Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design

2014-11-17: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.1 required)
- 1. Creating and visualizing a simple base-contour by four 2D-points
- 2. Append a 3rd coordinate column to get a vertex list
- 3. Predefined functions for the generation of often used planar polygons (PL)
- 4. More predefined functions for planar polygons in 3D (VL)
- 5. Calculation of the surface of a convex polygon
- 6. Calculation of all surfaces of convex polygon-based 2.5D-solid-volumes
- 7. Graphical user interface for STL import, export, and viewing

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 1.1 required)

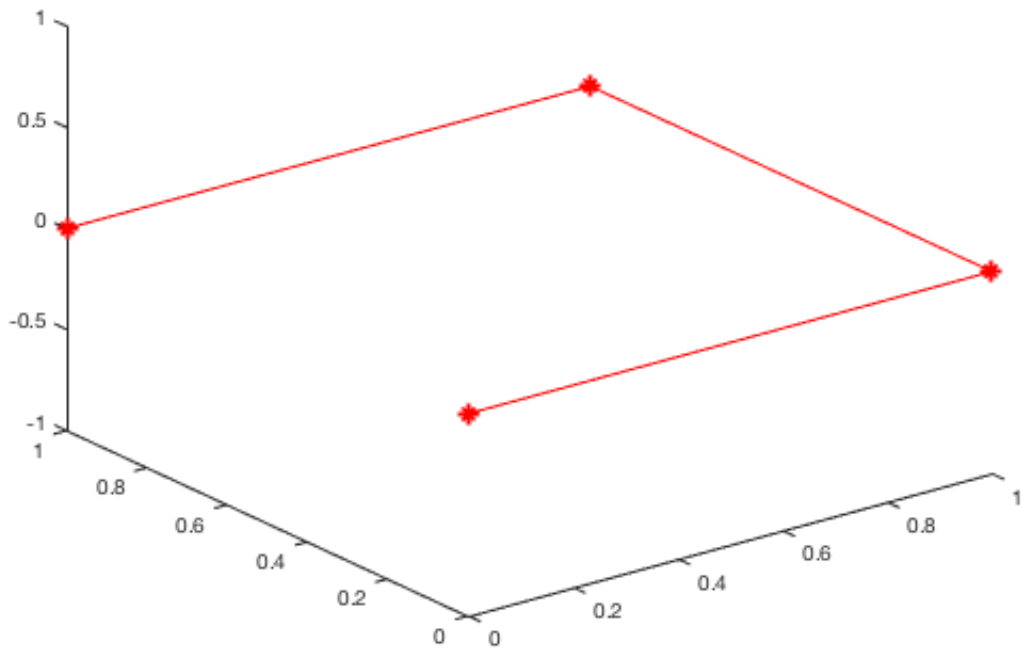
1. Creating and visualizing a simple base-contour by four 2D-points

A point list is a nx2 array. The number n is the number of 2D coordinate points [x y]. In general, such a point list can be the basis for designing a boundary surface model. We start with some simple functions to display polygons:

- **PLplot** to plot in 3D a nx2 point list (PL).

```
PL=[ 0 0; 1 0; 1 1; 0 1]
PLplot(PL);
```

```
PL =

     0     0
     1     0
     1     1
     0     1
```



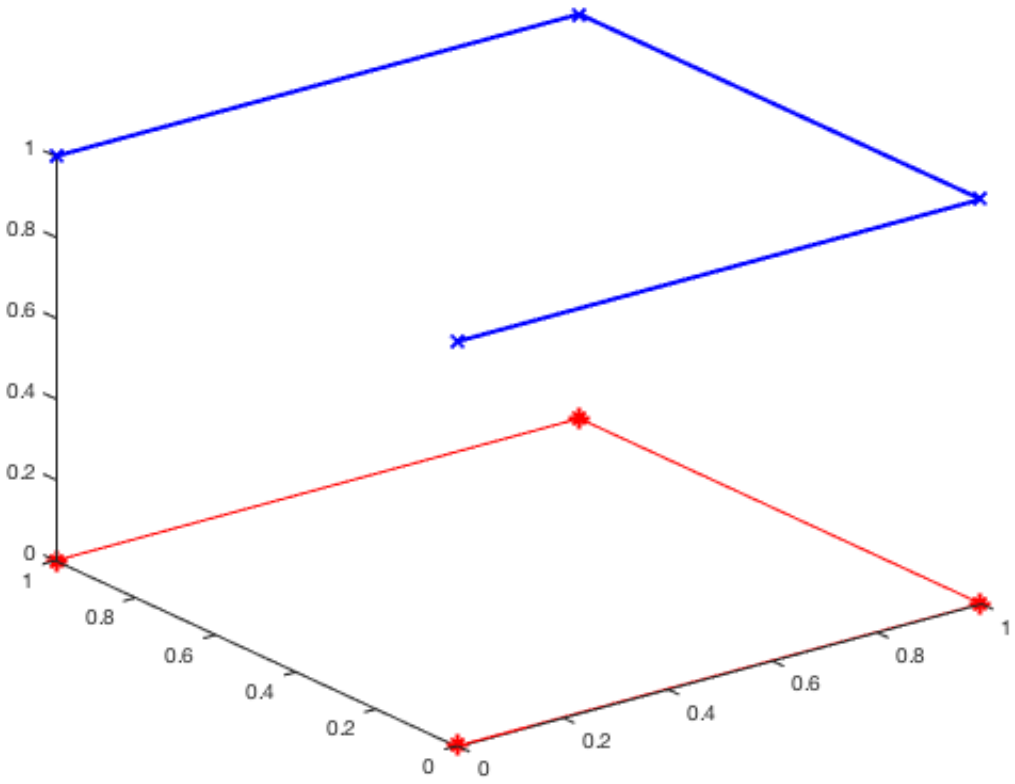
2. Append a 3rd coordinate column to get a vertex list

A vertex list is a nx3 array. The number n is the number of 3D coordinate points [x y z]. In fact, the point list can be transfered into a vertex list by appending a third column containing the z-coordinate such as zero or another z-coordinate.

