing high frequencies

In: **CPLin**: Close point list **percent**: Percentage <0..100> of the number of points that remain

Out: **CPLout**: CPL with reduced number of points

CPLremovearea

removes CPLs of small area using the polyshape class

In: **CPL**: Original CPL **sline**: minimal length (not area) **remregions**: if true, regions are removed

Out: **CPLN**: CPL without removed areas

CPLremovebuffer

just removes thin walls from CPLS (same as CPLdulledges)

In: **CPL**: CPL conotur **wall**: minimal wall thickness

Out: **CPLN**: New Contour without CPLwall

CPLremovecollinearpoints

removes 2D/3D collinear point with different results for opened and closed CPL/CVL

In: **CPL**: 2D/3D open or closed CPL; Behavior differers **ADmin**: [Amin Dmin] [Minimal area cross product, minimal Distance]; default is 1e-7 1e-7

Out: **NCPL**: New PL/CPL without collinear points

CPLremoveoverlap

For each contour of CPL the polyshape is simplified

In: **CPL**: Original CPL

Out: **CPL**: CPL without overlapping areas

CPLremoveregionsholes

removes CPLs of small areas and small holes using the polyshape class

In: **CPL**: Original CPL **sline**: minimal length (not area)

Out: **CPLN**: CPL without removed areas

CPLremovethinwall

just removes thin walls from CPLS same as CPLdulledges and CPLremovebuffer

In: **CPL**: CPL conotur **wall**: minimal wall thickness; default is 1.2

Out: **CPLN**: New Contour without thin walls

CPLremstraightAmin

removes points with a minal distance/area to a predecessor or successor

In: **CPL**: Closed Polygon Line in 2D or 3D **Amin**: Minimal Area between two succeeding points; default is 1e-3;

Out: **NPL**: New Vertex List

CPLrepair

returns a CPL for extrusion - finding, duplicating and separating split points (required for SGLib 5.4)

In: **OCPL**: Original Contour line **dd**: resolution for shifted points; default is 1e-6;

Out: **CPL**: Closed **NP**: Split point coordinates; not acceicbal

CPLRofCPL

uses the nan separated point list to create contours using PLgrowline

In: **CPL**: nan separated point list to be grown to a contour **R**: Radius for growing contours from lines **"melt"**: if used link contours are melted if they have 2 or more common crossing points

Out: **CPLR**: cell array with contours for each link

CPLselectbynearestP

returns an index list for selectNaN to select by the nearest Points/CPLs

In: **CPL**: CPL **PL**: Single point to check **aux**: density of auxiliary points; default is 0.1

Out: **ind**: index list **ddd**: distance to nearest point

CPLselectbyP

returns an index list to select enclosing CPLs

In: **CPL**: CPL **PL**: Single point to check

Out: **ind**: index list **CPL**: the input CPL

CPLselectinout

returns selected CPLs based on their enclosure index

In: **CPL**: CPL **si**: 0 = outside! surrounded by 1, 2 = surrounded by 2 etc. **in1st**: true = insode start, false = outside start

Out: **CPLS**: Selected CPLs **fi**: index list of the selected contours in CPL **ci**: enclosure/nesting index per contour from CPLsortinout

CPLselectregionbyP

select a polyshape region (given als CPL) by an interior point

In: **CPL**: CPL in 2D **P**: Point in 2D

Out: **CPLN**: CPL of selected polyshape region

CPLselectspatial

selects parts of CPL based on spatial or other relations

In: **CPL**: Closed Polygon List **rel**: spatial or other relation **do**: if used objects instead of CPLs are used in general **"object"**: if used, whole objects are selected/operated on instead of individual CPLs

Out: **CPLA**: Selected CPL

CPLsetallcw

sets all contours of a CPL to cw

In: **CPL**: CPL

Out: **CPL**: CPL with all polygons in cw

CPLsimplify

simplifies a CPL using polyshape simplyfy true

In: **CPL**: CPL

Out: **CPLN**: 'repaired' or 'simplified' CPL with keep colinear points

CPLskeleton

returns a skeleton with equal distances to all wall sizes

In: **CPL**: Contour **nn**: number of grid points; default is 100 **ri**: Amin for point removeal; default is -1

Out: **PLA**: Point list of Skeleton **ELA**: Edge list of Skeleton

CPLskeletonsearch

creates a point list and an edge list

In: **CPL**: Closed contour line **CPLorg**: original full contour used for collision-free path planning; default is CPL

Out: **PL**: Point list **EL**: Edge list

CPLsocketscrewed

Contour for an anti-rotation bushing

In: **diam**: [max min #] ; default is [10 8.5 2]

Out: **CPL**: Kontour

CPLsplitui

splits a CPL interactively into two CPLs

In: **CPLA**: Original Contour

Out: **CPLA**: contour left after removal (BLUE) **CPLB**: removed contour (RED) **CPLC**: contour used to cut (DASHED)

CPLspringSpiral

creates just the pathe and basic contour for a windup motor spring spiral

In: **RP**: [radius distance]; default is [30 0]. Outer radius of the spring; In case of D(1)>1 this is the distance to the axis **d**: Thickness of the spring contour; default is 1.2 **D**: Diameter and circular angle for the end; default is [0 1.7] **CPLC**: Optinal middle contour to be integrated for shaft **cc**: [Outer-spring-rounds Inner=Spring-Round; default is [2 3]

Out: **CPLN**: Contour of the spring for SGofCPLz **PLN**: Point list for the path

CPLsquareline

creates a sequence of boxes of different x-dimensions

In: **x**: vector of x-dimension of squares **y**: scalar y-dimension of the squares **d**: distance between the squares

Out: **CPL**: Closed Polygon Line **cp**: center points of the boxes. **d**: dimensions of boxes that

CPLstent

creates a CPL pattern for a laser cutted stent

In: **xx**: list with x-sizes of an odd number of stent openings; such as [5 5 5 5 5] **hy**: y-size of the stent openings **dxy**: material thickness of the stent pattern **ny**: odd number of repeating lines such as 1, 3, 5, etc. **cut**: if true the last repeated pattern its cut; default is true;

Out: **CPL**: Pattern of the Stent that can be used in SGbendedCPL **cp**: x coordinates of the first hole line

CPLstentline

creates a sequence of boxes of different x-dimensions

In: **x**: vector of x-dimension of squares **y**: scalar y-dimension of the squares **d**: distance between the squares

Out: **CPL1**: CPL of the original x-dimension boxes separated by distance d for **CPL2**: CPL of the inbetween fitting boxes separated by distance d **CPLX**: CPL of the cutting pattern open **sx**: x positions of the squares along the line

CPLsweep

returns a swept contour

In: **CPL**: Closed polygon line **wt**: angle w or [wmin wmax]; default is p/20 **cp**: center point for rotation; default is 0 **gw**: increase decrease contour; default is 0

Out: **CPLN**: Final contour

CPLsweep_simple

1st sweep fnct

In: **CPL**: closed contour (CPL) to sweep **w**: sweep angle in radians; default is pi/4 **cp**: center point [x y] of the sweep rotation; default is [0 0]

Out: **CPLN**: swept closed contour (union of the input contour rotated across the sweep angle)

CPLtextbanner

Returs a CPL for an size adjusted Textbanner or Contour

In: **txt**: Text or CPL to enclose **sz**: [width height border]; default is [120 25 4]; **r**: optional radius; default is automatic border/2 **rem**: if true; the contours that would fall out are removed; default

Out: **CPL**: of Text Banner **CPLT**: CPL of Text contour **CPLB**: CPL of Box Contour

CPLui

interactive fnct to create and modify CPLS (Only one Contour)

In: **CPL**: Optional CPL; default is []; **col**: Optional color; default is 'r-' **linw**: optional line width; default is 2

Out: **CPL**: Final CPL

CPLunionpatternPL

creates copied and united contours

In: **CPL**: CPL of contour **PL**: point list to copy

Out: **CPLN**: united contours - does not have one for each point of PL

CPLunionreplace

countour union after a subtraction of a bounding box or convex hull

In: **CPLA**: Contour A **CPLB**: Contour B **relparam**: relative parameter to adjust position **"type"**: bb, convexhull, delaunay, fillgap **"ol"**: optional overl; default is 0

Out: **CPL**: Final contour

CPLunionsorted

For 2.5D Design in Mechanical Engineering, there is a different CPLunion required

In: **A**: Contour A describing crossections **B**: Contour B describing crossections **sr**: spatial relation like for CPLtransrelCPL

Out: **CPLU**: Final Contour with separated union and subtract

CPLunsort

changes the order of nan separated CPL using unsortNaN

In: **CPL**: Original CPL

Out: **CPLU**: THe order of Conturs has changed

CPSdiffarea

return the norm difference between two polyshape areas by xor

In: **CPSA**: Closed Polygon or Polyshape A **CPSB**: Closed Polygon or Polyshape B

Out: **a**: resulting area

CPSlineui

interactive fnct to create and modify CPS lines (Only one Contour)

In: **CPS**: Optional polyshape **col**: Optional color; default is 'r' **linw**: optional line width; default is 2 **edge**: true = round, false = corner; default is false **caps**: true = round, false = corner; default is false

Out: **CPS**: Final CPS

CPSofgca

returns a struct called Colored PolyShape (CPS) similar to a polygon array object

In: **h**: optional handle to implement CPSofpolygon

Out: **CPS**: struct with ps, fc, fa, ec

CPSofgcalines

converts all or selected lines graphcis objects into CPS structs using a line thickness

In: **h**: handle or gca **si**: optional line thickness in mm such as 0.35, 0.5, .7mm

Out: **CPS**: CPS struct with an array of ps

CPSofpolygon

returns a struct called Colored PolyShape (CPS) similar to a polygon array object

In: **h**: handle to polygon

Out: **CPS**: struct with ps, fc, fa, ec

CPSouterboundary

returns the outer boundary as CPS or polyshape

In: **CPS**: CPS or polyshape

Out: **CPSN**: CPS or polyshape of outer boundary

CPSplot

plots a colored polygon from a CPS struct, a polyshape or a CPL (NO SIMPLIFICATION)

In: **CPS**: CPS or ps **fc**: optional FaceColor **fa**: optional FaceAlpha **ec**: optional EdgeColor **lw**: optional Edge Line Width

Out: **h**: handle to polygon graphics object

CPSremovearea

removes CPS/polyshape regions of small area

In: **CPS**: Original CPS/polyhsape **sline**: minimal length (not area)

Out: **CPSN**: remaining polyshapes **CPRA**: Removed polyshapes

CPSsimilaritymap

converts an array of CPS or polyshapes into a cell list of similar CPS

In: **CPS**: CPS struct or array of polyshape array **tol**: tolerance for turningdist; default is 1e-7 ~ eps2

Out: **S**: cell list of similar boundaries n x ..... **pso**: array of different shape: n elements **T2**: List of 3x3 transformation matrix related to ps **psc**: array of different shape n elements **ps**: full array of boundary contour

CPSui

interactive fnct to create and modify CPLS (Only one Contour)

In: **CPS**: Optional polyshape **col**: Optional color; default is 'r' **linw**: optional line width; default is 2

Out: **CPS**: Final CPS

ELofPL

creates an EL for a PL of succeeding points

In: **PL**: Point List **op**: if true; the Contour is open; default is false

Out: **EL**: Edge List

GPLauxgridpointsCPS2

creates a defined number of Gridpoints

In: **CPLR**: Contour **nr**: number of points to fill in; **up**:

Out: **GPL2**: Grid Point list, size is smaller or equal as nr **d2**: used distance **GPL1**: smallest gridpoint list with more than nr points

isCVP

returns whether a sequence of vertices is a contour vertex path

In: **VL**: Vertex list, that should represent a contour path in 3D

Out: **x**: true if all conditions are fullfilled

openedCPL

opens all contours in CPL(2D) and CVL(3D) without any other change beside removal of all consecutive identical points.

In: **CPL**: Contour 2D or 3D with NaN embedded contours **lim**: default is 1e-12 for removal of succeding points

Out: **NCPL**: New contours 2D/3D with all open contours without any consecutive identical point

openedCVL

closes all contours in CPL(2D) and CVL(3D) after removal of all consecutive identical points.

In: **CVL**: Contour 2D or 3D with NaN embedded contours **lim**: default is 1e-12 for removal of succeding points

Out: **NCVL**: New contours 2D/3D with all open contours without any consecutive identical point

PLcirctrapezoid

creates a trapezoid contour with a predefied angle

In: **bh**: [diameter/width and height]; default is [4 0] **wz**: rotation of CPL; default is 0 **al**: default is 51 degree; used to calculate the height

Out: **PL**: PL to replace a circular solid using

PLconnectopenradial

closes an open line by a contour

In: **PL**: single open line **R**: Radius

Out: **PLA**: Closed contour without doubled entries **PLN**: Externsion may contain start or end point

PLdecompose

returns a list of contour segments that belong to the same angle segment

In: **CPL**: SINGLE CONTOUR **da**: angle; default si pi/6 **mix**: if true; minxy is used to start with segment 1; default is false;

Out: **SIL**: Separation Index List

PLdiamond

returns a diamond shape optonal als convex hull for a CPL

In: **dim**: [width height] or CPL **al**: angle, currently unsused, i.e. 45 deg

Out: **PL**: CPL of a diamond in dim or that is a hull for the CPL

PLgrowline

converts a single open polygon into a grown contour

In: **PL**: Open point list line **s**: thickness **edge**: true= rounded edges; false = simple; default false; **caps**: true= rounded caps; false = straight; default is false;

Out: **CPL**: Closed Polygon line

PLofCL

returns an arbitrary planar point list to vizualize the correspondance list

In: **CL**: Correspondance list **circ**: if true, the point lists are arranged on circles instead of parallel lines; default is false

Out: **PL1**: Point list belongst to 1st col of CL **PL2**: Point list belongst to 2nd col of CL

plotcorrelation

plots a correlation for debugging

In: **CL**: Corrleation list (or FL) **w**: line width of the correlation lines; default is 1

Out: **h**: handle to the figure **PL1**: point list belonging to the 1st column of CL **PL2**: point list belonging to the 2nd column of CL

PLshortestpathinCPL

returns a path inside of a CPL using A* Zelinzky, Lozano-Perez, Dikstra

In: **CPL**: CPL **A**: Starting Point A inside the CPL **B**: End Point B inside the CPL **shr**: shrinkage value; default is sofBB/n **n**: number of grid points; default is 100 **smo**: 'A*', 'Bezier', 'fft' for smoothening

Out: **PLB**: Bezier smoothed path **PL**: Grid bases path **k**: grid numbers **shr**: dynamic shrinkage value

PLui

interactive fnct to create and modify CPLS (Only one Contour)

In: **PL**: Optional PL; default is []; **col**: Optional color; default is 'r-' **linw**: optional line width; default is 2

Out: **PL**: Final Point List

PSanalyze

classifies the polyshapes regions outer boundaries into equal shapes

In: **ps**: Polyshape

Out: **S**: struct containing each region

PSofgca

return the polyshapes of the polygons of current axis

In: **unchanged**: false = separated as cells; true = only on polyspahe

Out: **ps**: polyshape of current axis **CPL**: closed polygon line of ps

PSregions

returns the regions of a polyshape

In: **ps**: Input polyshape or colored polyshape struct

Out: **PSR**: Array of polyshapes, one per region

R12ofCPL

return standardized link and joint dimensions

In: **CPL**: List of Points for A0, B0, B1, A1

Out: **R12o**: [Outer and Inner radius] for links

RLofCVL

Returns Radii of a contour, same as CVLdimclassifier

In: **CVL**: 3D path in space **res**: default is 0.1

Out: **RL**: [cx cy cz Nr deg R vx vy vz] **RMM**: [rmin rmax]

SGofCPLflat

returns a SG struct of an open flat surface from a polyshape or closed polygon

In: **CPL**: Closed Polygon or Polyshape **T**: vector, rotation matrix, or HT matrix to locate the surface in space

Out: **SG**: SG surface struct (open flat surface, not watertight)

TofCVL

returns for curved contours the mean transformation matrix

In: **CVL**: CVL - curved - for planar use TofVL **ez**: if not empty ez of result T is forced into direction of ez

Out: **T**: Transformation matrix

TofPL

returns a 2D HT matrix for a point list

In: **PL**: Point list

Out: **T**: 3x3 HT matrix **ew**: eigenvalues of the point distribution, scaled and sorted [major minor]

VLNLOLplot

plot an open or closed CVL including Edge Normal Vector and Edge Orthogonal Vector

In: **VL**: MAGENTA Path, required **NL**: RED EDGE Normal Vectors **OL**: Blue = Edge Orthogonal Vector; RED=last, Green= first

Out: **h**: handle to graphic

VLpathdist

distance between two points on a CVP PAth

In: **VLD**: VLdistance(CVL) or CVL **s**: start index **el**: end index; if more than 1 index, the shortes forward and the shortest backward is selected

Out: **lll**: forward distance-backward-distance overall-length]; two rows in numel(k)>1

3.3 Planar 3D Polygons (VL) [guide]

3.4 Spatial 3D Polygons (CVL) [guide]

Functions in this section:

CVL2periodic — converts a perodic spatial contour into three periodic discrete fnct

CVLaddauxpoints — inserts auxiliary points and removes near points in CVLs

CVLclipT — returns a clipped CVL relative to a HT matrix

CVLedgecenter — Calculates the center/middle point between two edge points

CVLedgeNormal — Calculates normal vectors for curved closed contours

CVLfreeBoundaryVLFL — returns the CVL of the free boundary

CVLinsertT — inserts the x/y crossing points of a CVL and a HT matrix of a plane

CVLof2CPLzcorrelate — returns a CVL list for a correlated fusion of 2 CPL over a distance of z

CVLofconvexHull — returns the three cutting contours through the convex hull of a solid

CVLofCPL — returns a CVL from a CPL

CVLofCPLbendsinus — returns a wave-shape-bended CVL from a CPL

CVLofCPLbendspherical — bends a CPL on a cylinder or spherical ellipsoid

CVLofopenfaces — returns the closed vertex list for all open surfaces

CVLofRL — returns a CVL of a radial list and an ez vector

CVLofSGcutplanes — returns the three cutting contours at a frame coordinate system through a solid

CVLofSGcutT — returns the CVL of a crossing/slicing plane given by a HT or Frame string

CVLofSGcutTrot — creates contours and 2.5D solids from a rotated cross-slice of a solid.

CVLofSGfi — returns from a (interactively selected) facet index the Contour and Transformation matric

CVLofSGfreeboundary — returns the closed vertex list for all open surfaces

CVLofSGT — returns the CVL of the surface that belongs to a Frame

CVLofSGTFS — returns from a (interactively selected) facet index the Contour and Transformation matric

CVLofVLEL2 — processes a format of

CVLpointdistancetoedge — returns the edges that have the minimal distance to a point (Works in 3D)

CVLsample — Returns a sample vertex list for testing algorithms

CVLsortlinestraight — returns a reordered CVL

CVLzsweep — returns a swept CVL z Contour and a 2.5d solid

Detailed description

CVL2periodic

converts a perodic spatial contour into three periodic discrete fnct

In: **CVL**: CPL or PL; only the first are used **tp**: time; default is 2* pi

Out: **CPLx**: CPL [t x] **CPLy**: CPL [t y] **CPLz**: CPL [t z] **t**: time vector

CVLaddauxpoints

inserts auxiliary points and removes near points in CVLs

In: **CVL**: Original CVL **a**: auxiliary point maximum distance; default is 1 **e**: minimal distance to remove points; default is -1;

Out: **CVLN**: New CVL

CVLclipT

returns a clipped CVL relative to a HT matrix

In: **CVL**: Closed Polyggon in 3D **T**: Frame matrix **ax**: clipping area; [1 0 0]; [-1 0 0]; [0 1 0]; [0 -1 0]

Out: **CVLC**: Clipped contour

CVLedgecenter

Calculates the center/middle point between two edge points

In: **CVL**: contour vertex list

Out: **MVL**: Middle-point-of-edge vertex list

CVLedgeNormal

Calculates normal vectors for curved closed contours

In: **CVL**: CVL **Amin**: minimal area threshold passed to VLedgeNormal; default is 0.001

Out: **NL**: Normal vector list for each vertex **OL**: Orthogonal inplane vector list for each vertex

CVLfreeBoundaryVLFL

returns the CVL of the free boundary

In: **VL**: Vertex List, or Solid Geometry, or triangulation **FL**: Facet list, if VL is Vertex list

Out: **CVL**: Closed vertex contour list **TR2**: triangulation

CVLinsertT

inserts the x/y crossing points of a CVL and a HT matrix of a plane

In: **CVL**: CVL **T**: T matrix

Out: **NCVL**: NCVL matrix with crosing points with the xy plane

CVLof2CPLzcorrelate

returns a CVL list for a correlated fusion of 2 CPL over a distance of z

In: **CPLA**: Contour A in 2D **CPLB**: Contour B in 2D **z**: [h n] for [height, interlevels] or [0 z1 z2 ... zn]

Out: **CVLz**: CVLz with a **dz**: list of z values **MCA**: Deleted contour in CPLA **MCB**: Deleted contour in CPLB **CPRA**: Missing contour in CPLA **CPRB**: Missing contour in CPLB

CVLofconvexHull

returns the three cutting contours through the convex hull of a solid

In: **SG**: Solid geometry **T**: cutting coordinate system; default is eye(4) at center of solid

Out: **CVLX**: cutting contour x=0 (green) **CVLY**: cutting contour y=0 (blue) **CVLZ**: cutting contour z=0 (red) **T**: used T matrix; if input parameter T was empty

CVLofCPL

returns a CVL from a CPL

In: **CPL**: Contour **T**: direction such as 'x' or HT matrix **d**: distance from origin; default is 0

Out: **CVL**: CVL

CVLofCPLbendsinus

returns a wave-shape-bended CVL from a CPL

In: **CPL**: CPL **dim**: wave dimension; default is 'x' **hwv**: number of sin waves; default is 1 **scl**: default is sofCPL/10

Out: **CVL**: Final CVL

CVLofCPLbendspherical

bends a CPL on a cylinder or spherical ellipsoid

In: **CPL**: Point List **Rx**: Radius Rx **de**: distance/intrusion; default is -0.1 **Ry**: Optional radius Ry; default is 10000 **Rz**: Optional radius Rz; default is Rx **cent**: center solid before bending; default is true

Out: **CVL**: Vertex list of the bended point list

CVLofopenfaces

returns the closed vertex list for all open surfaces

In: **VL**: Vertex list **FL**: Facet list **thr**: threshold; default is 1e-3 for VLFLcorrectdoubledvertex

Out: **CVL**: Closed Vertex List **EL**: Edge List **CIL**: Contour Index List for EL

CVLofRL

returns a CVL of a radial list and an ez vector

In: **RL**: Radial list **ez**: optional ez vector; or ez vector list list

Out: **CVL**: Closed Vertex List

CVLofSGcutplanes

returns the three cutting contours at a frame coordinate system through a solid

In: **SG**: Solid geometry **T**: cutting coordinate system; must be inside of the solid!

Out: **CVLX**: cutting contour x=0 (green) **CVLY**: cutting contour y=0 (blue) **CVLZ**: cutting contour z=0 (red) **SG2**: used T matrix; if input parameter T was empty

CVLofSGcutT

returns the CVL of a crossing/slicing plane given by a HT or Frame string

In: **SG**: Solid Geometry with Frames **T**: 4x4 matrix or Frame name

Out: **CVL**: Convtour Vertex List **CPL**: Closed contour wrt Frame N; [n x 3]; z should be 0

CVLofSGcutTrot

creates contours and 2.5D solids from a rotated cross-slice of a solid.

In: **SG**: Solid Geometry **T**: Cutting Frame; default is eye(4) **wlim**: [wlim wmax]; default is [-pi/4 +pi/4]; **grow**: growing factor / slot size

Out: **CVLA**: Spatial contour [n x 3] of above line **CVLB**: Spatial contour [n x 3] of below line **CPLA**: Planar contour [n x 2] of above line **CPLB**: Planar contour [n x 2] of below line **A**: Solid for contour A **B**: Solid for contour B

CVLofSGfi

returns from a (interactively selected) facet index the Contour and Transformation matric

In: **SG**: Solid Geometry **fi**: optional facet index; default is '' => Interactive mode **sw**: contour selection default is [0 10]

Out: **CVL**: Closed Contour in Space **T**: Transformation Matrix (at position of fi) to create CVL from CPL (is ona surface) **CPL**: Planar Contour **cp**: center points related to CPL **fi**: facet index if selected interactively

CVLofSGfreeboundary

returns the closed vertex list for all open surfaces

In: **SG**: Solid Geometry **FL**: facet list; if given, first input is treated as the vertex list

Out: **CVL**: Closed Vertex List **EL**: Edge List **CIL**: Contour Index List for EL

CVLofSGT

returns the CVL of the surface that belongs to a Frame

In: **SGN**: Solid Geometry with Frames **N**: Frame name or Transfomraton Matrx [4x4]

Out: **CVL**: Convtour Vertex List **CPL**: Closed contour wrt Frame N; [n x 3]; z should be 0

CVLofSGTFS

returns from a (interactively selected) facet index the Contour and Transformation matric

In: **SG**: Solid Geometry **T**: optional facet index; default is '' => Interactive mode **sw**: contour selection default is [0 10] **alp**: alpha angle foe feature surface default is 0 **thr**: threshold to detect in plan points; default is 1e-3

Out: **CVL**: Closed Contour in Space **fs**: feature surface index fi that belongs to CVL **CPL**: Planar Contour **cp**: center points related to CPL **ii**: facet index if selected interactively

CVLofVLEL2

processes a format of

In: **VLA**: VLA or [p2 p2; p2 p3; ...] **ELA**: ELA if VLA is used

Out: **CVL**: CVL

CVLpointdistancetoedge

returns the edges that have the minimal distance to a point (Works in 3D)

In: **CVL**: Vertex list separated by nans **p**: point n x 3 to check **filter**: if true only the nearest contour is returned; default is true

Out: **ea**: [contourindex, startpointidex, distance to line/edge, minimaldistanceindex, distance to start/or end point]

CVLsample

Returns a sample vertex list for testing algorithms

Out: **VL**: sample solid vertex list **PL**: point list

CVLsortlinestraight

returns a reordered CVL

In: **XVL**: Original Line in 3D Space

Out: **NVL**: Corrected Line in 3D Space

CVLzsweep

returns a swept CVL z Contour and a 2.5d solid

In: **CVL**: CVL sliced Contour

Out: **CVLn**: Swept CVLz **SG**: 2.5D solid geometry built from the swept slices

3.5 General PL functions [guide]

3.5.1 PL modification [guide]

Functions in this section:

PLconvexhull — returns the convex hull of a 2D point [nx2] list

Detailed description

PLconvexhull

returns the convex hull of a 2D point [nx2] list

In: **PL**: Single Contour Point List

Out: **PHL**: Convex hull point list

3.5.2 Crossing Points [guide]

Functions in this section:

PLsplitpointsofCPL — returns splitpoints created by delaunaytriangulation

Detailed description

PLsplitpointsofCPL

returns splitpoints created by delaunaytriangulation

In: **CPL**: CPL separated by NaN

Out: **SPL**: List of splitpoints created by delaunaytriangulation

3.5.3 CPL Boolean Operations [guide]

Functions in this section:

CPLadd — adds the outer contours and subtracts the holes of two CPLs

CPLbool — Boolean Fncts based on the polyshape class or polygon class

CPLboolpshape — Boolean Fncts based on the polyshape class

CPLintersect — returns a Boolean intersection of 2 CPLs

CPLmeltbool — Merges a list of CPL by boolean add and sub - Similar to CPLunitesorted but splits the contour too

CPLsplitbool — returns a CPL that is split into two or more parts by a second contour

CPLsubtract — returns a Boolean subtraction of 2 CPLs

CPLunion — returns a Boolean addition of 2 CPLs

CPLunitecwccw — returns a CPL that has no crossing lines anymore but sorted wrt cw/ccw and adding/subtracting

CPLunitesorted — returns sorted and added closed polygons

CPLxor — returns a Boolean exclusive or of 2 CPLs

Detailed description

CPLadd

adds the outer contours and subtracts the holes of two CPLs

In: **CPLA**: CPLA A **CPLB**: CPLA B **ibool**: '+' drills the holes through all, 'x' finds contour that fits trough all holes

Out: **CPL**: Final Contour

CPLbool

Boolean Fncts based on the polyshape class or polygon class

In: **flag**: Operator **CPLA**: Contour A **CPLB**: Contour B

Out: **[CPLN]**: Final CPL

CPLboolpshape

Boolean Fncts based on the polyshape class

In: **fflag**: Operator **CPLA**: Contour A **CPLB**: Contour B

Out: **CPLN**: Final CPL

CPLintersect

returns a Boolean intersection of 2 CPLs

In: **A**: CPL A **B**: CPL B

Out: **CPL**: Resulting Boolean

CPLmeltbool

Merges a list of CPL by boolean add and sub - Similar to CPLunitesorted but splits the contour too

In: **CPL**: CPL **fullstep**: if false, only two layers are processed; default is false

Out: **CPL1**: resulting contour

CPLsplitbool

returns a CPL that is split into two or more parts by a second contour

In: **CPLA**: CPL to be splitted **CPLB**: CPL that overlaps with CPLA

Out: **CPLsub**: CPLA minus CPLB (BLUE) **CPLxor**: XOR of CPLA and CPLC (RED) **CPLB**: CPLC used to cut; if interactive mode was used;

CPLsubtract

returns a Boolean subtraction of 2 CPLs

In: **A**: CPL A **B**: CPL B **rel**: list of spatial relations same of CPLtransrelCPL

Out: **CPL**: Resulting Boolean

CPLunion

returns a Boolean addition of 2 CPLs

In: **A**: CPL A **B**: CPL B **sr**: spatial relation for CPLtransrelCPL

Out: **CPL**: Resulting Boolean

CPLunitecwccw

returns a CPL that has no crossing lines anymore but sorted wrt cw/ccw and adding/subtracting

In: **CPL**: CPL including crossing lines **ord**: true if CPLorder is used first; default is false

Out: **CPLA**: Processed CPL

CPLunitesorted

returns sorted and added closed polygons

In: **CPL**: Closed Polygon Line

Out: **CPLN**: systematically ordered and Boolean operated CPL

CPLxor

returns a Boolean exclusive or of 2 CPLs

In: **A**: CPL A **B**: CPL B

Out: **CPL**: Resulting Boolean

3.5.4 PL Comparison [guide]

Functions in this section:

PLcorrelate — returns the correlation of two single point lists

Detailed description

PLcorrelate

returns the correlation of two single point lists

In: **PLA**: Point list A **PLB**: Point list B **stype**: 'number','length','angle','center','nearest', default 'center' **ctype**: 'org','minx','mina'/'rot'; default is 'mina' **mtype**: 'none', 'norm', 'norm1' **zz**: optional z values for the vertex list; scalar sets z of contour B, 2-element sets z of both contours; default is empty

Out: **CL**: Correlation List **VL**: Vertex list [PLA;PLAB] without doubled points

3.6 CPL to Surfaces [guide]

Functions in this section:

FLcontourwallELn — returns the wall facets between 2 contours given by an edge list

Detailed description

FLcontourwallELn

returns the wall facets between 2 contours given by an edge list

In: **EL**: Edge list for both contours **n**: number of points/vertices, if not n=max(EL) or not size(PL,1)

Out: **FL**: upright walls between the 2 contours

3.7 CVL to Solid Geometries [guide]

Functions in this section:

VLFLofCVLdelaunay2D — returns a triangulation for an almost planar CVL

VLFLofCVLdelaunay3D — returns the top and below tessellation for ONE closed vertex list which cuts a convex solid into two parts

VLFLofVLELdelaunay3D — returns the top and below tessellation for ONE closed vertex list which cuts a convex solid into two parts

Detailed description

VLFLofCVLdelaunay2D

returns a triangulation for an almost planar CVL

In: **CVL**: 3D closed contour vertex list **aux**: additional points with optional grid distance > 0

Out: **VL**: Vertex list **FL**: Facet List

VLFLofCVLdelaunay3D

returns the top and below tessellation for ONE closed vertex list which cuts a convex solid into two parts

In: **CVL**: Closed Polygon Vertex List

Out: **VL**: Vertex List **FLA**: Facet List Above/Front of EL (CYAN) **FLB**: Facet List Below/Back of EL (YELLOW)

VLFLofVLELdelaunay3D

returns the top and below tessellation for ONE closed vertex list which cuts a convex solid into two parts

In: **OVL**: Original Vertex List **EL**: Edge List

Out: **VL**: Vertex List **FLA**: Facet List Above/Front of EL (CYAN) **FLB**: Facet List Below/Back of EL (YELLOW)

4 Surfaces [guide]

4.1 2D Surfaces [guide]

Functions in this section:

PLELDelaunay — return the Delaunay Triangulation

PLFLofCPLdelaunay — Returns 2D point list and facet list of a triangulated contour

PLFLofCPLdelaunayGrid — returns a point list and facet list for a grid filled contour list

PLFLofCPLpoly — Utilizes the poly2fv fnct for VLFL-LIb

PLFLselect — returns only the required points for a given edge list. The edge list is reindexed

TofVLFL — creates a T vector for surface, ez is the mean value of the normal vectors of the facets, considering the orientation of the outer border, c is the center of the convex hull

TR2ofimage — returns a Delaunay triangulation for a grayscale image

TRofCPL — just returns a 2D triangulation

viVLFLnearestneighborVL — returns the vertex indices of the nearest neighbors of a Vertex list

VLFLaddfacetcenterpoints — creates additional vertices in the middle of each facet

VLFLcircFLedges — modifies the Facet list by circulating rows to minimize the distance P2-P3.

VLFLcolcat2 — Same as VLFLcat2 but including colors

VLFLdeleteFLofvi — removes the faces that belongs to a specific vertex

VLFLfaceneighborhood — return the neighbors and the neighbor neighbors

VLFLfaceneighborswap — shows what happens if a triangle-pair is swapped

VLFLfreeboundarycircshift — changes Vertices or Facets to achieve a free boundary rotation

VLFLofCPL — returns the vertex and facet list from a closed contour list

VLFLofSG — returns one vertex list and one faces list for nested SG

VLFLofSurf — returns a vertex and facet list from a surf handle

VLFLoppositefaces — returns the list of faces that have opposit facets in the same solid

VLFLoptimizeFLWez — finds the surfaces correlation that creates the largest volume

VLFLregions — fnct to separate regions in face index list

VLFLremoveFacet — removes a single facet from a surface

VLFLremoveIntersectFacets — removes intersecting facets from a VLFL surface

VLFLremovekinks — removes kinks after SGofCVLz

VLFLremsmallfacets — removes small facets from an open surface

VLFLsmoothboundary — creates facets to smoothen or closing open boundaries without adding points

VLFLspherecurvedPLEL — returns VL and FL for a sphere-curved surface given by a point list and contour edge list

VLisInteriorofBBlist — returns for a list of Bounding Boxes and a single vertex the BB that contain the vertex

VLofSGnotreferenced — returns the logical index of vertices that are not referenced in a triangulation

VLofVLFLselfIntersection — returns the self intersection crossing points of a Surface

VMdistancetoSG — Creates a Voxel model with distance values from the surface

Detailed description

PLELDelaunay

return the Delaunay Triangulation

In: **PL**: Point list **EL**: Edge list

Out: **PLN**: Point list could have changed **ELN**: edge list could have changed **FLall**: Delaunay triangulation **in**: inside/outside status **FLin**: FLall(in,:) **FLout**: FLall(~in,:) **warn**: last warn result

PLFLofCPLdelaunay

Returns 2D point list and facet list of a triangulated contour

In: **CPL**: Contour point list (2D) of the enclosed surface **ADDVL**: Vertex list of additional points to include

Out: **XPL**: Point List (2D) of the **XFL**: Facet List of the enclosed surface **XEL**: edge list of the free boundary of the triangulation

PLFLofCPLdelaunayGrid

returns a point list and facet list for a grid filled contour list

In: **CPL**: Closed Polygon list **ds**: delta on contour; default is 4 **dx**: delta in x; default is ds **dy**: delta in y; default is dx

Out: **PL**: Resulting Point list **FL**: Resulting Facet list **EL**: Resulting Edge List

PLFLofCPLpoly

Utilizes the poly2fv fnct for VLFL-LIb

In: **CPL**: Contour points list

Out: **PL**: Point list **FL**: Facet list

PLFLselect

returns only the required points for a given edge list. The edge list is reindexed

In: **PL**: Original point list **FLG**: Original facet list

Out: **PL**: Shorter point list **SI**: Sorting index **FL**: Reindexed facet list

TofVLFL

creates a T vector for surface, ez is the mean value of the normal vectors of the facets, considering the orientation of the outer border, c is the center of the convex hull

In: **VL**: Vertex List **FL**: Facet List **fi**: optional facet index list

Out: **T**: HT Matrix of the surface **ez**: ez vector of the surface

TR2ofimage

returns a Delaunay triangulation for a grayscale image

In: **I**: Image nxm **t**: type of triangulation (default is 1)

Out: **[TR2]**: delaunay triangulation

TRofCPL

just returns a 2D triangulation

In: **CPL**: CLosed Polygon list

Out: **TR2**: 2D Triangulation

viVLFLnearestneighborVL

returns the vertex indices of the nearest neighbors of a Vertex list

In: **VLA**: Vertex List of Solid **FLA**: Facet List of Solid **VL**: Vertex list to check

Out: **vi**: neares neighbor index to VLA

VLFLaddfacetcenterpoints

creates additional vertices in the middle of each facet

In: **VL**: vertex list **FL**: facet list **nn**: recursive number of calls; default is 1 **md**: minimum distance between old and new points; default is 0

Out: **NVL**: New vertex list **NFL**: New facet list **NNVL**: List of the newly added facet-center vertices

VLFLcircFLedges

modifies the Facet list by circulating rows to minimize the distance P2-P3.

In: **VL**: Vertex list **FL**: Facet list **sortw**: flag to sort by angle instead of edge length; default is false **limit**: angle limit for reporting acute triangles; default is 0.01

Out: **FL**: Facet list with circulated rows to minimize distance |P2P3| **sina**: resulting cos alpha of angle at P1 **D**: matrix of the three edge lengths per facet

VLFLcolcat2

Same as VLFLcat2 but including colors

In: **V1**: Vertex list 1 **F1**: Face list 1 **C1**: Face color list 1 **V2**: Vertex list 2 **F2**: Face list 2 **C2**: Face color list 3

Out: **VL**: Vertex list VL=[V1;V2]; **FL**: Face list FL=[F1;F2+size(V1,1)] **FC**: Face colors FC=[C1;C2]

VLFLdeleteFLofvi

removes the faces that belongs to a specific vertex

In: **VL**: Vertex list **FL**: Facet list **vi**: index of vertices to remove frome the facet list

Out: **FLN**: New facet list **EL**: Open edge list **CIL**: CIL of open Edge list **CVL**: CVL of open edge list **ELu**: unconnectable open edge list after vertex removal

VLFLfaceneighborhood

return the neighbors and the neighbor neighbors

In: **VL**: Vertex list **FL**: Facet list **fi**: face index **n**: shells of neighbors

Out: **fil**: cell list of neighborhood

VLFLfaceneighborswap

shows what happens if a triangle-pair is swapped

In: **VL**: Vertex list **FL**: Facet list **fi**: face index **k**: number of neighbor (1..3) **NN**: neighbor list of the triangulation; if empty it is computed from VL/FL

VLFLfreeboundarycircshift

changes Vertices or Facets to achieve a free boundary rotation

In: **VL**: Vertex List **FL**: Facet list with exact TWO open boundaries **k**: shift number; default is 1

Out: **VLN**: New Vertex list wrt to old facet list **nE**: number of changes **FLN**: New Facet list wrt to old vertex list

VLFLofCPL

returns the vertex and facet list from a closed contour list

In: **CPL**: CPL

Out: **VL**: Vertex list **FL**: Facet list **TR**: delaunay triangulation object of the contour

VLFLofSG

returns one vertex list and one faces list for nested SG

In: **SG**: Solid geometry potentially nested

Out: **VL**: Vertex list **FL**: Facet list **FC**: Face color list

VLFLofSurf

returns a vertex and facet list from a surf handle

In: **s**: surface handle

Out: **VL**: Vertex list **FL**: Facet list

VLFLoppositefaces

returns the list of faces that have opposit facets in the same solid

In: **VL**: Vertex list **FL**: Facet list **dmax**: maximal distance **slec**: '&' or '|' '&' forces the faces to hit a surface of the same set; default is '&'

Out: **ofi**: logical index of facets **DL**: [si dist ei cp] **SGB**: Solid Geometry of the Bounding Box **FCL**: Facet center list **FNL**: Facet Normal list

VLFLoptimizeFLWez

finds the surfaces correlation that creates the largest volume

In: **VL**: Vertex List **FL**: Facet list **ez**: normal vector defining the alignment direction; if empty no reorientation is applied

Out: **FLN**: New Facet list wrt to original Vertex list **VLN**: New Vertex list wrt to original Facet list

VLFLregions

fnct to separate regions in face index list

In: **VL**: Vertex list **FL**: Facet list **FIL**: ONE facet index list; default is 1:size(FL,1)

Out: **fil**: cell list for separated regions

VLFLremoveFacet

removes a single facet from a surface

In: **VL**: Vertex list **FL**: Facet list **fi**: face index to be removed

Out: **FLN**: New Facet list

VLFLremoveIntersectFacets

removes intersecting facets from a VLFL surface

In: **VL**: Vertex list or SG **FL**: Facet List

Out: **NFL**: New Facet list **remi**: index of removed facets relative to FL **nii**: index of not intersecting facets relative to FL

VLFLremovekinks

removes kinks after SGofCVLz

In: **VL**: Vertexn list **FL**: Facet list **re**: recursion; default is true **ctyp**: error criterion for neighbor angles: 'min', 'max', 'mean', or 'diff'; default is 'min'

Out: **SGN**: New Sold **cnt**: number of changes

VLFLremsmallfacets

removes small facets from an open surface

In: **VL**: Vertex List **FL**: Facet List **Amin**: minimal Size

Out: **NVL**: New vertex list **NFL**: New facet list **rmvd**: number of removed facets

VLFLsmoothboundary

creates facets to smoothen or closing open boundaries without adding points

In: **VL**: Vertex list **FL**: Facet list **al**: closing if edge angle is > al; default is pi/4 **lr**: leg length maximum ratio; default is 1.5 **al2**: minimal angle value for closing; default is 0; **be**: required distance angle to parallel walls

Out: **FL**: List of old and appended new facets **NFL**: List of New facets

VLFLspherecurvedPLEL

returns VL and FL for a sphere-curved surface given by a point list and contour edge list

In: **PL**: Point list 2D **CEL**: Contour edge list **R**: Radius [-R 0 0] **d**: maximum point distance of contour and grid (sofrd)

Out: **VL**: Vertex list of surface **FL**: Facet list of surface **CEL**: Contour edge list

VLisInteriorofBBlist

returns for a list of Bounding Boxes and a single vertex the BB that contain the vertex

In: **BBL**: List of Bounding Boxes **VP**: Single Vertex

Out: **bi**: bounding box index for is on bounding box or inside bounding box

VLofSGnotreferenced

returns the logical index of vertices that are not referenced in a triangulation

In: **SG**: Solid Geometry

Out: **vi**: logical index of not referenced vertices

VLofVLFLselfIntersection

returns the self intersection crossing points of a Surface

In: **VL**: Vertex list **FL**: Facet list **sep**: surface separation {true,false}; default is false; **thr**: rounding threshold default is 1e-7

Out: **CVL**: Faces collision points **SG**: Cell list of Separated Geometry Surfaces **CVLi**: Cell list of faces collision points **CFi**: Cell list of the colliding facet indices

VMdistancetoSG

Creates a Voxel model with distance values from the surface

In: **SG**: Solid Geometry **d**: grid distance **invert**: if true; SG walls get maximal values instead of minimal; default is true **frame**: if true; A bounding box is used during calculation

Out: **VM**: Double Voxel Model for distance vectors **VMi**: Logical Voxel Model for iswall **xl**: vector of x grid coordinates of the voxel model **yl**: vector of y grid coordinates of the voxel model **zl**: vector of z grid coordinates of the voxel model

4.2 3D Surfaces [guide]

Functions in this section:

VLELofSGfeatureSurface — creates edge list, contour indexe list and feature surface index list of a solid

Detailed description

VLELofSGfeatureSurface

creates edge list, contour indexe list and feature surface index list of a solid

In: **SG**: Solid geometry

Out: **VL**: Vertex list **EL**: Edge list **CIL**: Contour index list **FSIL**: feature surface index list **SHIL**: Transformation matrix of the center of a surface **TT**: Transformation matrix of the center of a surface

4.3 Feature Surfaces [guide]

Functions in this section:

FEofSG — returns FE and optional VL of a solid geometry or of gca

FEofVLFL — returns FE and optional VL of a solid geometry or of gca

FLfeatureEdgeSurface — returns a surface facet list based on a given surface triangulation

FLfeatureEdgeSurface2 — returns a surface facet list based on a given surface triangulation

FLinsideSurfaceofSG — Returns the Facet Indices of inside surfaces

FSfeature — Selects features surfaces based on

FSofSG — Features surfaces of SG in contrast to selecting a FS FSselect and CPLofFS

FSofSGfacet — returns the facet index list of a Feature Surface for a given facet index

FSofSGnormal — finds all feature surfaces that fullfil the same normal vector condition

FSofSGT — returns the facet index list of a Feature Surface for a given coordinate system

MLofSG — calculates of the mounting face list of the solid

MLplot — plots the mounting faces of a solid

SGfeature — returns a cell list based result of a command sequence

SGinsideSurface — Checks the distance between the centers of the facets

TofSGfil — Provides the frame for a feature surface that is passed as a face index list.

TofSGFSselect — provides a frame that can be described using spatial terms as described in FSselect

TR3featureEdges — returns feature edge List plus edge attached facets

TR3mountingfaces — finds connected facets with same normal vector

TRfeatureEdgeFacets — returns the list of facets linked to edges of the featureEdges

VLFLofSGTsurface — returns the flange surface for a given T matrix frame name

Detailed description

FEofSG

returns FE and optional VL of a solid geometry or of gca

In: **SG**: if empty; VLFLofgca is used **ang**: angle; default is 0.4 ~ pi/8 **nf**: reserved parameter; currently unused **esort**: if true, sort the feature edges; default is true

Out: **FE**: Feature edge list **VL**: Vertex list for embedded SG

FEofVLFL

returns FE and optional VL of a solid geometry or of gca

In: **VL**: Vertex list if empty; VLFLofgca is used **FL**: Facet List **ang**: angle; default is 0.4 ~ pi/8 **esort**: sort edges; default is true

Out: **FE**: Feature edge list **VL**: Vertex list for embedded SG

FLfeatureEdgeSurface

returns a surface facet list based on a given surface triangulation

In: **TR3**: Surface volume triangulation **angle**: filter angle for feature edges

Out: **FL**: Facet List w.r.t TR3.Points

FLfeatureEdgeSurface2

returns a surface facet list based on a given surface triangulation

In: **TR3**: Surface volume triangulation **angle**: filter angle for feature edges

Out: **NFL**: Facet List w.r.t TR3.Points **PSIL**: Patch Surface Index List. **CEL**: Connecting Edge List CEL=[a b VLi VLj], a,b are indices of PSIL

FLinsideSurfaceofSG

Returns the Facet Indices of inside surfaces

In: **SG**: Solid Geometry **seper**: separate into independent surfaces first; default is true **maxtry**: Number of tries to find an opponing facet; default is 10 **wlim**: warning limit; default is 0.7 mm **ilim**: intended limit; distance that is intended; default is 0.1mm

Out: **FLi**: Facet surface index list **SG**: inside index of feature surface

FSfeature

Selects features surfaces based on

In: **SG**: Solid Geometry **FIL**: Cell list of facet indices **feat**: feature request string

Out: **F**: Result as cell list; scalars, vectors, frames, etc. **ty**: result data type string, e.g. 'scalar', 'point', 'direct', 't4', 'cpl', 'cvl'

FSofSG

Features surfaces of SG in contrast to selecting a FS FSselect and CPLofFS

In: **TR**: Solid Geometry or Triangulation **alpha**: maximum angle difference between two facets; default is 1 ~ 60 degree **sizesort**: if true, the surfaces are sorted by their size; default is false **Amin**: Minimum area to be classified as individual surface, used only during printout **"text"**: if used, the faces get text descriptors **"nvec"**: if used, normal vectors are plotted

Out: **CFL**: Index List for the Surface Indices **nF**: number of faces per feature surface **Ai**: Size of Feature surface **FNv**: Average Normal vector of FS

FSofSGfacet

returns the facet index list of a Feature Surface for a given facet index

In: **SG**: Solid Geometry **fi**: Facet index **alpha**: Alpha angle; default is .76

Out: **FIL**: Facet index list for one feature surface **fcl**: facet connectionlist

FSofSGnormal

finds all feature surfaces that fullfil the same normal vector condition

In: **TR**: Triangulation or SG **nv**: normal vector; default is [0 0 1] **alp**: alpha angle; default is 0.01 **rec**: if false, alpha is also the absolut limit to the nv vector; default is true;

Out: **FILk**: Cell list of facets that belong to the same feature surface **FILm**: Cell list of facets that belong to the same feature surface but are most similar to nv **FILi**: Cell list of one facet per surface that is most similar to nv

FSofSGT

returns the facet index list of a Feature Surface for a given coordinate system

In: **SG**: Solid Geometry **T**: Frame Name or HT matrix **dz**: optional backward distance for ez to find the ez-crossed facet index **alpha**: if an alpha angle is used, the FS will be recalculated anyway **dr**: optional orthogonal distance to ez to find the radial ez-crossing facets; not implemented yet

Out: **FIL**: Facet index list for one feature surface **fcl**: facet connectionlist

MLofSG

calculates of the mounting face list of the solid

In: **SG**: Solid Geometry **fi**: facet index or empty **fe**: feature edge angle (0.3) is default **amin**: minimal facet are to consider as start point;default is 0

Out: **ML**: Mounting face index list of FL (nx1), n=length(FL) **MA**: Array [nx5]= [mounting area index(1) area (1) normal vector(3)] **SG**: SG.ML und SG.MA added using ML and MA if fi=[];

MLplot

plots the mounting faces of a solid

In: **SG**: Solid Geometry **mi**: mounting face index; default is '' **fe**: feature angle; default is 0.5

Out: **h**: handle

SGfeature

returns a cell list based result of a command sequence

In: **SG**: Data Type to analyze features for **req**: SGcast and feature request or just feature request names

Out: **XX**: Result as cell list; **TT**: Types of the entries of the cell list **HSTACK**: Full history stack of the individual requests

SGinsideSurface

Checks the distance between the centers of the facets

In: **SG**: Solid Geometry **seperat**: Number of tries to find an opponing facet; default is 10 **wlim**: warning limit; default is 0.7 mm **ilim**: intended limit; distance that is considered as intended; default is 0.1mm **alfa**: optional factor; default is 1

Out: **FIL**: Facet surface index list **insFS**: inside index of feature surface

TofSGfil

Provides the frame for a feature surface that is passed as a face index list.

In: **SG**: Solid Geometry **fil**: Facet index list of a closed surface

Out: **T**: Coordinates frame of the facet index list **CPL**: Boundary contour of a planar feature surface

TofSGFSselect

provides a frame that can be described using spatial terms as described in FSselect

In: **SG**: Solid Geometry **ez**: ez for FSselect **descr**: more paramters for FSselect

Out: **T**: Frame **CPL**: Contour fits T

TR3featureEdges

returns feature edge List plus edge attached facets

In: **TR3**: Triangulation **alpha**: feature edge angle

Out: **EL**: Feature Edge List **Fi**: Index of TR3.ConnectivityList of attached facets

TR3mountingfaces

finds connected facets with same normal vector

In: **TR3**: Surface triangulation of a solid **fi**: singe facet index **fe**: feature edge angle; default is 0

Out: **FIL**: all facets connected to first facet with the same normal vector **nv**: normal vector of this facet **FnL**: all neighbor facets of FIL. WARNING: Always empty if fe>0 **NnL**: normal vector angle difference to nv of FnL **RL**: Radial list created by CVLdimclassifier **CVL**: CLosed Vertex list used for CVLdimclassifier **TR3**: TR# is case that an SG was used as input parameter

TRfeatureEdgeFacets

returns the list of facets linked to edges of the featureEdges

In: **TR3**: triangulation nx3 **alpha**: filterangle for featureEdges

Out: **[ELFL]**: Edge-facet list of the feature edges

VLFLofSGTsurface

returns the flange surface for a given T matrix frame name

In: **SG**: Solid Geometry **N**: Name of Frame **t**: thickness of the solid; default is 0.5 **d**: distance from the surface; default is 0.3 **s**: streching of the border line; default is 0

Out: **SVL**: Vertex List **SFL**: Facet List

4.4 Surfaces to Solid Geometries [guide]

5 Solid Geometry Modeling [guide]

5.1 SG basic shapes [guide]

Functions in this section:

LCLYK1_create — Script to collect different STL files to create a cell list of solids

SGanycubicPredatorCover — This fnct was made to motivate Markus Wörl

SGanycubicPredatorCoverPMMA — calculates size and testing elements for the plexi glas shielding for the Anycubic predator

SGbanditjethelmclip — creates a clip for a Bandit Jet Helmet

SGbase4PCB — returns base posts and pins for a PCB assembly

SGbearing — returns the SG of a simple bearing

SGbladeofpincone — creates a subtraction solid

SGbottleholder — returns the geometry of a simple bottle holder

SGbox — returns soldi geometry of a origin centered box

SGboxcage — creates a box as cage based on SGbarstructure

SGboxMacmini — designs a cylinder shaped box for my office desk to hide cables and stuff

SGcompass — creates a desktop housing for the magnetic compass (look like a Gyro compass)

SGcone — returns a solid geometry of a cone

SGconetruncated — returns a truncated cone by

SGcover4PCB — creates a cover with openings for cables etc

SGcylinder — returns a simple cylinder

SGdesignrules — Tim Lueth's design rules for creating 3D printable products for daily living

SGdumbbell — returns the shape of a dumbbell

SGdummyRjoint — creates a simple geometry to better imagine a rotating joint

SGegg — returns the Surface Geometry of an egg based on an evolvente

SGFilamentHolderFixation — returns a plate to fixate the distance of two independent filament holder stands type OTOTEC

SGfingerprotector — creates a simple finger guard for one finger

SGfischertechnikaxle — A pluggable axle for edges and gears for Fischertechnik

SGgyroid — creates a gyroid shape solid

SGhullstripsagittal — returns a string around the sagittal plane

SGinsertrevjoint — inserts a revolute joint in the center of a solid

SGiPadFlip2stand — creates a stand to insert an Apple iPad into the opening of an Samsung Flip2 55 Stand

SGiqanglerect — returns a recangular angle for changing directions in mechanisms

SGiqknob — returns a inch quarter knurled knob for inserting hexagon pin as bolt or shaft for mechanisms

SGiqlink — Creates links for inch quarter drive mechanisms

SGiqpin — returns a inch quarter hexagon pin as bolt or shaft for mechanisms

SGiqsocket — returns a socket or a foot for an inch quarter mechanism

SGjointball — returns ball and shell of a 3DoF joint

SGjuicermachineclip — A missing useful part for the Bosch juicer machine

SGjunkerhallpart — creates a part for a junker hall

SGlinkage — returns the solid geometry of a linkage

SGlinkchain — returns a foldable link chain

SGMacBookPlate — just plots a stand for a keyboard ontop of a 13" McBook

SGmicroscopetable — creates a part to connect a xy-table with a microscope stand of USB-Microscope

SGmotorFeeTech40 — returns a model of an SM40BL that can be printed and moved passively

SGmotorN20 — returns just the Geometry of a N20 motor

SGmotorN40 — returns the Geometry of a Chihai motor CHF-GM13-050S

SGmotorN40horiz — returns the Geometry of a N20 HEININGER Motor with 90 degree flansch

SGmotorshaft — returns a solid geometry for a motor shaft with undercut

SGmotorSolExpert — return the simplified solid Geometry of a 50:1 Gearbox Motor 6 V of www.sol-expert-group.de

SGNameplate — returns name plates

SGpincountersink — creates cone shaped pins and blades

SGpinring — Returns the shape of a ring or a blade to create revolute joints

SGpintorquegauge — Creates a pin with a defined diameter and height with a torque-loadable head

SGpinwrenchsize — creates a pin/heaxagon pin for a specified metric wrench size

SGpost — returns a force optimized post

SGraspicam2housing — returns a housing for a Raspberry Pi camera type 2 (adjustable lenses)

SGrobot — returns the geometry and kinematic model of the MIMED-LCL Robot

SGrobotTCPchanger — Simple Tool Changer for Yannick's robot

SGrobotTCPdesign — Tutorial fnct to design Robot Tools

SGruler — creates a ruler in dimensions that can be so small that it cannot be manufactured but displayed

SGshapecylindric — shapes a solid with a cylindrical intersection

SGsilentbrass — creates a holder for a Yamaha Silent Brass for a Thomann Cornett

SGslalomshoes — Finn-Foil-Protector for Slalomski

SGsphere — returns a spherical SG with minimal nr of points

SGsupportFS — STL processing fnct to increase or reduce the wall thick by material support

SGsurfacecutoutncover — cuts out a contour and create a CONVEX solid to cover the cavity

SGtext — returns a SG that does fit on the surface on another SG

SGtextbanner — Returns a solid an a CPL for a size adjusted Textbanner

SGtextstamp — adds a textstring on the surface

SGtextTUM — return a solid for the TUM logo

SGtorquelimiteraxial — returns a torque limiter part

SGtrapezoid — creates trapezoid pins for molding forms

SGturn4AM — simply turns an object if the upper printing surface is larger than the lower floor printing area

Detailed description

LCLYK1_create

Script to collect different STL files to create a cell list of solids

SGanycubicPredatorCover

This fnct was made to motivate Markus Wörl

Out: **SGall**: Final Solid

SGanycubicPredatorCoverPMMA

calculates size and testing elements for the plexi glas shielding for the Anycubic predator

SGbanditjethelmclip

creates a clip for a Bandit Jet Helmet

SGbase4PCB

returns base posts and pins for a PCB assembly

In: **Bsiz**: [x y z] size of a PCB **PinL**: Position of points **PinS**: [Ro, Ri, h, Ro, Ri, h, ...] **rel**: spatial relation for adusting the pin pattern

Out: **SG**: Solid Geometry

SGbearing

returns the SG of a simple bearing

In: **Sz**: [Di Do H os] oversize default is 0 **Tp**: 'plain', '2012', **Ax**: axis to left and right

Out: **SG**: Solid Geometry

SGbladeofpincone

creates a subtraction solid

In: **r**: r or [r h]; default is [1.5 1.5] **pl**: play/slot in addition to clearance fit **R**: Outer radius for create movement freedom in x/y **ol**: overlap for supplement

Out: **SGB**: Solid for subtraction/ Blade **SGA**: Solid as Suppelement **SGS**: Solid as Suppelement

SGbottleholder

returns the geometry of a simple bottle holder

In: **D**: Diameter of the bottle; default is 68mm **L**: Length of the Hold **B**: Wall tickness of all elements; default is 3 **H**: Height of the rings and struts **n**: number of struts

Out: **SA**: Geometry of the bottle holder with Frames 'B' and 'F'

SGbox

returns soldi geometry of a origin centered box

In: **S**: size vector type [x y z] **n**: numbers of edges; default =4 **pack**: for several box; true = separated; false = centered; default is false **noframe**: 'noframe' flag to omit the coordinate frame **"noframe"**: 'noframe' flag to omit the coordinate frame

Out: **SG**: VL / FL of the box

SGboxcage

creates a box as cage based on SGbarstructure

In: **siz**: [x y z]; default is [30 20 10] **dim**: [bx by bz]; default is [2 2 2] **typ**: 'xzzxxo' as default for the cage pattern

Out: **SG**: Box as bar stucture

SGboxMacmini

designs a cylinder shaped box for my office desk to hide cables and stuff

Out: **box**: SG of the box **cov**: SG of the cover

SGcompass

creates a desktop housing for the magnetic compass (look like a Gyro compass)

Out: **SG**: Just returns the geometry of the compass

SGcone

returns a solid geometry of a cone

In: **H**: Height **R**: Radius; default is H/2 **nf**: Number of facets, default is 16 **Hc**: height of the Truncated cone

Out: **SG**: Vertex list **FL**: Facet list of the cone **FLb**: Facet list of the bottom

SGconetruncated

returns a truncated cone by

In: **R1**: Bottom Radius **R2**: Top Radius **H**: Height **n**: Option number of polygon points

Out: **SG**: Solid Geometry **FLW**: Facet list of Wall **FLA**: Facet list of Floor **FLB**: Facet list of Roof

SGcover4PCB

creates a cover with openings for cables etc

In: **SGB**: Solid plate to cover **h**: [h w]; height interlab and wall thickness (default is 1.5) **sl**: default is slfit('t') **constr**: cell of {side number, CPL, spatial relations}

Out: **SGN**: Resulting cover solid

SGcylinder

returns a simple cylinder

In: **r**: Radius **h**: Height **n**: number of points **ri**: Inner Radius **ry**: Radius in y **al**: Angle of circle; default is 2*pi **"frames"**: flag; if set, add body-frame coordinate systems via SGTsetBFsimple

Out: **SG**: Solid Geometry

SGdesignrules

Tim Lueth's design rules for creating 3D printable products for daily living

SGdumbbell

returns the shape of a dumbbell

In: **type**: design method such as 'bool' or 'rot'

Out: **SG**: Solid Geometry

SGdummyRjoint

creates a simple geometry to better imagine a rotating joint

In: **R**: [Ro Ri] **H**: [Hi Ho] **w**: rotation angle; default is 0 **T**: Transformation matrix; default is eye(4)

Out: **SG**: SOlid Geometry of a dummy revolute joint

SGegg

returns the Surface Geometry of an egg based on an evolvente

In: **r**: radius; default is 10 **a**: alpha of evolvente; default is pi

Out: **SG**: Solid

SGFilamentHolderFixation

returns a plate to fixate the distance of two independent filament holder stands type OTOTEC

In: **M**: Screw Diameter; default is 2.5 **D**: Distance between Roller and Distance in a slot; default is [97.5 70]

Out: **SG**: SG for 3D Print

SGfingerprotector

creates a simple finger guard for one finger

In: **DLW**: default is [23 80 0.8 nan nan nan] **TL**: Cell list with z values to create rotational links afterwards

Out: **SG**: Solid Geometry for 3D print

SGfischertechnikaxle

A pluggable axle for edges and gears for Fischertechnik

Out: **SG**: SOlid Geometry

SGgyroid

creates a gyroid shape solid

In: **R**: Cubic size **w**: angle; default is 4*pi **D**: wall Thicknessl default is 0

Out: **SG**: Gyroid solid

SGhullstripsagittal

returns a string around the sagittal plane

In: **A**: Solid **x**: x coordinate of the strip center **b**: width of the strip **w**: wall thickness of strip

Out: **X**: Solid Geometry of the Strip

SGinsertrevjoint

inserts a revolute joint in the center of a solid

In: **SG**: Solid Geometry **To**: Position of the Joint **a**: alpha (amin, amax) **delta**: slot size **bs**: length and height of joint **ao**: Angle offset

Out: **C**: Joint Solid **A**: Solid with hole (green) **B**: Solid with pin (Yellow) **Tc**: Center transformation matrix

SGiPadFlip2stand

creates a stand to insert an Apple iPad into the opening of an Samsung Flip2 55 Stand

In: **L**: Length up to 320 mm; default is 100mm

SGiqanglerect

returns a recangular angle for changing directions in mechanisms

In: **L**: [Height Radius edge-number] **slb**: ring distance; default is slfit('s') separated **a**: ring anglel default is pi/2 90degree **rs**: ring size default is 2mm **w**: wall thickness; default is 2mm **slw**: Inch quarted pin fitting; default is slfit ('t') -

SGiqknob

returns a inch quarter knurled knob for inserting hexagon pin as bolt or shaft for mechanisms

In: **H**: Height default is 4 rings (16 mm) **iqpar**: [wall ring angle slfit('c')]; default is [2 2 pi/2 slfit('c')] **RL**: Wrench size diameter; default is 25.4/4 mm

Out: **SG**: Pin with Base frame and Follower frame

SGiqlink

Creates links for inch quarter drive mechanisms

In: **L**: Length in mm; default is 50 **slb**: Fitting for bearing; default is slfit('s') **a**: angle for bearing; default is 90 degree pi/2 **rs**: intrusion of ring; default is 2mm **w**: wall thickness ; default is 2mm **slw**: Fitting for IQ sockets; default is slfit('t')

SGiqpin

returns a inch quarter hexagon pin as bolt or shaft for mechanisms

In: **H**: Height default is 4 rings (16 mm) **iqpar**: [wall ring angle slfit('c')]; default is [2 2 pi/2 slfit('c')] **RL**: Wrench size diameter; default is 25.4/4 mm

Out: **SG**: Pin with Base frame and Follower frame

SGiqsocket

returns a socket or a foot for an inch quarter mechanism

In: **L**: [Height Radius edge-number] **slb**: ring distance; default is slfit('s') separated **a**: ring anglel default is pi/2 90degree **rs**: ring size default is 2mm **w**: wall thickness; default is 2mm **slw**: Inch quarted pin fitting; default is slfit ('c') -

SGjointball

returns ball and shell of a 3DoF joint

In: **R**: Radius of the Sphere; default is 18mm **wl**: Wall = size of the cavity; default is 2mm **pl**: play ~ wrt to clearance fit; default is 0.1 mm **olap**: exuberant edge; default is 0.2mm **plat**: if true; cylyndrical plates are attached

Out: **X**: {Ball, Shell} **CC**: optional collar/ring solid created when a platform is used; otherwise empty

SGjuicermachineclip

A missing useful part for the Bosch juicer machine

In: **name**: Name string on back side

Out: **SGB**: Solid Geomerty to print

SGjunkerhallpart

creates a part for a junker hall

In: **h**: list of sections; default is [5 10 10 5] **t**: radius of impression; default is [0 4] **a**: alpha for impressipm; default i pi/5 **w**: wall thickness; default is t(2)/4 **r**: radius for breaking edges; default is 0.5 **LL**: [a b] straight length an ramp length

Out: **SG**: Solid geometry

SGlinkage

returns the solid geometry of a linkage

In: **R**: Radius of the R-joint **D**: Distance between the two R-Joints **H**: Height on z of the linkage **RA**: Radius of the linkage (Default is 1.5 * R) **D2**: Optional Distance in x for a 3rd hole

Out: **SG**: Solid geometry VL/FL

SGlinkchain

returns a foldable link chain

In: **L1**: Length; default is 120mm for LCL **D**: Diameter; default is 12mm for LCL **M**: DIN912/985; default is [3 8] for M3 x 8 DIN912 **flat**: if true, the chain is built flat (planar); default is true **styl**: style string 'start'/'end' to adjust leg length and frames; default is empty

Out: **SGA**: Long leg or {Part A, PartB} **SGB**: Short leg

SGMacBookPlate

just plots a stand for a keyboard ontop of a 13" McBook

In: **keypad**: keypad variant; 'window' cuts a keypad opening, 'short' shortens the plate; default is false

Out: **A**: Solid Geometry

SGmicroscopetable

creates a part to connect a xy-table with a microscope stand of USB-Microscope

Out: **SGA**: Solid Geometry

SGmotorFeeTech40

returns a model of an SM40BL that can be printed and moved passively

In: **"STL"**:

Out: **SGall**: Screw, Nut and Box

SGmotorN20

returns just the Geometry of a N20 motor

In: **style**: motor style; 'encoder' adds an encoder body; default is standard

Out: **SG**: Solid Geometry of a N20 Motor

SGmotorN40

returns the Geometry of a Chihai motor CHF-GM13-050S

In: **sh**: Shaft Diameter and Shaft Flattening; default is [3 2.5] **sz**: [maxsize-x maxsize-y length-shaft diameter shaft-socket, height shaft-socket]; default is [15.4 12 9 4 1] **gb**: [maxsize-x maxsize-y length-shaft diameter shaft-socket, height shaft-socket]; default is [15.4 12 9 4 1] **PL**: Point list of screws; default is [0 -4.5; 0 4.5]

Out: **SG**: Solid Geometry of a N20 Motor by Sebastian Heininger Selection

SGmotorN40horiz

returns the Geometry of a N20 HEININGER Motor with 90 degree flansch

In: **sh**: Shaft Diameter and Shaft Flattening; default is [3 2.5] **sz**: [maxsize-x maxsize-y length-shaft diameter shaft-socket, height shaft-socket]; default is [10 12 6.5 4 .5] **gb**: gearbox [size-x size-y height motor-length Mscrews shift-x shift-y]; default is [18 12 10 14 1.6 5 0] **PL**: Point list of screws; default is [-2.5 -4.5;-2.5 +4.5]

Out: **SG**: Solid Geometry of a N20 Motor by Sebastian Heininger Selection

SGmotorshaft

returns a solid geometry for a motor shaft with undercut

In: **D**: Diamater, default is 0.25 **L**: Length, default is D **r**: Undercut, default is .25 **ol**: Step, default is r

Out: **SG**: Solid Geometry including Frames **PL**: Point List

SGmotorSolExpert

return the simplified solid Geometry of a 50:1 Gearbox Motor 6 V of www.sol-expert-group.de

In: **mn**: option Name such as G50V6 **d**: Diameter of the gear wheel; default is 10 **n**: Number of teeth of the gear wheel; default is 21 **h**: Height of the gear wheel; default is 7

Out: **SG**: Solid Geometry including Frames **SGG**: Gear 2mm, 15 Teethm

SGNameplate

returns name plates

In: **txt**: text such as 'Luisa' **siz**: size in mm; default is [0 15 2] **D**: Diameter of Holes **wall**: Distance default is [D D/2]

Out: **C**: solid geometry of the name plate (text plate with mounting holes)

SGpincountersink

creates cone shaped pins and blades

In: **d**: diameter or [diameter height]; default is [3 1.5] **ol**: overlap in z; default is 0.1 **bl**: oversize in radial; default i 0 **R**: Ring Radius **V**: Shape of Ring "H', "V',, "VR', 'HR', "ART1', ART2

Out: **SG**: Solid Geometry

SGpinring

Returns the shape of a ring or a blade to create revolute joints

In: **H**: List of height segments [h1 h2 h3] **R**: [Ri Ro] default is Ro=Ri+slfit(c) **a**: angle; default is pi/180*60 **n**: number od edges; default is nofrd **ol**: optional oversize at floor and top; default is 0; **os**: optional oversize to create Blades in a distance; default is 0

Out: **SG**: Solid geometry to print or subtract

SGpintorquegauge

Creates a pin with a defined diameter and height with a torque-loadable head

In: **d**: diameter **l**: length **m**: metric Nut and Scre concept for DIN985, DIN912 **L**: length of the screw/nut section; default is 8 **w**: wall thickness; default is 2mm **groove**: if true, adds a groove pattern along the pin; default is false

Out: **SG**: Solid Geometry (white)

SGpinwrenchsize

creates a pin/heaxagon pin for a specified metric wrench size

In: **D**: Metric Diameter **H**: Length of pin or 'c'(cylinder) for 'n'(ut) **n**: number of edges **B**: Width as subtraction sold **W**: Angle for rotation

Out: **SG**: Solid Geometry

SGpost

returns a force optimized post

In: **R**: Radius **H**: Height **Sh**: Shape parameter

Out: **SG**: Solid Geometry

SGraspicam2housing

returns a housing for a Raspberry Pi camera type 2 (adjustable lenses)

Out: **SGP**: Plate **SGB**: Box

SGrobot

returns the geometry and kinematic model of the MIMED-LCL Robot

In: **phi**: list of angles in rad; default is [0 0 0 0]

Out: **SG**: Cell list of 5 solids

SGrobotTCPchanger

Simple Tool Changer for Yannick's robot

Out: **D**: Toolside (red) Adapter **B**: Robot Side (gray) TCP **E**: tool adapter solid combined with the ball joint (spoon adapter)

SGrobotTCPdesign

Tutorial fnct to design Robot Tools

In: **SG**: Solid Geometry to mount to TCPtool **TN**: Frame Name at SG to move TCPtool to **d**: distance to adjust TCP tool

Out: **[TCPtool]**: Just the Tool

SGruler

creates a ruler in dimensions that can be so small that it cannot be manufactured but displayed

In: **L**: length of ruler; default is 7mm **H**: graduation step unit; default is 10^(ceil(log10(L/2)-1))

Out: **SGR**: Solid Geometry of a Ruler of length L and width L/5

SGshapecylindric

shapes a solid with a cylindrical intersection

In: **SG**: Solid Geometry **ez**: shaping direction;default is [1 0 0]; x axis **ce**: centered or not centered; default is false **R**: option Radius if not calculated from Solid; **dz**: z shift of the cutting contour; default is 0

Out: **SGN**: Final Solid

SGsilentbrass

creates a holder for a Yamaha Silent Brass for a Thomann Cornett

In: **DD**: dimensions [tubeDiameter tubeDistance lowerTubeLength clipWidth]; default is [13.3 75 60 8]

Out: **SG**: Silent brass holder with clearance fitting

SGslalomshoes

Finn-Foil-Protector for Slalomski

In: **LL**: length of proptector ; default is 50mm **SkiS**: size of Ski stern; default is [60 90 210 14] **Rail**: size of rail; default is [25 210 14] **FinS**: size of fin; default is [3 18 65] **Foil**: size of foil protection box; default is [43 43 43] **"cover"**: if true; the tube gets an floor cover

Out: **SG3**: Solid Geometry to print **CPL3**: Contour **CPL2**: contour of tube **CPL4**: contour of connector bars

SGsphere

returns a spherical SG with minimal nr of points

In: **R**: Radius **nr**: number of points; default is []; **wt**: top angle; default is +pi/2 **wb**: bottom angle; default is -pi/2 **Ry**: Radius in Y; default is R **Rz**: Radius in Z; default is R

Out: **SG**: Solid Geometry **FLW**: Faces of the wall, related to SG.VL **FLB**: Faces of the bottom, related to SG.VL **FLT**: Faces of the cover, related to SG.VL

SGsupportFS

STL processing fnct to increase or reduce the wall thick by material support

In: **SG**: Solid Geometry **fi**: list of indices of different FS **wt**: wall thickness; default is 2.5mm can also be negative **in**: intrusion; default is -0.05 == 50µm

Out: **SG**: Concat Geometry **SGA**: the added material support geometry (from SGofSurface)

SGsurfacecutoutncover

cuts out a contour and create a CONVEX solid to cover the cavity

In: **SG**: Solid Geometry **delta**: depth to cut **shift**: convex point list for cutting **VLC**: convex point list for cutting

Out: **SG**: Solid geometry with removed contour **B**: Cover plate to close the removed contour

SGtext

returns a SG that does fit on the surface on another SG

In: **SG**: Solid Geometry to mount the text **stext**: String to mount, lines with "\n" **ez**: Normal vector/Rotation of the union area to mount on [x y z w] **sx**: optional parameter for the size **sy**: size in y **ha**: horizontal alignment (if several lines) such as 'l' or 'r' **pos**: optional override position [x y z] for the text center; default is [nan nan nan]

Out: **SGT**: Solid geometry of the text string

SGtextbanner

Returns a solid an a CPL for a size adjusted Textbanner

In: **txt**: Text **sz**: [width height border thickness]; default is 120 25 5 2] **r**: optional radius; default is automatic

Out: **SG**: SG **CPL**: SG contour **CPLT**: Text contour **CPLB**: Box Contour

SGtextstamp

adds a textstring on the surface

In: **SG**: Solid Geometry **stext**: text for stamp **ez**: orientation for surface, or Frame name to turn! **col**: color; default is none **L**: text size/scaling passed to SGtext as Lx (and Ly if two values)

Out: **SGS**: New solid with stamp **B**: Just the Text Solid

SGtextTUM

return a solid for the TUM logo

In: **fs**: font size; default is 16 mm **z**: height; default is 1 mm

Out: **SG**: Solid Geometry

SGtorquelimiteraxial

returns a torque limiter part

In: **R**: Radius [Router Rinner] **m**: modul; number of functions / cosinus **h**: height of the base **s**: amplitude of curve (one half is ontop of base) **"sin"**: if set, uses a sinusoidal curve profile instead of the default zigzag

Out: **SG**: Solid geometry

SGtrapezoid

creates trapezoid pins for molding forms

In: **r**: [rin rout] **h**: height **rad**: radial edges radius **n**: number of points on the rotation circle

Out: **SG**: Solid Geometry

SGturn4AM

simply turns an object if the upper printing surface is larger than the lower floor printing area

In: **SG**: Solid Geometry

Out: **SGN**: Turned or unchanged Solid

5.2 SG Generation [guide]

5.2.1 SG of CPL Extrusion [guide]

Functions in this section:

SGofCPLbendonadrum — bends a CPL pattern into a cylindrical body (slowly) or rolls it onto a drum

SGofCPLcontour — extrudes a CPL as cross section contour along another CPL as extrusion contour

SGofCPLextrude — returns an extrudes solid mainly the same as SGofCPLz

SGofCPLextrudealongCPL — extrudes a contour ([x z]) along a 2D-contour [x y]

SGofCPLhatch — creates a solid using CPLhatch

SGofCPLreinforcement — creates edge reinforcements based on a CPL

SGofCPLrot — returns a solid geometry from of a z-axis rotated closed polygon list

SGofCPLrota — returns a solid geometry by rotateing a closed x z contour around z axis

SGofCPLsphere — returns solid geometry of a wrapped shell

SGofCPLT — returns a extruded SG from a CPL and a final T matrix

SGofCPLTL — extrudes a contour along a path given by TL

SGofCPLtransT — returns a solid from a 2D contour and a list of frames

SGofCPLz — returns a 2½D extruded Solid volume from a contour polygon list (CPL)

SGofCPLzchamfer — returns a solid with chamfered edges

SGofCPLzcontour — extrudes a CPL into a Solid Geometry modifying the CPL along z using a contour shape

SGofCPLzdelaunayGrid — returns a solid from a contour polygon list and grid resolution

SGofCPLzfillgaphull — extrudes several regions without subregions into one region

SGofCPLzgrid — creates a Contour by extrusion using a z coordinate list without changing top and floor surface

SGofCPLzgrow — creates solids with radial changing contours

SGofCPLzradialedge — similar to SGofCPLzchamfer

SGofCPLzrotxy — rotates a CPL and extrudes in z

SGofCPLzseparated — creates for each closed contour of a CPL an individual solid with an individual height

SGofCPLztwist — Creates twisted solids from a CPL and a turning angle

Detailed description

SGofCPLbendonadrum

bends a CPL pattern into a cylindrical body (slowly) or rolls it onto a drum

In: **CPLN**: Contour to bend **h**: height of the material in z **ol**: overlap; default is 0 **res**: resolution for CPL mesh grid; default is r/10

Out: **SGB**: Solid Geometry

SGofCPLcontour

extrudes a CPL as cross section contour along another CPL as extrusion contour

In: **RL**: CPL of a profile to be extruded [x z]; default is CPLsample(3) **CPL**: CPL of a contour to extrude the RL contour default is CPLsample(2)*20 **adjst**: if false; minimum x of RL becomes the point on CPL; default is true **r**: radius to avoid sharp angles; default is empty => automatic in case of adjst==true

Out: **SG**: Solid Geometry

SGofCPLextrude

returns an extrudes solid mainly the same as SGofCPLz

In: **CPL**: CPL **z**: length or [min max] **ez**: '"X" , "Y", OR 'Z' as extrusion direction **ci**: optional contour selection sich as 0 for outer boundary

Out: **SG**: Solid Geometry

SGofCPLextrudealongCPL

extrudes a contour ([x z]) along a 2D-contour [x y]

In: **RL**: Point list in x and z **CPL**: Contour path for extrusion; considered as close **adjst**: if true; minimal x value of RL is defined by the contour path CPL; default is true **r**: radius for outer contour; default is widhth of RL

Out: **SG**: Solid Geometry

SGofCPLhatch

creates a solid using CPLhatch

In: **CPL**: Contour **d**: [thickness and distance of hatch bars] **a**: angle of hatch or [hatch crosshatch] angles; default is 51 degree **zz**: height of solid **"cross"**: if used; there will be crossed angle=[a a+90]

Out: **SG**: Solid Geometry

SGofCPLreinforcement

creates edge reinforcements based on a CPL

In: **CPL**: Edge contour (inner CPL for positive dept, outer CPL for negative depth) **f**: [depth heigh radius]; default is [5 5 0]

Out: **F**: Solid edge reinforcement

SGofCPLrot

returns a solid geometry from of a z-axis rotated closed polygon list

In: **CPL**: Radius-Z List **n**: optional number of edges of the rotated polygon **rcorr**: true = increase that R is on circle; default R is incircle

Out: **SG**: Solid Geometry (SG.VL, SG.FL, SG.CPL)

SGofCPLrota

returns a solid geometry by rotateing a closed x z contour around z axis

In: **CPL**: Closed contour in [x z] **alp**: sweeping angle **mov**: false = cut; negative x values; true=move to zero **amin**: angle offset; default is 0 **h**: optional height to create gear worms etc. **autc**: flag for automatic contour correction; default is true

Out: **VL**: Vertex list or SG **FL**: Facet list

SGofCPLsphere

returns solid geometry of a wrapped shell

In: **CPL**: Closed Polygon List **Ri**: Radius of the sphere used for wrapping **d**: Thickness/height of the shell **ns**: distance between auxiliary points; default is sofrd(Ri)

Out: **SG**: Solid geometry

SGofCPLT

returns a extruded SG from a CPL and a final T matrix

In: **CPL**: CPL of several contours **T**: Transformation Matrix

Out: **SG**: VL,FL,T,Tname

SGofCPLTL

extrudes a contour along a path given by TL

In: **CPLC**: Contour to extrude along TL **TL**: array [4,4,n] of the path

Out: **SGN**: Final Solid including VL, FL **NFW**: just FL of the walls **FLA**: just FL of the floor **FLB**: just FL of the ceiling

SGofCPLtransT

returns a solid from a 2D contour and a list of frames

In: **CPL**: Closed Polygon List separated by nan **T1**: Frame 1 [4 x 4] or List of frames [4 x 4 x n] **T2**: Frame 2 [4 x 4], optional

Out: **SG**: Solid Geometry SG.VL, SG.FL **FLS**: Facets of the start cover **FLE**: Facets of the end cover **FLW**: Facets of the contour wall **TS**: starting frame **TE**: ending frame

SGofCPLz

returns a 2½D extruded Solid volume from a contour polygon list (CPL)

In: **CPL**: Contour polygon list (CPL) **z**: Height z can be scalar or [zmin zmax] **T**: optional Transformation after Solid creation; useful for cutting blades **"STL"**: if wall>0; CPL is the outside contour and wall is the thickness **"aux"**: if cover>0; The solid gets a conver **"wall"**: if floor>0; The solid gets a floor **"cover"**: this contour is subtracted from the cover contour **"csub"**: this contour is subtracted from the floor contour

Out: **SG**: Vertex list, Facet list, Point list, Edge list **FLW**: Facet list of wall **FLA**: Facet list of floor **FLB**: Facet list of ceiling

SGofCPLzchamfer

returns a solid with chamfered edges

In: **CPL**: Closed Polygon line **z**: Height z **ph**: phase default is 0.3 **ed**: curved edges; default is true

Out: **SG**: Resulting solid

SGofCPLzcontour

extrudes a CPL into a Solid Geometry modifying the CPL along z using a contour shape

In: **CPL**: Contour to extrude **z**: height for extrusion **ph**: scale of reduction ; default is 0.3 = 30% **ed**: round corners; default is false **shp**: such as 'circ', bezier' ,'mattheck'; see PLchamfer for strings

Out: **SG**: Final Solid Geometry **SFLW**: Surface facets of the wall **SFLA**: Surface facets of the floor **SFLB**: Surface facets of the ceiling

SGofCPLzdelaunayGrid

returns a solid from a contour polygon list and grid resolution

In: **CPL**: Contour polygon list (CPL) **z**: desired height in z-coordinates **ds**: delta on the contour; default is 10000 **dx**: delta grid in x; default is 10000 **dy**: delta grid in y; default is dx

Out: **SG**: Solid Geometry **CPL**: closed polygon list used for the triangulation

SGofCPLzfillgaphull

extrudes several regions without subregions into one region

In: **CPLA**: Original Contour consisting of several regions without subregions **z**: height to extrude **bf**: buffer around CPLA used for CPLconvexhulloutline - fillgap **ax**: optional auxiliary point distance for single contour at top **tp**: convex hull type passed to CPLconvexhulloutline: 'default', 'delaunay', or 'fillgap'; default is 'fillgap' **CPLB**:

Out: **SG**: Solid Geometry VL/FL **FLXW**: Just Wall with respect to SG.VL **FLXA**: Just Floor with respect to SG.VL **FLXB**: Just Ceiling with respect to SG.VL **CVL2**: Top Contour

SGofCPLzgrid

creates a Contour by extrusion using a z coordinate list without changing top and floor surface

In: **CPL**: Contour **zz**: list of z values **dx**: list of linear corresponding x deviation for each z value; default is 0 **dy**: list of linear corresponding x deviation for each z value; default is 0

Out: **SG**: SOlid Geometry **nva**: number of vertices per plane **FLW**: Only the walls **FLU**: Only the floor **FLO**: Only the cover

SGofCPLzgrow

creates solids with radial changing contours

In: **CPL**: Closed Contour **zz**: list of z values **dr**: list of corresponding buffer/grow values for CPLgrow (not CPLbuffer) **"debug"**: used in CPLgrow "norm","outw","hole", "fitt" default is 'hole' **"style"**: CPLgrow style used for contour offset: 'norm','outw','hole','fitt'; default is 'hole' **"aux"**: auxiliary point spacing for CPLaddauxpoints; forced disabled internally (not allowed with CPLbuffer)

Out: **SG**: Solid Geometry **nva**: list of points per z value **FLW**: Faces of wall **FLU**: Faces of floor **FLO**: Faces of cover

SGofCPLzradialedge

similar to SGofCPLzchamfer

In: **CPL**: Contour to extrude **z**: height or [zmin zmax] **rh**: [radius and height] **shape**: 'mattheck', 'bezier', 'circ', 'line'; default is mattheck **style**: edge style passed to SGofCPLzgrow; default is 'hole' **"debug"**: if set, prints the zz and dr edge vectors

Out: **SG**: Extruded solid with round edges on bottom and top

SGofCPLzrotxy

rotates a CPL and extrudes in z

In: **CPL**: CPL **z**: height **w**: roatation angle in [x y z]

Out: **SG**: Solid Geometry

SGofCPLzseparated

creates for each closed contour of a CPL an individual solid with an individual height

In: **CPL**: CPL **hl**: sorted list for height in CPL

Out: **SG**: Solid Geometry

SGofCPLztwist

Creates twisted solids from a CPL and a turning angle

In: **CPL**: CPL (centered at origin) **z**: height or [zmin zmax] **w**: turning angle

Out: **SG**: solid geometry

5.2.2 SG of CPL Connections [guide]

Functions in this section:

PL2T — Expands two HT Matrizes into 8 edges/vertices of the corresponding bar between T1 and T2

SGof2CPLsz — returns a solid defined by 2 contours and height

SGof2CPLTLcorrelate — creates a solid between two vessel structure CPL

SGof2CPLtransition4FDM — creates transition solid to avoid support structures for FDM printer

SGof2CPLz — returns a solid defined by 2 contours and height

SGof2CPLzbranch — Creates a Solid for an 1:n or n:1 solid branch

SGof2CPLzcorrelate — creates a solid between two vessel structure CPL

SGof2CPLzheurist — EXPERIMENT to show the connection of two contours

SGof2CVL — returns a solid that closes two

SGof2Dfunc — creates a surface or a solid of a 1D or 2D fnct

SGof2SGT — Creates a solid between two solids with frames

SGof2T — Creates a solid geometry between two frames

Detailed description

PL2T

Expands two HT Matrizes into 8 edges/vertices of the corresponding bar between T1 and T2

In: **T1**: Start of bar **T2**: End of bar **bx**: size in x direction **by**: size in y direction

Out: **PL**: List of 8 vertices to describe the edges of a bar from T1 to T2

SGof2CPLsz

returns a solid defined by 2 contours and height

In: **CPA**: ONE bottom Contour **CPB**: ONE top Contour **z**: height of the solid **stype**: "number", "length", "angle", "center"; default is "length" **ctype**: "rot" or "miny"; default is "rot" **"aux"**: ONE top Contour

Out: **SG**: Solid Geometry; VL=[CPA;CPB] **FLW**: Face list of Wall **FLA**: Face list of CPA **FLB**: Face list of CPB

SGof2CPLTLcorrelate

creates a solid between two vessel structure CPL

In: **CPLU**: Contour A **CPLO**: Contour B **TL**: List of HT Matrices used follow the path

Out: **SGN**: Solid Geometry **FLW**: Facet of walls wrt to SGA.VL **FLA**: Facet of floor wrt to SGA.VL **FLB**: Facet of cover wrt to SGA.VL **FLC**: Faces missing in A wrt to SGA.VL **FLD**: Faces missing in B wrt to SGA.VL

SGof2CPLtransition4FDM

creates transition solid to avoid support structures for FDM printer

In: **CPLB0**: Lower Contour **CPLi**: Higher Contour **rmin**: optional minimal angle **alpha**: optional transition angle; default is 50/180*pi

Out: **EE**: Transition solid **hE**: height of EE in z

SGof2CPLz

returns a solid defined by 2 contours and height

In: **CPA**: ONE bottom Contour **CPB**: ONE top Contour **z**: height of the solid **stype**: 'number', 'length', 'angle', 'center'; default is 'length' **ctype**: 'rot' or 'miny'; default is 'rot' **"aux"**: if used, the following number is applied as CPLauxpoints

Out: **SG**: Solid Geometry; VL=[CPA;CPB] **FLW**: Face list of Wall **FLA**: Face list of CPA **FLB**: Face list of CPB

SGof2CPLzbranch

Creates a Solid for an 1:n or n:1 solid branch

In: **CPLA**: Contour A **CPLB**: Contour B **z**: height or [height separation]; default is 0.5 **stype**: "number", "length", "angle", "center"; default is "length" **ctype**: "mina" or "miny"; default is "mina"

Out: **SG**: Solid Geometry of the branch **NFLW**: Wall Facets **NFLA**: Bottom Facets **NFLB**: Cover Facets

SGof2CPLzcorrelate

creates a solid between two vessel structure CPL

In: **CPLA**: Contour A or CVLz with CPLB as CVLz(max(z)) **CPLB**: Contour B **z**: [z n] [h and n interlevels] or list of z values [0 z1 z2 ...]