- **VLaddz** for converting a point list (PL) into a vertex list (VL) by adding a 3rd column for the z-coordinate.
- **VLplot** for displaying in 3D a nx3 vertex list (VL).

```
VL=VLaddz(PL,1)
VLplot (VL,'bx-',2);
```

```
VL =

     0     0     1
     1     0     1
     1     1     1
     0     1     1
```

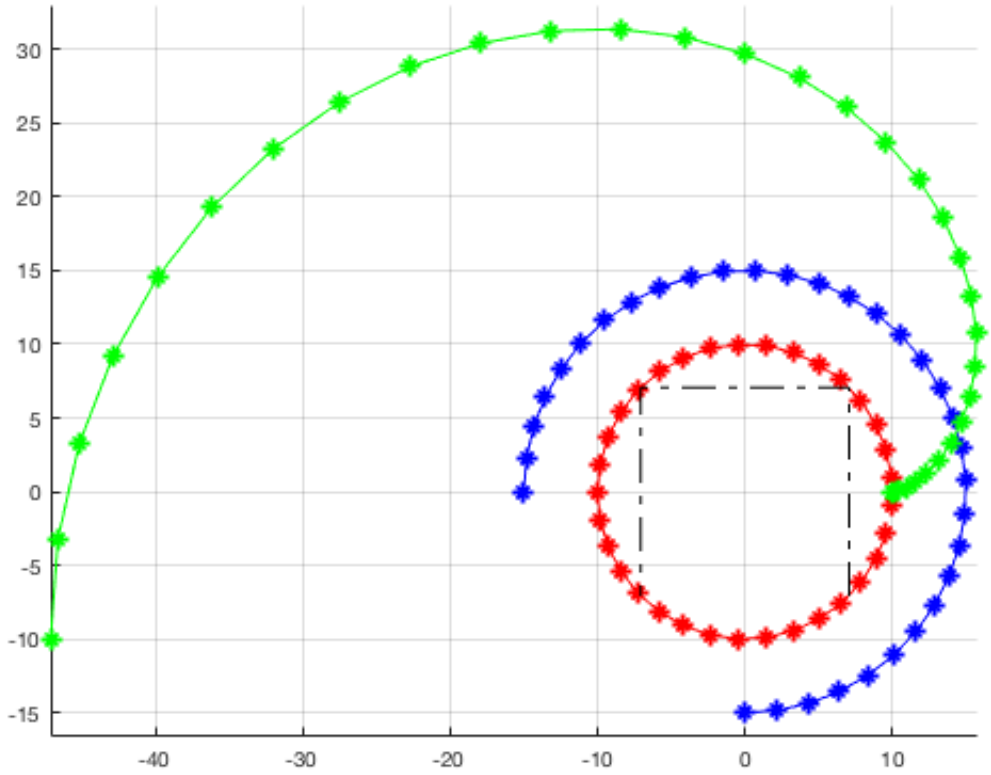


3. Predefined functions for the generation of often used planar polygons (PL)

For some often used contours, there are predefined functions that generate a nx2 coordinate point list (PL).

- **PLcircle** for a polygon with n points.
- **PLcircseg** for a circle segment with n points.
- **PLevolvente** for an evolvente as contour.

```
close all;
PL=PLcircle (10,33);                          % Radius 10, 33 points
PLplot(PL); view (0,90); axis equal; grid on;
PL=PLcircle (10,4);                          % Radius 10, 4 points
PLplot(PL,'k-.'); view (0,90); axis equal; grid on;
PL=PLcircseg (15,33,-pi/2,+pi);               % Radius 15, 33 points, 270 degree
PLplot(PL,'b-*'); view (0,90); axis equal; grid on;
PL=PLevolvente (10,pi/180*270,33)';           % Radius 10, 33 points, 270 degree
PLplot(PL,'g-*'); view (0,90); axis equal; grid on;
```

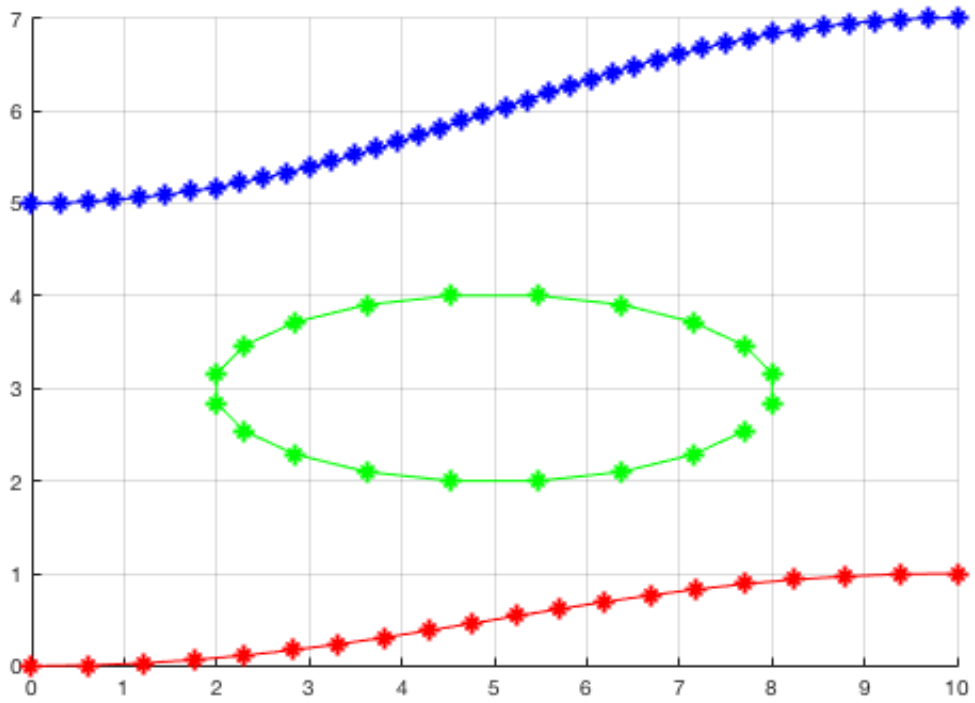


4. More predefined functions for planar polygons in 3D (VL)

Some functions for planar polygons create already 3D points (vertices) and the result of such a function is a vertex list (VL).