Out: **SGA**: Solid Geometry **FLW**: Facet of walls wrt to SGA.VL **FLA**: Facet of floor wrt to SGA.VL **FLB**: Facet of cover wrt to SGA.VL **FLC**: Faces missing in A wrt to SGA.VL **FLD**: Faces missing in B wrt to SGA.VL

SGof2CPLzheurist

EXPERIMENT to show the connection of two contours

In: **CPLA**: Contour A **CPLB**: Contour B **z**: height z **stype**: "number", "length", "angle", "center"; default is "length" **ctype**: "mina" or "miny"; default is "mina" **"aux"**: if used ith a number, the function CPLaddauxpoints is applied to both contours **"debug"**: if used some creation steps are shown

Out: **SG**: New Vertex List **NFW**: New Wall List **FLA**: New Bottom FL **FLB**: New TOP FL **CC**: Correspondance list **LA**: Index of Contours A in NVL **LB**: Index of Contours B in NVL

SGof2CVL

returns a solid that closes two

In: **CVL1**: Plane closed contour 1 **CVL2**: Plane closed contour 2

Out: **SG**: Solid Geometry of the Solid SG.VL=[CVL1;CVL2] **FLW**: Facet lit of the wall **FLA**: Facet lit of contour plate 1 **FLB**: Facet lit of contour plate 2

SGof2Dfunc

creates a surface or a solid of a 1D or 2D fnct

In: **ff**: handle to the function; default is @sin **ax**: axis [xmin xmax ymin ymax]; default is [-10 10 -10 10] **res**: step size; default is max(dy,dy)/99; **D**: Socket height; default is '' ==> only surface

Out: **SG**: Solid Geometry

SGof2SGT

Creates a solid between two solids with frames

In: **A**: Solid A **FA**: Framename of A **T1**: Current position frame FA in space; default is SGTget(A,FA) **B**: Solid B **FB**: Framename of A **T2**: Current position frame FB in space; default is SGTget(B,FB) **Rv**: Optional Radius for Solid **NCPL**: Optional Contour for Radius

Out: **SG**: Solid Geometry

SGof2T

Creates a solid geometry between two frames

In: **TA**: Frame A **TB**: Frame B **elo**: elongation of the solid for penetration; default is 0.001 **Rv**: Radius; default is 4 **CPL**: Optional Contour; Default is PLcircle(Rv)

Out: **SG**: Solid Geometry

5.2.3 SG of Meshes [guide]

Functions in this section:

SGofFS — returns a list of solid geometries that are attached to features surfaces

SGofpatch — returns a SG from a patch

SGofSG — converts cell list of solids into a solid struct

SGofSGTsurface — returns the flange surface for a given T matrix frame name

SGofVLdelaunay — returns a convex delaunay triangulation

SGofVLFL — returns a solid geometry from a vertex list and a facet list

SGofVLFLSGc — returns only the Solid Geometry of structs, Solid, Vertex or Facet list

SGofVLFLz — returns a fitting solid based on a surface, i.e. simplified version of SGofSurface

SGofVLtrans — returns a convex solid for a transformed vertex list

Detailed description

SGofFS

returns a list of solid geometries that are attached to features surfaces

In: **SG**: Solid geometry **alp**: alpha angle to calculate the feature surfaces; default is 1 **fsi**: feature surface list; default is all **t**: thickness of the solid; default is 0.5 used in SGofSurface **d**: distance from the surface; default is 0.3 used in SGofSurface **s**: distance from the surface; default is 0.3 used in SGofSurface **"amin"**: minimum surface area to keep a feature surface; default is 0

Out: **SGc**: Cell list of Solids related to fsi **FSc**: Cell list of Vertex list and facet list of the FS related to fsi

SGofpatch

returns a SG from a patch

In: **gh**: handle to patch; default is patchofgca **fuse**: if true; create only one SG; default is false;

Out: **SG**: Solid Geometry; single or cells

SGofSG

converts cell list of solids into a solid struct

In: **SGO**: cell list of structs

Out: **SG**: Solid struct

SGofSGTsurface

returns the flange surface for a given T matrix frame name

In: **SGT**: Solid Geometry **FNam**: Name of Frame **t**: thickness of the solid; default is 0.5 **d**: distance from the surface; default is 0.3 **s**: streching of the border line; default is 0

Out: **SG**: Solid Geometry

SGofVLdelaunay

returns a convex delaunay triangulation

In: **CVL**: Cell list of Contour Vertex Lists {VL1,VL2,VL3}

Out: **SG**: Solid Geometry (VL=[VL1;VL2,...]) **ViL**: Vertex index list (n x 2) wit start and end index **FiL**: Cells with Facet index Lists **FNL**: Facet normal vector list **FoL**: Facet index list for the surface without the enclosed ones

SGofVLFL

returns a solid geometry from a vertex list and a facet list

In: **VL**: Vertex list **FL**: Facet list **FC**: Facet color **sh**: if true; VLFLselect is used; default is false

Out: **SG**: Solid geometry structure

SGofVLFLSGc

returns only the Solid Geometry of structs, Solid, Vertex or Facet list

In: **SGc**: Any kind of SG: A.SG, A{}, A.VL,A.FL

Out: **SG**: Struct only with fields SG.Vl and SG.FL

SGofVLFLz

returns a fitting solid based on a surface, i.e. simplified version of SGofSurface

In: **VL**: Vertex list of a surface (nx3) **FL**: Facet list of a surface (nx3) **z**: thickness of the solid; default is 0.5

Out: **SG**: resulting solid that can be attached to the surface

SGofVLtrans

returns a convex solid for a transformed vertex list

In: **VL**: Vertex list (nx3) or (nx2) **T**: required translation; if scalar the t==>t=[0 0 t]; **TS**: transformation of the solid afterwards; default is []; **comp**: compensates the contraction for phi-x or phi-y; default is false

Out: **SG**: Solid geometry **ViL**: Vertex index list for the surface **FiL**: Facet index list for the base and tranformed contour **FNL**: facet normal list for the base and transformed contour **FoL**: Facet index list for the outside shell of the solid

5.2.4 SG from CVL, Surfaces & Floorplan [guide]

Functions in this section:

DTofPLELz — returns a Delaunay Tetrahedron for a point list and edge list

meshgridofBB — returns a meshgrid for the coordinates of a solid

ms3Dmodel2SG — converts a microsoft 3D model (found in ultimaker cura) into a SG structure

SGBB — Creates the Bounding Box Solid in the solids frame

SGboard — returns the solid geometry of a (raster) board

SGdrawingtemplateofCPL — Creates a 3D printable drawing template - slow but helpful

SGmodelSliderhole — Creates a SG including a key and a keyhole

SGofBB — returns a solid for a bounding box

SGofBBui — interactive mode to create a bounding box of the current camera view and 2 mouse clicks

SGofCVLz — reconstructs a solid from a CVLZ list

SGofCVLzconvexhullfillgap — returns a convex solid based on CVLzconvexhullfillgap

SGofFloorplan — creates from solid geometry of floor plan

SGofgca — returns the solid of the current figure

SGofGeoTIFF — creates a solid geometry from a depth image

SGofgo — returns handle and surface arrays of a graphics objects

SGofsmbFullModelSimulation — returns a solid geometry for an arbitrary time value of a simulation

SGofSurface — returns a fitting solid relativ to a surface

SGoftext — returns a solid for text optional bended

SGplateunder — returns a plate that is completely under the solid

SGsimpleNDItracker — creates a tracker for NDI spheres using a distance list

SGTbaseunderSG — creates a solid as base plated under a given solid

VLFLbolt — returns a 2½ D solid geometry of a bolt

VLFLhollowsnapaxle — generates a slotted hollow snap axle

VLFLinternalgear — Returns the vertex/facet list of a gear with internal involute teeth

VLFLlinkage — returns a 2½ D solid geometry of a linkage

VLFLofBB — return VL and FL for solid related to a bounding box

VLFLofCPLrot — returns a solid geometry from of a radius-z list

VLFLofPLELz — returns a 2½ D solid geometry of a contour

VLFLrotsymRLCL — returns a solid geometry from of a radius-z list and a contour list

VLFLrotsymRLn — returns a solid geometry from of a radius-z list

VLFLshaft — returns VL and FL of a solid axle

VLFLspacer — returns a 2½ D solid geometry of a spacer

Detailed description

DTofPLELz

returns a Delaunay Tetrahedron for a point list and edge list

In: **PL**: Point list for generating a delaunay tessellation in 2D **EL**: Edge list for the boundary constraints in 2D **z**: for the height in z of the solid

Out: **dt**: Delaunay triangulation object

meshgridofBB

returns a meshgrid for the coordinates of a solid

In: **SG**: Solid geometry **nx**: number of steps in x-direction **ny**: number of steps in y-direction **nz**: number of steps in z-direction **thr**: elongation before and after; default is 0

Out: **X**: X-Values of Meshgrid **Y**: Y-Values of Meshgrid **Z**: Z-Values of Meshgrid **VL**: Vertex List of the meshgrid **xx**: steps in x-direction **yy**: steps in y-direction **zz**: steps in z-direction

ms3Dmodel2SG

converts a microsoft 3D model (found in ultimaker cura) into a SG structure

In: **text**: textfile

Out: **SGc**: Cell list of solid geometry structure

SGBB

Creates the Bounding Box Solid in the solids frame

In: **SG**: Solid Geometry **d**: extension of bounding box; default is 0

Out: **SGbb**: Solid Geometry of an Bound Box in the same frame as the solid **v**: volume of the bounding box (product of its three side lengths) **BB**: Solid Geometry of an Bound Box in the same frame as the solid **TL**: transformation frame of the solid used to build the bounding box (TofVL)

SGboard

returns the solid geometry of a (raster) board

In: **x**: dimension in x **y**: dimension in y **h**: dimension in z **nx**: number of holes in x; default is 0 **ny**: number of holes in y; default is 0

Out: **B**: Solid geometry of the boards

SGdrawingtemplateofCPL

Creates a 3D printable drawing template - slow but helpful

In: **CPLU**: Linienzeichnung **pen**: pen diameter **unt**: if true; crossings are united; default is false **wall**: wall thickness; default is 1.5mm **height**: height of template; default is 1.5mm **dmin**: diameter to support crossing points **lnk**: if true, solids are connected; default is false **top**: upper pen diameter; default is width;

Out: **SG**: SG of a drawing template

SGmodelSliderhole

Creates a SG including a key and a keyhole

In: **RA**: Axle Radius **RO**: Outer Radius **lw**: Length of sliding distance **LA**: [L1 L2 L3] describing key length **LO**: [L1 L2 L3] describing key hole length **HO**: Width of a flange in distance RO; default is 2*RO **slot**: default is 0.3 mm (SLS & SLA) **WI**: Inner Wall Thickness

Out: **SGK**: Solid Geometry of Key **SGH**: Solid Geometry of Keyhole **SGP**: Solid Geometry of flange plate **SGT**: Optional Knob on the upper side; separated from SGK **SGB**: Optional Knob on the lower side; separated from SGK

SGofBB

returns a solid for a bounding box

In: **BB**: Result of BBofVL, i.e. [xmin xmax ymin ymax zmin zmax] **d**: enlargement in [x y z]; default is d=[1 1 1] **sep**: if true, create separated (cell-type) bounding boxes; default false

Out: **SG**: Solid box of size of the Bounding Box

SGofBBui

interactive mode to create a bounding box of the current camera view and 2 mouse clicks

In: **h**: handle to gca **viewd**: if true; the box depends on the view direction **plong**: flag to extend the box along the two picked click directions; default is false

Out: **SB**: Solid of SGtransT(SGofBB(dd,0),T) **dd**: bounding box **T**: Camera View Transformation **bb**: smallest BB in view direction

SGofCVLz

reconstructs a solid from a CVLZ list

In: **CVLz**: Stack of slices

Out: **SGC**: Reconstructed Solid Geometry

SGofCVLzconvexhullfillgap

returns a convex solid based on CVLzconvexhullfillgap

In: **CVLz**: Stack of Contours or SG **d**: minimal distance of gaps; default is 1 **slidein**: if true; the solid can be slide in; default is false

Out: **SGC**: Solid Geometry

SGofFloorplan

creates from solid geometry of floor plan

In: **FP**: Floor plan list [wall length, angle [opening]] **a**: optional final turning angle **w**: optional wall thickness; default is 4 **HC**: optional building height; default is 250 **Z**: Thickness of windows **td**: dimension text; default is false **"scale"**: string option 'scale' with a magnification factor applied to the result; default 1 **"STL"**: string option 'STL'; if set, writes the result to multiple STL files

Out: **SG**: Solid Geometry **SGW**: Windows Geometry **SGM**: scaled reference model centered on the floor plan **ST**: Windows Geometry

SGofgca

returns the solid of the current figure

In: **hgca**: optional axis handle

Out: **SG**: Solid Geometry of the patches/facets of the current figure

SGofGeoTIFF

creates a solid geometry from a depth image

In: **I**: Depth Image **L**: Längengrad **B**: Breitengrad **D**: minimum distance of mm as base below zmin

Out: **SGN**: Solid Geometry

SGofgo

returns handle and surface arrays of a graphics objects

In: **go**: graphics object

Out: **SG**: Cell list of Solids **h**: array of handles

SGofsmbFullModelSimulation

returns a solid geometry for an arbitrary time value of a simulation

In: **SGi**: List of Solid Geometries, result of smbFullModelSimulation **Ti**: List of HT Frames, result of smbFullModelSimulation **tm**: List of time points result of smbFullModelSimulation **t**: desired time

Out: **SG**: Solid Geometry **t**: Exact time of simulation

SGofSurface

returns a fitting solid relativ to a surface

In: **OVL**: Vertex list of a surface (nx3) **OFL**: Facet list of a surface (nx3) **t**: thickness of the solid; default is 0.5 **d**: distance from the surface; default is 0.3 **s**: streching of the border line; default is 0 **cn**: false=matlab vertex calc; true=corrected vertex normal; default is false; true is 10 times slower -> VLFLvertexNormal

Out: **SG**: resulting solid that can be attached to the surface **VL1**: VL oved along d **VL2**: VL moved d*ez **FLW**: Facet of list wall [VL1;VL2]

SGoftext

returns a solid for text optional bended

In: **text**: string **sz**: size in [x y z] or just [0 y z] [x 0 z] **t**: thickness of plate does reduce z; default is 0 **radx**: radius in x; default is planar **rady**: radius in y; default is planar **FN**: Fontname; default is 'default'

Out: **SG**: Solid Geometry **CPL**: Contour of the text **sz**: size of the object before bending

SGplateunder

returns a plate that is completely under the solid

In: **SG**: Solid Geometry **h**: [h os r] height oversize and radius; default is [1 0 0] **d**: gap between solid and plate (can be negative); default is -1e-3 **ez**: option vector for postion of the plate; default is '' ~ [0 0 -1] **FN**: optional frame wildcards for SGTcopyasopponent **"add"**: string option 'add'; if set, adds the plate to the solid **"flat"**: string option 'flat'; if set, makes the plate flat (sets gap d to -h+1e-3) **"fill"**: string option 'fill'; if set, fills the plate area

Out: **SGP**: Plate under SG

SGsimpleNDItracker

creates a tracker for NDI spheres using a distance list

In: **d**: distance list such as [40 50 60] **H**: height of the tracker **A**: Geometry of the mounting posts

Out: **SG**: Resulting Solid Geometry **cp**: center point

SGTbaseunderSG

creates a solid as base plated under a given solid

In: **SGR**: Solid or Cell list of SOlids

Out: **SG**: Solid under the first Element of SG **SGC**: Cell list with SG as now first element

VLFLbolt

returns a 2½ D solid geometry of a bolt

In: **R**: Outer Radius **H**: Height

Out: **VL**: Vertex list **FL**: Facet list

VLFLhollowsnapaxle

generates a slotted hollow snap axle

In: **R**: Radius of bore **RF**: Radius of flange **H**: Height of Bore **W**: Width of Snapper wall **HF**: Height of flange and snap **n**: number of facets **m**: number of hooks

Out: **VL**: Vetex list **FL**: Facet list

VLFLinternalgear

Returns the vertex/facet list of a gear with internal involute teeth

VLFLlinkage

returns a 2½ D solid geometry of a linkage

In: **RA**: Outer Radius **RI**: Inner Radius **H**: Height **D**: Distance in X

Out: **VL**: Vertex list **FL**: Facet list **PL**: Point list of the contour **EL**: Contour edge list

VLFLofBB

return VL and FL for solid related to a bounding box

In: **BB**: Bounding box [xmin xmax ymin ymax zmin zmax]

Out: **VL**: Vertex list **FL**: Facet list

VLFLofCPLrot

returns a solid geometry from of a radius-z list

In: **CPL**: Radius-Z List **n**: number of edges of the prismatic cross section **rcorr**: true = increase that R is on circle; default R is incircle

Out: **VL**: Vertex list **FL**: Facet list **CPL**: Resulting contour polygon list

VLFLofPLELz

returns a 2½ D solid geometry of a contour

In: **PL**: Point list in 2D **EL**: Edge list **z**: Height z can be scalar or [zmin zmax]

Out: **VL**: Vertex list **FL**: Facet list **FLW**: Facet list of the wall **FLU**: Facet list of the floor **FLO**: Facet list of the ceiling

VLFLrotsymRLCL

returns a solid geometry from of a radius-z list and a contour list

In: **RL**: Radius list (r z) **CL**: Contour list (x y)

Out: **VL**: Vertex list **FL**: Facet list

VLFLrotsymRLn

returns a solid geometry from of a radius-z list

In: **RL**: Radius-Z List **n**: number of edges of the prismatic cross section **rcorr**: true = increase that R is on circle; default R is incircle

Out: **VL**: Vertex list **FL**: Facet list (FLU;FLO;FLN) **FLN**: Facet list of the wall

VLFLshaft

returns VL and FL of a solid axle

In: **R**: Radius **h**: Height **n**: number of edges

Out: **VL**: Vertex list **FL**: Facet list

VLFLspacer

returns a 2½ D solid geometry of a spacer

In: **RA**: Outer Radius **RI**: Inner Radius **H**: Height

Out: **VL**: Vertex list **FL**: Facet list

5.3 Spatial Transformation [guide]

Functions in this section:

SGarrangeSGC — arranges all parts of a SG container

SGcopyTL — copies a solid using a cell list of frames

SGrotate — returns a solid rotated around the center of the solid

SGrotateservo — turns the SG of a servo motor to an upright position

SGstackn — returns stacked solid geometries

SGtrans — multi transformation modes for a solid geometry

SGtrans0 — moves a solid geometry (or a list) into the origin of the coordinate system

SGtrans1 — moves a solid geometry (or a list) into quadrant 1 of the coordinate system

SGtransP — moves a solid geometry (or a list) relative using a translation vector

SGtransR — rotates a solid geometry (or a list) relative using a rotation matrix

SGtransrelSG — changes position of a solid relative to another solid

SGtransrelT — tranforms a solid relative to known one

SGtransT — transforms a solid geometry or list using a homogenous transformation matrix

SGTrotate — returns a solid rotated around a Frame coordinate System

Detailed description

SGarrangeSGC

arranges all parts of a SG container

In: **C**: Solid Geometry container (see SGCaddSG) **Vol3D**: 3D printing volume (default: [200 200 250]) **Dist3D**: Distance between parts (default: [1 1 1])

Out: **SG**: Solid geometry

SGcopyTL

copies a solid using a cell list of frames

In: **SG**: Solid Geometry to copy **TL**: List of Frames **rel**: spatial relations for Trelation

Out: **SGC**: Solid including all copies of TL

SGrotate

returns a solid rotated around the center of the solid

In: **SG**: Solid Geometry **ax**: rotation axis sequence; 'xyz' or 'x', 'z','e', or 'r' for random with a value, 'e'=centered **val**: values for rotation; same as ax **Cent**: T or "0" or "P", or "R" or "T"

Out: **SGN**: Spatial rotated Solid including the frames **T**: 4x4 transformation matrix used for the rotation center

SGrotateservo

turns the SG of a servo motor to an upright position

In: **SG**: Original Servo Solid Geometry **dh**: relative slice height fraction for the contour cuts; default is 0.04 **rc**: optional rotation axis passed to SGrotate; default is none **rw**: rotation angle for rc; default is pi

Out: **SG**: Rotate Solid **T**: Frame of Shaft **CPL1**: Contour of Shaft **CPL2**: Contour of body

SGstackn

returns stacked solid geometries

In: **SG**: Solid geometry **n**: number **dz**: distance in z; default is 1mm

Out: **SGN**: Stacked solid geometries **h**: height of stack in z

SGtrans

multi transformation modes for a solid geometry

In: **SG**: Solid Geometry **T**: Transformation Value, Vector, or Matrix

Out: **SG1**: Transforms solid geometry **varargout**: optional additional output solids when a list of solids is transformed

SGtrans0

moves a solid geometry (or a list) into the origin of the coordinate system

In: **SG**: Solid geometry (.VL/.FL) or a cell list of SG **z0**: optional z0 value for minimal z value instead of 0.5*(zmax-zmin)

Out: **SG1**: First output solid geometry **varargout**: optional more output arguments

SGtrans1

moves a solid geometry (or a list) into quadrant 1 of the coordinate system

In: **SG**: Solid geometry (.VL/.FL) or a cell list of SG

Out: **SG1**: First output solid geometry **varargout**: optional more output arguments

SGtransP

moves a solid geometry (or a list) relative using a translation vector

In: **SG**: Solid geometry (.VL/.FL) or a cell list of SG **p**: Point as column vector (1x3)

Out: **SG1**: First output solid geometry **varargout**: optional more output arguments

SGtransR

rotates a solid geometry (or a list) relative using a rotation matrix

In: **SG**: Solid geometry (.VL/.FL) or a cell list of SG **R**: Rotation matrix (3x3) **p**: rotation matrix (3x3) or rotation vector (1x3) used to rotate the solid

Out: **SG1**: First output solid geometry **varargout**: optional more output arguments

SGtransrelSG

changes position of a solid relative to another solid

In: **SGA**: Transformed Solid Geometry **SGB**: Reference solid B that the relation refers to (may be empty for absolute placement) **TA**: spatial relations **TB**: gap, Optional parameter after an relation command **TC**: optional matching frame B for the 'ontopT' relation; default is a frame at the top of solid B

Out: **SG**: Transformed Vertices and Frames **T**: Transformation used to transform solid SGA

SGtransrelT

tranforms a solid relative to known one

In: **SG**: Solid Geometry **T0**: Reference Frame Matrix such as rotation axis or string for SGTget **T**: Relative Transformation Frame such as relative rotation; Frame to become the center **dv**: 'r' revolution or 'prismatic' if T is size(1) or size([1 3]) **"debug"**: flag to enable debug output

Out: **SGN**: Spatial transformed solid **TR**: resulting relative transformation matrix T0*T*inv(T0)

SGtransT

transforms a solid geometry or list using a homogenous transformation matrix

In: **SG**: Solid geometry (.VL/.FL) or a cell list of SG **T**: Homogeneous transformation matrix (4x4) **FN**: Optional Name of a Frame that should be used as alignment origin **T0**: optional additional transformation T0 applied to the solid before T; default empty

Out: **SG1**: First output solid geometry **varargout**: optional more output arguments

SGTrotate

returns a solid rotated around a Frame coordinate System

In: **SG**: Solid Geometry **N**: Frame; default is ez; nan means eigenvectors; default is 'B' **ax**: rotation axis sequence; 'xyz' or 'x' or 'z' **val**: values for rotation; same as ax

Out: **SGN**: Spatial rotated Solid includeing the frames

5.4 SG Modification [guide]

Functions in this section:

SGaddfacetcenterpoints — creates additional vertices in the middle of each facet

SGaddsurfpoints — adds gridpoints at surfaces and edges at all surfaces

SGbendcone — bends a solid geometry around a conic segment with defined geometry

SGbending — beates a solid geometry on a cylinder or spherical ellipsoid

SGbreakCorners — breaking the vertices of a solid geometry

SGcapsulate — capsules a solid including alignment

SGchamfer — chamfers the edges of a 2.5D Solid Geometry

SGconvexofSGintersect — returns a solid to intersect or subtract

SGcopypatternXYZ — returns a cell list of xyz copies or a fused solid of SG

SGcopyVL — returns a cell list of copies or a fused Solid of SG an positions of VL

SGdemoulding — Creates demolding chamfers for cast objects

SGextend — just extends the solid in one direction

SGflip — returns a SG with inverted facets - same fnct as 2014-11-30

SGfusedoubledvertices — fuses vertices with similar positions, shortens the VL, but the number of facets remains unchanged.

SGgrow — returns the same solid by with increased dimensions

SGgrowoutsideFS — grows only the outside feature surfaces of a solid

SGgrowVLFLfi — moves facets of an indexed surface along the normal vectors

SGhollowsolid — returns a copy of the solid with thin walls at an optional distance

SGmirrorTxy — mirrors a solid at the xy plane of a HT matrix

SGnegativeform — creates from a solid a negative form with specific features

SGpunchTCPLthrough — punching: shoots a contour through a solid

SGradialCorners — should breaks corners and edges of solids

SGreduceVLFL — returns a solid with reduced

SGremedgepoints — returns a surface model without edge points on straigt lines, but with surface points that are inside

SGremimprint — removes engravings on a defined feature surface

SGremsurfedgepoints — returns a surface model without edge points and surface points that are inside of a surface

SGremsurfpoints — returns a surface model without surface points that are inside of a surface - boundary/edge points are unchanged

SGscale2dim — scales a solid to fit into a box with given dimensions

SGshearing — shearing or bending of a Solid Geometry

SGshrinktofit — reduces the size that the solid fits into a bounding box

SGstentradsym — creates a radial symmetric stent geometry based on a given stretching ratio

SGstrechez — stretches Objects along an axis to fit intervals or positions

SGstretching — cuts a solid geometry in x and stretches in x

SGsweepTproj — sweeps the outer contour of a solid to create a swept solid

SGTdrill — drills a contour hole into a solid geometries frame

SGvaryfrom — simply adds small values in a determinstic way to the vertices of a solid

SGvertexongrid — returns a solid geometry with all vertices set to a grid

VLbreakEdges — returns a vertex list (VL) with broken edges

VLradialEdges — returns a vertex list (VL) with rounded edges

VLradialEdges2T — returns a vertex list (VL) with rounded/broken wrt to start and end frames

Detailed description

SGaddfacetcenterpoints

creates additional vertices in the middle of each facet

In: **SG**: Original solid **n**: recursive number of calls; default is 1 **md**: minimum distance between old and new points; default is 0

Out: **SGN**: New Solid

SGaddsurfpoints

adds gridpoints at surfaces and edges at all surfaces

In: **SG**: Solid geometry **alpha**: angle for surface differentiation; default is 0.01 **ds**: distance on edges; default is **dx**: distance in x; default is 1 **dy**: distance in y; default is dx

Out: **SGN**: Solid geometry with additional points on edges and surfaces

SGbendcone

bends a solid geometry around a conic segment with defined geometry

In: **SG**: Solid geometry to be bent **R1**: Radius of the cone at the xz-plane **R2**: Radius of the cone at a distance L along the +y-axis. Defaults to R1 **L**: Distance between R1 and R2. Defaults to furthest distance of solid geometry along +y-axis **Rz1**: radius in the z-direction. Together with R1, defines an elliptical cross-section **Rz2**: radius in the z-direction. Together with R2, defines an elliptical cross-section

Out: **SG**: deformed solid geometry. Positioned as if the cone was centered on the y-axis

SGbending

beates a solid geometry on a cylinder or spherical ellipsoid

In: **SG**: Solid Geometry **Rx**: Radius Rx **de**: distance/intrusion; default is -0.1 **Ry**: Optional radius Ry; default is 10000 **Rz**: Optional radius Rz; default is Rx **cent**: center solid before bending; default is true

Out: **SGN**: Solid Geometry

SGbreakCorners

breaking the vertices of a solid geometry

In: **SG**: Solid Geometry **vi**: index list or vertex list of SG **k**: new corner distance **sf**: corner reduction factor; default is 1 [*k]

Out: **SGB**: New solid with

SGcapsulate

capsules a solid including alignment

In: **SG**: Solid geometry **d**: wall thickness **align**: 'top','bottom','left','right','front','back' **delta**: shift relative to alignment

Out: **SGX**: Resulting solid

SGchamfer

chamfers the edges of a 2.5D Solid Geometry

In: **SG**: Solid Geometry with just two different z values **ph**: edge/phase size default is 0.3; [r z], [r zs ze] is also valid **ed**: curved edges; default is true **r**: radius for radial edges; default is ph*1.5 **stype**: default is 'angle'; see SGof2CPLzheurist **posit**: 'both', 'start', 'end' or calculated from ph **shape**: PL or 'line' (default), 'circ', see PLchamfer for all values **hole**: "norm", "outw", "hole", see CPLgrow for all values

Out: **SGN**: Solid with chamfered edges

SGconvexofSGintersect

returns a solid to intersect or subtract

In: **SG**: Solid Geometry A to intersect **SGinter**: Intersecting Solid B

Out: **CISG**: Convex intersecting solid that contains alls surfaces of SG intersecting with SGinter

SGcopypatternXYZ

returns a cell list of xyz copies or a fused solid of SG

In: **SG**: Solid Geometry **cp**: number of copies in [x y z] **dd**: distance between the copies **fuse**: if true; SG is struct not cell; default is true

Out: **SG**: cell list of SG

SGcopyVL

returns a cell list of copies or a fused Solid of SG an positions of VL

In: **SG**: Solid Geometry **VL**: Coordinates for creating a copy of SG **fuse**: if true; SG is struct not cell; default is true

Out: **SGC**: cell list of SG

SGdemoulding

Creates demolding chamfers for cast objects

In: **SG**: Solif Geometry **an**: angle for demolding chamfers; default is 2.5 degree **ct**: bordercut; default is -0.1 **hl**: height limitation is used **bu**: if used Kerbkonus bancklok is integrated **maxsiz**: default is 2000 (Shrinks automatically)

Out: **SG**: Modified Solid Geometry

SGextend

just extends the solid in one direction

In: **SG**: Solid Geometry **T**: cutting frame, or 'under (TofSG) or [0 0 -1] (TofSGez) **b**: [extension intrusion] default is [4 -.5]; **typ**: 'fill, sub, unite', default is 'unite' **"debug"**: if used; the function shows the progress

Out: **SGN**: extended Solid **SGS**: Solid supplement

SGflip

returns a SG with inverted facets - same fnct as 2014-11-30

In: **SG**: Solid geometry

Out: **SG**: Solid geometry with inverted facets

SGfusedoubledvertices

fuses vertices with similar positions, shortens the VL, but the number of facets remains unchanged.

In: **SG**: Vertex list **thr**: Threshold für round(VL,thr); default is 1e-17 DO NOT USE DIGITS

Out: **SGN**: Solid with fused vertex list

SGgrow

returns the same solid by with increased dimensions

In: **SG**: Solid Geometry **d**: grow distance **thr**: threshold for SGselect; default ist 1e-12 **met**: method for vertex normal calculation ('ML','TL1','TL2','YS','XX'); default 'YS'

Out: **SGN**: resulting solid geometry

SGgrowoutsideFS

grows only the outside feature surfaces of a solid

In: **SG**: Solid Geometry **t**: Increase/Decrease Value along the vertex normal vector

Out: **SG**: The grown (or shrunk) solid geometry

SGgrowVLFLfi

moves facets of an indexed surface along the normal vectors

In: **VL**: Vertex list **FL**: Facet list **fi**: Facet Indices **dt**: distance to grow

Out: **SG**: New Vertex list

SGhollowsolid

returns a copy of the solid with thin walls at an optional distance

In: **SG**: Solid Geometry **wthk**: Wall thickness **dist**: distance between original and new solid **met**: defining the fnct used for the vertex normals 'ML', 'TL1', '

Out: **NSG**: Copy with thin walls

SGmirrorTxy

mirrors a solid at the xy plane of a HT matrix

In: **SG**: Solid Geometry **T**: HT matrix or ez vector of char **D**: mirror plane offset distance along ez; default is 0

Out: **SGM**: mirrored solid geometry

SGnegativeform

creates from a solid a negative form with specific features

In: **SG**: Solid Geometry **w**: wall distance

Out: **SG**: A simple negative for or several parts of a form

SGpunchTCPLthrough

punching: shoots a contour through a solid

In: **SG**: Solid Geometry **To**: Optional T frame **CPL**: Closed Polygon Line

Out: **C**: Resulting Solid **Ts**: HT MAtrix for cutting in z direction **succ**: flag for success

SGradialCorners

should breaks corners and edges of solids

In: **A**: Solid Geometry **R**: radius; default is 1mm; corner break for negative R values! **vi**: list of corned index (n) or corner vertex list (nx3); default is ''

Out: **A**: Solid Geometry with rounded corners

SGreduceVLFL

returns a solid with reduced

In: **SG**: Original Solid Geometry **f**: reduction factor or number of facets per **sc**: if true and SG is a cell, distribute the reduction proportionally by vertex count; default is false

Out: **SG**: SG with reduced vertices and facets **FL**: Facet List related to original solid geometry (slows down by findVL) **vi**: index of new vertices related to original solid geometry

SGremedgepoints

returns a surface model without edge points on straigt lines, but with surface points that are inside

In: **SG**: Solid Geometry surface model **alpha**: alpha angle for feature surface; default is 0.01 **minf**: minimum facets per face; default is 5; smalles value **"repair"**: if used, the function SGrepairFLsimple is called in the end

Out: **SGN**: New surface model with reduced vertices and faces **FLN**: Reduced Facet list based on the original vertex list **FSi**: Feature surface index list

SGremimprint

removes engravings on a defined feature surface

In: **SG**: Solid Geometry **fil**: Facet index list; defining the surface **alp**: angle threshold for detecting orthogonal imprint facets; default is 0.2

Out: **SGN**: Solid without imprint

SGremsurfedgepoints

returns a surface model without edge points and surface points that are inside of a surface

In: **SG**: Solid Geometry surface model **alpha**: alpha angle for feature surface; default is 0.01 **minf**: minimum facets per face; default is 5; smalles value **"repair"**: flag to enable repair mode

Out: **SGN**: New surface model with reduced vertices and faces **FLN**: Reduced Facet list based on the original vertex list **FSi**: Feature surface index list

SGremsurfpoints

returns a surface model without surface points that are inside of a surface - boundary/edge points are unchanged

In: **SG**: Solid Geometry surface model **alpha**: alpha angle for feature surface; default is 0.01

Out: **SGN**: New surface model with reduced vertices and faces **FLN**: Reduced Facet list based on the original vertex list **FSi**: Feature surface index list

SGscale2dim

scales a solid to fit into a box with given dimensions

In: **SG**: Solid Geometry **dim**: size in [x y z] **scc**: dimensions to consider default is 'xyz' **T**: optional Coordinate system or frame name that defines x y zdefault is ''

Out: **SGN**: New Geometry

SGshearing

shearing or bending of a Solid Geometry

In: **SG**: Solid Geometry **PL**: point list; if scalar the value describes the delta **dd**: direction such as 'xz' or 'yz' **"mesh"**: if the string "mesh" is present, the solid is tetra-meshed first

Out: **SGN**: Sheared SG

SGshrinktofit

reduces the size that the solid fits into a bounding box

In: **SG**: Solid Geometry **dim**: Maximum dimensions **org**: flag for original size handling (org); default is false

Out: **SGN**: Shrinked Solid **k**: shrinking factor

SGstentradsym

creates a radial symmetric stent geometry based on a given stretching ratio

In: **xx**: list of streching ratio; default is [5 3 3 3 3 5] **d**: outer diameter of the stent **m**: thickness of the stent material **mag**: magnification factor

Out: **SG**: Solid Geometry **CPL**: Cutting pattern for Laser **B**: Solid Geometry Approximation of the final shape

SGstrechez

stretches Objects along an axis to fit intervals or positions

In: **SG**: Solid Geometry **ez**: strching axis; default is 'x' **zz**: intervalls ('dist') or positions ('pos') **ss**: scaling factor ('dist') or new positions ('pos') **"mode"**: default is 'dist', use also 'pos' **"debug"**: if used, steps are shown

Out: **SG0**: Streched Geometry **TL**: List of Frames in the Original **TK**: List of Frames in the Streched Object

SGstretching

cuts a solid geometry in x and stretches in x

In: **A**: Solid geometry **dx**: add d to all x-coodinates **xlim**: for all x-coordinates > xlim

Out: **A**: Stretched solid geometry

SGsweepTproj

sweeps the outer contour of a solid to create a swept solid

In: **SG**: Solid Geometry **wt**: angle interval [wmin wmax] **T**: sweeping z-axis or HT Matrix or Frame char **b**: buffer to increase swept volume **pc**: center point for building the sweep frame; default is centerVL(SG.VL)

Out: **SGS**: Swept Solid Geometry

SGTdrill

drills a contour hole into a solid geometries frame

In: **SG**: Solid Geometry **N**: Name of Frame; default is 'B' **h**: depth of hole **CPL**: radius or contour **"debug"**: flag to enable debug output

Out: **SG**: Solid Geometry with hole

SGvaryfrom

simply adds small values in a determinstic way to the vertices of a solid

In: **SG**: Solid to be modified

Out: **SGN**: Solid with slightly adjusted coordinates

SGvertexongrid

returns a solid geometry with all vertices set to a grid

In: **A**: Solid Geometry **grid1**: default is 1e-3 points mapped to the same coordinates **grid2**: default is 0.25*grid 1 grid the last point is mapped to

Out: **B**: Final Solid

VLbreakEdges

returns a vertex list (VL) with broken edges

In: **VL**: Vertex list nx3 **R**: Radius; default is 1

Out: **NVL**: New vertex list

VLradialEdges

returns a vertex list (VL) with rounded edges

In: **VL**: Vertex list nx3 **R**: Radius; default is 10 **ty**: 'rad', 'rmax', 'tan'

Out: **NVL**: New vertex list **Rmin**: Maximum Radius that was possible for the edges;

VLradialEdges2T

returns a vertex list (VL) with rounded/broken wrt to start and end frames

In: **VL**: Vertex list nx3 **R**: Radius; default is 1 **TA**: Rotation Matrix for Start Point **TB**: Rotation Matrix for End Point **ty**: edge type; default is 'radial'

Out: **NVL**: New vertex list

5.5 SG Boolean Operations [guide]

Functions in this section:

crossingfacets2VLFL — Fundamental fnct to return an index list of the facets of solid A intersected by solid B

SGaddrelSG — Spatial arrangement of a solid to another and combining both

SGbool5 — returns the correct result of a boolean operation on two closed elementary solids

SGboolanalyze — plots the two separated surfaces of A and B when crossed

SGboolDEBUG — reads in in the STL Geometries that created problems during SGboolh

SGboolh — returns boolean operations on solid geometries using a heuristic

SGbooltest — Procedure to test different cases using SGbool and CSGsample

SGconcat — Simple Concatenation of Solid Geometries

SGcrosssectionadjustment — A connecting body is designed that transfers and connects two bodies for the opposite cross-sections into each other.

SGintersect — Boolean intersection of Solid Geometries

SGreplace — First version to replace a part of a solid

SGsubtract — Boolean Subtraction of Solid Geometries

SGsupplement — Boolean addition of the solid parts of B that are outside of the convex hull of A

SGsurfacemeltbool — returns a solid which surfaces were added using SGbool

SGunion — Boolean Addition of Solid Geometries

SGunioncomplete — Adds to a solid the part of a nother solid which is outside the convex hull

SGunionreplace — adds a solid after removing the bounding box of the solid

SGxor — Boolean XOR of Solid Geometries

Detailed description

crossingfacets2VLFL

Fundamental fnct to return an index list of the facets of solid A intersected by solid B

In: **VLA**: Vertex list A **FLA**: Facet list A **VLB**: Vertex list B **FLB**: Facet list B **first**: if true; function stops after the first crossing; default is false **pwarn**: if true; functions warns in case of in-plane surfaces

Out: **XIA**: Crossed facets from Solid A **XIL**: Crossing / intersection index matrix **XIB**: Crossed facets from Solid B

SGaddrelSG

Spatial arrangement of a solid to another and combining both

In: **SGA**: Solid A **SGB**: Solid B **rel**: spatial relations **gap**: gap, optional parameter after an relation command

Out: **SG**: Final solid geometry

SGbool5

returns the correct result of a boolean operation on two closed elementary solids

In: **flag**: Boolean operator ('AB+x') **A**: Solid A (VL/FL) **B**: Solid B (VL/FL) **thr**: threshold for grid; default is 1e-5

Out: **SGR**: resulting surface geometry **SGX**: intermediate/auxiliary resulting surface geometry

SGboolanalyze

plots the two separated surfaces of A and B when crossed

In: **A**: Solid A; or number for CSGsample **B**: Solid B; **thr**: Grid resolution threshold for merging close vertices before analysis; default 1e-3

SGboolDEBUG

reads in in the STL Geometries that created problems during SGboolh

In: **cmd**: optional command for checking in further versions

Out: **SGs**: Cell list of 2 Solids {A,B} **erra**: errors of A of SGcheckmeshlab **errb**: errors of B of SGcheckmeshlab

SGboolh

returns boolean operations on solid geometries using a heuristic

In: **flag**: '+', 'x', 'A' ,'B' **A**: Solid A **B**: Solid B **rstr**: rotation axes string for the small rotation applied to B to avoid instabilities; default is 'xyz'

Out: **C**: result of boolean operation **unchng**: true if the operation failed and result equals input A (unchanged)

SGbooltest

Procedure to test different cases using SGbool and CSGsample

In: **tests**: Vector of CSGsample case numbers to run; default is 1:99 (all samples) **Bchar**: boolean operator character; default is 'x' **grid**: grid size to force all solids onto; default is 0 (off)

Out: **X**: Solid A of CSGsample **A**: Solid A of CSGsample **B**: Resulting Solid A-B

SGconcat

Simple Concatenation of Solid Geometries

In: **A**: Solid A **B**: Solid B **relpos**: list of relative positions for SGtransrelSG

Out: **SG**: Concatenation Result of A+B **T**: relative transformation matrix (4x4) applied to B by SGtransrelSG

SGcrosssectionadjustment

A connecting body is designed that transfers and connects two bodies for the opposite cross-sections into each other.

In: **A**: Solid A **B**: Solid B **ezrel**: ez vector of A - spatial relation; default is [0 0 1] **d**: distance between B and A

Out: **X**: Connecting Solid

SGintersect

Boolean intersection of Solid Geometries

In: **A**: Solid A **B**: Solid B **relpos**: list of relative positions for SGtransrelSG **"force"**: 'force' flag to raise a trial error if the intersection fails

Out: **SG**: Boolean Result of A+B

SGreplace

First version to replace a part of a solid

In: **SG**: Solid Geometry **A**: Volume of SG to be replaced; default is convex hull of B **B**: Solid Geometry to fill the replaced volume **ol**: oversize to cut; default is 0.05

Out: **SG**: Solid with B instead of A

SGsubtract

Boolean Subtraction of Solid Geometries

In: **A**: Solid A **B**: Solid B **relpos**: list of relative positions for SGtransrelSG **"force"**: 'force' flag to raise a trial error if the subtraction fails **"2nd"**: not used anymore - was to create B-A instead of A-B **"caller"**: 'caller' name string used in warning messages

Out: **SG**: Boolean Result of A-B **T**: HT matrix from SGtransrelSG positioning B relative to A; default is eye(4)

SGsupplement

Boolean addition of the solid parts of B that are outside of the convex hull of A

In: **SGA**: Solid A **SGB**: Solid B **relpos**: list of relative positions for SGtransrelSG

Out: **SG**: Boolean Result of A+(B-convechullof(A))

SGsurfacemeltbool

returns a solid which surfaces were added using SGbool

In: **SG**: Original solid. All surfaces must overlap **si**: indices of surfaces to melt; default is all; **Amin**: Minimal Area removal; parameter for SGboolTL; default 0.01 **grw**: Minimal grow factor; parameter for SGboolTL; default 1e-5

Out: **A**: Melted Object consisting of melted solids

SGunion

Boolean Addition of Solid Geometries

In: **A**: Solid A **B**: Solid B **relpos**: list of relative positions for SGtransrelSG

Out: **SG**: Boolean Result of A+B **T**: HT matrix from SGtransrelSG positioning B relative to A; default is eye(4)

SGunioncomplete

Adds to a solid the part of a nother solid which is outside the convex hull

In: **SGA**: Solid to be completed **SGB**: Solid whos parts outside of SGA should be added to SGA **spatialrel**: spatial relation such for SGransrelSG **"debug"**: if ised the creation is shown.

Out: **SGN**: New combined Solid

SGunionreplace

adds a solid after removing the bounding box of the solid

In: **A**: Solid A **B**: Solid to unite after subtracting the bound box **relparam**: relative arrangement parameter **"concat"**: option overlap parameter default is [0.1 0.1 0.1] **"ol"**: overlap parameter for the bounding-box subtraction; default is [0.1 0.1 0.1]

Out: **SG**: Modified Solid A

SGxor

Boolean XOR of Solid Geometries

In: **A**: Solid A **B**: Solid B **relpos**: list of relative positions for SGtransrelSG

Out: **SG**: Boolean Result of A+B

5.6 SG Cutting & Slicing [guide]

5.6.1 SG Cutting [guide]

Functions in this section:

SGcut — Cuts a solid geometry into 2 parts at a defined z-plane

SGcut2 — Cuts a solid geometry by inserting vertices and facets

SGcut2T — cuts out the SG that is below frame T1 and frame T2

SGcutBB — cuts out a bounding box from a solid

SGcutend — returns a cut object that is cut related to the BB

SGcutrecursivly — cuts a part recursively into pieces until it fits a maximum dimensions

SGcutT — cuts an object below a arbitrary plane using SGcut relative to the center of floor of bb

SGcutTcrossblade — removes a part out of a solid

SGcutTcylindricblade — uses a cylindric blade to create joint as part of a solid

SGcutTinsertpins — separates a solid similar to SGcutT but inserts connection points

SGpuzzlecut3D — creates from a large solid geometry smaller printable parts

Detailed description

SGcut

Cuts a solid geometry into 2 parts at a defined z-plane

In: **SG**: Solid Geometry (VL,FL) **z**: z value for slicing or interval [z1 z2] or character such as 'x', 'y', 'z'

Out: **SGA**: Solid below cutting plane (and above z2) (gray) **SGB**: Solid above cutting plane (and in between z2) (magenta)

SGcut2

Cuts a solid geometry by inserting vertices and facets

In: **SG**: Solid Geometry (VL,FL) **z**: z value for slicing

Out: **[SGA]**: Solid below cutting plane (and above z2)

SGcut2T

cuts out the SG that is below frame T1 and frame T2

In: **SG**: Solid Geometry **T1**: Frame T1 **T2**: Frame T2; if numerical scalar; T1 and T2 are symmetrically shifted from T2 along ez axis **ct**: not used yet;

Out: **SGM**: Middle Solid (Green) **SGO1**: Above frame T1 (Gray) **SGO2**: Above frame T2 (Magenta)

SGcutBB

cuts out a bounding box from a solid

In: **SG**: Solid geometry **ax**: axis, such as Bounding box **de**: delta to add on each side; default is 0; **TB**: use if BB ax was designed relative to a frame for instance with BBofSGcutT **"bool"**: if all the parts should be intersected wwith

Out: **SGN**: resulting solid

SGcutend

returns a cut object that is cut related to the BB

In: **SG**: SG if empty, SGofgca **ez**: string as 'top,'bottom','behind','left',right','infront', 'x+', 'x-, 'y+', 'y-', 'z+', 'z-', **d**: distance from outside border **expl**: explosition distance; Parts are separated; default is 0;

Out: **A**: remaining solid (magenta) **B**: removed solid (white) **T**: Transformation matrix used

SGcutrecursivly

cuts a part recursively into pieces until it fits a maximum dimensions

In: **SG**: Solid Geometry **lim**: cutting dimensions; default is [inf inf inf]

Out: **SG**: Cell list of Solid Geometries

SGcutT

cuts an object below a arbitrary plane using SGcut relative to the center of floor of bb

In: **SG**: Solid Geometry or empty for SGofgca **Ts**: cutting plane to cut under; or empty for interactive select **centxy**: if true, center the SG on xy-plane; default=false | new in 4.1 **expl**: explosion distance of the solids; **ct**: 'a'xial, 'c'oronal, 's'agital, default is '-a'

Out: **SGU**: New lower geometry (Magenta) **SGO**: New upper geometry (Gray) **Ts**: T Surface - HT Matrix of the cut if T was selected interactively **Ufi**: Facet index for the crossing facet of the lower part **Ofi**: Facet index for the crossing facet of the upper part

SGcutTcrossblade

removes a part out of a solid

In: **SG**: Solid Geometry **T**: cutting frame **d**: slot in size **ct**: orientation 'default' is '-a',; 'c', 's', 'a'

Out: **A**: Cutted Solid (green) **B**: Blade solid (red) **Tc**: Used cutting Frame

SGcutTcylindricblade

uses a cylindric blade to create joint as part of a solid

In: **SG**: Solid geometry **T**: Cutting plane x/y or empty for interactive selection **ct**: turn of cutting plane 'a', 's', 'c' **R**: [Radius blade-tickness starting angle]; default is [7 0.3 0] **H**: Height of cylindric blades and plate blade default is [10 0.08]

Out: **SGN**: {Lower Part (Green) and Upper Part (Red)}

SGcutTinsertpins

separates a solid similar to SGcutT but inserts connection points

In: **SG**: Solid Geometry **T**: Cutting frame; Z-Achsis spearates **rh**: radius and height of pins **sl**: default is slfit('c'); to be compatible with 5.3 or ealier (Oct 2023), use 0 here **"STL"**: 'STL' flag to write separated STL files of the parts

Out: **SGB**: Solid with Pins **SGA**: Solid with Holes **SGP**: Just the Pins **CPL**: cutting Contour **PL**: Point list for Pins **T**: HT matrix to transform PL to VL

SGpuzzlecut3D

creates from a large solid geometry smaller printable parts

In: **SG**: Solid Geometry **bsiz**: [maxx maxy maxz]; if values are <=1 it is not the size but divider

Out: **SGs**: List of puzzle elements

5.6.2 SG Slicing [guide]

Functions in this section:

CPLofSGslice — returns the CPL of a sliced SG

CPLofSGslice2 — return slices of a separated SG

CPLofSGslice3 — returns CPL and CVL for different sliced heights and directions

CPLofSGsliceauto — returns the slice with a significant z distance

CPLsingleSolidSlicer — Sliced CPL of a single closed surface

CVLzaddauxpoints — adds points to CVL z sorted contours

CVLzconvexhullfillgap — returns a CVLz stack with ONE convexhullcontour in each layer

CVLzcut — cuts out defined z levels

CVLzflipud — flips z values for a CVLz

CVLzisccwcorrected — corrects cw/ccw orientation and removes doubled start/end points

CVLzofSG — simple and reliable fnct

CVLzofSGsliceauto — returns the slices with a significant z distance

CVLzofSGslices — returns a stack of slices as CVL (CPL including z) of a SG

CVLzoutlinecontourfillgap — returns a CVLz stack with ONE convexhullcontour in each layer

CVLzremovearea — removes areas with a small distance to the opposite wall

CVLzreplace — deletes or replaces a contour in a CVL stack

CVLzseparate — returns a sliced CVL/CPL for a given z value

FILofrangez — returns an facet index list for facets crossing a plane

SGslicer — returns the delaunayTriangulation of the sliced plane

VLFLofrangez — returns a reduced facet list and resorted vertex list for all facets of a specified plane

Detailed description

CPLofSGslice

returns the CPL of a sliced SG

In: **SG**: Solid Geometry **z**: z value for slicing

Out: **CPL**: Closed Polygon List **warn**: result of SGslicer

CPLofSGslice2

return slices of a separated SG

In: **SG**: Solid geometry; incl. separated objects **z**: z-plane to slice

Out: **CPL**: CPL of the sliced plane

CPLofSGslice3

returns CPL and CVL for different sliced heights and directions

In: **SG**: Solid Geometry **z**: either a z value or a cell list of values

Out: **CPL**: Closed polygon list (n x 2) separated with NaN NaN **CVL**: Closed polygon vertex list (n x 2) separated with NaN NaN NaN

CPLofSGsliceauto

returns the slice with a significant z distance

In: **SG**: Solid Geometry **dz**: minimal z change; default is soBB(SG)/50 **sh**: Abstand zu einem Punkt nach oben und unten default is 0.001

Out: **CPL**: Closed Polygon Line **CVL**: Closed Vertex List of all z values **zz**: z value of the selected slice

CPLsingleSolidSlicer

Sliced CPL of a single closed surface

In: **VL**: Vertex list **FL**: Facet list **slz**: z value for slizing

Out: **CPL**: Closed Polygon List

CVLzaddauxpoints

adds points to CVL z sorted contours

In: **CVLz**: CVL z sorted contour **d**: forced maximum distance of points; default is smin

Out: **CVLz**: modified an corrected CPLz

CVLzconvexhullfillgap

returns a CVLz stack with ONE convexhullcontour in each layer

In: **CVLz**: Stack of CPL sort by z values **d**: minimal distance for slots inside each CPL **slidein**: if true, CVLZ are united with z value; default is false

Out: **CVLz**: Stack with Convex hull contours **zz**: vector of z values of the layers (from CVLseparatez)

CVLzcut

cuts out defined z levels

In: **CVL**: Original CVLz **z**: [z1] for z1..top or [z1 z2] for z1..z2

Out: **CVLA**: Upper Part or niddle part **CVLB**: Lower Part and Upper part

CVLzflipud

flips z values for a CVLz

In: **CVLz**: Original CVLz

Out: **CVLn**: CVLZ up and down

CVLzisccwcorrected

corrects cw/ccw orientation and removes doubled start/end points

In: **CVLz**: List of planar CPLs with **keep**: default is true **simplify**: default is true | NEVER CHANGE TO FALSE!

Out: **CVLz**: New CVLz in right order and

CVLzofSG

simple and reliable fnct

In: **SG**: Solid Geometry **n**: number of slices or slot distance **d**: optional slot distance for rounding/reducing contour points; default is 0 (off)

Out: **CVLz**: Stack of slices **vz**: z values

CVLzofSGsliceauto

returns the slices with a significant z distance

In: **SG**: Solid Geometry **dz**: minimal z change; default is soBB(SG)/50

Out: **CVL**: Closed Vertex List of all z values

CVLzofSGslices

returns a stack of slices as CVL (CPL including z) of a SG

In: **SG**: Solid Geometry **n**: number of slices; default is 10; if n is real, n is the slice distance **l**: optional limit for contour order; default is '' **thre**: reduction at top and bottom; default is 1e-2

Out: **CVL**: CVLz of the slices including the relevant c coordinate

CVLzoutlinecontourfillgap

returns a CVLz stack with ONE convexhullcontour in each layer

In: **CVLz**: Stack of CPL sort by z values **d**: minimal distance for slots inside each CPL **slidein**: if true, CVLZ are united with z value; default is false

Out: **CVLz**: Stack with Convex hull contours **zz**: y valuelist of CVLz

CVLzremovearea

removes areas with a small distance to the opposite wall

In: **CVL**: minimal required distance to survive as contour **res**: minimal required distance to survive as contour

Out: **CVLN**: CVL with removed areas the will creates problems during printing or SG reconstruction

CVLzreplace

deletes or replaces a contour in a CVL stack

In: **CVLz**: Stack of CVLz **z**: desired z to be replaced **CVLi**: CPL or CVLz to be inserted

Out: **CVL**: Resulting CVLz

CVLzseparate

returns a sliced CVL/CPL for a given z value

In: **CVLz**: Slices CVL **i**: simple index in zz list

Out: **CVL**: just the contour with the nth z value

FILofrangez

returns an facet index list for facets crossing a plane

In: **VL**: Vertex list **FL**: Facet list **z1**: z-plane for crossing **z2**: upper z-plane for crossing; default is z1

Out: **FIL**: +1 = above plane, -1 under plane, 0=crosses plane

SGslicer

returns the delaunayTriangulation of the sliced plane

In: **SG**: Solid Geometry (VL,FL) **z**: z value for slicing

Out: **TR2**: Delaunay Triangulation of the cutting plane **EL**: Border Edge List of TR2 (TR2.Constraints) **SIL**: Surface Index List (Index of separated surface areas) **FIL**: Facet index list of the cutted facets -1=below, +1= up, 0=cutted **FLO**: Facet list upper related to [SG.VL; TR2.Points] **FLU**: Facet list lower related to [SG.VL; TR2.Points] **warn**: logical flag set true if a warning occurred during slicing

VLFLofrangez

returns a reduced facet list and resorted vertex list for all facets of a specified plane

In: **VL**: Original vertex list **FL**: Original facet list **z1**: z coordinate of the slice **z2**: second z-coordinate, default is z1 **sname**: optional name for generating a printed protocol

Out: **RVL**: Reduced and sorted vertex list (z) **RFL**: Reduced and sorted facet list (index) **FLi**: index list of the reduced facets

5.7 SG Connection & Joints [guide]

Functions in this section:

SG2SGspherical — Takes a solid and projekts is orthogonal to a sphere to create spherical linkages for instance

SGbearing2012 — returns a bearing for a joint

SGbladeofCVL — returns a solid for an almost planar CVL the closing facets

SGbladeofPL — returns a solid with a shape of a cutting blade in z direction

SGboreholemountingpost — returns a base plate and housing for mounting printed circuit bards

SGconnect — returns a profiled contour rod between the centers of two solids

SGconnect2FS — returns the connecting straight solid structure between two surfaces of two solids

SGconnect2T — returns a Bezier contour based connection between two solids geoemtries

SGfindjointDoFofSG — try to find the DoF between two solids

SGinsertCut — returns a solid with an inserted cut

SGinsertJoint — creates joint structures inside of a solid geometry

SGinsertPeghole — returns a solid with an inserted peg and hole connector

SGinsertTslice — fnct that inserts required slice points into a surface before a stretch

SGlink — returns a structural link between two solid geometries

SGlinkstraight — returns a solid link between TA and TB

SGlinkwall — returns a solid geometry of a plain wall and attaches it to a solid geometry

SGplugofSGT — creates a plug for an SGT

SGpunchTCPLfirstwall — punchs from a frame or from an outside poit a hole with a CPL into the first wall

SGspherelink — returns a spherical link

SGtube — Simplified version of SGplugintube

Detailed description

SG2SGspherical

Takes a solid and projekts is orthogonal to a sphere to create spherical linkages for instance

In: **SG**: Solid Geometry (SG.VL, SG.FL) **Radii**: [Rx Ry Rz]; default is [100 100 100] **TCP**: Frame of the sphere to projekt on

Out: **SGN**: Transformed Solid

SGbearing2012

returns a bearing for a joint

In: **l**: length of the joint **Ro**: Radius of the outer ring **no**: number of facets of the outer ring **Ri**: Radius of the inner ring **Rd**: Radius of the drill boring **ni**: number of facets of the boring **wall**: change of radius **nf**: number of facet of the bearing

Out: **SGA**: Solid of the inner part **SGB**: Solid of the outer part **FAc**: top plane facet list of the inner part **FAf**: bottom plane facet list of the inner part **FBc**: top plane facet list of the outer part **FBf**: bottom plane facet list of the outer part

SGbladeofCVL

returns a solid for an almost planar CVL the closing facets

In: **CVL**: Closed vertex list; open polygons **sl**: Blade thickness; default=0.3 **ad**: size growing; default=0;

Out: **SG**: Solid Geometry of a blade

SGbladeofPL

returns a solid with a shape of a cutting blade in z direction

In: **sl**: thickness of the blade **PLX**: z or [x z] or PL with xz values; **PLY**: depth in Y **T**: Transformation Matrix **SGB**: Final Crossing Solid to limit the outer shape

Out: **SGX**: Cutting Blade Slid Geometry

SGboreholemountingpost

returns a base plate and housing for mounting printed circuit bards

In: **SG**: SG of the PCP in xy **d**: minimum diameter to detect bore holes; default is 2.0 **t**: thickness of contours and height of base plate; default is 1.0 **h**: height **stamp**:

Out: **XX**: Mounting Plate with fastener pods **SGM**: Solid Plate under mounting plate **SGP**: Contour of the PCB **PCB**: Contour of the PCB **SG**: the input solid, possibly rotated to align bore holes with the view

SGconnect

returns a profiled contour rod between the centers of two solids

In: **SGA**: Solid geometry A **SGB**: Solid geometry b **CL**: contour list of the rod's profile **pa**: starting point at A **pb**: starting point at B

Out: **SGW**: Solid geometry of the connecting rod

SGconnect2FS

returns the connecting straight solid structure between two surfaces of two solids

In: **SGA**: Solid A with VL and FL **SGB**: Solid B with VL and FL **FLAi**: Feature Surface index list at Solid A, or Frame Name or HT-Matrix **FLBi**: Feature Surface index list at Solid B or Frame Name or HT-Matrix **rf**:

Out: **SGC**: Combined Solid Geometry without Frames and **FLAC**: Facet list of A without FLAi related to SGC.VL **FLBC**: Facet list of B without FLBi related to SGC.VL **FLNC**: Facet list of new wall related to SGC.VL **SGN**: Solid Geometry only of the open wall

SGconnect2T

returns a Bezier contour based connection between two solids geoemtries

In: **FTA**: Solid geometry A **FTB**: Solid geometry B **TA**: Starting frame at A **TB**: Ending frame at B

Out: **FTW**: connecting solid geometry

SGfindjointDoFofSG

try to find the DoF between two solids

In: **A**: Solid A **B**: Solid B **slim**: range limit for search; default is 10; inf works too. **stol**: tolerance wrt to minimal point; default is .1; 1 = 100 Percent of minimal distance is accepted

Out: **DOF**: List [Avi Bvi dist nvec] of minimal distance point pairs **T**: Frame to define the axis of DoF

SGinsertCut

returns a solid with an inserted cut

In: **SG**: Solid Geometry **T**: Transformation for cut; default is eye(4) **sl**: cut slot size; default is 0.3 **sx**: transversal separation of cut; default is 0; **ad**: addendum ; default is 1 mm; tbi: if negative add is added to bb of SG

Out: **SG**: Solid Geometry, retesselated Amin=0.01 **T**: exact cutting point matrix **d**: distance values relative to T

SGinsertJoint

creates joint structures inside of a solid geometry

In: **SG**: Solid Geometry **T**: Frame for the Solid Geometry **typ**: Type of joint **x**: parameter set

Out: **SGR**: Solid geometry with the inserted joint

SGinsertPeghole

returns a solid with an inserted peg and hole connector

In: **SG**: Solid Geometry **T**: Separation plane **sl**: slot size peg and hole **sx**: diameter of slot size; or CPL of peg **ad**: doubled length of peg

Out: **[SG]**: Solid Geometry with a peg and a hole

SGinsertTslice

fnct that inserts required slice points into a surface before a stretch

In: **SG**: Solid Geometry **T**: Frame to cut by xy-plane **ct**: cutting mode if T is empty; see TofPcam

Out: **SG**: Solid consisting of SGU and SGO (unified vertices by SGshort) **SGU**: SGU lower part of wrt to frame T (Magenta) **SGO**: SGO upper part of wrt to frame T (Gray)

SGlink

returns a structural link between two solid geometries

In: **SGIA**: Solid A **SGIB**: Solid B **ctype**: HT Matrix for A **TA**: HT Matrix for B; Typical A+n*ez; **TB**: Contour list; default is a Radius 5 **CL**: Edge list for Contour; default is a wallsize 2 **ELU**: number of facets; default is 4; **nf**: number of facets of the shape contour; default is 4

Out: **SGlink**: Solid geometry of the link **nv**: Number of edges of the shape contour

SGlinkstraight

returns a solid link between TA and TB

In: **SGA**: Solid A **SGB**: Solid B **TA**: HT Matrix for A **TB**: HT Matrix for B; Typical A+n*ez; **CL**: Contour list; default is a Radius 5 **ELU**: Edge list for Contour; default is a wallsize 2 **nf**: number of facets; default is 4;

Out: **SGlink**: Vertex list **nv**: Facet list

SGlinkwall

returns a solid geometry of a plain wall and attaches it to a solid geometry

In: **SGA**: Solid geometry that should get a wall **n1**: ez vector points into the wall direction **dist**: distance, the wall should be created **sz**: depth of the wall **sx**: size in x **sy**: size in y

Out: **SGW**: Solid geometry of wall and linking structure

SGplugofSGT

creates a plug for an SGT

In: **SG**: Solid geometry with frames **N**: Name of frame, the plug is for **h**: Optional height of plug; default is 5mm **ms**: Diameter of a magnet; default is 10mm **tol**: Tolerance for all values; default is 0.2mm (smaller)

Out: **SGP**: Solid Geometry of the Plug

SGpunchTCPLfirstwall

punchs from a frame or from an outside poit a hole with a CPL into the first wall

In: **SG**: Solid Geometry **To**: Frame to punch; default is interactive Mode **CPL**: CPL or radius **olap**: overlap; default is radius or 1e-2 **plan**: if true, the surface is used to punchl default is false

Out: **SG**: Solid Geometry **p**: punch point **ez**: punch direction

SGspherelink

returns a spherical link

In: **Lz**: Length in Z **R1**: Upper radial values [Rx Ry Rz]; default is L/4 **R2**: lower radial values [Rx Ry Rz]; default is R1 **olap**: overlap of the three solids; default is 1e-3;

Out: **SGC**: Final solid

SGtube

Simplified version of SGplugintube

In: **RR**: [Inner Radius x Wall size]; default is 8 1.5 **LL**: Length of tube; default is 100 **sl**: slot size

Out: **SG**: Solid Geometry

5.8 SG Queries & Properties [guide]

Functions in this section:

BBofSG — returns the bounding box of a solid geometry

incenterVLFL — returns the center of a surface and of all facets

isonVLFLedge — returns the triangles a vertex is on edge

isplanarVLFL — return whether a point cloud or surface is planar

isSG — return whether an object is a Solid Geometry

nofVLFL — returns the number of facets with an area larger than 0.1 mm^2 and the facet list

SGanalyzeGroupParts — returns parts of penetrating solids group colored

SGanalyzePenetration — returns the structure of additive designed SG

SGarea — returns the area of the surfaces (VL/PL)

SGcolorcopy — copies facet colors of a reference solid to a requreste

SGcolorfaces — colors the faces of a solid

SGcolorseparation — returns a cell list or SG separated by colors

SGconvexHull — returns the convex hulls of a solid geometry

SGconvexHullsliced — creates a high resultion fittin convex contour in xy and a sliced on in z

SGconvexhullVL — creates the convex hull of a cloud of vertices

SGconvexHullXYZ — creates a convex solid by the intersection of the three projections in X Y Z

SGfacenum — returns the number of facets of a solid geometry

SGintersectFacetPoints — calculates a list of crossing facet pairs and crossing points of 2 solids

SGisconvex — returns whether a solid is convex

SGisempty — returns whether a solid is empty / has not faces or not

SGisidentical — try to compare two solids to find equality as fast as possible

SGlevelheight — changes the height of a solid depending on the x/y coordinates

SGlim — compresses the vertex coordinates outside a limit without changing the VL or FL

SGnearestpointsofSG — returns the nearest vertices and facets of two solid geometries

SGplotcell — plot a cell list solid in different colors

SGuniquesurfaces — returns unique surfaces from a solid geometry

SGweight — returns the estimated weight of a solid

TR3neighborsAngle — returns the angle difference between a facet and its neighbors - required for Feature Surfaces

VLcrossingSG — returns crossing points of two solid geometries

VLFLarea — returns the area of the surfaces (VL/PL)

VLFLfaceAngles — returns the three angles for the facets of a solid

VLFLinpolyhedron — returns the isInside Flag for a SG and a VL

Detailed description

BBofSG

returns the bounding box of a solid geometry

In: **SG**: Solid Geometry **sep**: separation of cells; default is false **kin**: Frame for optimized bounding boxes; default is 'B'

Out: **BB**: Bounding Box **SGBB**: optional Solid Geometry of the bounding box

incenterVLFL

returns the center of a surface and of all facets

In: **VL**: Vertex list [nx2] or [nx3] **FL**: Facet list **weight**: if true, weight facet centers by facet area; default false

Out: **CVL**: Center Vertex List **SCV**: Surface Center Vertex

isonVLFLedge

returns the triangles a vertex is on edge

In: **VL**: Vertex list of solid **FL**: Face list of solid **PL**: Point to check or Point List

Out: **NN**: Result [n x 4]

isplanarVLFL

return whether a point cloud or surface is planar

In: **VL**: Vertex List **FL**: optional Facet List to accelerate **thre**: Optional threshold; default is 1e-2 ~ 1%

Out: **r**: true if planar, i.e. m<thre **m**: maximal z value / size(BB)

isSG

return whether an object is a Solid Geometry

In: **SG**: Object

Out: **res**: result

nofVLFL

returns the number of facets with an area larger than 0.1 mm^2 and the facet list

In: **VL**: Vertex list **FL**: Facet list **amin**: threshold; default is 0.1 mm^2

Out: **nf**: number of facets >= amin **Ai**: Index list of facets >

SGanalyzeGroupParts

returns parts of penetrating solids group colored

In: **SG**: Solid Geometry of additive designed solids **gmethod**: method 'vertex','cross','solid'; default is solid

Out: **SGc**: List of solids sorted by non overlapping parts

SGanalyzePenetration

returns the structure of additive designed SG

In: **SG**: Solid Geometry **m**: method 'vertex','cross','solid', 'box'; default is cross

Out: **SGk**: Cell list of Solid Geoemtries **PL**: List containing set of penetrating parts **CL**: Crossing List

SGarea

returns the area of the surfaces (VL/PL)

In: **SG**: Solid Geometry

Out: **ASUM**: Area of the surfaces **A**: Area list for facet list

SGcolorcopy

copies facet colors of a reference solid to a requreste

In: **SGA**: Solid to be colored **SGB**: Colored reference solid

Out: **SGA**: Solid with copied colors

SGcolorfaces

colors the faces of a solid

In: **SG**: Solid Geometry **col**: color **ind**: index for coloring **alpha**: alpha value

Out: **SG**: Solid Geometry

SGcolorseparation

returns a cell list or SG separated by colors

In: **SG**: struct with fields; VL, FL, FC **sel**: selector; list of chars or a list of rgb values [0..1 0..1 0..1]

Out: **SGC**: cell list or colored Solid Geometry **used_colors**: table of used colors [R G B]

SGconvexHull

returns the convex hulls of a solid geometry

In: **SG**: Solid Geometry **sep**: false=one; true=separated solids

Out: **SGC**: Convex hull(s) of the Solid **TR3**: Delaunay Triangulation

SGconvexHullsliced

creates a high resultion fittin convex contour in xy and a sliced on in z

In: **SG**: Solid Geometry **ra**: z resolution and auxiliary point distance; default is [s/50 1] **Dcl**: Diameter of small holes in the contour to be closed; default is 3 for M3; **"debug"**: if used the creation process is shown

Out: **SGN**: Convex solid [FL=[FLA;FLB;FLW] **FLW**: facet list of wall **FLA**: facet list of floor **FLB**: facet list of cover

SGconvexhullVL

creates the convex hull of a cloud of vertices

In: **VL**: Vertex List or SG **d**: grow offset applied to the hull via SGgrow; default is 0

Out: **SG**: Solid Geometry

SGconvexHullXYZ

creates a convex solid by the intersection of the three projections in X Y Z

In: **SGP**: Original Solid **"debug"**: if used; the individual processing steps are shown

Out: **SGN**: intersection of the three projections in X Y Z

SGfacenum

returns the number of facets of a solid geometry

In: **SG**: Solid Geometry, can be nested

Out: **nF**: number of facets

SGintersectFacetPoints

calculates a list of crossing facet pairs and crossing points of 2 solids

In: **A**: Solid A **B**: Solid B

Out: **NPL**: New point list

SGisconvex

returns whether a solid is convex

In: **SG**: Solid Geometry

Out: **c**: true if convex, false if not

SGisempty

returns whether a solid is empty / has not faces or not

In: **SG**: Solid geometry

Out: **ie**: true if there are no surfaces

SGisidentical

try to compare two solids to find equality as fast as possible

In: **SGA**: Solid A; or a file name for a solid **SGB**: Solid B; or a file name for a solid **tol**: tolerance for area difference comparison; default is 1e-2

Out: **fdiff**: true if identical or false if not **TBA**: HT matrix to transfer B into the coordinate system of A **TAB**: HT matrix to transfer A into the coordinate system of B **fadi**: sums of size-sorted facet areas of A and B

SGlevelheight

changes the height of a solid depending on the x/y coordinates

In: **SG**: Solid Geometry **mmz**: [minvalue maxvalue maxheight rampsize]; default is [-inf inf nan] nan == no change **ax**: height-axis; dimension-axis]; default is [z y]

Out: **SG**: modified Solid

SGlim

compresses the vertex coordinates outside a limit without changing the VL or FL

In: **VL**: Vertex list **dim**: dimension 1 2 3 ord 'x' 'y' 'z' **lim**: limit max or [min max] **rat**: compression ratio default is 1000;

Out: **VLN**: Vertex list limited/compressed to the given dimension range

SGnearestpointsofSG

returns the nearest vertices and facets of two solid geometries

In: **A**: Solid A **B**: Solid B **slim**: range limit for search; default is 10; inf works too. **stol**: tolerance wrt to minimal point; default is .1; 1 = 100 Percent of minimal distance is accepted

Out: **DILM**: List [Avi Bvi dist nvec] of minimal distance point pairs **FLAi**: Facets a A with a least one point in DIL(:,1) **FLBi**: Facets a A with a least one point in DIL(:,2) **DIL**: Full List [Avi Bvi dist nvec] of minimal distance point pairs

SGplotcell

plot a cell list solid in different colors

In: **SG**: cell list of solid **c**: color **w**: width

Out: **h**: handle to plot

SGuniquesurfaces

returns unique surfaces from a solid geometry

In: **SG**: Solid Geometry **isautosep**: flag to auto-separate solid into surfaces first via SGsurfaces; default is false

Out: **SGC**: Cell list of Solid Geometries **UL**: List of elements with the same geometry **T**: List of 4x4 HT matrices for spatial transformation

SGweight

returns the estimated weight of a solid

In: **SG**: Solid Geometry **sw**: specific weight; default is 1.15g/cm^3 (Nylon) **res**: resolution; default is 2x2x2 mm^3

Out: **W**: Estimated weight in gramm **VM**: Logical Voxel model **nd**: dimensions of VM, same as size(VM) **xl**: x coordinate grid vector of the voxel model **yl**: y coordinate grid vector of the voxel model **zl**: z coordinate grid vector of the voxel model

TR3neighborsAngle

returns the angle difference between a facet and its neighbors - required for Feature Surfaces

In: **TR3**: Triangulation or SG

Out: **NAL**: Facet Normal Angle List wrt to NBL of each facet (3 angles) **NBL**: Neighbor List = neighbors(TR3) of each facet (3 indices) **FNL**: Facet normal List = faceNormal(TR3) of each facet (normal vector)

VLcrossingSG

returns crossing points of two solid geometries

In: **SGA**: Solid A or cell list of solids **SGB**: Solid B or cell list of solids

Out: **VL**: Vertex list of all crossing point

VLFLarea

returns the area of the surfaces (VL/PL)

In: **VL**: Vertex list (nx3) or Point List (nx2) **FL**: Facet list **fil**: optional list of facet; default is 1:size(FL,1)

Out: **ASUM**: Area of the surfaces **A**: Area list for facet list

VLFLfaceAngles

returns the three angles for the facets of a solid

In: **VL**: Vertex list **FL**: Facet list **thr**: threshold; default is eps2

Out: **FLa**: angle 1 2 3; [nf x 3] **ma**: minimal value angles [nf]

VLFLinpolyhedron

returns the isInside Flag for a SG and a VL

Out: **IN**: Vertex index list

5.9 SG Selection & Grouping [guide]

5.9.1 Selection [guide]

Functions in this section:

SGselect — selects solids from a cell list using a command language

SGselect3dPL — similar to select3d calls nearest point and vertices but for any given point

SGselectGroupofT — returns the part of the solid that contains a point, or a frame center

SGsurfaceselect — returns an isolated surface from a solid or SGofgca

Detailed description

SGselect

selects solids from a cell list using a command language

In: **SG**: cell list of solids **comsequ**: sequence of commands such as 'maxdim, mindim, maxbbvol, minbbvol, 'zmax','xmax','ymax', 'zmin','xmin','ymin' **"debug"**: if used the selection is shown step by step

Out: **SG**: cell ist of selected solids **SIL**: index/indices into the original list of the selected solids

SGselect3dPL

similar to select3d calls nearest point and vertices but for any given point

In: **SG**: Solid Geometry **p**: point to be checked

Out: **NL**: List of point with nearest distance to PL **FVL**: list of point on nearest vertex **fi**: index of nearest vertex

SGselectGroupofT

returns the part of the solid that contains a point, or a frame center

In: **SG**: Solid **T**: Frame or point **gmethod**: default is 'box'

Out: **SGN**: Selected Groups of Solids **SGW**: remaining groups of solids not containing the point

SGsurfaceselect

returns an isolated surface from a solid or SGofgca

In: **SG**: Solid geometry or empty for gca **fi**: facet index if numerical or cmd **vi**: vertex index **"debug"**: flag; if set, plot green selected and gray unselected surfaces for debugging

Out: **SGI**: Selected closed surface **SG**: SG with reordered facet list **SIL**: Surface index list for all surfaces **nfi**: facet index with respect to

5.9.2 Grouping & Collections [guide]

Functions in this section:

SGcell2array — converts a cell list of solids into a struct array of solids

SGcell2SG — converts all cells into one solid geometry

SGCpack — packs several solids side by side

SGplatesofSGML — converts a solid into a plated solid

SGseparatebyPez — searches infront and behind a ez vector for surfaces

SGseparatebyT — separates a solid into surfaces that are clearly below or above of a xy-plane

SGsortcollBB — returns a collision index list of all SG in the SG cell list

SGsortcollVLFL — returns a collision index list of all SG in the SG cell list and considers all VLFL

SGsortcollVLFLBB — returns a collision index list of all SG in the SG cell list but considers the BB of VLFL

SGsortfitinBB — returns a fits-in index list of all SG in the SG cell list

SGsortisinsideBB — returns a fits-in index list of all SG in the SG cell list

SGsurfaces — returns a cell list of separated CLOSED surfaces similar to SGseparate

SGsurfacessort — sorts the independent surfaces of a cell surface list

Detailed description

SGcell2array

converts a cell list of solids into a struct array of solids

In: **SGC**: Cell of Solids

Out: **SGN**: Array of Solids - requires the same struct fields for all structs

SGcell2SG

converts all cells into one solid geometry

In: **SGc**: {SGA; SGB; SGC;}

Out: **SG**: SG.VL, SG.FL

SGCpack

packs several solids side by side

In: **SG**: solid geometry collector, could be empty **B**: solid (VL,FL) **n**: number of added parts

Out: **SG**: solid geometry collector

SGplatesofSGML

converts a solid into a plated solid

In: **SG**: Original solid **th**: Thickness of the plates; default is 0.5 **pat**: CPL of a pattern to fill **w**: border size and pattern distance; default th **UVF**: list of unit vectors for filling;default=all ([])

Out: **NSG**: resulting solid consisting of plates **MA**: mounting face area table from MLofSG **SGi**: vertex index ranges [start end] of each plate within the resulting solid **FI**: [0 0 1;0 0 -1; 1 0 0; -1 0 0; 0 1 0; 0 -1 0]

SGseparatebyPez

searches infront and behind a ez vector for surfaces

In: **SGorg**: Original Solid Geometry **p**: center point **ez**: ez beam to find surface **FN**: optional frame name; default is 'C' **fl**: if true, flips the resulting frame 'C' by rot(0,pi,0); default is false **"debug"**: string flag 'debug' to plot the cutting frame and pause; default is off

Out: **SGout**: {behind-ez surface, infront-ez surface, untouched surface}

SGseparatebyT

separates a solid into surfaces that are clearly below or above of a xy-plane

In: **SG**: Solid Geometry **T**: Separating frame/ plane

Out: **SGX**: Surface that enclosed T (magenta); uncut **SGA**: Surfaces clearly ABOVE T (red) **SGB**: Surfaces clearly BELOW T (green) **SGC**: Surfaces that below and above of T (white)

SGsortcollBB

returns a collision index list of all SG in the SG cell list

In: **BBL**: Bounding Box List or Solid Geometry cell list with n entries

Out: **ELcoll**: Collision pair list | It is an undirected graph **CIL**: logical collision index list, size n **ss**: bounding bo size list **SG**: Solid Geometry list

SGsortcollVLFL

returns a collision index list of all SG in the SG cell list and considers all VLFL

In: **SGc**: Bounding Box List or Solid Geometry cell list with n entries

Out: **ELcoll**: Collision pair list | It is an undirected graph **CIL**: logical collision index list, size n **SG**: Solid Geometry list

SGsortcollVLFLBB

returns a collision index list of all SG in the SG cell list but considers the BB of VLFL

In: **SGc**: Bounding Box List or Solid Geometry cell list with n entries

Out: **ELcoll**: Collision pair list | It is an undirected graph **CIL**: logical collision index list, size n **SG**: Solid Geometry list

SGsortfitinBB

returns a fits-in index list of all SG in the SG cell list

In: **BBL**: Bounding Box List or Solid Geometry cell list with n entries

Out: **ELfit**: Fits in pair list | It is a directed graph **iso**: Neither inside nor encapsulation **xsi**: inbetween , i.e. encapuslation and enclosing **bsi**: encapsulating but not enclosed **isi**: nothing inside

SGsortisinsideBB

returns a fits-in index list of all SG in the SG cell list

In: **BBL**: Bounding Box List or Solid Geometry cell list with n entries

Out: **ELins**: Is enclosed-in pair list | It is a directed graph **iso**: Neither inside nor encapsulation **xsi**: inbetween , i.e. encapuslation and enclosing **bsi**: encapsulating but not enclosed **isi**: nothing inside

SGsurfaces

returns a cell list of separated CLOSED surfaces similar to SGseparate

In: **SG**: Solid geometry as single struct **slct**: surface index; default is all; SGseparate shows all

Out: **SGC**: cell list of solid geometry structs **n**: length of cell list **BB**: cell list of bounding boxes; see SGseparate

SGsurfacessort

sorts the independent surfaces of a cell surface list

In: **SG**: Solid Geometry as cell list **cmd**: optional sort command such as 'face', 'vertex', 'bbvol'

Out: **SG**: Solid Geometry **sl**: sorting list

6 Frames and Kinematics [guide]

6.1 Frame Definition [guide]

Functions in this section:

SGTBB — replaces the complex surface geometry by a simplified bounding box

SGTframeplot — plots one ore more frame of a solid

SGTframeselect — Fnct to select frame set

SGTget — returns a named frame of a solid geometry

SGTplot — simply plots solid including frames

SGTremove — removes a transformation frame from a solid geometry

SGTset — sets or replaces a named frame of a solid geometry

SGTsetofFS — creates a frame based on a features surface and a selector

SGTsetorigin — moves the solid into a position that the base frame become the origin

SGTui — returns a HT matrix for a manuel selected union space

T3ofT2 — converts a 3x3 HT-Matrix into a 4x4 HT-Matrix

Tof3P — returns a 4x4 HT matrix given by three points (sort3P)

Detailed description

SGTBB

replaces the complex surface geometry by a simplified bounding box

In: **SG**: cell list of solid geometries

Out: **SGbb**: cell list of bounding boxes

SGTframeplot

plots one ore more frame of a solid

In: **SGN**: Solid Geometry **N**: Name of the frame to plot, default is all **SN**: Solid Descriptor for frame names; default is []; **pf**: if true; TFiL is plotted if exists; default is true; **s**: size for frames to plot; default is sofBB(SGN)/5

Out: **h**: Called Frames

SGTframeselect

Fnct to select frame set

In: **SG**: Solid Geometry **Fset**: Starting Letter or set of starting letter **backw**: if true, matches frame names by reversed comparison; default is false

Out: **Fi**: Logical index of T **Ni**: names of the selected frames

SGTget

returns a named frame of a solid geometry

In: **SGN**: Solid Geometry **N**: Name of required Frame, or ez vector **TR**:

Out: **T**: Frame matrix **a**: index in Tname (Does work only in structs not in cells)

SGTplot

simply plots solid including frames

In: **SGN**: Solid Geometry **N**: Name of Frames to be plotted **SN**: Name of Solid for plotting

SGTremove

removes a transformation frame from a solid geometry

In: **SG**: SG (VL/FL plus Tname, T, TFiL) **N**: string with name of Frame; 'all' removes all frames

Out: **SG**: SG without named frame

SGTset

sets or replaces a named frame of a solid geometry

In: **SGN**: Solid Geometry **N**: Name of required Frame **T**: 4x4 matrix of the Frame; if empty SGTui is called **TFiL**: optional surface facet index that define the frame's surface

Out: **SGN**: Frame matrix

SGTsetofFS

creates a frame based on a features surface and a selector

In: **SG**: Solid Geometry **T**: Number of feature surface or normal vector of HT matrix **N**: Name of 'Frame' to be set **Rz**: optional rotation value, matrix or HT matrx **fs**: optional Feature symbol such as 'R1'; **fe**: feature edge/surface angle; default is 1rad ~ 60 degree

Out: **SGN**: Sg with added Frame **T**: Transformation matrix of frame **h**: handle to graphics patch **Rorg**: Radien list

SGTsetorigin

moves the solid into a position that the base frame become the origin

In: **SG**: Solid geometry **Nam**: Name of a frame or ez vector **TT**: optional 4x4 transformation post-multiplied onto the origin transform; default is empty

Out: **SGN**: spatial transformed SG **TI**: transformation matrix applied to the solid

SGTui

returns a HT matrix for a manuel selected union space

In: **SG**: Solid Geometry **N**: Name of the new/replaced Frame **fe**: feature edge angle; default 0.08; use 1 for freeform surfaces **FS**: Optional string selector for center points such as 'R1' **Rz**: Optional Rz rotation angle

Out: **SGN**: New SG with Tname, T, TFiL **T**: Transformation matrix **h**: handle to graphic **Rorg**: Rotation matrix of the created origin frame **FIL**: facet index list of the selected surface

T3ofT2

converts a 3x3 HT-Matrix into a 4x4 HT-Matrix

In: **T**: 3x3 HT Matrix **z**: optional z value

Out: **T**: 4x4 HT Matrix

Tof3P

returns a 4x4 HT matrix given by three points (sort3P)

In: **p1**: point 1 of the triangle, or VL **p2**: point 2 of the triangle **p3**: point 3 of the triangle

Out: **T**: HT Matrix describing the 3P coordinate system

6.2 Kinematics Chains [guide]

Functions in this section:

angle2vect2D — returns an angle or an angle interval into two unit vectors

attachmentpointsofA1B1TP — returns the grid points relative to effector from A1, B1 and a coordinate or pose

B1ofA1 — returns a second attachment point from a 1st Attachment point, distance and angle

CPLanimateTL — generic fnct to animate a set of contour on indivual pathes

CPLanimateTLasSG — generic fnct to animate a set of contour on indivual pathes as 2.5 layered solid geometries

CPLcrossCPLpathlimit — collision detection along a path in 2D

CPLinterlock — creates interlocking contours of attaching flat surfaces for kinematic purposes

CPLofA1B1P — returns a CPL exactly for the given three points

CPLofSGgears — provides the 2D envelope for a planar gearbox

CPLplotTL — simply plots a contour transferred wrt to list of frames

CPLRplotkinematicgraph — plots the kinematic graph in 3D with node/contour spheres and edge/joints connecting the contours

CPLsubtractTL — subtract a moving contour from another moving contour

CPLsubtractTLasSG — collison based subtraction of moving contours

CPLsubtractTLbs — subtract a moving contour from another moving contour, optionally cutting both contours equally

CPLsweepTL — returns the contour of a path swept contour

CPLunionTL — adds a moving contour to another moving contour

diffT — returns the difference between 2 4x4 HT matrices

ELgraphleafs — separates open branches in a edge list tree

ezofT — just returns the ez vector of an HT Matrix as row

graphdouble — recreates the edge list from a incidence matrix and also the inverted edge list

HTplot — plots a HT matrix

isequalT — returns the difference between 2 4x4 HT matrices

KMchain — calculates a kinematic structure from a given frame chain

KMofSGs — returns a Kinematic model for a chain of solids

KMplot — plots all parts of a kinematic model

KMreset — set all position frames of a kinematic model to eye(4)

KMT — returns a HT-Matrix from a kinematic model

L1L2RofBasenAttachmentpoints — returns dimensions from base points and attachment points

layerspacerlength — returns the levels of the spacer for collision avoidance and shaping

mat2celldim3 — converts an [a:b:n] array into a cell list {n x [a b]}

miofPLA0B0 — returns the index of the minimal relative angle to the axis between A0 and B0

mofzd — returns the module if diameter and number of teeth are known

mzs — returns a modified module for a giben number of teeth and slot

PLofT — returns a point list path from the translation part of a list of HT matrices

pofSG — returns a position based on spatial relations of SGtransrelSG

pofT — returns the translation part of frames or frame lists

radofT — returns only the radian turning angles of a list of T@ matrices

SGanalyzecrosscenter — approximates the center points of intruding surfaces

SGanalyzeJointFaces — returns joint type, Stator and Mover and Frame of Joint

SGangleoverlapT — Finds out whether a solid is insied a rotating angle of a frame and a angle interval

SGattach2SG — Assembles parts interactively by surface attachment

SGcenterFS — directly centers the desired feature surface

SGcutTextrudeT — returns a cutted an elongated solid by extrusion of the crosssection plane

SGgearchainrotate — rotates the axes of a gear chain

SGgearchainstraight — brings all the axes of a gear chain in a straight line

SGgearchainwindup — returns a gear chain for a windup motor or a minimal size gear box - use SGgearstamp afterwards

SGgearstamp — Marks gear wheels with numbers or text

SGisinternalframe — returns weather a frame is an internal frame of a solid

SGissurfaceframe — returns weather a frame is an outword or inward frame for a solid

SGManipulator03Base — Base, and Hand for Manipulator Design # 3

SGManipulator03Link — Link and Link extender for Manipulator Design # 3

SGManipulator04Base — Plate, Base, and Hand for Manipulator Design # 4

SGManipulator04Link — Link and Link extender for Manipulator Design # 4

SGManipulator05Base — Plate, Base, and Hand for Manipulator Design # 5

SGManipulator05Link — Link and Link extender for Manipulator Design # 5

SGManipulator07Link — Link and Link extender for Manipulator Design # 4

SGManipulatorBase — Plate, Base, and Hand for Manipulator Design # 4

SGManipulatorLink — Link with 2 Frames to create mechanisms or

SGN2SGT — finds hidden coded frames in a SG and creates frames for the solid

SGofR12 — creates a solid to show the minimal solid size for assembly methods

SGpianohingeanimate — Animates an ordered chain of solids relative to 'B' Base Frame and an angle limit

SGT2SGN — inserts frame information into the vertices and facets of a SG and removes the frame structures

SGTalignT — aligns two frame to have same ey-vectors by opposite ez vectors

SGTcalibchain — changes the base frames of a set of solid geoemtries

SGTchain — returns the spatial transformed solids of a kinematic chain

SGTchainanimation — creates an animation of kinematic chains

SGTcopyasopponent — copies a frame that crosses a surface as surface frame

SGTcopyinteralframes — copies frames of a solid which are located into the goal solid

SGTcopynames — copies Frames from one Solid into another one

SGTcopysurfaceframesout — copies all frames from one solid to another solid if the frames are outward frames

SGTcut — cuts a solid a an internal frame into separated pieced by a blade

SGTduplicateframe — duplicates a single frame of solid using a new name

SGTframerot — returns a frame of a solid

SGTframeselectrename — select frames, rename one, and remove the selected frames that were not renamed

SGTgetframeinfo — returns some basic information about a frame inside a solid

SGTmatchT — matches two frame to have same ey-vectors ez and ex vectors

SGtransTui — interactive fnct to fix a surface to the ground or another solid's surface

SGTregisterframes — finds the optimal HT matrix to register two sets of frames

SGTremovedistframes — removes frames that are outside the bounding box of the surface of a solid

SGTremoverelT — removes frames from an SGT that fullfill a condition

SGTrename — renames a frame of a solid geometry

SGTretain — removes frames but retains some

SGTsetBF — set frames 'B' and 'F' at default positions

SGTsetBFsimple — adds base frame and follower frame to a solid

SGTsetinternalframe — records coordinate frames based on optional user interaction

SGTsetofRL — creates frames from radial contour of a surface

SGTsetofRLcenter — returns a frame of a surface that is defined by drilling holes

SGTsetTL — adds a set of frames to a solid

SGTswap — Swaps two frames of a Solid

SGTtransR — rotates a selected frame of a solid

SGTtransT — Desired, but ambiguous, so only a help text is displayed.

SGturnfaces — rotates a solid wrt to the largest orthogonal feature surfaces

slpitch — returns the required fit dimension for steep screw thread

T2ofCVLpath — Returns Start frame and End Frame for a Contour path

T2ofT3 — returns 3x3 matrices from 4x4 matrices

Taplot — draws an anlge interval relative to a frame (3x3 or 4x4)

TcopyVLez — creates frames at specific positions in space

Tflipz — rotates / flips the ez axis of a HT matrix around by by 90 degree

TLadjustR — takes a List of Frames and adjustst start and end frame to T1 and T2

TLL2TL — Converts a long list of frames into an array of frames

TLof2T — returns a list of Frames between a start frame and an end frame

TLofn — returns a frame 1 of 36 with different ez vector and orientation

TLofPL — returns transformation frames for a PL path (!) the x axis is adjusted

TLofPLcloud — creates copies of a frame at the point list coordinates

TLofProt — creates a list of frames for rotations around a SINGLE point or SINGLE frame

TLplots — plots a list of HT matrices with plane, normal vector and prthogonal vectors

TLvaryfrom — this fnct takes a frame list and adds small values to the position and orientation

TmirrorTxy — mirrors frames or cell list of frames

Tof2P — returns a 4x4 HT matrix given by a point p and a vector ez [1st named T2P]

Tof2Tcrossez — adjusts the position of frames relative to a crossing point with another frame

TofA1B1GPL — Returns the Frame of the Pose wrt to

Tofchar — creates an HT frame based on spatial relations

TofCPLT — creates a list of frames inside a CPL or a planar CVL

TofcrossedSG — finds a surface point using ez of T and creates a frame

TofFSoutlinecenter — returns the HT Matrix fo the center of the FS

TofFSRLcenter — returns a Frame based on the smallest found radii inside a surface

TofFSui — returns the frames of a touched surface

TofLR — returns a 2D link matrix or a 3D link matrix

Tofn — provides a standardized test frame from 1 of 36 with different ez-vector and orientation

Tofplaneparam — returns the T matrix form for a plane parameter description - ax+by+cz+d=0

TofPLpath — creates a list of HT matrices from a point list path

TofRd — creates a frame after rotation in distance d along new ex

TofSGez — finds simple angle by ez vector using SGTsetBF

TofWheel — returns a HT matrix for a 2 wheel kinematic movement

tplotangle — plot for a list of HT matrices the angle, angle velocity and angle acceleration

Trelation — creates a frame relative to another frame

Trotate — returns a HT matrix rotated by

Tsvd — Correction fnct for an HT matrix for orthogonalizing a matrix after multiplication

uniqueT — returns the difference between 2 4x4 HT matrices

vecinterpolate — returns a vector row interpolation by

VLisbelowT — returns whether a list of points or a cell list of frames is under a frame

VLof2TR — returns a path with minimal number of axuiliary points to come from frame A to Frame B

VLofT — returns the origin T(1:3,4) of a frame list

VLofTtoP — returns a list o fpoints that create a shortes path towards PB with radius R

xyofez — returns for a simple ez vector the ex and ey orthonormal vectors

Detailed description

angle2vect2D

returns an angle or an angle interval into two unit vectors

In: **ww**: angle or angle intervall; default is [0 pi/3]; or pi/3 **cp**: center point or 3x3 Frame; default is [0 0];

Out: **v1**: unit vector of angle min length 1 = T1(1:2,1)') **v2**: unit vector of angle max length 1 = T2(1:2,1)') **T1**: 3x3frame wrt to cp and v1 **T2**: 3x3frame wrt to cp and v1

attachmentpointsofA1B1TP

returns the grid points relative to effector from A1, B1 and a coordinate or pose

In: **A1**: Position of A1 (Crank attachment point) **B1**: Position of B1 (Follower attachment point) **TP**: Pose or coordinate of coupler point

Out: **GPA**: Relative grid position of Crank attachment to Effector pose **GPB**: Relative grid position of follower attachment to Effector pose **TPP**: Pose in case that a point was used

B1ofA1

returns a second attachment point from a 1st Attachment point, distance and angle

In: **A1**: Attachment Point 1 relative to Origin/Pose center **L2**: Length of coupler **w**: angle of Pose to achieve

Out: **B1**: Position of Attachment Point 2

CPLanimateTL

generic fnct to animate a set of contour on indivual pathes

In: **cnam**: string name of contour **cset**: cell list of {CPL, TL-HT-array, colstr}; such as [PLsquare(10), TL, 'r'] **"WCR"**: optional list of angle values to show instead of steps

Out: **h**: handle to last plot

CPLanimateTLasSG

generic fnct to animate a set of contour on indivual pathes as 2.5 layered solid geometries

In: **cnam**: string name of contour **cset**: cell list of {CPL, TL-HT-array, colstrm [lmin lmax]}; such as [PLsquare(10), TL, 'r' such as [0 1] **"h"**: value following "h", the layer height; default is 5 **"WCR"**: value following "WCR", the optional crank angle list; default is nan **"steps"**: value following "steps", the number of animation steps; default is nan

Out: **h**: handle to last plot

CPLcrossCPLpathlimit

collision detection along a path in 2D

In: **CPLW**: Static Object **CPLE**: Moving Object **TL0**: List of Frames to move **WW**: List of corresponding values of **lim**: allowed area collision; default is 0

Out: **wint**: angle intervals with respect to WW **AA**: maximum intersection per interval in percent of CPLE **aint**: area at start of interval and end of interval **cutw**: cutting index

CPLinterlock

creates interlocking contours of attaching flat surfaces for kinematic purposes

In: **CPLA**: Single closed contour A (larger contour / base) **CPLB**: Single closed contour B (smaller contour / mover) **Pal**: [interlock-height angle nozzle accuracy]; default is [1.4 40/180*pi 0.4] **"debug"**: if used, the design steps are shown

Out: **CPLA**: modified contour A **CPLB**: modified contour B **CPLX**: contact surface space / blade

CPLofA1B1P

returns a CPL exactly for the given three points

In: **A1**: Position of A1 **B1**: Position of B1 **P0**: 1) Position of Pose or 2) Frame of Pose wen A1 B1 is reached **R12o**: Outer and inner Radius for linkages **CPLN**: CPLE as additional contour; 0 for an automatic shaped

Out: **CPLE**: CPLE **T**: Frame of Pose at **CPLX**: CPLE at Position T

CPLofSGgears

provides the 2D envelope for a planar gearbox

In: **SG**: Cell array of gears comming from SGgearchainwindup etc.

Out: **CPLN**: Contour the is required for the housing and the axes bore

CPLplotTL

simply plots a contour transferred wrt to list of frames

In: **CPL**: 2D planar contour **TL**: List of frames [4x4xn] **parameter**: similar to PLplot **"animate"**: if given as 'animate', the contours are drawn one frame at a time

Out: **h**: handle to plot

CPLRplotkinematicgraph

plots the kinematic graph in 3D with node/contour spheres and edge/joints connecting the contours

In: **CPLR**: cell array of contours **T**: cell array of frames **CMAT**: CMAT or Incidence matrix B or Radius for checking CMAT or empty

Out: **h**: handle to graphics objects[

CPLsubtractTL

subtract a moving contour from another moving contour

In: **CPL1**: Contour to subtract from **TL1**: path of contour 1 must have the same length of TL2 or 3x3 **CPL2**: Contour to be subtracted **TL2**: path of contour 2 must have the same length of TL1 or 3x3 **buf**: buffer to increase CPL2 before start **"debug"**: if used the function shows results stepwise

Out: **CPLN**: New contour

CPLsubtractTLasSG

collison based subtraction of moving contours

In: **LLLL**: cell list {CPL1, TL1, Level1; CPL2, TL2, Level; ...} **subc**: contour index row of the contour that is reduced **"h"**: **"WCR"**: optional for for graphical output => height **"debug"**: optional for for graphical output => list of angles

Out: **LLLN**: List in which the cntour of subc ist modified by subtracting the colliding contour in the same layer

CPLsubtractTLbs

subtract a moving contour from another moving contour, optionally cutting both contours equally

In: **CPL1**: Contour to subtract from **TL1**: path of contour 1 must have the same length of TL2 or 3x3 **CPL2**: Contour to be subtracted **TL2**: path of contour 2 must have the same length of TL1 or 3x3 **buf**: buffer to increase CPL2 before start **"both"**: if used both contours are cut equally to achieve collision free movements **"debug"**: if used the function shows results stepwise

Out: **CPLN**: New contour of CPL1 **CPLU**: New contour of CPL2 if 'both' has been used

CPLsweepTL

returns the contour of a path swept contour

In: **CPL**: Contour with Center **TL**: HT list [3 x 3 x n] with n frames

Out: **CPLS**: CPLunion of all transformed CPLs

CPLunionTL

adds a moving contour to another moving contour

In: **CPL1**: Contour to add to from **TL1**: Contour to be added **CPL2**: path of contour 1 must have the same length of TL2 or 3x3 **TL2**: path of contour 2 must have the same length of TL1 or 3x3 **buf**: buffer to increase CPL2 before start

Out: **CPLN**: New contour

diffT

returns the difference between 2 4x4 HT matrices

In: **TA**: HT Matrix A **TB**: HT Matrix B

Out: **delta**: [dp dz dy wz wy]

ELgraphleafs

separates open branches in a edge list tree

In: **EL**: Edge List **rekurs**: if true; the remaining tree contains just closed loops

Out: **ELlf**: AL Branches of the leaves sorted outwards **ELtr**: remaining tree of the tree unchanged sorted by loops if rekursively **SIL**: Select index list; undependent **knod**: list of node **term**: list of leafs (ELlf(:,2))

ezofT

just returns the ez vector of an HT Matrix as row

In: **T**: DH Matrix 4x4

Out: **ez**: T(1:3,3)'

graphdouble

recreates the edge list from a incidence matrix and also the inverted edge list

In: **B**: Incidence matrix n-node-rows x m-edge-cols

Out: **ELE2N**: Edge list for B (Node to node) **ELN2E**: Edge list after exchange of nodes and edges (now the nodes are edges) **G**: Original Node fitting to the EL connecting nodes **GD**: Inverted Graph created by ELN2E

HTplot

plots a HT matrix

In: **T**: HT matrix **Torg**: Origin of T

Out: **h**: handle to go

isequalT

returns the difference between 2 4x4 HT matrices

In: **TA**: HT Matrix A **TB**: HT Matrix B **check**: letters of distances to check default is 'pz' **thr**: default 1e-4;

Out: **is**: logical true **delta**: distance vector [origin ez ey line] from diffT between the two frames

KMchain

calculates a kinematic structure from a given frame chain

In: **KM**: Kinematic model {SG,'Name',HT-Matrix} **chain**: string 'A.A-A.B-' **dphi**: rotation angle in degrees applied along the chain; default 90

Out: **KM**: Kinematic model with adjusted Frames KM{:,3}

KMofSGs

returns a Kinematic model for a chain of solids

In: **SGs**: Cell list of solids: {SG1,SG2,SG3...} **dphi**: angle degree (default is 105 degree) for all joints **thr**: threshold for collision test; default is 0.1 (==2*0.05)

Out: **KM**: Kinematic Model including a simple chain **XVL**: generated chain of frames

KMplot

plots all parts of a kinematic model

In: **KM**: Kinematic model **c**: color used for plotting the parts; default 'r'

KMreset

set all position frames of a kinematic model to eye(4)

In: **KM**: Kinematic model {nx3}

Out: **KM**: Kinematic model {nx3}

KMT

returns a HT-Matrix from a kinematic model

In: **KM**: Kinematic Model {Solid ' Name'} **N**: Name "Solid.Frame"

Out: **T**: HT Matrix **i**: index of the solid within the kinematic model that matched the requested name

L1L2RofBasenAttachmentpoints

returns dimensions from base points and attachment points

In: **A0**: Base Point A0 **B0**: Base Point B0 **A1**: Attachment Point A1 **B1**: Attachment Point B1

Out: **l1**: length of swing **l2**: length of coupler **r**: length of crank **eb**: ellbow down (1) or ellbow up(2) **d**: distance between the base points

layerspacerlength

returns the levels of the spacer for collision avoidance and shaping

In: **se**: [startindex-in-layers endindex-in-layers] **lay**: [layer indices] default is [0 2 1 1] for ground, effector, crank swing] **R12**: just for visualization **H**: just for visualization

Out: **ll**: levels of spacer **d**: length of spacer **s**: direction of spacer

mat2celldim3

converts an [a:b:n] array into a cell list {n x [a b]}

In: **TL**: [4 x 4 x n] array of n 4x4 frames

Out: **NL**: cell list of [4x4 frames

miofPLA0B0

returns the index of the minimal relative angle to the axis between A0 and B0

In: **PL**: Point List (A1) of a fourbar linkage **A0**: Base Point A0 **B0**: Base Point B0

Out: **mi**: shift index for PL=circshift(PL,-mi)

mofzd

returns the module if diameter and number of teeth are known

In: **z**: number of teeth **d**: diameter of gear

Out: **m**: module

mzs

returns a modified module for a giben number of teeth and slot

In: **m**: module **z**: number of teeth **s**: slot; default is slfit ('t') % transition fit

Out: **mr**: reduced module to guaranteee the slot size

PLofT

returns a point list path from the translation part of a list of HT matrices

In: **T0**: List of matrices in 2D

Out: **PL**: Point list

pofSG

returns a position based on spatial relations of SGtransrelSG

In: **SG**: Solid **rel**: spatial relations valid in SGtransrelSG

Out: **p**: [x y z], coordinate

pofT

returns the translation part of frames or frame lists

In: **T**: T or TL

Out: **p**: p or PL

radofT

returns only the radian turning angles of a list of T@ matrices

In: **T0**: List of HT matrices in 2D

Out: **w**: list of angles (atan2)

SGanalyzecrosscenter

approximates the center points of intruding surfaces

In: **SG**: Solid Geometry consisting of overlapping separated surfaces **isexact**: if true; the real surface instead of BB is used; default is false

Out: **CLL**: collison list [ind-A ind-B [x y z] vol-A vol-B] **SGB**: cell list of surfaces of SG **BB**: cell list of bounding boxes of SG

SGanalyzeJointFaces

returns joint type, Stator and Mover and Frame of Joint

In: **SG**: Solid Geometry **SGB**: optional second solid geometry; currently unused

Out: **SGi**: Cell list of Solid Geoemtries **PL**: List containing set of penetrating parts **CML**: Corresponding Faces

SGangleoverlapT

Finds out whether a solid is insied a rotating angle of a frame and a angle interval

In: **SG**: Solid Geometry **T**: Frame ex defines the 0 axis for rotatation **R**: Radus of join (geoemtric correct, not fitting adjustment inside SGangleoverlapT) **wlim**: angle interval; default is [0 2*pi] **zoff**: optional height in z about T

Out: **isolap**: true or false if the angle interval at T with radius R and height zoff cuts the solid **wlap**: complete overlap interval **wint**: intersection of wlap and wlim

SGattach2SG

Assembles parts interactively by surface attachment

In: **A**: Solid A to attach to Solid B **B**: Solid B **usez**: if true, the z axis is used during frame registration; default is true **minmaxR**: minimal and maximal radius to consider; default is [1.0 inf]

Out: **A**: Solid A **B**: Solid B

SGcenterFS

directly centers the desired feature surface

In: **SG**: Solid geomety **fi**: cell list of facet index or **"rot"**: optional rotation angles **"debug"**: if used the debug function will be activated

Out: **SGN**: Solid with new center **T**: Original coordinate frame of FS **CPLA**: facet index of FS **fi**: facet index of FS

SGcutTextrudeT

returns a cutted an elongated solid by extrusion of the crosssection plane

In: **SG**: Solid Geometry **T**: Cutting Frame **Td**: Straight distance in z or Relative link **olap**: overlap of new and old parts **type**: straight, rad, rmax,tan; default is straight **unite**: if true and possible; function tries to combine the three elements again to one

Out: **SGN**: Result if there is only one solid; unite=true **Touch**: Transformation matrix of click, Global Start Frame **T**: Global End Frame **TN**: Global End Frame

SGgearchainrotate

rotates the axes of a gear chain

In: **SG**: Cell list of gear wheel with base frames 'B' **f**: array of gear ration with respect to the 1st one f(1)==1 **W**: array of desired rotatation of the following axis **SGA**: cell array of corresppponding wahser or axes to be moved with the gears

Out: **SG**: cell array of gears with modified position and rotation **SGA**: cell array of corresppponding wahser or axes

SGgearchainstraight

brings all the axes of a gear chain in a straight line

In: **SG**: Cell list of gear wheel with base frames 'B' **f**: array of gear ration with respect to the 1st one f(1)==1 **SGA**: cell array of corresppponding wahser or axes to be moved with the gears

Out: **SG**: cell array of gears with modified position and rotation **SGA**: cell array of the axis-aligned solids (spacers/washers) accompanying the gears; default is empty

SGgearchainwindup

returns a gear chain for a windup motor or a minimal size gear box - use SGgearstamp afterwards

In: **mhr**: [m h r hr] of gear; default is [1 5 1.25 0] **z**: array of number of teeth **"stack"**: if used the gears will be stacked as desired **"shaft"**: if used this radius or CPL will be used for 1st and last **"debug"**: if used the design will me shown **"anime"**: if used the gearbox will be animated **"backlash"**: if used the module is reduced to achieve a slot with the given value

Out: **SG**: cell list of gears **m**: array of modules used **f**: ratio factor of the gears **SGW**: cell list of washer or spacer for the gears in SG **CPLW**: Contour of the shaft spacer SGW **CPLC**: closed polygon list of the shaft spacer SGW

SGgearstamp

Marks gear wheels with numbers or text

In: **SG**: Cell list of gear wheels without a number **otxt**: optional other text; not used yet

Out: **SG**: Cell list of gears with engraved position within the gear sequence

SGisinternalframe

returns weather a frame is an internal frame of a solid

In: **SG**: Solid Geometry **T**: 4x4 Frame

Out: **VIL**: 1 internal, 0 = external

SGissurfaceframe

returns weather a frame is an outword or inward frame for a solid

In: **SG**: Solid Geometry **T**: 4x4 Frame **d**: tolerance; default is 1

Out: **VIL**: 1 outward, -1=inward, 0 = neither nor

SGManipulator03Base

Base, and Hand for Manipulator Design # 3

In: **LS**: Length of Base and Hand; default is 40

Out: **A**: Base PLate **B**: Base **U**: Hand **EF**: Effector

SGManipulator03Link

Link and Link extender for Manipulator Design # 3

In: **LL**: Axis Distance; default is 40

Out: **SL**: Link 2.5mm screw and 2.5mm Nut **SL2**: Link 2.5mm screw nut and 2.5mm screw nut

SGManipulator04Base

Plate, Base, and Hand for Manipulator Design # 4

In: **LS**: Length of Base and Hand; default is 40

Out: **A**: Base PLate **B**: Base **K**: 7 DoF **U**: Hand **EF**: Effector

SGManipulator04Link

Link and Link extender for Manipulator Design # 4

In: **LL**: Axis Distance; default is 40

Out: **SL**: Link 2.5mm screw and 2.5mm Nut **SL2**: Link 2.5mm screw nut and 2.5mm screw nut

SGManipulator05Base

Plate, Base, and Hand for Manipulator Design # 5

In: **LS**: Length of Base and Hand; default is 40

Out: **A**: Base PLate **B**: Base **K**: 7 DoF **U**: Hand **EF**: Effector

SGManipulator05Link

Link and Link extender for Manipulator Design # 5

In: **LL**: Axis Distance; default is 40

Out: **SL**: Link 2.5mm screw and 2.5mm Nut **SL2**: Link 2.5mm screw nut and 2.5mm screw nut

SGManipulator07Link

Link and Link extender for Manipulator Design # 4

In: **LL**: Axis Distance; default is 40 **Shap**: if true; the links are shaped; default is false **m**: screw diameter; default is 2.5

Out: **SL**: Link 2.5mm screw and 2.5mm Nut

SGManipulatorBase

Plate, Base, and Hand for Manipulator Design # 4

In: **LL**: Length of Base and Hand; default is 40 **Shap**: Surface shaping; default is true **LD**: Radius of the tips; default is 5mm **LH**: Height of the parts; default is 6mm

Out: **A**: Base PLate **B**: Base **K**: 7 DoF **U**: Hand **E**: Effector

SGManipulatorLink

Link with 2 Frames to create mechanisms or

In: **LL**: Axis Distance; default is 40; or [Startpoint;Endpoint] **Shap**: if true; the links are shaped; default is false **LD**: Radius of the Tip rounding; default is 5 **LH**: Height of the Elements also used as shaping Radius; default is 6 **DD**: if true; the shape is doubled for chaining; default is false **FR**: frame level flags [F B]; nonzero places the frame on top (z=LH); default is [0 0]

Out: **SG**: Link Geometry

SGN2SGT

finds hidden coded frames in a SG and creates frames for the solid

In: **SG**: Solid Geometry **del**: if true; the coded facets are removed; default is true;

Out: **SGT**: Solid Geometry including frames

SGofR12

creates a solid to show the minimal solid size for assembly methods

In: **R12**: Dimensions of a link **DIN**: if true; the function SGdesignDIN912DIN985is used; default is false

Out: **SG**: Solid of this link **RMHL**: Minimal Dimensions

SGpianohingeanimate

Animates an ordered chain of solids relative to 'B' Base Frame and an angle limit

In: **SG**: cell list of Solids that are an ordered kinematik chain including 'B' Frames **wlim**: angle limit; scalar or rows of angle limits **ELX**: Extended edge list to describe the kinematic chain ["B"-index "F"-index T-Frame-index] sign(T)=> -wlim **"rewind"**: if used, the links move forward and backwards

Out: **hh**: handle to graphics objects

SGT2SGN

inserts frame information into the vertices and facets of a SG and removes the frame structures

In: **SGT**: Solid Geometry with Frames

Out: **SGN**: Solid Geometry with coded frames and a normal vector list

SGTalignT

aligns two frame to have same ey-vectors by opposite ez vectors

In: **SG**: Solid Geometry **T**: Frame or Frame name of SG **TM**: eye(4) or new coordinate system to match **az**: optional final turn around ez

Out: **SGN**: New Position of SG using SGtransT

SGTcalibchain

changes the base frames of a set of solid geoemtries

In: **SGs**: Set of solids, with base and folloer frame **phi**: offset angle for the **Fram**: Frame name; default is 'B'

Out: **SGn**: Set of solids, with turned frames

SGTchain

returns the spatial transformed solids of a kinematic chain

In: **SGs**: Cell list of n Solid Geometries **phi**: vector of length n of radial rotations around the F-z frame **z**: optional z value in ez direction **nums**: optional chaining; nums can also be SGTframechain **extr**: extra chaining [SG1ind FFrame SG2ind BFrame] such as [1 'F' 2 'B'] **B**: Standard Base frame string; default is 'B' **F**: Standard follower frame string; default is 'F'

Out: **SGn**: Cell list of n spatial transformed Solid Geometries

SGTchainanimation

creates an animation of kinematic chains

In: **SGs**: Cell list of Solids to move **IVL**: Rows of angle values for the joints **z**: z value for explosion drawings **Fchain**: number of steps between two rows of IVL; default is 1 **stp**: number of interpolation betweem two rows **axo**: optional axis definition

Out: **h**: handle to the drawing **axG**: automated created axis/Bounding box for the movement

SGTcopyasopponent

copies a frame that crosses a surface as surface frame

In: **B**: Solid to receive the frame **A**: Solid that sends the frame **NA**: Name of the Solid to copy **NB**: optional new Name of B

Out: **B**: Solid that received the frame

SGTcopyinteralframes

copies frames of a solid which are located into the goal solid

In: **SGA**: Solid to add the frames to **SGB**: Solid to copy the frames from

Out: **SGN**: Final Solid including the frames ot the Origina

SGTcopynames

copies Frames from one Solid into another one

In: **SGA**: Solid to receive frames **SGB**: Solid to cope frames from **ListC**: optional list of frames to copy **ListU**: optional new name of frames to copy **Overw**: if true; force overwrite

Out: **SGA**: Modified Solid with unique framenames

SGTcopysurfaceframesout

copies all frames from one solid to another solid if the frames are outward frames

In: **SGA**: Solid to copy frames to **SGB**: Solid to copy frames from

Out: **SGA**: Solid to copy frames to

SGTcut

cuts a solid a an internal frame into separated pieced by a blade

In: **SG**: Solid Geometry with internal Frames **FN**: Frame Name **d**: [Thickness of cutting slice, buffer of CPL] default is [0.1 1]

Out: **A**: List of Surfaces

SGTduplicateframe

duplicates a single frame of solid using a new name

In: **SG**: Solid **FN**: Frame Name **DN**: Duplicated Name **flip**: if true; fliped around y, align to match

Out: **SG**: Solid with additional duplicated Frame

SGTframerot

returns a frame of a solid

In: **B**: Solid Geometry **N**: Frame name **R**: Rotation Matrix

Out: **B**: Solid with rotate frame

SGTframeselectrename

select frames, rename one, and remove the selected frames that were not renamed

In: **SG**: Solid Geometry **Fset**: string selection has to start with **Sel**: string selection has to end with **new**: New Name for selection

Out: **SG**: Solid Geometry with a renamed frame

SGTgetframeinfo

returns some basic information about a frame inside a solid

In: **SG**: Solid Geometru **T**: Frame

Out: **cinf**: crossing information **CPLA**: crossing contour **VLC**: vertex list of the frame crossing points (from crosspointsVLFLofT)

SGTmatchT

matches two frame to have same ey-vectors ez and ex vectors

In: **SG**: Original Position **T**: Frame Name of SG or Frame **TM**: eye(4) or new coordinate System **az**: final turn around ez

Out: **SGN**: Final Position of SG

SGtransTui

interactive fnct to fix a surface to the ground or another solid's surface

In: **A**: Frame A **TA**: Frame to attach; default is empty **TB**: Frame to attach to or Solid B; default is eye(4)

Out: **A**: Solid attached to x/y plane or to another frame

SGTregisterframes

finds the optimal HT matrix to register two sets of frames

In: **A**: Solid A **B**: Solid B **NA**: Frame name wildcard or cell list of A; default is empty **NB**: Frame name wildcard or cell list of B; default is NA **usez**: if true, only set with same ez are options; default is true

Out: **T**: Transformation Matrix **NSB**: Transformed Sold with removed Frames **A**: Solid with removed Frames

SGTremovedistframes

removes frames that are outside the bounding box of the surface of a solid

In: **SG**: Solid Geometry with frames.

Out: **SG**: Solid Geometry will all frames removed that are outside the BB of SG **ww**: true if any frames was removed

SGTremoverelT

removes frames from an SGT that fullfill a condition

In: **SG**: Solid Geometry with Frames **T**: Cutting Frame **rel**: relation such as 'uo' ==means under and on plane

Out: **SG**: SOlid Geometry with Frames only that fulfill the condition rel

SGTrename

renames a frame of a solid geometry

In: **SG**: Solid geometry **Norg**: Original name **Nnew**: New name **rm**: if true; existing frame Nnew is deleted first; default is false

Out: **SG**: Solid geometry with changed frame name

SGTretain

removes frames but retains some

In: **SG**: Solid Geomnetry struct with fields 'Tname', 'T', 'FTiL' **FN2R**: List of frame names to retain (all other frames are removed)

Out: **SG**: Struct with removed frames

SGTsetBF

set frames 'B' and 'F' at default positions

In: **SG**: Solid Geometry without Frames Frames 'B' and 'F' **varargin**: List of Frames to add; default is varargin={'B',[0 0 -1],'F',[0 0 1]}; **"new"**: 'new' to force creation of a new base frame **"alpha"**: 'new' to force creation of a new follower frame **"new"**: if the string "new" is present, all existing frames are removed first **"alpha"**: value following "alpha", the angle tolerance for face selection; default is pi/4

Out: **SGN**: Solid with Frames 'B' and 'F' **T**: T matrix

SGTsetBFsimple

adds base frame and follower frame to a solid

In: **SG**: Solid Geometry **PL**: xy position of centers; default is center of BB **az**: optional rotation

Out: **SGN**: Solid Geometry including base frame and follower frame

SGTsetinternalframe

records coordinate frames based on optional user interaction

In: **SG**: Solid Geometriy to add frames by mouse click **FN**: String for Framenames **Nr**: Number of Frames to enter; default is inf

Out: **SG**: Solid Geometry with new frames

SGTsetofRL

creates frames from radial contour of a surface

In: **SG**: Solid Geometry **RL**: Optional Radial Contour list (CVLdimensioning) - [cx cy cz Nr deg R vx vy vz] **N**: Starting Letter for Frames; default is "Hole" **minmaxR**: mimimal and maximal radius to consider default is [1.25 10] **minmaxW**: minimal & maxium circum angle to consider; default is [40 360]

Out: **SG**: New Solid with additional Frames **RR**: radial contour list rows used to create the frames **fi**: facet index (from SGprofilesurfacedimension)

SGTsetofRLcenter

returns a frame of a surface that is defined by drilling holes

In: **SG**: Solid Geometry **fi**: optiona facet index; default is ''; ==> interavtive **N**: Name of Frame **minmaxR**: minmax for radius default is [1.1 1.9] **minmaxW**: minmax for arc circle default is [170 360]

Out: **SG**: Solid including Frame

SGTsetTL

adds a set of frames to a solid

In: **SG**: Solid geometry **FN**: Leadning Frame letter or string **TL**: array [4 x 4 x n] of Frames **nn**: maximum number to support ith leading zeros; default is size(TL,3)

Out: **SG**: Solid Geometry with additonal Frames from the Frame list

SGTswap

Swaps two frames of a Solid

In: **SG**: Solid **FA**: Frame name A **FB**: Frame name B

Out: **SGN**: Solid with swapped frame names

SGTtransR

rotates a selected frame of a solid

In: **SG**: Solid geometry **FN**: Frame name **R**: Rotation matrix **p**: translation vector

Out: **SG**: SG with modified Frame

SGTtransT

Desired, but ambiguous, so only a help text is displayed.

In: **SG**: Solid with frames for SGTalignT, SGTmatchT **T**: Frame name **TM**: Reference frame; default is eye(4) **az**: rotation around final z axis

Out: **SGN**: would be the transformed SG by SGTalignT, SGTmatchT

SGturnfaces

rotates a solid wrt to the largest orthogonal feature surfaces

In: **SG**: Original Solid Gometry

Out: **SG**: Turned and Centered Solid Geometry

slpitch

returns the required fit dimension for steep screw thread

In: **D**: Diameter in mm **p**: pitch; default is pitch of DIN13

Out: **slp**: required distance for steep screw threads

T2ofCVLpath

Returns Start frame and End Frame for a Contour path

In: **XVL**: Simple Contour Path **T1**: Only if T1 and T2 are known already => calls Ti=TLadjustR(Ti,T1,T2); **T2**: Only if T1 and T2 are known already => calls Ti=TLadjustR(Ti,T1,T2);

Out: **TL**: Inside Start-Frame and Outside End-Frame **dw**: angle difference [torsion inclination] in rad **Ti**: Full List of Frames

T2ofT3

returns 3x3 matrices from 4x4 matrices

In: **T3**: 4x4 homogeneous transformation matrix or list of them

Out: **T2**: 3x3 (2D) transformation matrix

Taplot

draws an anlge interval relative to a frame (3x3 or 4x4)

In: **T**: Frame or cell list of frames **wint**: Angle interval **s**: size (like sofgca*200) **c**: color of the angle area **N**: optional name of the frame

Out: **h**: handle to graphics

TcopyVLez

creates frames at specific positions in space

In: **VL**: Point list or vertex list **ez**: ez vector default is [0 0 1] **wa**: rotation around ez **po**: position offset that is added to each translation vector * (i-1); default is [0 0 0] **wo**: angle offset that is added to each translation vector * (i-1); default is 0 or [0 0 0] **TT**: Frame to transfer all resulting frames with; default is ''

Out: **TL**: List of Frames

Tflipz

rotates / flips the ez axis of a HT matrix around by by 90 degree

In: **T**: Standard HT matrix

Out: **TN**: rotate around y by 180 deg

TLadjustR

takes a List of Frames and adjuststst start and end frame to T1 and T2

In: **TL**: Frame list **R1**: R or T Matrix **R2**: R or T Matrix

Out: **TL**: Corrected Frame list

TLL2TL

Converts a long list of frames into an array of frames

In: **TLL**: Long list format [n*4 x 4]

Out: **TLA**: array format [4 x 4 x n]

TLof2T

returns a list of Frames between a start frame and an end frame

In: **TA**: Start Frame **TB**: End Frame **elo**: elongation; default is 0.001 **R**: Radius for bending **Cs**: contour style' default is radial

Out: **TL**: list of frames [4x4xn] along the path from start to end frame

TLofn

returns a frame 1 of 36 with different ez vector and orientation

In: **n**: 1..36 for an frame; if n==0 a random solution is selected **poff**: offset value; default is [0 0 0] **woff**: angle osset default is 0 **d**: distance between the two frames default is 100

Out: **T1**: List of two frames or T1 **T2**: T2

TLofPL

returns transformation frames for a PL path (!) the x axis is adjusted

In: **PL**: Point list **usem**: if true (default is false); the y vector shows to the center of rotation, default is x shows towrds the next point **nonc**: if true (default), the TL will be elongated by the last point (and is not closed); **z**: z value for T3 cases; default is 0 **"TLofCVL"**: if used, the output (using true,true) is similar to TLofCVL

Out: **TL2**: [3x3x n] 2D HT matrix array **TL3**: [4x4x n] 3D HT matrix array

TLofPLcloud

creates copies of a frame at the point list coordinates

In: **PL**: Point list in 2D or 3D **wlim**: not used yet **woff**: not used yet

Out: **TL**: Just copies of frames at the position of PL

TLofProt

creates a list of frames for rotations around a SINGLE point or SINGLE frame

In: **P**: SINGLE Point [x y]; default is [0 0]; or SINGLE frame **wlim**: [wmin wmax]; default is [0 pi] **n**: number of frames; default is n=120

Out: **TL**: page list of Frames [3x3xn] or [4x4xn] **PLa**: contour for future use

TLplots

plots a list of HT matrices with plane, normal vector and prthogonal vectors

In: **TL**: Lust of HT matrices

Out: **h**: handle to graphics

TLvaryfrom

this fnct takes a frame list and adds small values to the position and orientation

In: **TL**: Frame list **dd**: [dx dy dz] = default is [1 1 1]/1234 **rr**: [wx wy wz]; default is [1 1 1]*34/14333

Out: **TLN**: modified frame list

TmirrorTxy

mirrors frames or cell list of frames

In: **TA**: frame or cell list of frames to be mirrored **T**: mirror axis is given by ez, T, ez, 'x'

Out: **TB**: mirrored fram of mirrored frame list

Tof2P

returns a 4x4 HT matrix given by a point p and a vector ez [1st named T2P]

In: **p**: Point of straight line **ez**: Direction of a straight line **typ**: Type selector for the frame construction

Out: **T**: Matrix descibring a line using p and ez

Tof2Tcrossez

adjusts the position of frames relative to a crossing point with another frame

In: **T1**: Frame 1 **T2**: Frame 2

Out: **TA**: Moved Frame 1 **TB**: Moved Frame 1

TofA1B1GPL

Returns the Frame of the Pose wrt to

In: **A1**: Attachment Point A1 in Space **B1**: Attachment Point B1 in Space **GPLA**: Attachment Point A1 on Effector **GPLB**: Attachment Point B1 on Effector

Out: **TP**: Pose in Space **TA1**: Fram A1 in Space **TB1**: Fram B1 in Space

Tofchar

creates an HT frame based on spatial relations

In: **T**: char of ez, or empty for eye(4) **relparam**: additional relative parameters for Ttransrel

Out: **TN**: new T matrix

TofCPLT

creates a list of frames inside a CPL or a planar CVL

In: **CPL**: CPL **TF**: T matrix, or ez vector of ezofchar **nr**: [number of points, radius, distance to cpl boundary **style**: "contour", "grid", "xgrid"

Out: **TL**: list of frames with same ez as TF **ACVL**: cell list of contour vertex lists transformed by TF

TofcrossedSG

finds a surface point using ez of T and creates a frame

In: **SG**: Solid to be crossed **T**: Starting frame **sd**: optional start distance from T

Out: **T**: Frame HT matrix **d**: distance to 1st wall **aa**: full list [facet index; distance] as of crosspointVLFL

TofFSoutlinecenter

returns the HT Matrix fo the center of the FS

In: **SG**: SOlid Geometry **fi**: list of indices ot list of features identical to FSselect **"debug"**: if used the ctreation is shown

Out: **T**: Center frame of the surface **CPLA**: Outer conotur in the eye(4) CS **fi**: faces index list

TofFSRLcenter

returns a Frame based on the smallest found radii inside a surface

In: **SG**: Solid Geometry **fi**: optional fi; if empty, interactive mode **al**: alpha value for surface detection; default is 0 **minmaxR**: default is [1.1 4] **minmaxW**: default is [170 360]

Out: **TF**: Front Frame **TB**: Backside Frame **RL**: Radii list **URL**: selected radii

TofFSui

returns the frames of a touched surface

In: **SG**: Solid Geometry **FS**: Selector such as 'R1' **rz**: optional ez rotation **fe**: optional feature angle

Out: **T**: HT Matrix of the Frame **FIL**: index list of faces of the identified surface (FSofSGT)

TofLR

returns a 2D link matrix or a 3D link matrix

In: **L**: length of a link or [L wz] or [L wx wy wz] **R**: angle of a joint at the beginning of the link or just [wz] or [wx wy wz] **T0**: Frame the is is linked to default is eye(4); or List of L or cell list of T

Out: **T**: Homogenoues transformation matrix

Tofn

provides a standardized test frame from 1 of 36 with different ez-vector and orientation

In: **n**: 1..36 for an frame; if n==0 a random solution is selected **poff**: offset value; default is [0 0 0] **woff**: angle osset default is 0

Out: **T**: the selected test frame [4x4]

Tofplaneparam

returns the T matrix form for a plane parameter description - ax+by+cz+d=0

In: **a**: [a b c d] or just a **b**: b **c**: c **d**: d

Out: **T**: Matrix form **VL**: [x 0 0; y 0 0; z 0 0;

TofPLpath

creates a list of HT matrices from a point list path

In: **PL**: Point list path

Out: **T0**: List of T2 homogenoues transformation matrices **PS**: pose list

TofRd

creates a frame after rotation in distance d along new ex

In: **w**: rotation angle or rotation vector **d**: distant along dx **e**: direction; default is 'x'; not used yet

Out: **TR**: HT matrix

TofSGez

finds simple angle by ez vector using SGTsetBF

In: **SG**: Solid Geometry **ez**: desired ez vector **alp**: allowed deviation; default is pi/4

Out: **T**: Found frame **err**: angle difference between ez vectors

TofWheel

returns a HT matrix for a 2 wheel kinematic movement

In: **D**: Distance between the two weels in x **w**: movement in rad

Out: **TS**: Center Frame after movement **PL**: path of movement **TR**: Frame of result of right wheel **TL**: Frame of result of left wheel

tplotangle

plot for a list of HT matrices the angle, angle velocity and angle acceleration

In: **T**: Transformation matrix list **wcr**: optional angle interval

Trelation

creates a frame relative to another frame

In: **T**: Frame; default is eye(4) **rel**: sequence of spatial relations

Out: **T**: resulting HT frame

Trotate

returns a HT matrix rotated by

In: **T**: Transformation matrix **ax**: rotation axis 'x,'y','z' or axial, coronal, sagittal **val**: value; default is p/2

Out: **NT**: New Transformation matrix

Tsvd

Correction fnct for an HT matrix for orthogonalizing a matrix after multiplication

In: **T**: Original HT Matrix

Out: **TS**: svd corrected T matrix with corrected R=[ex ey yz]

uniqueT

returns the difference between 2 4x4 HT matrices

In: **T**: list of HT Matrix A **check**: letters of distances to check default is 'pz' **thr**: default 1e-4;

Out: **T**: logical true **ia**: index a **ic**: index c

vecinterpolate

returns a vector row interpolation by

In: **VL**: list of row vectors **stp**: step number for interpolation

Out: **IVL**: Interpolated vector list

VLisbelowT

returns whether a list of points or a cell list of frames is under a frame

In: **VL**: Vertex List **T**: Frame the describe plae **thr**: threshold for on; default is 1e-5

Out: **b**: logical of below **a**: logical of above **o**: logical of on

VLof2TR

returns a path with minimal number of axuiliary points to come from frame A to Frame B

In: **TA**: Frame A, z-vector shows forward creating the start of the path **TB**: Frame B, z-vector shows backwards creating the end of the path **R**: Minimal Radius **ty**: not supported yet, default is 'skeleton' => CVLpathwarning

Out: **CVL**: List of 2 to 4 points **RVL**: List with more points to create a radial path

VLofT

returns the origin T(1:3,4) of a frame list

In: **TL**: [n*4 x 4] list or cell list of 4x4 frames

Out: **VL**: vertex list of the frame origins

VLofTtoP

returns a list o fpoints that create a shortes path towards PB with radius R

In: **TA**: Frame **PB**: Final point to reach **R**: minimal required radius

Out: **CVL1**: List of 2 to 4 points starts at TA leads towards PB

xyofez

returns for a simple ez vector the ex and ey orthonormal vectors

In: **ez**: Richtungsvektor eines Pfades

Out: **ex**: orthogonal vector to ez **ey**: orthogonal vector to ex and ez **ez**: normalized vector ez

6.3 Forward and Inverse Kinematic [guide]

Functions in this section:

forwkinrplan3 — forward kinematic of a planar rz 2 arm rotating around an arm z axis

invkinplan2 — returns the inverse kinematic for a planr 2 arm robot

invkinrplan3 — returns the angles for a rotating planar two arm kinematic

ROBgetJointPosition — returns the position of a set of joints of a robot

Detailed description

forwkinrplan3

forward kinematic of a planar rz 2 arm rotating around an arm z axis

In: **d1**: distance of link 1 **a2**: width of link 2 **a3**: width of link 3 **WL**: Linst of angle (n x 3)

Out: **VL3**: Coordinates of link3 **VL2**: Coordinates of link2 **VL1**: Coordinates of link1 **RL3**: Rotation matrix at link 3 [3,3,n] **TL3**: HT matrix at link 3 [4,4,n]

invkinplan2

returns the inverse kinematic for a planr 2 arm robot

In: **l1**: length first link **l2**: length second link **PL**: Point list (n x 2) or [r dx dy] or [r dx dy n] or

Out: **W**: Angle Pair [n x 2] for solution 1 **U**: Angle Pair [n x 2] for solution 2 **B1**: Endpoint of Arm 1 for solution 1 **B2**: Endpoint of Arm 1 for solution 2 **rs**: real solutions index; true ~ solution exist **PL**: used Point list in case that PL was empty or a triple for a circle **ss**: Singularity Index rs&ss

invkinrplan3

returns the angles for a rotating planar two arm kinematic

In: **d1**: Height of the base **a2**: length of link 2 **a3**: length of link 3 **VL**: Vertex list [n x 3]

Out: **aW**: Set of Configuration A (ellbow up) **aU**: Set of Configuration B (ellbow down) **bW**: Set of Configuration C (reverse ellbow up) **bU**: Set of Configuration D (reverse (ellbow down) **ri**: row index of VL for real solutions **VL**: Vertex list if created by this function

ROBgetJointPosition

returns the position of a set of joints of a robot

In: **KGID**: Kinematic Group ID **jset**: Joint set; defualt is empty == all joints of KGID

Out: **t**: time vector (t0 t1) **qi**: joint values of all or the requested set if joints [q1 q2]

6.4 Symbolic Definitions [guide]

Functions in this section:

SGTofDenavitHartenberg — returns from Denavit-Hartenberg values, a solid, frames and a symbolic description for DH

SGTofDHset — returns the solids for a kinematic chain defined by DH parameter

solveequations — returns the symbolic solutions for a give set of equations - What we learn in math

subsvar — replaces several symbols in an equation

symbforwkine — returns a forward kinematic calculation from a symbolics links list description

symfun2sym — converts a symfun into a sym (=command formula)

symfunchng — changes the parameter of a given equation

symfunparam — returns individualized string for symbolic parameters and definition

symsofequation — returns or defines the symbols of a equation system

Detailed description

SGTofDenavitHartenberg

returns from Denavit-Hartenberg values, a solid, frames and a symbolic description for DH

In: **D**: Distance in z; default is 100 **A**: Distance in x; default is 40 **be**: fixed rotation value around x-axis or [x-axis/yaxis]; default is [pi/2 0] **r**: optional Radius of the tube; default is max(D,A)/10 **g**: optional angle as offset for a rotation at the base; default is 0

Out: **SGT**: Solid Geometry including Frames **T**: symbolics equation including substitued values for parameter **TT**: symbolics equation without substituted values for parameter

SGTofDHset

returns the solids for a kinematic chain defined by DH parameter

In: **DHS**: Denavit-Hartenberg Parameter [nx3] [d a wx] or [d a xw xy] **r**: optional diameter for tube **g**: optional offset for phiz

Out: **SG**: Cell list of Solid Geoemtries **Ti**: Cell ist of Symbolic Equation **SGN**: Compiled Solid using the kinematic chain **Tii**: Cell list of complete symbolic equations **E**: Final movement matrix of the TCP

solveequations

returns the symbolic solutions for a give set of equations - What we learn in math

In: **eq**: equation, use '==' for equal sign and ',' for separation **srch**: symbols your are looking for **givn**: optional given values % not supported yet

Out: **res**: result as struct

subsvar

replaces several symbols in an equation

In: **f**: original term / equation **sv**: array of symbols to substitute or [s1 v1; s2 z2]; **nv**: array of substitutes

Out: **g**: resulting term / equation

symbforwkine

returns a forward kinematic calculation from a symbolics links list description

In: **a**: unused yet **b**: cell list of n symbolics symbolic HT matrices x of 1doF rotational links **WL**: List of k angles of n links [k x l]

Out: **Ti**: cell list of n symbolic HT matrices of b link end points **LL**: matrix [n x 3 x k] **H**: matlab function handles the calculation of Ti

symfun2sym

converts a symfun into a sym (=command formula)

In: **AF**: symbolic function

Out: **AS**: symbol separated function **a**: argnames

symfunchng

changes the parameter of a given equation

In: **AF**: symbolic function **L**: Letter for specification of a joint

Out: **AS**: symbolic function with changed parameters **a**: argnames

symfunparam

returns individualized string for symbolic parameters and definition

In: **AF**: symbolic equation **L**: Letter for individual equation

Out: **vpar**: variable string (parameter separated by comma) **vsyms**: symbol definition string (parameter separated by space)

symsofequation

returns or defines the symbols of a equation system

In: **eq**: equation systems to analyze

Out: **S**: String of Symbols **v**: cell list of Symbols **ev**: evaluation string

6.5 Fourbar Linkage [guide]

Functions in this section:

calc4BarAngle — returns 4-Bar-Linkage points for link B

plot4Bar — plots a 4-Bar-linkage

SGfourBarLinkageKit — Creation kit for Four-Bar-Linkages (4Bar/Linkages)

synth4Bar2Pose — returns 4 points for a 4 Bar linkage

synth4Bar3Pose — returns 4 points for a 4 Bar linkage

Detailed description

calc4BarAngle

returns 4-Bar-Linkage points for link B

In: **A0**: Base point of link A **B0**: Base point of link B **A1**: Position of point A1 **lb**: length of link B **lc**: length of coupler (A1B1)

Out: **B1**: Right hand solution for positive x **B1a**: Left hand solution for positive x **x**: straight distance from A1B0 **y**: orthogonal distance from A1B0 **e**: unit vector A1B0 **o**: orthogonal vector A1B0

plot4Bar

plots a 4-Bar-linkage

In: **A0**: Base point of link A **B0**: Base point of link B **A1**: End point of link A **B1**: End point of link B **wl**: list of angles; default is [0..2pi] **cc**: color of the coupler; default is ''

SGfourBarLinkageKit

Creation kit for Four-Bar-Linkages (4Bar/Linkages)

In: **task**: String for Task, default is bar **2**: 2nd parameter for shapes **3**: 3rn parameter for shapes/mode

Out: **SG**: Solid geometry of the four-bar linkage kit

synth4Bar2Pose

returns 4 points for a 4 Bar linkage

In: **C1**: Coupler point C Pose 1 **D1**: Coupler point D Pose 1 **C2**: Coupler point C Pose 2 **D2**: Coupler point D Pose 2 **d**: distance from C1/C2/ to A1/A2 **la**: length from mA to A0 **lb**: length from mB to B0

Out: **A0**: Base link A **B0**: Base link B **A1**: End link A **B1**: End link B **P12**: Pole (A0=B0)

synth4Bar3Pose

returns 4 points for a 4 Bar linkage

In: **C1**: Coupler point C Pose 1 **D1**: Coupler point D Pose 1 **C2**: Coupler point C Pose 2 **D2**: Coupler point D Pose 2 **C3**: Coupler point C Pose 3 **D3**: Coupler point D Pose 3 **d**: distance from C1/C2/C3 to A1/A2/A3; default [1 -1]

Out: **A0**: Base link A **B0**: Base link B **A1**: End link A **B1**: End link B

6.6 Linkages and Coupler Curves [guide]

6.6.1 Coupler Curves and Joint Kinematics [guide]

Functions in this section:

CPLRanalyzeTkinematics — Analyses the contour regions' struture and the hinge frames - detects need of and executes modifications

CPLRanimate — animate the 2D contours of a linkage if the movement is calculated by PLofLinkagecouplercurve

CPLRinserthingeclosed — returns CPL regions linked by Frames prepared for extrusion

CPLRinserthingelinkage — returns CPL regions linked by Frames prepared for extrusion

CPLRinserthingelist — returns CPL regions linked by Frames prepared for extrusion

CPLRintersectsweep — unites interseceted contours of a linkage

CPLRmeltbyCycles — removes closed 2-loops and 3-loops of contour even of non-binary joints

CPLRmeltbyIndex — simple fnct for melting contours in a cell array of linkage contours by a specified index

CPLRmove — Animates a moving linkage using PLofLinkagecouplercurve for experiments

CPLRofELC — create a minimal contour set of link joint sets

CPLRofELT — melts based on ELT and PLA and optional CPLR the contours

CPLRofPLEL — creates a set of linkages based on on a PL and EL and radius information

CPLRofPLELplain — creates a set of linkages based on on a PL and EL and radius information

CPLRofPose — creates a CPLR of a pose or pose solution

CPLRposechange — adjust the pose of the individual link contours in CPLR to fit into a new index of the joint coordinates

CPLRposesetbycrank — Sets a linkage driven by two base points an a crank link to a new crank angle

CPLRsample — returns sample contours for the linkage analysis

CPLRshapelink4pose — shapes a link contour to use other contours a conotur path space or to avoid a predefined additional contour

CPLRsimplifyEL — unifies contours whos corresponding edge list rows corrensponds to the same joints' index and reorders the edge list

CPLRsubtractjointsweep — modifies the link contours which are cut or blocked by static/moving joints contours

CPLRsubtractjointtrack — modifies the link contours which are cut or blocked by static/moving joints contours or static/moving link contours or creates minimal link contour if requested

CPLRsubtractTLbs — CPLR collision free shaphing and forming of contour links during movements

CPLRunionsweep — unites swept contours of a linkage

EELofCPLR — Creating the extended edge list and the graph using the joint indices as weight

ELCofELT — create a ELC cell array from ELT

ELTofCPLR — Short version that creates only the truss edge list from a contour and joint coordinates

ELTofELC — returns the edge list fo the truss from the Link contour owened joint list ELC

wlim4CPLRinserthingeclosed — returns the absolute joint angles wrt a given Pose of a linkage used in CPLRinserthingeclosed

Detailed description

CPLRanalyzeTkinematics

Analyses the contour regions' struture and the hinge frames - detects need of and executes modifications

In: **CPLR**: cell array of regions to be connected by joints **T**: cell array of hinge frames to be planned - could be calculated by does not need to **R**: Radius of the hinges, used only for limiting the search area **"debug"**: if used, the individual calculation steps arte shown

Out: **CPLRN**: modified cell array of link contours, unsorted wrt the original **CMAT**: correlation matrix (incidence matrix) n-row-contours x m-cols-frames **B**: incidence matrix (double) of the kinematics, contours by joint frames

CPLRanimate

animate the 2D contours of a linkage if the movement is calculated by PLofLinkagecouplercurve

In: **CPLR**: n x {CPL} cell array of link contours **Pall**: m x 2 x k Point coordinates of m Joints in k poses **ELC**: n x {} cell array containing the joint in each link **posi**: pose index within Pall(:,:,posi) that describes the pose of CPLR

Out: **h**: handle to graphics

CPLRinserthingeclosed

returns CPL regions linked by Frames prepared for extrusion

In: **CPLR**: Contour in which the hinges are to be inserted **T**: cell list of Frames. Empty Frames mean interactive selection, Numeric scalar means n interactive frames **R**: same as in SGcylhinge but only R(1) used **wlim**: angle limit [amax] or [amin amax] with respect to the x axis of the frame **"debug"**: is helpful to avoid material reduction when there are steep angles as in "CPSinserthingelist" **"solid"**: if used, the individal steps are shown as in "CPSinserthingelist"

Out: **CPLR**: Cell list of regions to be used in SGofCPLRTzpianohingeclosed **T**: cell list of frames to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **R**: radius used for joints to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **EEL**: Extended edge list [stator-index rotor-index signed-frame-index] to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **CPLCA**: Connnecting contour availabe to connct stator contour CPLA to the joint **CPLCB**: Connnecting contour availabe to connct rotor contour CPLB to the joint **wlim**: cell ist of angle interval or singe angle interval [wmin wmax]

CPLRinserthingelinkage

returns CPL regions linked by Frames prepared for extrusion

In: **CPLR**: Contour in which the hinges are to be inserted **T**: cell list of Frames. Empty Frames mean interactive selection, Numeric scalar means n interactive frames **R**: same as in SGcylhinge but only R(1) used **wlim**: angle limit [amax] or [amin amax] with respect to the x axis of the frame **NCind**: NCind is required to identy the order of subtraction **ELC**: if used, the individal steps are shown as in "CPSinserthingelist" **"cage"**: if used, the frames do not cut trough as in "CPSinserthingelist" **"debug"**: if used, the contour finally is closed again (not implemented yet)

Out: **CPLR**: Cell list of regions to be used in SGofCPLRTzpianohingeclosed **T**: cell list of frames to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **R**: radius used for joints to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **EEL**: Extended edge list [stator-index rotor-index signed-frame-index] to be used in Cell list of regions to be used in SGofCPLRTzpianohingeclosed **CPLCA**: Connnecting contour availabe to connct stator contour CPLA to the joint **CPLCB**: Connnecting contour availabe to connct rotor contour CPLB to the joint **wlim**: cell ist of angle interval or singe angle interval [wmin wmax]

CPLRinserthingelist

returns CPL regions linked by Frames prepared for extrusion

In: **CPL**: Contour in which the hinges are to be inserted **T**: cell list of Frames. Empty Frames mean interactive selection, Numeric scalar means n interactive frames **R**: same as in SGcylhinge but only R(1) used **wlim**: angle limit [amax] or [amin amax] with respect to the x axis of the frame **"both"**: is helpful to avoid material reduction when there are steep angles as in "CPSinserthingelist" **"debug"**: if used, the individal steps are shown as in "CPSinserthingelist" **"nocut"**: if used, the frames do not cut trough as in "CPSinserthingelist" **"close"**: if used, the contour finally is closed again (not implemented yet)

Out: **CPLR**: Cell list of regions to be used in SGofCPLRTzpianohinge **T**: cell list of frames to be used in SGofCPLRTzpianohinge **R**: radius used for joints to be used in SGofCPLRTzpianohinge **EEL**: Extended edge list [stator-index rotor-index signed-frame-index] to be used in SGofCPLRTzpianohinge **CPLCA**: Connnecting contour availabe to connct stator contour CPLA to the joint **CPLCB**: Connnecting contour availabe to connct rotor contour CPLB to the joint **wlim**: cell ist of angle interval or singe angle interval [wmin wmax]

CPLRintersectsweep

unites interseceted contours of a linkage

In: **CPLR**: cell array with link contour in pose posi **Pall**: matrix for joint movement m x 2 x k ; with m joints and k poses **ELC**: cell array explaining whoich joints belon to the poses **posi**: current pose index for CPLR **ai**: indices of contours which are intersected with bi **bi**: indices of contours which are intersected with ai **buf**: [abuf bbuf] default is [0 0]; is scalar than it is copied to both sides **"debug"**: if used, the design process is shown

Out: **CPLU**: united interseceted contours

CPLRmeltbyCycles

removes closed 2-loops and 3-loops of contour even of non-binary joints

In: **CPLR**: cell array of link contours **PLA**: option point list with terminal and joint coordinates **styl**: 'union','fill'; default is union **"debug"**: if used, the steps are shown

Out: **CPLR**: finally reduced CPLR **PLA**: optional list of points

CPLRmeltbyIndex

simple fnct for melting contours in a cell array of linkage contours by a specified index

In: **CPLR**: cell array of contours **PLA**: intersection center point list **indx**: index of intersecting contours to melt **styl**: if used, additional steps are shown **"debug"**: if used, additional steps are shown

Out: **CPLRN**: reduced and resorted cell array of contours

CPLRmove

Animates a moving linkage using PLofLinkagecouplercurve for experiments

In: **CPLR**: CPLR or ELC describing a linkage **PLA**: List of joint coordinates in a specific pose **BASE**: pair of joint that defined the base of the linkage **MOVE**: pair of joint that defined the crank/drive mover of the linkage **diffw**: angle intervall wrt to the pose defined by MOVE(1)=>MOVE(2); default is [-2*pi+1e-6…+2*pi-1e-6] **"STL"**: **"angle"**: **"debug"**:

CPLRofELC

create a minimal contour set of link joint sets

In: **ELC**: cell array of link with owned joints **PLA**: point coordinates of joints **R**: Radius for joints; the last give one is copied to the rest; default is 2 **"shape"**: default is 'spike' - each link is designed using "CPLrack4PL"

Out: **CPLR**: Cell array of contours **ELT**: Edge list truss

CPLRofELT

melts based on ELT and PLA and optional CPLR the contours

In: **ELT**: EL truss n x [start-joint end-joint contour-index] **PLA**: PL of the joints m x [x y] **R**: CPLR OR Radius! existing CPLR if the contours should be melted **"style"**: style tu fuse point using CPLrack4PL default is 'delaunay'

Out: **CPLR**: CPLR according to the new ELT **ELT**: ELT with same start-joint and end-joint by changed contour index **a**: sort index used for sorting CPLR

CPLRofPLEL

creates a set of linkages based on on a PL and EL and radius information

Out: **varargout**: cell array of link contours

CPLRofPLELplain

creates a set of linkages based on on a PL and EL and radius information

In: **PL**: Point list or popint sequence list [P1;P2;P3...] or [P1 P2; P2 P3] **EL**: Edge list if a pure point list was used **R**: Outer and optional inner radius; default is [2 1] **"arc"**: if used, the following parameter will force link contours with this radius **"nsin"**: if used, the following value creates sinusoidal link contours with this number of periods; default is 0

Out: **CPLR**: cell array of link contours

CPLRofPose

creates a CPLR of a pose or pose solution

In: **PS**: Pose struct at least using **pslt**: Number of pose; default is 1 **aps**: actual pose solution

Out: **CPLR**: CPLR{1}=Base, CPLR{2}=Effector, CPLR{3}= **PLA**: [A0;B0;B1;A1] - wrt pose pos **PS**: modified pose struct with corresponding pose pslt to Solution aps

CPLRposechange

adjust the pose of the individual link contours in CPLR to fit into a new index of the joint coordinates

In: **CPLR**: n x {CPL} cell array of link contours **Pall**: m x 2 x k Point coordinates of m Joints in k poses **ELC**: n x {} cell array containing the joint in each link **posi**: pose index (original) **newi**: new pose index; if new==0; TL returns the joint coordinates for

Out: **CPLRN**: Cell array of contours in the new pos **PLN**: Point coordinates in the new pose **EL2**: joints used for transformation **TL**: Either point coordinates of EL2(:,2) is case of new==0 or frame list for new pose **TLpos**:

CPLRposesetbycrank

Sets a linkage driven by two base points an a crank link to a new crank angle

In: **CPLR**: cell array of link contours in a defined pose **PLA**: joint coordinates in this defined pose **BASE**: two joints of PLA as base points such as [1 2] **CRANK**: two joints of PLA as crank such as [2 3] **w**: desired angle of CRANK link in absolute coordinates **dw**: optional angle interval for crank if the output should be used for 2D/3D design **"debug"**: if used, the design is shown

Out: **CPLR**: cell array of link contours in the new pose **Pall**: All joint coordinates during movement **Wall**: All koint angles during movement **ELT**: Extended edge list of truss **ELC**: array which joint is owned by the links **wlimfe**: angle interval limitation of crank wrt to the defined angle **posi**: index in Pall of the pose

CPLRsample

returns sample contours for the linkage analysis

In: **Nr**: selected number to show or return if scalar **"melt"**: if used, there was already a proprocessing in CPLRofCPL using melt

Out: **CPLR**: cell array of contour links **PA**: point coordinates **txt**: description text of the sample

CPLRshapelink4pose

shapes a link contour to use other contours a conotur path space or to avoid a predefined additional contour

In: **CPLR**: n x {CPL} cell array of link contours **Pall**: m x 2 x k Point coordinates of m Joints in k poses **ELC**: n x {} cell array containing the joint in each link **posi**: [pose index (original), shape index (desired pose)] default shape index = posi **Aind**: index of CPLR contour to be sphaped as bar with radius R **Bind**: Index list of CPLR contours that should be used as path space OR CPLW area that should be avoided **R**: Radius of the contour link bar geometry; default is 2 **"debug"**: if used the design steps are shown

Out: **CPLR**: n x {CPL} cell array of link contours with modified contour Aind

CPLRsimplifyEL

unifies contours whos corresponding edge list rows corrensponds to the same joints' index and reorders the edge list

In: **CPLR**: cell array with n contours of a linkage **ELO**: edge list [nx2] between points that correspond to the n contours in CPLR **PLO**: Point list with m points that correspond to the m points used to create the contours **"debug"**: if used, the step are shown to modify the edge list and contour list and order

Out: **CPLR**: shortend and reordered cell array of contours **ELN**: shortend and reordered edge list fitting to the contours

CPLRsubtractjointsweep

modifies the link contours which are cut or blocked by static/moving joints contours

In: **CPLR**: cell array of link contours of a linkage **Pall**: Pall complete points list of m joints of all k poses **ELC**: cell array containing for each join the list of joints inside this contour **posi**: pose number that corresponds to the pose contours in CPLR **R**: Radius circular joints | cage wall | intrusion; default is [2 0 0] **"debug"**: if used individual steps are shown

Out: **CPLRN**: The modified and newly planned link contours

CPLRsubtractjointtrack

modifies the link contours which are cut or blocked by static/moving joints contours or static/moving link contours or creates minimal link contour if requested

In: **CPLR**: cell array of link contours of a linkage **Pall**: Pall complete points list of m joints of all k poses **ELC**: cell array containing for each join the list of joints inside this contour **posi**: pose number that corresponds to the pose contours in CPLR **R**: Radius circular joints | cage wall | intrusion; default is [2 0 0] **prios**: priority contours (highest priority/static first); default is [1:numel(CPLR) **shaps**: contour index list to shape; default is empty **npath**: if used inividual steps are shown.

Out: **CPLRN**: The modified and newly planned link contours

CPLRsubtractTLbs

CPLR collision free shaphing and forming of contour links during movements

In: **CPLR**: n row cell array of initial contours of a linkage **Pall**: point coordinate list of all joints of the linkage during movement **ELC**: n row cells containing the joint indices that are part of the link contour **posi**: start pose index of Pall to identify the start pose index **Aind**: Contour indices in CPLR to subtract from during motion **Bind**: Contour indices in CPLR to subtract from Aind during motion **R**: Default is 0; if used the Contour in Aind are collision free shaped between the points **"debug"**: if used the contour desige is shown step by step

Out: **CPLR**: CPLR final contour after processing

CPLRunionsweep

unites swept contours of a linkage

In: **CPLR**: cell array with link contour in pose posi **Pall**: matrix for joint movement m x 2 x k ; with m joints and k poses **ELC**: cell array explaining whoich joints belon to the poses **posi**: current pose index for CPLR **ai**: slection of contours that should be part of the result **buf**: buffer to increase the size for fittings **"debug"**: if used the individual steps are plotted

Out: **CPLU**: united sweept contours

EELofCPLR

Creating the extended edge list and the graph using the joint indices as weight

In: **CPLR**: cell array of link contour **PLA**: base or crossing points that cannot be detected by intersection **tol**: tolerandce to map coordinates; default is 0.95 **"debug"**: if used, additional steps are shown **"original"**: if used, only the given joint coordinates PLA are used and no intersection centers are added

Out: **EEG**: Extened Edge list of a contour node graph [start end joint index] **PLA**: Point coodinates of the 3rd row of the EEG **G**: contour node graph | edges are joints | weight is PLA index **loopC**: ascending in length node cyclebasis of G (showing the contours) **loopE**: ascending in length edge cyclebasis of G (showing the weights) **ni**: length of den cycles in loopC and loopE **ELT**: Edge list truss (complete bar system with 3bar cycles)

ELCofELT

create a ELC cell array from ELT

In: **ELT**: Edge list truss [joint-A joint-B contour-E] | truss have binary rods

Out: **ELC**: edge list contour cell array - of a truss

ELTofCPLR

Short version that creates only the truss edge list from a contour and joint coordinates

In: **CPLR**: cell array with contours and intersecting areas **PLA**: Point list for joint to get more accurate intersection centers **"original"**: if used, only the original joint coordinates PLA are used (no added intersections)

Out: **ELT**: Edge list of a truss; 3rd coll are link contours **PLN**: list of Points may differ if there are more crossing points **warn**: true if joint count was reduced because some joint coordinates are not linked **ELC**: edge list of the link contours

ELTofELC

returns the edge list fo the truss from the Link contour owened joint list ELC

In: **ELC**: cell array of the link owned joints **PLA**: Ppoint coordinates of joints, required for triangulation and collinearity

Out: **ELT**: Edge list of the truss

wlim4CPLRinserthingeclosed

returns the absolute joint angles wrt a given Pose of a linkage used in CPLRinserthingeclosed

In: **CPLR**: cell array of link contours (only used for visualiziation debug or nargout==0) **PLA**: PLA joint coordinates in a predefined pose **ELT**: Fully triangualted connection of joints (EELofCPLR in PLofLinkagecouplercurve) **Pall**: Fully list of changing PLA over the movement intervall (PLofLinkagecouplercurve) **ELC**: Fully list of contours and moving joint indices with this contour (EELofCPLR in PLofLinkagecouplercurve) **Wadd**: optional n x 4 list of [Cindex Cindex w-min w-max] for absolute joint openings in the current pose **"debug"**: if used the result is animated and CPLR and ELC are required

Out: **wlimA**: absolute opening angle of joints wrt a given pose required for CPLRinserthingeclosed **TLPLA**: List of absolut coordinate frames for using the opening angles wlimA within CPLRinserthingeclosed **NPind**: index list for poses for each absolut joint [i i1 i2], TofPose(i1-i, i2-i) **NCind**: IMPORTANT index list for joint indices for each conotour [ci j1 j2 j.. jq], TofPose(j1, j2)

6.6.2 Pose Definition and Manipulation [guide]

Functions in this section:

A1ofA1B1GPLPose — returns the list of the crank attachment points for poses

BBofLPS — returns the Bounding box of a Linkage Polyshape struct

center3Pose — return center points for 3 poses (position and orientation)

center3Poseeffektorgrid — returns the rotations centers for grid points of the effector contour

center4Pose — return center points for 3 poses (position and orientation)

colofPose — returs colors from Pose and 4Bar standard colors

CPLofLPS — returns unsorted the CPLs of a LPS

getPoseconstraintvalue — returns whether a condition string is used within a Pose field

isPoseconstraint — returns whether a condition string is used within a Pose field

KDWofPoseattachments — returns the values for K, D, and W for given attachment points

KDWPoseinsideCPL — creates a grid point list and the values for KDW

KDWPosevariation — returns three variant lists for center3Pose as a first recommendation

LPSanimate — animation of a Linkage Polygon structure after interactive definition of the order

LPSconnect — searches for a connection between two links using LPSjointmatrix

LPSjointmatrix — returns a matrix of revolute joints

LPSofgca — Defines a new SG-Lib class LPS Linkage Polyshape and loads it from the gca polygons

LPSplot — plots a linkage polyshape struct

LPSpunchholesui — EXPERIMENT for testing LPSofgca and LPSplot

PLofPosecentrodes — returns fixed centrode (Rastpolbahn) and moving centrode (Gangpolbahn) of a fourbar linkage solution

PoseaddGPL — creates a Grid inside/on a contour

Poseflip — flips the direction of pose vectors in a Pose struct

Posemagnify — scales the size of Pose struct dimensions

Posenorm — changes the length of pose vectors to a common length and shifts the start point

PoseofCPLcutui — allows the interactive shaping and move of poses of contours

PoseofCPLui — allows the interactive move of poses of contours

Poseofimageui — interactive fnct to defines three poses in a drawing image

PoseofPoseui — allows the interactive move of poses of contours that exist already in a Pose struct

PoseofT — returns pose struct from a list of HT matrices either 3x3 or 4x4

Poseorderchange — returs a pose struct with poses in changed order

Poseplot — plots a pose struct

Poseplotsolution — plots A0, B0, and geometry and animation of fourbar linkage design

Poseplotspace — plot the different contours of a pose in different colors

PosereadAPD — reads in a pose struct for syntheses tasks

PosesampleCover — creates a pose struct sample for a box hinge lid

PosesampleCTFieldgenerator — creates a sample to design a turnable field generator fixture

PosesampleHook — Pose sample for Hook Movement Task

Posesampleupdate — inserts a pose struct and a name into the posesamles.mat file

PosesampleWalk — pose sample will be integrated into Posesample

PosesolutionofA0B0A1B1 — returns a Pose Struct similar to a fourbar synthesis solution simply from fixed Values

Posestripfields — removes alls fields that seem results not conditions from pose synthesis fncts

PosetransrelCPLE — shifts pose definition and CPLE using an HT transformation

PosetransrelGPL — shifts the poses into an effector grid point

PosetransT — transform the pose vectors in a Pose struct

Posetransui — turns and moves interactively poses and recalculate fourbar linkage solutions

Poseui — returns a pose struct

PosewriteAPD — writes a pose struct as ascii text file

TofPose — returns 3x3 HT matrices for planar transformation in 2D from pose list

TofPoseattachmentpoints — returns the HT matrices from A1 to Pose and B1 to Pose

TofPoseGPL — returns the frames for A1 and B1 from a Pose and 2 GPL attachment points

Detailed description

A1ofA1B1GPLPose

returns the list of the crank attachment points for poses

In: **A1**: Single A1 of a fourbar linkage **B1**: Single B1 of a fourbar linkage **GPLA**: A1 relative to Pose **GPLB**: B1 relative to Pose **TT**: Single Pose or List of Poses

Out: **A1L**: Attachment Point position for the Poses **B1L**: attachment point B1 position for the poses

BBofLPS

returns the Bounding box of a Linkage Polyshape struct

In: **LPS**: Linkage Polyshape struct

Out: **bb**: bounding box **CPL**: CPLincluding all CPLs of LPS

center3Pose

return center points for 3 poses (position and orientation)

In: **A**: List of at least 3 points that define a pose **B**: List of at least 3 points that define a direction in combination with A **k**: list of checking values for a point along the line A+k*(B-A)/|(B-A)| **dall**: displacement orthogonal to pose line; default is 0 **wall**: turning angle of the pose relative to pose; default is 0; **CPL**: Optional area for solutions **move2**: shift of pose 1 along its own direction; default is 0

Out: **S**: [k C r d w] **rs**: real solutions if CPL is used **Sall**: All Solutions S=Sall(rs,:) **PSall**: All Poses **dw**: Angle step/deviation used between the pose variants

center3Poseeffektorgrid

returns the rotations centers for grid points of the effector contour

In: **A**: Start point or Pose **B**: End Point or Pose **CPLE**: Optional contour or size of squared box **GPL**: number of point (integer) or distance (if not integer) **m**: distance to outer contour; default is 0 **exactCPL**: true if no outer boundary but exact CPLE is used

Out: **GPL**: Center Point and Radius **S**: Grid Coordinates relative to CPLE **P1**: Grid Coordinates relative to Pose 1 **P2**: Grid Coordinates relative to Pose 2 **P3**: Grid Coordinates relative to Pose 3

center4Pose

return center points for 3 poses (position and orientation)

In: **A**: List of at least 3 points that define a pose **B**: List of at least 3 points that define a direction in combination with A **k**: list of checking values for a point along the line A+k*(B-A)/|(B-A)| **dall**: displacement orthogonal to pose line; default is 0 **wall**: turning angle of the pose relative to pose; default is 0; **CPL**: Optional area for solutions **err**: error tolerance for matching the 4th pose; default is the k step size

Out: **S**: [k C r d w] **rs**: real solutions if CPL is used **Sall**: All Solutions S=Sall(rs,:) **PSall**: All Poses **dw**: Angle step/deviation used between the pose variants

colofPose

returs colors from Pose and 4Bar standard colors

In: **cs4Bar**: color string using 4Bar link/area desciption letters such as 'E--' for effector dashed

Out: **cs**: that uses standard Matlab colors such as 'r--'

CPLofLPS

returns unsorted the CPLs of a LPS

In: **LPS**: Linkage Polyshape **nn**: selected Polygons; default is 1:size(LPS,1) **res**: resultion für CPL; default is 0.025mm

Out: **CPL**: CLPs of an LPS

getPoseconstraintvalue

returns whether a condition string is used within a Pose field

In: **PS**: Pose including a field "cnstr" for constraints OR string **tags2find**: condition string to find a value constraint

Out: **found**: true or false or a value

isPoseconstraint

returns whether a condition string is used within a Pose field

In: **PS**: Pose including a field "cnstr" for constraints OR string **cond**: condition string to find in constraint

Out: **found**: true or false

KDWofPoseattachments

returns the values for K, D, and W for given attachment points

In: **A**: Pose including PS.A, PS.B, PS.A1, PS.B1 **B**: optional Pose Values for End Points **A1**: Attachment Point for A1 [X Y Pose#] **B1**: Attachment Point for B1 [X Y Pose#]

Out: **K**: [ka kb] on pose line as values relativ distance to pose starting point **D**: displacement parallel to pose orientation **W**: turning angle BEFORE displacement realtive to origin **err**: error value

KDWPoseinsideCPL

creates a grid point list and the values for KDW

In: **CPL**: Effector Contour of a Pose **d**: distance between the grid points; default is s/20 **m**: either buffer around CPL or explicit contour for grid area

Out: **KDW**: [K D W] for GPL **GPL**: GPL inside contour

KDWPosevariation

returns three variant lists for center3Pose as a first recommendation

In: **PS**: Pose to be searched using center3Pose **nk**: number if k variants; default is **nd**: number of d variants; default is 5 **dw**: degree resolution ; default is 15

Out: **K**: List for k values; default are 40. **D**: List for d values; default are 5 **W**: List for values; ; default are 24 for 15 degree

LPSanimate

animation of a Linkage Polygon structure after interactive definition of the order

In: **LPS**: Linkage Polygon structure **ORD**: optional known order of the CPLs in LPS

Out: **PS**: Connection Matrix, same format as LPSconnect **CC**: Connection Matrix, same format as LPSconnect

LPSconnect

searches for a connection between two links using LPSjointmatrix

In: **LPS**: Linkage polygon structure **SI**: Start index in LPS **EI**: End index in LPS

Out: **CC**: CC=[connection-number | Link1-index | Link1-hole-index | Link2-index | Link2-hole-index]

LPSjointmatrix

returns a matrix of revolute joints

In: **LPS**: Linkage polygon structure **thr**: threshold for linage overlap

Out: **IPD**: index point list (doubles only) **IPL**: index point list (complete)

LPSofgca

Defines a new SG-Lib class LPS Linkage Polyshape and loads it from the gca polygons

In: **h**: handle to axis; default is gca

Out: **LPS**: Linkage polyshape of plot(h)

LPSplot

plots a linkage polyshape struct

In: **LPS**: Linkage Poly Shape **"CPL"**: **"FRAME"**: if used, the link contours (CPL) are plotted (default on) **"TEXT"**: if used, the joint frames are plotted

Out: **h**: handle to graphics

LPSpunchholesui

EXPERIMENT for testing LPSofgca and LPSplot

In: **LPS**: Linkage polygon structure or handle to gca **RRio**: Bore hole radius; default is [2 Nan] **PPL**: optional point list of joint positions; if given, holes are placed non-interactively; default is empty

Out: **LPS**: Linkage polygon structure

PLofPosecentrodes

returns fixed centrode (Rastpolbahn) and moving centrode (Gangpolbahn) of a fourbar linkage solution

In: **PS**: Pose incl. GPL and GAL **aps**: index in GAL or [iA iB] for GAL **npnt**: number of points of the curve **ncur**: number of curves desired; default=0 (Pose Coupler Curve);(will be removed)

Out: **PLF**: Point list of Coupler Points Movement **PL0**: Transformation matrix list of Coupler Points Movement **T0**: angle list of crank returned by invkinplan2 **wcr**: Transformation matrix list of joint A0 (only rotation) **TA0**: Transformation matrix list of joint B0 (only rotation) **TB0**: Transformation matrix list of Attachment point A1 (circular Movement) **TA1**: Transformation matrix list of Attachment point B1 (circular Movement)

PoseaddGPL

creates a Grid inside/on a contour

In: **PS**: Pose with Contour CPLE **nr**: Number of Grid points to be achieved **bf**: distance to outer contour; default is 0; **fl**: if false; the conotur is filled; default is false

Out: **PS**: Pose with CPLE and GPL

Poseflip

flips the direction of pose vectors in a Pose struct

In: **PS**: Pose struct with A and B fields to describe Poses **nn**: list of poses to flip; default is all

Out: **PS**: Pose with fliped direction

Posemagnify

scales the size of Pose struct dimensions

In: **PS**: Pose **m**: magnification factor **all**: unused flag; default is false

Out: **PS**: Magnified Pose struct

Posenorm

changes the length of pose vectors to a common length and shifts the start point

In: **PS**: Pose.A and Pose.B **l**: desired length; default is 1 **ka**: starting offset relative to A

Out: **PS**: Adjusted Pose.A and Pose.B **e**: direction vector **o**: orthogonal vector

PoseofCPLcutui

allows the interactive shaping and move of poses of contours

In: **CPL**: Effector contour **CPLW**: Background Contour

Out: **PS**: Pose Struct include A, B, CPLE, CPLB, CPLW

PoseofCPLui

allows the interactive move of poses of contours

In: **CPLP**: Effector contour **CPLB**: Background Contour **CPLW**: Warning/Blocked Contour **CPLM**: mounting/mask contour (CPLM); default is []

Out: **PS**: Pose Struct include A, B, CPLE, CPLB, CPLW

Poseofimageui

interactive fnct to defines three poses in a drawing image

In: **I**: image with a hand sketch or a simple drawing, or also a Pose in a second try

Out: **PS**: Pose struct with start points, end points B and effector CPL **CPLB**: separated Background contour that was not processed as pose

PoseofPoseui

allows the interactive move of poses of contours that exist already in a Pose struct

In: **PS**: Pose Struct include A, B, CPLE, CPLB, CPLW

Out: **PS**: Pose Struct include A, B, CPLE, CPLB, CPLW

PoseofT

returns pose struct from a list of HT matrices either 3x3 or 4x4

In: **T**: List of transformation matrix either [3*n x 3], [3,3,n] [4*n,4],[4,4,n] **CPL**: Optional Contour to save as PS.CPLE

Out: **PS**: Pose = PS.A, PS.B (X-axis) PS.C (y-axis)

Poseorderchange

returs a pose struct with poses in changed order

In: **PS**: Pose struct **c**: change pair

Out: **PS**: New Pose struct

Poseplot

plots a pose struct

In: **PS**: PS.A = Starting Points, PS.B = End Points **l**: optional length **col**: optional color **CPLE**: optional shape contour **k**: displacement along axis; default is 0 **ts**: text size of descriptor; default is 0 **psel**: selected poses to plot; default is 1:size(PS.A,1) **"nocorr"**: flag; if set, skip correcting base points A0/B0 to the center of 3 poses

Out: **h**: handle to plot **l**: selected pose length if cal. automatic

Poseplotsolution

plots A0, B0, and geometry and animation of fourbar linkage design

In: **PS**: Pose struct containing field "solut" **sel**: number, sequence or constraint string to select; default is all **show**: 'plotit', 'animat" "both" **nums**: number of shown solutions [10 1]=>10 automatic [10 -1]=>10 interactive **Roi**: Radius out and inner; default is []==automatic

Out: **h**: handle to plot **oldtitle**: this is the original title before modified by Poseplotsolution

Poseplotspace

plot the different contours of a pose in different colors

In: **PS**: Pose struct containing CPLM, CPLW, CPLB

Out: **h**: handle to graphics objects

PosereadAPD

reads in a pose struct for syntheses tasks

In: **fname**: file name for read

Out: **PS**: struct with poses content **fname**: full file name used

PosesampleCover

creates a pose struct sample for a box hinge lid

Out: **PS**: Pose sample for a box hinge lid

PosesampleCTFieldgenerator

creates a sample to design a turnable field generator fixture

Out: **PS**: Sample pose struct

PosesampleHook

Pose sample for Hook Movement Task

Out: **PS**: returns the Pose task for a hook

Posesampleupdate

inserts a pose struct and a name into the posesamles.mat file

In: **Name**: Name to be used by Posesample(<Name>) **PS**: Pose struct **repl**: true if it should be replaced if existing already

Out: **new**: true if added

PosesampleWalk

pose sample will be integrated into Posesample

Out: **PS**: Posesample(12)

PosesolutionofA0B0A1B1

returns a Pose Struct similar to a fourbar synthesis solution simply from fixed Values

In: **A0**: Base point A0 **B0**: Base point B0 **A1**: Coupler point A1 **B1**: Coupler point B1 **Tref**: Pose reference in this configuration

Out: **PS**: PS containing 1 Pose, four joints, relative points to define center

Posestripfields

removes alls fields that seem results not conditions from pose synthesis fncts

In: **PS**: Any Pose struct.