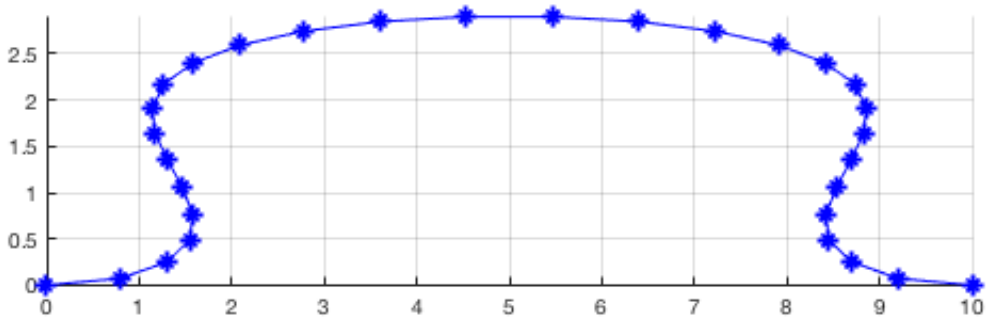
- **VLpolygon** to generate elliptic contours.
- **VLBezier4P** to generate a Bezier-curve using 4 points.
- **VLBezierC** to generate a Bezier-curve using as many points as possible.
- **VLremstraightCVL** to remove obsolete points on straight lines.

```
close all;
VL=VLpolygon(20,3,1,[5 3 0]);
VLplot (VL,'g*-'); show, axis equal, view (0,90); grid on; hold on;
VL=VLBezier4P([0 0 0],[4 0 0],[6 1 0],[10 1 0],20);
VLplot (VL,'r*-'); show, axis equal, view (0,90);
VL=VLBezierC([0 5; 4 5; 6 7; 10 7],40);
VLplot (VL,'b*-'); show, axis equal, view (0,90); grid on
```



- **VLBeziernoose** to generate a Bezier-curve spring-element.

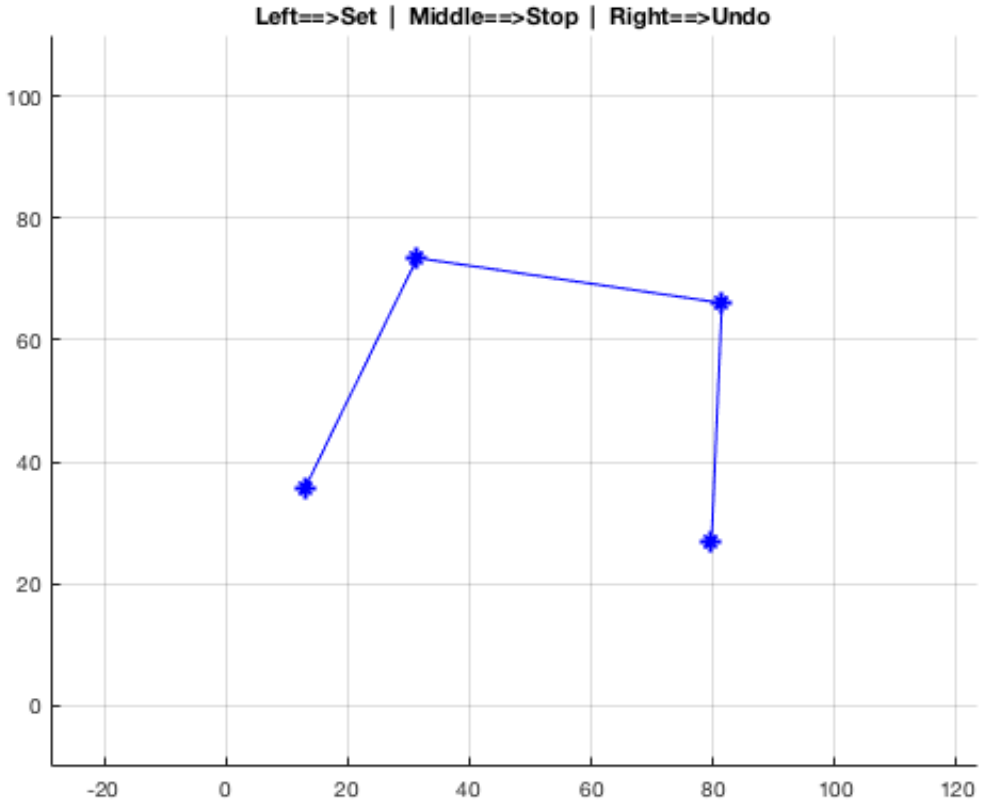
```
close all;
VL=VLBeziernoose(10,2,3,3,30);
VLplot (VL,'b*-'); show, axis equal, view (0,90); grid on
```



- **VLui** as an user interface to enter points by mouse clicks.

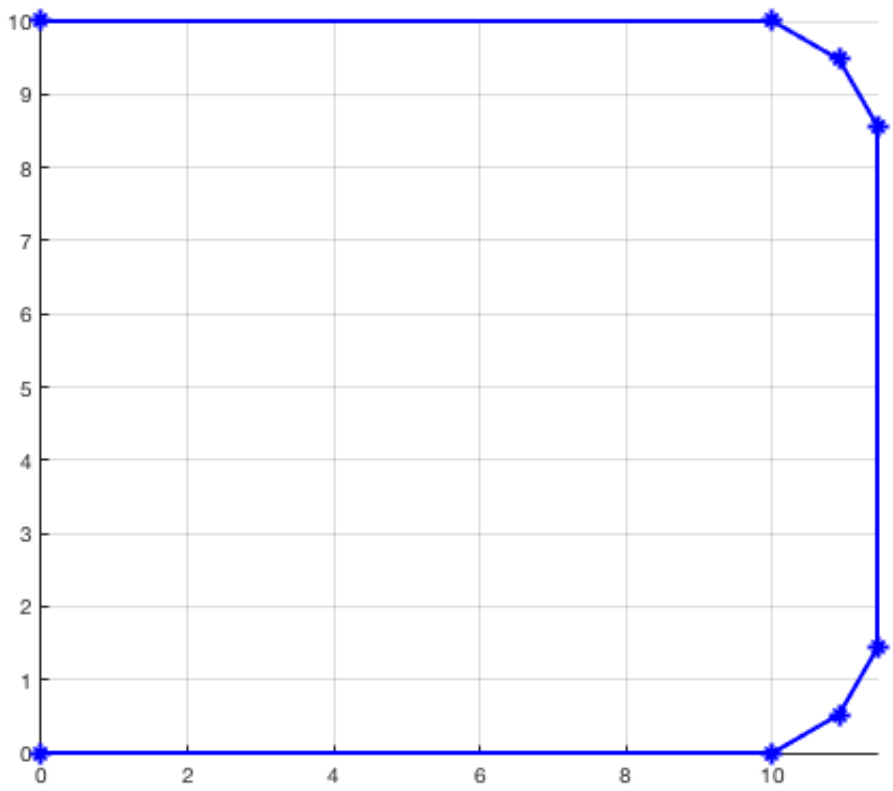
```
close all;
VL=VLui
VLplot (VL,'b*-'); show, axis equal, view (0,90); grid on
```

```
VL =

   12.9000   35.6000         0
   31.4000   73.5000         0
   81.5000   66.1000         0
   79.8000   26.9000         0
```