Out: **PS**: Struct without resulting fields

PosetransrelCPLE

shifts pose definition and CPLE using an HT transformation

In: **PS**: Pose with A and B and CPLE **TN**: relative movement of the CPLE origin

Out: **PSN**: Pose with shifted A and shifted B and shifted CPLE

PosetransrelGPL

shifts the poses into an effector grid point

In: **PS**: Pose **ng**: optional number of grid points **nn**: selected grid point; default is nearest

Out: **PS**: Resulting Pose

PosetransT

transform the pose vectors in a Pose struct

In: **PS**: Pose struct with A and B fields to describe Poses **nn**: list of poses to flip; default is all **T**: Transformation matrix **rel**: if true; T is not rel [0 0] but the origin of the pose; default is false

Out: **PS**: Modified Pose strcut (just A and B are changed)

Posetransui

turns and moves interactively poses and recalculate fourbar linkage solutions

In: **PS**: Pose **CPLE**: Optional size or contour of effector **GPL**: Grid number or Grid list **m**: alternative buffer size of **gp**: Grid point density **lim**: currently unused

Out: **PS**: Pose with changed Lists for A, B, GAL

Poseui

returns a pose struct

In: **m**: number of poses; default is 3 **res**: resolution; default is 2% of sofBB **ax**: axis; default is unchanged

Out: **Pose**: Pose.A = Start Points of pose, Pose.B = End Points of pose

PosewriteAPD

writes a pose struct as ascii text file

In: **PS**: Pose **fname**: filename default is expname **"nocheck"**: flag; if set, skip the check for required fields A and B

Out: **[fullname]**: full file name for read

TofPose

returns 3x3 HT matrices for planar transformation in 2D from pose list

In: **AL**: Point list [n x 2] for points A or a struct with fields A and B **BL**: Point list [n x 2] for points B **ind**: selected indices; default is all

Out: **TAi**: Transformation Matrix in A for each Pose; ex = B(i,:)-A(i,:); t=A(i,:) **ww**: rotation angle of ex wrt to the worl coordinate system (since SGLib 5.6)

TofPoseattachmentpoints

returns the HT matrices from A1 to Pose and B1 to Pose

In: **PS**: Absolute Pose coordinates or complete pose consisting of PS.A, PS.B, PS.A1, PS.B1 **A1**: absolute coordinates of A1 when P1 is reached **B1**: absolute coordinates of B1 when P1 is reached

Out: **TBP**: TAP=TA1\T0; to transform Pose origin coordinates into absolute Coordinate v TB1*TBP **TAP**: TBP=TB1\T0; to transform Pose origin coordinates into absolute Coordinate v TA1*TAP **TA1**: TA1=TofPose(A1,B1); % Simple frame in A1 with direction to B1 **TB1**: TB1=TofPose(B1,A1); % Simple frame in B1 with direction to A1

TofPoseGPL

returns the frames for A1 and B1 from a Pose and 2 GPL attachment points

In: **TP**: World frame of Pose **A1**: Point A1 in GPL **B1**: Point B1 in GPL **CD**: CODE for meaning of A1 and B1 'GLP'

Out: **TA1**: World frame of A1 **TB1**: World frame of B1 **TP**: World frame of Pose

6.6.3 Four-bar Synthesis and Analysis [guide]

Functions in this section:

checkfourbar123solutions — just checks an limits solutions of 1-2-3 Pose syntheses

checkfourbar1Poseattachpermutation — Performs a fourbar-linkage syntheses based on ONE defined Pose

checkfourbar2Poseattachpermutation — Returns solutions for the 2 Pose Sythesis

checkfourbar3Pose — checks whether a fourbar joint solution exists for results of center3Pose

checkfourbar3Poseattachpermutation — returns only subset of permutations of grid points to achieve a real fourbar linkage solution

checkfourbar3Posepermutation — calculates from a center3pose solution all possible permutations and the fourbar parameters

checkfourbarcouplerlength — checks the solutions of invkinplan2 for a fourbar linkage calculation

checkfourbarNPose — checks whether a fourbar joint solution exists for results of center3Pose

checkfourbarsegment — calculates the angle intervals for a fourbar linkage and appoint point to the intervall

CPLfourbarblockswing — creates the contour that limit the swing movement

CPLfourbarcontour — Contour that fullfills a fourbar condition

CPLfourbarlinkage — creates for a specified fourbar linkage the cpls for crank, coupler, swing

CPLfourbarlinkage2 — creates for a specified fourbar linkage the cpls for crank, coupler, swing

CPLfourbarlinkagehull — returns the hull shapes of crank, and swing

CPLfourbarmoved — returns a different pose for an fourbar defined by 4 pose points

CPLofPosefourbarshape — returns the basic shapes of a pose fourbar linkage solution

CPLRmeltbyCJMAT — melts link contours if they have 2 or more common joints

CPLRmeltbyJCIRC — melts link contours if 3 separated contours a are linked by 3 binary joints

CPRofPosefourbar — returns center point list and radius list for a fourbar linkage for a given GPL and GAL

CPSinserthinge — 1st solution to integrate revolute joints into CPL/CPS to be extruded later to SG

CPSinserthingelist — inserts a list of hinge geoemtries into a CPL/CPS. The first frame cuts, the following not

DoFGruebler — Gruebler-Chebychev-Equation to calculate the DoF

DoFGrueblerJacobiRank — DoF by the jacobi Rank (only double joint rods are considered)

ELlongestpath — returns a list of unique directed kinematic chains from an edge list

ELofCPLRconnections — returns the contour edge list for the contour defining point connections and the join edge list describing wich contours are connected by the join

findCollinearEdges — returns the groups with collinear edges

fourBarCLLL2SGdesign — Converts a contour layered link list into a real solid geometry construction to 3D print

fourbarcollABGPL — returns the crank angle interval for an object liked to a Pose

fourBarLinkage — multi purpose fnct for a 4-Bar-Linkage

fourBarLinkageanimate — animates a fourbarlinkage based on 4 Points, a Frame, and a Contour

fourBarLinkageCPL — multi purpose fnct for a 4-Bar-Linkage

fourBarLinkageplot — plots and animates the movement of a fourbar linkage

fourBarLinkageplotanim — plots and animates the movement of a fourbar linkage

fourBarLinkageplotanimui — plots and animates the movement interactively of a fourbar linkage

fourbarplotABGPL — simply plots a fourbar linkage based on points - recommended in SG-Lib 5.1

fourBarplotPoints — just plot a specific fourbar configuration based on points

fourBarposeCPLmotion — returns the motion of all parts of a fourbar linkage

fourbarposeCPLshaping — returns collision free path shaped contour for a specific fourbar linkage

fourBarposelayering — Creates the layering and the shaping of all 8 elements of a Fourbar linkage

fourBarposeplotsolution — Standard procedure to animate fourBarposesyntheses results

fourbarposesearchcouplercurve — sorts 4bar solutions wrt minimal distance of coupler path to a given set of points

fourBarposesortsolution — sorts the solutions of the fourbar struct by some criteria and even selects

fourBarposesyntheses — fourbar pose synthesis

invkinplanarlinkage — inverse kinematics of a linkage in absolute coordinates by two base point coordinates and one absolute angle (-pi/2 .. pi2/2)

invkinplanarlinkagerot — inverse kinematics of a linkage in absolute coordinates by two base point coordinates and one absolute 360 deg angle

invkinplanarlinkagerotK — inverse kinematics of a non rigid linkage in absolute coordinates by two base point coordinates and one angle

jacobiantruss — jacobian of a truss

PLcircleoval4obstacle — similar to PLcircleoval but is considering static and moving obstacles

PLELsample — returns PL and EL to create linkage networks using CPLRofPLEL

PLofCPLRcrossings — returns a point list of contour crossing that could be used as joint coordinates

PLofFourbarcouplercurve — Creates the coupler courve for an Fourbar given by four points

PLofLinkagecouplercurve — returns for a specific planar linkage, base points and drive, the limitation angles and movements of all joints

PLofPall — Auxiliary fnct to get points of a specific joint for single, several or all poses

PLofPosecouplercurve — returns the path and the matrix list of the pose movement

poletriangle3Pose — calculates the edges of the pole triangle of three poses

Posefourbaranglelimit — limits the angle range of a specific fourbar solution

Posefourbarangles — Returns the angular position of the crank for the positions used in the four-bar linkage.

Posefourbaranimate — animates the fourbar linkage given Pose parameter

Posefourbarcheck — shows the shape of effector, crank and swing

PosefourbarCPLE — creates a CPL for an Effector containt 3 Points

Posefourbarplotcouplercurve — simply plots a coupler curve for fourbar linkage

Posefourbarposelimit — finds the angle limits for a fourbar movement

Posefourbarsearchcouplercurve — selects coupler courves based on the coupler curve

PosefourbarselectAP — simply select a solution from the Pose attachment point list

Posefourbarselectcouplercurve — selects coupler courves based on the coupler curve

Posefourbarselectui — interactive tool to plot poses and to select two attachment/base points for a fourbar linkage

Posefourbartorquetrans — returns the torque transmission factor for crank to swing and crank to pose

Poseoffourbarlinkage — creates a Pose struct with 3 or more poses for given fourbar linkgae parameters L1, L2, R, D/[A0;B0]

poseofPSanalyze — returns a Pose and a CPL from an analyzeed 3 Pose Polygon

realfourbarsolutions — creates an index list to map fourbarlinkage solution points to movement types

selectLPSui — selects interactively Frames or Frame connections from a lingage Polygon Struct

SGofCPLzpianohingelist — extrudes a solid from a single contour and inserts a set of piano hinges after cutting the contour into pieces

SGpianohinge — creates a piano-hinge style revolute joint and attaches predefined contours as link arms

textEELPL — fnct to plot the exteded contour-node joint-edge list with coordinates or joint indices

TLofPalledge — Auxiliary fnct to get points of a specific joint for single, several or all poses

Detailed description

checkfourbar123solutions

just checks an limits solutions of 1-2-3 Pose syntheses

In: **PS**: Pose struct with fields CPLM, CPLE, CPLW, AND solut **cnstr**: constraints such as "ISROT" **grdnx**: [crank dist, link dist, effector dist] buffers used for collision checks; default is [.01 .01 .01] **show**: 'plot', 'animat', 'both'

Out: **PS**: Pose struct with reduced number of solutions

checkfourbar1Poseattachpermutation

Performs a fourbar-linkage syntheses based on ONE defined Pose

In: **PS**: Pose struct including CPLE, A, B **psel**: selector of pose **grdnx**: [grid nr, link radius, effector radius]; default [30 2.5 0.1] **solu**: [A0i B0i A1i B1i] in CPLM and CPLE, if negative animated **cnstr**: optional constraint string **shows**: if true, each found solution is displayed; default is true

Out: **PS**: Pose struct

checkfourbar2Poseattachpermutation

Returns solutions for the 2 Pose Sythesis

In: **PS**: Pose struct **psel**: Selection of poses to fix; typically the end poses **m**: [GridNr distlink dist base] **oslim**: angle limits [start end] for the selected poses; default derived from psel second row **fille**: if true, the effector grid points are filled inside the contour; default is false

Out: **PS**: Pos including field 'solut'

checkfourbar3Pose

checks whether a fourbar joint solution exists for results of center3Pose

In: **A**: Starting Points of 3 Poses **B**: Ending Points of 3 Poses **S**: Solutions comming from center3Pose **dws**: selected pair of distance and angle **k**: selected values for k; default are first and last of S **pl**: length of pose **ki**: Direct access to k values

Out: **ci**: configuration for pose of a fourbar linkage **s**: rotation segment for earch pose **ord**: movement order of poses; right-hand-rule **corg**: original configuration for each pose **wres**: angle segments; 0 => no solution exist; [] => no limits exist **A0**: Rotation center of crank **B0**: Rotation center of swing

checkfourbar3Poseattachpermutation

returns only subset of permutations of grid points to achieve a real fourbar linkage solution

In: **PS**: Pose; only PS.A and PS.B are used **psel**: optional CPLE or size of square **GNr**: Grid point list relative to origin or resolutin **fillCPL**: CPLE buffer buffer size or exact CPL

Out: **PS**: [i j ellbow segment startangle endangle] **C**: Grid point list relative to origin size of GPL **R**: Radius of circle for each grid point size of GPL **P1**: Grid point list relative to pose 1 size of GPL **P2**: Grid point list relative to pose 2 size of GPL **P3**: Grid point list relative to pose 3 size of GPL **ei**: index of Grid points with solution

checkfourbar3Posepermutation

calculates from a center3pose solution all possible permutations and the fourbar parameters

In: **A**: Pose starting points **B**: Pose end points **S**: Solutions created by center3Pose **dws**: distance and angle; default is first

Out: **FP**: Existing Solution list [L1 L2 R D C U A0 B0 iA iB kA kBd w] **FPall**: Table containing all possible parameters

checkfourbarcouplerlength

checks the solutions of invkinplan2 for a fourbar linkage calculation

In: **B1**: REAL Solutions of joint 1 configuration ellbow down **B2**: REAL Solutions of joint 1 configuration ellbow up **PL**: REAL Point list for TCP **l2chk**: expected length of link 2 **tol**: tolerance default is 1e-7 , i.e. eps2

Out: **ok**: true of there is no problem **ind1**: index of B1 with problem **ind2**: index of B2 with problem **err**: maximum accoured error **r1**: average link length l2 for configuration ellbow down **r2**: average link length l2 for configuration ellbow up

checkfourbarNPose

checks whether a fourbar joint solution exists for results of center3Pose

In: **A**: Starting Points of 3 Poses **B**: Ending Points of 3 Poses **S**: Solutions comming from center3Pose **dws**: selected pair of distance and angle **k**: selected values for k; default are first and last of S **pl**: length of pose

Out: **ci**: configuration for pose of a fourbar linkage **pinfo**: Position error of the solution

checkfourbarsegment

calculates the angle intervals for a fourbar linkage and appoint point to the intervall

In: **l1**: l1 of swing arm **l2**: l2 of swing arm **r**: r of crank **A0**: Distance or A0 or [A0;B0] **PL**: Points list to check or Pose with field A for PL **na**: number of points for plotting contours PL1 and PL2

Out: **wres**: false or empty or [seg_min seg_max] or [seg1_min seg1_max seg_2min seg_2max] **s**: list of angle segment per point zero, one or two **WL**: angles of Points **ord**: order of points **PL1**: PLR-CYAN CONTOUR Point list for rotation of complete segment 1 (blue, cyan) **PL2**: PLR=YELL CONTOUR Point list for rotation of complete segment 2 (yellow)

CPLfourbarblockswing

creates the contour that limit the swing movement

In: **CPLG**: Unmoved contour (Ground Links) **CPLC**: Contour of Cranc **TA1**: Motion path of Cranc **CPLS**: Contour of Swing **TB1**: Motion path of Swing **buff**: [width of block, slot, radius for A0/B0]; default is [3 0.5 20] **A0B0**: Point list of A0 and B0 that limits the size of the block with buff(3)

Out: **CPLBlock**: Contour of Block to add to CPLG

CPLfourbarcontour

Contour that fullfills a fourbar condition

In: **L**: [L1 L2 L3 L4]; default is [78,33,112,33]

Out: **CPL**: Points list **IW**: limiting angles **CPLR**: minimal CPLR to plot or move

CPLfourbarlinkage

creates for a specified fourbar linkage the cpls for crank, coupler, swing

In: **P1**: Position of Crank Center **P2**: Position of TCP (end of coupler) **P3**: Position of link 1 (start of coupler) **P4**: Position of Base of link 1 (2R robot base) **P5**: Position of POI (0=P3 1= P2) or [POI delta x] **R12**: Ro and Ri Outer and inner Radius **CPLX**: Contour of the Shape linked to POI **dw**: optional turing angle

Out: **CPL1**: CPL of swing, 2R-robot link 1 **CPL2**: CPL of coupler, 2R-robot link 2 **CPLR**: CPL of crank, radial position specification **CPL**: complete contour for sweeping **R12o**: [Ro and Ri] Outer and inner Radius **CPLX**: CPL of Final Contour **TX**: Final transformation matrix

CPLfourbarlinkage2

creates for a specified fourbar linkage the cpls for crank, coupler, swing

In: **P1**: A0 Position of Crank Center **P2**: A1 Position of TCP (end of coupler) **P3**: B1 Position of link 1 (start of coupler) **P4**: B0 Position of Base of link 1 (2R robot base) **P5**: C0 Position of POI in mm or [POI delta x] **R12**: Ro and Ri Outer and inner Radius **CPLE**: Contour of the Shape linked to POI **dw**: turning angle

Out: **CPL1**: CPL of swing, 2R-robot link 1 **CPL2**: CPL of coupler, 2R-robot link 2 **CPLR**: CPL of crank, radial position specification **CPL**: complete contour for sweeping **R12**: [Ro and Ri] Outer and inner Radius **CPLE**: CPL of Final Contour **TX**: Final transformation matrix

CPLfourbarlinkagehull

returns the hull shapes of crank, and swing

In: **L1**: Length of Swing **L2**: Length of Coupler **R**: Radius of Crank **D**: Distance between Swing and Crank **roi**: outer and inner radius if link geometry **eb**: 1=ellbow down; 2=ellbow up **wres**: optional angle area to check

Out: **ARCR**: Hull of crank movement **ARC1**: Hull of swing movement **ARCB**: Contour to connect base points of crank and swing **CPLRT**: crank shape contour transformed into its coordinate system **CPL1T**: swing shape contour transformed into its coordinate system **CPL2T**: coupler shape contour transformed into its coordinate system

CPLfourbarmoved

returns a different pose for an fourbar defined by 4 pose points

In: **A0**: A0 of a fourbar pose **B0**: B0 of a fourbar pose **A1**: A1 of a fourbar pose **B1**: B1 of a fourbar pose **wA0**: desired crank angle at A0

Out: **NA0**: A0 of a fourbar pose when crank is at wA0 **NB0**: A0 of a fourbar pose when crank is at wA0 **NA1**: A0 of a fourbar pose when crank is at wA0 **NB1**: A0 of a fourbar pose when crank is at wA0 **err**: deviation of angle at NA1

CPLofPosefourbarshape

returns the basic shapes of a pose fourbar linkage solution

In: **PS**: Pose struct **aps**: selected index of PS.GAL **sv**: automated size for selection or [ro ri]; default is 1/8 of min([L1;L2;R])

Out: **CPLR**: Crank contour **CPL1**: Rocker Contour **CPL2**: Minimal Coupler Contour with 3 holes **CPLE**: Effector Contour = PS.CPLE & CPL2 **PS**: Modified struct

CPLRmeltbyCJMAT

melts link contours if they have 2 or more common joints

In: **CPLR**: cell array of link contours **PLA**: optional list for joint center points **"debug"**: if used, the individual step for melting are shown

Out: **CPLR**: reduced cell array of link contours if contours overlap at two or more joint centers

CPLRmeltbyJCIRC

melts link contours if 3 separated contours a are linked by 3 binary joints

In: **CPLR**: cell array of link contours **PLA**: optional list for joint center points **"debug"**: if used, the individual step for melting are shown

Out: **CPLR**: reduced cell array of link contours if binary joints form a closed chain with 3 elements

CPRofPosefourbar

returns center point list and radius list for a fourbar linkage for a given GPL and GAL

In: **PS**: Pose including GPL and GAL **ord**: optional required order

Out: **C**: center point list corresponds to GPL **R**: radius list corresponds to GPL **P1**: GPL Point list transformed into Pose 1 **P2**: GPL Point list transformed into Pose 2 **P3**: GPL Point list transformed into Pose 3 **api**: list of valid points for solutions of GAL => list of invalid points for solutions of GAL **oi**: list of valid solutions of GAL to fullfil order

CPSinserthinge

1st solution to integrate revolute joints into CPL/CPS to be extruded later to SG

In: **CPL**: Contour to be separated and supported by a joint **T**: Frame to cut along the x-axis or interactive 2 point selection **R**: same as in SGcylhinge **wlim**: limitation for movement (angle for CPLNA and CPLNB); default is pi/2 **"debug"**: if used, the relative movement during subtraction is shown **"nocut"**: if used the frame is only used to cut at the joint, nowhere else

Out: **T**: Frame to insert a piano hinge (SGcylhinge) **CPLNA**: Contour of moving part **CPLNB**: Contour of static part **CPLU**: Contours that are not part of the movement **CPLCA**: Contour for the hinge connector to CPLA **CPLCB**: Contour for the Hinge connector to CPLB **wlimA**: Frame that shows into the direction of the center of CPLCA

CPSinserthingelist

inserts a list of hinge geoemtries into a CPL/CPS. The first frame cuts, the following not

In: **CPL**: Contour in which the hinges are to be inserted **T**: cell list of Frames. Empty Frames mean interactive selection, Numeric scalar means n interactive frames **R**: same as in SGcylhinge but only R(1) used **wlim**: angle limit [amax] or [amin amax] with respect to the x axis of the frame **"both"**: if used instead of single substraction, both parts are equally reduced. **"debug"**: if used, the individal steps are shown **"nocut"**: if used, the contour finally is closed again (not implemented yet) **"close"**: if used, the contour is finally closed again (not implemented yet)

Out: **T**: cell list of frames **CPLNA**: cell list of moving contours wrt T{i} **CPLNB**: cell list of static contours wrt T{i} **CPLU**: cell list of independent contours wrt T{i} **CPLCA**: connector contour for CPLNA to a piano hinge wrt T{i} **CPLCB**: connector contour for CPLNB to a piano hinge wrt T{i} **wlimA**: angle limitations of CPLCA, there is also wlimB, and CPLX

DoFGruebler

Gruebler-Chebychev-Equation to calculate the DoF

In: **NC**: Number of links or CPLR cell array of intersecting link contours **NJ**: Number of joints

Out: **DoF**: degrees of freedom from the Gruebler-Chebychev equation

DoFGrueblerJacobiRank

DoF by the jacobi Rank (only double joint rods are considered)

In: **ELT**: Edge list truss **PLA**: Points of joints

Out: **DoF**: DoF considering only the bars that have to ends at points

ELlongestpath

returns a list of unique directed kinematic chains from an edge list

In: **EEL**: Edge list, only the fist two cols are used

Out: **PP**: cell list of point indices that create longest list **G**: graph object built from the edge list **PPL**: cell array of all point-index paths found **CC**: allcycles(G)

ELofCPLRconnections

returns the contour edge list for the contour defining point connections and the join edge list describing wich contours are connected by the join

In: **CPLR**: cell array of contours **PLA**: point list of join center point **tol**: tolerance for similar rotation centers; default is 0.95

Out: **CMAT**: full correlation nxm matrix of n contour in CPLR and m joint coordinates in PLA **ELC**: n cell list of which joints belong to a contour **ELJ**: m cell list of which contours are connected by a joint (must be nm x 2 for standard linkages) **CJMAT**: full n x n common joint matrix for nodes (2 or more means melted) **EEL**: m x 2 extended edge list [start-contour end-contour joint-index] - contour numbers > numel(CPLR) identify singele or multi-contour-joint (>2contourz) **loopC**: loops of contour-node-graph (egdes are joints)

findCollinearEdges

returns the groups with collinear edges

In: **ELT**: EL of a truss **PLA**: Point list of a truss **tol**: default is 1e-6

Out: **grps**: cell array of edges that form a collinear structure **ni**: number of edges in each collinear group

fourBarCLLL2SGdesign

Converts a contour layered link list into a real solid geometry construction to 3D print

In: **CLLL**: contour layered link list **R12**: [Ro Ri H]; default is 5 2.5 6] **RACK**: Additional Rack Points for Fixation **"layer"**: not recommended but possible to change the layers [CPLR CPLE CPLC CPLS CPLW] **"debug"**: if used, the design is shown step by step **"assembly"**: 'DIN912985' or 'DIN912BUSH' or 'DIN7991' as assembly method **"mirrR"**: 'mirrR', <distance> creates a second mirrored linkage in distance of the rack **"massE"**: 'massE' boolean, creates a massive solid effector, similar is massS

Out: **SG**: Cell list of Solid Gemetries **SLLL**: Same as CLLL but with SGs instead of CPLs % SLLL={SGE;SGC;SGS;SGG;SGW;SGA0;SGB0;SGA1;SGB1}

fourbarcollABGPL

returns the crank angle interval for an object liked to a Pose

In: **CPLW**: Static Obstacle **A0**: A0 or Pose PS **B0**: B0 or solution number **A1**: A1 or empty if PS is used **B1**: B1 or empty if PS is used **GPLA**: GPLA or empty if PS is used **GPLB**: GPLB or empty if PS is used **wlim**: Angle limitation

Out: **wint**: allowed angle interval without collision

fourBarLinkage

multi purpose fnct for a 4-Bar-Linkage

In: **a**: ground distance; alternative Point 1 **b**: crank length; alternative Point 2 **c**: coupler length; alternative Point 3 **d**: Follower length; alternative Point 4 **wb**: rotation angle at a/b **k1**: C+k1*(D1-C) **k2**: orthogonal to k1 in mm **CPLX**: optional closed polygon list defining a clipping/bounding region for plotting; default is a rectangle derived from the link lengths

Out: **C**: Point C **s**: solution exist or not **D1**: Point D1 **D2**: Point D2 **wb**: rotation angle at a/b **wa1**: rotation angle at a/d (solution 1) **wa2**: rotation angle at a/d (solution 2)

fourBarLinkageanimate

animates a fourbarlinkage based on 4 Points, a Frame, and a Contour

In: **A0**: Base Point A0 on ground link; or CPL [A0;B0;B1;A1] **B0**: Base Point B0 on ground link **A1**: Attachment Point A1 crank-coupler **B1**: Attachment Point B1 follower-coupler **TP**: Frame of Pose **CPLE**: Contour relative to Pose

Out: **h**: handle to graphics objects

fourBarLinkageCPL

multi purpose fnct for a 4-Bar-Linkage

In: **a**: ground distance; alternative Point 1 **b**: crank length; alternative Point 2 **c**: coupler length; alternative Point 3 **d**: Follower length; alternative Point 4 **CPL**: Closed Polygon List **k1**: C+k1*(D1-C) **k2**: orthogonal to k1 in mm

Out: **C**: Point C **s**: solution exist or not **D1**: Point D1 **D2**: Point D2 **wb**: rotation angle at a/b **wa1**: rotation angle at a/d (solution 1) **wa2**: rotation angle at a/d (solution 2)

fourBarLinkageplot

plots and animates the movement of a fourbar linkage

In: **l1**: length of link1 at Base B0 **l2**: length of link2 / coupler **r**: radius of crank at Base A0 or Point List around A0 **A0**: distance, A0-B0, Position of A0 or List [A0;B0] **np**: number of points on the crank circle

Out: **h**: handle to last graphics objects

fourBarLinkageplotanim

plots and animates the movement of a fourbar linkage

In: **l1**: length of link1 at Base B0 **l2**: length of link2 / coupler **r**: radius of crank at Base A0 **A0**: distance, A0-B0, Position of A0 or List [A0;B0] **tmove**: time to animate; default is 2 sec **elup**: elbow up (true) elbow down (false); default is false; **transl**: transparancey for swept volume (0..1); default is 0; **ka**: either ka or[ka w] or [ka w disp]

Out: **h**: handle to last graphics objects

fourBarLinkageplotanimui

plots and animates the movement interactively of a fourbar linkage

In: **l1**: length of link1 at Base B0 **l2**: length of link2 / coupler **r**: radius of crank at Base A0 **A0**: distance, A0-B0, Position of A0 or List [A0;B0] **tmove**: time to animate; default is 2 sec **elup**: elbow up (true) elbow down (false); default is false; **transl**: transparancey for swept volume (0..1); default is 0; **ka**: either ka or[ka w] or [ka w disp]

Out: **h**: handle to last graphics objects

fourbarplotABGPL

simply plots a fourbar linkage based on points - recommended in SG-Lib 5.1

In: **A0**: Base point of crank **B0**: Base point of swing **A1**: Attachment point of crank at coupler **B1**: Attachment point of swing at coupler **GPLA**: Position of attachment point A1 relativ to a pose **GPLB**: Position of attachment point B1 relativ to a pose **R12o**: R12ofCPL [Out and inner radius of link and bore] **wlim**: if used, the movement path of crank, swing and pose are drawn

Out: **h**: handle to the plot;

fourBarplotPoints

just plot a specific fourbar configuration based on points

In: **A0**: A0 or CPL with ORDER [A0; B0; B1; A1] **B0**: Base of swing **A1**: End of Crank **B1**: End of Swing **TP**: Position or Frame of Pose **se**: if true, the effector contour is shown; default is true

Out: **h**: handle to go

fourBarposeCPLmotion

returns the motion of all parts of a fourbar linkage

In: **PS**: Pose with field "solut" **aps**: solution number **npnt**: number of points along the path **att**: contour attached to "CPLP0", "CPLA0", "CPLB0", "CPLA1", "CPLB1"; just for graphical output **CPL**: contour to attach; default is an arrow **wsol**: limited rotation angle of crank **"R12"**: 'R12' option to set the link contour radii [Rout Rin]

Out: **PL0**: Point list of pose movement **TL0**: List of 3x3 HT of pose movement **wcr**: corresponding angle list of cranc during motion **TA1**: List of 3x3 HT of A0 movement **TB1**: List of 3x3 HT of B0 movement **TA0**: List of 3x3 HT of A1 movement **TB0**: List of 3x3 HT of B1 movement

fourbarposeCPLshaping

returns collision free path shaped contour for a specific fourbar linkage

In: **PS**: Pose and fourbar linkage struct including GPL and GAL **aps**: [solution, limiting order] **CPL**: Contour to shape; if empty an simple contour is created **att**: Given CPL is moved relatively/is attached to 'E'ffector, 'C'rank, or 'Swing' or 'WORLD' **XCPLW**: Static obstacles in the environment and will cut the moving CPL **XCPLE**: Obstacle contours attached to Pose and will cut the moving CPL **XCPLC**: Obstacle contour moved with crank and will cut the moving CPL **XCPLS**: Obstacle contour moved with swing and will cut the moving CPL

Out: **CPL**: Collision free contour

fourBarposelayering

Creates the layering and the shaping of all 8 elements of a Fourbar linkage

In: **PSX**: Pose struct include PS.solut field **aps**: Single pose or selection of poses **R12**: [Rout Rin Height]; default is [5 2.5 6] **lay**: default is [0 2 1 1] **Shape**: "A*" (default) or "Bezier" **"debug"**: covers the crank behind the effector **"hide"**: [angle-offset angle-range]; default is [0 360] **"wlim"**: 'wlim' flag to apply the angle limits

Out: **LLLM**: Contour Link Level Layering ==> fourBarCLLL2SGdesign **TL0**: Frame list of the coupler **TA1**: Frame of the crank endpoint A1 **TB1**: Frame of the swing endpoint B1

fourBarposeplotsolution

Standard procedure to animate fourBarposesyntheses results

In: **PS**: Pose struct include PS.solut field **aps**: Single pose or selection of poses **vis**: 'plot', 'animate', 'both' **num**: number of solutions and delay time; default is [1000 -1]; negative time requires interaction **Roi**: [outer inner] radius of the coupler point marker; scalar sets outer, inner=0.5

Out: **h**: handle to graphics **oldtitle**: original axis title

fourbarposesearchcouplercurve

sorts 4bar solutions wrt minimal distance of coupler path to a given set of points

In: **PS**: Pose struct **CPLF**: Point List or contour considered as SET of points! **acc**: [min max]; default is [0 inf] **rots**: true==rotating, false=non rotating; default is '' **succ**: unused flag; default is empty

Out: **PSX**: Pose struct sorted by minimal square distance **di**: sum of minimal square distance

fourBarposesortsolution

sorts the solutions of the fourbar struct by some criteria and even selects

In: **PS**: Pose with field "solution" **cmd**: command list such as 'path-stretch', [limits], **lim**: limitationa

Out: **PS**: Pose with sorted or selected poses **kk**: last used and sorted criteria

fourBarposesyntheses

fourbar pose synthesis

In: **PS**: Pose struct including fields A, B, CPLE, CPLM, CPLW **posel**: selected poses, number selects algorithm too **grids**: grid point numbers for CPLE and CPLM **order**: optional order of poses for ccw crank movement **limit**: optional limitation at 1st or last pose **cnstr**: optinal constraints for checkfourbar123solutions **fille**: if true, the grid points are filled inside the contour; default is false **"attach"**: if given as 'attach', the grid points are transformed relative to the effector poses

Out: **PS**: Pose including field "solut" with solutions

invkinplanarlinkage

inverse kinematics of a linkage in absolute coordinates by two base point coordinates and one absolute angle (-pi/2 .. pi2/2)

In: **PL**: List of Joint coordinates **EL**: Linkage connection between the joints **fixInd**: indices of two joints that form the base for the linkage **fixPos**: Selected coordinates of the two joints; default are PL coordinates **knownEdge**: One known edge of EL to be moved **knownAngle**: Absolute angle of this edge in rad relative to the x-axis [-pi/2 .. pi2/2]

Out: **S**: Solution struct, containing PL EL L(ength, all) w(angles, all), resnorm=abs(fval)

invkinplanarlinkagerot

inverse kinematics of a linkage in absolute coordinates by two base point coordinates and one absolute 360 deg angle

In: **PL**: List of Joint coordinates **EL**: Linkage connection between the joints **fixInd**: indices of two joints that form the base for the linkage **fixPos**: Selected coordinates of the two joints; default are PL coordinates **knownEdge**: One known edge of EL to be moved **knownAngle**: Absolute angle of this edge in rad relative to the x-axis (360 degree)

Out: **S**: Solution struct, containing PL EL L(ength, all) w(angles, all), resnorm=abs(fval)

invkinplanarlinkagerotK

inverse kinematics of a non rigid linkage in absolute coordinates by two base point coordinates and one angle

In: **PL**: List of Joint coordinates **EL**: Linkage connection between the joints **fixInd**: indices of two joints that form the base for the linkage **fixPos**: Selected coordinates of the two joints; default are PL coordinates **knownEdge**: One known edge of EL to be moved **knownAngle**: Absolute angle of this edge in rad relative to the x-axis (360 degree) **K**: stiffness koefficient vector of the edges (default is all inf) **tolRigid**: Tolerance for the rigid-link constraint

Out: **S**: Solution struct, containing PL EL L(ength, all) w(angles, all), resnorm=abs(fval)

jacobiantruss

jacobian of a truss

In: **ELT**: rows of edges between a start joint and an end joint m x [start end .....] **PLA**: 2D point list with all joints used in ELT - No unused points are allowed in the list **tol**: numerical tolerance; default is 1e-6

Out: **J**: Jacobian symmetric matrix **ELN**: modified ELN if some points of PLA were not used in the truss **PLN**: modified PLN if some points of PLA were not used in the truss

PLcircleoval4obstacle

similar to PLcircleoval but is considering static and moving obstacles

In: **R**: Radius **n**: optional number of points for half circle **L**: [A0; B0] **CPLO**: optional obstacle contour **TL**: frame or frame list that describes the movement von CPLO **"strict"**: if used, also A0 and B0 must be collision free

Out: **CPLN**: Point List of the contour from A0 to B0

PLELsample

returns PL and EL to create linkage networks using CPLRofPLEL

In: **Nr**: sample number **"arc"**: if used, the following value forces link contours with this arc radius; default is 0

Out: **PLN**: Point list of joint coordinates **ELN**: Edge list to define links between the points PLN **CPLR**: cell array of link contours

PLofCPLRcrossings

returns a point list of contour crossing that could be used as joint coordinates

In: **CPLR**: cell array of contours **PLA**: optional point list with more accurate coordinates to map to **tol**: tolerance radius to identify circle centers as identical; default is 0.95mm

Out: **MMR**: full matrix [a b x y] for crossing points of contour a with contour b **PLU**: Unique Center Point list **PLT**: Unique Terminal Point list that are not part of joints

PLofFourbarcouplercurve

Creates the coupler courve for an Fourbar given by four points

In: **A0**: Postion of base point crank **B0**: Postion of base point swing **A1**: Endpoint of crank **B1**: Endpoint of swing **GPLA**: optional attachment point of crank at coupler **GPLB**: optional attachment point of swing at coupler **wsol**: optional angle limitations **CPLE**: PLots CPLE in interactive mode

Out: **PL0**: Point list of coupler courve **TL0**: Frames if coupler courve **wlim**: angle limitation =s **PLC**: Point list of radial crank movement **PLS**: Point list of radial swing movement **wA1**: Angle of the crank at endpoint A1 **wcr**: Angle values of the crank movement (blue)

PLofLinkagecouplercurve

returns for a specific planar linkage, base points and drive, the limitation angles and movements of all joints

In: **CPLR**: cell array of linkage contours | OR | ELT if no contour analysis requested **PLA**: Point list of the joiint coordinates **FixI**: index of the two base points; default is [1 2] **SelI**: selected edge/link that should move **wlim**: angle limit the be checked for movements; default is [0 2*pi] **ELTadd**: additional edges to reduce the DoF of a truss **"planar"**: if used the movement does not allow any direct edge crossings **"debug"**: if used steps of calculating are shown

Out: **PalI**: Point list of n joints x 2 x k angles [n x 2 x k] **WalI**: Angel list if m edges x k angles **ELT**: ELT (EELofCPLR) is the edge list that triangulates also rigid multi-joint contours **ELC**: ELC (EELofCPLR) is the list of joints per contours required for pose transformation | OR EMPTY IF CPLR=ELT **wlimfe**: the maximum angle interval that really can achieved by the movement constraints starting n this pose **posi**: starting pose index for the starting angle of the crank

PLofPall

Auxiliary fnct to get points of a specific joint for single, several or all poses

In: **PalI**: Pall of PLofLinkagecouplercurve **ai**: indices of points of interest **ki**: pose indices; default is 1:size(Pall,3)

Out: **PLA**: Requested Point list

PLofPosecouplercurve

returns the path and the matrix list of the pose movement

In: **PS**: Pose incl. GPL and GAL **aps**: index in GAL or [iA iB] for GAL **npnt**: number of points of the curve **ncur**: number of curves desired; default=0 (Pose Coupler Curve);(will be removed)

Out: **PL0**: Point list of Coupler Points Movement **T0**: Transformation matrix list of Coupler Points Movement **wcr**: angle list of crank returned by invkinplan2 **TA0**: Transformation matrix list of joint A0 (only rotation) **TB0**: Transformation matrix list of joint B0 (only rotation) **TA1**: Transformation matrix list of Attachment point A1 (circular Movement) **TB1**: Transformation matrix list of Attachment point B1 (circular Movement)

poletriangle3Pose

calculates the edges of the pole triangle of three poses

In: **A**: Point List for A; or Pose struct **B**: Point List for B if A is not pose struct

Out: **PL**: Point List [P12;P23;P31] **A123**: Zero Point **A0**: Center of A0 **R**: Radius (Distance A1-A0, A2-A0, A3-A0)

Posefourbaranglelimit

limits the angle range of a specific fourbar solution

In: **PS**: Pose struct including PS.GAL **aps**: number of attachment point solution **ival**: [min max] angle of the crank

Out: **PS**: Pose struct with modified PS.GAL angle interval

Posefourbarangles

Returns the angular position of the crank for the positions used in the four-bar linkage.

In: **PS**: Pose using A0,A1,B0,B1 or PS.GAL & PS.GPL if aps ~= []; **aps**: specific solution of PS.GAL to replace A0,A1,B0,B1

Out: **wcr**: list of crank angles for the pose list PS.A **A1i**: list of endpoints Ai for the Poses **err**: error in distance

Posefourbaranimate

animates the fourbar linkage given Pose parameter

In: **PS**: Pose (A0,A1,B0,B1,A) **ap**: optional selected solution in GAL if desired **lim**: if set, limit crank motion via Posefourbarposelimit; values >100 taken as a pose order

Out: **hx**: handle to graphics

Posefourbarcheck

shows the shape of effector, crank and swing

In: **PS**: Pose **aps**: solution to check **lim**: if true; limit the movement to the pose order **CPLB**: Optional Background Shape **CPLE**: Optional Effector Shape **CPLR**: Optional Crank Shape **CPL1**: Optional Swing Shape

PosefourbarCPLE

creates a CPL for an Effector containt 3 Points

In: **PS**: Pose struct or A1 of GPL **csi**: [crank-index swing-index] of GLP for A1 and B1 or B1 of GPL **R12o**: [Outradius Innerradius] **orig**: '0','A','B' to define the origin; default is '0'

Out: **CPLE**: Minimal Effector contour containing Coupler Pose center **PL**: [P0;A1;B1] **T**: T to transform the coupler into the origin; default is eye(3)

Posefourbarplotcouplercurve

simply plots a coupler curve for fourbar linkage

In: **PS**: Pose including A0, B0, A1, B1, or GPL and GAL **aps**: select solution or combination of attachment points in GAL, requires GPL and GAL **col**: col of coupler courve; default is 'r-' **lw**: line width of coupler courve; default is 2 **cot**: col of coupler orientation; default is 'r-' **lim**: if true, limit pose motion via Posefourbarposelimit; default is false

Out: **h**: handle to line

Posefourbarposelimit

finds the angle limits for a fourbar movement

In: **PS**: Pose struct including GPL and GAL **aps**: optional index for test **ford**: optional order; default is default order or 123 **delay**: pause between plots when nargout==0; true pauses, >0 pauses that many seconds; default is 0

Out: **PS**: Pose struct with modified PS.GAL List and limited angles! **wlim**: limitation for angle [0 0] if no solution possible for input ford **ford**: order of poses; 0 if no solution possible for input ford

Posefourbarsearchcouplercurve

selects coupler courves based on the coupler curve

In: **PS**: Pose struct including CPLE, GPL, GAL **CPLF**: Exact contour to fit best. **rots**: true = full rotation, false = no rotation, default is both **CPLR**: Area for Crank Base Point; default is not limits **CPL1**: Area for Swing Base Point; default is CPLR **type**: comparison metric type; 'shape' compares by contour area difference; default is 'shape'

Out: **PSX**: Pose with only the slelected solutions left **di**: index list of the solutions in GAL that were selected

PosefourbarselectAP

simply select a solution from the Pose attachment point list

In: **PS**: Posestruct **aps**: solution number or combination of attachment points

Out: **PS**: New PS.GPL = Grid Point list, PS.GAP = Grid Point Attachment List **i**: Solution selected if aps was used with 2 input parameters **A0**: Base Point Crank **B0**: Base Point Swing **A1**: Attachment Point Crank **B1**: Attachment Point Swing

Posefourbarselectcouplercurve

selects coupler courves based on the coupler curve

In: **PS**: Pose struct including CPLE, GPL, GAL **dim**: number of curves, or limits of the curve, or enclosing contour **rots**: true = full rotation, false = no rotation, default is both **CPLR**: Area for Crank Base Point; default is not limits **CPL1**: Area for Swing Base Point; default is CPLR

Out: **PS**: Pose with only the slelected solutions left **gapi**: index list of the solutions in GAL that were selected

Posefourbarselectui

interactive tool to plot poses and to select two attachment/base points for a fourbar linkage

In: **PS**: Posestruct **ord**: Optional required order such as 123

Out: **PS**: New PS.GPL = Grid Point list, PS.GAP = Grid Point Attachment List **comb**: index into GAL of the selected attachment/base point combination

Posefourbartorquetrans

returns the torque transmission factor for crank to swing and crank to pose

In: **PS**: Pose **aps**: selected solution **nw**: number of auxiliary point **Ti**: optional point relative to pose; not implemented yet

Out: **tcp**: transmission crank pose **tcs**: transmission crank swing **wcr**: crank angles

Poseoffourbarlinkage

creates a Pose struct with 3 or more poses for given fourbar linkgae parameters L1, L2, R, D/[A0;B0]

In: **L1**: Length of Swing **L2**: Length of Coupler **R**: Radius of Crank **D**: Distance or [A0; B0] **ebow**: 1== ellbow down; 2== ellbow up **segm**: segment 1 or segment 2 **np**: number of poses; default is 3 **TAP**: Pose relative to A1; 'A1','B1', dx, [dx dy]; Pose of P rel. to B1; not implemented yet

Out: **PS**: Pose struct PS.A, OS.B, PS.A0, PS.B0

poseofPSanalyze

returns a Pose and a CPL from an analyzeed 3 Pose Polygon

In: **XX**: Result of PSanalyze **i**: optional selection if there several polygon as set of at least 3 copies

Out: **PS**: PS.A start points PS.B end Point PS.CPL moving CPL. **CPL**: Kontour of a CPL that exist at least 3 times **CPLB**: combined contour of the remaining base polygons

realfourbarsolutions

creates an index list to map fourbarlinkage solution points to movement types

In: **rs**: real solution flag of invkinplan2 **mtyp**: 'forw', 'backw', 'alt'

Out: **nni**: mapping index for PL, B1, B2, W, U of invkinplan2 **seg**: 0 = closed movement, 1 = one segment, 2 - two segments

selectLPSui

selects interactively Frames or Frame connections from a lingage Polygon Struct

In: **LPS**: Linkage Polygon Structure or handle to graphics containing LPS **typ**: "Frame", "Link"

Out: **LL**: {LPS element row index | name of Element | Hole index | Distance to point | Name of Hole | Frame}

SGofCPLzpianohingelist

extrudes a solid from a single contour and inserts a set of piano hinges after cutting the contour into pieces

In: **CPL**: Contour to be extruded; default is a square **H**: Height for CPLz; default is 40 **T**: Cell list of frames or numerical scalar for number of joints; default is 1; **RR**: default is [2 1.5 pi/4] **wlim**: default is: [-pi/4 pi/4] **"STL"**: if used an STL-File is written if nargout==0; **"debug"**: if used the creation of the linkages are shown **"both"**: if used, the creation of the linkages is shown

Out: **SGall**: cell list of individial solids created by SGanalyzeGroupParts **T**: list of frames (interactive use) **ELN**: Edge-Frame-List [Base-index Follower-index Frame-index] **SGF**: assembled solid chain of the hinged parts (via SGTchain) **CPLO**: the original input contour **WLIM**: matrix of angle limits [amin amax] per hinge

SGpianohinge

creates a piano-hinge style revolute joint and attaches predefined contours as link arms

In: **R**: [Radius Pin-height pin-angle]; default is [2 1.5 pi/4]; 1.2 ist the lower limit for R(1) **L**: [Hinge-length segment-number]; default is [40 4] **CPLG**: Connection Contour for the rotating part at [0 0]; default is a rectangle in y **CPLY**: Connection Contour for the static part at [0 0]; default is a rectangle in y **flip**: if false, the 1st hinge element is fixed to the rotor, otherwise the stator; default is false **"STL"**: if used, the hinge is written as STL **"debug"**: if used, the creation of the hinge is shown **"flip"**: if used, the parameter flip is preset, but can be overwritten

Out: **SGall**: {SGtop,SGmiddle,SGbottom} **SGUO**: {SGbottom,SGtop}

textEELPL

fnct to plot the exteded contour-node joint-edge list with coordinates or joint indices

In: **MMR**: Extended Joint edge list [startcontour endcontour join-index] or [startcontour endcontour joint-x joint-y] **JPLU**: Unique Joint coordinate point list **ind**: joint selector index; default is 1:size(JPLU,1)

Out: **ttt**: cell array of desciption

TLofPalledge

Auxiliary fnct to get points of a specific joint for single, several or all poses

In: **Pall**: Pall of PLofLinkagecouplercurve **ei**: edge with two point indices [ai bi] as link pose of interest **ki**: pose indices; default is 1:size(Pall,3)

Out: **TLE**: Requested Pose list

6.6.4 Four-bar and Hinge Solids [guide]

Functions in this section:

SGfourbarhullplate — returns the hull shapes of crank, and swing

SGfourbarlinkage — creates solid geometry parts for a given linkage

SGfourbarlinkage2021 — EXPERIMENT TO CREATE 4BAR LINKAGES FOR FRANZ IRLINGER

SGfourbarplate — returns a base plate for a fourbar linkage experiment

SGof2CPLTzpianohinge — connects two contours CPLA and CPLB by a pianohinge with given rotation center, frame, radius and height

SGofCPLRTzpianohinge — creates 2.5 extruded solids of linkage contours using piano hinges

SGofCPLRTzpianohingeclosed — creates 2.5 extruded solids of linkage contours using piano hinges

SGofCPLRwlim — Quick creation of a 3D-printable linkage from link contours, joint coordinates, and motion interval

SGofCPLRz — creates a solid geometry of a linkage

SGofPosefourbar — returns a fourbar linkage geometry for a given attachment point solution

Detailed description

SGfourbarhullplate

returns the hull shapes of crank, and swing

In: **ARCR**: Hull of crank movement **ARC1**: Hull of swing movement **ARCB**: Contour to connect base points of crank and swing **roi**: outer and inner radius if link geometry **h**: heiht; default is [roi(1)/2]

Out: **SG**: Solid Geometry

SGfourbarlinkage

creates solid geometry parts for a given linkage

In: **L1**: Length of Swing **L2**: Length of coupler **R**: Length of Crank **D**: Distance between Base points of Swing and Crank **roi**: [ro [ri]], outer radius, inner radius **h**: height; default is ro **eb**: ellbow configuration 1/2 **CPLE**: Optional end-effector contour (CPLE) for interactive/animation mode

Out: **SGM**: Solid Geometry of all Parts mounted **SGAll**: Solid Geometry of all Parts packed

SGfourbarlinkage2021

EXPERIMENT TO CREATE 4BAR LINKAGES FOR FRANZ IRLINGER

In: **LA**: Length A **LB**: Length B **LC**: Length C **LD**: Length D **Ri**: solution angle' default is 1st **fi**: 'c' or 't' for fitting; default is 't' **LV**: Level; default is [0 1 2 1]

Out: **SGall**: 4 Links and 6 Pins to create the linkage

SGfourbarplate

returns a base plate for a fourbar linkage experiment

In: **D**: Distance between A0 and B0 or [L1;L2:R;D;[eb]] **roi**: outer and inner diameter of the links **h**: height of the links **s**: step size **sdate**: true = attach date **siz**: size of board

Out: **SG**: Solid Geometry of board, pins, and date string

SGof2CPLTzpianohinge

connects two contours CPLA and CPLB by a pianohinge with given rotation center, frame, radius and height

In: **CPLA**: Contour A (blue) **CPLB**: Contour B (blue) **H**: Height and number of links as used for SGpianohinge; default is [10 4] **TA**: Rotation center frame fixed to CPLA **RR**: Joint Radius - used for SGpianohinge; default is 2 **wlim**: rotation angle of CPLB around TA (relativ to CPLA) **"debug"**: if used, the design steps are shown

Out: **SGD**: cell arayy of elements **ai**: index in SGD that belong to CPLA **bi**: index in SGD that belong to CPLB **ci**: index in SGD that belong neither ot CPLA nor CPLB **CPLA**: modified contour A including the bridge to the joint center **CPLB**: modified contour B including the bridge to the joint center

SGofCPLRTzpianohinge

creates 2.5 extruded solids of linkage contours using piano hinges

In: **CPLR**: cell array with link contours comming from CPSinserthingelist **H**: [Hinge-length segment-number] (different to SGofCPLz) as in required in SGpianohinge **T**: cell array of Frames that were used returned from CPSinserthingelist **R**: R: [Radius Pin-height pin-angle] as required for SGpianohinge **EEL**: Extended Edge list of the kinematic chain [A-index B-index Frame-index] Frame-index sign only required for wlim **CPLCA**: Cell array of hinge connecting contour for contour-A **CPLCB**: Cell array of hinge connecting contour for contour-AB **wlim**: if used, the design steps are shown

Out: **SGR**: Cell array of solids (no Frames) that are connected by thee EEL chain of T-Frames **EEL**: Extend Edge List **WLIM**: angle limits for EEL **KC**: Kinematic chain

SGofCPLRTzpianohingeclosed

creates 2.5 extruded solids of linkage contours using piano hinges

In: **CPLR**: cell array with link contours comming from CPSinserthingelist **H**: [Hinge-length segment-number] (different to SGofCPLz) as in required in SGpianohinge **T**: cell array of Frames that were used returned from CPSinserthingelist **R**: R: [Radius Pin-height pin-angle] as required for SGpianohinge **EEL**: Extended Edge list of the kinematic chain [A-index B-index Frame-index] Frame-index sign only required for wlim **CPLCA**: Cell array of hinge connecting contour for contour-A **CPLCB**: Cell array of hinge connecting contour for contour-AB **wlim**: if used, the design steps are shown

Out: **SGR**: Cell array of solids (no Frames) that are connected by thee EEL chain of T-Frames **KEL**: Extend Edge List **WLIM**: angle limits for EEL **KC**: Kinematic chain

SGofCPLRwlim

Quick creation of a 3D-printable linkage from link contours, joint coordinates, and motion interval

In: **CPLR**: n x {CPL} cell array of link contours **PLA**: PLA m x 2 Point coordinates of m Joints in this pose of CPL **BASES**: point indices that define the base; default is [1 2] **CRANK**: point indices that define the crank; default is [2 3] **diffw**: movement angle of the crank relative to start pose;default is [-pi10 + pi/10] **R**: Radius for the hinges; default is [1.5] **H**: Height and Hinge elements; default is [10 4] **"cage"**: cage thickness outside of the joints

Out: **SGall**: Solid Geometry ready to 3D print **PLA**:

SGofCPLRz

creates a solid geometry of a linkage

In: **CPLR**: cell array with a link contours **Pall**: Point list x k all m joint coordinates over k poses **ELC**: n x 1 cell array with the joint indeces that belon to a link **posi**: the pose in which the link contours of CPLR were defined **R**: [Radius and cage] used for the joints; default is [2 0] **H**: Height and joint elements; default is [10 4] **"debug"**: if used the design steps are shown **"stl"**: if used an stl file is written as file (bambulab) and folder (snapmaker)

Out: **SGR**: SG cell array containing the individual links

SGofPosefourbar

returns a fourbar linkage geometry for a given attachment point solution

In: **PS**: Pose struct including PS.GPL and PS.GAL **ap**: solution number of PS **lim**: optional limitation of pose order **roi**: [outer and inner radius and slot size [ro ri si]

Out: **SGM**: Solid Geometries mounted **SGA**: Solid Geometries ready for printing

7 Automatic Design, Engineering & Simulation [guide]

7.1 Parameterized Shape Generation [guide]

Functions in this section:

CPLofSGpinholes — returns the Geometry

lofbendinggirder — returns the length of a bended girder

PLlemo — returns the contour of a insertion hole for a LEMO plug/socket

R12ofM — returns the minimal dimensions of a link for a screw nut connection

SGbar — returns a box without frames

SGbarcelonaspacer — Solid protector for Barcelona lounger by Mies van der Rohe

SGbarstructure — creates rectangle as bar structure

SGbearingslide — erzeugt ein Gleitlager nach Spezifikationen für einen AM Printer

SGblockundertable — Simply a block when tables wobble on uneven surfaces

SGbookwedge — returns a wedge for a book or a tablet or medal frames

SGboxradial — similar to SGbox but with radial edges in 2.5 D

SGbraitenbergvehicle — creates a simple solid similar to the vehicles of von Braitenberg

SGcableclip — returns cableclip that can be fixed in 90 degree by a nail

SGcapontop — returns a solid that covers the geometry from top to a specified length

SGcase — returns the solid geometry for a cubic case with cover

SGcatheter — Creates a solid of a catheter

SGcatheterfeeder — creates rollers to move wires and catheters

SGcathetertorquer — creates different catheter torquers to be activated by SGcathetercapsule

SGchristmasstar — writes a STL file for Printing a christmas star

SGclampCTtable — returns the solid geometry for a CT table clamp

SGclosurecapofFS — Automated Design of Closure Cap

SGcupholderinsert — returns a SG as insert for cup holder in a car

SGdragchain — creates a 3D printable already assembled drag chain

SGdragchainelement — returns a single element of a drag chain

SGdragstrip — creates a dragchain cables as texture strip

SGdrillnotch — sets a simple drilling notch on top of a surface

SGendoscope — Creates an Endoscope in KARL STORZ Design

SGEurtraccap — Solid Geometry Cap for an EUTRAC 3Phase electric rail

SGfurniturespacer — Solid protector to protect the touch of the furniture to a wall

SGgearBevel — Returns the vertex/facet list of a bevel gear with involute teeth

SGgearchain — returns a gear chain als simple sequence or stacks

SGgearCrownwheel — Returns the vertex/facet list of a crown wheel with involute teeth

SGgearInternal — returns the vertex/facet list of a gear with internal involute teeth

SGgearpair — returns a combination of two gears

SGgearRack — Returns the vertex/facet list of a gear with involute teeth

SGgearSpur — Returns the vertex/facet list of a gear with involute teeth

SGgearwheelsocketspanner — This is a socket spanner with clearance fitting for screws

SGhandle — returns a simple handle to be fixed by screws

SGhandrailclamp — An example for a snap-in solution of a handrail-adapter for a specified force

SGhook4railsquared — creates a hook for squared shaped rails

SGiso4026 — SG of Hexagon socket set screw with flat point

SGiso4762 — SG of Hexagon socket set screw with cylindric head

SGlemoFAGEGG — Plug and socket for the LEMO FAG EGG family

SGmagnetcap — returns a closed magnet adapter cap for M22

SGmakitaadapter — returns the Solid Geometry of an Makita 14V-18V Akku adapter

SGmedalsframe — creates a plate to present 60mm medals in combination with a book wedge

SGmeshground — creates a ground mesh that is a flexible texture on a FDM printer

SGmieleTwinDosadapter — returns a Adapter for Miele TwinDos detergent cartridges (print with 80%)

SGmodelLink4 — Creates Solid for a Link im Style "Link4"

SGmotorFeeTech — slow fnct that creates solids with text

SGmotorshaftcoupling — creates a rigid motor shaft coupling

SGmotorspindleclutch — creates a motor spindle clutch between a motor shaft and a steep spindle

SGmotorwindup — creates wind up motor with a 3d printed spring

SGmultisocketclamp — returns a clamp to fix a multisocket under a desktop table plate

SGmusicstandextension — supplies extension elements for an orchestra music stand

SGofCPLcommandui — Simple commandline based 2.5 solid modeler for the use of a caliper

SGofCPLzbellows — create a straight bellows based on a CPL

SGpedestal — creates simple solids for pedestals

SGpinshaft — creates a pin using PLshaftrad

SGplugintube — create a tube that can be elongated piece by piece

SGrackbars — creates solid bars to assemble a rack for mechanisms

SGradialhooks — Creates hooks for a rotating bayonet catch

SGreolink — Designing a cover for a Reolink camera

SGreolink2pipeclamp — Produces a bolt plate for screwing the Reolink camera to a double tube clamp with a joint.

SGrescBox — Creates for the iMedCap Rescue Robotics Project a 1:10 Model of the Patient Box

SGrobotTCPplate — returns a TCP flange to mount Fischertechnik® compatible grooves

SGrodlink — returns a cylinder with inclined top and bottom surfaces

SGsbbase — Creates a base for a simple ball bearing four-bar linkage

SGsblink — returns a link for insertion of a bearing, optional a sleeve, and an inner hexagon at the shaft end

SGsbpin — returns a pin printined at 80% to be inserted by a hammer

SGservoflange — Manually designs a flange connection for a servo motor

SGshaftaxis — creates shaft axis combinations

SGshieldbase — creates a shieldbase

SGshutterbeltcover — Creates covers for shutter belts as an example for design using SG-Lib

SGsnaprivet — returns a solid geometry for a snap rivet

SGspring — returns a cylindric spring

SGspringSpiral — returns a spiral spring geometry

SGspringSpiralbolt — creates the shaft-axis combination for a commercial windup motor spring (80% fillin)

SGsteepspindle — returns the geometry of a steep spindle and a nut

SGstent — returns a cylindric stent consisting of two spirals

SGsubtractnaddTL — Basic fnct to support repeated frame based union and subtraction of solids

SGtorquefuse — returns a SG consisting of three parts to implement a clutch

SGwebcamcover — Simply a cover for a desktop webcam

SGwheelspiral — returns a solid of a involute spoked wheel which limit a torque jump or spike

SGwheelspoked — returns a solid of a spoked wheel

SGwrenchsocketspanner — This is a socket spanner with clearance fitting for screws

sofbendinggirder — returns the expected bending of a girder/cantilever

VLFLexpandPL — extrudes a square along a path given as PL

VLFLsnapfit — returns VL and FL of a snapping hollow axle

Detailed description

CPLofSGpinholes

returns the Geometry

In: **SG**: Solid Geometry **ez**: Projection direction (ez) **ms**: Marker/mesh size **wt**: Wall thickness

Out: **CPLB**: Outer contour **CPLA**: Inner contour

lofbendinggirder

returns the length of a bended girder

In: **h**: height of the cross section of bended beam **b**: width of the cross section of bended beam **s**: bending distance; default h **F**: bending force to achieve distance s; default 10 N **E**: E modulus default 1200 MPa (Polyamid PA12) **Slim**: Limit of bending stress

Out: **l**: length of a bended girder **alpha**: bending angle **Smax**: bending stress **r**: bending radius **c**: bending contour

PLlemo

returns the contour of a insertion hole for a LEMO plug/socket

In: **d**: Outer Diameter and inner diameter; default is [12 10.5] **sl**: slot; default is 0 **ww**: rotation angle to rotate the resulting contour; default is 0 (no rotation)

Out: **CPL**: Contour of hole

R12ofM

returns the minimal dimensions of a link for a screw nut connection

In: **M**: Metric size of screw **w**: wall dimension; default is 1.25

Out: **RMHL**: [Radius M-screw Height Length] **SG**: optional solid geometry

SGbar

returns a box without frames

In: **S**: size [x y z]; or a set of boxes given as [x y z x n rows] **n**: number of edges **"col"**: if used 'col','r', the solid is colored

Out: **SG**: Solid Geometry

SGbarcelonaspacer

Solid protector for Barcelona lounger by Mies van der Rohe

In: **dwdh**: [diameter wall baselength height]; default is [32 3 120 15] **styl**: style of the rack contour: 'min', 'spoke', 'bar', 'circ', 'box', or 'star'; default is 'min'

SGbarstructure

creates rectangle as bar structure

In: **siz**: [x y] **bh**: [db dh] **typ**: 'o' or 'z' or 'x'; default is 'x' **al**: angle of cross; default is 51 degree

Out: **SG**: Solid Geometry

SGbearingslide

erzeugt ein Gleitlager nach Spezifikationen für einen AM Printer

In: **Sz**: Size of the Sliding Bearing **fit**: 'c form2b' transition fit for formlab 2

Out: **SGI**: Inner Part for the Screw or Shaft **SGO**: Outer Part for "Interference Fit"

SGblockundertable

Simply a block when tables wobble on uneven surfaces

In: **LBH**: [length width height wall-thickness height-x height-y; default is [40 25 4.5 2.4 10 2] **"STL"**: if used, and STL file is written

SGbookwedge

returns a wedge for a book or a tablet or medal frames

In: **z**: [height-front distance height-back thickness]; default [20 200 35 2] **b**: height of the wedge **r**: radius; default is 0 **h**: height of a stand; default is 0 **"close"**: if used the wedge is closed on the ,floor side **"book"**: if used the wedge is elongated on the floor side

Out: **SG**: Solid Geometry

SGboxradial

similar to SGbox but with radial edges in 2.5 D

In: **siz**: [x y z] size **rrr**: radius for radial edges;de fault is 2 **ddd**: optional extrusion direction; default is 'z'

Out: **SG**: Solid geometry

SGbraitenbergvehicle

creates a simple solid similar to the vehicles of von Braitenberg

In: **BLH**: [Width length height]l default is [30 40 10] **COL**: Optional colors for sensor wheel text; default is [yrc] **NAM**: optional name ontop; default is empty;

Out: **SG**: Solid Geometry

SGcableclip

returns cableclip that can be fixed in 90 degree by a nail

In: **BDW**: Nail length, Diameter of cable, wall thickness; default is [3 5 2]

SGcapontop

returns a solid that covers the geometry from top to a specified length

In: **SG**: Solid Geometry **b**: [z and w] depth and wall thickness **T**: Optional frame relative to SG (Char) to specify upwards **ctyp**: cover type; default is 'default' Try 'plug' or 'full', or 'simple' **cpos**: reference for coverage 'none', 'frame' , 'surface' **offs**: used for plug to make sure that the plug looks outside

Out: **SGC**: Solid Geometry that covers SG from the top with Frames 'F' and 'B' **SG**: Solid Geometry with modified Frame 'F' to adjust to Cap **CPLh**: Convex hull contour **CPL**: Contour of the cap

SGcase

returns the solid geometry for a cubic case with cover

In: **S**: Outer size as vector [x y z] **wall**: Wall Thickness; default is 0.7 **slot**: Slot distance; default is 0.5 for separation of box and cover **lr**: desired raster size; default is no raster **"STL"**: if used an STL directory is written **"InDim"**: if used S contains the inner dimensions

Out: **SGB**: SG of the Box **SGC**: SG of the Cover

SGcatheter

Creates a solid of a catheter

In: **PLR**: Diameter of segements; last value is diamter; default is [8 6 2.33] (8 French) **PLL**: Length of segments; default is [2 25 20]); **HLL**: Length and displacement of handle; default is [11 8]

Out: **SGC**: SG of Catheter **AParts**: Parts list

SGcatheterfeeder

creates rollers to move wires and catheters

In: **ss**: [r1-motorshaft r2-motorshaft r3 roller w-silicon]; default is [9 2 3 2.5]) **typ**: "Single", "Double", **fix**: Kerbkonus default is [2.0 6] **hhh**: [overall-height slider-thickness]; default is [24 1.5]

Out: **SGA**: Roller A **SGB**: Roller B **SGC**: silicon coat solid geometry **AParts**: parts list

SGcathetertorquer

creates different catheter torquers to be activated by SGcathetercapsule

In: **CCtyp**: such as 'MANDRIN', or 'MICRO2.8F.018' **CAP**: size of roller; default is [35 20 14 3 6] mm

Out: **SGA**: Front side (Catheter) including Frames **SGB**: Back side (Luerlock) including Frames

SGchristmasstar

writes a STL file for Printing a christmas star

In: **n**: number of corners; default is 7 **txt**: name ontop **DR**: Diameter; default is 50

SGclampCTtable

returns the solid geometry for a CT table clamp

In: **LL**: Width of clamp **M**: Diameter of metric screw (positive core hole; negative nut); default is 0 **clpar**: same parameter as PLclampCTtable; default may be [40 3 0.5 3]

Out: **SG**: Solid geometry including Frame at centered screw position

SGclosurecapofFS

Automated Design of Closure Cap

In: **SG**: Solid Geometry **fi**: one facet index of FS or empty for interactive selection **t**: thickness; default is one **in**: inlay if true; onlay if false; default is false

Out: **SGcap**: SG of Closure Cap (green) **SGfix**: SG of Fixture (blue) in case of inlay to add to Solid **CPLo**: Outer contour **CPLi**: Inner contour **T**: Frame of the cap **ez**: z-vector of the cap

SGcupholderinsert

returns a SG as insert for cup holder in a car

In: **h**: [height angle width and length; default is [60 17 50 75]

SGdragchain

creates a 3D printable already assembled drag chain

In: **L**: Length **R**: Bending radius

Out: **SG**: Solid geometry

SGdragchainelement

returns a single element of a drag chain

In: **L**: Distance of axis; default is 10 **R**: [Height Width and axis diameter]; default is [8 11 4] **al**: rotating angle; default is pi/2 [0 pi/2] **h**: material thickness; default is 1.2 mm

Out: **SGall**: solid geometry of the drag chain element

SGdragstrip

creates a dragchain cables as texture strip

In: **siz**: [Length Width wall pinheight pinwidth pindistance **ch**: cable heigt; default is 1.4mm **alp**: hook angle; default is 51 degree

Out: **SGA**: Lower Strip **SGB**: Upper Strip

SGdrillnotch

sets a simple drilling notch on top of a surface

In: **T**: Frame or cell{Solid, Framename} **M**: metric diameter of the notch

Out: **B**: the drilling notch solid (cylinder plus cone tip) placed at frame T

SGendoscope

Creates an Endoscope in KARL STORZ Design

In: **D**: Diameter **L**: Length

Out: **SG**: Solid Geometry

SGEurtraccap

Solid Geometry Cap for an EUTRAC 3Phase electric rail

In: **par**: cap parameters [width height length slot wallthickness]; default is [36 32 40 16 1.2]

Out: **SG**: Solid Geometry of a 36mm x 32 mm x 40mm cap

SGfurniturespacer

Solid protector to protect the touch of the furniture to a wall

In: **ddd**: [x y wall distance height]; default is [25 25 3 65 25] **ope**: if true; the parts are open in [x y]; default is [false false]

Out: **SG**: Solid Geometry

SGgearBevel

Returns the vertex/facet list of a bevel gear with involute teeth

In: **M**: Module, default is 1 [mm] **Z**: Number of teeth, default is 20 **H**: Height of involute gear, default is 2 [mm] **Di**: Diameter of the bore, default is 0 [mm] **Beta**: Angle for helical gearing, default is 0 [degree] **Alpha**: Pressure angle, default according to DIN 867 is 20 [degree]. It is Alpha<=acosd(1-(2.25/Z)) **SW**: Step-Width for resolution of the gear's contour, default is 0.5 [mm] **BetaV**: Helical angle for herring bone gearing, default is 0 [degree]

Out: **SG**: SG of bevel gear **tz**: Offset in z-direction [mm]

SGgearchain

returns a gear chain als simple sequence or stacks

In: **m**: [m h r hr] of gear; default is [1 5 1.25 0] **z**: array of number of teeth **a**: [relative angle change for next gear] **w**: optional overall angle of chain; default is not defined; **"stack"**: if used the gears are vertically stacked to a gear box **"rack"**: rack type; default is 'bar'; see CPLrack4PL **"crank"**: if used, the first gear is covered **"stl"**: if used, the radius or CPL is used for gear #1

Out: **SGfin**: cell list of gears **TL**: cell list of Base frame **f**: cell list of ratio **SGR**: SG of Rack

SGgearCrownwheel

Returns the vertex/facet list of a crown wheel with involute teeth

In: **M**: Module, default is 1 [mm] **Z**: Number of teeth, default is 20 **H**: Height of involute gear, default is 2 [mm] **Di**: Inner diameter **T**: Height of base body [mm] **AlphaN**: Pressure angle of the paired spur gear, default according to DIN 867 is 20 [degree]. **hac**: addendum factor [-], default is 1 **ma**: Machine accurracy [mm], default is 0.1 [mm] corresponding to EOS Formiga P100

Out: **SG**: SG of crown wheel **da**: Diameter of the center of the tooth flank

SGgearInternal

returns the vertex/facet list of a gear with internal involute teeth

In: **M**: Normal module, default is 1 [mm] **Z**: Number of teeth, default is -20 **H**: Height of involute gear, default is 2 [mm] **Di**: Distance between df and outer diameter, default is 2 [mm] **Beta**: Angle for helical gearing, default is 0 [degree] **Alpha**: Normal pressure angle, default according to DIN 867 is 20 [degree] **SW**: Step-Width for resolution of the gear's contour, default is 0.1 [mm] **BetaV**: Helical angle for herring bone gearing, default is 0 [degree]

Out: **SG**: Solid Geometry

SGgearpair

returns a combination of two gears

In: **A**: Shaft center distance as sum of the two radii, default is 20 [mm]; Give positive value for spur gear pair and negative value for gear pair with one internal gear **i**: Gear ratio, default is 2.5 []; Has to be divisible by 0.5! Negative value defines a combination of spur gear and gear rack **sigma**: Angle between the axis, default is 0 [Â°]; A value of -90 defines a crownwheel **h**: Height of the gears, default is 3 [mm] **di**: Diameter of an inner bore hole, default is 0 [mm] **type**: Flag for the design of helical ('h') or herring bone ('v') teeth, default is '' ['string]

Out: **SGgp**: Pair of gears for the gear stage [SGT] **SGg1**: First gear of the pair as single SGT [SGT] **SGg2**: Second gear of the pair as single SGT [SGT]

SGgearRack

Returns the vertex/facet list of a gear with involute teeth

In: **M**: Normal module, default is 1 [mm] **Z**: Number of teeth, default is 18 **H**: Height of involute gear, default is 2 [mm] **Di**: Defines Thicknes of the rack base , default is 2 [mm] **Beta**: Angle for helical gearing, default is 0 [degree] **Alpha**: Normal pressure angle, default according to DIN 867 is 20 [degree] **BetaV**: Helical angle for herring bone gearing, default is 0 [degree] **hac**: Addendum factor [-], default is 1

Out: **SG**: Solid Geometry

SGgearSpur

Returns the vertex/facet list of a gear with involute teeth

In: **M**: Normal module, default is 1 [mm] **Z**: Number of teeth, default is 18 **H**: Height of involute gear, default is 2 [mm] **Di**: Diameter of the bore, default is 0 [mm] **Beta**: Angle for helical gearing, default is 0 [degree] **Alpha**: Normal pressure angle, default according to DIN 867 is 20 [degree] **SW**: Step-Width for resolution of the gear's contour, default is 0.075[mm] **BetaV**: Helical angle for herring bone gearing, default is 0 [degree]

Out: **SG**: Solid Geometry

SGgearwheelsocketspanner

This is a socket spanner with clearance fitting for screws

In: **mnw**: [module, n-teeth, reduction from feet-diameter]; default is [1 19 0] **H**: Height of gear; default is 15 **HN**: [Length Width Height height adapter] of handle; default is [30 5 10 10] **"STL"**: if used, an STL folder will be written

Out: **SG**: Solid Geometry of Wrench socket spanner **CPLC**: Contour used

SGhandle

returns a simple handle to be fixed by screws

In: **bsiz**: size of handle [length radius-xy radius height height] default is [100 14 9 20] **dins**: optional screws for bushings such as [2.5 6] **shap**: default is true; use false if there are boolean problems

Out: **SG**: Handle geometry

SGhandrailclamp

An example for a snap-in solution of a handrail-adapter for a specified force

In: **xywF**: [x y w F]; default is [52mm 32mm 2m 20N] **"count"**: if used a countersunk head screw is used instead of a cylindric head screw

Out: **SG**: Solid Geometry

SGhook4railsquared

creates a hook for squared shaped rails

In: **DR**: polyline small function; default is 31 **dH**: thickness of the hook **hr**: function works with 'line','bezier','circ','mattheck'; default is 'mattheck' **"shape"**: contour shape flag, one of 'line','bezier','circ','mattheck'; default is 'mattheck'

Out: **SG**: Final solid geometry

SGiso4026

SG of Hexagon socket set screw with flat point

In: **M**: Diameter (millimeter) **L**: Length (millimeter) **imb**: true=imbus; false=simple screw

Out: **SG**: Solid geoemetry of Hexagon socket set screw with flat point

SGiso4762

SG of Hexagon socket set screw with cylindric head

In: **M**: Diameter (millimeter) **L**: Length (millimeter)

Out: **SG**: Solid geoemetry of Hexagon socket set screw with flat point

SGlemoFAGEGG

Plug and socket for the LEMO FAG EGG family

Out: **FAG**: Plug **EGG**: Socket

SGmagnetcap

returns a closed magnet adapter cap for M22

In: **MDH**: Magnet Diameter and height; default is [15 4] **CDH**: Cap Diameter and Height and Wall; default is [24.8 5 0.4] **SDH**: Screw Diameter and height and Matric; default is [24.8 5 0]; 0=Automatic screw diameter

Out: **SGall**: ={SGNut,SGScr}

SGmakitaadapter

returns the Solid Geometry of an Makita 14V-18V Akku adapter

Out: **SGP**: Geoemtrie for printing or integration

SGmedalsframe

creates a plate to present 60mm medals in combination with a book wedge

Out: **SGC**: solid geometry of the medals presentation plate with pockets

SGmeshground

creates a ground mesh that is a flexible texture on a FDM printer

In: **s**: [x-wdith y-width grid opening wires height]; default is [100 12 2 .45 0] **"fix"**: if used; the grid is adjusted to create a best fit

Out: **SGF**: Solid **xx**: xx values **yy**: yy values

SGmieleTwinDosadapter

returns a Adapter for Miele TwinDos detergent cartridges (print with 80%)

Out: **SGall**: SG of tube and tube with funnel

SGmodelLink4

Creates Solid for a Link im Style "Link4"

In: **L**: Length **L1**: Level of Frame B; default is 0 **L2**: Level of Frame F; default is 1 **K**: Knob position; default is '' **H**: Height of link; default is 9

Out: **SG**: Solid Geometry including Frames **SGA**: Left joint of the link **SGB**: Right joint of the link

SGmotorFeeTech

slow fnct that creates solids with text

In: **model**: motor model selector: 'SM120BL'/'120', 'SM85CL'/'85', 'SM40BL'/'40', etc.; default is 'SM120BL'

Out: **SG**: SOlid Geometry of the Motor

SGmotorshaftcoupling

creates a rigid motor shaft coupling

In: **MM**: [d1 d2]; default is [3 4] **LD**: [L D]; default is 20 9 **DH**: if true there are two grubscews per side **MS**: metrix size of grub scew; defaul is M3 **"thread"**: if true, there is a thread designed; useless for M3 or FDM

Out: **SGA**: SG of coupling

SGmotorspindleclutch

creates a motor spindle clutch between a motor shaft and a steep spindle

In: **DM**: [Shaft-Diameter Shaft-Flattening, Coupling-Diameter Coupling-Height]; default is [3 2.5 9 10] **SP**: Spindle Diameter x Spindle Length and Specified Pitch]; default is [6 10 NaN]; NaN means default **M**: [Bushing-Diameter x Screw-Length; default is [2 4] **slt**: fitting; default is 'c' **isleft**: if true a left-handed thread is used; default is false

Out: **SGA**: Spindle clutch **SGB**: spindle **SGN**: Lock nut **AParts**: cell list parts list

SGmotorwindup

creates wind up motor with a 3d printed spring

In: **SP**: Springparameter as in SGspringSpiral; default is [30,5,0.8,5] **AX**: Axis description such as last parameter in SGspringSpiral; default is [6.4 6 5 0] **fgns**: gear ratio and stack level; default is [0 3]; 0 stands for default values **"nocage"**: if used the gears are not covered by the rack **"axis4"**: if used and value <>0 a 4th axis is integrated into the housing with shaft geometry **"frontw"**: if used in combination with axis4 it has front wheel drive **"STL"**: if used a directory with all parts is written to desktopdir **"anime"**: if used without STL it animates the motor

Out: **SGall**: Cell list of parts

SGmultisocketclamp

returns a clamp to fix a multisocket under a desktop table plate

In: **xywFd**: [x y wall Force opening of the clamp; default is [57 42 2 20 10] **backs**: parameters for a second (back-to-back) clamp; default is [0 0 2 20 10 4]; if backs(1)>0 a double clamp is built

Out: **SG**: Solid Geometry

SGmusicstandextension

supplies extension elements for an orchestra music stand

In: **L**: Length of the extension default is 240 => 240-70 = 17cm extension on the left and right **"STL"**: if used, an STL file is written on desk

SGofCPLcommandui

Simple commandline based 2.5 solid modeler for the use of a caliper

In: **CLINE**: CPL command string, use '?' for info

Out: **SG**: solid Geometry

SGofCPLzbellows

create a straight bellows based on a CPL

In: **CPL**: Contour **z**: z or [zmin zmax] **wave**: [width-wave height-wafe start-end straight resolution] **"STL"**: if used a folder will be written

Out: **SG**: Solid Geometry

SGpedestal

creates simple solids for pedestals

In: **F**: Size of plate; default is [900 700 15] **S**: Size of stands; default is [120 40 50]

Out: **SG**: Solid geometry

SGpinshaft

creates a pin using PLshaftrad

In: **r**: Radius per step **d**: distance between steps **b**: Radius for broken edges; see also PLradialEdges **ramp**: if true use ramps instead of corners; default is false

Out: **SG**: Solid Geometry

SGplugintube

create a tube that can be elongated piece by piece

In: **RR**: [Ri Wt n] [Inner-Radius wall-thickness edges]; default is [8.0 1.5 0]; Minimal wall size is 1.5mm **LL**: [LL LU LO] [length lower-overlap top-overlap]; default is [160 30 30] **sl**: slot size to increase **"cap"**: if used there will be caps to plug-in **"close"**: if closed (LL(2)==0 || LL(3)==0; floor and/or the top are closed

Out: **SG**: Solid of the tube **SGB**: lower cap **SGC**: top cap **AParts**: Parts-List

SGrackbars

creates solid bars to assemble a rack for mechanisms

In: **L**: Length **MM**: Module **d**: [dx dy] spacing between bushings along the rods; default is [20 20] **LY**: length of the Y (cross) rod B; default is 80

Out: **A**: solid geometry of the long axis rod A **B**: solid geometry of the cross rod B

SGradialhooks

Creates hooks for a rotating bayonet catch

In: **RRR**: [Inner-Radius Length Slot-Height Hook-Height optional-tube] default is [20 10 2 5 2 0] **nmax**: maximum numbers of Hooks; default is 5

Out: **SGN**: Solid Geometry of Radial Hooks **CPLh**: Exact contour without fittinhs

SGreolink

Designing a cover for a Reolink camera

Out: **SGall**: cell array with {cover , base plate}

SGreolink2pipeclamp

Produces a bolt plate for screwing the Reolink camera to a double tube clamp with a joint.