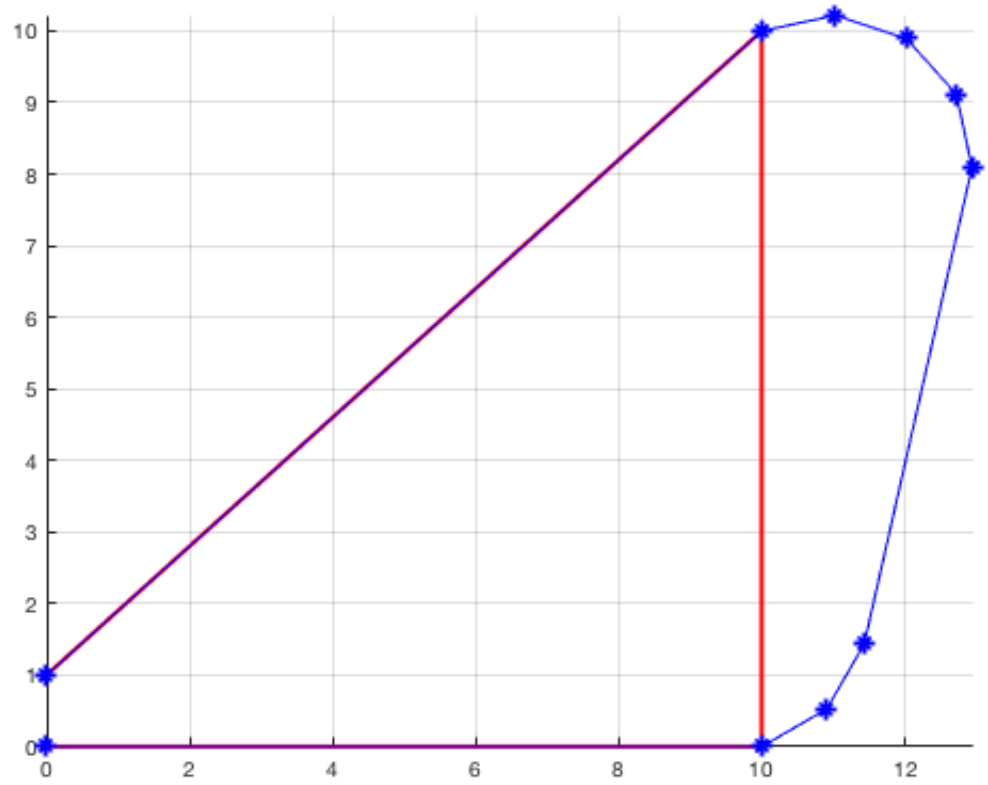
- **VLRadius4P** for inserting points to generate radial curves instead of corners.

```
close all
VL=VLRadius4P([0 0 0],[10 0 0], [10 10 0], [0 10 0], pi/6, 2);
VL=VLremstraightCVL (VL);
VLplot (VL,'b*-',2); show, axis equal, view (0,90); grid on
```



- **VLRadiusC** for inserting points to generate radial curves instead of corners.

```
close all
VLORG=[[0 0 0];[10 0 0];[10 10 0];[0 1 0]];
VLplot (VLORG,'r*-',2); show, axis equal, view (0,90); grid on; hold on
VL=VLRadiusC(VLORG, pi/6, 2);
VL=VLremstraightCVL (VL);
VLplot (VL,'b*-',1); show, axis equal, view (0,90); grid on
```



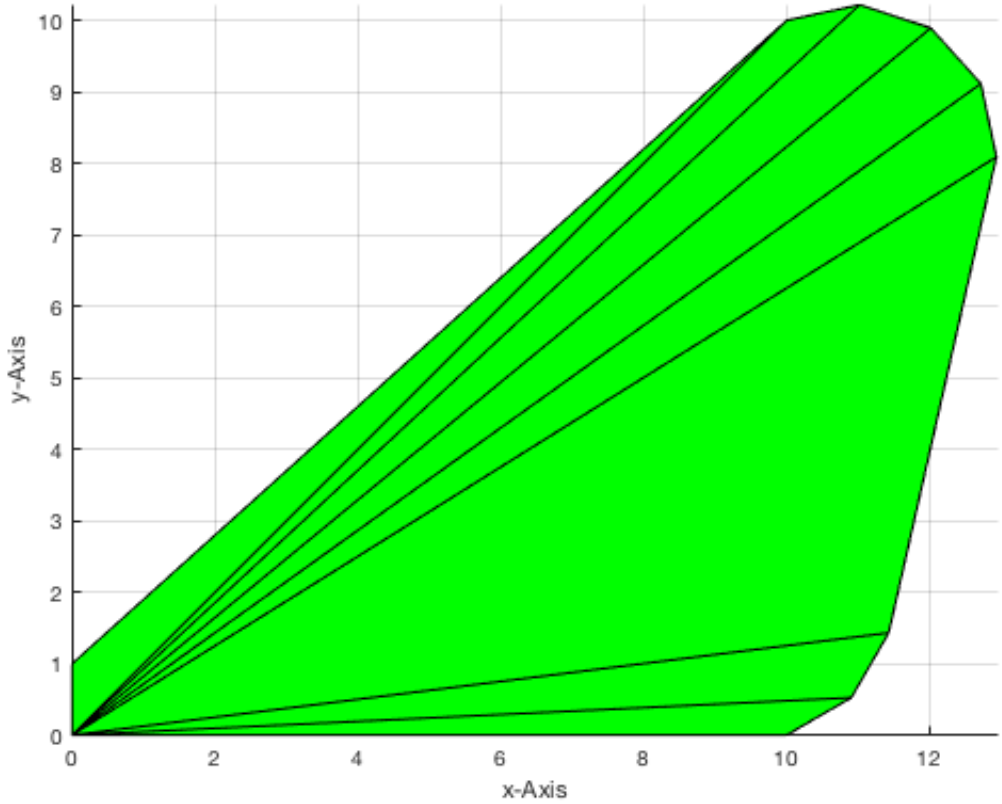
5. Calculation of the surface of a convex polygon

If we have a closed convex polygon, it is possible to generate a surface desciption by a facet list (FL) describing triangle facets. This is called tesselation of the closed polygon/surface. For closed convex polygons, the simplest form are facets from the 1st to the 2nd and 3rd points [1 2 3], then from the 1st to the 3rd and 4th [1 3 4], and so on. The facet list (FL) is therefor a nx3 index list to the point list or vertex list (VL). To use this concept we have some basic functions. For non convex functions we see later some more solutions.