In: **element**: 'boltplate' **"STL"**: 'STL' flag to also write an STL file

Out: **SGall**: Silid Geometry

SGrescBox

Creates for the iMedCap Rescue Robotics Project a 1:10 Model of the Patient Box

In: **s**: [length width height ramp profile]; default is [220 70 75 70 4] or so **"STL"**: if used, the box is separated writen to an STL-Directory

Out: **SG**: Cell list of Solids {front-red, back-green}

SGrobotTCPplate

returns a TCP flange to mount Fischertechnik® compatible grooves

In: **bb**: size of Groove block; default is [30 30 5]; 0 is used als default value **sl**: Slot / wall size **"5DoF"**: if used; the plate can be used for 5DoF ERGOSURG RAMBOO robot

Out: **B**: Groove Geometry **SG**: SG of Yannik's TCP plate including Fischertechnik@ tongues

SGrodlink

returns a cylinder with inclined top and bottom surfaces

In: **L**: Length of the rod, distance of the frames **RF**: [Rx Ry] at the upper end of the rod **RB**: [Rx Ry] at the lower end of the rod **ezF**: ez at the upper end of the rod **ezB**: ez at the lower end of the rod **llf**: Length of follower link as factor of max (RF) **w**: rotation angle default is 0

Out: **SG**: Final solid Geometry

SGsbbase

Creates a base for a simple ball bearing four-bar linkage

In: **L**: Distance between A0 and B0; default is [50 0 0] **sdl**: [Di Do th w] Inner and Outer Diameter, width, and wall thickness **hex**: optional diameter of an inner hexagon in the shaft **fit**: if true, the pin use slfit('b') and will interfere; default is false

Out: **B**: Simple Bearing Base for a linkage with NO TEXT

SGsblink

returns a link for insertion of a bearing, optional a sleeve, and an inner hexagon at the shaft end

In: **L**: [L level-F Level-B]; Distance and level [-n .. 0 .. n]; default is [50 1 0] **sdl**: [Di Do b] Inner D, Outer D and width of Bearing; default [6 13 5] **hex**: optional diameter of an inner hexagon in the shaft **fit**: if true, the pin use slfit('b') and will interfere; default is false **rod**: a rod is created instead of a link; use true or ez vectors

Out: **A**: Geometry of the Link **SL_F**: SLEEVE if abs(level)>1 for Pin 1

SGsbpin

returns a pin printined at 80% to be inserted by a hammer

In: **H**: Length/Height chamfer] of the Pin; default is [14 1] **D**: Diameter of the Pin default is 6 **n**: number of copies **s**: fitting 'u' or 'b' (nominal value) **KK**: [KK1 KK2] bushing sizes to subtract at the front/back ends; default is [0 0]

Out: **SG**: Solid Geometry of a pin printined at 80% to be inserted by hammer

SGservoflange

Manually designs a flange connection for a servo motor

In: **info**: [Do Di M N W] **H**: [Height Plate height]; default is [10 0] **sl**: slfit ; default is slfit('c')

Out: **SG**: Solid Geometry of Part

SGshaftaxis

creates shaft axis combinations

In: **LENS**: cell array of length and transition such as {[10 1],[10 1],[10 1]} **CPLS**: cell array of contours such as{5,PLwrench(6.4,6),3) **"rad"**: if rad is used, not the conturs but circle contours are fused

Out: **SGL**: Final Shaft-Axis solid

SGshieldbase

creates a shieldbase

In: **ddd**: [length width height tickness]; default is [117 4 60 2] **rrr**: [stand extension stand height]; default is [10 4]

Out: **SGA**: resulting solid

SGshutterbeltcover

Creates covers for shutter belts as an example for design using SG-Lib

In: **cov**: [Length Width Height]; default is [182 55 5] **"stamp"**: option text string for backside

Out: **SG**: Solid Geometry

SGsnaprivet

returns a solid geometry for a snap rivet

In: **d**: diameter **h**: height (without base and snaps) of single unit **n**: number of units (multiplies height) **F**: Push force to open solid; def= 0 N **s**: base size; default is one **sl**: slot size

Out: **SG**: Solid Geometry

SGspring

returns a cylindric spring

In: **l**: length of spring **d**: outer diameter of spring **n**: number of windings **s**: ratio of compressed spring default is 0.5 **m**: number of parallel yarns **r**: optional radius

Out: **SGX**: Combined spring solid **SG**: Spring solid **SGA**: Spring part A **SGB**: Spring part B

SGspringSpiral

returns a spiral spring geometry

In: **r**: Distance between center and 2nd axis; default is 30 **w**: number of windings as integer of pi; default is 10 * pi **d**: thickness; default is 1.2 mm **hc**: height; default is 5 mm **k**: [wrench-size n-edges diameter n-edges] default is [6.4 4 5 0] **"STL"**: if used a STL file is written **"ccw"**: if used for Freewheel and Spring, torque is ccw and windup is cw **"CPLu"**: if used an optional contour is cut out of the lid (for example shape of SGfreewheel)

Out: **SG**: Solid Geometry of the Spring **SGC**: SG of a safety cage cover **SGL**: SG of a safety cage lid with opening for CPLu **CPLC**: Contour of the cage **CPLX**: Contour of the shaft bearing and the axis bearing

SGspringSpiralbolt

creates the shaft-axis combination for a commercial windup motor spring (80% fillin)

In: **Db**: [center height strength; default is [12 6 0.7] **FA**: [Wrench size; edges; axis diameter edges]; default is [6.4 6 4.5 0] **LA**: shaft-below-spring shaft-above-spring axis above upper shaft]; default is 10 5 25] **HS**: [x-size y-size hook-size distance hook-to center; default is [41 37 6 19]] for 35mm coil **"STL"**: if used, the bolt is written as STL file **"ccw"**: if used the housing is for ccw spring loading **"axis"**: if used, a bore hole for the optional axis is added in the housing floor

Out: **SGall**: {Bolt Housing Spring} print bolt with 80% infill **CPLN**: Inner Contour of Housing **CPLB**: Contour of Bolt to fix the hook **CPLS**: Contour of shaft to remove **CPLA**: contour of axis to remove

SGsteepspindle

returns the geometry of a steep spindle and a nut

In: **MA**: [DM-Diameter | Screw-Length | Nut-Height | Hole-Diameter] all in mm (checked bei DIN13/985) **p**: pitch by winding. Per default it is the DIN13 value multplied with 2 **CPL**: A contour to create a nut; per default the standard DIN985 shape and height is used **sl**: fitting between screw and nut; default is 'c' for clearance **BL**: [Diameter Length Top Length-Bottom] of bearing; default is [0 M M] **isleft**: if true a left thread is used **"cap"**: value following "cap", [diameter height] of an added cap; default is [0 0]

Out: **A**: Solid Geometry of Spindle **B**: Solid Geometry of Nut **C**: Solid Geometry of open Nut **E**: Solid Geometry of Spindle and Nut with conical transition **F**: Plugable spindle **AParts**: Parts list

SGstent

returns a cylindric stent consisting of two spirals

In: **l**: length of spring **d**: outer diameter of spring **n**: number of windings **s**: ratio of compressed spring default is 0.5 **m**: number of parallel yarns **r**: optional radius

Out: **SGX**: Solid mesh **SGA**: Bottom ring **SGB**: Top ring

SGsubtractnaddTL

Basic fnct to support repeated frame based union and subtraction of solids

In: **SGA**: SOlid to add first and subtract later **SGSUB**: SUBTRACTION SOLID **TLX**: List of Frames relative to SGA **SGADD**: SUPPLEMENT SOLID **TSA**: Frame relative to SGSUB and SGADD; default is 'C' **"debug"**: if the string "debug" is present, debug mode is enabled **"caller"**: value following "caller", the caller name for messages; default is title of caller

Out: **SGA**: Final solid

SGtorquefuse

returns a SG consisting of three parts to implement a clutch

In: **dim**: Dimensions **wt**: Wall thickness

Out: **SGC**: Clutch Geometry

SGwebcamcover

Simply a cover for a desktop webcam

In: **dpth**: depth of the webcam; default is 30 (30 for new Logitech, 22 for old Logitech)

Out: **SG**: Solid Geometry

SGwheelspiral

returns a solid of a involute spoked wheel which limit a torque jump or spike

In: **R**: [Ro Ro2 Ri]; default 15 **H**: Heightand width of the spokes; default is [5 2] **n**: number of spokes; if n<0 the spokes are ccw **A**: 'T30' if there is a need for a torx axle; Can also be a CPL **shf**: shaft==true [slfit('t')], axle=false [slfit('c')]; default is true **"DIN433"**: if used a pinring whaser of a DIN433 size is added **"STL"**: if used an STL file is written

Out: **SG**: Solid Geometry **CPL**: CPL that was extruded

SGwheelspoked

returns a solid of a spoked wheel

In: **R**: [Ro Ro2 Ri]; default 15 **H**: Heightand width of the spokes; default is [5 2] **n**: number of spokes **A**: 'T30' if there is a need for a torx axle; Can also be a CPL **shf**: shaft==true [slfit('t')], axle=false [slfit('c')]; default is true **"STL"**: if used an STL file is written **"DIN433"**: if the string "DIN433" is given, a DIN433 washer is integrated

Out: **SG**: Solid Geometry **CPL**: CPL that was extruded

SGwrenchsocketspanner

This is a socket spanner with clearance fitting for screws

In: **Dnw**: [wrench-diameter, n-edges, wall]; default is [6.4 6 2.4] OR solid **H**: Height of Tube; default is 30 **HN**: [Length Width Height] of handle; default is [30 5 10] **"STL"**: if used, the solid is written to an STL file

Out: **SG**: Solid Geometry of Wrench socket spanner **CPLC**: closed polygon list of the socket cross-section (outer circle with inner wrench profile)

sofbendinggirder

returns the expected bending of a girder/cantilever

In: **l**: length of the girder/cantilever **h**: height of the girder/cantilever **b**: width of the girder/cantilever **F**: Applied force at the end of the girder/cantilever **E**: E modulus of material **Slim**: Bending stress limit of material

Out: **s**: estimated bending of the girder/cantilever **alpha**: resulting bending angle of the girder/cantilever **Smax**: estimated bending stress at the fixed side

VLFLexpandPL

extrudes a square along a path given as PL

In: **PL**: List of points **b**: size in x/y-direction **h**: size in z-direction **cl**: true if closed polygon; default false

Out: **VL**: Vertex list of the structure **FL**: Facet list of the structure

VLFLsnapfit

returns VL and FL of a snapping hollow axle

In: **Rl**: Radius of the bore **RA**: Radius of the flange **D**: Width of the axle **Hf**: Height of the flange **H**: Height of the bore **sh**: size of the hook **Hh**: height of the hook

Out: **VL**: Vertex list **FLW**: Facet list

7.2 Parameterized Links and Joints [guide]

Functions in this section:

bofbendinggirder — returns the width of a bended girder

Fofbendinggirder — returns binding force of a bended girder

PLgearrackDIN — returns a PL for gearrack

SGmodelJoint — returns 3 separated Solids for automated design of joints

SGmodelKeyhole — Creates a SG including a key and a keyhole

SGmodelLink1 — SG of a cylindric rod

SGmodelLink2 — Creates Solid for a Link im Style "Link2"

SGmodelLink3 — returns solid geometries for R linkages

SGmodelNode — returns a solid geometry of a post as fixed node

SGsimpleHinge — returns a simple swivel hinge

Detailed description

bofbendinggirder

returns the width of a bended girder

In: **h**: height of the cross section of bended beam **l**: length of the bended beam **s**: bending distance; default h **F**: bending force to achieve distance s; default 10 N **E**: E modulus default 1200 MPa (Polyamid PA12) **Slim**: Limit of bending stress

Out: **b**: width of cross section of a bended girder **alpha**: bending angle **Smax**: bending stress **r**: bending radius **c**: bending contour

Fofbendinggirder

returns binding force of a bended girder

In: **l**: length of the bended beam **h**: height of the cross section of bended beam; default i 1.5 **b**: width of the bended beam; default is h **s**: bending distance; default h **E**: E modulus default 1200 MPa (Polyamid PA12) **Slim**: Limit of bending stress

Out: **F**: bending force to achieve distance s **alpha**: bending angle **Smax**: bending stress **r**: bending radius **c**: bending contour

PLgearrackDIN

returns a PL for gearrack

In: **m**: modul **l**: number of teeth if integer; maximal length if real; **h0**: optional height **tig**: false=tooth a origin; true=gap at origin **re**: rounded edges; default is false **re**: Rack length

Out: **PL**: Point list [n x 3] **l**: exact length **p**: tooth + gap size = pi * m

SGmodelJoint

returns 3 separated Solids for automated design of joints

In: **Ty**: Type such as 'R' **pos**: position parameter **SL**: Design of stator flange (0,1,) default 0 **EL**: Design of mover flange (0,1,) default 0 **EL**: Design of mover flange (0,1,) default 0

Out: **SMJ**: Surface model of the joint **TB**: Base Frame HT **TF**: Follower Frame HT

SGmodelKeyhole

Creates a SG including a key and a keyhole

In: **RA**: Axle Radius **RO**: Outer Radius **rw**: limiting turning angle; default is 2*pi **LA**: [L1 L2 L3] describing key length **LO**: [L1 L2 L3] describing key hole length **HO**: Width of a flange in distance RO; default is 2*RO **slot**: default is 0.3 mm (SLS & SLA) **WI**: inner wall thickness; default is 1

Out: **SGK**: Solid Geometry of Key **SGH**: Solid Geometry of Keyhole **SGP**: Solid Geometry of flange plate **SGT**: Optional Knob on the upper side; separated from SGK **SGB**: Optional Knob on the lower side; separated from SGK

SGmodelLink1

SG of a cylindric rod

In: **L**: Length **D**: Diameter **SL**: Base Level **EL**: Frame Level **RH**: Hole Diameter; default is D/4 **DL**: Length of the flange; default is 2*D~150 degree

Out: **SG**: Solid Geometry **w**: maximum turning angle

SGmodelLink2

Creates Solid for a Link im Style "Link2"

In: **L**: Length **L1**: Level of Frame B; default is 0 **L2**: Level of Frame F; default is 1 **K**: Knob position; default is '' **H**: Height of link; default is 9

Out: **SG**: Solid Geometry including Frames **SGA**: SG of Bearing of Start Joint **SGB**: SG of Bearing of End Joint

SGmodelLink3

returns solid geometries for R linkages

In: **L**: Link length **lb**: Back link length **lf**: Front link length

Out: **SGA**: Solid geometry of the link

SGmodelNode

returns a solid geometry of a post as fixed node

In: **D**: Diameter of the post; default is 10 **L**: Height of the post; default is 15 **R**: Rotation matrix; default is eye(3) **CPL**: Contour of the post; default is PLstar(D,16); **ez**: Orientation vector; [0 0 1]; [0 -1 0] etc.

Out: **SG**: Solid Geometry **h**: height of solid **L**: Length of the node

SGsimpleHinge

returns a simple swivel hinge

In: **H**: height in z **D**: depth in y; default is H/2 **X**: Depth in x; default is D/2 **sl**: slot size; default is 0.3mm for SLS **ch**: chain; default is false **oa**: opening angle starting at 5*pi/4; default is []==closed **ba**: angle intervall [intervall, start]; not supported yet **Ri**: inner Radius; default is R/1.5

Out: **SG**: Complete Solid Geometry **SG1**: Upper Part, 'yellow **SG2**: Lower Part **SG3**: Middle Part

7.3 Automatic Shape Generation [guide]

Functions in this section:

ashft — creates a irrational number between and 66 degree (1.155 rad)

CPLTopCM_SUN — automatic design of a compliant mechanism based on topology optimization techniques.

SGattachMotorN40 — adds motor rack to another rack and subtracts the bore for an motor axis

SGboxbolted — creates a box that is closed by a simple lid cover

SGboxturnswitch — Solid Geometry for 6V Batterie box plus 1 turn-switch to support simple DC-Motor control

SGcablewinch — creates a solid for a cable winch

SGcapontopprojection — create a simple cap ontop

SGcatheterroller — Creates for a cather lock a gear drive

SGclosurecapFDM — creates a top cover for FDM Printing

SGcut2fit — cuts a large solid into smaller pieces to be fixed with screws

SGfreewheel — Creates a freewheel for a wind up motor

SGgearboxhousing — creates a housing for a gear box

SGgearratio — calculates automatically the required gear sequence for a gear box

SGhollowrhombus — creates a hollow soild for FDM printer

SGmotorhousing — returns a housing between Base frame 'B' and end of motor

SGofCPLzscrew — Extrudes a CPL in z with an internal scew to print the part separately as FDM

SGofFSCPLextrude — creates a solid be extrusion of the feature surface selection

SGpipeclamp — designs a foldable double pipe clamp with hinge.

SGrackofSGshaftaxis — design automatically racks for rotating shafts and axis

SGsculpturepedestal — create a pedestal for a sculpture

SGseedcube — Creates a small cube below the surface with just one Frame 'F" to start SGdesingconcepts

SGselectGroupofBBvol — returns the solid ordered by volumes

SGservomounting — shows how to create servo motor mounting sets

SGstamptextFDM — creates an imprint or an relief if allowed

SGTconnectingplug — returns a solid part that can be used to connect both solids by a positive connection

TLconnecting2SG — connects 2 solids by a straigt connecting solid between the center, using feature surfaces

VLFLsnaprivet — returns the solid geometry (VL,FL) of a male snap rivet push pin with a speficied opening force

Detailed description

ashft

creates a irrational number between and 66 degree (1.155 rad)

In: **i**: number between 1 and 25

Out: **a**: angle to turn

CPLTopCM_SUN

automatic design of a compliant mechanism based on topology optimization techniques.

SGattachMotorN40

adds motor rack to another rack and subtracts the bore for an motor axis

In: **SGR**: Solid the motor should be screwed with **SGA**: Solid that will be driven by motor axis **PL**: List of Point **LW**: Dimension of housing length and screw size; see SGdesignMotorN40 **"debug"**: if individual designsteps should be shown

Out: **SGR**: Modified SGRack **SGA**: Modified SGAxis **SGS**: Motor Solid **Parts**: list of parts for assembly

SGboxbolted

creates a box that is closed by a simple lid cover

In: **CPL**: Inner Contour required for assembly **H**: Inner height required for assembly **w**: [wall cover floor screws]; default is [2 2 2 2]; **MM**: [Diameer Length] default is 2.6 6 **"assembly"**: "DIN7991" or "DIN912985", or "DIN912bush" **"radial"**: radius for outside box radius; default is 1 **"screws"**: number of screws; default is 4 **"bolt"**: if used, there are bolts under the screw sockets

Out: **A**: SG of the Box **C**: SG of the Lid/Cover **Aparts**: Parts list for Assembly

SGboxturnswitch

Solid Geometry for 6V Batterie box plus 1 turn-switch to support simple DC-Motor control

Out: **SG**: cell list of solids {box, cover, knob, knob}

SGcablewinch

creates a solid for a cable winch

In: **wsz**: [Ri Ro H] Inner Outer Diameter and Height **drl**: [D l] Diameter of cable and optional length

Out: **SG**: Solid Geometry **lmin**: Cable length

SGcapontopprojection

create a simple cap ontop

In: **SG**: Solid geometry **T**: ez vector for upward, or T matrix **w**: width of shell and cover default is [3 3]; **h**: height of shell; default is full length **sl**: slot; default is slfit('t')

Out: **SG2**: Resulting cap for solid

SGcatheterroller

Creates for a cather lock a gear drive

In: **CLR**: Diameter of segements; last value is diameter; default is [8 6 2.33] (8 French) **CLL**: Length of segments; default is [2 25 20]); **HLL**: Length and displacement of handle; default is [11 8] **PLL**: [Diameter width wall space]; default is [20 12 3 5]

Out: **SGG**: Gear cylinder for cather lock ('white') **SGC**: SG of catheter ('green') **SGP**: Podest for cather lock ('yellow') **AParts**: Parts list

SGclosurecapFDM

creates a top cover for FDM Printing

In: **SG**: Solid geometry to be closed at the top **CPLU**: Optional CPL of an inlay; default is []; **hib**: optonal [height and border] for an inlay; default is [0 1.2]

Out: **SGC**: Solid Geometry of cap AddSG **SGO**: Solid Geometry of post ADDplug **hmin**: height of plug

SGcut2fit

cuts a large solid into smaller pieces to be fixed with screws

In: **SG**: Solid Geometry **maxl**: maximal length in largest dimension **dins**: screw diameter; default is [2.5 6 0] for DIN912 985 **ttyp**: separation type; default is 'L'

Out: **SGS**: cell list of parts

SGfreewheel

Creates a freewheel for a wind up motor

In: **Do**: [% Diameter Height Thickness fixed hooks cavitiesperhook spring thickness] default is [30 5 3 1.2 3 8 1] **SS**: Shaft description ; please add transition fitting values **SP**: if used; all STL files are written into a folder **"STL"**: if used, the freewheel (winding) is in the counter-clockwise direction - Not really required **"ccw"**: if used, the freewheel winding runs counter-clockwise; must match SGspringSpiral

Out: **SG**: {Inner Ring, Outer Ring} **CPLO**: closed polygon list of the outer ring contour

SGgearboxhousing

creates a housing for a gear box

In: **SG**: Cell array of gears created by SGgearchainwindup etc. **w**: [wall floor cover lid] thickness; default is [1.2 1.2 1.2 0.8] **"convex"**: if used the shape is a convex hull **"flooradd"**: this contours are added to the floor in addition (freewheel) **"floorsub"**: this contours are subtracted from the floor in addition (freewheel) **"lidadd"**: contours added to the lid/cover in addition; default 0 (none) **"lidsub"**: contours subtracted from the lid/cover in addition; default 0 (none)

Out: **SGB**: {Wall, Floor, Lid} **CPLi**: Inner axes contour used for the housing **CPLo**: Outer box contour used for the housing

SGgearratio

calculates automatically the required gear sequence for a gear box

In: **f**: desired ration **nn**: optional number of gear pair layers **mm**: minimum and maximum number of teeth; default is [9 55] **"short"**: flag option; when set returns the shorter (first) teeth-sequence solution instead of the last **"anime"**: flag option; when set generates an animated gear chain windup

Out: **zzl**: teeth sequence **rat**: ratio **ee**: relative error

SGhollowrhombus

creates a hollow soild for FDM printer

In: **SG**: Solid Geometry **WNA**: [wall thickness, edge number, angle in degree]; default is [1.2 0 55]

Out: **SG**: Solid with included rhombus to save material and printing time

SGmotorhousing

returns a housing between Base frame 'B' and end of motor

In: **M**: SG Model of Motor **w**: wall width of housing; default is 2mm [w e sl] **D**: Diameter of hole at front side; default is 9mm; 0 and inf is possible **FN**: frame name used as motor base origin; default is 'B'

Out: **C**: Cell array of housing and slinder **P**: Position/frame of the housing

SGofCPLzscrew

Extrudes a CPL in z with an internal scew to print the part separately as FDM

In: **CPL**: Conotur to extrude **z**: z or [zmin zmax] **M**: metric diameter of a screw; default is automatic **ht**: height and minimal wall thickness of the thread; default is [5 1.2] **z0**: cutting position; default is the middle **"STL"**: if used an STL folder is written

Out: **SGScr**: Lower part with the screw **SGNut**: Upper part containing nut and cap

SGofFSCPLextrude

creates a solid be extrusion of the feature surface selection

In: **SG**: Solid Geometry **dz**: elongation **"OLAP"**: if used with a value > 0 ; instead of the Feature Surface a cut projektion (mm) is used **"STL"**: if used, an STL file is written

Out: **SGN**: Extrusion s **SGNN**: Final Solid **CPL**: Used Contour **T**: Transformation matrix

SGpipeclamp

designs a foldable double pipe clamp with hinge.

In: **D**: Diameter of th pipe or CPL contour; default is D=80 **Hw**: Height and width of the clamp; default is [40 5] **Dh**: Diameter of the Hinge; default is 4 ; warning if smaller than 2.4 **MLD**: [Diameter Length displacement] of a DIN912/DIN985 locking; default is 0 **"STL"**: if used an strl file is written

Out: **SGA**: Solid Geometry, complete as SG for grouping use SGanalyzePenetration

SGrackofSGshaftaxis

design automatically racks for rotating shafts and axis

In: **SGC**: cell list of shafts/axis in z direction **h**: desired height and width of rack **zz**: list of z values for the racks; default is [-inf inf]

Out: **SGS**: cell array of racks **CPLR**: cell array of rack contours

SGsculpturepedestal

create a pedestal for a sculpture

In: **SG**: Solid Geometry **Otext**: Text to add on sculpture; Lines separated by '\n' **ht**: desired height in z; default is 130; 0 for original size **ov**: overlap of hte pedestral; default is -1mm **ez**: orientation of text; default is [0 -1 0]; (Backside) **sc**: flag to enlarge the pedestal to fit the sculpture base; default is false

Out: **SG**: SG of complete Figure including pedestral and text **SGB**: SG of pedestral **SGT**: SG of text

SGseedcube

Creates a small cube below the surface with just one Frame 'F" to start SGdesingconcepts

In: **h**: height (size) of cube); **N**: name of the single frame; default is 'F'

Out: **SG0**: A small cube [1 x 1 x h] with one Fram 'F" at position eye(4)

SGselectGroupofBBvol

returns the solid ordered by volumes

In: **SGorg**: Original Solid **gtyp**: group method for SGanalyzePenetration

Out: **SGN**: largest Solid **SGall**: All Solids sorted by volume of bb **vol**: vol of bb **bb**: bounding box

SGservomounting

shows how to create servo motor mounting sets

In: **SG**: STL of Servo **h1**: height of shaft **h2**: height of housing **w**: shaft sweep angle interval; default is [pi/4 0] **fnam**: frame name of the shaft; default is 'Shaft'

Out: **A**: shaft connector solid **C**: servo connector solid

SGstamptextFDM

creates an imprint or an relief if allowed

In: **SG**: Solid Geometry **ez**: surface direction of text **txt**: text **siz**: maximum size; default is 20 **"debug"**: if used, the design is shown **"relief"**: if used insted of an imprint also a relief is allowed **"rot"**: text rotation angle in radians; default is 0 **"trans"**: in-plane translation [dx dy] offset of the text; default is [0 0]

Out: **SG**: Modified SG

SGTconnectingplug

returns a solid part that can be used to connect both solids by a positive connection

In: **SGA**: Solid A with a hole/cavity in direction of Frame FA **SGB**: Solid B with a hole/cavity in direction of Frame FB **FF**: Frame Name of a Frame at Solid SGA; default is 'F' **h**: Frame Name of a Frame at Solid SGB; default is 'B' **ATyp**: height **BTyp**: SGA Positive Connection Type for SGTconnect; default is 'plug'

Out: **PC**: connecting plug solid joining SGA and SGB

TLconnecting2SG

connects 2 solids by a straigt connecting solid between the center, using feature surfaces

In: **SG1**: Solid A to connect **SG2**: Solid B to connect

Out: **SGX**: Connecting Solid **SGA**: SGunion(SG1,SG2,SGX) **TX**: cell {TX1,TX2} with the two connection frames on SG1 and SG2

VLFLsnaprivet

returns the solid geometry (VL,FL) of a male snap rivet push pin with a speficied opening force

In: **H**: Height of the bore **Ro**: Radius of the bore **w**: wall thickness of the snap; default 1 **Rf**: Radius of the flange; default Rb+Dw **Hf**: Height of the flange; default Dw **m**: number of hooks; default 6 **F**: Force of a Hook; default is 10N; **sm**: if true, melts the resulting solid via SGmelting; default false

Out: **VL**: Vertex list **FL**: Facet list

7.4 FEM and Shape Optimization [guide]

7.4.1 Mesh and Model Creation [guide]

Functions in this section:

FEDiscreteGeometryFromSG — similar to pdemodelofSG (2017), but simplified

pdemodelofSG — creates a pde model from a solid geometry

SGmeshtetrahedron — creates a tetramesh of a Solid Geometry - same as SGmeshtetrahedron

SGtetramesh — creates a tetramesh of a Solid Geometry

Detailed description

FEDiscreteGeometryFromSG

similar to pdemodelofSG (2017), but simplified

In: **SG**: Solid Geometry

Out: **fem**: Finite Elemente / Feature Edge model for pde

pdemodelofSG

creates a pde model from a solid geometry

In: **SG**: Solid Geometry **h**: Hmax parameter for generateMesh **e**: e modulus default is 3000Pa = 3e9

Out: **model**: pde model for pde tool **SG**: The solid geometry (passed through)

SGmeshtetrahedron

creates a tetramesh of a Solid Geometry - same as SGmeshtetrahedron

In: **SG**: Solid Geometry **h**: maximum mesh length; default is sofBB/20

Out: **SGn**: Solid Geometry based on a PDE mesh **VL**: Vertex list **FL**: Facet list **TR**: Tetrahedron triangulation

SGtetramesh

creates a tetramesh of a Solid Geometry

In: **SG**: Solid Geometry **h**: maximum mesh length; default is sofBB/20

Out: **SGn**: Solid Geometry based on a PDE mesh **VL**: Vertex list **FL**: Facet list **TR**: Tetrahedron triangulation

7.4.2 Load Definition and FS [guide]

Functions in this section:

pdeplotfaces — simply plots the surfaces to select

SGplotsurfaceload — plots the surface load of a solid geometry

Detailed description

pdeplotfaces

simply plots the surfaces to select

In: **model**: pde model

SGplotsurfaceload

plots the surface load of a solid geometry

In: **SG**: Solid Geometry **fixfacet**: list of fix Feature surfaces (green) **loadfacet**: list of loaded Feature surfaces (red) **loadvec**: vector of load per Feature surface

7.4.3 Stress and Displacement [guide]

Functions in this section:

pdeplotresult — plots the displacement in 3 quadrants

pdesolvesurfaceload — calculates the FEM analysis using pde for a pde mesh model

pdestressstatic — returns the calculated static stress inside a SG based on a pde model

Detailed description

pdeplotresult

plots the displacement in 3 quadrants

In: **model**: pde model created by pdemodelofSG **result**: pde result created by pdesolvesurfaceload

Out: **h**: handle to figure

pdesolvesurfaceload

calculates the FEM analysis using pde for a pde mesh model

In: **model**: pdemodel created by pdemodelofSG **fixfacet**: list of facet that are fixed **loadfacet**: list of faces that are loaded **loadvec**: direction of load

Out: **result**: simulation result **model**: simulated model; called by reference!

pdestressstatic

returns the calculated static stress inside a SG based on a pde model

In: **model**: pde-model (attention: modified / call by reference) **result**: simulation result of function pdesolvesurfaceload **result**: see Property Names

Out: **stresstensor**: List of stress tensors **sVonMises**: static von-Mises stress

7.4.4 Shape Optimization [guide]

Functions in this section:

SGshapeOptiCAO — returns the optimized shape of a given structure based on biological growth

SGshapeOptiSKO — plots the surface load of a solid geometry

Detailed description

SGshapeOptiCAO

returns the optimized shape of a given structure based on biological growth

In: **SG**: Solid Geometry of the initial design proposal(unit is mm) **iFixed0**: see property names **iLoad0**: index of the loaded face; default is 8 **LoadList**: load vector [x y z] applied to the loaded face; default is [0 0 -1e4] **h**: maximum mesh element size; default is s/20

Out: **SG**: Solid Geometry of the optimized structure **result**: FEM anaysis result of the optimized structure(from PDE Toolbox) **model**: pdemodel of the optimized structure(from PDE Toolbox) **IterationAnalysis**: logged information of every optimization iteration

SGshapeOptiSKO

plots the surface load of a solid geometry

In: **SG**: Solid Geometry

Out: **SG2**: Optimized solid geometry **resultSKO**: SKO optimization result struct

7.5 Multi-Body Simulation [guide]

7.5.1 Simscape Model and Block Assembly [guide]

Functions in this section:

getCurrentPoint — this fnct returns the currentPoint but also the axis limits

SGmodelLink — SG of a cylindric rod

SGmodelMass — creates a mass point including 2 frames

smbAddBlock — Auxiliary fnct to create Block for Simulink/SimScape

smbAddFrame — SimMechanics creates a Rigid Transformation Block

smbAddFrameSensor — adds s subsystem that records the frame movement

smbAddLine — SimMechanics creates line between block ports

smbConvertSubsystem — SimMechanics converts blocks to a subsystem

smbCopyConnections — smb copies all existing conntections of a block to another

smbCreateBlockConst — Creates a block with a constant value

smbCreateConnection — adds a connection line between 2 blocks.

smbCreateDrive — SimMechanics adds a position control block to an existing joint

smbCreateJoint — creates a joint block in the actual system

smbCreateSG — SimMechanics creates a subsystem for a Solid Geometry

smbCreateSGJoint — Creating a Subsystem for a 3D Printable Jint

smbCreateSGMass — smbcreates a double mass model with random inertia and point mass

smbCreateSGNode — creates a fixed node in SimMechanics

smbCreateSineWave — Creates a sine Wave as Simulink Signal

smbCreateSpring — smb creates a spring damper link between two point masses

smbCreateStopJointC — smb creates a joint with limits (fixed boundaries)

smbCreateStopJointR — smb creates a joint with limits (fixed boundaries)

smbCreateSubsystem — SimMechanics creates and opens a subsystem

smbDeleteDrive — SimMechanics removes the drive block of a joint

smbDeleteUnconnectedLines — removes unconnected lines in Simulink/SimMechanics

smbDrawNow — shows a snapshot of the simulink/simscape diagram

smbExpandSubsystem — SimMechanics expanding a subsystem

smbFilename — returns a temporary filename or directory

smbfindPMIOPort — finds and renames the SimScape PMIOPort of a subsystem

smbGetAllLines — returns a connecting list of Blocks and Ports

smbGetAllPorts — SimMechanics returns the port of block

smbGetBlockInfo — Auxiliary fnct to get information on simulink/simscape structures

smbGetConnectedBlocks — returns a list with connected blocks of a block or system

smbGetLibraryPath — returns for a simulink/simscape block the library path

smbGetPortofFrame — returns the subsystem Port for a given subsystem Frame

smbGetPorts — SimMechanics returns the port of block

smbGetSubSystempath — smb returns the filepath of the current block

smbhelp — This fnct is intended to provide a quick and efficient introduction to the programmable interface of simulink and simscape.

smbNameofPort — returns the Connection of a Port

smbNewGridPos — return grid positions for new blocks

smbNewLineRouting — smb reroutes the lines in the existing image

smbNewSystem — SimMechanics creates an empty new system

smbPosgcb — sets the gcb to predefined position

smbSetJointInputTorque — smb changes a joint actuation to InputTorque

smbSetPosition — Sets Position Size Orientation of Blocks

smbSetRotationMatrix — Sets a Block Rotation Matrix

smbSetTransformationMatrix — Sets a Block Transformation

smbSetTranslationVector — Sets a Block Translation in mm

smbSGtransP — Moves a SG without changing the

smbshow — shows the system/subsystem diagram of the current block

smbSysPortLineTable — SimMechanics returns the block and ports and line tables

smbWhich — return the full path name for a given block name

Detailed description

getCurrentPoint

this fnct returns the currentPoint but also the axis limits

Out: **p**: 2 point describing the view axis of the last click **ax**: axis limits for each point -1 to small + 1 to big 0=inside of view

SGmodelLink

SG of a cylindric rod

In: **L**: Length **D**: Diameter **SL**: Base Level **EL**: Frame Level **RH**: Hole Diameter; default is D/4 **DL**: Length of the flange; default is 2*D~150 degree

Out: **SG**: Solid Geometry **w**: maximum turning angle

SGmodelMass

creates a mass point including 2 frames

In: **R**: Radius

Out: **SG**: Solid geometry

smbAddBlock

Auxiliary fnct to create Block for Simulink/SimScape

In: **SysName**: System Name; default is gcs **BlockName**: Block name **LibPath**: Library path **BlockExt**: optional Block extension]

Out: **h**: handle to the added block

smbAddFrame

SimMechanics creates a Rigid Transformation Block

In: **FName**: String with the Tranformation Name

Out: **h**: handle to Block

smbAddFrameSensor

adds s subsystem that records the frame movement

In: **block**: Name of coordinate frame in simMultibody

smbAddLine

SimMechanics creates line between block ports

In: **P1**: Name of Port 1 **P2**: Name of Port 2

Out: **h**: handle to Line

smbConvertSubsystem

SimMechanics converts blocks to a subsystem

In: **SSName**: SubSystem Name **blocks**: List of Blocks

Out: **h**: handle to the created subsystem block

smbCopyConnections

smb copies all existing conntections of a block to another

In: **B1**: Block string top copy connections from **B2**: Block string to copy connections to **opt**: 'replace' deletes the Block B1 afterwards; default is ''

smbCreateBlockConst

Creates a block with a constant value

In: **SName**: Generator name; default is 'Const' **EPort**: Optional Port to connect **amp**: optional value; default is 1

smbCreateConnection

adds a connection line between 2 blocks.

In: **F1**: Frame 1 **F2**: Frame 2 **T**: Additional Frame for rotation; 'match','align' **SGL**: reserved solid geometry list; currently unused

Out: **h**: handle to line

smbCreateDrive

SimMechanics adds a position control block to an existing joint

In: **Joint**: Joint name; default is gcb

smbCreateJoint

creates a joint block in the actual system

In: **JType**: Joint Type ['R','H','P','C','E','S','G,'W'];; default='W' **JName**: 'Joint Name'; default=random number **SFrame**: Start Frame to connect (must exist already); default='' **EFrame**: End Frame to connect (must exist already); default=''

Out: **BName**: Block handle name

smbCreateSG

SimMechanics creates a subsystem for a Solid Geometry

In: **SG**: Solid Geometry **SGName**: SG Name within SimMechanics; default is 'SOLID' **SGcol**: Solid Color ('w','.....'); default is SG.col **SFrame**: Optional Block to connect the Base **EFrame**: Optional Block to connect the Follower

Out: **u**: name of SG block in Sim Mechanics

smbCreateSGJoint

Creating a Subsystem for a 3D Printable Jint

In: **JTyp**: Rotational Type used for SGmodelJoint **SGName**: Name within SimMechanics **A**: Solid Geometry **SFrame**: Block to be connected optionally to Stator **EFrame**: Block to be connected optionally to Mover

smbCreateSGMass

smbcreates a double mass model with random inertia and point mass

In: **SG**: Solid Geometry; default is a sphere with radius 1 **SGName**: Optional Name; default is 'MASS' **SGcol**: Optional Color

Out: **SGName**: name of the created mass subsystem block

smbCreateSGNode

creates a fixed node in SimMechanics

In: **p**: position [x y z] **SGName**: Name; default is Node **SGcol**: Color; default is 'k' **R**: Rotation Matrix; default is eye(3)

smbCreateSineWave

Creates a sine Wave as Simulink Signal

In: **SName**: Generator name; default is 'Sine Wave' **EPort**: Optional Port to connect **amp**: optional amplitude; default is +pi

smbCreateSpring

smb creates a spring damper link between two point masses

In: **M1**: Mass 1 **M2**: Mass 2 **SGName**: optional name; default is 'SPRD; **len**: len in mm **sprg**: spring constant in N/m **damp**: Damping in N/(m/s)

Out: **h**: spring-damper block

smbCreateStopJointC

smb creates a joint with limits (fixed boundaries)

In: **Name**: Local Name of the joint in the diagram **bound**: boundaries; default is [-10mm +10mm]

smbCreateStopJointR

smb creates a joint with limits (fixed boundaries)

In: **Name**: Local Name of the joint in the diagram **bound**: boundaries; default is [i2*pi +2*pi]

smbCreateSubsystem

SimMechanics creates and opens a subsystem

In: **SSName**: Namestring of a Subsystem

smbDeleteDrive

SimMechanics removes the drive block of a joint

In: **Joint**: Name of joint; default is gcb

smbDeleteUnconnectedLines

removes unconnected lines in Simulink/SimMechanics

In: **asys**: system to handel; default is gcs

smbDrawNow

shows a snapshot of the simulink/simscape diagram

In: **dsys**: optional system name; default is bdroot **xy**: optional forced image size; [x y]; default is automatic

Out: **I**: Image captured

smbExpandSubsystem

SimMechanics expanding a subsystem

In: **SSName**: Name of Subsystem. If empty gcb is used

smbFilename

returns a temporary filename or directory

In: **FName**: desired Filename

Out: **fn**: full path of the desired filename

smbfindPMIOPort

finds and renames the SimScape PMIOPort of a subsystem

In: **SName**: subsystem name **renam**: if true the 2020a names are converted into 2018a names

Out: **PMIO**: names of the PMIO Ports

smbGetAllLines

returns a connecting list of Blocks and Ports

In: **Block**: Block, Block list, System

Out: **LL**: List of connections (n x 4)

smbGetAllPorts

SimMechanics returns the port of block

In: **SName**: Name of System or Block; if empty gcs is used

Out: **PL**: List of all Port Strings **CL**: List of all Connectable Ports **RCL**: list of real connector port names derived from the connectable ports

smbGetBlockInfo

Auxiliary fnct to get information on simulink/simscape structures

In: **Block**: Block Name; default is gcb

Out: **Name**: Name of Block **Blockpath**: Part of Block **LibPath**: Library Path

smbGetConnectedBlocks

returns a list with connected blocks of a block or system

In: **Block**: Block or System or list of Blockss

Out: **CBL**: Connect Block List (cell)

smbGetLibraryPath

returns for a simulink/simscape block the library path

In: **B**: Block or if empty => gcb

Out: **Lpath**: Library path **Bpath**: Path to current subsystem

smbGetPortofFrame

returns the subsystem Port for a given subsystem Frame

In: **FName**: Frame Name in a Subsystem

Out: **PName**: Name of Port in the Current System

smbGetPorts

SimMechanics returns the port of block

In: **Block**: Name of Block including system; if empty gcb is used

Out: **PN**: Cell of Port Strings **LP**: cell of all lines; column 1 line handle, column 2 source block, column 3 destination block **oi**: indices of lines whose destination is the block (outgoing lines) **ii**: indices of lines whose source is the block (incoming lines)

smbGetSubSystempath

smb returns the filepath of the current block

In: **block**: Block (path or handle)

Out: **SubName**: Subsystem file path **block**: Block handle

smbhelp

This fnct is intended to provide a quick and efficient introduction to the programmable interface of simulink and simscape.

smbNameofPort

returns the Connection of a Port

In: **b**: Block Name **ph**: Port Name used

Out: **s**: String

smbNewGridPos

return grid positions for new blocks

In: **Nr**: Current number of Grid position; default is 0 **Grid**: Grid size; default is [100 100] **Colum**: Number of columns; default is 5

Out: **p**: [x y] of a grid **Np**: Number in the grid

smbNewLineRouting

smb reroutes the lines in the existing image

In: **asys**: system; default is gcs

smbNewSystem

SimMechanics creates an empty new system

In: **SName**: Desired name for new system (No spaces allowed) **grav**: gravity vector; default is 5.66 m/s2 * [-1 -1 -1] (9.81)

smbPosgcb

sets the gcb to predefined position

In: **Nr**: Grid position

smbSetJointInputTorque

smb changes a joint actuation to InputTorque

In: **Joint**: Unique(!) Joint name 'R1' or 'R1.J' **InpPS**: Physical-signal input driving the joint torque

smbSetPosition

Sets Position Size Orientation of Blocks

In: **BName**: Block name or gcs **P**: Position **O**: orientation('right','up','left','down'; default is is unchanged **s**: size; default is unchanged

smbSetRotationMatrix

Sets a Block Rotation Matrix

In: **BName**: Block Name or gcb **R**: 3x3 Rotation Matrix

smbSetTransformationMatrix

Sets a Block Transformation

In: **BName**: Block Name or gcb **T**: 4x4 Transformation Matrix **rel**: default is 'match'

smbSetTranslationVector

Sets a Block Translation in mm

In: **BName**: Block Name or gcb **t**: Translation Vector (x y z)

smbSGtransP

Moves a SG without changing the

In: **SGName**: Solid Name that has Baseframe and Follower Frame **delta**: Delta vector

smbshow

shows the system/subsystem diagram of the current block

In: **block**: block handle';default is gcb

smbSysPortLineTable

SimMechanics returns the block and ports and line tables

In: **SName**: Name of the Simulink system/subsystem

Out: **bht**: Table of blocks, ports and lines

smbWhich

return the full path name for a given block name

In: **BName**: Block name

Out: **FullName**: full name of the block including system **a**: cell list of all found systems and blocks searched

7.5.2 Simscape Library and Blocks [guide]

Functions in this section:

smbPSBlockname — return the name of a block realtive to Library

smbPSLibCompile — compiles the PS Library in the current smbPSLibname

smbPSLibCreateBlock — Writes the code of a block as .ssc file

smbPSLibInstall — Installs a collection of additional blocks

smbPSLibname — returns the name a SimMultiBody Library Directory

smlArduinoexample — creates a servo motor simulink model for stand-alone application on arduino

smlDeleteall — delete all lines and block of a block diagram

smlhelp — simply open the help page for programmatic model editing

smlNewSystem — opens a new simulink sheet - overwrites an exiting one

Detailed description

smbPSBlockname

return the name of a block realtive to Library

In: **block**: block description

Out: **bn**: Block path

smbPSLibCompile

compiles the PS Library in the current smbPSLibname

smbPSLibCreateBlock

Writes the code of a block as .ssc file

In: **FString**: String of the code of a blockfuunction **Fext**: desired extension; default is '.ssc' **FName**: Filename to be used

Out: **FName**: Analyze Filename only if FName =''

smbPSLibInstall

Installs a collection of additional blocks

smbPSLibname

returns the name a SimMultiBody Library Directory

In: **FName**: smb_function name; default is

Out: **fn**: smb_function file name;default is directory name

smlArduinoexample

creates a servo motor simulink model for stand-alone application on arduino

smlDeleteall

delete all lines and block of a block diagram

In: **asys**: default is gcs **typ**: default is {'line','block}

smlhelp

simply open the help page for programmatic model editing

smlNewSystem

opens a new simulink sheet - overwrites an exiting one

In: **SName**: Name of the new system such as **Coder**: Name of Sheet Example; default is 'arduino'

7.5.3 Simulation Run and Output [guide]

Functions in this section:

smbFullModelSimulation — returns a SG configuration at a given time

smbLogOutputPort — activates a Blocks's Output Port for Data logging

smbSetSimulationtolerance — ability to change the Simulation Tolerance for smbSimulate

smbSimulate — starts the simulation of simscape multibody root

smbTofSimOut — returns T matrix and Time from a simulation

smbVideoSimulation — creates a video an shows the title in a figure

Detailed description

smbFullModelSimulation

returns a SG configuration at a given time

In: **t**: Optional time value; default is 0

Out: **SG**: Compiled SG at the specific given time **BNi**: Names of the used solid geometries **SGi**: Solid geometries **Ti**: T matrixes of the solids over time **tm**: list of time values

smbLogOutputPort

activates a Blocks's Output Port for Data logging

In: **block**: Block **Nr**: Output Port Number **SigNam**: signal name; default is 'OutP.<Nr>'

Out: **ph**: handle to the logged output port

smbSetSimulationtolerance

ability to change the Simulation Tolerance for smbSimulate

In: **tol**: default is 0.001; USE 0.1 for MTT ==> ConsistencyTolFactor

smbSimulate

starts the simulation of simscape multibody root

In: **st**: stop time; default is 10 **tol**: simulation tolerance to avoid singularity problems; default is 1

Out: **simOut**: simulation result **xout**: xout parameter **sout**: logged signal output ('sout')

smbTofSimOut

returns T matrix and Time from a simulation

In: **simOut**: Simulation Result **Name**: Name of Frame

Out: **T**: Transformation matrix [4 x 4 x n] **time**: time list [nx1]

smbVideoSimulation

creates a video an shows the title in a figure

In: **st**: Stop time; default is 10 **vFName**: Optional File name

Out: **i**: Image of Frame 1 **vFName**: Created FileName

7.6 Engineering and CAD Standard Parts [guide]

7.6.1 DIN/ISO Standards [guide]

Functions in this section:

DIN13 — returns the DIN-13 table for a metric threads

DIN20273 — Diameter of drilling holes wrt to DIN EN 20273

DIN336 — returns the parameter of the DIN 336 table

DIN433 — returns the DIN433 table for a 2D Metric Washer

DIN464 — returns the DIN464 table for a metric threads with knureld head

DIN4AMfitting — DIN similar table for fitting offsets with different AM-Printers

DIN4AMtemperature — request for tested printing temperatures

DIN7991 — returns the DIN965/ISO7046 table for a metric threads

DIN912 — returns the DIN912/ISO4762 table for a metric threads

DIN913 — returns the DIN913/ISO4026 table for a metric threads

DIN934 — returns the DIN934 table for a metric threads

DIN965 — returns the DIN965/ISO7046 table for a metric threads

DIN985 — returns the DIN985/ISO476 table for a metric threads

DIN_S_BANCLOK_E — returns the Dimensions of the Kerbkonus Product S-BANCLOK-E

DINfindinTab — searches an entry within a table name

DINhelp — open different urls for DIN and ISO Standard Machine Elements

DINthreadtorque — returns the DIN912 torque for tightening screw threads

DINtorx — returns the main parameter for designing a metric torque adapter

Detailed description

DIN13

returns the DIN-13 table for a metric threads

In: **m**: metric treat diameter **Srch**: parameter name sich as 'pitch' to replace M as return parameter **larg**: if true, return the parameter selected by Srch as first output instead of the diameter M; default true **"MofD"**: string option 'MofD'; if set, look up the table row by core diameter instead of by nominal diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DIN20273

Diameter of drilling holes wrt to DIN EN 20273

In: **M**: 0 for table or M for line **typ**: fine, middle or coarse as optionl default is empty

Out: **M**: Used metric Diameter **TL**: Table of Washers diameter **c**: recommendation for play

DIN336

returns the parameter of the DIN 336 table

In: **d**: bore diameter

Out: **d**: Exact Diameter **P**: Pitch **b**: Core drill hole

DIN433

returns the DIN433 table for a 2D Metric Washer

In: **M**: Desired metric diameter

Out: **M**: Metric Diameter in Tab **TL**: Table row **Taborder**: Table colum names

DIN464

returns the DIN464 table for a metric threads with knureld head

In: **M**: Desired metric diameter

Out: **M**: Metric Diameter in Tab **TL**: Table row **Taborder**: Table colum names

DIN4AMfitting

DIN similar table for fitting offsets with different AM-Printers

In: **M**: Search term such as "Formiga' **Q**: Optional search term in Taborder

Out: **M**: Number oder Char such as "Formiga' **TL**: Table or table entry if Q is used **Taborder**: Name of Table cols **AMList**: cell list of AM printer names and their fitting-set codes

DIN4AMtemperature

request for tested printing temperatures

In: **quest**: known informaiton as cell list such

Out: **answ**: table entry

DIN7991

returns the DIN965/ISO7046 table for a metric threads

In: **M**: metric treat diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M **Taborder**: Names of entries

DIN912

returns the DIN912/ISO4762 table for a metric threads

In: **M**: metric treat diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DIN913

returns the DIN913/ISO4026 table for a metric threads

In: **M**: metric treat diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DIN934

returns the DIN934 table for a metric threads

In: **M**: metric treat diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DIN965

returns the DIN965/ISO7046 table for a metric threads

In: **M**: metric treat diameter

Out: **M**: M that was used for the table entry **TL**: Table entry for M **Taborder**: Names of entries

DIN985

returns the DIN985/ISO476 table for a metric threads

In: **m**: metric treat diameter **Srch**: column name to return from the table; default is 'M' **larg**: if true, choose the larger table entry on inexact match; default is true

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DIN_S_BANCLOK_E

returns the Dimensions of the Kerbkonus Product S-BANCLOK-E

In: **M**: metric treat diameter **Srch**:

Out: **M**: M that was used for the table entry **TL**: Table entry for M

DINfindinTab

searches an entry within a table name

In: **M**: Scalar to find in Tab **Tab**: Table with M as first col; Table name is important **takelarg**: if true nearest larger is selected; default is true

Out: **TL**: Table row

DINhelp

open different urls for DIN and ISO Standard Machine Elements

DINthreadtorque

returns the DIN912 torque for tightening screw threads

In: **m**: metric diameter **mat**: material; default is V2A

Out: **Tq**: Torque **TL**: Table

DINtorx

returns the main parameter for designing a metric torque adapter

In: **Mot**: metric, or 'M2.5' or 'T30'

Out: **M**: metric value **TL**: [Metric, Torx, PLhexalobular-Diameter, tightening torque]

7.6.2 Standard Components (DIN/ISO parts) [guide]

Functions in this section:

SGbitISO1173 — Bite Geometry related to Bit DIN 3126 ISO 1173

SGDIN13 — returns a screw geometry related to DIN13 (same as SGscrewDIN)

SGDIN433 — Design a washer in accordance with DIN433

SGDIN464 — SG of a knureld screw

SGDIN7991head — countersunk screw DIN 7991

SGDIN912 — creates a Solid Geometry with respect to DIN 912

SGDIN912head — creates a Solid Geometry of a screw head with respect to DIN 912

SGDIN965 — returns the DIN965/ISO7046 table for a metric threads

SGDIN965head — returns the DIN965/ISO7046 table for a metric threads

SGDIN985 — creates a Solid Geometry with respect to DIN 985

SGDINbushingE — creates a solid for a kerb konus bushing

SGfourbarlinkageA0B0DIN912985 — Designs a 4bar-Linkage as solid geometry

SGgearandrackDIN — returns a gear and gear rack combination

SGgearDINrot — Calculate a gear with the count from 9.12 to 48; for internal and external gears

SGgearpairDIN — provides a simple gear ratio stage as Solid Geometry

SGgearrackDIN — returns a gear rack for a gear

SGgearwormDIN — returns a gear worm solid

SGlockingDIN912DIN985 — Returns two bodies with DIN912/DIN985 locking method

SGManipulatorDIN912DIN985ballbearing — returns all the parts of a 7DoF manipulator than can be assembled by screws and ballbearings

SGslidingcarriage — returns a slider for 2 rails

SGslidingcoverofCPLz — returns the part of a plane sliding cover for a solid with defined wall sizes

SGtorxDIN — returns a TORX Pin with an optional opening for a kerbkonus bushing

Detailed description

SGbitISO1173

Bite Geometry related to Bit DIN 3126 ISO 1173

In: **n**: Bit size/number **l**: Length **d**: Diameter

Out: **SG**: Solid Geometry

SGDIN13

returns a screw geometry related to DIN13 (same as SGscrewDIN)

In: **D**: Diameter (negative creates a screw cutter or nut) **L**: Length **p**: optional pitch **CPL**: Outer Diameter or CPL, creates nuts when D<0 and Holes if D>0 **shr**: shrinkage for nuts; default is 0; use slfit('c') or NaN for steep thread nuts **left**: if true, a left winding is designed **"Type"**: "A" (default) DIN13, "B" for steep thread with 1 winding, "C" for steep thread and max winding **"STL"**: if used an STL file will be writtemn

Out: **SG**: Solid Geometry **kd**: core diameter

SGDIN433

Design a washer in accordance with DIN433

In: **M**: Diameter metric

Out: **SG**: Solid Geometry including Frames **TL**: table row entry for the selected metric M from the DIN433 table (via DINfindinTab)

SGDIN464

SG of a knureld screw

In: **M**: Metric Diameter **L**: Length of Screw **H**: Optional knurled head size

Out: **SG**: Solid Geomtry **M**: Metric size M used

SGDIN7991head

countersunk screw DIN 7991

In: **M**: metric treat diameter **L**: if true; the dimensions are ISO conform not DIN; default is DIN

Out: **SG**: countersunk screw head Geometry **TL**: length of the head

SGDIN912

creates a Solid Geometry with respect to DIN 912

In: **M**: Diameter metric **L**: Length metric mm

Out: **SG**: Solid Geometry including Frames **M**: actual metric M used for the DIN912 table entry

SGDIN912head

creates a Solid Geometry of a screw head with respect to DIN 912

In: **M**: Diameter metric **Tp**: Length metric mm **ch**: if true, the head is chamfered (SGofCPLzchamfer); default is true

SGDIN965

returns the DIN965/ISO7046 table for a metric threads

In: **M**: metric treat diameter **L**: Length of the screw including head **iso**: if true; the dimensions are ISO conform not DIN; default is DIN **typ**: fine, middle or coarse if subtraction solid for drilling hole

Out: **SG**: countersunk screw head Geometry **l**: length of the head

SGDIN965head

returns the DIN965/ISO7046 table for a metric threads

In: **M**: metric treat diameter **iso**: if true; the dimensions are ISO conform not DIN; default is DIN **ch**: Chamfer / countersink height

Out: **SG**: countersunk screw head Geometry **l**: length of the head

SGDIN985

creates a Solid Geometry with respect to DIN 985

In: **MM**: Diameter metric **CPL**: optional CPL cross-section contour of the nut; default is a hexagon PLcircle(TL(2)/2*dofn(6),6) **p**: optional pitch parameter passed to SGDIN13; default is empty

Out: **SG**: Solid Geometry including Frames **TL**: DIN985 table entry for the selected metric M

SGDINbushingE

creates a solid for a kerb konus bushing

In: **M**: metric 2.0, 2.5, 3, 4, 5, 6

Out: **SG**: Solid Geometry **TL**: Table from DIN_S_BANCLOK_E **SGX**: Subtraction solid for intereference fit (hammer or press)

SGfourbarlinkageA0B0DIN912985

Designs a 4bar-Linkage as solid geometry

In: **CPL**: [A0; B0; B1; A1; [GPLA GPLB]] to define the 4bar linkage | Nr of PS.solut **wlim**: angle limitatins; if empty the full intervall is used **R12o**: Outer an inner radius for linkage [Rout Rin]; default is R12ofCPL **PS**: Pose including fields CPLE **MM**: [metrix diameter metric length overlap; default is [2.5 6 0] **MPL**: Rack assembly point to add **FIX**: fixation method '985' or 'bush' **bbb**: radial bearing spec; if empty a plain DIN912/985 connection is used

Out: **SGall**: {SGC,SGE,SGS,SGG,SG0}

SGgearandrackDIN

returns a gear and gear rack combination

In: **m**: module **z1**: teeth number of gear **z2**: teeth number of rack **R**: bore of gear **h**: height of gear and rack

SGgearDINrot

Calculate a gear with the count from 9.12 to 48; for internal and external gears

In: **m**: modul; negativ values for internal gears; 0 for testing **z**: number of teeth **R**: radius for a shaft/axle or CPL for inner contour **h**: height of the gear; or [zmin zmax slot]; Base = eye(4); [h(1)+slot h(2)-slot-space(1)] **tig**: false - gap; true = tooth; at zero line; default is false; **al**: desired angle for the following tooth frame **"space"**: [height and width of spacer] that is always on the upper side default is [0 1.2] **"wmin"**: height and width of a spacer always placed on the upper side; default is [0 1.2]

Out: **SG**: Solid Geometry **al**: angle in steps of 2*pi/z **df**: foot (root) circle diameter of the gear from PLgearDIN

SGgearpairDIN

provides a simple gear ratio stage as Solid Geometry

In: **m**: modules **z1**: number of teeth 1 **z2**: number of teeth 1 **h**: height of the gear **R**: radius of axle **d**: height of the bearing

Out: **SG1**: Gear 1 **SG2**: Gear 2 **SG3**: Bearing 1 **SG4**: Bearing Top

SGgearrackDIN

returns a gear rack for a gear

In: **m**: module **z**: number of teeth if integer or length if real **h0**: base size **b**: width of rack / height of gear **tig**: false = gap; true= tooth **re**: rounded edges; default is false

Out: **SG**: Solid Geometry

SGgearwormDIN

returns a gear worm solid

In: **m**: modul **dw**: diameter middle circle **wg**: number of windings; default is 5; if wg<0; Len=abs(wg) **R**: Radius or CPL for an inner contour

Out: **SGW**: Solid of gear work **SGX**: Gear and Shaft

SGlockingDIN912DIN985

Returns two bodies with DIN912/DIN985 locking method

In: **M**: [M Diameter Length and distance] of DIN 981 Screw **SGs**: cell with Solids {SGA, SGB} **FNs**: cell with Frame Names {'F','B'} or simple relation such as behind **"supp"**: if used, material is supplemented to hold the screw, nut and head **"align"**: if used, the position of SGB is aligned with A, otherwise remains unchanged.

Out: **SGs**: cell list {SGA,SGB} mit new frames 'C' if spatial relations were used

SGManipulatorDIN912DIN985ballbearing

returns all the parts of a 7DoF manipulator than can be assembled by screws and ballbearings

In: **LL**: Distance between Axis; default is 40 **LR**: Radius; default is 5.1 **LH**: Height default is 7.1 **BS**: Ball Bearing size (Ri, RO, W] default is [2.5 7 3.5] **SS**: Screw Seize {L oh] Length overhead default is [8 -1]

Out: **A**: Baser plate **B**: 1st link and 4th link **C**: optiona 7DoF **D**: 2nd link **U**: 3rd link and 5th link **E**: 6th link end effector

SGslidingcarriage

returns a slider for 2 rails

In: **bbb**: [Ri Ro W] Inner radius, outer Radius width of bearing **xaL**: Distance of bars; and width of carrier; default is [60 max] **w**: [wb wc] width of slider and thickness of carriage; default is [1.5 3] **sIt**: fitting char or slot size; default is 't' for slfit **BL**: thickness of rail cover; default is 0 (no cover) **H**: Height of plattform aboce rails; default is min

Out: **SG**: Solid geometry of carriage **SGE**: Solid geometry of end block **SGC**: Solid geometry of cover **SGF**: solid geometry of the final end block **AParts**: parts list **CPLC**: rail cover projection contour

SGslidingcoverofCPLz

returns the part of a plane sliding cover for a solid with defined wall sizes

In: **CPL**: Contour of a convex hull **w**: [w h e sl] wall thickness of the walls ; height of the plate **T**: Frame to align the cover; not supported yet **op**: [x y] for an opening slot at y-min

Out: **SG**: cellPlate consisting of at least three parts

SGtorxDIN

returns a TORX Pin with an optional opening for a kerbkonus bushing

In: **D**: Diameter or string 'T30' etc. **L**: length and edge; default is [14 0.3] **KK**: kerbkonus metrix [top bottom] default is [0 0]; could be autom. too **sI**: fitting only for SGX; default is 'c'

Out: **SG**: Solid Geometry **SGX**: Subtraction solid **TL**: selected table of DINtorx **PLX**: hexalobular subtraction contour PL used for the fitting

7.6.3 Automatic Part Designs [guide]

Functions in this section:

SGcutT4design — creates a DIN912-DIN985 joint in an arbitrary solid by mouse click

SGdesignAxlebearing — cretaes the material subtraction from a solid for radial bearings and axle

SGdesignBallbearing — creates subtraction solids for a bearing connection at a specific frame position of a solid

SGdesignBoltedLid — creates a cover lid and a fixation for an open 2.5 designed box

SGdesignConeconnector — designs a peg in hole connection using a truncated cone that can be printed with FDM

SGdesignCutDIN912DIN985 — Creates a body cut with subsequent screwing possibility bue straigt cut or z cut

SGdesignDIN7991 — creates a an assemly methods for DIN7991 or DIN965

SGdesignDIN7991BushingE — creates subtraction solids for a screw fixation at a specific frame position of a solid

SGdesignDIN912BushingE — creates subtraction solids for a screw fixation at a specific frame position of a solid

SGdesignDIN912DIN985 — creates subtraction solids for a screw fixation at a specific frame position of a solid

SGdesignDIN912DIN985axialbearing — creates subtraction solids for a screw fixation and a axial bearing at a specific frame position of a solid

SGdesignDIN912DIN985axradbearing — creates subtraction solids for a screw fixation and a axial bearing at a specific frame position of a solid

SGdesignDIN912DIN985radialbearing — creates subtraction solids for a bearing connection at a specific frame position of a solid

SGdesignDrivenshaft — Designs a torque transmitting shaft

SGdesignDriveSpindle — returns the solid geometry of a gripper spindle

SGdesignfaceconnectingrod — creates a connecting rod between two faces of a solid

SGdesignGripperjaw — returns the geometry of gripper jaws, gripper finger and complete mechanisms for an object to grasp

SGdesignIgusdragchain — creates the subtraction and supplemet solids for fixing a igus drag chain

SGdesignMotorN20 — creates subtraction solids for the design of motor driven mechanisms

SGdesignMotorN40horiz — Designs a subtraction solids for the N20 L-type encoder motor

SGdesignPeginHole — design subtraction solid and supplement solid for peg in hole connections using fittings

SGdesignPintle — Designs a simple pindle bolt connection

SGdesignPluginFrame — adds a plug-in sleeve to a solid

SGdesignSupplement — creates cylinders or CPLs that are big enough to encapsulate the fixture elements

SGdesignSupplement2connect2SG — first appraoch to create bridges between two solids

SGdesignSupplement2connectSG — fills in a solid block to connect the surfaces of SG

SGdesignSupplementPlugin — designs a plugin holder for another solid

SGdesignTongueGrooveRadial — creates a radial tongue and groove connection for a contour or an opening in a 2.5D structure

SGtoolTip — returns a Subtraction Solid for a "tool" for SGbool

Detailed description

SGcutT4design

creates a DIN912-DIN985 joint in an arbitrary solid by mouse click

In: **SG**: Solid Geometry **T**: T (z is upward towards red), X is right hand **ct**: 'a' for 'axial, 'c' for coronal, 's for sagittal; default is 'a' **SSG**: Subtraction Solid, SSG.H=Head, SSG.N=Nut, SSG.S=screw

Out: **CU**: Tibia - lower limb - green Radius/Ulna **CO**: Femur - upper limb - red Humerus **T2**: Cutting Frame - that can be used for SGcutT (x/y plane cuts)

SGdesignAxlebearing

cretaes the material subtraction from a solid for radial bearings and axle

In: **SG**: Solid geometry that needs a bearingx **FN**: Framename or T or {'FN1', 'FN2'} **sbb**: [Ri Ro depth os], default is [2.5 6 1.8] **sch**: search for walls; default is false

Out: **SG**: Modified Solid **SS**: Length of Axle **S**: screw/bearing model solid for SGdesignSupplement **dd**: axle depth measured between the crossing faces

SGdesignBallbearing

creates subtraction solids for a bearing connection at a specific frame position of a solid

In: **sdl**: [Ri Ro Width Ol] **ins**: Insertion Tunnel **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10

Out: **P**: Subtraction Solid for plug side including frame 'C' **N**: Subtraction Solid for bearing side including frame 'C' **S**: Geometry of Plug including frame 'C' **B**: Bearing Model

SGdesignBoltedLid

creates a cover lid and a fixation for an open 2.5 designed box

In: **SG**: Solid Geometry without a cover lid **T**: Frame Name, 4x4 matrix or ez-vector; default is [0 0 1] **MM**: [diameter length] metrix of screw; default is [M2.0 6] **w**: width of walls and height of cover **H**: size of bolts **"assembly"**: DIN7991 or DIN912bush or DIN912985 **"screws"**: number of screws; default is 4 **"bolt"**: if used, there is a bolt below the screw fixation

Out: **A**: New Solid with scrw fixation **C**: Cover lid **S**: screws used **mt**: not used yet, for compatibilty with other SGdesignfunction **Aparts**: parts list

SGdesignConeconnector

designs a peg in hole connection using a truncated cone that can be printed with FDM

In: **RHA**: Radius Height Angle; default is [2.0 10 55] **Rmin**:

Out: **Hsub**: Subtraction Solid for Ground Level **Nsub**: not used yet **S**: not used yet **Hadd**: not used yet **Nadd**: Supplement Solid for Top Level

SGdesignCutDIN912DIN985

Creates a body cut with subsequent screwing possibility bue straigt cut or z cut

In: **SG**: Solid Geometry **T**: Frame to cut **DINS**: size of screws default is [2.5 6 0] **TTPP**: screw type 'TT' or 'PP' **zcut**: false or 'L' or 'Z' or 'M' **supp**: add solids as supplement; defaut is false **bsiz**: default is 0.05 **"debug"**: if the string "debug" is present, debug plots are shown

Out: **SGH**: Head side (positive ez) **SGN**: Nut side (positive ez) **T**: Used cutting frame

SGdesignDIN7991

creates a an assemly methods for DIN7991 or DIN965

In: **sdl**: [M-Screw Length overlength and spacer]; default [2.5 6 0 0] **tpl**: expected overleng to achieve a through hole instead of a blind hole ; default is 40 **nut**: 'none' or 'bushing' or 'nut' on foot side; not implemented yet

Out: **H**: Subtraction solid for head side including frame C to align on head side **N**: Subtraction solid for nut side including frame C to align on foot side **S**: Geometric Model of a inbus screw DIN 7991/965 with frame C to align on head side **mt**: remaining wall thickness **parts**: parts list for assembly of one connection, just a screw

SGdesignDIN7991BushingE

creates subtraction solids for a screw fixation at a specific frame position of a solid

In: **mlb**: [M-Screw Length overlength and spacer]; default [2.5 6 0 0] **ins**: Characters for "T"unnel or "P"ocket; default is 'TT' for [HEAD NUT] **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10

Out: **Hsub**: Subtraction Solid for Head side including frame 'C' **Nsub**: Subtraction Solid for Bushing side including frame 'C' **SS**: Srew and Nut for Visualziation including frame 'C' **Hadd**: remaining wall thickness **Nadd**: parts list for assembly of one connection **AParts**: parts list for assembly of one connection (DIN7991 screws and Kerbkonus BancLok-E bushing)

SGdesignDIN912BushingE

creates subtraction solids for a screw fixation at a specific frame position of a solid

In: **sdl**: [M-Screw Length overlength and spacer]; default [2.5 6 0 0] **ins**: Characters for "T"unnel or "P"ocket; default is 'TT' for [HEAD NUT] **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10 **W**: Rotation angle for Pocket if required; default is 0;

Out: **H**: Subtraction Solid for Head side including frame 'C' **N**: Subtraction Solid for Bushing side including frame 'C' **S**: Srew and Nut for Visualziation including frame 'C' **mt**: remaining wall thickness **parts**: parts list for assembly of one connection

SGdesignDIN912DIN985

creates subtraction solids for a screw fixation at a specific frame position of a solid

In: **sdl**: [M L d] = Metrix Length and delta in nut for screw connection **ins**: Characters for "T"unnel or "P"ocket; default is 'TP' for [HEAD NUT] **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10 **W**: Rotation angle for Pocket if required; default is 0;

Out: **H**: Subtraction Solid for Head side including frame 'C' **N**: Subtraction Solid for Nut side including frame 'C' **S**: Srew and Nut for Visualziation including frame 'C' **mt**: remainig wall thickness **parts**: parts list for assembly of one connection

SGdesignDIN912DIN985axialbearing

creates subtraction solids for a screw fixation and a axial bearing at a specific frame position of a solid

In: **sdl**: [M L d] = Metrix Length and delta in nut for screw connection **ins**: Characters for "T"unnel or "P"ocket; default is 'TP' for [HEAD NUT] **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10 **W**: Rotation angle for Pocket if required; default is 0; **abb**: size of axial bearing; default is [3 6 3.5 3.2 5.8]

Out: **H**: Subtraction Solid for Head side including frame 'C' **N**: Subtraction Solid for Nut side including frame 'C' **S**: Srew and Nut for Visualziation including frame 'C' **mt**: final wall thickness between head and bearing **parts**: parts list for assembly of one connection

SGdesignDIN912DIN985axradbearing

creates subtraction solids for a screw fixation and a axial bearing at a specific frame position of a solid

In: **sdl**: [M L d] = Metrix Length and delta in nut for screw connection **ins**: Characters for "T"unnel or "P"ocket; default is 'TP' for [HEAD NUT] **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10 **W**: Rotation angle for Pocket if required; default is 0; **abb**: size of axial bearing; default is [3 6 3.5 3.2 5.8] **rbb**: size of radial bearing; default is [2.5 6 1.8]

Out: **H**: Subtraction Solid for Head side including frame 'C' **N**: Subtraction Solid for Nut side including frame 'C' **S**: Srew and Nut for Visualziation including frame 'C' **mt**: final wall thickness between head and bearing **parts**: parts list for assembly of one connection

SGdesignDIN912DIN985radialbearing

creates subtraction solids for a bearing connection at a specific frame position of a solid

In: **bbb**: [Ri Ro Width] of the ball bearing **sdl**: [length overize] length and oversize of the screw DIN912 **ins**: "TT", "TP, "PT", "PP"; Default is "TT" **tpl**: Length of Insertion T/P for [HEAD NUT]; default is 10 10 **W**: Rotation angle for the depth of the pocket

Out: **H**: Subtraction Solid for plug side including frame 'C' **N**: Subtraction Solid for bearing side including frame 'C' **S**: Geometry of Plug including frame 'C' **mt**: remaining material width **parts**: parts list for assembly of one connection

SGdesignDrivenshaft

Designs a torque transmitting shaft

In: **M**: Diameter of [Cylinder Torx-Top Torx Bottom] **L**: Length of [Middle Top Bottom] **sI**: fitting for subtraction solids **KK**: Kerbkonus [Top bottom] **"debug"**: if the string "debug" is present, debug plots are shown

Out: **H**: Subtraction solid of Top Torx ('F') **N**: Subtraction solid of Bottom Torx ('B') **S**: Geometry of the Torque pin

SGdesignDriveSpindle

returns the solid geometry of a gripper spindle

In: **d**: spindle parameters [length diameter pitch]; default is [30 0 0]; if diameter is 0 it becomes 6, if pitch is 0 it is set to 2*b(2) from DIN13 **SGR**: optional solid geometry attached to the cage; if 'tcp' a robot TCP plate is generated; default is empty

Out: **SG**: cell array of solid geometries {jaw 1, jaw 2, spindle, cage, cap, gear}

SGdesignfaceconnectingrod

creates a connecting rod between two faces of a solid

In: **SG**: Solid Geometry **f**: [face #1 face #2] **elo**: elongation into SG; default is 1 **Rv**: Radius for curves; default is 2.5mm **CPL**: Cross sectton contour; default is PLcircle(Rv)

Out: **SGP**: Solid of the rod **f**: facets in case of interactive mode **SGN**: Solid with additional rod (SGunion)

SGdesignGripperjaw

returns the geometry of gripper jaws, gripper finger and complete mechanisms for an object to grasp

In: **SG**: Object to be gripped **sz**: SG of maximum jaw or maximum dimensions of a cuboid **sI**: slot size; default is **fp**: Finger end solid; or position of a cuboid **"sub"**: if used, the gripper jaw is cut by the SG **"fsub"**: if used, the gripper jaw is attached optimally to the surface orthognal vector **"surf"**:

Out: **SGB**: {SG1, SG2} Finger geometry with Frames 'B' to attach actuators **T0**: Center fram of movement in ez **TB**: Attachment Frames at object SG

SGdesignIgusdragchain

creates the subtraction and supplemet solids for fixing a igus drag chain

In: **M**: [2 5 2 7],true); % M2 DIN 7991, Distance 5, Pin 2 length 7)

Out: **Nsub**: Subtraction solid (green) **Nadd**: Supplement solid (yellow) **S**: Screw (magenta) **AParts**: Parts list

SGdesignMotorN20

creates subtraction solids for the design of motor driven mechanisms

In: **LM**: [LengthofMotor gearbox/Supplement wall]; default is [26 9 2.0] **"cutx"**: **"cuty"**:

Out: **H**: Head / Shaft subtraction solid (Red) **N**: Head / Mtor subtraction solid (Green) **S**: Shape of Motor N20 (Magenta) **Supp**: Supplement solid (Yellow) **Parts**: required parts for assembly

SGdesignMotorN40horiz

Designs a subtraction solids for the N20 L-type encoder motor

In: **LM**: [housinh-height wall-thickness topwall-thickness rotation-angle] default is [14 1.5 2 0] **sh**: Shaft Diameter and Shaft Flattening; default is [3 2.5] **sz**: [maxsize-x maxsize-y length-shaft diameter shaft-socket, height shaft-socket]; default is [10 12 6.5 4 .5] **gb**: gearbox [size-x size-y height motor-length Mscrews xshift yshift]; default is [18 12 10 14 1.6 5 0] **PL**: Point list of screws; default is [-2.5 -4.5;-2.5 +4.5]

Out: **Hsub**: Head Subtraction Solid (Shaft Side) **Nsub**: Nut Subtraction Solid (Motor Side) **SGM**: Motor Geometry **Hadd**: Head Supplement (Shaft Side) **Nadd**: Nut Supplement (Motor Side) **AParts**: Part list

SGdesignPeginHole

design subtraction solid and supplement solid for peg in hole connections using fittings

In: **CPL**: Desired contour; **h**: [lp lh]; length peg; length hole; hole per default peg plus slfit('c') **sl**: fitting; per default 'c'

Out: **PA**: Supplement solid for peg using frame 'C' and 'alignT' **HS**: Subtraction solid for hole using frame 'C' and 'alignT'

SGdesignPintle

Designs a simple pindle bolt connection

In: **MM**: default is [3 6 0 0]; [M-Bolt Length overlength and spacer]; default [2.5 6 0 0] **sl**: default is 'tt' for transition and clearance fittion on head side and nut side

Out: **Hsub**: Subtraction solid on head side **Nsub**: Subtraction solid on nut side **SGB**: SG of Bolt USED TO BE PRINTED OUT! **Hadd**: Supplement solid on head side **Nadd**: Supplement solid on nut side **AParts**: Parts list

SGdesignPluginFrame

adds a plug-in sleeve to a solid

In: **SG**: Solid Geometry **T**: HT Frame or ez vector (roughly) or Frame name **H**: height of the sleeve **sl**: fitting; default is 't' for transition **styl**: straight or bezier1 or bezier2; default is bezier1

Out: **SG**: Solid plus plug-in sleeve **SGP**: jsut the plug-in sleeve

SGdesignSupplement

creates cylinders or CPLs that are big enough to encapsulate the fixture elements

In: **S**: Screw or Fiaxtion type using a Frame 'C' as separation **F**: Frame Name default is 'C' **w**: minimum wall size **align**:

Out: **SGHS**: Solid for Head SGdesignSupplement, green **SGNS**: Solid for Nut SGdesignSupplement, yellow **r**:

SGdesignSupplement2connect2SG

first appraoch to create bridges between two solids

In: **SGA**: Solid A **SGB**: Solid B **wt**: connector width; default is automatic mode '' **ra**: **"debug"**:

Out: **SGA**: Solid A with optional Bridging solid **SGB**: Solid B with optional Bridging solid **SGC**: Connecting solid **CPLC**:

SGdesignSupplement2connectSG

fills in a solid block to connect the surfaces of SG

In: **SG**: Solid Geometry **T**: Frame or z-axis used for 2.5 projection solid **wt**: wall thickness for fillgap **ra**: radius for radial edges **zz**: height ; default is [zmin zmax]

Out: **SGN**: Final Solid **SG2**: Just filling block solid

SGdesignSupplementPlugin

designs a plugin holder for another solid

In: **SGA**: Solid Geometry **wt**: [wall thickness height modification]; default is [1.5 0] **sl**: slot size; default is 'c' fit or a number **ez**: default is [0 0 1]; not used **CPL**: CPL out contour; default is same as contour of SGA **fill**: if true; SGB is a block; default is false; **"top"**:

Out: **SGB**: Supplement solid to plug in SGA

SGdesignTongueGrooveRadial

creates a radial tongue and groove connection for a contour or an opening in a 2.5D structure

In: **CPLu**: Contour of an opening in a solid (CPLofFS) | OR a SG solid **T**: Transformation of CPLu (CPLofFS); default is eye(4) | Or an ez vector **h**: [h-inside h-outside]; default os [3 5]; **wi**: [wall-inside wall-outside] default is [0.8 0] **ft**: **"STL"**:

Out: **Hsub**: Subtraction solid **Nsub**: not used yet **S**: not used yet **Hadd**: not used yet **Nadd**: Lower Contour **T**: Coordinate frame that has been used if ez was input parameter

SGtoolTip

returns a Subtraction Solid for a "tool" for SGbool

In: **d1**: outer diameter **d2**: tip diameter **w**: optional angle; default is 90 degree

Out: **SG**:

7.6.4 Fits (interference / clearance) [guide]

Functions in this section:

islfit — returns a probable fitting for an expected 3D Printer from two dimenions

SGfitcalibration4Printer — Create solids to measure the friction force of printed for the fitting concept

SGfittinggauge — creates a solid for testing fits on a 3D printer

SGwarnfittinguse — devolper fnct to be used in a SG fnct if fittings are used within the fnct.

slfit — returns the slot size value for a desired fitting during programmed design procedures

Detailed description

islfit

returns a probable fitting for an expected 3D Printer from two dimenions

In: **dm**: Measured Dimension DIAMETER of a bore **de**: Expected Dimension DIAMETER of a bore **diam**: optional specific printer name if not default one **AP**: Name of Printer for fitting test

Out: **ss**: single char abbreviation **AP**: full name of printer **s**: full fitting description

SGfitcalibration4Printer

Create solids to measure the friction force of printed for the fitting concept

In: **w**: default wall thickness is 1.2 mm

SGfittinggauge

creates a solid for testing fits on a 3D printer

In: **D**: list of diameters to test; default is 5:.1:5.5

Out: **SG2**: Solid **SG1**: cell array of cylindric fitting gauge solids (each with size label)

SGwarnfittinguse

devolper fnct to be used in a SG fnct if fittings are used within the fnct.

slfit

returns the slot size value for a desired fitting during programmed design procedures

In: **typ**: clearance type such as 'c', 't', 'i' **name**: name fo a printer such as FORM2 **"STL"**: if true; the printer is not set

Out: **sl**: slot size ==> deivide by 2 for radius **Printer**: exact printer name

7.6.5 Dimensioning [guide]

Functions in this section:

CVLdimclassifier — analyses contours and classifies different drawing elements

PLdimensioning — plots a measurement ruler under a CPL

SGdimensioning — creates a drawing of the solid including dimensions

SGdimensioning4 — Creates a simple 4 quadrant dimensioning view for a solid Geometry

SGdimensioningFront — Creates a Front View Dimensioning Figure

SGdimensioningSide — Creates a Side View Dimensioning Figure

SGdimensioningTop — Creates a Top View Dimensioning Figure

SGprofiledepthdimension — interactive fnct that returns the distance between surfaces

SGprofilesurfacedimension — creates a drawing of the touched surface

SGTdimensioning — creates a four-quadrant view related to a frame or the center of a solid geometry

TofCVLdimclassifier — returns Frames for radii in 3D-Contours as CVL

viewdimensioning — changes the view parameter for an engineering view

Detailed description

CVLdimclassifier

analyses contours and classifies different drawing elements

In: **OVL**: Contour vertex List **res**: resolution for clustering radial centers; default is 0.1

Out: **DVL**: List for dimensioning **RL**: [cx cy cz Nr deg R vx vy vz] **RIL**: Points **Rnv**: List of normal vectors related to RL

PLdimensioning

plots a measurement ruler under a CPL

In: **CPL**: CPL or PL **acc**: dimensioning accuracy (rounding resolution); default is 0.1 **unt**: unit string appended to dimensions; default is '' (empty) **fnts**: font size for dimension text; default is 16