- **FLofVL** to generate the facet list (FL) for a **convex** polygon.
- **VLFLplot** to plot a surface given by a vertex list (VL) and a facet list (FL).

```
close all
FL=FLofVL(VL)
% FL=FLofCVL(VL)
VLFLplot (VL,FL,'g'); axis equal; view (0,90); grid on
% view (-30,30);
```

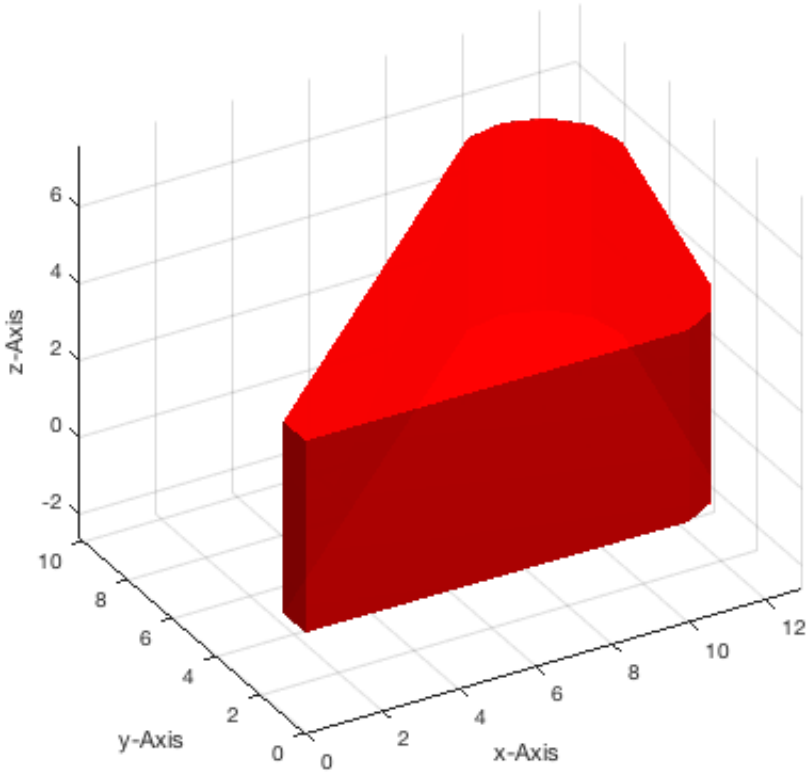
```
FL =

     1     2     3
     1     3     4
     1     4     5
     1     5     6
     1     6     7
     1     7     8
     1     8     9
     1     9    10
```



6. Calculation of all surfaces of convex polygon-based 2.5D-solid-volumes

- **VLFLofPLz** to extrude a convex polygon to a solid volume.
- **VLFLplotlight** to adjust the rendering parameter of the current graphic.

```
close all
[VL,FL]=VLFLofPLz (VL(:,1:2),5);
VLFLplot (VL,FL); axis equal; view (-30,30); grid on
VLFLplotlight(1,0.9); show;
```



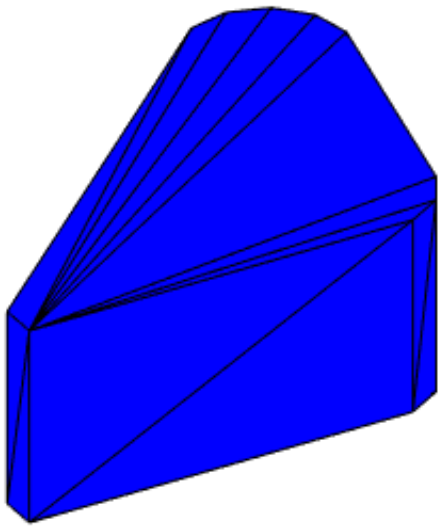
7. Graphical user interface for STL import, export, and viewing

Currently tested only for OSX (Apple Macintosh), there is also a graphical user interface available for displaying the surface objects, to import STL-Files and to export STL-Files. In this example, the tool is just introduced, to explain the capabilities to implement also graphical design tools for solid object modeling.

- **VLFLviewer** to show surface models, to import and to export STL-Files.

```
VLFLviewer (VL,FL,'b'); view (-30,30);
```

'VLFLviewer' : 16-Sep-2018 16:33:27



```
VLFLlicense
% * Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-18
% * Tim Lueth, executed and published on 64 Bit PC using Windows with Matlab 2014b on 2014-
11-18
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:33:28!
Executed 16-Sep-2018 16:33:30 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import

2014-11-18: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.1 required)
- 2. Import and export of STL-files in ASCII format and binary format
- 3. Checking surface volume data and STL-files
- 4. Generation of text, numbers, characters and formulas as solid volume
- 5. Turning and mirroring of solids by manipulating the vertex lists (VL)
- 6. Spatial transformation of solids by manipulating the vertex lists (VL)
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

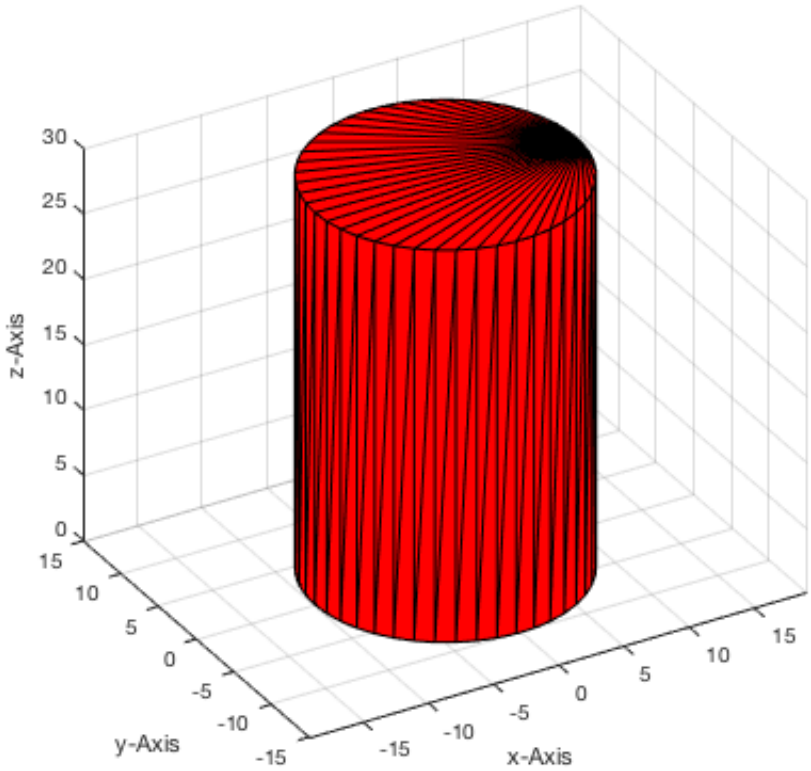
Motivation for this tutorial: (Originally SolidGeometry 1.1 required)

2. Import and export of STL-files in ASCII format and binary format

Often it is useful to import surface data fo solid volumes from STL-Files generated by other programs such as CATIA, ProEngineer, Solidworks etc. On the other hand we want to export our data for documentation, 3D-printing or the exchange with other users. The STL-File format is the most common file format for surface models. It supports ascii-text-format and binary formatted files. For export and import we need a couple of functions:

- **VLFLwriteSTL** for writing STL-files in ascii file format.
- **VLFLwriteSTLb** for writing STL-files in binary file format.