Out: **h**: handle to drawings

SGdimensioning

creates a drawing of the solid including dimensions

In: **SG**: Solid geometry **ang1**: angle 1 similar to view **ang2**: angle 2 similar to view **pnt**: crossing point **acc**: dimensioning accuracy; default is 0.1

Out: **CPL**: closed polygon list of the computed dimension contour

SGdimensioning4

Creates a simple 4 quadrant dimensioning view for a solid Geometry

In: **SG**: Solid geometry **pt**: crossing point **acc**: accuracy for dimensioning; default is 0.1

Out: **hf**: heandle to figure

SGdimensioningFront

Creates a Front View Dimensioning Figure

In: **SG**: Solid geometry **pnt**: crossing point **acc**: accuracy; default is 0.1

Out: **CPL**: CPL of Side-view

SGdimensioningSide

Creates a Side View Dimensioning Figure

In: **SG**: Solid geometry **pnt**: crossing point **acc**: accuracy; default is 0.1

Out: **CPL**: CPL of Side-view

SGdimensioningTop

Creates a Top View Dimensioning Figure

In: **SG**: Solid geometry **pnt**: crossing point **acc**: accuracy; default is 0.1

Out: **CPL**: CPL of Topview

SGprofiledepthdimension

interactive fnct that returns the distance between surfaces

In: **SG**: Solid Geometry **fi**: facet index **sw**: single wall; if true only the touched surface is returned

Out: **dl**: list of wall thickness along the line **fi**: touched facet index

SGprofilesurfacedimension

creates a drawing of the touched surface

In: **SG**: Solid Geometry **fi**: optional facet index **sw**: Definde contour levels

Out: **RL**: Point list **fi**: Radii list **PL**: facet index **ez**: surface normal direction ez at the selected facet

SGTdimensioning

creates a four-quadrant view related to a frame or the center of a solid geometry

In: **SG**: Solid Geometry **Fr**: Frame Name; default is 'B' **pnt**: center of view related to Frame

Out: **CPL1**: CPL of XY-Plane: subplot(2,2,4) **CPL2**: CPL of XZ-Plane: subplot(2,2,2) **CPL3**: CPL of YZ-Plane: subplot(2,2,1)

TofCVLdimclassifier

returns Frames for radii in 3D-Contours as CVL

In: **CVL**: 3D-Contours **minmaxR**: mimimal and maximal radius to consider default is [1.25 10] **minmaxW**: minimal & maxium circum angle to consider; default is [40 360] **ez**: force ez vector

Out: **TR**: cell list of frames for the classified radial contours

viewdimensioning

changes the view parameter for an engineering view

In: **a**: view angle a **b**: view angle b **crossp**: optional crossing point / plane **alpha**: alpha value

Out: **CPL**: Contour Point list x y 0 **CVL**: Contour vertex list x y z **X**: Crossing Solid

7.6.6 Completed CAD Designs [guide]

Functions in this section:

AIM_Robot — returns a solid geometry for a robot used in AIM lectures

FTclutch — returns a clutch for Fischertechnik Gear Motor

SGBAUHAUSbox — designs a BAUHAUS style boxes to hide cables and stuff

SGBAUHAUSchairscratch — returns the SG for a scrat protector of a specific Bauhaus style chair

SGBAUHAUScupholdinsert — returns a SG as insert for cup holder in a car

SGBAUHAUSdeskcylinder — designs a BAUHAUS style boxes to hide cables and stuff

SGBAUHAUSflorence — returns a solid geoometry for an Florence Knoll Basset chair or sofa

SGBAUHAUSnotesbox — returns a simple box with cover lid for a note paper block

SGBAUHAUSpenholdertilted — constructs a tilted pen holder

SGBAUHAUStectaF51 — returns models of the BAUHAUS Walter Gropius Chair produced by TECTA in Germany

SGcatheterGauge — prints a gauge for measuring the catheter diameters in mm using a 3D printer

SGfurniture — Solid geoemtries of sample furniture

SGfurniturewedges — returns furniture wedges for assemble of shelves etc/ with an height imprint

SGhook4rail — returns a solid for a rail hook looking to face

SGhotrodCarbody — returns different SG of the Hotrot projects

SGjuicerAdapter — creates a spare part for a BOSCH juicer

SGmakerworld — Simply opens the WWW-page of grabcad.com in the system browser

SGmodelOR — returns solid model of OR device models

SGprinterstand — creates posts or stands for furniture or in this case hp laser printer

SGservoshaft — separates Solid Geoemetry of Servo and Servoshaft

SGsocketcover — returns an image relief on top of a socket cover

SGstentPeterEwert01 — Solid Geometry of a Stent for Peter Ewert

SGstentVeitPhillipMohamadAbdelhafez — SG of a silikon stent using the deisng rules of Veit Phillip and Mohamed Abdelhafez

SGvenacavainferiorring — creates a ring for the Vena Carva Inferior

SGveribor120 — returns the shape of a 120 mm Veribor

SGveribor120yannick — returns a plate to connect 2 Veribor 120 by M8 SCrew and to fix the LCL base with 4 times M5

SGveribor210 — returns the Geometry of a Veribor210 of Year 2020

Detailed description

AIM_Robot

returns a solid geometry for a robot used in AIM lectures

Out: **SGrobot**: Solid Geometry of a printable Robot for AIM

FTclutch

returns a clutch for Fischertechnik Gear Motor

Out: **O**: Solid structure containing vertex list and facet list

SGBAUHAUSbox

designs a BAUHAUS style boxes to hide cables and stuff

In: **siz**: [x y z r w l]; default is [160 25 20 2] **b**: width of opening; default is z/2 **cutw**: angles for openings; default is [0, pi] **both**: if true cover and box are open, default is false **f**: [phase width height]; defautl is [size(5) siz(5)] **"STL"**: if given as 'STL', STL files of box and cover are written **"square"**: if given as 'square', the outer size is enlarged for a square shape

Out: **box**: SG of box **cov**: SG of cover

SGBAUHAUSchairscratch

returns the SG for a scrat protector of a specific Bauhaus style chair

SGBAUHAUScupholdinsert

returns a SG as insert for cup holder in a car

In: **h**: [height angle width and length; default is [60 17 50 75]

SGBAUHAUSdeskcylinder

designs a BAUHAUS style boxes to hide cables and stuff

In: **siz**: [diameter height height wall slot]; default is [160 30 2 2 0] **num**: number of edges; default is []; **cutw**: angles for openings; default is [0, -pi/5, +pi/5] **"STL"**: 'STL' flag to also write an STL file

Out: **box**: SG of box **cov**: SG of cover

SGBAUHAUSflorence

returns a solid geoometry for an Florence Knoll Basset chair or sofa

In: **B**: Seat width in cm - THE sofa is 15 cm wider

Out: **SGC**: Solid Geometry of chair

SGBAUHAUSnotesbox

returns a simple box with cover lid for a note paper block

In: **isSTL**: if used, an STL directory is written for a slicer **"STL"**: if used, an STL directory is written for a slicer

SGBAUHAUSpenholdertilted

constructs a tilted pen holder

In: **SH**: Pen length and holder height; default is [150 0]; 0 means automatic calculation **MX**: Maximal height of the upper side of the pen holder; default is 100 **Dw**: Diameter of a circular pen holder

Out: **SGall**: Solids to print

SGBAUHAUStectaF51

returns models of the BAUHAUS Walter Gropius Chair produced by TECTA in Germany

In: **BN**: body number 1/2/3; 0 creates a full set **"STL"**: if used, an STL file is written

Out: **SGD**: Solid Geometry of a single chair / sofa

SGcatheterGauge

prints a gauge for measuring the catheter diameters in mm using a 3D printer

In: **ddlst**: list of slot diameters default is [0:0.1:3] **prtnam**: printer name for calibrating the design parameter DIN4AMfitting **slotc**: default is transition fit 't' (see DIN4AMfitting) **gname**: name on the Gauge

Out: **SGL**: Solid Geometry of the gauge

SGfurniture

Solid geoemtries of sample furniture

In: **typ**: box, couch, locker, table, chair, haslev, carpet, balkon **a**: turning angle in deg **pos**: shift vector in [x y z] **xyz**: dimensions as x y **par**: parameter beside dimensions

Out: **SG**: Solid geometry

SGfurniturewedges

returns furniture wedges for assemble of shelves etc/ with an height imprint

In: **H**: height of the wedges; default is 5 **L**: Length and width of the wedges; default is [50 30] **"STL"**: if used an STL file is written

Out: **SG**: Solid geometry

SGhook4rail

returns a solid for a rail hook looking to face

In: **DR**: Diameter of Rail; default is 30mm **WS**: Wall size; default is DR/3 **DL**: Elongation for hook; default is DR/2 **NM**: optional name text engraved on the hook; default is copyright text with date

Out: **SG**: Hook for a Rail

SGhotrodCarbody

returns different SG of the Hotrot projects

In: **name**: Abbreviation of SG such as Ehot+, EHot **nF**: number of faces; default is 60000

Out: **SG**: Solid Geometry **mtab**: vector of axis dimension

SGjuicerAdapter

creates a spare part for a BOSCH juicer

In: **Do**: Outer Diameter **Di**: Inner Diameter **h**: height of the adapter; default is 40

Out: **SG**: solid geometry of the juicer adapter (concentric inbus contours extruded to height h)

SGmakerworld

Simply opens the WWW-page of grabcad.com in the system browser

SGmodelOR

returns solid model of OR device models

In: **part**: Part identifier (name or index) to build

Out: **SG**: Solid geometry of part

SGprinterstand

creates posts or stands for furniture or in this case hp laser printer

In: **HW**: [X Y Z wall]; default is [90 90 90 10] **"STL"**: if used an STL file is written

Out: **SG**: Single SG of a stand/post

SGservoshaft

separates Solid Geoemetry of Servo and Servoshaft

In: **SG**: Solid Geometry **h**: height of shaft; default is 2.9 mm **T**: HT Matrix or Name for from' default is 'Shaft'

Out: **SGB**: Solid Geometry of Shaft **SGA**: Solid Geometry of Servo without shaft **SGC**: thin cylindrical solid marking the shaft axis

SGsocketcover

returns an image relief on top of a socket cover

In: **I**: Image that should become the relief

Out: **SGall**: Socket cover to print

SGstentPeterEwert01

Solid Geometry of a Stent for Peter Ewert

In: **mag**: Stretch/magnification ratio of the stent

Out: **SG**: Resulting stent solid geometry **CPLN**: Resulting stent contour

SGstentVeitPhillipMohamadAbdelhafez

SG of a silikon stent using the deisng rules of Veit Phillip and Mohamed Abdelhafez

In: **L**: Length (full length, upper opening, lower opening] **D**: Diameter (Stent, uper opening, lower opening] **w**: wall thicknes [wall upper ring radius, lower ring radius] **halfring**: false=fullring true=halfring; default is true;

Out: **SG**: Solid Geometry

SGvenacavainferiorring

creates a ring for the Vena Carva Inferior

In: **DA**: Diameter of the Carva Inferior vein **dmin**: minimal diameter of the ring **dmax**: maximal diameter of the ring

Out: **SGr**: Solid of Ring to placed through catheter **SGvci**: Solid Geometry of Carva Inferior vein

SGveribor120

returns the shape of a 120 mm Veribor

Out: **SG**: Solid Geometry of a Veribor 120

SGveribor120yannick

returns a plate to connect 2 Veribor 120 by M8 SCrew and to fix the LCL base with 4 times M5

Out: **C**: Solid Geometry of plate (Magenta) **H1**: solid geometry of the M8 mounting bushing

SGveribor210

returns the Geometry of a Veribor210 of Year 2020

Out: **SG**: Solid Geometry of Veribor210 **HH**: solid geometry of the mounting bushing/hole insert

7.7 Robotics, Hardware and Sensors [guide]

7.7.1 Serial Robots (JACO, Raambo, FeeTech) [guide]

Functions in this section:

CPLraamboTCPgearnut — returns the CPL of the Raambo-Robot TCP gear wheel

jaco05 — returns all the parts fo the jaco robot in specified geometry

JACOget — returns information on the JACO robot by direct calls

JACOmodel — creates an STL file for magnified/shrinked jaco link

JACOset — sets parameter of the real JACO robot and of the global JACO model

JACOsim — Simulates the behavior of the JACO robot

JACOstruct — creates a global struct JACOs that is used by JACOsim, JACOget, JACOset

raambocommandlines — implements a command line based communication to a Raambo®-Robot via serial link (USB)

raamboGET — Same as raamboSET but without setting values

raamboscreen — implements a character based communication to a Raambo®-Robot via serial link (USB)

raamboSET — sends robot cmommands throught the global RAPORT to a Raambo-Robot

raambostartup — sets important parameter for experiments with the Raambo Robot

SGjacohand — returns the JACO hand with fingers

SGmotorFeeTech40flange — robot servo motor flange

SGraamboTCPbagcutter — returns a solid that can cut a sterile bag in combination mit the knurled gear cover of Raambo robot

SGraamboTCPcreatefulltoolset — this is a fnct the creates one after another all required parts for RaamboTCP tools

SGraamboTCPgearwheeldisk — returns the gearwheeldisk of the Raambo robot as designed by Erik Löwe

SGraamboTCPgearwheeldiskTL — returns the gearwheeldisk of the Raambo robot as designed by Tim Lueth

SGraamboTCPknurledgearcover — returns a knurled tube TCP cover for Raambo robot

SGraamboTCPknurledgearcoverTL — returns a torque limited knurled tube TCP cover for Raambo

SGraamboTCPMagnetM22 — creates gear wheel for the 6th axis of Raambo

SGraamboTCPmagnetpull — returns a magnet releasing tool for the Raambo robot

SGraamboTCProtatingrodM16 — returns a simple rod with magnet coupling and an M16 internal thread

SGraamboTCPstraightmoveM16 — returns a push-pull TCP for M16x10x5 threads

SGraamboTCPtoolM16extension — creates a tool extension for different geometries with a M16x10x5 Screw and Nut

SGraamboTCPtoolM16ofCPL — returns a tool holder for an M16x10 5mm pitch for the Raambo robot

SGraamboTCPtoolstand — returns a tool stand for a designers desktop

Detailed description

CPLraamboTCPgearnut

returns the CPL of the Raambo-Robot TCP gear wheel

In: **"STL"**:

Out: **CPLO**: Outer contour designed by EL **CPLi**: Inner contour as DIN13 using slfit('t') **CPLo**: outer contour as DIN13 using slfit('t') and D=24.8

jaco05

returns all the parts fo the jaco robot in specified geometry

Out: **SGn**: Solid Geometry containing all parts and the box for printing

JACOget

returns information on the JACO robot by direct calls

In: **CMD**: Command string **PVAL**: Numeric Parameter Values

Out: **ANSW**: Answer **t**: time delay between call and answer

JACOmodel

creates an STL file for magnified/shrinked jaco link

In: **gl**: robot link number **siz**: shinking/magnifiaction size; default is 0.25 **tck**: thickness addon; default is 1.0 **tol**: tolerance; default is 0.3

Out: **SGS**: Final Solid including frames

JACOset

sets parameter of the real JACO robot and of the global JACO model

In: **CMD**: Command string **PVAL**: Parameter Values als cell or numeric value

Out: **ANSW**: Answer/response from the robot **t**: Timestamp

JACOsim

Simulates the behavior of the JACO robot

In: **CMD**: command string **PVAL**: Parameter Values als cell or numeric value

Out: **ANSW**: Answer/response from the robot **t**: Timestamp

JACOstruct

creates a global struct JACOs that is used by JACOsim, JACOget, JACOset

raambocommandlines

implements a command line based communication to a Raambo®-Robot via serial link (USB)

In: **PORT**: already opened PORT such as COM1 or /dev/tty.usb11001

raamboGET

Same as raamboSET but without setting values

In: **cmd**: command list same as raamboSET

Out: **x**: result devlivered by the raambo firmware

raamboscreen

implements a character based communication to a Raambo®-Robot via serial link (USB)

In: **PORT**: already opened PORT such as COM1 or /dev/tty.usb11001

raamboSET

sends robot cmommands throught the global RAPORT to a Raambo-Robot

In: **cmd**: cmd such as "?", or speed, goal, pos **val**: value for all or one motor **mot**: selection of motors to be addressed

Out: **x**: result devlivered by the raambo firmware

raambostartup

sets important parameter for experiments with the Raambo Robot

In: **pose**: 'straight', 'sleep', array of encoder goal positions

SGjacohand

returns the JACO hand with fingers

Out: **SGn**: CEll list of hand plus 3 fingers

SGmotorFeeTech40flange

robot servo motor flange

In: **hg**: height of gear; default is 5 **Dmag**: diameter and height of countersunk magnet [D h]; default is [15 3] **"STL"**: if set, writes the parts as STL files

SGraamboTCPbagcutter

returns a solid that can cut a sterile bag in combination mit the knurled gear cover of Raambo robot

In: **"STL"**:

Out: **SGH**: dumb plug **SGG**: plug with small straight and then tampered open tip **SGC**: plug with tampered open tip

SGraamboTCPcreatefulltoolset

this is a fnct the creates one after another all required parts for RaamboTCP tools

SGraamboTCPgearwheeldisk

returns the gearwheeldisk of the Raambo robot as designed by Erik Löwe

In: **"STL"**:

Out: **SGP**: SG of gear wheel disk

SGraamboTCPgearwheeldiskTL

returns the gearwheeldisk of the Raambo robot as designed by Tim Lueth

In: **DHM**: [Diameter Height] of Magnet; default is [15 4] **"STL"**: if used an STL folder is written in the desktopdir

Out: **SGP**: SG of gear wheel disk for the FeeTech 60 motor **SGScr**: SG, height 15mm, comparable to the height of SGraamboTCPMagnetM22TL or SGraamboTCPMagnetM22 **SG2**: Tool height 15mm, to move the wheel inside a 15mm knureld gearwheel ring

SGraamboTCPknurledgearcover

returns a knurled tube TCP cover for Raambo robot

In: **LK**: length/dimensions of the knurled tube [min max]; scalar is expanded to [0 LK]; default is 50 **"STL"**: if set, writes the solids as STL files

Out: **SGall**: named solid set of the knurled-gear-cover parts (tube, cut cap piece, hand-wheel)

SGraamboTCPknurledgearcoverTL

returns a torque limited knurled tube TCP cover for Raambo

In: **LK**: STL: **"STL"**: 'STL' flag to also write an STL file

Out: **SGall**: All resulting solids

SGraamboTCPMagnetM22

creates gear wheel for the 6th axis of Raambo

In: **CPLu**: Contour of an inlay **hm**: height of the inlay; default is 5 **"STL"**: if used an STL File of both parts are written

Out: **SGNut**: Gear-wheel-cap for the TCP motor of Raambo holding a magnet **SGScr**: Gear-wheel-cap screw for the TCP motor of Raambo M22

SGraamboTCPmagnetpull

returns a magnet releasing tool for the Raambo robot

In: **"STL"**:

Out: **SGall**: named solid set of all magnet-pull tool solids for export

SGraamboTCProtatingrodM16

returns a simple rod with magnet coupling and an M16 internal thread

In: **LL**: Length; default standard is 50 **"STL"**: if used an STL folder will be written

Out: **SGall**: Cell arrary with Rod and Magnet screw nut

SGraamboTCPstraightmoveM16

returns a push-pull TCP for M16x10x5 threads

In: **"STL"**:

Out: **SGall**: collection of a required components for Raambo

SGraamboTCPtoolM16extension

creates a tool extension for different geometries with a M16x10x5 Screw and Nut

In: **PL**: Point list; default is PL=[0 -1; 0 15; 30 15] ;

Out: **SGN**: Solid with Screw and Nut

SGraamboTCPtoolM16ofCPL

returns a tool holder for an M16x10 5mm pitch for the Raambo robot

In: **CPLP**: Cross section of object to hold **LW**: if used an STL folder is written **"diamond"**: if set, uses a diamond-shaped contour around CPLP instead of the plain contour **"STL"**: if set, writes the complete probe guide as an STL file

Out: **SGPG**: Solid of probe guide with M16x10 pitch 5mm

SGraamboTCPtoolstand

returns a tool stand for a designers desktop

In: **"STL"**:

Out: **SGall**: Final solid for the desktop

7.7.2 FeeTech Motor Control [guide]

Functions in this section:

FTSCScmd — returns the command sequence for the FeeTech SCS RSC 485 Protocol of the URT-1 Board

FTSCScmdfindserial — Searches for an URT-1 Board, tries different baud rates and searches for connected motors

FTSCSdata — converts read data from the URT-1 board into the data format

FTSCSmotorlist — Searches for Motor ID connected to the FeeTEch URT-1 Board after opening the serial port

Detailed description

FTSCScmd

returns the command sequence for the FeeTech SCS RSC 485 Protocol of the URT-1 Board

In: **servoID**: ServoID; default is 1 **MemAddr**: Memory Address; default is 42 **data**: uint16 data to write; default is empty **Fun**: default is 3 for write **getd**: number of uint16 to read

Out: **cmd**: uint8 data array to send using write **str**: Hex string for documentation

FTSCScmdfindserial

Searches for an URT-1 Board, tries different baud rates and searches for connected motors

In: **btab**: list of baudrates; default is [1000000, 500000, 250000, 128000, 115200, 76800, 57600, 38400]

Out: **ID**: ID of the connected motors **COM1**: open serial port if motors were found

FTSCSdata

converts read data from the URT-1 board into the data format

In: **datarec**: data received from the URT-1 board by the read function; cell array if there are several answers **typ**: data type such 'unit8' or 'uint16', default is 'uint8'

Out: **data**: data in the type format

FTSCSmotorlist

Searches for Motor ID connected to the FeeTEch URT-1 Board after opening the serial port

In: **COM1**: Already Open Serial Port

Out: **ID**: array of IDs

7.7.3 USB and Serial Interfaces [guide]

Functions in this section:

LaserInit — Initializes the USB-Serial Port and Requests Data from the Leuze Scanner

sbuffercreate — creates a struct buffer for async read/write

sbufferget — reads bytes out of a struct buffer

sbufferinfo — returns information on a struct buffer

sbufferwrite — writes bytes into a struct buffer

SCavailable — returns the list of serial communciation interfaces on OSX

SCtest — tests a serial communication line on apple osx

serialportused — returns the ports that are in use already

SGmicroscopeclamp — delivers a spare part for a typical USB microscope camera in 2020

smlArduinohelp — help to connect an arduino to matlab since 2018b

USBhelp — Help text for the use of USB serial communication on MAC

USBpololu — USB-communication with a Pololu Maestro Controller (Dual-Port Mode)

USBsearch — searches for specific USB serial devices on a MAC

Detailed description

LaserInit

Initializes the USB-Serial Port and Requests Data from the Leuze Scanner

Out: **m**: vector of measured distances (mm), one per sample; nan where out of range **t**: vector of timestamps (s) for each sample

sbuffercreate

creates a struct buffer for async read/write

In: **n**: number of bytes; default is 256

Out: **b**: buffer struct

sbufferget

reads bytes out of a struct buffer

In: **b**: struct buffer **n**: number of bytes to get

Out: **b**: updated buffer **rbytes**: array with received bytes **err**: error number

sbufferinfo

returns information on a struct buffer

In: **b**: struct buffer

Out: **f**: number of free bytes **l**: maximum number of bytes **u**: number of used bytes

sbufferwrite

writes bytes into a struct buffer

In: **b**: struct buffer **mbytes**: bytes array to write

Out: **b**: updated buffer **err**: error number

SCavailable

returns the list of serial communciation interfaces on OSX

SCtest

tests a serial communication line on apple osx

In: **port**: port description; default is '/dev/tty.usbmodem00091661'

Out: **s**: serial port object opened on the given port at 9600 baud

serialportused

returns the ports that are in use already

Out: **su**: serial in use

SGmicroscopeclamp

delivers a spare part for a typical USB microscope camera in 2020

In: **DH**: Difference height; default is 0;

Out: **SG**: Solid Geometry to print **SGM**: Solid of the Ring **SGB**: Solid of the Ball

smlArduinohelp

help to connect an arduino to matlab since 2018b

In: **request**: Help request / topic

USBhelp

Help text for the use of USB serial communication on MAC

USBpololu

USB-communication with a Pololu Maestro Controller (Dual-Port Mode)

In: **cmd**: command string (uppercased), e.g. 'OPEN', 'CLOSE', 'STATUS'

Out: **rcom**: persistent serial communication object of the Pololu controller

USBsearch

searches for specific USB serial devices on a MAC

7.7.4 RealSense Cameras [guide]

Functions in this section:

SR300allimages — returns a depth image 640x480 from the IntelRealSense300 camera

SR300depthimage — returns a depth image 640x480 from the IntelRealSense300 camera

SR300getVL — return a VL of the current RealSense

SR300getVLFL — returns a surface from the surface camera

SR300install — downloads from TUM-MIMED the Intel Realsense +300 driver and installs it on MAC or WIN

SR300start — initializes the Intel RealSense 300 Scanner and creates a global struct IntelRealSense300

SR300stop — initializes the Intel RealSense 300 Scanner and creates a global struct IntelRealSense300

SR300videoimage — returns a video image 1920x1080 from the IntelRealSense300 camera

Detailed description

SR300allimages

returns a depth image 640x480 from the IntelRealSense300 camera

Out: **vimg**: 1920 x 1080 x uint3 - color image **dimg**: 640x 480 x 3 uint8 - depth image **VL**: Vertex List of surface points - [n x 3]

SR300depthimage

returns a depth image 640x480 from the IntelRealSense300 camera

Out: **img**: 480x640x3 uint8 **data**: original data of the camera

SR300getVL

return a VL of the current RealSense

In: **limb**: depth limit for taken z values

Out: **VL**: Vertex List of selected depth **VLA**: Complete Vertex List

SR300getVLFL

returns a surface from the surface camera

In: **zl**: zlimitation; default is 400

Out: **SG**: Struct with vertices VL and Facets FL

SR300install

downloads from TUM-MIMED the Intel Realsense +300 driver and installs it on MAC or WIN

SR300start

initializes the Intel RealSense 300 Scanner and creates a global struct IntelRealSense300

SR300stop

initializes the Intel RealSense 300 Scanner and creates a global struct IntelRealSense300

SR300videoimage

returns a video image 1920x1080 from the IntelRealSense300 camera

Out: **img**: 920x1080x3 uint8 **data**: original data of the camera

7.7.5 NDI Vicra Tracking [guide]

Functions in this section:

VICRA — class library for the NDI Stereo camera

Vplotstats — simply checks whether the NDI camera has been initialized already

Vreset — resets the NDI camera and starts the communication.

Vstray — returns the list (n x 3) of visible stray markers by the NDI camera

VTget — returns immediately the Matrix of a registered Tracker by the NDI camera

VTget2 — gets a immediate relation Matrix between A and B: A_T_B.

VTget2p — gets a precise relation Matrix between A and B: A_T_B.

VTgetp — returns a precise pose (HT-Matrix) of a registered tracker by the NDI camera

Detailed description

VICRA

class library for the NDI Stereo camera

Vplotstats

simply checks whether the NDI camera has been initialized already

Vreset

resets the NDI camera and starts the communication.

Vstray

returns the list (n x 3) of visible stray markers by the NDI camera

Out: **L**: is a list of position vectors. **n**: is the number of positions in the vector (redundant) **tcam**: is the time of measurement **tpc**: timestamp when the data was requested from the camera (in pc-time seconds)

VTget

returns immediately the Matrix of a registered Tracker by the NDI camera

In: **nr**: is the number of the requested Tracker

Out: **T**: is the HT matrix of the requested tracker nr. **rms**: is the exact system time when the matrix was recorded **tcam**: is the RMS of the spheres with respect to the stored geometry **tpc**: timestamp when the data was requested from the camera (in pc-time seconds)

VTget2

gets a immediate relation Matrix between A and B: A_T_B.

In: **nr1**: the reference tracker number **nr2**: the tracker of interest number

Out: **T**: resulting relative HT-Matrix A_T_B **tcam**: timestamp when the transformation data was captured by the camera (in camera-time seconds) **tpc**: timestamp when the data was requested from the camera (in pc-time seconds)

VTget2p

gets a precise relation Matrix between A and B: A_T_B.

In: **nr1**: the reference tracker number **nr2**: the tracker of interest number

Out: **T**: resulting relative HT-Matrix A_T_B **tcam**: some distance norm in millimeter **tpc**: timestamp when the data was requested from the camera (in pc-time seconds)

VTgetp

returns a precise pose (HT-Matrix) of a registered tracker by the NDI camera

In: **nr**: the tracker of interest's number

Out: **T**: HT-Matrix of the requested Tracker **rms**: some distance norm **tcam**: timestamp when the transformation data was captured by the camera (in camera-time seconds) **tpc**: timestamp when the data was requested from the camera (in pc-time seconds)

7.8 Design Automation Concepts [guide]

7.8.1 Script Language [guide]

Functions in this section:

SGcmd — Shortcut for SGofCPLcommand but supports also sprintftext format

SGofCPLcommand — Simple commandline based 2.5 solid modeler for the use of a caliper

Detailed description

SGcmd

Shortcut for SGofCPLcommand but supports also sprintftext format

In: **par**: see SGofCPLcommand

Out: **SG**: see SGofCPLcommand **CPLO**: see SGofCPLcommand **warn**: see SGofCPLcommand **ocstr**: see SGofCPLcommand **macrol**: see SGofCPLcommand **STAT**: SGofCPLcommand

SGofCPLcommand

Simple commandline based 2.5 solid modeler for the use of a caliper

In: **cstr**: CPL command string, use '?' for info **blines**: number of lines for export; default is 195 **MCRload**: Thickness / height **SGload**: Gap parameter 1 **CPLload**: Gap parameter 2 **STKload**: Gap parameter 3

Out: **SG**: solid Geometry **CPLO**: Resulting contour **warn**: Warning messages **ocstr**: cleaned original command string **macrol**: list of defined macros **STAT**: status struct with fields SG, CPL, MACRO and stack

7.8.2 Jarvis Concept [guide]

Functions in this section:

fourbardesignbysiri — generates 4bar-linkages from a dictation fnct of Siri on MAC

jarvis — Natural Language Interface JARVIS

jarvisChatSGLib — this fnct/script is a ChatGPT based control of the SGLib and defines the development start of SGLib 5.5

jarvisdictionary — reads writes a excel sheet containing string to replace and code to execute

jarvismemory — sets or gets a pose from jarvis / fourbardesignbysiri

jarvisunits — parses the list of arguments, try to interpret last argument as unit

siri2jarvis — converts spoken or written natural language sentences into a simple grammar

Detailed description

fourbardesignbysiri

generates 4bar-linkages from a dictation fnct of Siri on MAC

In: **txt**: string lines to interpret or empty as interpreter **PS**: Pose strcut that should be modified **d1**: circle diameter for the CPLM contour; default is 10 **d2**: second circle diameter (oval); default equals d1 **n**: number of edges of the circle contour; empty for round

Out: **PS**: Pose specified by language or text string

jarvis

Natural Language Interface JARVIS

In: **domain**: knowledge domains; default is '4Bars'

jarvisChatSGLib

this fnct/script is a ChatGPT based control of the SGLib and defines the development start of SGLib 5.5

jarvisdictionary

reads writes a excel sheet containing string to replace and code to execute

In: **fname**: file name of the excel-sheet to use **excel**: if true, open the excel sheet after reading; default is false

Out: **fulltab**: Table **b**: names **c**: comment lines

jarvismemory

sets or gets a pose from jarvis / fourbardesignbysiri

In: **PSN**: New Pose to store in global variable fourbardesignbysirimem

Out: **PS**: Pose stored currently in global variable fourbardesignbysirimem

jarvisunits

parses the list of arguments, try to interpret last argument as unit

In: **args**: list of arguments

Out: **fact**: factor for multiplication such as 1e-3 for 'm' or 1e3 for 'k' **mayu**: unit **newvar**: new varargin list

siri2jarvis

converts spoken or written natural language sentences into a simple grammar

In: **str**: string to interpret **fn**: filename of the dictionnary used **noexec**: unused flag; default is false

Out: **cmd**: cell list of commands, the first is always a string **str**: string that was finally processed **comtab**: table of terminal words that were allowed as commands

7.8.3 Claude Interface [guide]

7.8.4 chatGPT Interface [guide]

Functions in this section:

chat — single line communication with openAI's chat gpt with persistent chat history

chat1 — single line communication with openAI's chat gpt

chat2table — Take a char text and create a table or a cell array.

chatas — communicates with an openAI in a named chat based on a thread and assistant

chatclipcorrect — prepares sepell checking and translation of the clipboard

chatcode — returns matlab code and writes a matlab fnct into the working directory

chatdiagramm — Generates MATLAB code to display a diagram based on OpenAI Chat.

chatfilename — erzeugt einen 60 Zeichen filename unter verwendung von chat1

chatGPT — simply opens chatGPT web-browser window

chatimage — creates a picture/sketch/diagram/image of a text using "dall-e-3"

chatimage2video — creates a video file on disk by openAI' Sora (12 seconds cost about 2 Dollar) based on a picture

chatm — single line communication with openAI's chat gpt with persistent chat history and chat-memories.txt

chatscienceclozetext — Chat fnct to supplement scientific cloze texts

chatvideo — creates a video file on disk by openAI' Sora (12 seconds cost about 2 Dollar)

chatweb — Simply calls chatGPT internet page

PDFcreateOpenAIfilenames — renames files in a folder by an openAI LLM interface including Name and adresse

SGcreateDocuZIP4GPT — creates a zip file including all SG-Lib html files od the directory SGLIB_HELP_Structures

SGcreateJSONfnctSignatures4GPT — Generates a JSON for GPT from the SG-Lib documentation "JSONfnctSignatures4Matlab"

Detailed description

chat

single line communication with openAI's chat gpt with persistent chat history

Out: **y**: Response from openAI when using your paid openAI API account

chat1

single line communication with openAI's chat gpt

Out: **y**: Response from openAI when using your paid openAI API account

chat2table

Take a char text and create a table or a cell array.

In: **txt**: text desciption of a table **"cell"**: if used, an cell instead of a table is returned

Out: **TAB**: matlab table **code**: code to create the matlab table; cann be used in a matlab file later

chatas

communicates with an openAI in a named chat based on a thread and assistant

In: **chatname**: chatname **message**: message to the chat (assistant / thread)

Out: **allresp**: cell array of the response content from the assistant/thread

chatclipcorrect

prepares sepell checking and translation of the clipboard

Out: **corr**: corrected and optional translated text

chatcode

returns matlab code and writes a matlab fnct into the working directory

In: **descr**: Complete header including input out parameter an their description **fname**: desired name of cuntin

Out: **flong**: full file name of the m file; Names start with CC_ and can get an additional **txt**: code of the matlba file

chatdiagramm

Generates MATLAB code to display a diagram based on OpenAI Chat.

In: **data**: A char text with data preferably in Excel format (columns separated by semicolons) **fname**: An optional Filename; default is titleofcaller **todo**: Explanation of what exactly is desired **myname**: author name credited in the generated code; default is username

Out: **y**: matlab code for execution

chatfilename

erzeugt einen 60 Zeichen filename unter verwendung von chat1

In: **descr**: Textbeschreibung **datetxt**: Optional date text to include in the file name

Out: **shrt**: 60 Zeichen Filename

chatGPT

simply opens chatGPT web-browser window

chatimage

creates a picture/sketch/diagram/image of a text using "dall-e-3"

In: **descr**: description of the picture

Out: **img**: image **GPT**: GPT Response of openAIImages

chatimage2video

creates a video file on disk by openAI' Sora (12 seconds cost about 2 Dollar) based on a picture

In: **I**: Image or file name of the image. It will be resized to fit 1280x720 or other formats **prompt**: description of the video content - the more the better **nvp**: addition parameter such as "size", "720x1280"

Out: **response**: response from openAI - mp4Bytes = response.Body.Data;

chatm

single line communication with openAI's chat gpt with persistent chat history and chat-memories.txt

Out: **y**: Response from openAI when using your paid openAI API account

chatscienceclozetext

Chat fnct to supplement scientific cloze texts

In: **clozetext**: scientific cloze text

Out: **resp**: response with completed cloze text

chatvideo

creates a video file on disk by openAI' Sora (12 seconds cost about 2 Dollar)

In: **prompt**: description of the video content - the more the better **nvp**: addition parameter such as "size", "720x1280"

Out: **response**: response from openAI - mp4Bytes = response.Body.Data;

chatweb

Simply calls chatGPT internet page

PDFcreateOpenAIfilenames

renames files in a folder by an openAI LLM interface including Name and adresse

In: **last**: number of documents; default is 1 **dname**: directory name to parse default is Postordner (LUETH)/*.pdf

SGcreateDocuZIP4GPT

creates a zip file including all SG-Lib html files od the directory SGLIB_HELP_Structures

In: **maxsize**: maximum file size in bytes; default is 2000000 (2 MB)

Out: **fname**: file name of the zip file

SGcreateJSONfnctSignatures4GPT

Generates a JSON for GPT from the SG-Lib documentation "JSONfnctSignatures4Matlab"

In: **fnameML**: default is [SGlibpath filesep 'functionSignatures.json'] **fnameGPT**: default is [SGlibpath filesep 'GPTfunctionSignatures.json']);

Out: **fnameGPT**: location of GPT JSON for SG-Lib default is fnameGPT

8 STL & 3D Printing & Voxel Models [guide]

8.1 STL Import/Export [guide]

Functions in this section:

appsavegcaSTL — Application callback procedure for writing the current figure as STL

appuireadSTL — Application callback procedure for reading a STL file

SGarchivefilename — returns a filename for fast access to similar geometries

SGmeshlab — opens a solid geometry in an STL viewer

SGread3MF — opens and reads a 3mf file with respect to the microsoft specification

SGreadMAT — loads a Solid Geometry or other struct as mat file on disk

SGreadOBJ — reads in an OBJ file

SGreadPLY — reads in a Point Cloud in PLY Format and creates a surface model by marching cube

SGreadSTL — reads in an SG from an STL File

SGreadSTLdir — reads all STL files of a folder into one SG cell list

SGreadSTLreducedFL — Reads in a STL file and reduces immedeatly the number of faces

SGreadstructfile — Reads in an matlab struct from an ascii struct file

SGreadtable — This fnct reads in EXCEL tables with comments

SGwriteIcon — Writes an Icon/JPG File with a 3D view of a SG

SGwriteMAT — saves a Solid Geometry or other struct as mat file on disk

SGwriteMultipleSTL — writes a cell list of solid geometry in different STL files

SGwriteSeparatedSTL — writes solid geometry in different STL files

SGwriteSTEP — This fnct exports Solid Geometries as STEP File to disk

SGwriteSTL — Writes a STL file for a solid geometry

SGwriteSTLarchive — Creates an archive directory for separated solids and surfaces of different Solids/STL files

SGwriteSTLplot — just a combination of SGwriteSTL and SGTplotalpha

SGwriteSTLshrinked — writes shrinked STL files on disk

SGwritestructfile — writes a matlab struct into an ascii struct file

SGwritetable — writes a table similar to writetable

SGwriteVRML — Writen an STL Geometry in a VRML File Format.

STEPmanifoldSolidBrepofSG — returns the text lines for a ISO-10303-21 AP242 Manifold Geometry Boundary Representation

STLAsctoBin — reads and STL-File in ASCII format and writes it in binary format

stldir — returns path, or filename, or list of existing files

STLreadfiletype — returns 'a' for ascii, 'b' for binary and -1 for file not found

stlsolid — generates a STL file body from a vertex list and facet list

VLFLreadOBJ — reads in an OBJ file

VLFLreadSTL — reads an ascii STL-File as vertex & facet list

VLFLreadSTLb — reads a STL-File in binary format

VLFLwriteSTL — writes a STL object given by vertex list and facet list in a STL-file

VLFLwriteSTLb — writes a STL object in binary mode given by a vertex list and facet list into a STL-file

Detailed description

appsavegcaSTL

Application callback procedure for writing the current figure as STL

appuireadSTL

Application callback procedure for reading a STL file

SGarchivefilename

returns a filename for fast access to similar geometries

In: **SG**: Solid Geometry

Out: **fn**: filename without STL **fndim**: part for dimension comparison (Volumen and X*Y*Z) **fngeo**: part for geometry comparision (VL/FL)

SGmeshlab

opens a solid geometry in an STL viewer

In: **SG**: Solid Geometry **colored**: if false there are no colors; default is true; **RBchange**: if true; A.FC=A.FC(:,[3 2 1]); default is false

SGread3MF

opens and reads a 3mf file with respect to the microsoft specification

In: **fname**: filename

Out: **SGc**: XML text **text**: XML text

SGreadMAT

loads a Solid Geometry or other struct as mat file on disk

In: **FName**: Filename of mat file to read

Out: **SG**: struct read from file

SGreadOBJ

reads in an OBJ file

In: **FName**: File Name **mag**: magnification; default is 1 **check**: checker; default is false

Out: **SG**: Solid Geometry **H**: Colors, not used in SG-Lib 4.7 **AbC**: Colors, not used in SG-Lib 4.7 **FName**: File Name, added in SG-Lib 4.7

SGreadPLY

reads in a Point Cloud in PLY Format and creates a surface model by marching cube

In: **FName**: File Name or Empty for interactive selection **mag**: magnification of point cloud; default is 1000 **res**: resolution to avoid holes; default is 4

Out: **SG**: Solid Geometry **VL**: Vertex list **ptCloud**: original point cloud

SGreadSTL

reads in an SG from an STL File

In: **FName**: File Name **mag**: optional magnification value **rel**: spatial relations and modifications by SGtransrelSG such as 'shrink','150

Out: **SG**: Solid Geometry **H**: header string 80 chars **AbC**: Attribute by Count per facet uint16(n), SG.FC=bgr162rgb(AbC); **FName**: FileName

SGreadSTLdir

reads all STL files of a folder into one SG cell list

In: **dirname**: folder name or single file in a folder **mag**: magnification; default is 1 **fac**: maximum number of facets overall **col**: Color to assign to the read solids

Out: **SG**: Cell list of Solids **AParts**: created from StandardParts_parts.txt in this folder using SGpartslistfindproductcodes

SGreadSTLreducedFL

Reads in a STL file and reduces immedeatly the number of faces

In: **PAR**: same parameter set as SGreadSTL **"nFL"**: optional string parameter; default is 60000

Out: **SG**: Solid Geometry

SGreadstructfile

Reads in an matlab struct from an ascii struct file

In: **fname**: filename of the struct

Out: **PS**: Struct **fname**: used filename

SGreadtable

This fnct reads in EXCEL tables with comments

In: **fname**: filename of excel file **sortc**: column index to sort rows by; if >0 the table is sorted on this column; default is 4 **conv**: if true, dates are converted to strings and numeric strings to doubles; default is true

Out: **Tab**: Table **VNames**: Variable Names **CTab**: Comment lines, cell list

SGwriteIcon

Writes an Icon/JPG File with a 3D view of a SG

In: **SG**: Solid Geometry **FName**: Full File Name. The Extension defines the Format (.JPG, .PNG)

SGwriteMAT

saves a Solid Geometry or other struct as mat file on disk

In: **SG**: Solid Geometry variable name **FNAME**: optinale file name for desktop write

Out: **FNAME**: Complete filename path name extension

SGwriteMultipleSTL

writes a cell list of solid geometry in different STL files

In: **SG**: A cell list of different solids for different file **FNAME**: optional file name **ONAME**: optional text name **Partext**: Partslist as text string, create by SGpartslistastext **iscenter**: 'x'y'z are the axis to center the object **"turn"**: if used, the objects are automatically turned optimally 3D printing **"nochange"**: if used, the objects are not centered which is required for multi color printing (snapmaker 2026)

Out: **ndir**: new directory

SGwriteSeparatedSTL

writes solid geometry in different STL files

In: **SG**: Solid Geometry **FNAME**: File and Directory Name **ONAME**: Solid Name

Out: **ndir**: name of the created directory holding the separate STL files

SGwriteSTEP

This fnct exports Solid Geometries as STEP File to disk

In: **SG**: Solid Geometry file name **FNAME**: File name

Out: **fname**: STEP file name

SGwriteSTL

Writes a STL file for a solid geometry

In: **SG**: Object name **FNAME**: Optional file name **ONAME**: Optional solid name **vcol**: Optional facet color for the complexe solid **NL**: Optional Normal vector list **"turn"**: string option 'turn'; if set, turns the solid for additive manufacturing (SGturn4AM) before writing **"noturn"**: string option 'noturn'; if set, keeps the orientation and does not turn the solid before writing **"allframes"**: string option 'allframes'; if set, writes all frames of the solid into the STL file

Out: **fname**: file name of STL

SGwriteSTLarchive

Creates an archive directory for separated solids and surfaces of different Solids/STL files

In: **SG**: Solid Geometry **atype**: 'solid' or 'surfaces' depending on separating demand **nmax**: Maximum number of solids per archive

Out: **SGC**: Index list of the written files

SGwriteSTLplot

just a combination of SGwriteSTL and SGTplotalpha

In: **SG**: Solid Geometry to write as STL **sname**: filename for STL file and PNG file

Out: **fnames**: filename of STL file

SGwriteSTLshrinked

writes shrinked STL files on disk

In: **SG**: Solid Gemetry **size**: maximum dimensions in mm oder shinking factor; default is 100mm **fname**: optional file name

Out: **sname**: file name **SG**: Shrinked Solid

SGwritestructfile

writes a matlab struct into an ascii struct file

In: **STRUCT**: struct to save on disk **fname**: optional file name

Out: **fullname**: full file name for struct file

SGwritetable

writes a table similar to writetable

In: **fname**: full filename **Tab**: table to write **Comm**: Comment lines **fsort**: if true, the function tries to sort the table first

Out: **fname**: full filename of the written table

SGwriteVRML

Writen an STL Geometry in a VRML File Format.

In: **SG**: Solid Geometry **FNAME**: Optional Filename **ONAME**: optional Solid name; default is by username **vcol**: optional Color for the solid

Out: **fname**: File name of the written VRML file

STEPmanifoldSolidBrepofSG

returns the text lines for a ISO-10303-21 AP242 Manifold Geometry Boundary Representation

In: **SG**: Solid Geometry to create a Manifold Geometry Bopundary representation **alpha**: Limit angle for separating feature surfaces; default is 0.01 **mm**: number of used step lines already for other BREPS; default is 0

Out: **ttorg**: STEP text lines for the Manifold Geometry Bopundary representation **mm**: last number of created step line

STLAsctoBin

reads and STL-File in ASCII format and writes it in binary format

In: **fname**: Filename on desktop

stldir

returns path, or filename, or list of existing files

In: **fname**: strinf to search for in the stl default directory

Out: **flist**: stldir or path to stldir or list of all files that contains the search string **fdir**: corresponding dir result **htab**: cell table of matching files with hyperlink commands to read them

STLreadfiletype

returns 'a' for ascii, 'b' for binary and -1 for file not found

In: **FName**: Full file name including path and extension '.STL'

Out: **c**: -1=not found, 'a'=asci format, 'b'=binary format

stlsolid

generates a STL file body from a vertex list and facet list

In: **fid**: is the file identifyer of the file to write in, -1= screen **VL**: is the given list of vertexes **FL**: is the given list of facets

VLFLreadOBJ

reads in an OBJ file

In: **FName**: File Name **mag**: magnification; default is 1 **check**: checker; default is false

Out: **VL**: Vertex list **FL**: Facet list **H**: Header, not used in SG-Lib 4.7 **AbC**: Colors, not used in SG-Lib 4.7 **FName**: Filename

VLFLreadSTL

reads an ascii STL-File as vertex & facet list

In: **FName**: Filename of the STL-File (ASCII) **mag**: magnification factor **"check"**: if true, VLFLcheckmeshlab is called; default is false

Out: **VL**: Vertex List **FL**: Facet List **S1**: solid string **S2**: endsolid string **STL**: The original facet list that uses no vertex list **AbC**: Attribute by Count per facet uint16(n) **NL**: Normal vector list

VLFLreadSTLb

reads a STL-File in binary format

In: **FName**: Filename **mag**: optional magnification factor 1000 for m into mm **"check"**: optional call of VLFLchecker(VL,FL);

Out: **VL**: Vertex list (x y z) **FL**: Facet list (v1 v2 v3) **H**: Header 80 Uint8 **n**: Number of triangles **AbC**: Attribute by Count per facet uint16(n) **NL**: Normal vector list (x y z) **stmp**: solid name string extracted from the STL header (text between quotes)

VLFLwriteSTL

writes a STL object given by vertex list and facet list in a STL-file

In: **VL**: is the vertex list **FL**: is the facet list **FNAME**: name of the desired file on Desktop **ONAME**: name of the STL object

VLFLwriteSTLb

writes a STL object in binary mode given by a vertex list and facet list into a STL-file

In: **VL**: Vertex list **FL**: Facet list **FNAME**: Name of the desired file on Desktop **ONAME**: Solid name or string **vcol**: color **NL**: optional normal vector list (x y z) per facet; default is empty, then normals are computed

Out: **fname**: File name of the written STL file

8.2 STL Packaging [guide]

Functions in this section:

amazoncode2link — returns a valid link from the 10-digit alphanumeric codes or opens the amazon product page (german amazon) in the browser

amazonlink2code — returns the 10-digit alphanumeric codes code from an amazon product page link

binpacking3D — returns a straight forward bin packing

filenameadddate4gcode — CHANGES THE NAME FOR GCODE FILES

gcodecleanup2dir — creates folder for gcode files if they are anycubic gcode

gcodecopytemp — just makes copies of an ANYCUBIC/CURA gcode file with changed hot end and base temperature

productcode — Database fnct to store components to order

SGarrangeSG — arranges a cell list of solids in a single row

SGboxing — returns a case for boxing a SG model

SGboxpacking — combines SGpacking and SGboxing

SGClayout — returns a modified arranged solid table

SGcopyrotZ — copies a solid around the z-axis

SGgcodefilechangetemp — changes the temperature of hotend and base plate in a marlin gcode file

SGgcodegetinfo — Auxiliary fnct to get some parameter by interpreting gcode files or strings

SGgcoderead — reads in a ascii gcode file and creates a line table

SGgcodesimulate — plots a simulation of the gcode-processing using Marlin dialect

SGpartsfuncparams — creates a documentation string to explain the fnct call including all parameters

SGpartslist — appends parts to a parts list or parts lsit to parts list

SGpartslistastext — appends parts as text lines to a parts list and returns text lines

SGpartslistfindproductcodes — creates a text line or cell list of hyperlinks to order parts of a parts list

SGrotate4print — rotates an object to achieve as long as possible lines on the floot

VLFLboxing — returns for a given surface, two contours edge list, a solid box structure

Detailed description

amazoncode2link

returns a valid link from the 10-digit alphanumeric codes or opens the amazon product page (german amazon) in the browser

In: **code**: 10 alphanumerical char code of amazon **ext**: default is 'de'; but can be used to open other contries too

Out: **url**: url to use as web(url)

amazonlink2code

returns the 10-digit alphanumeric codes code from an amazon product page link

In: **link**: url from the prodcut page

Out: **code**: 10 alphanumerical char code of amazon

binpacking3D

returns a straight forward bin packing

In: **LAY**: Layout dimensions; default [100,60,80] **BOXL**: nx4 [index size-x size-y size-z] **grid**: grid size; default is 1 mm

Out: **BOXL**: nx4 [index pos-x pos-y pos-z] **h**: estimation of a square side length if all bin have the same height

filenameadddate4gcode

CHANGES THE NAME FOR GCODE FILES

In: **fname**: filename (wildcards allowed) **renn**: if true, the file will be changed (if possible) **fmdat**: Date format **protf**: Protocol flag

Out: **newfile**: full filename to be written or written **fname**: core filename without prefix and postfix **fdname**: Dated file name

gcodecleanup2dir

creates folder for gcode files if they are anycubic gcode

In: **dname**: directory name

gcodecopytemp

just makes copies of an ANYCUBIC/CURA gcode file with changed hot end and base temperature

In: **fname**: existing gcode file name

productcode

Database fnct to store components to order

In: **code**: code **text**: descriptor **over**: overwrite if text is already stored **"mail"**: string flag 'mail' to send the product link by email; default is off

Out: **url**: url 2 order the parts **text**: descriptor if just the code was used in a parts list

SGarrangeSG

arranges a cell list of solids in a single row

In: **SGc**: Cell list of Solid Geometries **dist**: fixed distance vector **adim**: 'x' or 'y' or 'z' for arrangement

Out: **SG**: One single solid without any cell structure

SGboxing

returns a case for boxing a SG model

In: **SG**: Solid geometry to be boxed **wall**: Wall thickness; default is 1 **slot**: Space between box and enclosed part; default is 1 **stext**: Text ontop of box **ms**: size of the square holes in the walls **ha**: 'c', 'l', 'r' for text alignment

Out: **SGB**: Solid geometry of the housing box and cover

SGboxpacking

combines SGpacking and SGboxing

In: **SG**: SG or SGcs to pack and box **tstr**: text string for cover **dim**: Box dimensions for printing **turn**: false==keep orientation; true==optimize space; default is true

Out: **SGC**: complete print job SG **SGA**: packaged solids of SG **SGB**: box aroound SGA

SGClayout

returns a modified arranged solid table

In: **ST**: Solid Table List **LayX**: Layout dimension in X **LayY**: Layout dimension in Y **LayZ**: layout dimension in Z; internally forced to 100

Out: **STN**: Sorted table list that fit in the layout **ST**: Sorted table list of the parts that did not fit into the layout **UX**: Used rectangle x dimension **UY**: Used rectangle y dimension **A**: Percentage of use

SGcopyrotZ

copies a solid around the z-axis

In: **SG**: Solid geometry **rotangz**: list of rotation angles **tdist**: translation vector before each copy **n**: number of repeats

Out: **SGN**: resulting solid

SGgcodefilechangetemp

changes the temperature of hotend and base plate in a marlin gcode file

In: **TEXT**: gcode string or gcode file name **HotE**: Temperature for hotend, default is [220 225] **Base**: Temperature for base plate, default is [60 65]

Out: **TEXT**: new gcode string or written gcode file **cname**: file name for gode file

SGgcodegetinfo

Auxiliary fnct to get some parameter by interpreting gcode files or strings

In: **TEXT**: gcode string or gcode file name

Out: **inf**: struct containing information of gcode

SGgcoderead

reads in a ascii gcode file and creates a line table

In: **fname**: File Name or empty for dialog

Out: **TEXT**: unsigned char **s**: new line positions **fname**: full file name of the read gcode file

SGgcodesimulate

plots a simulation of the gcode-processing using Marlin dialect

In: **fname**: file name of gcode file or empty for dialog

SGpartsfuncparams

creates a documentation string to explain the fnct call including all parameters

In: **params**: exact same list of parameters as used in function definition **"Partslist"**: if the string "Partslist" is present, the text is added to a parts list

Out: **txt**: resulting text

SGpartslist

appends parts to a parts list or parts lsit to parts list

In: **parts**: parts list or empty for a new list **"file"**: either new parts or a parts list or a text output

Out: **parts**: parts list **txt**: text output to write into a file **fname**: name of the written parts text file

SGpartslistastext

appends parts as text lines to a parts list and returns text lines

In: **oldlist**: parts list or empty as text string or cell list **addlist**: parts list to add as as text string or cell list

Out: **newlist**: parts list as text list **AParts**: parts list as cell list

SGpartslistfindproductcodes

creates a text line or cell list of hyperlinks to order parts of a parts list

In: **AParts**: parts list as cell or text or file name

Out: **urls**: text line with hyperlink descriptions to order components **texs**: cell list of the trimmed part-code text lines (label plus 'PC[...]' code) for matched parts **AParts**: parts list normalized to a trimmed char string

SGrotate4print

rotates an object to achieve as long as possible lines on the floot

In: **SG**: Soli Geometry **ez**: direction of the base plate; default is [0 0 1] **st**: number of auxiliary grid points; default is at least 3

Out: **SGN**: Rotated Geometry

VLFLboxing

returns for a given surface, two contours edge list, a solid box structure

In: **VLLA**: Vertex List **FLIF**: Facet list of the inner surface **ELB**: Bottom contour edge list **ELT**: Bottom contour edge list **wall**: wall size; default is 1

Out: **VLLA**: Vertex list **FLIF**: Facet list of the inner contour (now swapped) **FLOF**: Facet list of the outer contour (now swapped) **FLBF**: Facet list of the bottom surface **FLTF**: Facet list of the top surface

8.3 STL Surfaces Analysis [guide]

Functions in this section:

insideCPS — tests whether a point list is inside of a CPL polyshape or CPS

isInteriorofCPL — tests whether a point is inside of a closed contour in 2D!

SGpacking — returns a package of all solids of a cell list

surfacesofSG — returns all feature surfaces (open boundaries) that have a limited angle between their normal vectors

VLFLtrianglegeometry — returns all important information on the triangles of a facet list

Detailed description

insideCPS

tests whether a point list is inside of a CPL polyshape or CPS

In: **CPL**: Vertex list of the contour **PL**: Point to check **tol**: relative tolerance band around the contour for the on-contour test; default is 1e-5

Out: **k**: +1==inside, -1==outside, 0== on the contour

isInteriorofCPL

tests whether a point is inside of a closed contour in 2D!

In: **CPL**: Vertex list of the contour **PL**: Point to check

Out: **ii**: +1==inside or on, 0==outside

SGpacking

returns a package of all solids of a cell list

In: **SGC**: Cell list of solid geometries **dim**: dimension of package, default is [85 65 100] or 'eos' **dist**: minimal distance between solids; default is 1; **ez**: optional ez vector for turning limitation; default is empty; **turn**: true== turn to flat; false==unchanged orientation; default is true

Out: **SGN**: Packed solid geometry as one solid **SG**: Packed solid geometry as cell list

surfacesofSG

returns all feature surfaces (open boundaries) that have a limited angle between their normal vectors

In: **TR**: Solid Geometry or Triangulation **alpha**: maximum angle between two facets; default is 1 ~ 60 degree **v**: resolution tolerance for grouping; default is 1e-3

Out: **FIL**: Index List for the Surface Indices **k**: maximum value of FIL **FNL**: Facet Normal List - TR3neighborsAngle **NAL**: Neighbor Angle List - TR3neighborsAngle **NBL**: Neighbor List - TR3neighborsAngle **Ai**: Area of the indexed surfaces 1:k

VLFLtrianglegeometry

returns all important information on the triangles of a facet list

In: **VL**: Vertex list **FL**: Facet list [1 2 3; etc **fi**: face index; makes sense only if there is no output

Out: **w**: angle [w1 w2 w3] wrt. [v1 v2 v3] **d**: length [d1 d2 d3] wrt. opponent edge to [v1 v2 v3] **A**: area of triangle **NL**: Normal vector of facets **NN**: Neighbor of facet wrt. opponent edge to [v1 v2 v3]

8.4 Vector Export (SVG/DXF) [guide]

Functions in this section:

CPL2DXF — Returns LWPOLYLINE strings with increasing object number

CPLwriteDXF — writes a 2D DXF File from a CPL

CPLwriteSVG — writes a SVG file for a laser cutter

Detailed description

CPL2DXF

Returns LWPOLYLINE strings with increasing object number

In: **CPL**: CPL to convert into LWPOLYLINEs **pcnt**: already existing number of objects; default is 76

Out: **texta**: textstring for inserting into DXF file

CPLwriteDXF

writes a 2D DXF File from a CPL

In: **CPL**: CPL **fname**: optional file name for write; default path is desktop **fcl**: force contour close; default is true

Out: **fname**: filename of DXF file **text**: written text in file

CPLwriteSVG

writes a SVG file for a laser cutter

In: **CPL**: Contour list **FNAME**: Filename **ONAME**: Creator Name; default is calling function name **cutter**: if true; CPLs are sorted inside out in different colors; default is false **colmap**: stroke color or colormap; default is {} **stkwd**: stroke width

8.5 Voxel Models [guide]

8.5.1 VM Import & Scaling [guide]

Functions in this section:

cartofvox — returns cartesian coordinates from voxel coordinates

dicomCollectionSeparation — selects the fitting other files for a stack of images or a directory

VMofimagestack — performs the inverse radon on an image stack

VMreaddicom — reads an image (nxn) or image stack (nxnxm) from a dicom directory

VMreaddicomdir — reads a dicom directory into an image stack

VMreadnifti — reads in a nifty volumentric file

VMresize — resizes a voxel image model

VMuidicom — reads in a dicom file or opens a dialog first

voxofcart — returns voxel coordinates from cartesian coordinates

Detailed description

cartofvox

returns cartesian coordinates from voxel coordinates

In: **vs**: voxel size [bx by bz] **vox**: voxel coordinate vector

Out: **p**: cartesian coordinate vector

dicomCollectionSeparation

selects the fitting other files for a stack of images or a directory

In: **DirName**: empty for selection or directory name or file name **FName**: optional file name or cmd suc as 'separated'

Out: **A**: table of dicomCollection of this directory **FL**: fitting file list for using dicomreadVolume

VMofimagestack

performs the inverse radon on an image stack

In: **imnam**: image matrix or folder name (File name) **r90**: rotate the image minus 90 degree; default is true **scl**: scale the image; default is 1 **angle**: default i 360

Out: **VM**: Voxel Model

VMreaddicom

reads an image (nxn) or image stack (nxnxm) from a dicom directory

In: **pathname**: optional pathname **filename**: optional pathname **xsiz**: size of te image; default is 512; use 320 for conebeam C-arm Vario3D

Out: **IM**: voxel volume model (VM) **info**: DICOM metadata/info struct

VMreaddicomdir

reads a dicom directory into an image stack

In: **pathname**: path of dicom directory

Out: **IM**: Image stack **vs**: estimated voxel size **TAG**: '' **zL**: slice location list in [mm] **sL**: pixel size in [mm]

VMreadnifti

reads in a nifty volumentric file

In: **fname**: filename

Out: **VM**: Voxel Model **vs**: voxel size

VMresize

resizes a voxel image model

In: **VM**: voxel mode; [X x Y x Z] **dm**: dimension either scale [0.5 0.5 0.5] or vs **vs**: optional voxel size for rescaling; default is dm

Out: **FVM**: New Voxel Model with new dimensions **vs**: Voxel size after resizing

VMuidicom

reads in a dicom file or opens a dialog first

In: **filename**: optional filename

Out: **X**: voxel model of dicom stack **tags**: empty, if no dicom file

voxofcart

returns voxel coordinates from cartesian coordinates

In: **vs**: voxel size [bx by bz] **p**: cartesian coordinate vector

Out: **vox**: voxel coordinate vector

8.5.2 VM Visualization [guide]

Functions in this section:

ELofmatrixneigbors — creates a complete edge list for a edgelist 4x4 neighborhood of a matrix

ELofmatrixneigborweight — creates a weighted edge list for a 4x4 neighborhood wrt to weight criteria

GPLnearestpoint — returns the nearest point index within a Grid point list

GPLofmatrix — creates for a 2D matrix a grid point list with coordinates

marchingcubes — marching cubes algorithm

matrixplot — plots a 2D matrix as grid point list with coordinates

VLplotvoxel — plots a voxel at the position VL

VMaxis — simply returns the minimal and maximal intensity values of a Voxel Model

VMcaxis — adjusts the caxis of the VMplot diagram

VMcaxis2 — adjusts the caxis of the VMplot diagram

VMgetSubplot — returns the current active subplot

VMginput — returns a point position by clicking into VMplot

VMimage — returns three crosssectional views of an image stack (voxel model)

VMimrot90 — returns an 90 degree rotated image (nxm) or image stack (nxmxk)

VMintensityscale — returns an intensity scaled voxel image

VMmontage — plots all images of an image stack into one figure

VMneighborhood — returns of a n x m or n x m x k 2D or 3D Matrix the neighborhood values

VMof2Dprojections — this fnct reads in a directory full of projection images and creates a voxel model

VMofSGweight — returns a voxel model of a Solid Geometry using SGweight/SGisinterior

VMofVL — returns a Voxel model from a vertex list

VMplot — plots a 3D volumetric matrix in 2x2 view

VMplotvoxels — plots a voxel model as transperent cubes

VMpseudo3D — returns an 2D image of pseudo 3D rendered surface

VMscatter — Plots a 3D matrix as particle model

VMwindowplot — plots a voxel model using the window caxis fnct

Detailed description

ELofmatrixneigbors

creates a complete edge list for a edgelist 4x4 neighborhood of a matrix

In: **rc**: 2D Matrix or [rows cols]

Out: **EL4**: straight neighbor edge list related to reshape(M,'',1) **EL8**: diagonal neighbor edge list related to reshape(M,'',1)

ELofmatrixneigborweight

creates a weighted edge list for a 4x4 neighborhood wrt to weight criteria

In: **M**: 2D Matrix or [rows cols] **method**: such as 'abs' **neighb**: '8' or '4'; default ist '8'

Out: **EL**: Edge list related to reshape(M,'',1)

GPLnearestpoint

returns the nearest point index within a Grid point list

In: **GPL**: Grid Point List **p**: point to check **k**: subset of point indices to search; default is all points 1:size(GPL,1)

Out: **i**: nearest point **p**: point coordinates

GPLofmatrix

creates for a 2D matrix a grid point list with coordinates

In: **M**: Matrix (n x m) **flpud**: flipud for rows values; default is true **ax**: [xmin xmax ymin ymax]; default is [1 size(M,2) 1 size(M,1)]

Out: **GPL**: GPL coordiate list; sorted 1:numel(M) **X**: X coordiate of GPL; sorted 1:numel(M) **Y**: Y coordiate of GPL; sorted 1:numel(M)

marchingcubes

marching cubes algorithm

In: **x**: x of meshgrid **y**: y of meshgrid **z**: c of meshgrid **c**: cubes in voxel model **iso**: threshold **colors**: Optional scalar field for per-vertex colors

Out: **V**: Vertices **F**: Faces **col**: colors

matrixplot

plots a 2D matrix as grid point list with coordinates

In: **M**: Matrix (n x m) **flpud**: flipud for rows values; default is true **ax**: [xmin xmax ymin ymax]; default is [1 size(M,2) 1 size(M,1)] **txtsize**: text size; default is 0; **plotnan**: true => nan plotted as square; default is false

Out: **h**: handle to graphics objects

VLplotvoxel

plots a voxel at the position VL

In: **VL**: point or list of points **vs**: size of voxel; default is [1 1 1] **c**: globol color or list of colors of same size as VL **a**: globol transparency or list of transparency of same size as VL **ea**: edge transparency of the voxels; default is 0.1

VMaxis

simply returns the minimal and maximal intensity values of a Voxel Model

In: **V**: Voxel model

Out: **vax**: [min max]

VMcaxis

adjusts the caxis of the VMplot diagram

In: **cax**: [cmin cmax]; default is min and max of the current! surfaces

Out: **cax**: selected Values if called without input paramter

VMcaxis2

adjusts the caxis of the VMplot diagram

In: **VM**: [cmin cmax]; default is min and max of the current! surfaces **sect**: number of sections for the histogramm; default is 64

Out: **ca**: selected Values if called without input paramter

VMgetSubplot

returns the current active subplot

Out: **subpn**: current subplot position; empty if figure is not valid

VMginput

returns a point position by clicking into VMplot

In: **n**: number of points **u**: update VMplot; default is true

Out: **PL**: position **t**: subplot

VMimage

returns three crosssectional views of an image stack (voxel model)

In: **IM**: 3D Matrix **p**: optional point of interest (row, column, slice)

VMimrot90

returns an 90 degree rotated image (nxm) or image stack (nxmxk)

In: **I**: image or image stack

Out: **NI**: new image or image stack

VMintensityscale

returns an intensity scaled voxel image

In: **V**: Voxel mode **cax**: desired intensity interval; default is [0 1]

Out: **c**: [minc maxc] **V**: scaled to 0 to 1

VMmontage

plots all images of an image stack into one figure

In: **V**: Voxel model

Out: **h**: handle of image

VMneighborhood

returns of a n x m or n x m x k 2D or 3D Matrix the neighborhood values

In: **M**: Matrix, the neighborhood is desired **nn**: neigborhood

Out: **NH**: Matrix of size [size(M) n] consisting already all neighbors **ni**: Neighborhoodindex for the use like M(ni), which is the same as NH **iM**: matrix of linear indices of M (same size as M)

VMof2Dprojections

this fnct reads in a directory full of projection images and creates a voxel model

In: **dirname**: directory name or file name; default is empty => manual select **angle**: angle list or maximum area value of the projection images; default is 360 **siz**: size of the images; default is squared image matrix **lim**: limit; default is '' == automatic **scl**: scaling; default is 1 **r90**: rotate the images 90 degree; default is true

Out: **VM**: Voxel model **SG**: Surface Model **IMG**: Image Stack

VMofSGweight

returns a voxel model of a Solid Geometry using SGweight/SGisinterior

In: **SG**: Solid Geometry **d**: size of voxel

Out: **VM**: Voxel Model **sz**: Voxel size **SGN**: Optional Solid by marching cube **xl**: list of x values of the Voxel Model **yl**: list of x values of the Voxel Model **zl**: list of x values of the Voxel Model

VMofVL

returns a Voxel model from a vertex list

In: **VL**: Vertex list **res**: Optional resolution; default is 2m **neigh**: flag to include neighboring voxels; default is true

Out: **VM**: Voxel model **vs**: size of voxel **SG**: optional Solid Geometry

VMplot

plots a 3D volumetric matrix in 2x2 view

In: **IM**: 3D Matrix **p**: optional point of interest; default is center **SG**: optional solid of the Volume;

VMplotvoxels

plots a voxel model as transperent cubes

In: **VM**: Voxel Model **vs**: size of voxels; default is [1 1 1] **c**: color default is 'k' **ea**: edge alpha value; default is 0.5

VMpseudo3D

returns an 2D image of pseudo 3D rendered surface

In: **V**: Voxel model **lim**: surface intensity value **maxc**: maximum used color; default is 4095 **s**: cut value: start to display the real slice values; default is 0

Out: **I**: Image

VMscatter

Plots a 3D matrix as particle model

In: **VM**: 3D Matrix to plot

Out: **h**: handle to graphics object

VMwindowplot

plots a voxel model using the window caxis fnct

In: **V**: Voxel model **cax**: [cmin cmax]; **minv**: minimum voxel value; values <0 are precentage **maxv**: maximum voxel value; values <0 are precentage

Out: **VX**: Windowed voxel image

8.5.3 VM to VL, CPL [guide]

Functions in this section:

VLofVM — returns the voxel coordinates as point list (can be long)

Detailed description

VLofVM

returns the voxel coordinates as point list (can be long)

In: **VM**: Voxel model (mxmxn) or image stack **vs**: optional voxel size

Out: **VL**: Vertex list **C**: voxel intensity values for each returned vertex

8.5.4 VM to SG Surface Model [guide]

Functions in this section:

SGofVMdelaunay — returns a surface model for a delaunay reconstructed volume model

SGofVMisosurface — returns a surface model by the marching cubes

SGofVMmarchcube — returns a surface model by the marching cubes

TR3ofVM — returns a

VLFLofVMdelaunay — returns a surface model for a delaunay reconstructed volume model

VLFLofVMmarchcube — returns a surface model by the marching cubes

VLofimage — converts an image into a contour or a solid

VMofSG — returns a voxel model of a Solid Geometry

Detailed description

SGofVMdelaunay

returns a surface model for a delaunay reconstructed volume model

In: **V**: Voxel Model (~=0 is reconstruction) **vs**: voxel size, optional **s**: step size

Out: **SG**: Solid Geometry **TR4**: Delaunay tetrahedron model

SGofVMisosurface

returns a surface model by the marching cubes

In: **V**: Logical Volume Model **vs**: voxel size, optional **cl**: closes open facets by adding empty planes; default is true **rem**: if true, remove surface edge points; default is false

Out: **SG**: Vertex list

SGofVMmarchcube

returns a surface model by the marching cubes

In: **V**: Logical Volume Model **vs**: voxel size, optional **cl**: closes open facets by adding empty planes; default is true

Out: **SG**: Vertex list

TR3ofVM

returns a

In: **V**: Voxel model **vs**: Voxel size **s**: Iso (threshold) value

Out: **TR4**: Tetrahedral (3D) triangulation **TR3**: Surface triangulation

VLFLofVMdelaunay

returns a surface model for a delaunay reconstructed volume model

In: **V**: Voxel Model (~=0 is reconstruction) **vs**: voxel size, optional **s**: step size

Out: **VL**: Vertex list **FL**: Facet list **TR4**: Delaunay tetrahedron model

VLFLofVMmarchcube

returns a surface model by the marching cubes

In: **V**: Logical Volume Model **vs**: voxel size, optional

Out: **VL**: Vertex list **FL**: Facet list

VLofimage

converts an image into a contour or a solid

In: **I**: image array **pixsize**: size of a pixel **h**: desired height; default is 0 (Contour) **nf**: frame in pixels; if nf>0 the image is impressed; default is 0

Out: **VL**: Vertex list (nx2) for h=0; (nx3) for h>0; **EL**: Edge list **FL**: Facet list **CPL**: Closed Polygon list based on contour

VMofSG

returns a voxel model of a Solid Geometry

In: **SG**: Solid Geometry **n**: Voxel size vector; default is [64 64 64] **adjxy**: adjust x and y size; default is false **shsl**: delay per slice if progress should be shown

Out: **VM**: Voxel Model **vs**: Voxel size **SGN**: Optional Solid by marching cube

9 Testing, Development, AI & Media [guide]

9.1 Sample Generation [guide]

Functions in this section:

CPLsample — returns a a closed polygon list for different tests and experiments

CSGsample — Cross-Pair Samples returns two solids as examples for crossing solids

CVLzsample — creates a CVLz sample for different Purposes

FSsample — create open surfaces (no solids) with the parameter to create missing faces etc.

PLsample — returns a 2D PL list similar to VLsample

PLsampletriangles — creates extreme triangle geometries

Posesample — creates sample poses for the pose syntheses of linkages

SGboxsphere — Solid of a box with an containing sphere, mainly used for testing

SGCLsample — returns examples for the SGofCPLcommand strings

SGcmdsample — Creates a SGcommandstring or a solid for some sample geometries

SGcolorsample — creates boxes in colors with text

SGfischertechniksample — returns SG for some Fischertechnik samples

SGfittingsample — returns a solid for fitting experiments

SGjacosample — creates magnified solids in the workspace

SGjointsample — returns Contours and cutting frames to test fnct such as CPSinserthinge, CPSinserthingelist, SGofCPLzpianohingelist

SGsample — returns a solid geometry for sample bodies used for testing and experiments

SGsample4APredator — Creates the SG for the Anycubic Z-Level testing solid

SGservosample — returns STL Solids of Servomotors

stdexplain2 — Explains the behavior of the estimation of mean and standard deviation for measurements

Tsample — shows samples of frames based on Tofn

ttestexplain — This Matlab fnct is intended to illustrate the effect of the t-test

VLFLsample — returns OPEN surfaces as samples for procedure testing

VLsample — returns a sample vertex list for testing algorithms

VLsampletriangles — returns pairs of vertex lists for triangle tests

Detailed description

CPLsample

returns a a closed polygon list for different tests and experiments

In: **Nr**: Number of sample closed polygon list or specific CPL for display **T**: Parameter for SGtrans (Value, Vector, Matrix)

Out: **CPL**: Closed Polygon List **TR3**: Optional delaunay triangulation of CPL using PLELofCPL **TX**: text or annotation data associated with the sample polygon

CSGsample

Cross-Pair Samples returns two solids as examples for crossing solids

In: **Nr**: Nr of Example **spatrel**: spatial relations such as 'ontop',-15 **"tetra"**: if used the solid get a mesh triangulation using SGtetramesh

Out: **A**: Solid A **B**: Solid A

CVLzsample

creates a CVLz sample for different Purposes

In: **Nr**: Number of **T**: optional Parameter to Transform CVLz

Out: **CVLz**: Set of planar CPLS' in different z levels

FSsample

create open surfaces (no solids) with the parameter to create missing faces etc.