```
close all;
PL=PLcircle(10);
[VL,FL]=VLFLofPLz (PL,30);
VLFLplot(VL,FL); view (-30,30); grid on;
```




```
VLFLwriteSTL(VL,FL,'STL-ASCII','by My Name');
VLFLwriteSTLb(VL,FL,'STL-BINAR','by My Name');
```

```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/STL-ASCII.STL in ASCII MODE
completed.
WRITING STL (90 vertices, 176 facets) FILE /Users/timlueth/Desktop/Toolbox_test/STL-BINAR.S
TL in BINARY MODE
completed.
```

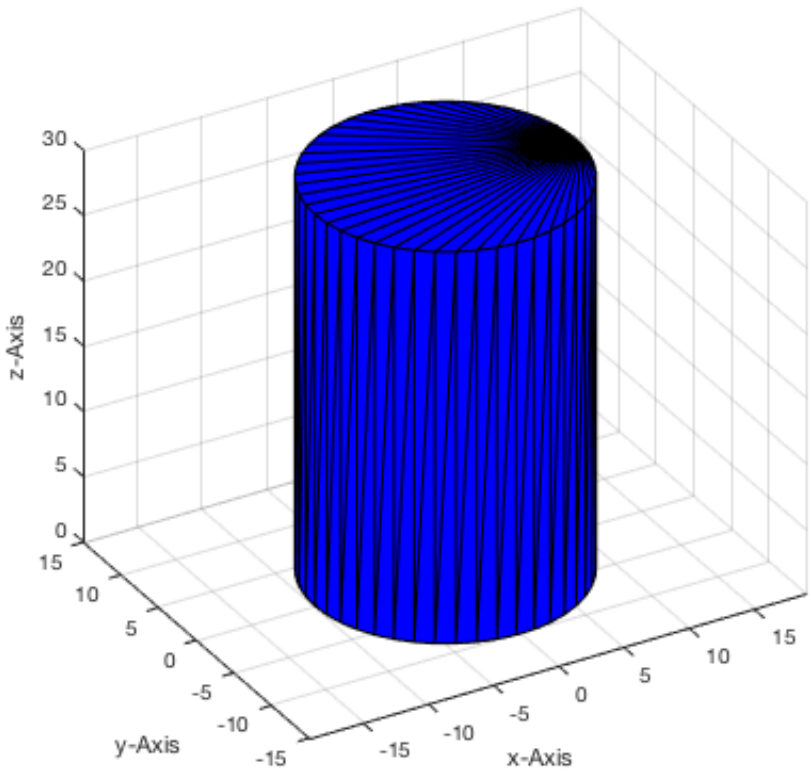
Similar it is possible to read the files in again

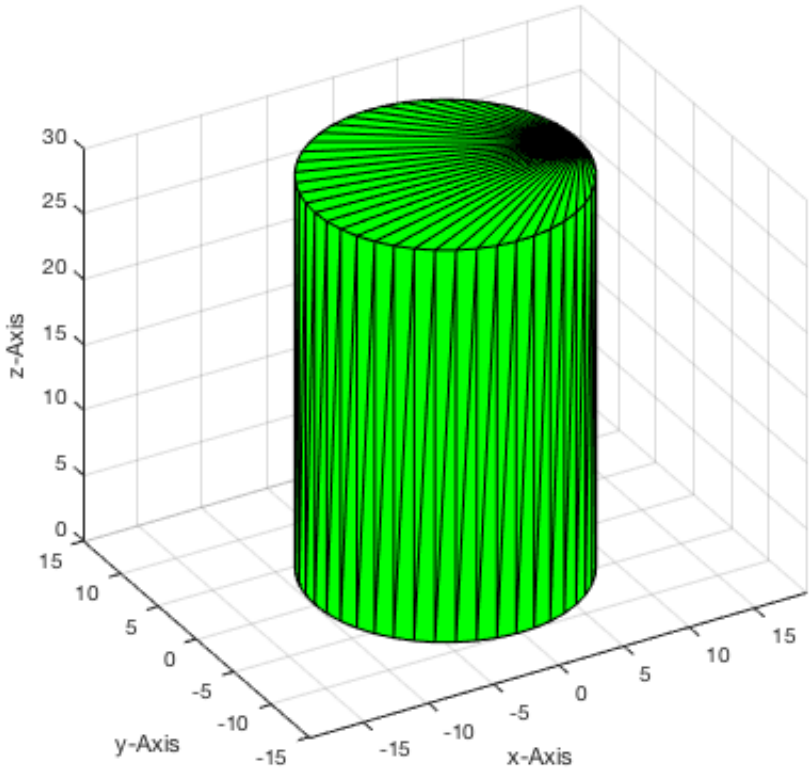
- **VLFLreadSTL** for importing STL-files in ascii file format.
- **VLFLreadSTLb** for importing STL-files in binary file format.

```
close all;
[VL,FL]=VLFLreadSTL ('STL-ASCII');
figure(1); VLFLplot(VL,FL,'b'); view (-30,30); grid on;
[VL,FL]=VLFLreadSTLb ('STL-BINAR');
figure(2); VLFLplot(VL,FL,'g'); view (-30,30); grid on;
```

```
LOADING ASCII STL-File: /Users/timlueth/Desktop/Toolbox_test/STL-ASCII.STL scaling factor:
1
Processing 1234 lines:
```

```
Finishing solid AOI-LIB:"STL-ASCII by My Name" 16-Sep-2018 16:33:32 16-Sep-2018 16:33:32 LO
ADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/STL-BINAR.STL
Header: AOI-LIB:"STL-BINAR by My Name" 16-Sep-2018 16:33:32
Number of facets: 176
0..
```





3. Checking surface volume data and STL-files

Especially, when reading in STL-Files that are generated by other programs and libraries it makes sense to check the data quality. For that purpose there is a function that will be explained later in more detail. This function is called at the end of each STL import.

- **VLFLchecker** is used to analyze vertex list (VL) and facet list (FL)

```
VLFLchecker(VL,FL);     % Check the data structure

% There are some more procedures to view and analyze solid volumee data
```

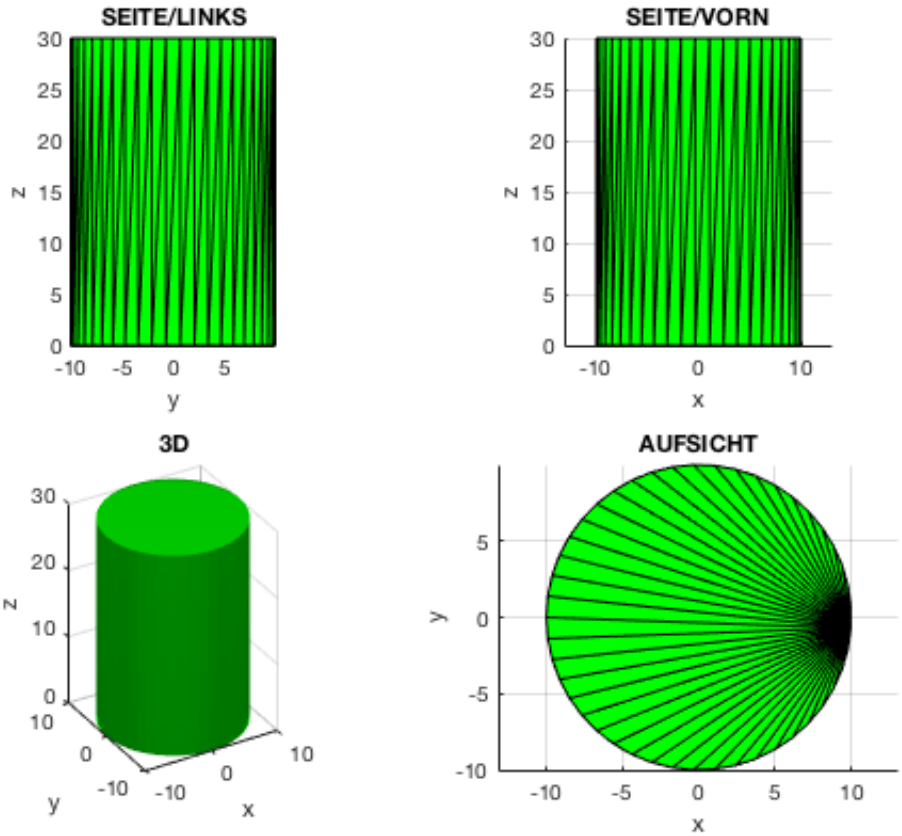
```
VLFLchecker: 90 vertices and 176 facets.
    0 FACET PROBLEMS DETECTED (ERRORS)
    0 VERTEX PROBLEMS DETECTED (OBSOLETE WARNING)
    0 EDGE PROBLEMS DETECTED (NON MANIFOLD WARNING)
    0 SOLID/EDGE PROBLEMS DETECTED (OPEN SOLID WARNING)
```

- **BBofVL** generates the bounding box dimensions of the solid
- **VLFLui** is simple user interface to open an STL file
- **VLFLminimize** eliminates doubles in VL and FL
- **VLFLnormf** calucates norm vector direction and size
- **VLFLplot4** figure with 4 subplots
- **VLFLselect** selected vertex list for a given facet list

- **VLFLseparate** find different independen objects in VL and FL
- **VLFLshort** remove unused vertices from VL
- **VLFLsurface** returns only vertex list and facet list for one surface
- **VLFLvertexfusion** shrinks vertex list by merging extremy near vertices

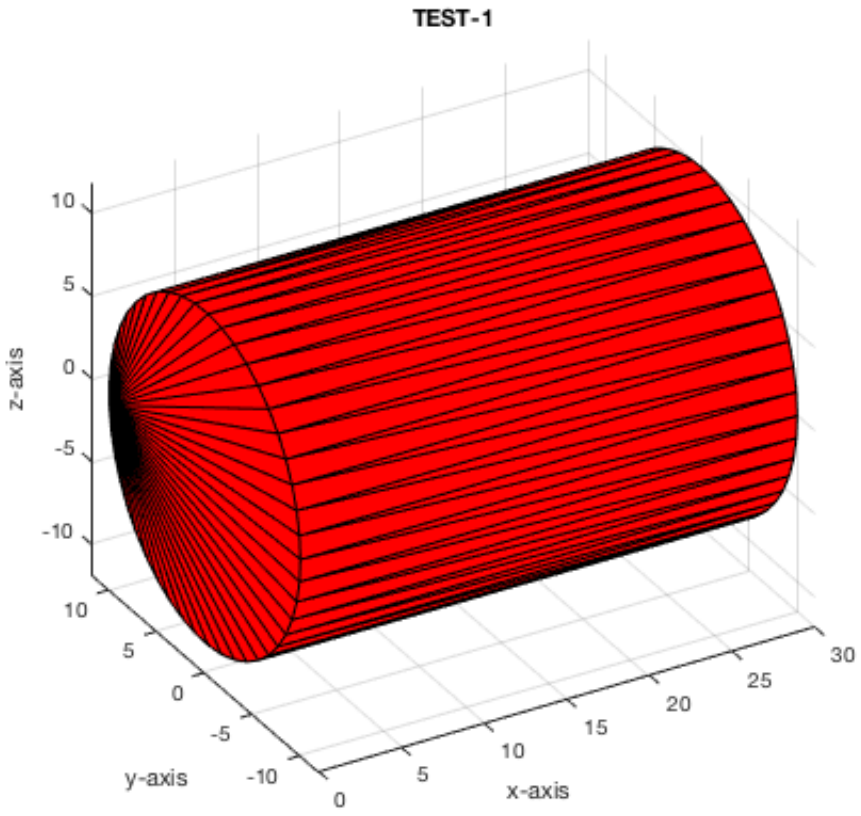
```
BBofVL(VL)
close all; VLFLplots4 (VL,FL,'g');
```