In: **Nr**: Sample number **T**: optional position in space **rF**: number of missing triangles

Out: **VL**: Vertex list; or solid if nargout<2 **FL**: Facet list **SG**: Solid Geometry struct **CVL**: Opben boundary als CVL

PLsample

returns a 2D PL list similar to VLsample

In: **n**: sample of Point List

Out: **PL**: returns PL if n>0

PLsampletriangles

creates extreme triangle geometries

In: **z**: extreme dimension; default is 1e-15 **n**: number of additional random point around the triangle

Out: **PL**: point list of the sample triangle plus random points

Posesample

creates sample poses for the pose syntheses of linkages

In: **n**: selection number or string in Posesampels.mat(:,1) **l**: scaling factor applied to pose points A and B; default is 1

Out: **PS**: PS.A = list of start point; PS.B=list of end points

SGboxsphere

Solid of a box with an containing sphere, mainly used for testing

In: **S**: Diameter [x y z] **n**: number of edges in x y **ns**: number of edges of the sphere; default is empty (SGsphere default)

Out: **SG**: Solid Geometry

SGCLsample

returns examples for the SGofCPLcommand strings

In: **Nr**: Number of Sample **T**: Transformation matrix

Out: **SG**: Solid Geometry or Vertex list **FL**: Facet list **Str**: Command string

SGcmdsample

Creates a SGcommandstring or a solid for some sample geometries

In: **name**: name of solid or empty **exts**: optional extra command string appended to the sample command

Out: **SG**: command string **cmd**: solid geometry

SGcolorsample

creates boxes in colors with text

Out: **SG**: Colored Boxes

SGfischertechniksample

returns SG for some Fischertechnik samples

In: **name**: name of a solid or just empty to see all supported ones

Out: **SG**: Solid Geometry

SGfittingsample

returns a solid for fitting experiments

In: **d**: [d r1 r2 r3 ....] distance and radius

Out: **SG**: Solid Geometry

SGjacosample

creates magnified solids in the workspace

In: **n**: number of link or string of link such as 'JCF' **m**: magnification size; default is .48

Out: **SG**: Solid Geometry of the JACO robot module in magnification size m

SGjointsample

returns Contours and cutting frames to test fnct such as CPSinserthinge, CPSinserthingelist, SGofCPLzpianohingelist

In: **Nr**: list of examples to be shown; if one is selected, it will be extruded an animated; default is all **"debug"**: if used, additional steps are shown

Out: **CPL**: Contour **T**: Frames to insert cuts **SG**: Solid Geometry as Reference **wlim**: [wmin wmax] **txt**: description

SGsample

returns a solid geometry for sample bodies used for testing and experiments

In: **Nr**: Nr of sample solid; default is show **T**: Parameter for SGtrans (Value, Vector, Matrix)

Out: **SG**: Solid geometry **FL**: Facet point list wrt to SG.VL

SGsample4APredator

Creates the SG for the Anycubic Z-Level testing solid

Out: **SG**: Solid Geometry

SGservosample

returns STL Solids of Servomotors

In: **n**: number or name string **cmd**: optional command such as 'add' or 'rm' **SGN**: Servo solid to add if 'add' is used

Out: **SG**: Resulting servo-motor solid

stdexplain2

Explains the behavior of the estimation of mean and standard deviation for measurements

In: **X**: Mean Value **s**: Standard Deviation **n**: number of samples **xy**: optional measurement values size = [n x 2] **vidname**:

Tsample

shows samples of frames based on Tofn

In: **n**: if empty, the sample frame output is cleared and not returned

Out: **Ti**: last sample frame from Tofn

ttestexplain

This Matlab fnct is intended to illustrate the effect of the t-test

In: **x**: optional list of sample **X**: optional mean value of the distribution; default is calculated from x **s**: optional standard deviation value of the distribution; default is calculated from x **n**: optionla number of sample; default is calculated from x **Xr**: optional reference value ; default is none **sr**: optional deviation of the distributio; default is 0; **nr**: optional number of samples of the reference; default is infinity

Out: **f**: vector of probability density values of the distribution at the sample points **e**: vector of the cumulative distribution (integral/error) values at the sample points **cv**: two-column contour of x versus probability density used for plotting the distribution

VLFLsample

returns OPEN surfaces as samples for procedure testing

In: **Nr**: Number of sample **T**: Optional spatian transformation

Out: **VL**: Vertex List **FL**: Facet List **SG**: Solid SG.VL,SG.FL **CVL**: Open Edge contour

VLsample

returns a sample vertex list for testing algorithms

In: **Nr**: Nr of sample solid; default is none **T**: Parameter for VLtrans (Value, Vector, Matrix)

Out: **VL**: Solid geometry **PL**: Point list

VLsampletriangles

returns pairs of vertex lists for triangle tests

In: **Nr**: Number of traingle test **cmd**: such as 'last','flipA','Acirc1','Acirc2','rand','swapAB','rot'

Out: **VLA**: 3 Points of VLA or if nargout==1, [VLA;VLB] **VLB**: 3 Points of VLB if nargout>=2 **cp**: List of unique cross points if nargout>=3 **cpa**: points in A **cpb**: points in B **EL1**: Edge list of crossing constraints **EL2**: edge constraint list of triangle B crossing

9.2 Test Generation [guide]

Functions in this section:

subfig — creates subfigures on the screen similar to subplot

Detailed description

subfig

creates subfigures on the screen similar to subplot

In: **p**: position similar to subplot

9.3 SG Check [guide]

Functions in this section:

CLdecomposecontoursVLCL — decomposes missing edges from open facets of a solid

CLmissingedgesVLFL — returns missing edges of open facets

cross2F — Fundamental fnct of generating the intersection points of two triangles

crossC2P — returns how often a line crosses a contour

crossedgeVL — returns whether a 2D line crosses a 2D contour

EFLgenerate — returns an extended facet list including normal vector and size

EFLselectA — reduce the extended facet list to areas of a minimal size

FLatEL — returns all facets of the facet list attached to edges in the edge list

FLsort — Sorts FL by smallest vertex index and removes(!) degenerated facets

inside2C — analyzes the crossings of two vertex lists

intersectcontour — returns from 3 points and a intersection matrix a Vertex list and Edge list

intersectfacets — returns from 3 points and an intersection matrix of a VLFL-solid a vertex list and a facet list

intersectmatrix — returns an intersection matrix between a triangle facet and a VLFL-solid

iscollofSG — calculate collision vertices of two solids or a solid chain

SGcheckboundaryedges — returns the open boundaries of the surface

SGcheckfacedist — Checks the distance between the centers of the facets

SGcheckmeshlab — performs the most important checks for STL files

SGchecknonmanifoldedges — returns the edges and facets with non manifold edges

SGchecknonmanifoldvertices — returns the non manifold vertices, i.e. identical points used in separated surfaces

SGfixerrors — removes typical errors after surface manipulation

ULofVLFL — returns a list in the union format

VLcheckvertexaccuracy — returns the sufficient number of digits for rounding the vertex coordinates of a solid

VLELchecker — performs different tests on edge structures and displays the results

VLFLcheckboundaryedges — returns the open boundaries of the surface

VLFLchecker — performs a check of the vertex list and facet list to detect non manifold problems

VLFLcheckmeshlab — performs the most important checks for STL files

VLFLchecknonmanifoldedges — returns the edges and facets with non manifold edges

VLFLchecknonmanifoldvertices — returns the non manifold vertices, i.e. identical points used in separated surfaces

VLFLcorrectdoubledvertex — finds and removes doubled vertex entries and shows degenerated facts

VLFLcrossF — returns the intersection contours of ONE triangle with a VLFL-solid

VLFLsurface — returns a vertex index list of all vertices that belong to the same surface

VLFLsurface2 — returns a vertex index list of all vertices that belong to the same surface

VLULuniontype — analyzes the type of union from an union list

Detailed description

CLdecomposecontoursVLCL

decomposes missing edges from open facets of a solid

In: **VL**: Vertex list **AMELS**: All missing edge list sorted **CIL**: Contour index list

Out: **AMELS**: All missing edge list sorted and decomposed **CIL**: Contour index list **nme**: Number of Contours

CLmissingedgesVLFL

returns missing edges of open facets

In: **VL**: Vertex list **FL**: Facet list

Out: **MEL**: Missing edge list **CIL**: Contour index list (COL 1: start index, COL2: end index) **n**: size(CIL,1)

cross2F

Fundamental fnct of generating the intersection points of two triangles

In: **p1**: Point 1 of intersecting triangle A **p2**: Point 2 of intersecting triangle A **p3**: Point 3 of intersecting triangle A **pa**: Point 1 of intersected triangle B **pb**: Point 2 of intersected triangle B **pc**: Point 3 of intersected triangle B

Out: **t**: d1 d2 d3 da db dc n1 n2 n3 na nb nc **n**: n=0; n=-1 => planar problem; opposite facets; n=+1 => planar problem; same facets; **tx**: planar case; [x y z k1 k2 e1 ea]

crossC2P

returns how often a line crosses a contour

In: **VL**: vertex list of the contour **p1**: start point **p2**: end point

Out: **k**: number of crossings **CL**: [i index of VL, k value distance p2 to p1, k value distance VL(i+1)-VL(i) **CCL**: Cross point list

crossedgeVL

returns whether a 2D line crosses a 2D contour

In: **VL**: Vertex list **k0**: Index 0 **k1**: Index 1 **k2**: Index 2

Out: **i**: index of the beginning of the crossed line **j**: index of the end of the crossed line, ie. i+1

EFLgenerate

returns an extended facet list including normal vector and size

In: **VL**: Vertex list **FL**: Facet list **fi**: optional facet index, if only one facet is of interest

Out: **EFL**: Extended facet list

EFLselectA

reduce the extended facet list to areas of a minimal size

In: **EFL**: Extend facet list from EFLgenerate **Amin**: Minimum considered union area size

Out: **FL**: Selected facet list **RET**: Extended facet list of the selected facets **m**: number of areas **maxA**: Maximum found area size **AL**: Distribution of area size

FLatEL

returns all facets of the facet list attached to edges in the edge list

In: **FL**: Facet list **EL**: Edge list **i**: Index in edge list

Out: **AFL**: Attached facet list

FLsort

Sorts FL by smallest vertex index and removes(!) degenerated facets

In: **FL**: Original facet list

Out: **FL**: Optimized facet list **n**: length of FL **dn**: number of removed degenerated or doubled facets

inside2C

analyzes the crossings of two vertex lists

In: **VL**: Vertex list of the contour **PL**: 2nd contour to be analyzed **debug**: shows text if debug is true, default is false

Out: **X**: Result vector **CCL**: Crossing contour lines list **EPL**: Result matrix for each point of PL

intersectcontour

returns from 3 points and a intersection matrix a Vertex list and Edge list

In: **TL**: Intersection matrix of a triangle p1p2p3 **p1**: Point 1 **p2**: Point 2 **p3**: Point 3

Out: **AVL**: Vertex list of the crossing points of the triangel **AEL**: Edge list of the crossing points of the triangel

intersectfacets

returns from 3 points and an intersection matrix of a VLFL-solid a vertex list and a facet list

In: **TL**: Intersection matrix of a triangle P1P2P3 **p1**: Point 1 of triangle **p2**: Point 2 of triangle **p3**: Point 3 of triangle **VL**: Vertex list of solid **FL**: Facet list of solid

Out: **XVL**: Vertex list of the surface edges of the solid **XEL**: Edge list of the surface edges of the solid

intersectmatrix

returns an intersection matrix between a triangle facet and a VLFL-solid

In: **p1**: Point 1 of triangle **p2**: Point 2 of triangle **p3**: Point 3 of triangle **VL**: Vertex list **FL**: Facet list

Out: **TL**: intersection Matrix **cind**: logical index of the cut facets **pind**: logical index of the perforated facets **tind**: logical index of the facets fully inside the triangle

iscollofSG

calculate collision vertices of two solids or a solid chain

In: **SGA**: Solid A **SGB**: Solid B; if empty SGA is used **full**: true=full FL; false=Bounding box; default is false **firstc**: true = stops after first crossings; false=complete; default=true **shape**: if true; bounding boxe is relative to base frame; default=true

Out: **res**: List of crossing points

SGcheckboundaryedges

returns the open boundaries of the surface

In: **SG**: Solid Geometry

Out: **EL**: Edge list of open boundaries **CEL**: CEL edge list to group EL segments **fi**: facet index list of the facets **VL**: Vertex list of SG **FL**: Facet list of SG

SGcheckfacedist

Checks the distance between the centers of the facets

In: **SG**: Solid Geometry **fnum**: reduce to face numbers; default is 3000; **wlim**: warning limit; default is 0.7 mm **ilim**: intended limit; distance that is considered as intended; default is 0.1mm

Out: **wi**: complete list of distances **cpl**: complete list of cross points

SGcheckmeshlab

performs the most important checks for STL files

In: **SG**: Solid Geometry **vcor**: Flag to also check self-intersections **intl**: maximum number of facets for self-intersection tests; default is 300 **"noprint"**: flag to suppress printed output

Out: **res**: [nonmanifold-Edges nonmanifold-faces openboundary-Edges openboundary-Groups]

SGchecknonmanifoldedges

returns the edges and facets with non manifold edges

In: **SG**: Solid Geometry

Out: **NMEL**: Edge list of non manifold eges **FLi**: facet index list of problematic facets **VL**: Vertex list of SG **FL**: Facet List of SG

SGchecknonmanifoldvertices

returns the non manifold vertices, i.e. identical points used in separated surfaces

In: **SG**: Solid Geometry

Out: **vvi**: Index list of non manifold vertices / multiple used vertices **fi**: index list of faces attached to non manifold vertices **VL**: Vertex list of SG **FL**: Facet list of SG

SGfixerrors

removes typical errors after surface manipulation

In: **A**: Solid A **thr**: threshold for VLFLshort2; default is 1e-20;

Out: **A**: Solid with removed errors

ULofVLFL

returns a list in the union format

In: **VL**: Vertex list of an object **FL**: Facet list of an object **Amin**: Minimal area of union

Out: **UL**: Union list **k**: number of unions

VLcheckvertexaccuracy

returns the sufficient number of digits for rounding the vertex coordinates of a solid

In: **VLorg**: Original Vertex List

Out: **atab**: full table of significant vertex distances after rounding

VLELchecker

performs different tests on edge structures and displays the results

In: **VL**: Vertex list **EL**: Corresponding edge list **d**: Detail of protocoll; default is full **stext**: optional string for output protocoll

VLFLcheckboundaryedges

returns the open boundaries of the surface

In: **VL**: Vertex List **FL**: Facet List

Out: **EL**: Edge list of open boundaries **CEL**: CEL edge list to group EL segments **fi**: facet index list of the facets

VLFLchecker

performs a check of the vertex list and facet list to detect non manifold problems

In: **VL**: Vertex list of the object **FL**: Facet list of the object **long**: true==exact error messages; default is false

Out: **VLC**: List of sorted vertices to analyze the problematic vertices **EL**: List of sorted edges to analyze the problematic edges **PEL**: List of non-manifold edges

VLFLcheckmeshlab

performs the most important checks for STL files

In: **VL**: Vertex list **FL**: Facet list **vcor**: map to grid; default is false **intl**: number of maximum facets for self intersection tests; default is 3000 **"noprint"**: flag to suppress printed output

Out: **res**: [nonmanifold-Edges nonmanifold-faces openboundary-Edges openboundary-Groups]

VLFLchecknonmanifoldedges

returns the edges and facets with non manifold edges

In: **VL**: Vertex list **FL**: Facet list

Out: **NMEL**: Edge list of non manifold eges **FLi**: facet index list of problematic facets

VLFLchecknonmanifoldvertices

returns the non manifold vertices, i.e. identical points used in separated surfaces

In: **VL**: Vertex list **FL**: Facet list

Out: **vi**: Index list of non manifold vertices / multiple used vertices **fi**: index list of faces attached to non manifold vertices **nc**: number of separated surfaces in the solid **X**: cell list of separated surfaces

VLFLcorrectdoubledvertex

finds and removes doubled vertex entries and shows degenerated facts

In: **VL**: Vertex list **FL**: Facet list **thr**: resolution; default is 1e-12

Out: **VLN**: Shortend certex list with original, not rounded vertex values **FLN**: Corrected facet list **chngn**: number of removed vertices

VLFLcrossF

returns the intersection contours of ONE triangle with a VLFL-solid

In: **p1**: p1 of intersecting triangle **p2**: p2 of intersecting triangle **p3**: p3 of intersecting triangle **VL**: Vertex list of solid **FL**: Facet list of solid

Out: **AVL**: Vertices of the intersecting triangle/contour (p1,p2,p3) **AEL**: Edge list (contour) of the intersecting triangle/contour **XVL**: Vertices of the intersection surface contour of the VLFL-solid **XEL**: Edge list (contour) of the intersection surface contour of the VLFL-solid

VLFLsurface

returns a vertex index list of all vertices that belong to the same surface

In: **VL**: Complete vertex list **FL**: Complete facet list **V**: One vertex that is part of the surface v is element of VL

Out: **SFL**: Surface facet list **VIL**: Vertex index list **n**: Number of vertices of the surface

VLFLsurface2

returns a vertex index list of all vertices that belong to the same surface

In: **VL**: Complete vertex list **FL**: Complete facet list **V**: One vertex that is part of the surface v is element of VL

Out: **SFL**: Surface facet list **VIL**: Vertex index list **VNV**: Vertex normal vector list

VLULuniontype

analyzes the type of union from an union list

In: **VL**: Vertex list **UL**: Union list **n**: union of interest

Out: **d**: List of distances between the points of the convex hull **w**: List of angles between the points of the convex hull **da**: sorted list of the distances **wa**: sorted list of the distances

9.4 SG Repair [guide]

Functions in this section:

CVLpathwarning — warns if a path contains angles larger than pi/2

CVLzcheckprinting — checks for glued parts and missing walls

FLrepair — adds missing triangles to a facet list

PLcorrelationcheck — important fnct to detect problems that are difficult to see for a human

SGcheck4cura — Analyzes and checks Solid orientation for printing an a FDM printer

SGcheckcode — Auxiliary fnct for finding error lines

SGcheckselfintersection — checks andd returns the number of self intersecting facets

SGchecktrianglegeometry — returns the extreme values for angle, edge length and area

SGmakedoublefaces — move a number faces with slightly different position (creating errors)

SGmakedoublevertex — copies a number of vertices as doubles with slightly different position

SGmakemissingfacet — removes a number of vertices which creates missing facets

SGmakemissingvertex — removes a number of vertices which creates missing facets

SGrepairFLsimple — returns a solid with closed FL of the simpliest condition

SGrepairnonmanifoldvertices — returns a solid without non manfold vertices

SGslicercheck — checks for problems with faces and edges in the slicing plane

SGTcheckSTLwrite — A simple test fnct to double check the writing of frames into STL files

SGwatertightofSG — returns within 60 seconds a watertight reconstruction

VLcheckboundaryaccuracy — returns the sufficient number of digits for rounding to achieve minimal open boundaries

VLFLcheckselfintersection — checks andd returns the number of self intersecting facets

VLFLchecktrianglegeometry — returns the extreme values for angle, edge length and area

Detailed description

CVLpathwarning

warns if a path contains angles larger than pi/2

In: **CVL**: Closed vertex list (path) **warna**: warning angle; default is pi/2*0.99 **ignor**: if true (default), CvL is considered as open, if false start to end is also considered

Out: **wa**: true if angles exist larger thatn pi/2

CVLzcheckprinting

checks for glued parts and missing walls

In: **CVLz**: CVLz or SG to slice in mm distance **dmelt**: minimal distance to avoid melting; default is 0.45 **minw**: minimal wall thickness to avoid collaps; ; default is 0.7 **warn**: Warning flag

Out: **CVLz**: CVLz that estimates printing result

FLrepair

adds missing triangles to a facet list

In: **FL**: Facet list **VL**: Optional Vertex list **nedge**: flag to process facets along the open boundary edges; default is false

Out: **FL**: Facet list with missing triangles **nc**: number of closed triangles **no**: number of still open triangles **nl**: number of unconnected boundaries (Solid separation)

PLcorrelationcheck

important fnct to detect problems that are difficult to see for a human

In: **CL**: Correlation List of PLcorrelate **CPLA**: Contour A used in PLcorrelate **CPLB**: Contour B used in PLcorrelate **text**: optionl text for testing

Out: **err**: if true, there is an error in the Correlation list

SGcheck4cura

Analyzes and checks Solid orientation for printing an a FDM printer

In: **SG**: file name or solid geometry **turn**: if true, the part is turned; default is true

Out: **SG**: Solid Geometry **fname**: file name

SGcheckcode

Auxiliary fnct for finding error lines

In: **FN**: File Name **warnid**: search for a specific warning/error ID **depend**: search complete dependence tree; default is false

Out: **R**: Result of checkcode

SGcheckselfintersection

checks andd returns the number of self intersecting facets

In: **VL**: Vertex list or SG **csep**: separate solids before check {true,false}; default is false

Out: **n**: number of facets **d**: facets list

SGchecktrianglegeometry

returns the extreme values for angle, edge length and area

In: **SG**: Solid Geometry

Out: **wmin**: Minimal angle **emin**: Minimal edge length **Amin**: Minimul area **ai**: index of areas with minial area

SGmakedoublefaces

move a number faces with slightly different position (creating errors)

In: **SG**: Solid geometry; default is sphere (25); **fi**: indices of faces to remove; default is 1 **alpha**: alpha for separating the feature surfaces; default is .76 **d**: displacement in direction of normal vector; **"flip"**: if used; the orientation of the faces is turned to backwards **"auxc"**: create auxiliary points along the boundary edge **"auxg"**: create auxiliary grid points within the feature surface

Out: **SG**: SG with removed vertices and removed facets **SGB**: Surfaces with displacement **SGA**: Surfaces without displacement

SGmakedoublevertex

copies a number of vertices as doubles with slightly different position

In: **SG**: Solid geometry; default is sphere (25); **Vi**: indices of faces to remove; default is 1 **alpha**: alpha for separating the feature surfaces; default is .76 **d**: displacement in direction of normal vector; **"flip"**: if used; the orientation of the faces is turned to backwards **"auxc"**: create auxiliary points along the boundary edge **"auxg"**: create auxiliary grid points within the feature surface

Out: **SG**: SG with removed vertices and removed facets **SGB**: Surfaces with displacement **SGA**: Surfaces without displacement

SGmakemissingfacet

removes a number of vertices which creates missing facets

In: **SG**: Solid geometry; default is sphere (25); **fi**: indices of faces to remove; default is 1

Out: **SG**: SG with removed vertices and removed facets

SGmakemissingvertex

removes a number of vertices which creates missing facets

In: **SG**: Solid geometry; default is sphere (25); **Vi**: indices of vertices to remove; default is 1

Out: **SG**: SG with removed vertices and removed facets

SGrepairFLsimple

returns a solid with closed FL of the simpliest condition

In: **SG**: Solid Geometry SG.VL,SG.FL

Out: **SG**: Repaired Solid

SGrepairnonmanifoldvertices

returns a solid without non manfold vertices

In: **SG**: Solid Geometry

Out: **SGc**: Resulting Geometries as cells

SGslicercheck

checks for problems with faces and edges in the slicing plane

In: **SG**: Solid Geometry **z**: z value for slicing **tol**: tolerance for z coincidence; default is 1e-3

Out: **n**: [number of faces, number of edges] empty if there are no problems **F**: index list of faces **L1**: index list of edge 1-2 **L2**: index list of edge 2-3 **L3**: index list of edge 3-1

SGTcheckSTLwrite

A simple test fnct to double check the writing of frames into STL files

In: **A**: Solid to be checked

SGwatertightofSG

returns within 60 seconds a watertight reconstruction

In: **SG**: Solid Geometry

Out: **SGN**: Reconstruction Resolution 128

VLcheckboundaryaccuracy

returns the sufficient number of digits for rounding to achieve minimal open boundaries

In: **VL**: Original Vertex List or SG **FL**: Original Facet List

Out: **otab**: full table of significant vertex distances after rounding **SGN**: corrected Solid Geometry

VLFLcheckselfintersection

checks andd returns the number of self intersecting facets

In: **VL**: Vertex list or SG **FL**: Facet List **csep**: separate solids before check {true,false}; default is false

Out: **n**: number of facets **d**: facets list

VLFLchecktrianglegeometry

returns the extreme values for angle, edge length and area

In: **VL**: Vertex list **FL**: Facet list

Out: **wmin**: Minimal angle **emin**: Minimal edge length **Amin**: Minimul area **ai**: index of areas of minal area

9.5 Software Development [guide]

9.5.1 File & Folder Handling [guide]

Functions in this section:

bufferwrite — fills a buffer with byte oriented data

closeall — simply closes all windows and files

copyfilebycond — renames a file by moving it - conflict handling in case the target file exists already

desktopdir — returns the name of the desktop directory path name for PC and MAC

dir2cell — folder files as cell list

explainMfileheader — creates from a header of a m file a explain documenation file

expname — returns a filename with the current date

fcreationdate — returns the creation date for a file or file list - very slow

fcreationdate2 — 2nd version fcreationdate and much faster 2019's version

filecompareconditon — strange fnct for paring english as condition

filenameinc — increases a file name and check it's existence by option

filenamestartswithdate — returns whether a file name starts with the creation date of this file

filepartl — compiles a filename from parts similar to fileparts

ftouch — file touch on mac - use carefully - does not support wildcards but only single files

image2video — adding titles & end titles to video files

ispath — returns whether a string is part of the file path

lastuidir — returns the file parts of the last uiopend file

matlabdir — returns the default matlab directory that is also used for custom toolboxes

matlabtoolboxdir — returns the default matlab add-on toolbox root directory for all toolboxes

matrixpath — searches a path in a potential field matrix from start point to matrix value = zeros

mfile2mlxfile4markup — Converts an Markup or publishable M file into an mlx live script

movefiles — similar to movefile but works with wildcards

parsedocTL — returns the documentation string related to Tim Lueth's documentation rules

publishTL — publishes the file in the pcode directory

readtextfile — simply reads an ascii text file

renamefile — renames a file by moving it - conflict handling in case the target file exists already

repairtextfileEOF — Removes unwanted EOL char(26) from text files

SGcreatebin2mfile — reads any file, and writes a matlab fnct that can create a copy of this file

SGdimofFilename — returns the volume and size sorted dimensions from a SGarchivefilename

SGexplainfilefind — returns the files with names of type 'SGexplain_*.txt' in different return formats

SGscriptfileexecution — simplified interface for a stand alone version of the SG-Lib 5.0

SGsizeofFilename — returns the number of vertices and facets from a SGarchivefilename

SGtoolboxdir — return the path during development or after toolbox installation

showpath — Returns the search path as an n x 1 cell array.

svgpolylineofCPL — writes CPLs as polygons or polyeder lines for a laser cutter

tryuigetfile — give the user a try to select a file

webdownload — downloads a file from a url and changes the name to date and tim

writetextfile — writes a character string into a text file

Detailed description

bufferwrite

fills a buffer with byte oriented data

In: **B**: Buffer **p**: Pointer to last used byte

Out: **B**: Buffer **p**: Pointer to last used byte

closeall

simply closes all windows and files

copyfilebycond

renames a file by moving it - conflict handling in case the target file exists already

In: **name**: Original Name **newname**: Expected New name **condition**: default is "not older and not smaller' **"silent"**: if given as 'silent', suppresses output messages

Out: **overw**: true if rename/overwrite was sucessfull

desktopdir

returns the name of the desktop directory path name for PC and MAC

In: **Fname**: optional filename relative to desktopdir **cp**: copy as desktopdir; 'set' to set or 'reset' to reset

Out: **dname**: desktop directory path name

dir2cell

folder files as cell list

In: **dd**: result of a dir command

Out: **ddl**: cell list of file names

explainMfileheader

creates from a header of a m file a explain documenation file

In: **funcname**: Name of calling function whose % header contains explain text **keyword**: keyword or 'fdelete'

Out: **textname**: file name of the created or used documentation file

expname

returns a filename with the current date

In: **pfix**: prefix default is 'EXP-'

Out: **fname**: file or directory name

fcreationdate

returns the creation date for a file or file list - very slow

In: **fname**: file name or file wildcard **silent**: if true, suppress progress output; default is false

Out: **crdat**: creation date or cell list

fcreationdate2

2nd version fcreationdate and much faster 2019's version

In: **fname**: File name

Out: **fList**: List of files with creation dates

filecomparecondition

strange fnct for paring english as condition

In: **inf1**: information struct for source **inf2**: information struct for target **text**: text condition; default is 'newer and not smaller'

Out: **logic**: true if condition fulfilled **text**: final evaluated text

filenameinc

increases a file name and check it's existence by option

In: **fname**: file name **step**: step size; default is +1 **fchk**: check if file or folder; default is true **srch**: default is false; try to increase the file name **dchn**: increase also all file names in folder names; default is true;

Out: **fni**: Incremented file name **nstr**: incremented numeric string part of the file name

filenamestartswithdate

returns whether a file name starts with the creation date of this file

In: **fullname**: full file name

Out: **x**: [1 1 1] if this is correct

filepartl

compiles a filename from parts similar to fileparts

In: **fname**: original file name **fmt**: 'e' if extension should be added; 'p' if path should be added

Out: **fn**: filename

ftouch

file touch on mac - use carefully - does not support wildcards but only single files

In: **fname**: exact filename

image2video

adding titles & end titles to video files

In: **fname**: file name

ispath

returns whether a string is part of the file path

In: **npath**: name of path

Out: **l**: true or false

lastuidir

returns the file parts of the last uiopend file

Out: **fp**: file path plus filesep **fn**: file name **fe**: file extension

matlabdir

returns the default matlab directory that is also used for custom toolboxes

Out: **mldir**: default user matlab directory

matlabtoolboxdir

returns the default matlab add-on toolbox root directory for all toolboxes

Out: **mldir**: default user matlab add-on toolbox directory

matrixpath

searches a path in a potential field matrix from start point to matrix value = zeros

In: **M**: Matrix as result of **GL**: Goal start tart search from **AP**: Attraction point; default is Matrix==0 **ST**: Style of Line "push', 'pull' , 'dist'; default is pull

Out: **PL**: PLrc list from start to end

mfile2mlxfile4markup

Converts an Markup or publishable M file into an mlx live script

In: **fname**: matlab m-file within the search path **"force"**: **"help"**: if used, the mlx file will be overwritten anyway **"delete"**: if used, the final html file is on TL's developer disk SGLIB_HELP_Structures/html/ **"gpt"**:

movefiles

similar to movefile but works with wildcards

In: **Oname**: Source file list **Tname**: Target file list or directory

parsedocTL

returns the documentation string related to Tim Lueth's documentation rules

In: **fname**: File name

Out: **Dclass**: Documentation information

publishTL

publishes the file in the pcode directory

In: **fname**: file name without '.m'

Out: **err**: error message if publishing failed; empty otherwise

readtextfile

simply reads an ascii text file

In: **fname**: file name **tsize**: numer of characters to read from the file start **lines**: if tsize=" the following lines are extracted **"UTF"**: if 'UTF-8' is used, an "UTF-16" (Matlab 2024b) is returned as UTF-8

Out: **TEXT**: ASCII content uin8 / char **s**: separator by newlin **fname**: filename

renamefile

renames a file by moving it - conflict handling in case the target file exists already

In: **name**: Original Name **newname**: Expected New name **condition**: default is "not older and not smaller'

Out: **overw**: true if rename/overwrite was sucessfull

repairtextfileEOF

Removes unwanted EOL char(26) from text files

In: **fname**: full filepath **remchar**: character code to remove; default is 26 (EOF)

SGcreatebin2mfile

reads any file, and writes a matlab fnct that can create a copy of this file

In: **binfile**: name of a binay file on disk

SGdimofFilename

returns the volume and size sorted dimensions from a SGarchivefilename

In: **fn**: filename in directory found by search

Out: **nvf**: returns the volume and size sorted dimensions is containing this information

SGexplainfilefind

returns the files with names of type 'SGexplain_*.txt' in different return formats

In: **typ**: 'file','string','comma','hyper'; default is 'string'

Out: **b**: list of files

SGscriptfileexecution

simplified interface for a stand alone version of the SG-Lib 5.0

SGsizeofFilename

returns the number of vertices and facets from a SGarchivefilename

In: **fn**: filename in directory found by search

Out: **nvf**: returns the V+ and F= values if the filename is containing this information

SGtoolboxdir

return the path during development or after toolbox installation

Out: **tbdir**: toolboxdir

showpath

Returns the search path as an n x 1 cell array.

In: **nn**: cell array index

Out: **ppp**: [n x 1] cell array of all path or the selected entries

svgpolylineofCPL

writes CPLs as polygons or polyeder lines for a laser cutter

In: **fid**: File Identifyer **CPL**: Close Contour List **stroke**: red, blue, etc. **width**: 1, 3 **fill**: none, or red, blue, etc.

tryuigetfile

give the user a try to select a file

In: **xy**: numerical parameter, empty or existing filename **fselect**: such as '*.avi;*.m4v;*.mp4;*.mov;*.mpg'; default is *.*

Out: **xy**: numerical parameter, empty or existing filename

webdownload

downloads a file from a url and changes the name to date and tim

In: **url**: complete url to download the file **FNAME**: new file name; default is datestr; false=> original name

Out: **fname**: file name on your file system **orgfname**: original file name **data**: data from url source

writetextfile

writes a character string into a text file

In: **fname**: filenname for desktop or full path **txt**: ascii text

Out: **wfname**: full filename

9.5.2 System, macOS & Hardware Tools [guide]

Functions in this section:

AVRarduinoMega — just opens the website of this company

cputic — similar to tic but uses the cpu timer

cputoc — similar to toc but uses the cpu timer

Dashboard — just opens the web side for formlab dashboard

gcsb — get current system's block

homebrewtest — Simply tests the existence of homebew on intel and silicon macs and sets the path

JAVAemailmsg2cell — reads email as text or creates an chatGPT answer

JAVAemailServerImap — opens a JAVA mail server account

JAVAemailTUMimap — opens a JAVA imap mail server account with certificate for mail.tum.de

openbydoubleclick — opens/starts a file by the operating system

osx — simplified version that shows osx version matlab version and sg lib version

pingIP — sends a single ping within one second to check the reachability of FTP server

RRcputime — returns realtime difference and cputime difference since first call

RRhelp — returns the help text for the Round-Robin realtime-environment

RRkeyboardCallback — this callback fnct collects keyboard character and stores it into two global variables

RRrun — starts Round-Robin Task-List Realtime Environment in stopped state

RRshell — Runs an interpreter shell for a limited period of time to simulate a realtime system

SGrobotarduinoIDE — downloads and installs the arduino IDE for programming the LCL robot of TUM/MIMED

verJAVA — fnct to check the JAVA Version of Matlab and OSX

verOSX — just tired to find out why and when matlab chashes

workdir — changes to the working directory and/or defines it

Detailed description

AVRarduinoMega

just opens the website of this company

In: **text**: string for search

cputic

similar to tic but uses the cpu timer

cputoc

similar to toc but uses the cpu timer

Out: **tc**: Elapsed cpu time in seconds

Dashboard

just opens the web side for formlab dashboard

gcsb

get current system's block

In: **bl**: block; default is gcb

Out: **s**: name of block

homebrewtest

Simply tests the existence of homebew on intel and silicon macs and sets the path

Out: **hbp**: path to the Homebrew installation, or empty if not found

JAVAemailmsg2cell

reads email as text or creates an chatGPT answer

In: **msgs**: array of javax.mail messages to parse

Out: **ttxt**: email body text; a cell array of all messages if more than one msg **Efrom**: sender email address **Edate**: received date string **Esubj**: email subject **Eto**: recipient address(es) **ANSW**:

JAVAemailServerImap

opens a JAVA mail server account

In: **account**: such as 'tim.lueth@ergosurg.com' **FOI**: Folder of interest; default is 'INBOX' **passw**: **server**:

Out: **msgs**: ALL Messages of the FOLDER **store**: connection to the server

JAVAemailTUMimap

opens a JAVA imap mail server account with certificate for mail.tum.de

In: **account**: default is gu28kix **FOI**: Folder of interest; default is 'INBOX' **passw**: password for email account. No F2A support

Out: **msgs**: ALL Messages of the FOLDER **store**: connection to the server

openbydoubleclick

opens/starts a file by the operating system

In: **fname**: Filename; default is in the desktopdir **aname**: application name **nn**: selector of fname, if fname is a cell list **ml**: if true; the focus switches back to matlab; default is false!

osx

simplified version that shows osx version matlab version and sg lib version

Out: **osx**: OSX Release | Matlab Release | SG-Lib Release

pingIP

sends a single ping within one second to check the reachability of FTP server

In: **IP**: IP address **tt**: timeout time; default is 1 second

Out: **isreachable**: true if a response was found

RRcputime

returns realtime difference and cputime difference since first call

In: **RRtictime**: only used for reset by tic

Out: **t**: time difference since first call **tc**: CPUtime difference since first call

RRhelp

returns the help text for the Round-Robin realtime-environment

RRkeyboardCallback

this callback fnct collects keyboard character and stores it into two global variables

In: **src**: source object of the callback (figure handle) **event**: keyboard event data with Character, Key, and Modifier fields

RRrun

starts Round-Robin Task-List Realtime Environment in stopped state

In: **RRtasklist.t0**: Clock cycle time; default is 0.1 **RRmaxtime**: maximum duration time; default is 600 seconds

RRshell

Runs an interpreter shell for a limited period of time to simulate a realtime system

SGrobotarduinoIDE

downloads and installs the arduino IDE for programming the LCL robot of TUM/MIMED

In: **sources**: if true; the web pages are opened to the original sources

verJAVA

fnct to check the JAVA Version of Matlab and OSX

Out: **varargout**: MATLAB's Java version string (char of mlj.Version)

verOSX

just tired to find out why and when matlab chashes

Out: **osxs**: osx version **mlw**: warning on matlab stability

workdir

changes to the working directory and/or defines it

In: **wd**: working directory path

Out: **rwd**: **wdtext**:

9.5.3 Toolbox Build & Documentation [guide]

Functions in this section:

clipboardadd — adds comment characters from the clipboard

clipboardrem — removes comment characters from the clipboard

depuseString — searches for string using the system command grep

evalTLvars — modifies variables of a calling fnct

helptextofcaller — returns the helptext of the calling fnct

isatleastVer — checks the minimal version/release number of matlab

publishLaTeX — displays on screen a LaTeX formula

publishSGPDF — just publishes the VLFL_EXP## files to save time in the current directory

SGexplain — returs help texts similar to a wiki system

SGlibedithtmlhelptext — open the html help text in the 'SGLIB_HELP_Structures/html' folder

SGlibHelpTutorials — A feature for tutorial authors that provides hyperlinks to display and open existing tutorials.

SGlibmovepathtotop — moves the SG-Lib directory up in the search path

SGlibpath — returns the filepath of the toolbox directory

SGoutdated — Execution stops by reporting that the fnct is outdated

verML — returns the currently used Matlab Release

Detailed description

clipboardadd

adds comment characters from the clipboard

In: **maxc**: Maximum comment column width

Out: **C**: Resulting clipboard text with comment characters added

clipboardrem

removes comment characters from the clipboard

In: **maxc**: Maximum comment column width

Out: **C**: Resulting clipboard text with comment characters removed

depuseString

searches for string using the system command grep

In: **sstr**: search string **sstr**: if true; comment lines s are ignored; default is false **greps**: default is *.m **"sortbydate"**: if used, the output is sorted by creation date

Out: **l**: list of lines in the files of the current dirctory **f**: list of mfiles that contains the string

evalTLvars

modifies variables of a calling fnct

In: **DEFPAR**: Default variable values **TLPAR**: Parameter string (varargins) to overload the default values

Out: **TLPAR**: combined parameter string of the applied variable assignments

helptextofcaller

returns the helptext of the calling fnct

Out: **htext**: help text

isatleastVer

checks the minimal version/release number of matlab

In: **name**: Release name 'R2017a' **TB**: Toolbox name ;default is 'Matlab'

Out: **is**: true or false

publishLaTeX

displays on screen a LaTeX formula

In: **str**: String that contains the formula

publishSGPDF

just publishes the VLFL_EXP## files to save time in the current directory

In: **nrs**: sequence of numbers; default is 1:80

SGexplain

returs help texts similar to a wiki system

In: **word**: name of a search text **parm**: additional parameter such as 'edit' **fname**: File name for database; default is 'SGexplainDocumentation.txt'

SGlibedithtmlhelptext

open the html help text in the 'SGLIB_HELP_Structures/html' folder

In: **Fname**: Name of function

SGlibHelpTutorials

A feature for tutorial authors that provides hyperlinks to display and open existing tutorials.

Out: **llc**: cell array of MATLAB edit hyperlinks to the found HELP_SGLIB_*.m tutorial files

SGlibmovepathtotop

moves the SG-Lib directory up in the search path

SGlibpath

returns the filepath of the toolbox directory

Out: **fpath**: filepath without filesep

SGoutdated

Execution stops by reporting that the fnct is outdated

In: **datestr**: optional datestr to show when it was marked as outdated **altstr**: optional recommendation for replacement

verML

returns the currently used Matlab Release

In: **name**: Tool to search for

Out: **v**: RElease such as 'R2017a'; empty if missing;

9.5.4 Debugging [guide]

Functions in this section:

dbcheck — checks whether a variable is stored in global workspace and listed in SGvar

dbload — imports a workspace varibale into a fnct

dbpause — similar to pause, but shows the calling fnct and if dbprintmode is on, jumps to the line

dbplot — plots a SG objects (SG, PL, CPL, CVL, etc.) for debug purposes

dbprintflog — fnct that logs similar to former frintf(2,<args>) a text tino a log file

dbprintftext — simplified text output in debug scenarios

dbprintfwarning — Like printf but prints a warning showing the calling fnct, optionally as a hyperlink

dbprintmode — switches on the possibility to reach code lines when using dbprintf fnct

dbreturn — this is a fnct Mathworks should implement for debugging reasons

dbsave — writes for debugging variables into the global workspace

Detailed description

dbcheck

checks whether a variable is stored in global workspace and listed in SGvar

In: **vari**: variable **val**: optional value

Out: **ri**: true if variable is unchanged or not exisiting; false if different

dbload

imports a workspace varibale into a fnct

In: **var2base**: string list of variables to be read in from the global workspace

dbpause

similar to pause, but shows the calling fnct and if dbprintmode is on, jumps to the line

In: **str**: string, if there is no time, or zero or seconds **str2**: prompt if a time is used

Out: **ch**: key character that was used in the graphics figure to continue after pause

dbplot

plots a SG objects (SG, PL, CPL, CVL, etc.) for debug purposes

In: **SG**: List of SG,CVL,CPL etc **"asccw"**: plot all contoure showing the orientation cw/ccw **"asPL"**: plot contours as PL instead of CPLs **"cont"**: continue without pause after plotting **"nosave"**: don't save the variable as normal **"sub"**: use subplots instead of a single figure **"sep"**: show the variable names **"stop"**: 'names' flag to also plot the names

dbprintflog

fnct that logs similar to former frintf(2,<args>) a text tino a log file

In: **argslist**: liat of arguments equal to to sprintf

dbprintftext

simplified text output in debug scenarios

In: **var2base**: list of parameters with text and variables

dbprintfwarning

Like printf but prints a warning showing the calling fnct, optionally as a hyperlink

In: **var2base**: printf-style format string and arguments

dbprintmode

switches on the possibility to reach code lines when using dbprintf fnct

In: **md**: true or false; default is true;

Out: **dbmode**: true or false

dbreturn

this is a fnct Mathworks should implement for debugging reasons

In: **var2base**: list of variables that should be in the workspace after return

dbsave

writes for debugging variables into the global workspace

In: **var2base**: list of variables that should be in the workspace after return **"verbose"**: if given as 'verbose', suppresses the modification messages

9.5.5 String, Cell & Data Utilities [guide]

Functions in this section:

array2cell — convert object arrays into cell arrays

array2tableTL — simple auxiliary fnct for debugging

arrayof — returns an array of a class object

AWG2mm — converts American Wire Gauge to Wire Diameter

cell2cellarray — creates a cell list that contains all elements of a cell within one cell array

cell2class — converts a cell list of elements of the same class into an array of this class

cell2matdim2 — converts a cell list of frames into an array format

cell2matdim3 — converts a cell list of frames into an array format

cellsize — returns the size of cell elements

cellstradjustlen — adjust in each col of acell array or table the length of char string

cellstrfind — returns the position of a string in a cell list of string

celltype — creates any type check for cells - Missing matlab fnct or at least unknown to me

Charriere2mm — returns the outer diameter in mm of a cather in the Charriere scale

class2cell — converts an array of a class into a cell list of the same type

code3Byte2single — converts 3 Bytes plus 8 Bit code into a single

containscell — finds a string segment into a cell list of strings

datenumarray — creates a for time rows of financial data a vector of datenum dates

datestrroman — returns a datestring as roman number

digitofnum — returns the digit at a specific position

erfGauss — returns the error fnct for the Gaussian distribution

erfinvGauss — returns the inverse error fnct for the Gaussian distribution

Fahrenheit2Celsius — converts temperature values from Fahrenheit to Celsius

French2mm — returns the outer diameter in mm of a cather in the Charriere scale

Gauge2mm — returns the outer diameter in mm of a Cather in Gauge scale

indexvariant — creates multidimensional index sequences for testing or other purposes

indofrc — converts an index list into a rows cols list

isemptycell — similar to isempty but works with cells

ismemberincell — finds the logical indices of an member inside a cell list of DIFFERENT types

ismonotonously — returns whether a sequence in monotonously increasing or decreasing

KW2PS — returns Horse Power from KW

lengthNaN — returns the length of the individual entries of a nan separated list

limitinterv — value between [min and max]

linearinterp — linear interpolation

materialparameters — Hyperlink fnct to call up material parameters WWW pages in the browser

matrixdistancefield — calculates a matrix distancefield based on a start point and optional end point

matrixfillborder — fills a matrix similar as matrixdistanceofCPL but with discrete values

matrixfilldistance — fills a potential field

matrixfillseedpoints — fills a matrix similar as matrixdistanceofCPL but with discrete values

matrixlocalminmax — returns the local Minima and Maxima of a mtrix

matrixneighbors — returns the coordinates for matrix neighbors

matrixPLplot — plots a matrix and a point list

matrixrand — returns random values for coordinates of a matrix

matrixskeleton — creates the skeleton path for a matrix

matrixvalues — returns the values of a matrix from a list of coordinate entries

meanGauss — returns mean and standard deviation wrt accuracy aspects

mm2French — converts Diameters in mm in French units

mm2inch — returns mm in inch

mod1circsequ — returns a forwared sequence from start to end and a backward sequence from start to end

number2vec — converts a number into vector of integer

numofstrtab — returns a row number of a cell list of strings

parsedatestr — parse a datestr and returns the corresponding datenum and datestr

parts2single — creates a single float from a int exponent (0 255) and a 24 Bit integer mantissa

PS2KW — returns KW from Horse Power

randdice — same as rand but only numbers between 1 and 6

randminv — returns a random fnct that has a minimal value

randnorm — same as normrnd but a different argument more than rand

randol — return an random number with at least a minimum value unit the next perdecimal

randT — return a random T matrix

rcofind — converts an index list into a rows cols list

repmatfill — fills an array with values or shortens the array

retainfields — retains a list of fields from a struct

risemonotonously — modified jumps of 2pi into a monotonously rise sequence

roundgauss — returns a mean value and standard deviation in an accuracy wrt to the sample number N

scalesequence — returns a streched sequence between an minimal and maximal value

single2parts — returns the parts of a 32 Bit float

stripfields — removes a list of fields from a struct

strrepmatlab2ascii — converts text lines from the matlab editor to ascii text

strrepn — replaces several string elements by using strrep

ticksofminmax — returns a list of unique increasing values between bb(1) and bb(2)

uniquecell — unique command for cells

uniqueNaN — finds copies of lists such as CPL with in a nan separted list

unsortNaN — change the order the entries of a NaN separted list

Detailed description

array2cell

convert object arrays into cell arrays

In: **array**: arrays of structs

Out: **c**: cell list of input values

array2tableTL

simple auxiliary fnct for debugging

In: **AR**: array **VariableNames**: list of variabale names for cols; if 1st element is true => rownames

Out: **ART**: Table

arrayof

returns an array of a class object

In: **cobj**: class object **r**: rows **c**: cols

Out: **clarr**: array of class objects

AWG2mm

converts American Wire Gauge to Wire Diameter

In: **awg**: AWG **len**: optional cable length in METER **Ucc**: optional cable voltage **Isc**: optional cable current

Out: **dmm**: Wire diameter

cell2cellarray

creates a cell list that contains all elements of a cell within one cell array

In: **A**: Hieraarchical cells

Out: **X**: [1 x n] cell array

cell2class

converts a cell list of elements of the same class into an array of this class

In: **cells**: cell list of any class type

Out: **x**: resulting array of the class type

cell2matdim2

converts a cell list of frames into an array format

In: **TL**: Cell list of frames

Out: **TLA**: Array format of Frames

cell2matdim3

converts a cell list of frames into an array format

In: **TL**: Cell list of frames

Out: **TLA**: Array format of Frames

cellsize

returns the size of cell elements

In: **C**: cell array

Out: **cs**: size of array

cellstradjustlen

adjust in each col of acell array or table the length of char string

In: **OCC**: Original cell array or table

Out: **CC**: cell array or table in which all strings of a column have the same length

cellstrfind

returns the position of a string in a cell list of string

In: **cellstr**: cellstr **str**: search string

Out: **f**: array of position for each cell elememt

celltype

creates any type check for cells - Missing matlab fnct or at least unknown to me

In: **C**: Cell ist **tppe**: type tu check such as @ischar or @isnumeric

Out: **a**: logical array, true where the cell element passes the type check

Charriere2mm

returns the outer diameter in mm of a cather in the Charriere scale

In: **cha**: Catheter size in Charriere

Out: **mm**: Outer diameter in mm

class2cell

converts an array of a class into a cell list of the same type

In: **objs**: object array of a class

Out: **c**: cell list of the class array

code3Byte2single

converts 3 Bytes plus 8 Bit code into a single

In: **e**: code/exponent (no 0, 126, 127) **u**: 3 Bytes uint8

Out: **k**: single precision 4 Bytes coded

containscell

finds a string segment into a cell list of strings

In: **A**: Cell List of Strings **sstr**: string to search **IgnoreCase**: if true; case is ignored

Out: **ind**: index list

datenumarray

creates a for time rows of financial data a vector of datenum dates

In: **dateS**: Start date **dateE**: End date

Out: **datevec**: Array of datetime values

datestrroman

returns a datestring as roman number

In: **S**: datestr YYYY-MM-DD; default is datestr

Out: **S**: roman number string

digitofnum

returns the digit at a specific position

In: **n**: number **d**: digit of number

Out: **z**: digit value

erfGauss

returns the error fnct for the Gaussian distribution

In: **x**: normalized t value (1 = 1 sigma) (X-Xref)/s

Out: **y**: error probability (for negative values)

erfinvGauss

returns the inverse error fnct for the Gaussian distribution

In: **x**: probability

Out: **y**: sigma value

Fahrenheit2Celsius

converts temperature values from Fahrenheit to Celsius

In: **fh**: temperature in Fahrenheit

Out: **cel**: temperature in Celsius/Kelvin

French2mm

returns the outer diameter in mm of a cather in the Charriere scale

In: **cha**: Catheter size in Charriere

Out: **mm**: Outer diameter in mm

Gauge2mm

returns the outer diameter in mm of a Cather in Gauge scale

In: **gau**: Gauge Value

Out: **mms**: Outer diamter in mm

indexvariant

creates multidimensional index sequences for testing or other purposes

In: **vec**: row or col vector; default is [1 2 3] **dim**: integer as dimension; default is 2

Out: **zz**: index field [numel(vec)^dim x dim]

indofrc

converts an index list into a rows cols list

In: **Ms**: size of matrix **PL**: rc list

Out: **ind**: index list

isemptycell

similar to isempty but works with cells

In: **A**: cell

Out: **e**: logical list

ismemberincell

finds the logical indices of an member inside a cell list of DIFFERENT types

In: **C**: Cell list **c**: Single entry to find **csens**: character sensitive; default is false **fullc**: full compare instead of starting chars; default is false **backw**: if true, entries are flipped left-right before testing; default is false **conta**: if true, uses contains instead of startsWith for char comparison; default is false

Out: **a**: logical field

ismonotonously

returns whether a sequence in monotonously increasing or decreasing

In: **x**: sequence [n x 1] or [1 x n] **circ**: if true; x is a circular sequence with one jump; default is false

Out: **mon**: 1= icreasing -1 => decreasing 0 == not monotonously **fd**: indices of the faulty values **k**: -1 = mainly decreasing, +1 = mainly increasing

KW2PS

returns Horse Power from KW

In: **kw**: SI unit for power in KW

Out: **ps**: horse power value

lengthNaN

returns the length of the individual entries of a nan separated list

In: **CPL**: nan separated list

Out: **l**: length of the individual segments of the nan separated list **a**: starting indices

limitinterv

value between [min and max]

In: **x**: number **xlim**: [min max]

Out: **y**: value between [min and max]

linearinterp

linear interpolation

In: **tab**: table or array **x**: [n x 1] row list of sample or a single value **colum**: column to use for interpolation of x

Out: **y**: complete table row for interpolated x

materialparameters

Hyperlink fnct to call up material parameters WWW pages in the browser

In: **text**: keywords

matrixdistancefield

calculates a matrix distancefield based on a start point and optional end point

In: **M**: Matrix **st**: Start points of value zeros

Out: **DM**: Matrix-Distancefield

matrixfillborder

fills a matrix similar as matrixdistanceofCPL but with discrete values

In: **M**: Matrix with nan as untouchable regions **neighb**: '8' oder '4' as neighborhood; default is '8'

Out: **MM**: Matrix with discrete distance values

matrixfilldistance

fills a potential field

In: **M**: Matrix **lst**: start point list

Out: **MM**: Distancematrix

matrixfillseedpoints

fills a matrix similar as matrixdistanceofCPL but with discrete values

In: **M**: Matrix with nan as untouchable regions **neighb**: '8' oder '4' as neighborhood; default is '8' **PLrc**: List of Start points using PLrc defining the

Out: **MM**: Matrix with discrete distance values

matrixlocalminmax

returns the local Minima and Maxima of a mtrix

In: **M**: Original Matrix **ELtyp**: default is 'max' **Neigb**: neighborhood type '4' or '8'; default is '4'

Out: **Min**: List of Minima **Max**: List of Maxima **MM**: Full minimum maximum matrix **EL**: EL list containing ELtype of ELofmatrixneigborweight

matrixneighbors

returns the coordinates for matrix neighbors

In: **Ms**: size(Matrix) **RCL**: [row col] coordinate **masktyp**: '4' or '8' as mask type **noncirc**: noncircular or circular neighborhood; default is true

Out: **RCN**: row cols neighbors; nan for non existing

matrixPLplot

plots a matrix and a point list

In: **M**: Matrix entries **PL**: [r c] or [y x] list

Out: **h**: handle to graphics objects

matrixrand

returns random values for coordinates of a matrix

In: **M**: Matrix **n**: number of random coordianates

Out: **PLrc**: coordinate list for matrix M

matrixskeleton

creates the skeleton path for a matrix

In: **M**: Matrix **distf**:

Out: **SEL**: Skeleton Edge List

matrixvalues

returns the values of a matrix from a list of coordinate entries

In: **M**: Matrix **el**: list of [r c] coordinates

Out: **Res**: Result

meanGauss

returns mean and standard deviation wrt accuracy aspects

In: **x**: Test (sample) List **Xref**: optional reference value for t-test

Out: **X**: Mean Value **ss**: Standard deviation **sx**: Standard deviation of X **se**: error value for deviation

mm2French

converts Diameters in mm in French units

In: **mm**: diameter (!) in mm

Out: **cha**: diameter in french

mm2inch

returns mm in inch

In: **mm**: mm

Out: **inch**: Outer diameter in mm

mod1circsequ

returns a forwared sequence from start to end and a backward sequence from start to end

In: **s**: start index **e**: end index **n**: number of points in the list [1:n]

Out: **forw**: [s .. e] always increasing numbers **back**: [s .. e] always decreasing numbers

number2vec

converts a number into vector of integer

In: **s**: string or integer scalar

Out: **vec**: vector if integer

numofstrtab

returns a row number of a cell list of strings

In: **tok**: string to compare with; **strtab**: cell list; frist col = string

Out: **nr**: cell list row, col 2: end

parsedatestr

parse a datestr and returns the corresponding datenum and datestr

In: **DStr**: Datestr in unknown format **nf**: format, default is 29

Out: **dn**: datenum **ds**: datenum

parts2single

creates a single float from a int exponent (0 255) and a 24 Bit integer mantissa

In: **e**: exponent/code **m**: 24 Bit integer

Out: **k**: single precision float

PS2KW

returns KW from Horse Power

In: **ps**: horse power value

Out: **kw**: SI unit for power in KW

randdice

same as rand but only numbers between 1 and 6

Out: **rd**: equal distributed random numbers between 1 6

randminv

returns a random fnct that has a minimal value

In: **r**: rows **c**: cols **rmin**: minimal value or [rmin rmax]; default is [rmin 1]

Out: **r**: random matrix

randnorm

same as normrnd but a different argument more than rand

In: **r**: rows **c**: cols; default is r **m**: mean **s**: standard deviation

Out: **y**: random numbers

randol

return an random number with at least a minimum value unit the next perdecimal

In: **r**: rows or minimum number **c**: col **rmin**: minimum number; default is 0.01

Out: **r**: randum number, or matrix

randT

return a random T matrix

In: **d**: distance from original; default is 10..20

Out: **T**: HT Matrix

rcofind

converts an index list into a rows cols list

In: **s**: size of matrix **il**: index list

Out: **RCL**: Roews cols list

repmatfill

fills an array with values or shortens the array

In: **line**: [1 x m] array with values **n**: desired length **val**: value for fill up; default is line (end)

Out: **line**: resulting line with length n

retainfields

retains a list of fields from a struct

In: **S**: struct **L**: List of field names as varargin

Out: **S**: struct with removed fields

risemonotonously

modified jumps of 2pi into a monotonously rise sequence

In: **v**: angle vector **m**: offset to jump; default is 2*pi

Out: **v**: angle vector with values outside the interval 2*pi

roundgauss

returns a mean value and standard deviation in an accuracy wrt to the sample number N

In: **x**: Mean Value **s**: Standard deviation **n**: number of measurement; if 1 we use 2 significant numbers

Out: **xr**: mean rounded **sr**: standard deviation rounded

scalesequence

returns a streched sequence between an minimal and maximal value

In: **x**: value sequence - only first and last values are used **n**: number of desired values **usemean**: use mean value if n==1 and numel(unique(x))>1; default is false

Out: **y**: new sequence as [1 x n] row vector

single2parts

returns the parts of a 32 Bit float

In: **k**: single precision number

Out: **e**: 8 Bit Exponent mantissa (>1 ==128..255, <1 == 0..128) **m**: 23 Bit fraction mantissa **s**: sign; 0==1positive 1==negative **ub**: 3 uint8 Bytes (sign plus mantissa, 24 Bit)

stripfields

removes a list of fields from a struct

In: **S**: struct **Fieldnames**: List of field names as varargin

Out: **S**: struct with removed fields

strrepmatlab2ascii

converts text lines from the matlab editor to ascii text

In: **lin**: a text line

Out: **lin**: textline with exchanged characters

strrepn

replaces several string elements by using strrep

In: **origStr**: Original String **oldSubStr**: Substring to exchange **newSubstr**: New Substring

Out: **S**: Resulting string

ticksofminmax

returns a list of unique increasing values between bb(1) and bb(2)

In: **bb**: [min max] **res**: distance between ticks **det**: resolution default is .1

Out: **tck**: unique increasing values between bb(1) and bb(2)

uniquecell

unique command for cells

In: **C**: cell list

Out: **C**: cell list with unique elements **ii**:

uniqueNaN

finds copies of lists such as CPL with in a nan separted list

In: **CPL**: Original NaN sperated list

Out: **CPLU**: Reduced Nan separted list with unique entries **ai**: indes list for

unsortNaN

change the order the entries of a NaN separted list

In: **CPL**: Original nan separated list

Out: **CPLU**: Unsorted nan separated list

9.5.6 Media & Document Files [guide]

Functions in this section:

bambudownloadvideoIP — downloads in parallel to printing processes the timelapse videos from the bamboo printer

Videoquickmovecursor — Moves a cursor symbol on the current axes to a specified point for video recording

Detailed description

bambudownloadvideoIP

downloads in parallel to printing processes the timelapse videos from the bamboo printer

In: **aname**: local folder name to save the bambu's timelapse folder **IP**: default is '129.187.200.238'

Videoquickmovecursor

Moves a cursor symbol on the current axes to a specified point for video recording

In: **p**: target point coordinates (2D or 3D) **n**: number of frames for the move in drawnowvid; default is 20 (1 second) **col**: color of the cursor symbol; default is 'k'

9.5.7 Web & Company Data [guide]

Functions in this section:

amazon — just opens the website of this company

amazonparts — just opens the website of this company

DeepL — just opens the browser with the DeepL translation tool

find2logical — converts a index list into a logical vector

FINdataM2EURread — delivers the EUR money supply M2 directly from European Central Bank apir

FINdataM2USAread — delivers the USD money supply M2 directly from FRED in St. Louis, USA, via API

FINdataPOP — downloads and saves the population of several countries from the world bank homepage

finddoubledrows — returns the indices of the lines with copies of a earlier row

finduniqueVL — similar to unique but more reliable

loadweb — loads mat files from an web url

SGgrabcad — Simply opens the WWW-page of grabcad.com in the system browser

Detailed description

amazon

just opens the website of this company

In: **searchterms**: string for search

amazonparts

just opens the website of this company

In: **text**: string for search

DeepL

just opens the browser with the DeepL translation tool

In: **req**: text string to translate; German umlauts are replaced before URL encoding

find2logical

converts a index list into a logical vector

In: **ind**: index list **N**: numel of logical row vector

Out: **l**: logical row vector

FINdataM2EURread

delivers the EUR money supply M2 directly from European Central Bank apir

Out: **M2EUR**: [datum values] as coordinate list **dates**: datenum array **values**: vallue array

FINdataM2USAread

delivers the USD money supply M2 directly from FRED in St. Louis, USA, via API

Out: **M2USD**: [datum values*1000] as coordinate list **dates**: dates as datetime struct **values**: values as double list

FINdataPOP

downloads and saves the population of several countries from the world bank homepage

In: **"force"**: **"DOI"**: if used, the download of the new data is enforced

Out: **popu**: population data vector read from the World Bank API

finddoubledrows

returns the indices of the lines with copies of a earlier row

In: **VL**: list of rows

Out: **c**: indices of the second, third, and so on, occur to be removed

finduniqueVL

similar to unique but more reliable

In: **NVL**: vector row list to search in **VL**: vector row list to search for **tol**: tolerance for warning; default is 1e-12 **warn**: if true; warnings are shown; default is true

Out: **ic**: list of index entries; 3rd result of unique **err**: deviation between VL and best fit in NVL **ia**: list of index entries; 2nd result of unique

loadweb

loads mat files from an web url

In: **mname**: name and extension of the file **overwr**: true=overwrite; default is false. **urlp**: web path; default is 'http://www.mimed.mw.tum.de/fileadmin/w00bhh/www/Matlab_Toolboxes/'

Out: **wname**: full path to loaded file

SGgrabcad

Simply opens the WWW-page of grabcad.com in the system browser

9.5.8 General / Misc Helpers [guide]

Functions in this section:

ageofbirthday — returns the age in years of a birthday string and an optional reference date string

AIM_NavCarm — Creation of Pictures for a Navigation C-Arm scenario

AIM_NavEndo — Creation of Pictures for a Navigation C-Arm scenario

alibaba — just opens the website of this company

allmembers — similar to ismember(A,S,'rows') but returns all members of a set

AM_3DPrinting — just opens the website of this company

atomiccovalentradius — returns the atomic radius or the covalent bond distance of two atoms determined from it

bgr162rgb — converts a 16 Bit AbC color code (BGR 3 x 5 + 1 Bit) into a color vector [R G B] 3 x 8 Bit

cell2array — returns an nxm array from a nxk cell; m=max(size(C{:}))

clear4lectures — Auxiliary fnct for Tim Lueth to record lectures using Matlab and OBS

colofn — returns a color char for a number 0..7

commandwindowTL — same as original fnct but deployable

connectofmat — return the element that have a connection in the matrix

crosspointVLFL — returns the surfacepoint of a solid hit by a beam

crosspointVLFL2 — returns the surfacepoint of a solid hit by a beam

depuseStringmult — Finds files that contain multiple strings linked by an AND condition.

dimpowercable — returns the required dimension for a motor power cable or the number of required power cables

dircrd — same as dir but returns the creation date instead of the modification date

distofintervall — distance within an intervall

docSG — just shows the documentation of the calling fnct

docTL — same as original fnct but deployable

editTL — same as original fnct but deployable

ejectvolume — Auxiliary fnct to unmount and eject a extrenal drive such as a sd card

ELdistanceofmatrixneighbors — Calculates from a potential field consisting of doubles and nan the neighborhood weights

epo — just opens European Patent Office Login Page

eps2 — returns the rounding limit for triangular or quadratic problems

epsofVL — returns the minimal points distance in a set of vertices or two groups

excelpaste — converts the content of the clipboard for a multi line insert/past into EXCEL on MAC

existvolume — returns the full path of a volume's diretory if this volume exists i.e. it is mounted

ezofchar — returns an ez vector simply to the numbers of a dice

fcostbenefit — cost or benefit fnct

findchangerow — returns in a list of succeding rows, when the row is different to its predecessor

flipvar — returns two flipped variable

fnameofmatlabcode — returns the name of the first matlab fnct within a matlab code

FTcontour — returns half profile of Fischertechnik dove tail contour

getfuncparamStr — similar as getfuncparams but works with strings prpoerties

getgcatitle — returns the current title of gca as string

getjobs — returns the jobs running on the local cluster

getvarg — returns if a string is part of varargin and the next parameter

googlePatentSearch — fnct opens browser with google patent page searches for patents and downloads patentlist as cvs

histograms — calculates histograms for matrices or images or voxel models etc.

hostname — SAME AS USERNAME BUT CHANGED OUTPUT returns user name and host name using JAVA

humidityoftemp — calculates the humidity at temperature change

imat — returns a matrix including row column and column row

imshow4 — Plots RGB images in 4 color segments

int32orfloat — changes type to integer float is an integer

interPL — returns a value as linear interpolation using a PL

iscalledbycmdline — returns whether a fnct was called by commandline directly

iscollofVLBB — returs false if there is no collision/attaching

isint32 — returns true or false if a number has no fraction

isintegerbv — returns whether a number is an integer by value not by type

islink — returns whether a file is just a link

isodd — returns whether a numer is even or odd

ispatch — returns whether a handle or the current graphics object is a patch

isrealr — returns whether all elements on a row are real

LLsort — sorts a kinematic chain list similar to an edge list

looper — converts a sequence into a one-line sequence for a For loop

lsegment — returns all supporting points of a line/edge including start and end point, separated

maprows — returns index mapping list for two rows list (error free since 2017-04-24)

mdfind — spotlight using matlab on mac

meanT — returns the man value of a T matrix

MGTcoefficientsai2bi — converts a z-transform Denominator polynomial into Numerator polynomial

MGTcoefficientsPID2bi — returns an PID controller equivalent sequence of an numerator polynomial coefficients of the FIR Controller in a closed loop control

MGTcontrollIRIIR — processes an input sequence in a closed control loop

MGTmotor — Fnct to explain the behavior of a motor or any mass damper spring system

MGTsystemFIR — generates a system response for a transfer fnct that has only bi coefficients

MGTsystemIIR — implements a IIR structure for signal filter or control tasks

MGTsysteminvFIR — calculates the FIR coefficients by solving a linear equation system

mod1 — returns mod fnct for elements 1:k

modN — returns mod fnct for elements N:N+k(-1)

note — opens a text file with a keyword in the matlab editor

openinmatlabeditor — same as opentoline but creates a file if not existing

opentolineTL — same as original fnct but deployable

p3ofn — returns 3 numbers for a given number n

paste2code — if called on the commandwindow, it returns the contant of the clipboard as matlab code

permutevector — creates vector list based on parametersets

pgrid — returns a point of a 2D grid

proxyTL — sets proxy values to access API services trough the TUM proxy

rcmod1 — return rows and colum for a given number and col length

readtable2code — reads tables and creates strings in the command window that can be used as code in matlab m files

removeimat — removes a row and a column of a indexed matrix

replacemat — replaces values in vectors and matrices

reversesortindex — returns the reverse sort index for a sort index

RLaddauxpoints — add supporting points to an open 2D contour to guarantee a specified points distance

RLdelauxpoints — deletes supporting points of an open 2D contour to generate staight lines

roman — returns a number as roman number

rounddiv — rounds a vector or matrix to a defined number of digits

rounds — is just an abbreviation if rounds(x,s,'significant')

saveTL — a modified version of the save command

sendemailTUM — sends an email using curl via TUM LRZ SMTP postout.lrz.de:587

SGaddfields — adds field of a field name list to all elements of a solid geometry

SGbatchdelete — deletes the global SGBatchIDtable of the SGbatch fncts

SGbatchjob — creates a job from a string using matlab's batch command and

SGbatchresult — returns the results of a job if it is finished or stores it in the workspace

SGfieldnames — collects all field names in different Solid Geometries

SGinspect — zooms to inspect specific vertices or facets

SGnamedset — creates a textfield to understand the parameters of the solid

SGopentoline — same as opentoline which does not work anymore in 2020b Update6 on Catilina

SGprint — writes an STL file and opens the slicer afterwards

SGprintPreForm — sends an SG to the PreForm app of Formlab

SGshortopti — returns the vertex list with optimal number of loss less reduced vertices

SGstampname — creates a textfield to understand the parameters of the solid

SGTframeChain — returns a list of frame connections for a cell list if solids

SGtrialerror — generates an error message indicating a new attempt

SGversion — returns a license string

shiftindexofinvkinplan2 — analyzses the real solution index of invkinplan2 and returns shift index and solution tbale

showprogress — creates a progress text in 5% steps and deletes it finally

signthr — returns a sign fnct related to a threshold

size2index — converts a size list into an index list [start end]

sizeVL — return the size of the bounding box of a vertex list

spotlightsearch — spotlight using matlab on mac

stamp — creates a text line for publishable code with currenct OSX version | Matlab Version | SGlib version

textHorizontalBlockAlign — returns cell of strings which length are limited in length

thisfuncname — just returns the name of the calling fnct

timlueth — just open the entries of tim lueth at Matlab central

uniquehist — performs the unique command and creates a full histogram count afterwards

unixgrep — same as unix grep fnct

unsortrows — swaps many different rows of a matrix

userIP — return internal and externam ip address of the used computer

username — returns user name and host name using JAVA

V2A — returns a set of strength calculation for a steel bending girder

VanDerWaalsRadius — returns the van der Waals Radius in pcm that can be used for a Kalottenmodel

vec2number — converts a vector row into a number for vector elements of 1 to 9

vectorinBB — returns a list of integer points along a vector that are inside of a BB

verSG — systematic names for versions - use osx for all

VofRsphere — Lead or steel balls must be installed in cavities to create ballast

webTL — same as original fnct but deployable

Wegertseder — just opens the website of company Wegertseder for often used normed elements

whichSG — searches for patters in filenames of matlab files

whoSG — SG list of global variables stored in the work space

XXprintmode — switches on the possibility to reach code lines when using dbprintf fnct

XXreadcomment — returns the text of the comments of the comment file as hyperlinks

Detailed description

ageofbirthday

returns the age in years of a birthday string and an optional reference date string

In: **day**: string of birthday, such as '30.11.1965) **refd**: string of refence date, such as {1.6.2016'; default is now

Out: **age**: age in years **chkd**: numerical value of date **refd**: numerical value of reference date

AIM_NavCarm

Creation of Pictures for a Navigation C-Arm scenario

AIM_NavEndo

Creation of Pictures for a Navigation C-Arm scenario

alibaba

just opens the website of this company

In: **text**: string for search

allmembers

similar to ismember(A,S,'rows') but returns all members of a set

In: **A**: Vector to comapre **S**: List of vectors

Out: **IL**: List of Indices

AM_3DPrinting

just opens the website of this company

In: **text**: string for search

atomiccovalentradius

returns the atomic radius or the covalent bond distance of two atoms determined from it

In: **Str**: Kovalent Bond String such as 'H-O-H'

Out: **R**: List of Atomic Radius and Van der Waals Radius **SG**: List of Solid Geometries fitting to the Radius table

bgr162rgb

converts a 16 Bit AbC color code (BGR 3 x 5 + 1 Bit) into a color vector [R G B] 3 x 8 Bit

In: **AbC**: List of "Attribute by Count" or BRG16Bit code

Out: **rgb**: List of Matlab RGB Values

cell2array

returns an nxm array from a nxk cell; m=max(size(C{:}))

In: **C**: cell array of number rows of different length

Out: **A**: Array with constant number of columns, NaN

clear4lectures

Auxiliary fnct for Tim Lueth to record lectures using Matlab and OBS

In: **text**: 'reset' will set 13 points, 'set will set to 20 point

colofn

returns a color char for a number 0..7

In: **i**: integer number **colbar**: order of colors; default is 'kbgcmyrw' **colstr**: colorstring such as 'R*--'

Out: **c**: color char string

commandwindowTL

same as original fnct but deployable

connectofmat

return the element that have a connection in the matrix

In: **CL**: Connection matrix **n**: input connection value/list; default is ''

Out: **no**: vector of all row/vol that are somehow connected to input vector n

crosspointVLFL

returns the surfacepoint of a solid hit by a beam

In: **VL**: Vertex list of a solid **FL**: Facet list of a solid **p**: beam starting point **ez**: beam direction vector

Out: **cp**: cross point on surface, empty if there is no crossing point **t**: distance (t * ez) **fi**: facet index of 1st crossed surface **a**: full list of facet index and distance [fi, d]

crosspointVLFL2

returns the surfacepoint of a solid hit by a beam

In: **VL**: Vertex list of a solid **FL**: Facet list of a solid **p**: beam starting point **ez**: beam direction vector; if empty the facet normal vectors are used

Out: **cp**: cross point on surface, empty if there is no crossing point **t**: distance (t * ez) **fi**: facet index of 1st crossed surface **a**: full list of facet index and distance [fi, d]

depuseStringmult

Finds files that contain multiple strings linked by an AND condition.

In: **term1**: search term 1 **term2**: search term 2 and more

Out: **ll**: cell list of file that containt all of the search strings

dimpowercable

returns the required dimension for a motor power cable or the number of required power cables

In: **P**: Power in Watt; default 10 W **U**: Voltage in Volt; default is 5 V **d**: optional existing diameter;

Out: **d**: diameter of cable or number of cables **dP**: loss of power by cable heating / resistor **text**: printable text string that contains the specification to order cable and connectors **vend**: printable text string that contains a vendor for the required components if known

dircrd

same as dir but returns the creation date instead of the modification date

Out: **xx**: result if dir

distofintervall

distance within an intervall

In: **a**: value 1 **b**: value 2 **inter**: intervall; default is [-pi +pi]

Out: **d**: distance () **iv**: value a in intervall

docSG

just shows the documentation of the calling fnct

In: **fname**: optinal fnc name if the caller is not desired

docTL

same as original fnct but deployable

In: **params**: same as original function

editTL

same as original fnct but deployable

In: **params**: same as original function

ejectvolume

Auxiliary fnct to unmount and eject a extrenal drive such as a sd card

In: **vnames**: name of voume or directory of volume

ELdistanceofmatrixneighbors

Calculates from a potential field consisting of doubles and nan the neighborhood weights

In: **M**: potential **eltype**: default is 4 neighborhood **nneigh**: neighborhood type '4' or '8'; default is '4' **noncirc**: flag to treat matrix as non-circular; default is true

Out: **EL**: Edge List [Start End Distance] **G**: indices of Matrix **iM**: weighted graph

epo

just opens European Patent Office Login Page

eps2

returns the rounding limit for triangular or quadratic problems

Out: **to**: rounding limit for square problems

epsofVL

returns the minimal points distance in a set of vertices or two groups

In: **NVL**: Vertex list with appended split points **n**: number to split the vertex list into 2 parts [1:n,:] & [n+1:end,:]

Out: **e**: smallest number larger 0 between the points **ea**: epsilon within the first group **eb**: epsilon within the second group **minl**: vector of minimal point distances per vertex

excelpaste

converts the content of the clipboard for a multi line insert/past into EXCEL on MAC

existvolume

returns the full path of a volume's diretory if this volume exists i.e. it is mounted

In: **vname**: volume title

Out: **dname**: matlab path to volume dir; ends always with filesep

ezofchar

returns an ez vector simply to the numbers of a dice

In: **c**: input char in roman letters (I,..VI) **T**: Optional different Coordinate system

Out: **ez**: normal vector **c**: selected char **T**: optional coordinate system passed in; returned unchanged

fcostbenefit

cost or benefit fnct

In: **c**: [plateau-width center-shift center-value]; default is [0 0 1] **r**: [ramp width and minimal value[;default is [2 0] **t**: type of functiopn "lin", "atan", "bell"; default is bell (1/1+x^2)

Out: **ff**: handle to cost equation **PL**: Point list of function **SG**: Solid Geometry of function

findchangerow

returns in a list of succeding rows, when the row is different to its predecessor

In: **c**: list of succeeding rows

Out: **i**: index list **CIL**: Change index list

flipvar

returns two flipped variable

In: **a**: Variable A **b**: Variable B

Out: **b**: Variable B **a**: Variable B

fnameofmatlabcode

returns the name of the first matlab fnct within a matlab code

In: **txt**: matlabcode of a function or script **frntchars**: default is 'CC_' **backchars**: default is '.m'

Out: **fname**: function name of matlabcode or empty if a script

FTcontour

returns half profile of Fischertechnik dove tail contour

Out: **PL**: Point list of dove tail contour

getfuncparamStr

similar as getfuncparams but works with strings prpoerties

In: **n**: number or string of argument **args**: arguments such as varargins **v**: default value or empty, if simple bollean **alpha**: fillin flag passed to getfuncparams when n is numeric **"debug"**: true if the string "debug" is present in the arguments **"A0"**: value following the property "A0" in the arguments, else default

Out: **RR**: result **var**: modified varargin **narg**:

getgcatitle

returns the current title of gca as string

Out: **s**: string

getjobs

returns the jobs running on the local cluster

Out: **jobs**: array of jobs **names**: cell list of job names

getvarg

returns if a string is part of varargin and the next parameter

In: **argin**: varargin **paramstr**: searchstring **np**: number of required arguments, default is 0 **P**: optional initial value returned in P if the search string is not found

Out: **f**: false/true if the searchstring was found **P**: varargins following the searchstring

googlePatentSearch

fnct opens browser with google patent page searches for patents and downloads patentlist as cvs

In: **keywords**: keywords for AND search

Out: **data**: data table with the results **fname**: file name that corresponds to the last search in the download directory

histograms

calculates histograms for matrices or images or voxel models etc.

In: **A**: Image matrix **n**: number of equal sized groups; if negative e-true) **e**: true = equal group size

Out: **H**: Histogram (nx2) Group Value and Group size **IQ**: Table of Group Values **AE**: Image Matrix with clustered values

hostname

SAME AS USERNAME BUT CHANGED OUTPUT returns user name and host name using JAVA

Out: **host**: host name **user**: user name **mcpu**: CPU/hardware description string derived from the host name (portion after the second space)

humidityoftemp

calculates the humidity at temperature change

In: **th**: [temperatur humidity] at the moment **t2**: [new temperatur]

Out: **h2**: humidity of t2 is reached and there is no water change

imat

returns a matrix including row column and column row

In: **D**: matrix 2x2

Out: **DN**: matrix plus row column and column row

imshow4

Plots RGB images in 4 color segments

In: **I**: RGB image or grayscale **maxrc**: maximum size of the longer image side for resizing; default is none

int32orfloat

changes type to integer float is an integer

In: **a**: float

Out: **a**: same number as float or as integer

interPL

returns a value as linear interpolation using a PL

In: **PL**: Point list (nx2) using x and y values **x**: input value within the interval of xmin and xmax of PL

Out: **y**: interpolated value for y or NaN outside of xmin and xmax of PL

iscalledbycmdline

returns whether a fnct was called by commandline directly

Out: **b**: logical true if length(dbstack) <=2

iscollofVLBB

returs false if there is no collision/attaching

In: **VLA**: Vertex list A **VLB**: Vertex list B

Out: **c**: true if bounding boxes collide or attach

isint32

returns true or false if a number has no fraction

In: **a**: number

Out: **b**: true if isequal((rem(a,1),0)

isintegerbv

returns whether a number is an integer by value not by type

In: **x**: scalar, array or vector

Out: **l**: logical value,array or vector

islink

returns whether a file is just a link

In: **fname**: file name

Out: **islink**: yes if the fname is a link

isodd

returns whether a numer is even or odd

In: **x**: number

Out: **r**: true if odd

ispatch

returns whether a handle or the current graphics object is a patch

In: **h**: graphics object handle, default is gco

Out: **b**: true/false

isrealr

returns whether all elements on a row are real

In: **VL**: vector list with rows and columns [n x m]

Out: **ri**: index list; r(i)==true if all elements of VL(i,:) = real

LLsort

sorts a kinematic chain list similar to an edge list

In: **LL**: Link list (n x 2) of joints **GL**: Base list of static joints, optional []; **VL**: Vertex list, optional

Out: **CIL**: Connection index list **LLN**: Sorted Link List **V**: Node index list including classification

looper

converts a sequence into a one-line sequence for a For loop

In: **ind**: index list for a for-Loop

Out: **ind**: corrent single line index list for a for-Loop

lsegment

returns all supporting points of a line/edge including start and end point, separated

In: **p1**: Starting point of an edge **p2**: End point of an edge **t**: list of points: [p=p1+t*(p2-p1) , direction]

Out: **AVEL**: Edge list of an outer contour

maprows

returns index mapping list for two rows list (error free since 2017-04-24)

In: **VLA**: List A **VLB**: List B

Out: **a**: used to transform index list based on A indices **b**: used to transform index list based on B indices

mdfind

spotlight using matlab on mac

Out: **b**: result of spotlight as cell list

meanT

returns the man value of a T matrix

In: **TL**: cell list or numerical sequence of 4x4 matrices

Out: **T**: mean value of a set of HT Matrices

MGTcoefficientsai2bi

converts a z-transform Denominator polynomial into Numerator polynomial

In: **ai**: ai coefficients of a strucuture of type 1/ai polynomial **thr**: default is 1e-3

Out: **bi**: ai coefficients of an equivalent strucuture of type bi polynomial **cbi**: cumulative difference of bi coefficients, [bi 0]-[0 bi]

MGTcoefficientsPID2bi

returns an PID controller equivalent sequence of an numerator polynomial coefficients of the FIR Controller in a closed loop control

In: **P**: Proportional Part P*u[k] **I**: Integral Part I*sum(u[k:-1:k-n]) **D**: Differential part D*(u[k]-k[k-1]) **n**: number of bi numerator polynomial coefficients of the FIR Controller

Out: **bi**: numerator polynomial coefficients of the FIR Controller

MGTcontrolIIRIIR

processes an input sequence in a closed control loop

In: **w**: Input signal sequence **aiR**: denominator polynomial coefficients of the controller IIR **biR**: numerator polynomial coefficients of the controller IIR **aiM**: denominator polynomial coefficients of the motor/process IIR **biM**: numerator polynomial coefficients of the motor/process IIR **step**: if true, the function is plotted step by step during processing

MGTmotor

Fnct to explain the behavior of a motor or any mass damper spring system

In: **u**: input signal **m**: mass constant kg; default is 0.1kg (increasing means higher mass) **d**: damper constant kg/sec; default is 0.01 kg/sec (increasing means higher damping) **c**: spring constant kg/sec^2; default is 1 kg/sec^2 (increasing means higher stiffness) **ss**: if true, progress is shown step by step

Out: **y**: motor response **ai**: ai coifficients calculated from m d c

MGTsystemFIR

generates a system response for a transfer fnct that has only bi coefficients

In: **u0**: numer sequence describin a signal / function **bi**: [1 x n] Coefficients [b0 b1 b2 b3 .....bn-1] **sh**: if true; the progess is shown step by step

Out: **y**: response of y "masked" with coefficients bi

MGTsystemIIR

implements a IIR structure for signal filter or control tasks

In: **u0**: input signal **ai**: Denominator polynomial coefficients **bi**: Counts polynomial coefficients **sh**: Show progress step by step if true

Out: **y**: result signal

MGTsysteminvFIR

calculates the FIR coefficients by solving a linear equation system

In: **u**: input sequence **y**: output sequence

Out: **bi**: FIR coefficients (convolution function), magenta colored subplot

mod1

returns mod fnct for elements 1:k

In: **n**: number **k**: divider **shft**: default is 1

Out: **a**: rest

modN

returns mod fnct for elements N:N+k(-1)

In: **n**: number **k**: divider **N**: Interval start

Out: **a**: rest

note

opens a text file with a keyword in the matlab editor

In: **fname**: keyword used as filename later **create**: if true, the file will be created if not existing already

openinmatlabeditor

same as opentoline but creates a file if not existing

In: **fname**: file name **ln**: line for cursor

opentolineTL

same as original fnct but deployable

In: **params**: same as original function

p3ofn

returns 3 numbers for a given number n

In: **n**: number b **solut**: optional selection variant for the three indices; default is 0

Out: **a**: 1 **b**: 1*n/3 **c**: 2*n/3

paste2code

if called on the commandwindow, it returns the contant of the clipboard as matlab code

In: **n**: number of chars per line; default is 112; **clip**:

Out: **text**: text from the clipbord

permutevector

creates vector list based on parametersets

In: **ps1**: paraneter set 1 **ps2**: paraneter set 2 **ps3**: paraneter set 3

Out: **VL**: Value vector list

pgrid

returns a point of a 2D grid

In: **x**: x-grid **y**: y-grid **lx**: x-size of the grid **ly**: y-size of the grid **nx**: number of grid points in x **ny**: number of grid points in y **type**: 0 = grid starts at 0; 1=grid starts at lx(nx)/2; default is 1

Out: **p**: point (1:3,1) of the grid

proxyTL

sets proxy values to access API services trough the TUM proxy

In: **name**: proxy setting name such as 'tum' or 'reset'; default is empty (shows current proxy)

rcmod1

return rows and colum for a given number and col length

In: **n**: index **cx**: number of cols **sq**: if true; cx is the length of a squared field; default is false

Out: **rc**: [row col] =[1... 1...]

readtable2code

reads tables and creates strings in the command window that can be used as code in matlab m files

In: **fname**: file name of a table such as "temptable.xls"

removeimat

removes a row and a column of a indexed matrix

In: **D**: indexed matrix **r**: row to delete **c**: column to delete

Out: **ND**: Matrix without this column and row

replacemat

replaces values in vectors and matrices

In: **A**: vectors or matrices **valold**: existing values **valnew**: new values

reversesortindex

returns the reverse sort index for a sort index

In: **iO**: result of sortrows

Out: **rO**: reverse sort order

RLaddauxpoints

add supporting points to an open 2D contour to guarantee a specified points distance

In: **RL**: Open 2D contour list **d**: maximum distance between two points **clse**: true=close contour; default is true

Out: **RL**: Open 2D contour list

RLdelauxpoints

deletes supporting points of an open 2D contour to generate staight lines

In: **RL**: Open 2D contour list

Out: **RL**: Open 2D contour list **i**: index list of the original contour list

roman

returns a number as roman number

In: **n**: number

Out: **S**: String of roman number **nL**: separated divisor list

rounddiv

rounds a vector or matrix to a defined number of digits

In: **x**: Number, Vector, Matrix **lim**: Limit

Out: **x**: result

rounds

is just an abbreviation if rounds(x,s,'significant')

In: **x**: floating point number **s**: number of

Out: **y**: rounded x

saveTL

a modified version of the save command

In: **vname**: variable name to save under the same name as .mat file **fname**: filename **copyn**: if true; the variable name changes to fname

sendemailTUM

sends an email using curl via TUM LRZ SMTP postout.lrz.de:587

In: **to**: email receiver address **subject**: email subject **body**: email body **from**: email sender address (such as xxxx@mytum.de) **pass**: passwort of sender address **"nosend"**: if used the email is not send, but the curl command is prompted

Out: **status**: status of curl **result**: result of curl

SGaddfields

adds field of a field name list to all elements of a solid geometry

In: **SG**: Solid Geometry **FNN**: cell list of all field names similar to fieldnames

Out: **SGN**: New solid with all elements have the

SGbatchdelete

deletes the global SGBatchIDtable of the SGbatch fncts

In: **ID**: Optional command selecting which batch entries to delete

SGbatchjob

creates a job from a string using matlab's batch command and

In: **cmd**: matlab command to execute as parallel job

Out: **job**: parallel job

SGbatchresult

returns the results of a job if it is finished or stores it in the workspace

In: **cmd**: ID or job or command string

Out: **result**: fetched job result, or empty if not finished

SGfieldnames

collects all field names in different Solid Geometries

In: **SG**: Solid Geometry

Out: **FN**: Names of fields of the different SG structs and cells

SGinspect

zooms to inspect specific vertices or facets

In: **SG**: Solid Geoemetry **vi**: vertex indices to inspect **fi**: facet indices to inspect

SGnamedset

creates a textfield to understand the parameters of the solid

In: **Solids**: Solid Geometries as varargin separated by comma

Out: **SGall**: SG cell list including stampnames and stampdates

SGopentoline

same as opentoline which does not work anymore in 2020b Update6 on Catilina

In: **fname**: filename **line**: line number

SGprint

writes an STL file and opens the slicer afterwards

SGprintPreForm

sends an SG to the PreForm app of Formlab

In: **SG**: Solid Geoemetry to print **ptyp**: 'arrange', 'box', 'pack', 'sep' **sdim**: build volume dimensions; 'eos' sets [242 193 320] mm, else [inf inf inf]; default is 'eos'

Out: **SG**: The solid geometry, arranged for printing

SGshortopti

returns the vertex list with optimal number of loss less reduced vertices

In: **SG**: Original SG **otyp**: optimization type: 'vertex' or 'boundary'

Out: **SGN**: SG.VL, SG.FL with optimal number of loss less reduced vertices

SGstampname

creates a textfield to understand the parameters of the solid

In: **SG**: Solid Geometry **txtparams**: parameter to create a string such as in sprintftext

Out: **SG**: SG including stampname and stampdate

SGTframeChain

returns a list of frame connections for a cell list if solids

In: **nums**: number, or list of chain elements **extr**: extra chaining [SG1ind FFrame SG2ind BFrame] such as [1 'F' 2 'B'] **B**: Standard Base frame string; default is 'B' **F**: Standard follower frame string; default is 'F'

Out: **LFrm**: Link Frame Tabe **LL**: Link list [start index , End index] in SGc **CCstr**: compiled string like the extra chaining

SGtrialerror

generates an error message indicating a new attempt

SGversion

returns a license string

Out: **outstr**: current license (last compiled version) **Toolboxname**: toolbox name; 'SG-Lib' **VNAME**: version name string **LDAYS**: license duration in days **LNAME**:

shiftindexofinvkinplan2

analyzses the real solution index of invkinplan2 and returns shift index and solution tbale

In: **rs**: reals solution index of invkinplan2

Out: **shi**: shift indexe for circshift (rs,-shi) **wtab**: angle access table (start end)

showprogress

creates a progress text in 5% steps and deletes it finally

In: **i**: current index **n**: maximum index number **dv**: optional step size; default is 5;

signthr

returns a sign fnct related to a threshold

In: **x**: value

Out: **s**: sign

size2index

converts a size list into an index list [start end]

In: **ni**: list of [size of element]

Out: **il**: list of [startindex endindex]

sizeVL

return the size of the bounding box of a vertex list

In: **VL**: Vertex list of the object

Out: **sx**: size in x OR [xmin xmax ymin ymax zmin zmax] **sy**: size in y **sz**: size in z **mx**: minimal value in x **my**: minimal value in y **mz**: minimal value in z

spotlightsearch

spotlight using matlab on mac

Out: **b**: result of spotlight as cell list

stamp

creates a text line for publishable code with currenct OSX version | Matlab Version | SGlib version

textHorizontalBlockAlign

returns cell of strings which length are limited in length

In: **ncol**: maximum number of letters in a row [c r] **A**: cell of strings

Out: **B**: cell of string **C**: single string of the aligned lines joined by newlines

thisfuncname

just returns the name of the calling fnct

Out: **t**: name of calling function **td**: name of calling fnct with current date and time appended

timlueth

just open the entries of tim lueth at Matlab central

uniquehist

performs the unique command and creates a full histogram count afterwards

In: **A**: array A

Out: **C**: unique arrary, may be sorted **ia**: index vector ia **ic**: index vector ib **h**: number of entities of C in A

unixgrep

same as unix grep fnct

In: **sstr**: search string **greps**: search file

Out: **b**: result

unsortrows

swaps many different rows of a matrix

In: **A**: Original matrix

Out: **A**: unosrted matrix, randomly exchnanged rows

userIP

return internal and externam ip address of the used computer

Out: **ipi**: internal ip address **ipe**: externally seen ip address

username

returns user name and host name using JAVA

In: **datetag**: date format appended to user name; '', 'roman', 'std'/'arab'/'yyyy', or a datestr format; default is none

Out: **user**: user name **host**: host name

V2A

returns a set of strength calculation for a steel bending girder

In: **d**: Diameter of the steel rod **v**: Safety factor; default is 3.5 **Rm**: Tensile Load in N/mm^2 (MPa) as material constant; default is 520 (V2A) **E**: Young's modulus in N/mm^2 (MPa); default is 210000 (steel)

VanDerWaalsRadius

returns the van der Waals Radius in pcm that can be used for a Kalottenmodel

In: **Str**: String abbreviation of the element such as 'H' or "He' etc.

Out: **vr**: radius if exisiting for the element **E**: Solid Geometry of the V-d-W-Sphere **kr**: atomic core radius in pm for the element

vec2number

converts a vector row into a number for vector elements of 1 to 9

In: **ord**: vector of integer between 1 and 9

Out: **z**: number **s**: string of number

vectorinBB

returns a list of integer points along a vector that are inside of a BB

In: **BB**: Bounding Box **v**: vector to check **p**: point to start with; default is 0

Out: **n**: rows of integer [min i max i] **nn**: row [minx maxx miny maxy minz maxz] **VL**: Vertex list including p0 if possible

verSG

systematic names for versions - use osx for all

In: **vdate**:

Out: **t**: name of SG-Lib version

VofRsphere

Lead or steel balls must be installed in cavities to create ballast

In: **R**: Radius [r h]; default is [r r] **M**: Material such as 'Fe', 'Pb', 'Si'

Out: **V**: Volume **W**: Weight **M**: Material

webTL

same as original fnct but deployable

In: **params**: same as original function

Wegertseder

just opens the website of company Wegertseder for often used normed elements

In: **text**: string or number of DIN norm such as 7979 or '7979'

whichSG

searches for patters in filenames of matlab files

In: **pat**: pattern to search **day**: if used the output is given as string sequence too **"date"**: **"list4doc"**:

Out: **flist**: list fo files in working directory **mlist**: matlab functions of this name

whoSG

SG list of global variables stored in the work space

In: **vari**: variable in global workspace **desc**: description of variable

Out: **ri**: returns true or false if not existing or existing in global SG_variables

XXprintmode

switches on the possibility to reach code lines when using dbprintf fnct

In: **md**: true or false; default is true;

Out: **XXmode**: true or false

XXreadcomment

returns the text of the comments of the comment file as hyperlinks

In: **fname**: name of function ; default is last used function on command line

Out: **txtwh**: comment text as hyperlinked string, each line prefixed with a comment marker

9.6 AI, Data and Integration [guide]

9.6.1 LLM / ChatGPT (OpenAI) [guide]

Functions in this section:

AImodelNameSet — Changes the AI model used by default in chat

json2matlab — converts a json reaction of an openAI-Chat into a matlab code line

json2TextMatlabcodeUsecase — supports json code as response of an chat - simply processes the response struct message

jsonplansequence — returns a plan step sequence from a single chat task

openAIaddMessageCurl — adds a "message" to a recently created "thread"

OpenAIcostAPI — opens the web page of company openAI to show the users expenses

openAIcreateAssistantCurl — creates an openAI V2 assistant with CURL

openAIcreateThreadCurl — creates an empty "thread" for a "message" and an "Assistant V2" (CURL version)

openAIcreateVectorstore — creates an opAI vector store (CURL Version)

openAIdatestr — converts the openAI date format into datenum into datestr (GMT)

openAIDeleteAssistantCurl — deletes an v2 assistant based on its assistant ID (CURL version)

openAIdeleteFileFromVectorStoreCurl — deletes (failed) files from vector stores (CURL Version)

openAIdeleteThreadCurl — deletes a thread for security reasons as threads cannot be accessed if by user throught the API if the thread id is lost

openAIdeleteuploadedFile — deletes uploaded binary files to openAI even with 2024b (CURL Version)

openAIdeleteVectorstore — deletes an openAI vector store by its id (webwrite Version)

openAIdownloadFile — download an uploaded binary files from openAI even with 2024b (CURL Version)

openAIgetAssistantInfo — returns the relevant information corresponding to an assistant ID

openAIgetAssistantsCurl — shows all assistants of your API Key (CURL version)

openAIgetCHATenvironment — gets the environment variables required for the CHAT by loading the environment file

openAIgetInfo — prints a list of uploaded files, vector-stores, assistants. Threads are hidden by the API in general

openAIgetmodels — returns all API GPT models of openAI - testing fnct for openAI (webread version)

openAIgetRunStatusCurl — returns the status for a "thread" "run"

OpenAIgetThreadList — returns the list of threads stored in the file such as OPENAI-<username>-IDs.txt

openAIgetThreadMessages — returns the answer of an assistant's thread's message/question

openAIgetThreadRuns — Returns all runs of a given thread (Assistant v2, curl, macOS).

openAIgetThreadslastMessages — returns the last n messages/conversation of the chat

openAIgetuploadedFiles — get the list of uploaded binary files to openAI even with 2024b (CURL Version)

openAIgetVectorStoreFilesCurl — get files an their status within a vector store (CURL Version)

openAIgetVectorstores — gets the names/IDs of all openAI vector store (CURL Version)

OpenAIlogID — records locally the ids of openAI structures which are not recorded anywhere, such as thread-IDs. Required to delete threads

OpenAIPDFemailanswers — analyzes Emails and returns an email to the sender

OpenAIPDFtextclassification — analyzes PDF scans with OCR content and create a file name for it

OpenAIrequest — returns the openAI Chat anwer for a single question

openAIsetAssistantInstructions — renames an existing Assistant V2 with a new Name (CURL POST version)

openAIsetAssistantName — renames an existing Assistant V2 with a new Name (CURL POST version)

openAIsetCHATenvironment — sets environment variables for chats using Assistant V2 (Thread and Assistant)

openAIsetVectorstoreName — renames an existing Vectore-Store with a new Name (CURL POST version)

openAIstartRunCurl — Starts an "assistant" "run" for a "thread"

openAIupdateAssistantVectorStore — simple reregister a Vectorstore of an Assistant

openAIupdateVectorstoreFile4Assitant — updates a single file of a vector store of an Assistent be uploading a new version and deleting the old one

openAIuploadFile — upload binary files to openAI even with 2024b (CURL Version)

openAIwaitForRunCurl — uses openAIgetRunStatusCurl to wait for a thread run

Detailed description

AImodelNameSet

Changes the AI model used by default in chat

json2matlab

converts a json reaction of an openAI-Chat into a matlab code line

In: **SGfuncname**: Name of the current SG function extracted from the AIexecplan **SGfuncargs**: Struct containing the parameternames as fields and expected values extracted from the AIexecplan **Variables**: History of already used SGresults as cell list of SGfuncnames with the call number

Out: **SGansw**: evaluated result from the matlab code extracted from the json answer. **cmdstr**: the command string executed within json2matlab to create res

json2TextMatlabcodeUsecase

supports json code as response of an chat - simply processes the response struct message

In: **msg**: message as standard response struct

Out: **text**: text response of the message **matlabcode**: matlabcode of the message **usecasemd**: usecase strategy as learned by assistant **resp**: JSON struct of the answer message

jsonplansequence

returns a plan step sequence from a single chat task

In: **Userinp**: User input as one string **AIchat**: AIchat **AIhistory**: AI History

Out: **Userscript**: plan sequence as single string **AIhistory**: modified AI History

openAIaddMessageCurl

adds a "message" to a recently created "thread"

In: **thread_id**: thrad/chat ID **text**: Question/Message in the Chat

Out: **resp**: JSON response struct

OpenAIcostAPI

opens the web page of company openAI to show the users expenses

Out: **cost**: returned empty (opens the OpenAI usage/limits web pages instead)

openAIcreateAssistantCurl

creates an openAI V2 assistant with CURL

In: **vectID**: Vector-store id **instr**: Instructions for the agent; default is especially for SG=Lib

Out: **assID**: Assistant ID **resp**: JSON response struct

openAIcreateThreadCurl

creates an empty "thread" for a "message" and an "Assistant V2" (CURL version)

Out: **thrID**: Thread ID of an empty thread

openAIcreateVectorstore

creates an opAI vector store (CURL Version)

In: **stname**: Store name, must be unique in the SG-Lib, in contrast to openAI

Out: **id**: vectore-store id **resp**: response struct

openAIdatestr

converts the openAI date format into datenum into datestr (GMT)

In: **createdAI**: datenum of openAI (posix format)

Out: **ds**: datestring

openAIDeleteAssistantCurl

deletes an v2 assistant based on its assistant ID (CURL version)

In: **assID**: assistant ID

Out: **resp**: JSON response struct

openAIdeleteFileFromVectorStoreCurl

deletes (failed) files from vector stores (CURL Version)

In: **vid**: vector store id **fid**: file id

Out: **resp**: response struct

openAIdeleteThreadCurl

deletes a thread for security reasons as threads cannot be accessed if by user throught the API if the thread id is lost

In: **thread_id**: thread id must be logged by OpenAIlogID as in

Out: **resp**: Response from the OpenAI API

openAIdeleteuploadedFile

deletes uploaded binary files to openAI even with 2024b (CURL Version)

In: **fid**: file name

Out: **result**: file id

openAIdeleteVectorstore

deletes an openAI vector store by its id (webwrite Version)

In: **store_id**: Store id returned by openAigetVectorstores

Out: **resp**: response struct

openAIdownloadFile

download an uploaded binary files from openAI even with 2024b (CURL Version)

In: **fid**: openAI file id

Out: **result**: json struct; use struct2cell(result.data)'

openAIgetAssistantInfo

returns the relevant information corresponding to an assistant ID

In: **assistant_id**: assistant ID

Out: **resp**: JSON response struct

openAIgetAssistantsCurl

shows all assistants of your API Key (CURL version)

Out: **names**: list of names **ids**: list of IDs **resp**: JSON struct

openAIgetCHATenvironment

gets the environment variables required for the CHAT by loading the environment file

In: **Name**: Name of the CHAT, Case ignored

Out: **threadID**: Thread ID **asstID**: Assistant V2 ID **EnvNam**: Environment code **fullname**: full name of the environment file

openAIgetInfo

prints a list of uploaded files, vector-stores, assistants. Threads are hidden by the API in general

openAIgetmodels

returns all API GPT models of openAI - testing fnct for openAI (webread version)

Out: **ccc**: cell list of model ids **resp**: response struct

openAIgetRunStatusCurl

returns the status for a "thread" "run"

In: **thrID**: Thread ID **runID**: Run ID

Out: **resp**: JSON response struct

OpenAIgetThreadList

returns the list of threads stored in the file such as OPENAI-<username>-IDs.txt

In: **"check"**:

Out: **thrIDs**: just the list of threads used in ID log file **resp**: response of all still existing threads from openAIgetThread **list**: list of entries in the openAI logfile

openAIgetThreadMessages

returns the answer of an assistant's thread's message/question

In: **thrID**: thread id, independent on which agent responded to the question **lastmsgID**: if used, it is possible to get older Messages (is_more)

Out: **answer**: text answer **resp**: JSON response struct

openAIgetThreadRuns

Returns all runs of a given thread (Assistant v2, curl, macOS).

In: **thrID**: Thread ID

Out: **resp**: JSON decoded struct of all RUNS **data**: data of resp **use**: usage of data **tok**: sum of all tokens of all runs

openAIgetThreadslastMessages

returns the last n messages/conversation of the chat

In: **thrID**: Thread ID **nn**: number of messages to receive; default is 1

Out: **outm**: list of json responses **resp**: response of all required answer json structs

openAIgetuploadedFiles

get the list of uploaded binary files to openAI even with 2024b (CURL Version)

Out: **result**: json struct; use struct2cell(result.data)'

openAIgetVectorStoreFilesCurl

get files an their status within a vector store (CURL Version)

In: **vid**: vectore store id

Out: **resp**: response struct

openAIgetVectorstores

gets the names/IDs of all openAI vector store (CURL Version)

Out: **ids**: names of vector stores **resp**: ids of vector stores **names**: response struct

OpenAIlogID

records locally the ids of openAI structures which are not recorded anywhere, such as thread-IDs. Required to delete threads

In: **ID**: ID of openAI files, vectorstores, assistant, threads, messages etc. **desc**: description, optionally

Out: **ccc**: content of fname=[workdir filesep 'openAI-' username '-IDs.txt']

OpenAIPDFemailanswers

analyzes Emails and returns an email to the sender

In: **fname**: current file name **txtlen**: number of characters read

Out: **answ**: Text message

OpenAIPDFtextclassification

analyzes PDF scans with OCR content and create a file name for it

In: **fname**: current file name **txtlen**: Maximum text length sent for classification

Out: **fclass**: proposed file name

OpenAIrequest

returns the openAI Chat anwer for a single question

In: **Userinp**: A charcter chain as input; default is empty **AIhistory**: the struct that is used by addResponseMessage, to maintain a chat history; default ist empty **RequestHistory**: a cell list that describes the history of Userinp in case of a conversation

Out: **AIansw**: Answer to the last Userinp **AIhistory**: AI history of response messages if a conversion is planned **RequestHistory**: The history of Userinp if a a conversion is planned

openAIsetAssistantInstructions

renames an existing Assistant V2 with a new Name (CURL POST version)

In: **assID**: Assistant ID **assID**: new inatructions for Assistant

Out: **result**: JSON result struct

openAIsetAssistantName

renames an existing Assistant V2 with a new Name (CURL POST version)

In: **assID**: Assistant ID **newName**: new name for Assistant

Out: **resp**: JSON result struct

openAIsetCHATenvironment

sets environment variables for chats using Assistant V2 (Thread and Assistant)

In: **Name**: Name of Chat - Such as SG' - No spaces allowed, upper case **asstID**: assistant ID for the chat **threadID**: thread ID for the chat

Out: **EnvNam**: Environment to load **fullname**: full path of the written .env file in the workdir

openAIsetVectorstoreName

renames an existing Vectore-Store with a new Name (CURL POST version)

In: **vector_store_id**: Vectore-Store ID **new_name**: new name for Vectore-Store

Out: **resp**: JSON result struct

openAIstartRunCurl

Starts an "assistant" "run" for a "thread"

In: **thread_id**: Thread ID **assistant_id**: Assistant ID

Out: **run_id**: Run ID

openAIupdateAssistantVectorStore

simple reregister a Vectorstore of an Assistant

In: **asstID**: openAI assistant ID **vectID**: openAI vector ID

Out: **resp**: JSON response struct

openAIupdateVectorstoreFile4Assitant

updates a single file of a vector store of an Assistent be uploading a new version and deleting the old one

openAIuploadFile

upload binary files to openAI even with 2024b (CURL Version)

In: **fname**: file name

Out: **file_id**: file id

openAIwaitForRunCurl

uses openAIgetRunStatusCurl to wait for a thread run

In: **thrid**: Thread ID **runid**: Run ID

Out: **resp**: JSON response struct

9.6.2 Expert Systems and STRIPS Planning [guide]

Functions in this section:

TIMSRESEARCH — creates web page content from a resesrch milestone diary

XPpermute — Auxiliary fnct to create permutation of sets

XPSblocksworld — Expert System implementing the first General Problem Solver (AI Programm)

XPSblocksworldequation — RULES for the XPSEQS blockworld problem

XPSconditionfindneval — creates all combinations of a variable based CONDITION that could fit a fact list CONDITION

XPSEQS — Expert System implementing the first General Problem Solver (AI Programm)

XPSGPS — Expert System implementing the first General Problem Solver (AI Programm) using STRIPS

XPSrulechecker — checks XPS rules for easy human errors during rule definition

XPSrulesread — reades rules from an excel sheet

XPSruleswrite — writes rules into an excel sheet

XPSslidingpuzzle — Expert System implementing the first General Problem Solver (AI Programm)

Detailed description

TIMSRESEARCH

creates web page content from a resesrch milestone diary

In: **KW**: list of key words **"web"**: an existing report should be converted into html **"video"**: an existing report should be converted into a video (batch) **"force"**: a new report is enforced

Out: **resp**: text response

XPpermute

Auxiliary fnct to create permutation of sets

In: **sc**: separator; default is '' **lists**: cell lists of arguments

Out: **LL**: cell list of all combinations of the permutation

XPSblocksworld

Expert System implementing the first General Problem Solver (AI Programm)

Out: **ops**: cell list [n x 4] describing action-name, precondition, add-state, diff-state **start**: string array describing the start state **goal**: string array describing the final state

XPSblocksworldequation

RULES for the XPSEQS blockworld problem

Out: **ops**: cell list [n x 4] describing action-name, precondition, add-state, diff-state **start**: string array describing the start state **finish**: string array describing the final state

XPSconditionfindneval

creates all combinations of a variable based CONDITION that could fit a fact list CONDITION

In: **cond**: string array of conditions with Variables of Type $Str **state**: string array of facts **VLIST**: Additional string arrys with variables

Out: **RLIST**: all possible condition strings that are part of state **VLIST**: all possible additional condition strings that use the same combination of VARS/INST **VARS**: all used symbol names in cond **INST**: all symbol combinations used as variables that fullfill the condition in state **SYMS**: string array of all samples

XPSEQS

Expert System implementing the first General Problem Solver (AI Programm)

In: **ops**: cell list [n x 4] describing action-name, precondition, add-state, diff-state **start**: string array describing the start state **finish**: string array describing the final state **"debug"**: if used, the progress is shown **"first"**: if used, the first solution is returned

Out: **solut**: solutions **endstate**: end state of the solutioin **solutstates**: state change of the solutions **plan**: all plans generated

XPSGPS

Expert System implementing the first General Problem Solver (AI Programm) using STRIPS

In: **ops**: cell list [n x 4] describing action-name, precondition, add-state, diff-state **start**: string array describing the start state **finish**: string array describing the final state

Out: **solut**: solution - successful plans **plan**: complete plan list

XPSrulechecker

checks XPS rules for easy human errors during rule definition

In: **ops**: STRIPS rules [Action, Condition-List, Add-list, Delete-list, NOT-Condition-List]

XPSrulesread

reades rules from an excel sheet

In: **fname**: filename to read from

Out: **T**: rules list

XPSruleswrite

writes rules into an excel sheet

In: **ops**: operators **fname**: filename

Out: **rname**: rules full file name

XPSslidingpuzzle

Expert System implementing the first General Problem Solver (AI Programm)

Out: **ops**: cell list [n x 4] describing action-name, precondition, add-state, diff-state **start**: string array describing the start state **goal**: string array describing the final state

9.6.3 AI Text, String and Cell Helpers [guide]

Functions in this section:

backspstr — returns a backspace character string to remove the given string

bug194 — Removed (not converted) all 194 bugs in texts

cell2strarr — converts string arrays in cell lines into a string array

cell2strlines — converts a cell list fo string tinto a single string line

char2html — converts german "Umlaute" and other speciel characters into UTF8 html code

char2strarr — converts a list of string inside a char chain into a string array

charcell2stringcellarray — converts xps cell list of string arrays into cell list of character chains

dbprintchar — similar to an ancient hex editor, this fnct plots codes and chars of a string or char array

evalhyperlinktab — evaluates an hyperlink text input from a table

fprintfvec2 — prints vectors in a way that it can used in code generation

hexstr2str — converts a uint8 string into a hex string array

hexstr2uint8 — converts a hexadecimal string into a uint8 array

html2char — Converts HTML code in UTF8 into ASCII characters.

isUTF8 — Returns the number of codes that are not in the ASCII range 0..127 or the German umlauts.

latin1ofUTF — Provides a latin1 code for UTF so that the ASCII files are read correctly.

markdown2html — converts standard text or mardown text into html and pdf

num2strCNT — creates numberstring including addition zeros

num2strUP — creates a string containing a unit prefix

numelbytes — returns the number of bytes a variable allocates

save2clipboard — simply saves the content of a variable to the "clipboard" to use it with paste

sprintfhyperlinktab — creates a hyperlink text string from a table

sprintfilehyperlink — creates a string containing two hyperlinks to open an file

sprintftext — creates as simple as possible a string for design parameters

sprintfvec — prints vectors in a way that it can used in code generation

sprintfvecUP — string format for engineering applications

str2codelines — creates from long string, several lines that can be used as matlab code

str2datenum — converts am formatted date string into a date num for numerical analysis

str2hexsstr — converts a uint8 string into a hex string array

str2numendsWith — returns the number that are part of in a string

strarr2cell — converts a string array into a cell list of string

strarr2char — converts a single row or single col string array into a char chain - no matrix

strbreaklines — Breaks a text string into paragraphs of a maximum number of chars per line

strcounter — detects an integer number at the end of a string and increments it

strfind2double — searches for pattern in char string and returns the next value as double real

strfindafter — finds text between start tokens and endtokens

strfindtag — returns a string after a search string

strhyperlink — returns a simple html hyperlink text for fprintf

strhyperlink2str — separates hyperlinks into real text and a list of keywords plus a list of urls

stringcellarray2charcell — converts xps cell list of string arrays into cell list of character chains

stringfnctresult — creates the cell list of output parameters of a batch call

stringpatterncompare — compares strings including variable strings

stringpatternpermute — creates symbol permutation for string pattern equations

strrepalias — replaces a set of similar words by a specified one

testparsing — performs an interactive test parsing of a grammar fnct

varsofstringpattern — finds all the symbolvariables in a string array

wordfind — finds words in a string (character chain)

xpcells2strings — converts cell list of character chains into one character chain of quotation marks separated sentences

xpstrings2cells — converts a char chain with quation mark separated sentences into a cell list of character chains

Detailed description

backspstr

returns a backspace character string to remove the given string

In: **str**: string to remove

Out: **bs**: backspaces to remove string

bug194

Removed (not converted) all 194 bugs in texts

In: **rrr**: textcode with potential 194 errors by wrong textread format

Out: **rrr**: rrr after conversion in latin1 and removed 194 bugs

cell2strarr

converts string arrays in cell lines into a string array

In: **ccc**: cell list of string lines **joinch**: if a joint char is used the string array has only one column

Out: **stra**: stringarray

cell2strlines

converts a cell list fo string tinto a single string line

In: **tab**: cell list of string **nl**: number of elements per line; default is 10000 **sing**: separate string by separation char such as """" or ''

Out: **outstr**: outstring

char2html

converts german "Umlaute" and other speciel characters into UTF8 html code

In: **tinp**: string including UTH chars

Out: **tout**: A character string in which the special characters have been converted into UTF-8.

char2strarr

converts a list of string inside a char chain into a string array

In: **chr**: char chain with quation mark separated strings **splt**: splitchar default is ""

Out: **strarr**: string array

charcell2stringcellarray

converts xps cell list of string arrays into cell list of character chains

In: **chrc**: cell array of char chains with quotation marks

Out: **strac**: cell array of string arrays

dbprintchar

similar to an ancient hex editor, this fnct plots codes and chars of a string or char array

In: **m**: string or char array

evalhyperlinktab

evaluates an hyperlink text input from a table

In: **tab**: hyperlink table **text**: list of words including optional string forcewww for executing 1st entry

fprintfvec2

prints vectors in a way that it can used in code generation

In: **f**: file pointer; default is '' **s**: format string **X**: number

hexstr2str

converts a uint8 string into a hex string array

In: **hs**: rows

Out: **text**: String array with [... x len] chars

hexstr2uint8

converts a hexadecimal string into a uint8 array

In: **STR**: string or char array with ot without spaces

Out: **U8**: Uint8 array of the hex data **CSHex**: Checksum in Hexadecimal

html2char

Converts HTML code in UTF8 into ASCII characters.

In: **tinp**: html UTF8 code

Out: **tout**: ascii code

isUTF8

Returns the number of codes that are not in the ASCII range 0..127 or the German umlauts.

In: **txt**: text to check

Out: **n**: number of special chars **io**: position in char(txt) **ochar**: uint8 of the special chars

latin1ofUTF

Provides a latin1 code for UTF so that the ASCII files are read correctly.

In: **txt**: uint8 text chain

Out: **c**: latin1 text chain

markdown2html

converts standard text or mardown text into html and pdf

In: **txt**: text or markdown file name **fname**: output file name; default is a timestamped PUB name with username

Out: **pdname**: full path of the generated PDF file

num2strCNT

creates numberstring including addition zeros

In: **i**: index **n**: maximal number of indices **txt**: optional leading string

Out: **nstr**: number string padded with leading zeros

num2strUP

creates a string containing a unit prefix

In: **instr**: number or string **unit**: number unit such as "Pa" or 's' or 'sec' **dig**: digits behind comma **alld**: if true, ending zeros are cut; default is false

Out: **sstr**: final number string

numelbytes

returns the number of bytes a variable allocates

In: **m**: variable

Out: **bytes**: number of bytes allocated

save2clipboard

simply saves the content of a variable to the "clipboard" to use it with paste

In: **var**: variable to be saved into the cipboard

sprintfhyperlinktab

creates a hyperlink text string from a table

In: **tab**: hyperlink table **nl**: maximum chars of per line; default is 120 **sort**: if true the items are sorted in alphabetic order; default is true

Out: **outstr**: hyperlink string **tab**: table with replace macros such as '@google'

sprintfilehyperlink

creates a string containing two hyperlinks to open an file

In: **fname**: full file name **appname**: application name used to open the file; default is empty **strname**: display string for the hyperlink instead of the file name; default is empty

Out: **stext**: hyperlinkstring

sprintftext

creates as simple as possible a string for design parameters

In: **textelementlist**: list of numbers, chars, and textlines

Out: **txt**: string to bu used for fprintf or SGpartslist

sprintfvec

prints vectors in a way that it can used in code generation

In: **f**: file pointer; default is '' **s**: format string **X**: number **sepchar**: separation char; default is ','

Out: **ostr**: output string **nstr**: naked string without brackets and semicolon

sprintfvecUP

string format for engineering applications

In: **vec**: vector with doubles **unit**: unit string such as 'mm' **dig**: number of digits; default is 2 **alld**: if true (default); ending zeros are cut **sep**: separator between numbers; default is '/' **unitall**: if true (false); all numbers get only a unit if they differ

Out: **si**: string of units

str2codelines

creates from long string, several lines that can be used as matlab code

In: **Str**: string to break into lines **ll**: maximum length of line **fle**: reduced number of chars if 1st line or string

Out: **clstr**: String broken into code lines

str2datenum

converts am formatted date string into a date num for numerical analysis

In: **txt**: text of date format **loc**: location default is 'de_DE' **add**: additional date format string to try; default is empty

Out: **dtn**: datenum **wfmt**: successful date format

str2hexsstr

converts a uint8 string into a hex string array

In: **text**: char array or uint8 array to convert into hex **len**: hex numbers per row ; default is 40 (80 chars)

Out: **TT**: String array with [... x len] chars

str2numendsWith

returns the number that are part of in a string

In: **ttt**: string or cell list of strings

Out: **nn**: rows with numbers for each string of the cell list

strarr2cell

converts a string array into a cell list of string

In: **strarr**: stringarray **rememt**: if true, empty entries are removed

Out: **ccc**: cell list with string entries

strarr2char

converts a single row or single col string array into a char chain - no matrix

In: **strarr**: string array **sep**: separator style; "" wraps each element in quotes joined by spaces, ',' joins with commas; default is ""

Out: **chr**: char chain that containts a string array

strbreaklines

Breaks a text string into paragraphs of a maximum number of chars per line

In: **InpStr**: Input String **numc**: number of chars per line

Out: **OutStr**: Output String

strcounter

detects an integer number at the end of a string and increments it

In: **fn**: original string **step**: step size **addn**: flag to append a '0' counter to names without a number; default is false

Out: **fni**: new string name **ostr**: original numbered string **cstr**: changed numbered string

strfind2double

searches for pattern in char string and returns the next value as double real

In: **TEXT**: ASCII TEXT string **codenr**: char patter to be matched

Out: **cr**: real number als double of the next chars up to space or newline

strfindafter

finds text between start tokens and endtokens

In: **txt**: text to search **srch**: str to find **endstr**: endstring **endnl**: if true new line is also accepted as endstr

Out: **ff**: index of starting positions **ee**: index of end positions **gg**: index of positions of the search token itself (start of match) **hh**: index of end positions including the end token **str**: last matched substring between start and end token

strfindtag

returns a string after a search string

In: **txt**: text **srch**: search string **endstr**: end tag to **endnl**: optional new line

Out: **nam**: trimmed name tag **nxt**: trimmed remainder of the text after the end tag

strhyperlink

returns a simple html hyperlink text for fprintf

In: **text**: text string **url**: url of link

Out: **hlstr**: html string to use even in fprintf

strhyperlink2str

separates hyperlinks into real text and a list of keywords plus a list of urls

In: **str**: string includes hyperlinks

Out: **str**: string without hyperlinks **wrds**: list of keywords used in in original string **urls**: list of urls used in in original string

stringcellarray2charcell

converts xps cell list of string arrays into cell list of character chains

In: **ops**: Input string-cell array **sep**: separator style for the char chain; default is "" (quote-wrapped, space-joined)

Out: **zzz**: Resulting char cell array

stringfnctresult

creates the cell list of output parameters of a batch call

In: **str**: function equation string

Out: **le**: list of narout **re**: equation

stringpatterncompare

compares strings including variable strings

In: **pattern**: pattern such as "move $A from $B to $C" **txt**: text to analyze such as "move A from B to C"

Out: **PAR**: List of parameters

stringpatternpermute

creates symbol permutation for string pattern equations

In: **equat**: string array of equation **symb**: string array of symbol names **samp**: cell list of samples for each symbol

Out: **VV**: permutation of the string pattern equations **LL**: permutation of unique symbols **CC**: index of LL that has been used for VV **kk**: number of valid permutations generated (count of unique-symbol substitutions applied)

strrepalias

replaces a set of similar words by a specified one

In: **str**: string **word**: {'animal', 'fox', 'dog', 'wolf', 'wolfhound', 'dog', 'hound'; 2nd line}

Out: **str**: modified string

testparsing

performs an interactive test parsing of a grammar fnct

In: **fname**: name of the grammar function such as jarvisgrammar **cmdstr**: optional command string for testing

varsofstringpattern

finds all the symbolvariables in a string array

In: **cond**: string array with terms that use $XXX as variable name

Out: **var**: string array with variable Name including %char **patt**: The parsed pattern template

wordfind

finds words in a string (character chain)

In: **txt**: charchain to search in **word**: word to look for **sep**: separators; default is {',' ' ';'}

Out: **pp**: array of hits **ccc**: separated words

xpcells2strings

converts cell list of character chains into one character chain of quotation marks separated sentences

In: **ccc**: cell list of sentences such as [{'space on a'} {'a on table'}]

Out: **sss**: sentences separate by quotation marks

xpstrings2cells

converts a char chain with quation mark separated sentences into a cell list of character chains

In: **sss**: sentences separate by quotation marks such as '"a on b" "b on c"'

Out: **ccc**: cell list of sentences such as [{'space on a'} {'a on table'}]

9.6.4 Web and External Data Sources [guide]

Functions in this section:

googleScholarSearch — performs a google scholar search for keywords and uses openal to process the web site content

Detailed description

googleScholarSearch

performs a google scholar search for keywords and uses openal to process the web site content

Out: **refs**: referneces as Harvard List **pdfLinks**: link to locally copied html of the google page

9.6.5 FileMaker Interface [guide]

Functions in this section:

FMclose — closes a connection

FMgetData — returns all field values from a table of an open FileMaker database

FMgetFieldTabs — returns structure of the FielMaker database

FMhelp — returns a help text for the FileMaker-Matlab interface

FMinitJDBC — checks the existence of the FileMaker JDBC driver and copies it into the javaclasspath

FMinsertSQLcmd — inserts an additional dataset into a FM's database's table

FMopen — opens a FileMaker Database by the FileMaker Application

FMselectSQLcmd — returns a SQL string for a database request in a intuitive order and syntax

FMsqlQuery — returns the answer to a SQL request to a FileMaker database

Detailed description

FMclose

closes a connection

In: **conn**: Connection to FileMaker Database

FMgetData

returns all field values from a table of an open FileMaker database

In: **conn**: Connector to Database **fieldname**: Field Name **tabname**: Table Name

Out: **ans**: fetched query result (field values from the table)

FMgetFieldTabs

returns structure of the FielMaker database

In: **conn**: Connector to Database

FMhelp

returns a help text for the FileMaker-Matlab interface

FMinitJDBC

checks the existence of the FileMaker JDBC driver and copies it into the javaclasspath

In: **cname**: full path and filename to "fmjdbc.jar". default is searching in searchpath

FMinsertSQLcmd

inserts an additional dataset into a FM's database's table

In: **xxx**: curly bracket term {'Fieldname 1',value1,'Fieldname 2',value2, ...}

Out: **inscmd**: SQL Command in which the term $$FMTABNAM$$ must be replaced by the table name of the database

FMopen

opens a FileMaker Database by the FileMaker Application

In: **fname**: filename **user**: user name default is Admin **passw**: password; default is ''

Out: **conn**: connector. **err**:

FMselectSQLcmd

returns a SQL string for a database request in a intuitive order and syntax

In: **FMselectfield**: Single Search Field name or a list of fields in curly brackets **FMselectwords**: search term conditions name or a list in curly brackets **FMgetfield**: Single return field; or a list of fields in curly brackets; default is FMselectfield **"OR"**: if used, multiple conditions are connected by OR **"AND"**: if used, multiple conditions are connected by AND; default

Out: **SQLcmd**: SQL Command in which the term $$FMTABNAM$$ must be replaced by the table name of the database

FMsqlQuery

returns the answer to a SQL request to a FileMaker database

In: **conn**: Connector to a FileMaker database **sqlreq**: sql request

Out: **ans**: answer **curs**:

9.6.6 Encryption [guide]

9.7 Video, Audio and Media [guide]

9.7.1 Video [guide]

Functions in this section:

createvideoscript — fnct to create video tutorials of text, videos, figures

imageaudio2video — creates video clips from an image and an audio file with the same name

imageFigureMovie — Creates a cell list of frames to write a movie later

imageVideoEndtitle — creates an endtitle image for a video

imageVideoFrames — extracts images from a video file

imageVideoImagePage — creates an coyright image for a video

imageVideoTitle — returns a video titel image for a given existing video

imageVideoWrite — writes an image several times into an open videostream

video2nosound — uses ffmpeg to create a movie copy with removed audio stream and clear filename

videoCopyCutMovies — copies part of a video and creates titles and end title

videoCopyFrames — copies the frame from Videoreader to Videowriter

Videoquickanimatego — rotates a set of graphics objects around axis in different speed radios

Videoquickcloseandopen — Simple concept to create videos from figures

videoquickmovecursor — fnct to move an curson symbol on the currect axes to a specified point

Videoquickrotate3D — records a manual movement of the 3D View using rotate3d by drawnowvid

Videoquickstart — Simple concept to create videos from figures

Videoquickwritegcf — Simple concept to create videos from figures

Videoquickwritelastcommand — writes in a video the line of the last command

Videoquickwritetextpage — Simple concept to create videos from figures

videoWriteClipMovie — creates a video from images and videos without sound

Detailed description

createvideoscript

fnct to create video tutorials of text, videos, figures

In: **fname**: filename of script file **dbug**: **time**: **WName**:

imageaudio2video

creates video clips from an image and an audio file with the same name

In: **IName**: Image file full name **isize**: number of rows or [c r]; default is 1080 **AName**: Audiofilename or number of default time seconds (45 Sec)

Out: **WName**: Name of AVI file with audio track

imageFigureMovie

Creates a cell list of frames to write a movie later

In: **figh**: 'start', 'end', or gcf or figure handle **FName**: Optinal filename for command 'save'

Out: **movie**: [] or movie if figh=='end; **FName**:

imageVideoEndtitle

creates an endtitle image for a video

In: **xy**: [x y] or filenamepath of video **ETitles**: EndTitle single row! **cols**: ['w','k'] is default

Out: **I**: Image to write **WName**:

imageVideoFrames

extracts images from a video file

In: **xy**: filename of the video file **ptime**: array or cells of time; default is 0

Out: **Ic**: cells of images **vr**: video reader object

imageVideoImagePage

creates an coyright image for a video

In: **xy**: [x y] or filenamepath of video **I**: Image to include **ETitles**: EndTitle single row! **cols**: ['w','k'] is default

Out: **I**: Image to write

imageVideoTitle

returns a video titel image for a given existing video

In: **xy**: [x y] or existing video filepath **STitles**: {'Title','Subtitle','Author','$Date'} **cols**: default is ['w' 'k'] **ptime**: snapshot frame time; default is 0 **WName**:

Out: **I**: Titel Image **WName**: Images of ptime

imageVideoWrite

writes an image several times into an open videostream

In: **v**: videoWriter **I**: Image; default is getframe(gcf) **n**: number of frames; default is 1

video2nosound

uses ffmpeg to create a movie copy with removed audio stream and clear filename

In: **fname**: full file name to modify **"force"**: if used, an existing (noaudio) file will be overwritten if it exists

videoCopyCutMovies

copies part of a video and creates titles and end title

In: **WName**: Created Filename **RName**: {'REadfile',[0 inf],{'Titles'},{EndTitles},[0,1,2]} **style**: ['w' 'k' 'Bank Gothic' '0.8] **skip**:

videoCopyFrames

copies the frame from Videoreader to Videowriter

In: **v**: videoWriter **vr**: videoReader

Videoquickanimatego

rotates a set of graphics objects around axis in different speed radios

In: **dw**: [steps angle]; default is [60 2*pi] **i**: {handle frame ratio style}; default is ['',eye(4),1 'rotz'] **i**: {handle frame ratio style}; default is ['',eye(4),1 'rotz'] **i**: {handle frame ratio style}; default is ['',eye(4),1 'rotz'] **"rewind"**:

Out: **hh**: array of handles

Videoquickcloseandopen

Simple concept to create videos from figures

In: **vw**: video writer

Out: **nn**:

videoquickmovecursor

fnct to move an curson symbol on the currect axes to a specified point

In: **p**: point coordinates **n**: number of frames in drawnowvid; default is 20 (1 second) **col**:

Videoquickrotate3D

records a manual movement of the 3D View using rotate3d by drawnowvid

In: **lim**: waiting time before stop OR awt sequnce list for **org**:

Out: **awt**: List of view angles and time period

Videoquickstart

Simple concept to create videos from figures

In: **fname**: optional file name; default is name of caller **showlast**: if true; if called from cmdline, the last command is shown in the video

Out: **vw**: Video Writer

Videoquickwritegcf

Simple concept to create videos from figures

In: **vw**: video writer **I**: Image to write; default is I=getframe(gcf); **n**: number of frames; default is 1

Videoquickwritelastcommand

writes in a video the line of the last command

Videoquickwritetextpage

Simple concept to create videos from figures

In: **vw**: video writer **text**: Title of video; default is name of caller **sec**: number of seconds; default is 0.8 **reso**: size; default is global Videoquick_vw size **FS**: optional font size; default is 18

videoWriteClipMovie

creates a video from images and videos without sound

In: **WName**: Filename; default is desktopdir **ImageOrFile**: Image or Filename **cduration**: number of seconds to display the image

9.7.2 Audio [guide]

Functions in this section:

audioeditspokentext — this keyboardbased fnct is to edit spoken text

audioreadTL — similar to audioread but with sound output after readin

audiorecorderTL — records an audio sample from a specified device

audiorecorderTLautostop — records an audio sample from a specified device

audiorecordspokentext — records a list of sentences an compiles it to a longer text

audioresample — resamples the audiosignal with a different sample rate

audioseparatebreaks — separates an audio sequence of spoken text into breaks

audiotrim — trims an audio signal and adds a short break

audiovocoder — changes the pitch of the signal without changing the length

audiowriteTL — writes an audio sequence with fixed frequency of 44100 Hz

soundTL — same as sound by support audio output devices

Detailed description

audioeditspokentext

this keyboardbased fnct is to edit spoken text

In: **y**: Original Signal **FS**: Sample Rate **DEV**:

Out: **y**: Final Text

audioreadTL

similar to audioread but with sound output after readin

In: **AName**: Filename; User interaction required if empty **nFS**: desired sample frequency

Out: **y**: Signal **FS**: Sample Frequency

audiorecorderTL

records an audio sample from a specified device

In: **DV**: device string' default is webcam **FS**: Sampe rate; default is 16000 **tim**: time to samplel default is 1 sec **SH**: if true; a time scale is displayed; default is true

Out: **y**: Signal **FS**: Sample rate

audiorecorderTLautostop

records an audio sample from a specified device

In: **DV**: device string' default is webcam **FS**: Sampe rate; default is 16000 **damp**: damping for stopping recorning; default is 0.1 **SH**: if true; a time scale is displayed; default is true **tim**:

Out: **y**: Signal **FS**: Sample rate

audiorecordspokentext

records a list of sentences an compiles it to a longer text

In: **org**: Signal to append; default is empty **FS**: Sample rate default is 16000 **DV**: device default is 'pro x' **def**: sentence length; default is 10 seconds

Out: **B**: Final Sequqnce

audioresample

resamples the audiosignal with a different sample rate

In: **OSig**: Original Signal **FSo**: Original Sample rate; default is 44100 **FSn**: NEw Sample ratel default is 16000

Out: **NSig**: New Signal **FSn**: New Sample rate

audioseparatebreaks

separates an audio sequence of spoken text into breaks

In: **y**: Audio Signal **FS**: Sample Rate; default is 16000 **damp**: damping for break detection; linear = ; defalt 0.01 = -40dB **brkdur**: minimum break durance default is 0.3 seconds **minvoic**: minimal voice length; default is 1 second

Out: **Voice**: List of [start and end] samples for the voice segments **VLen**: length of the individual segments

audiotrim

trims an audio signal and adds a short break

In: **y**: samples **FS**: Sample rate **n**: seconds to add silence; default is 0.2 **thr**: threshold for silence; default is 0.01

Out: **y**: sample signal

audiovocoder

changes the pitch of the signal without changing the length

In: **OSig**: Original signal **acc**: pitch; default is 1.03

Out: **NSig**: New Signal **NSR**: New Sample rate; not used yet

audiowriteTL

writes an audio sequence with fixed frequency of 44100 Hz

In: **OSig**: Original Audio **FS**: Original Audio Frequency; writing rate is always fixed to 44100 Hz **FName**: Filenamep; default is EXP-YYYY-MM-DD

Out: **Aname**: Full filename of written file

soundTL

same as sound by support audio output devices

In: **y**: Samples **FS**: Sample rate **DV**: Device Name string to use **SH**: if true; a time scale is displayed; default is true

9.8 Tutorials and Worked Examples [guide]

9.8.1 Guided Tutorials [guide]

Functions in this section:

HELP_SGLIB_AutomaticDesignofConnetingSolids — Basic tutorial for the SG-Lib on designing solids to connect to points/frames,surfaces

HELP_SGLIB_AutomaticLinkJointConnection — Basic tutorial for the SG-Lib on designing screwed assembly connnections

HELP_SGLIB_BooleanConcatenationofSolids — Basic tutorial for the SG-Lib on merging an fusion of solids

HELP_SGLIB_ClosedContoursPolyboolPolyshape — Basic tutorial for the SG-Lib on the foundations of closed contours for surface design

HELP_SGLIB_Datatypes — Basic tutorial for the SG-Lib on data types and abbreaviations

HELP_SGLIB_DesignofRobotArms — Basic tutorial for the SG-Lib for designing robot arms with serial kinematics

HELP_SGLIB_Joints4arbitraryContours — TUTORIAL to explain how to create arbitray parameterized linkages from joint coordinates and link-joint relationships

HELP_SGLIB_Joints4CPLopenChains — TUTORIAL to explain how to insert a list of joints into a contour to create an open kinematic chain printable in 3D

HELP_SGLIB_RelativeSpatialRelationofSolids — Basic tutorial for the SG-Lib on relative spatial relations

HELP_SGLIB_Simple3DextrusionfromContours — Basic tutorial for the SG-Lib on creation on solid body geometries from extrusion of contours

HELP_SGLIB_SingleJoint4CPL — TUTORIAL to explain how to insert a single joint M1.5 or larger into a 3D printable struture

Detailed description

HELP_SGLIB_AutomaticDesignofConnetingSolids

Basic tutorial for the SG-Lib on designing solids to connect to points/frames,surfaces

HELP_SGLIB_AutomaticLinkJointConnection

Basic tutorial for the SG-Lib on designing screwed assembly connnections

HELP_SGLIB_BooleanConcatenationofSolids

Basic tutorial for the SG-Lib on merging an fusion of solids

HELP_SGLIB_ClosedContoursPolyboolPolyshape

Basic tutorial for the SG-Lib on the foundations of closed contours for surface design

HELP_SGLIB_Datatypes

Basic tutorial for the SG-Lib on data types and abbreaviations

HELP_SGLIB_DesignofRobotArms

Basic tutorial for the SG-Lib for designing robot arms with serial kinematics

HELP_SGLIB_Joints4arbitraryContours

TUTORIAL to explain how to create arbitray parameterized linkages from joint coordinates and link-joint relationships

HELP_SGLIB_Joints4CPLopenChains

TUTORIAL to explain how to insert a list of joints into a contour to create an open kinematic chain printable in 3D

HELP_SGLIB_RelativeSpatialRelationofSolids

Basic tutorial for the SG-Lib on relative spatial relations

HELP_SGLIB_Simple3DextrusionfromContours

Basic tutorial for the SG-Lib on creation on solid body geometries from extrusion of contours

HELP_SGLIB_SingleJoint4CPL

TUTORIAL to explain how to insert a single joint M1.5 or larger into a 3D printable struture

9.8.2 Worked Examples (VLFL_EXP) [guide]

Functions in this section:

exp_2018_12_16_plan_2r_rob — EXPERIMENT to explain relationship between planar 2R-robot and fourbar linkage

VLFL_EXP01

VLFL_EXP02

VLFL_EXP03

VLFL_EXP04

VLFL_EXP05

VLFL_EXP06

VLFL_EXP07

VLFL_EXP08

VLFL_EXP09

VLFL_EXP10

VLFL_EXP11

VLFL_EXP12

VLFL_EXP13

VLFL_EXP14

VLFL_EXP15

VLFL_EXP16

VLFL_EXP17

VLFL_EXP18

VLFL_EXP19

VLFL_EXP20 — Tutorial for creating SIMSCAPE/SIMMECHANICS diagrams from solid body geometries

VLFL_EXP21 — Creates a 4-bar-Linkage and Drives it actively

VLFL_EXP22

VLFL_EXP23 — Tutorial for automatic design of SG aut of simulation date

VLFL_EXP24

VLFL_EXP25

VLFL_EXP26 — Tutorial for using universal planar links

VLFL_EXP27 — Tutorial 27: fourbar-linkage 2-pose synthesis and linkage export for 3D printing

VLFL_EXP28

VLFL_EXP29

VLFL_EXP30

VLFL_EXP31

VLFL_EXP32

VLFL_EXP33

VLFL_EXP34

VLFL_EXP35

VLFL_EXP36

VLFL_EXP37

VLFL_EXP38

VLFL_EXP39

VLFL_EXP40

VLFL_EXP41

VLFL_EXP42

VLFL_EXP43

VLFL_EXP44

VLFL_EXP45 — Tutorial to demonstrate the power of SGofCPLcommand

VLFL_EXP46 — TUTORIAL 46 - Publishable Tutorial for creating a planetary gear using SGofCPLcommand

VLFL_EXP47

VLFL_EXP48 — This is a Tutorial to understand different types of transmission gears

VLFL_EXP49 — Tutorial to understand different types of transmission gears

VLFL_EXP50

VLFL_EXP51

VLFL_EXP52

VLFL_EXP53

VLFL_EXP54

VLFL_EXP55

VLFL_EXP56

VLFL_EXP57

VLFL_EXP58 — TUTORIAL to download STL files with the surface model of a human

VLFL_EXP59 — Publishable tutorial how to use the fnct SGcutTcylindricblade

VLFL_EXP60 — This is a Tutorial to understand different types of transformation and surfaces

VLFL_EXP61

VLFL_EXP62

VLFL_EXP63

VLFL_EXP64

VLFL_EXP65

VLFL_EXP66

VLFL_EXP67

VLFL_EXP68

VLFL_EXP69 — Tutorial on designing a 4-link linkage for a hook attachment (approx. 1 minute)

VLFL_EXP70 — Tutorial to show how to design walking machines in different design types

VLFL_EXP71 — Tutorial for creation a fourbar linkage that follows a line

VLFL_EXP72

VLFL_EXP73 — Tutorial for testing the chat fncts for the openAI interface

VLFL_EXP74 — Tutorial for testing the STEP exporter

VLFL_EXP77

VLFL_EXP78

VLFL_EXP79

VLFL_EXP80

VLFL_EXP81

VLFL_EXP81_joints — TUTORIAL TO SHOW THE USE OF THE JOINT CREATION BY CUTTING A CONTOUR INTO PIECES

VLFL_EXP99

VLFL_EXP_2019_09_25_Glide

VLFL_EXP_SnakeYK

Detailed description

exp_2018_12_16_plan_2r_rob

EXPERIMENT to explain relationship between planar 2R-robot and fourbar linkage

VLFL_EXP01

VLFL_EXP02

VLFL_EXP03

VLFL_EXP04

VLFL_EXP05

VLFL_EXP06

VLFL_EXP07

VLFL_EXP08

VLFL_EXP09

VLFL_EXP10

VLFL_EXP11

VLFL_EXP12

VLFL_EXP13

VLFL_EXP14

VLFL_EXP15

VLFL_EXP16

VLFL_EXP17

VLFL_EXP18

VLFL_EXP19

VLFL_EXP20

Tutorial for creating SIMSCAPE/SIMMECHANICS diagrams from solid body geometries

In: **SVG**:

VLFL_EXP21

Creates a 4-bar-Linkage and Drives it actively

In: **SVG**:

VLFL_EXP22

VLFL_EXP23

Tutorial for automatic design of SG aut of simulation date

In: **SVG**:

VLFL_EXP24

VLFL_EXP25

VLFL_EXP26

Tutorial for using universal planar links

In: **SVG**:

VLFL_EXP27

Tutorial 27: fourbar-linkage 2-pose synthesis and linkage export for 3D printing

In: **SVG**:

VLFL_EXP28

VLFL_EXP29

VLFL_EXP30

VLFL_EXP31

VLFL_EXP32

VLFL_EXP33

VLFL_EXP34

VLFL_EXP35

VLFL_EXP36

VLFL_EXP37

VLFL_EXP38

VLFL_EXP39

VLFL_EXP40

VLFL_EXP41

VLFL_EXP42

VLFL_EXP43

VLFL_EXP44

VLFL_EXP45

Tutorial to demonstrate the power of SGofCPLcommand

In: **SVG**:

VLFL_EXP46

TUTORIAL 46 - Publishable Tutorial for creating a planetary gear using SGofCPLcommand

In: **SVG**:

VLFL_EXP47

VLFL_EXP48

This is a Tutorial to understand different types of transmission gears

In: **SVG**:

VLFL_EXP49

Tutorial to understand different types of transmission gears

In: **SVG**:

VLFL_EXP50

VLFL_EXP51

VLFL_EXP52

VLFL_EXP53

VLFL_EXP54

VLFL_EXP55

VLFL_EXP56

VLFL_EXP57

VLFL_EXP58

TUTORIAL to download STL files with the surface model of a human

In: **SVG**:

VLFL_EXP59

Publishable tutorial how to use the fnct SGcutTcylindricblade

In: **SVG**:

VLFL_EXP60

This is a Tutorial to understand different types of transformation and surfaces

In: **SVG**:

VLFL_EXP61

VLFL_EXP62

VLFL_EXP63

VLFL_EXP64

VLFL_EXP65

VLFL_EXP66

VLFL_EXP67

VLFL_EXP68

VLFL_EXP69

Tutorial on designing a 4-link linkage for a hook attachment (approx. 1 minute)

VLFL_EXP70

Tutorial to show how to design walking machines in different design types

In: **SVG**:

VLFL_EXP71

Tutorial for creation a fourbar linkage that follows a line

In: **SVG**:

VLFL_EXP72

VLFL_EXP73

Tutorial for testing the chat fncts for the openAI interface

In: **SVG**:

VLFL_EXP74

Tutorial for testing the STEP exporter

In: **SVG**:

VLFL_EXP77

VLFL_EXP78

VLFL_EXP79

VLFL_EXP80

VLFL_EXP81

VLFL_EXP81_joints

TUTORIAL TO SHOW THE USE OF THE JOINT CREATION BY CUTTING A CONTOUR INTO PIECES

VLFL_EXP99

VLFL_EXP_2019_09_25_Glide

VLFL_EXP_SnakeYK

New and not yet assigned [guide]

Functions in this section:

bplot — plots grid boxes at a desired position

Bravaislattice — returns a vertex list for a Bravais lattice

bugexp_2020_09_13_PLofVL — This is definitely not a Matlab bug but a SG-Lib bug

bugexp_2021_04_13_PLcorrelate — EXPERIMENT to show a problematic Solid Design from 2 multi region CPLS

camangle — sets the camera angle and edjust the distance so that there is almost no change in the image

CPLofPosefourbarshaping — returns collision free path shaped contour for a specific fourbar linkage

CPLRplot — plots a linkage consisting of link contours and joint frames in 2D

dbprintf — like printf but shows the calling fnct and with dbprintmode also an hyperlink

exp_2018_08_17_ATMEGA256 — SCRIPT for analyzing drilling holes in an STL printed circuit board based on image projection

exp_2018_08_25_nonmanedge — EXPERIMENT to repair non manifold vertices of TWO OR MORE Surfaces

exp_2018_08_25_nonmanvert — EXPERIMENT to repair non manifold vertices

exp_2018_11_02_bladecut — EXPERIMENT SGpunchblade

exp_2018_11_24_SGcuterror — return a SG with two open boundaries resulting from SGcut

exp_2018_12_18_walking — Walking fourbar

exp_2018_12_19_walkingshape — EXPERIMENT to show the swept hull of the links

exp_2018_12_20_cuttingshape — EXPERIMENT to automatically cut a shape to fit into another shape during 4bar movement

exp_2018_12_22_CPL2CPLchain — EXPERIMENT to analyze a CPL to identify links and revolute joints of a mechanism

exp_2019_01_16_4barcases — EXPERIMENT THAT SHOWS THE MAIN CONFIGURATIONS OF A FOURBAR LINKAGE

exp_2019_01_27_chair — EXPERIMENT TO EXPLAIN THE DESIGN OF PATIENT INDIVIDUAL KINEMATIC OPTIMIZED CHAIRS

exp_2019_02_27_call_by_reference — Experiment to show how to implement call by reference in matlab

exp_2019_03_28_Shape — will be the fnct to automatically defined the effector shape

exp_2019_07_16_cap — CPLofCVLz(CVL,top/

exp_2020_01_27_speedtest — EXperiment zum Vergleich der VErfahren zur Separierung planarer Dreiecke

EXP_2020_02_17_accuracy — This experiment was used to detect error in coplanar and barycentric transformations

EXP_2020_02_21_PLFLofPLFLequidelaunay — Experiment on equidistant Delaunay retriangulation of a polygon by adding inset grid points

exp_2020_07_02_subtractjoint — Sucvessfull experiment to create a universal revolute joint subtraction part

exp_2020_07_31_LinkBallbearing — Experiment to print dummyplugs for bearing links

exp_2020_08_13_refresh — creates an experiment compared to the ones almost 8 years ago

exp_2020_08_17_fittings — EXPERIMENT To create a set of fitting cylinders with different slot sizes between the walls

exp_2020_08_18_snapper — This experiment is used to determine the excess length of snap hooks

exp_2020_09_03_link — Experiment to cut a solid into two pieces and to design a screw connection between both parts

exp_2020_09_19_select3Dsurface — EXPERIMENT how to select interactively FS, surface, and complete solids

exp_2020_09_27_Goldschienen — Experiment testing 0.9 mm gold pins as cheap hinge joints, generating STL sleeves of varying length

exp_2020_09_27_Scharniere — creates an experiment to check the desing of hinges

exp_2020_09_30_slidingbearing — EXPERIMENT TO test the size of the IGUS sliding bearings

exp_2020_10_03_KinTree — EXPERIMENT to learn how to set joint in an interactive mode

exp_2020_10_18_360degree — shows and creates a movie of two robot arm segments that are limited to 360 degree movements

exp_2020_12_31_pins — EXPERIMENT TO CREATE JOINTS FOR 3D PRINTING

exp_2021_01_05_box — EXPERIMENT to visualize the rotation of a revolute joint

exp_2021_01_05_pinrings — EXPERIMENT TO EXPLAIN THE USE OF SGpincountersink

exp_2021_01_07_bikechain — Inpressive Experiment to show how to print reliable joints with FDM printers

exp_2021_01_16_xblockfix — Addon- for Anycubic Predator

exp_2021_01_20_4barfranz — EXPERIMENT TO CREATE 4BAR LINKAGES FOR FRANZ IRLINGER

exp_2021_01_25_convolution — EXPERIMENT to exaplin the demographic change

exp_2021_01_26_insertRjoint — Motor experiment with 1/4 Inch connector, printer separatetly

exp_2021_01_27_insertRjoint — Motor experiment with 1/4 Inch connector, printer assembled

exp_2021_01_28_insertRjoint — Motor experiment with 1/4 Inch connector, printer assembled

exp_2021_01_31_link — EXPERIMENT to create a playfree double axial beared joint with a defined distance

exp_2021_02_03_boxjoint — EXPERIMENT to create hinges for a hollow box

exp_2021_02_07_Convolution — Fnct to axplaon convolution

exp_2021_02_09_FIRIIR — EXPERIMENT TO SHOW HOW TO DESIGN A FORWARD CONTROL SYSTEM FROM AN IMPULSE RESPONSE

exp_2021_03_06_Bearing — Experiment to create simple but practicable bearing solutions for rotating joints

exp_2021_03_10_rodlinkjaco — experiment building a rod link for a Jacobean-style mechanism

exp_2021_03_12_yannickcut — EXPERIMENT to modify the links of the Yannick robot

exp_2021_03_13_yannickcut2 — EXPERIMENT to modify the links of the Yannick robot

exp_2021_03_14_yannickcut3 — EXPERIMENT to modify the links of the Yannick robot

exp_2021_03_22_linkrodcoll — creates a set of STL files to build robots with inline

exp_2021_11_26_Holefix — Experiment to prove that holes/contours with different wall thicknesses can be sliced

exp_2021_12_01_Pose2synth — shows how to perform a 3 Pose Fourbar syntheses and a 2 Pose Fourbar syntheses and

exp_2021_12_27_walk — example to learn how to desgin a real walking machine

exp_2022_01_06_fourBarposesyntheses — EXPERIMENT TO SHOW THE USE OF THE FNCT fourBarposesyntheses

exp_2022_01_09_kerbonus — Experiment to validate dimensions of kerb konus inserts

exp_2022_01_16_servosample — EXPERIMENT TO EXPLAIN HOW TO ADD AUTOMATICALLY BEARING AND AN AXLE TO A SOLID

exp_2022_01_17_axle — EXPERIMENT TO SEE HOW TO USE SGdesignDrivenshaft and getfuncparamStr

exp_2022_01_17_Theo_Jansen — EXPERIMENT TO USE THE ORIGINAL SAND BEAST DIMENSIONS FROM THEO JANSEN OF 1991

exp_2022_01_19_Posechain — EXPERIMENT TO CREATE A RELATIVE MOVING FOURBAR LINKAGE AS FOOT FOR A WALKIN Mechanism

exp_2022_02_04_Layering — EXPERIMENT TO UNDERSTAND THE COMPLEXITY OF LAYERING A LINKAGE

exp_2022_04_19_numbers — experiment for formating numbers wrt exp(1), pi, 2^ as fractions

exp_2022_04_21_ADAM_MOTORS — EXPERIMENT TO SHOW THE SIMPLICITY OF INTEGRATING DRIVES INTO BODIES

exp_2022_04_25_Adam — EXPERIMENT JUST TO INSERT A MOTOR INTO ADAM

exp_2022_06_20_FDMcompare — comparison of printing time

exp_2022_12_05_learncamera — EXPERIMENT TO SHOW HOW TO MOUNT AN IMAGE IN SPACE

exp_2023_03_04_gripper — to show how to create fully automatic a complete 2 jaw gripper from an STL file

exp_2023_03_14_gears — EXPERIMENT TO SHOW HOW TO AUTMATIC DESIGN A MOTOR GEAR

exp_2023_12_21block — EXPERIMENT TO CREATE AND DETECT SITUATONS IN BLOCK WORLD PROBLEMS

exp_2024_06_09_gearboxhousing — creates a complete gear box including housing

exp_2024_06_15_Hookplate — Bayonet catch with radial hooks

exp_2024_06_16_CPLtonguegroove — EXPERIMENT THAT BECAME LATER SGdesignTongueGrooveRadial

exp_2024_07_20_Frame85 — Experiment to convert rotating movement into a translation movement

exp_2024_08_04_DIN13 — returns a screw and a nut for an arbitrary steep thread for testing different thread geometries

exp_2025_07_31_FMenglish — EXPERIMENT to use openAI for spell checking and comprehension testing of SG-Lib documentation in the FM database

exp_2025_08_17_SGLibGPTAlgorithmus — open the FM database of SGLib and the mfile/ and inserts a CHatGPT explaination of the code into html and Fm Database

exp_2025_08_20_updateFMmatlabcode — open the FM database of SGLib and inserts the original matlabcode from the m file

exp_2026_03_07_pianohinge — EXPERIMENT to cut a 2D contour arbitrarily with two mouse clicks (or an HT frame), insert a rotational joint, and then extrude it using a piano hinge.

exp_2026_03_08_Nakashima — EXPERIMENT for Dr. Shinsuke Nakashima to explain the use of the SGLib for 4Bar linkagr design of Tibiua/Femur

exp_2026_03_12_pianohingelist — EXPERIMENT how to integrqte several piono hinges into an extruded contour

exp_2026_03_13_linkages — Experiment to design linkage-lever hinges that minimize the profile of the rotating lever and integrate part of the hinge into a solid body

exp_2026_03_25_kinematic_analysis — EXPERIMENT TO TEST WHICH GRAPH methods ARE SUITABLE FOR KINEMATIC STRUTURE ANALYSIS

exp_2026_03_28_crosslink — EXPERIMENT TO CREATE CROSS LINK OR STAR TYPE LINKE USING PIANO HINGES

exp_2026_04_01_crosstable — AN EXPERIMENT TO ADDRESS THE CHALLENGES INVOLVED IN DESIGNING THE JOINTS FOR A CROSS TABLE

exp_2026_04_02_4Bar — EXPERIMENT TO CREATE A CLOSE 4BAR with integrated piano hinges

exp_2026_04_04_crosstable — AN EXPERIMENT TO ADDRESS THE CHALLENGES INVOLVED IN DESIGNING THE JOINTS FOR A BIG CROSS TABLE

exp_2026_05_13_boxmove4bar — EXPERIMENT TO SHOW HOW TO MODIFY A MULTI_LAYER 4 BAR LINKAGE INTO A PLANAR ONE

exp_2026_05_29_closed_linkage_joint — Script to find a linkage design bug

fprintfvec — auxiliary fnct for writing ALL vector elements in ONE SINGLE row!

FSselect — Selects features surfaces based on

getfuncparams — processes varargin list -

getkeyboardchar — wait for keyboardkey pressed

GPLauxgridpointsCPS — returns for a given 2D contour point list auxiliary points to define a plane surface

insideCPL — tests whether a point is inside of a closed contour in 2D!

isonVLFL — returns whether a point is on a surface

isT — returns whether a variable is a HT matrix or a list of HT matrices

kw — returns the calender week

pcodedirTL — creates a temporary directory for the creation of pcodes

PofBB — returns a point relative to the bounding box

PSplot — plots a polyshape but also a cell list of polyshapes

SGcast — class name cast fnct for SG library

SGcenter — centers a solid on the x y or z axis

SGlib — simply opens the WWW pages SGlib.org and Matlab central

SGofCPLzplugin — created extruded solids that can be pluged in if they are very long

SGselect3d — Interactive selection of a point, vertex, or facet in a axis

SGTconnect — connects the geometries of two solids based on surfaces that are identified by frame names

SGTcopy — copies Frames from one Solid into another one

uipolygonselect — returns the handles of the polygon near a point or the cursor position

VLFL_EXP82 — TUTORIAL TO SHOW THE USE OF THE JOINT CREATION BY CONNECTING CLOSED CONTOURS BY JOINTS

VLnearestNeighbor — returns the nearest points in VL1 to the points in VL2

XX — Creates comment lines including formated variables with a hyperlink including variables

Detailed description

bplot

plots grid boxes at a desired position

In: **VL**: Vertex list **bs**: Box size

Out: **h**: handle to graphics object

Bravaislattice

returns a vertex list for a Bravais lattice

In: **A**: [a1 ;a2 ;a3] three row vectors for the Bravais equation **SG**: Solid Geometry or Bounding Box to be filled by the lattice

Out: **VL**: Vertex list of the lattice

bugexp_2020_09_13_PLofVL

This is definitely not a Matlab bug but a SG-Lib bug

In: **VL**: vertex list if this function should test your VLFL problem **FL**: facet list if this function should test your VLFL problem

bugexp_2021_04_13_PLcorrelate

EXPERIMENT to show a problematic Solid Design from 2 multi region CPLS

camangle

sets the camera angle and edjust the distance so that there is almost no change in the image

In: **a**: Camera angle

Out: **a**: Camera angle

CPLofPosefourbarshaping

returns collision free path shaped contour for a specific fourbar linkage

In: **PS**: Pose and fourbar linkage struct including GPL and GAL **aps**: [solution, limiting order] **SCPL**: Contour to shape **rel**: CPL is moved relative to 'P'ose, 'C'rank, or 'Swing' or 'WORLD' **XCPLW**: Static obstacles in the environment **XCPLA**: Obstacle contours attached to Pose **XCPLR**: Obstacle contour moved with crank **XCPL1**: Obstacle contour moved with swing

Out: **CPL**: Collision free contour

CPLRplot

plots a linkage consisting of link contours and joint frames in 2D

In: **CPLR**: cell array of contours **T**: cell array of frames **wlim**: optional angle interval for frames **colstr**: optional color string default is 'brmcg' (good colors to read w/o 'k') **colalp**:

Out: **h**: handle to graphics objects

dbprintf

like printf but shows the calling fnct and with dbprintmode also an hyperlink

In: **var2base**: parameter such as of printf

exp_2018_08_17_ATMEGA256

SCRIPT for analyzing drilling holes in an STL printed circuit board based on image projection

exp_2018_08_25_nonmanedge

EXPERIMENT to repair non manifold vertices of TWO OR MORE Surfaces

In: **SG**: Solid Geometry

exp_2018_08_25_nonmanvert

EXPERIMENT to repair non manifold vertices

In: **SG**: Solid Geometry

Out: **SGc**: Resulting Geometries as cells

exp_2018_11_02_bladecut

EXPERIMENT SGpunchblade

In: **B**: Solid to be cut interactively

exp_2018_11_24_SGcuterror

return a SG with two open boundaries resulting from SGcut

Out: **SG**: Solid Geometry with two open edges resulting from SGcut

exp_2018_12_18_walking

Walking fourbar

In: **k**: position of the foot relative to the 4bar **r**: radius of the crank relative to link 1

exp_2018_12_19_walkingshape

EXPERIMENT to show the swept hull of the links

In: **k**: position of the foot relative to the 4bar **rp**: radius of the crank relative to link 1 **b**: position of base between minimum and maximum value **CPLT**: Contour attached to c

Out: **A**: Area touched by link 1 **B**: Area touched by link 2 **C**: Area touched by crank **D**: Area touched by contour CPLT **a**: swept contour of link 1 for the last pose (CPL) **b**: swept contour of link 2 for the last pose (CPL) **c**: swept contour of the crank for the last pose (CPL)

exp_2018_12_20_cuttingshape

EXPERIMENT to automatically cut a shape to fit into another shape during 4bar movement

In: **k**: position along the couple length of POI [0..1, 0..1] **rp**: radius relation to l1+l2 **b**: base point [0 .. 1) with respect to minimal and maximal distance **CPLT**: Moving Contour attached to POI **CPLX**: Crossing Contour that limits the shape of CPLT

Out: **CPLT**: Optimized Contour that with maximum area within CPLX

exp_2018_12_22_CPL2CPLchain

EXPERIMENT to analyze a CPL to identify links and revolute joints of a mechanism

In: **CPL**: CLosed Polygon line

Out: **SG**: cell list of link structures **CCPL**: edge list to connect the links **EL**: shape of the connecting joints **DCPL**: centers of the contours **lm**:

exp_2019_01_16_4barcases

EXPERIMENT THAT SHOWS THE MAIN CONFIGURATIONS OF A FOURBAR LINKAGE

exp_2019_01_27_chair

EXPERIMENT TO EXPLAIN THE DESIGN OF PATIENT INDIVIDUAL KINEMATIC OPTIMIZED CHAIRS

In: **PS**:

Out: **PS**:

exp_2019_02_27_call_by_reference

Experiment to show how to implement call by reference in matlab

In: **A**: Input Parameter that will be changed into a string by this function

exp_2019_03_28_Shape

will be the fnct to automatically defined the effector shape

In: **PS**: Pose including PS.GAL and PS.GPL **aps**: selected solution of PS.GAL **lim**: if true; the contours are limited to the limiting order 123 etc. **smooth**: CPLfft for smoothening the effector contour **optcrank**: if true the cank is shaped **optswing**: if true the swing is shaped

Out: **PS**: Final Pose including shaped effectors

exp_2019_07_16_cap

CPLofCVLz(CVL,top/

exp_2020_01_27_speedtest

EXperiment zum Vergleich der VErfahren zur Separierung planarer Dreiecke

In: **Nr**: Testnr. of VLsampletriangles **n**: number of tests; default is 10000

EXP_2020_02_17_accuracy

This experiment was used to detect error in coplanar and barycentric transformations

In: **bary**: false = cartesian, true = barycentric **z**: accuracy; default is 1e-15 (lowest limit! for PLofVLcoplanar)

Out: **PLT**:

EXP_2020_02_21_PLFLofPLFLequidelaunay

Experiment on equidistant Delaunay retriangulation of a polygon by adding inset grid points

In: **PL**: point list of the polygon vertices **FL**: facet list

exp_2020_07_02_subtractjoint

Sucvessfull experiment to create a universal revolute joint subtraction part

In: **A**:

Out: **C**: universal revolute joint subtraction part **A**:

exp_2020_07_31_LinkBallbearing

Experiment to print dummyplugs for bearing links

exp_2020_08_13_refresh

creates an experiment compared to the ones almost 8 years ago

Out: **X**:

exp_2020_08_17_fittings

EXPERIMENT To create a set of fitting cylinders with different slot sizes between the walls

exp_2020_08_18_snapper

This experiment is used to determine the excess length of snap hooks

exp_2020_09_03_link

Experiment to cut a solid into two pieces and to design a screw connection between both parts

In: **orient**: **R**: **TT**:

exp_2020_09_19_select3Dsurface

EXPERIMENT how to select interactively FS, surface, and complete solids

In: **alp**: angle to select faces **mbf**: force button mode 1, 2, 3

Out: **VL**: Vertex list that belongs to touched graphics object **FL**: Facet list that belongs to touched graphics object **fil**: logical facet index list of selection **hp**: touched graphics object from select3d

exp_2020_09_27_Goldschienen

Experiment testing 0.9 mm gold pins as cheap hinge joints, generating STL sleeves of varying length

exp_2020_09_27_Scharniere

creates an experiment to check the desing of hinges

exp_2020_09_30_slidingbearing

EXPERIMENT TO test the size of the IGUS sliding bearings

In: **D**:

exp_2020_10_03_KinTree

EXPERIMENT to learn how to set joint in an interactive mode

Out: **TL**:

exp_2020_10_18_360degree

shows and creates a movie of two robot arm segments that are limited to 360 degree movements

In: **t**: time for video length

exp_2020_12_31_pins

EXPERIMENT TO CREATE JOINTS FOR 3D PRINTING

In: **slot**: default is 0.1 (+ slfit('c')) **rund**:

exp_2021_01_05_box

EXPERIMENT to visualize the rotation of a revolute joint

exp_2021_01_05_pinrings

EXPERIMENT TO EXPLAIN THE USE OF SGpincountersink

exp_2021_01_07_bikechain

Inpressive Experiment to show how to print reliable joints with FDM printers

Out: **XXX**: Geometry of a bike chain

exp_2021_01_16_xblockfix

Addon- for Anycubic Predator

exp_2021_01_20_4barfranz

EXPERIMENT TO CREATE 4BAR LINKAGES FOR FRANZ IRLINGER

In: **LA**: Length A **LB**: Length B **LC**: Length C **LD**: Length D **ns**: solution angle' default is 1st **fi**: 'c' or 't' for fitting; default is 't'

Out: **SGall**: 4 Links and 6 Pins to create the linkage

exp_2021_01_25_convolution

EXPERIMENT to exaplin the demographic change

In: **num**: age value used to look up the population and mortality rate; default is 23

exp_2021_01_26_insertRjoint

Motor experiment with 1/4 Inch connector, printer separatetly

exp_2021_01_27_insertRjoint

Motor experiment with 1/4 Inch connector, printer assembled

exp_2021_01_28_insertRjoint

Motor experiment with 1/4 Inch connector, printer assembled

In: **rs**: radius of the **sl**: Distance between sleeve and Pin, default slfit('t') **a**: opening agnle; default is 60 degree

exp_2021_01_31_link

EXPERIMENT to create a playfree double axial beared joint with a defined distance

In: **rs**: oversize of rings **sl**: slot distance default is 'c' **a**: angle default is 90 degree

exp_2021_02_03_boxjoint

EXPERIMENT to create hinges for a hollow box

In: **FN**: Frame of SGbox suc ahs X+ **line**: cutting line in 'x' or 'y' **Rdz**:

Out: **CPLA**:

exp_2021_02_07_Convolution

Fnct to axplaon convolution

In: **num**: age values used to sample the distributions; default is 1:100

exp_2021_02_09_FIRIIR

EXPERIMENT TO SHOW HOW TO DESIGN A FORWARD CONTROL SYSTEM FROM AN IMPULSE RESPONSE

exp_2021_03_06_Bearing

Experiment to create simple but practicable bearing solutions for rotating joints

Out: **X**: Two solids, arranged for printing

exp_2021_03_10_rodlinkjaco

experiment building a rod link for a Jacobean-style mechanism

In: **L**: link lengths [L LF LB]; default is [50 -1 0] **sdl**: shaft dimensions [inner outer height width]; default is [6 13 5 3] **hex**: hex flag/size; default is 0 **fit**: fit flag; default is false **rod**: rod option; default is empty **LL**: ramp and straight lengths [straight ramp]; default derived from link length; a scalar is expanded

exp_2021_03_12_yannickcut

EXPERIMENT to modify the links of the Yannick robot

In: **SG**: Solid Geometry **d**: Straight distance in z **unite**: if true; function stries to combine the three elements again to one

Out: **SGN**: Result if there is only one solid; unite=true

exp_2021_03_13_yannickcut2

EXPERIMENT to modify the links of the Yannick robot

In: **SG**: Solid Geometry **T**: Cutting Frame **Td**: Straight distance in z or Relative link **unite**: if true; function stries to combine the three elements again to one

Out: **SGN**: Result if there is only one solid; unite=true **T**: Transformation matrix of click

exp_2021_03_14_yannickcut3

EXPERIMENT to modify the links of the Yannick robot

In: **SG**: Solid Geometry **T**: Cutting Frame **Td**: Straight distance in z or Relative link **unite**: if true; function stries to combine the three elements again to one **type**: transition shape between the two segments ('straight', 'smooth', 'rad', 'rmax', 'tan'); default is 'smooth'

Out: **SGN**: Result if there is only one solid; unite=true **T**: Transformation matrix of click

exp_2021_03_22_linkrodcoll

creates a set of STL files to build robots with inline

exp_2021_11_26_Holefix

Experiment to prove that holes/contours with different wall thicknesses can be sliced

In: **incr**: if true; the wall is increased by 1 mm; default is true;

exp_2021_12_01_Pose2synth

shows how to perform a 3 Pose Fourbar syntheses and a 2 Pose Fourbar syntheses and

exp_2021_12_27_walk

example to learn how to desgin a real walking machine

In: **PS**: Pose created by checkfourbar1Poseattachpermutation; default is automatic **nrs**: **AA**:

Out: **PS**: resulting pose **SG4bar**:

exp_2022_01_06_fourBarposesyntheses

EXPERIMENT TO SHOW THE USE OF THE FNCT fourBarposesyntheses

exp_2022_01_09_kerbonus

Experiment to validate dimensions of kerb konus inserts

exp_2022_01_16_servosample

EXPERIMENT TO EXPLAIN HOW TO ADD AUTOMATICALLY BEARING AND AN AXLE TO A SOLID

In: **SG**: Solid Geometry **FN**: Frame Name **sbb**: BEaring size; default [6 13 5 0]

Out: **SGN**: resulting solid with the automatically added bearing and axle

exp_2022_01_17_axle

EXPERIMENT TO SEE HOW TO USE SGdesignDrivenshaft and getfuncparamStr

In: **"debug"**:

exp_2022_01_17_Theo_Jansen

EXPERIMENT TO USE THE ORIGINAL SAND BEAST DIMENSIONS FROM THEO JANSEN OF 1991

exp_2022_01_19_Posechain

EXPERIMENT TO CREATE A RELATIVE MOVING FOURBAR LINKAGE AS FOOT FOR A WALKIN Mechanism

In: **PSA**: base pose set (already synthesized) used as reference for the relative poses **ss**: index of the synthesis solution to use; default is 1

Out: **PS**: resulting relative pose set with the synthesized four-bar linkage

exp_2022_02_04_Layering

EXPERIMENT TO UNDERSTAND THE COMPLEXITY OF LAYERING A LINKAGE

In: **PSX**: Pose struct including "solut" field **aps**: solution number **lay**: layering; currently only used for assembly, motion is fixed to [0 2 1 1] **R12**: Radius for the links

Out: **SG**: Final solid geometry

exp_2022_04_19_numbers

experiment for formating numbers wrt exp(1), pi, 2^ as fractions

In: **n**: number

Out: **fnn**: string of number

exp_2022_04_21_ADAM_MOTORS

EXPERIMENT TO SHOW THE SIMPLICITY OF INTEGRATING DRIVES INTO BODIES

exp_2022_04_25_Adam

EXPERIMENT JUST TO INSERT A MOTOR INTO ADAM

exp_2022_06_20_FDMcompare

comparison of printing time

exp_2022_12_05_learncamera

EXPERIMENT TO SHOW HOW TO MOUNT AN IMAGE IN SPACE

In: **TT**: View Direction of Kamera

Out: **h**: **TTN**:

exp_2023_03_04_gripper

to show how to create fully automatic a complete 2 jaw gripper from an STL file

In: **SG0**: Solid to be grasoed or readin of file

exp_2023_03_14_gears

EXPERIMENT TO SHOW HOW TO AUTMATIC DESIGN A MOTOR GEAR

In: **f**: ratio; default is 1/17 **a**: orientation **dh**: if used, enables debug plotting during design **"debug"**:

Out: **SG**: Just the gears

exp_2023_12_21block

EXPERIMENT TO CREATE AND DETECT SITUATONS IN BLOCK WORLD PROBLEMS

exp_2024_06_09_gearboxhousing

creates a complete gear box including housing

In: **rat**: gear ratio; default is 30 **dax**: distance of the axes; default is 25 **hh**: height of the gear wheels; default is 5 **AX**: [shaft-wrench shaft-edges axis-diameter axis edges **"STL"**: if used the gear box ist written as STL directory **"convex"**: if used the gear box housing ist convex hull shaped **"freewheel"**: contains the parameter for a freewheel and replaces the shaft by an axis for the 1st wheel

Out: **SGall**: Cell array with all solids

exp_2024_06_15_Hookplate

Bayonet catch with radial hooks

exp_2024_06_16_CPLtonguegroove

EXPERIMENT THAT BECAME LATER SGdesignTongueGrooveRadial

exp_2024_07_20_Frame85

Experiment to convert rotating movement into a translation movement

In: **"STL"**:

exp_2024_08_04_DIN13

returns a screw and a nut for an arbitrary steep thread for testing different thread geometries

In: **t**: Type "A" or "B" or "C" or "D" - not all are correctly implemented **p**: pitch; 0 => DIN13 **a**: 's' screw, or 'n' nut or 'b' both; default is both

exp_2025_07_31_FMenglish

EXPERIMENT to use openAI for spell checking and comprehension testing of SG-Lib documentation in the FM database

exp_2025_08_17_SGLibGPTAlgorithmus

open the FM database of SGLib and the mfile/ and inserts a CHatGPT explaination of the code into html and Fm Database

In: **fname**: function name as word **tlen**:

Out: **halgo**: html algothims de

exp_2025_08_20_updateFMmatlabcode

open the FM database of SGLib and inserts the original matlabcode from the m file

In: **fword**: function name as word

exp_2026_03_07_pianohinge

EXPERIMENT to cut a 2D contour arbitrarily with two mouse clicks (or an HT frame), insert a rotational joint, and then extrude it using a piano hinge.

In: **T**: T frame for cutting the contour; default is empty (separate by mouse clicks)

Out: **T**: Frames of mouse clicks

exp_2026_03_08_Nakashima

EXPERIMENT for Dr. Shinsuke Nakashima to explain the use of the SGLib for 4Bar linkagr design of Tibiua/Femur

In: **fFemur**: filename for Femur STL **fTibia**: filename for Tibia STL

Out: **PS**: Pose struct for synthesis and animation

exp_2026_03_12_pianohingelist

EXPERIMENT how to integrqte several piono hinges into an extruded contour

In: **T**: T frame for cutting the contour; default is empty (separate by mouse clicks)

Out: **T**: Frames of mouse clicks

exp_2026_03_13_linkages

Experiment to design linkage-lever hinges that minimize the profile of the rotating lever and integrate part of the hinge into a solid body

In: **wlim**: [wmin wmax] **"debug"**: if used,everything is in debug mode **"turn"**: turn angle, for testing the frame coordinate systems

exp_2026_03_25_kinematic_analysis

EXPERIMENT TO TEST WHICH GRAPH methods ARE SUITABLE FOR KINEMATIC STRUTURE ANALYSIS

exp_2026_03_28_crosslink

EXPERIMENT TO CREATE CROSS LINK OR STAR TYPE LINKE USING PIANO HINGES

exp_2026_04_01_crosstable

AN EXPERIMENT TO ADDRESS THE CHALLENGES INVOLVED IN DESIGNING THE JOINTS FOR A CROSS TABLE

exp_2026_04_02_4Bar

EXPERIMENT TO CREATE A CLOSE 4BAR with integrated piano hinges

In: **A0**: A0 of the 4Bar **B0**: B0 of the 4Bar **A1**: A1 of the 4Bar **B1**: B1 of the 4Bar **R**:

exp_2026_04_04_crosstable

AN EXPERIMENT TO ADDRESS THE CHALLENGES INVOLVED IN DESIGNING THE JOINTS FOR A BIG CROSS TABLE

exp_2026_05_13_boxmove4bar

EXPERIMENT TO SHOW HOW TO MODIFY A MULTI_LAYER 4 BAR LINKAGE INTO A PLANAR ONE

exp_2026_05_29_closed_linkage_joint

Script to find a linkage design bug

In: **w0**:

fprintfvec

auxiliary fnct for writing ALL vector elements in ONE SINGLE row!

In: **fh**: file handle **X**: vector **fs**: num format; default is '%d'

FSselect

Selects features surfaces based on

In: **SG**: Solid Geometry **FIL**: Cell list of facet indices, or normal vector for FSofSGnormal **rel**: list of descriptors, can also start with an angle of FIL is a normal vector **"array"**: if used, instead of a ccell list, an array of Face indices is returned **"debug"**: if used, the selection is shown

Out: **FILN**: unsorted cell list of FILs for the selected surfaces - just the **FLHoles**: unsorted cell list of FILs for hiles in the selected surfaces **SG**: The solid geometry (passed through)

getfuncparams

processes varargin list -

In: **n**: number of parameter varargin **args**: varargin **v**: default value **m**: default is false; if true, variable is filled with default values **b**: **b**: **b**:

Out: **nv**: value for variable or logical of found if there are two output arguments **ea**: true if argument was empty

getkeyboardchar

wait for keyboardkey pressed

In: **bycallbackfnct**: if true, a call bakc function is used, if false ginput; default is true;

Out: **c**: Captured keyboard character

GPLauxgridpointsCPS

returns for a given 2D contour point list auxiliary points to define a plane surface

In: **CL**: Contour list 2D [x y] **dx**: Maximum auxiliary points in x **dy**: Maximum auxiliary points in y

Out: **GPL**: Grid point list **k**: index whether k is inside or outside; use k=reshape(k,size(X)) **X**: X grid used for calucaltion **Y**: Y grid used for calucaltion

insideCPL

tests whether a point is inside of a closed contour in 2D!

In: **VL**: Vertex list of the contour! MUST BE CLOSED OTHERWISE START/END POINTS ARE NOT DETECTED **PL**: Point to check

Out: **tp**: +1==inside, -1==outside, 0== on the contour

isonVLFL

returns whether a point is on a surface

In: **VLA**: Vertex list **FLA**: Facet list **VL**: Vertex list to test

Out: **vi**: On Surface Index of VL

isT

returns whether a variable is a HT matrix or a list of HT matrices

In: **T**: 3x3 or 4x4 HT frame or array or cell list

Out: **ii**: result true or false

kw

returns the calender week

In: **dstr**: empty (returns the current kw), # (returns the dates) or date (returns the kw of the date)

Out: **w**: week, or dates

pcodedirTL

creates a temporary directory for the creation of pcodes

Out: **dname**: full path of the temporary pcode directory (date-stamped folder on the desktop)

PofBB

returns a point relative to the bounding box

In: **VLO**: Vertex list, or SG or graphics handle **loc**: location string 3 chars '+", '-', 'c' **turn**: minimal bounding box; relative to

Out: **p**: coordinate relative to bounding biox

PSplot

plots a polyshape but also a cell list of polyshapes

In: **ps**: polyshape or list of polyshapes

Out: **h**: handle to plat, ie. polygons

SGcast

class name cast fnct for SG library

In: **SG**: Any SG library data type

Out: **classname**: SG classname

SGcenter

centers a solid on the x y or z axis

In: **SG**: Solid Geometry **cstr**: center string such as 'xyz'

Out: **SG**: Centered Solid **P**: applied translation vector [x y z]

SGlib

simply opens the WWW pages SGlib.org and Matlab central

SGofCPLzplugin

created extruded solids that can be pluged in if they are very long

In: **CPL**: CPL consisting of a an inner and outer contour **z**: [zmin zmax]; default ist [0 length] **l**: [segement-length overlapp] default is [160 30]; **"debug"**: if the string "debug" is present, debug plots are shown **"side"**: value following "side"; a siderail cell spec added to the extruded solid; default is nan

Out: **SG**: cell list of segments

SGselect3d

Interactive selection of a point, vertex, or facet in a axis

In: **SG**: Solid geometry; if empty gca is used **tx**: optional menue header

Out: **p**: point **v**: vertex **vi**: vertex index **f**: facet **fi**: facet index **haxis**: gca

SGTconnect

connects the geometries of two solids based on surfaces that are identified by frame names

In: **SGA**: Solid A **SGB**: Solid B **TNA**: Frame Name A **TNB**: Frame Name B **d**: distance in z **rf**: refine flag passed to SGconnect2FS; default is true

Out: **SGC**: Final solid with copied frame names

SGTcopy

copies Frames from one Solid into another one

In: **SGA**: Solid to receive frames **SGB**: Solid to cope frames from **Framenames**: optional list of frames to copy **"transT"**: HT matrix applied to SGB before copying frames; default is eye(4)

Out: **SGA**: Modified Solid with unique framenames

uipolygonselect

returns the handles of the polygon near a point or the cursor position

In: **p**: explicit point or use cursor position if p is empty **styp**: graphics object type to search for; default is 'polygon'

Out: **h**: handle of relative polygons **hp**: handle of all polygons **pins**: polygons in which p is located **pnv**: polygons, pi is near a vertex **p**: the point used for selection (cursor position if input was empty)

VLFL_EXP82

TUTORIAL TO SHOW THE USE OF THE JOINT CREATION BY CONNECTING CLOSED CONTOURS BY JOINTS

VLnearestNeighbor

returns the nearest points in VL1 to the points in VL2

In: **VL1**: Vertex list or Point List [m x 3] **VL2**: Vertex list or Point List [n x 3]

Out: **nn**: Index in VL1 which is nearest to VL2 [n x 1] **nd**: distances from each VL2 point to its nearest VL1 point [n x 1] **pp**: coordinates of the nearest VL1 points [n x 3]

XX

Creates comment lines including formated variables with a hyperlink including variables

In: **comment**: comment line including space and $code$ execution