```
ans =

  -10.0000    9.9756   -9.9939    9.9939         0   30.0000
```




```
close all; VLFLseparate(VL,FL);
```

```
Analyzing 90 facets for separation z=[0.0mm|30.0mm]
Object TEST-1 with 176 facets

MVL =

     0   -10     0
```



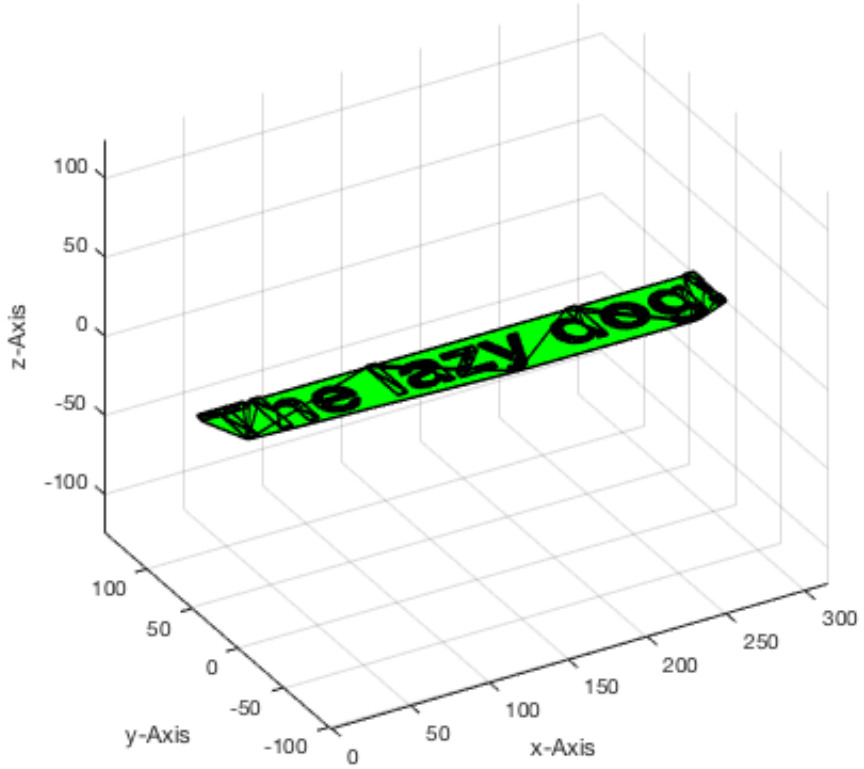
4. Generation of text, numbers, characters and formulas as solid volume

Often you want to write some numbers or code on top of a solid object. For that purpose there is a currently slow function that is able to convert a Matlab-string (even with LaTex-code) into a solid object.

- **VLFLtextimage** writes a line using the text command and converts it into a solid volume
- **VLFLtext** does the same for a very limited number of characters

```
close all;
[VL,FL]=VLFLtextimage('The lazy dog!');
VLFLplot (VL,FL,'g'); view (-30,30);

[VL,FL,d]=VLFLtext('TL-MMXI-XII-XVII');
VLFLwriteSTL (VL,FL,'exp_2011_12_17', 'by Tim C Lueth');
```

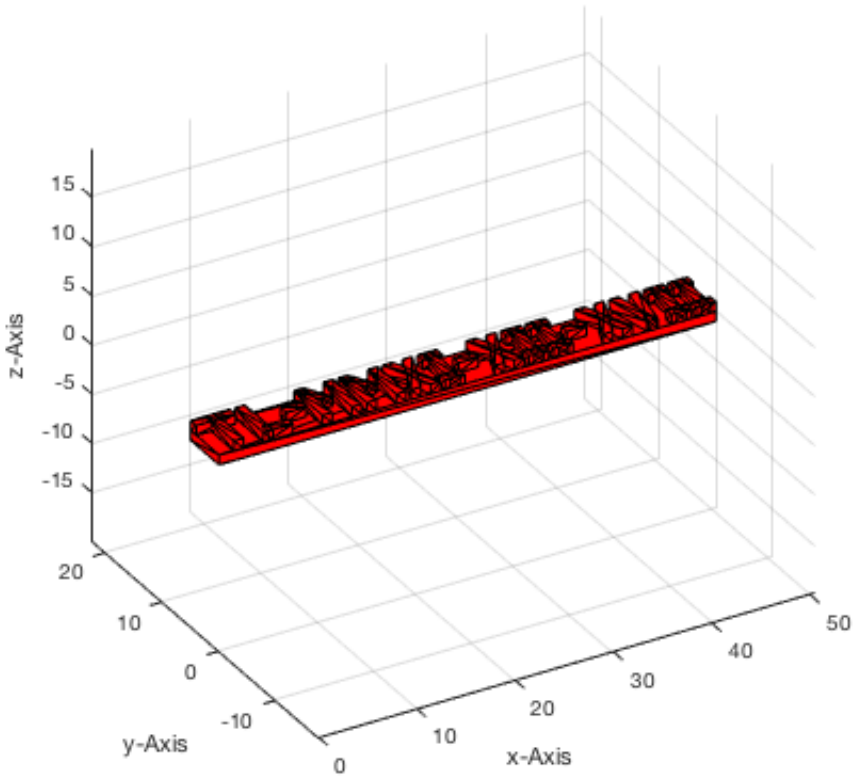
```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/exp_2011_12_17.STL in ASCII MODE
completed.
```



5. Turning and mirroring of solids by manipulating the vertex lists (VL)

Turning an object and mirroring is quite simple by exchanging a column of the vertex list to change the sign of a column. To show the use of the functions we generate first a simple roman date string as solid volume.

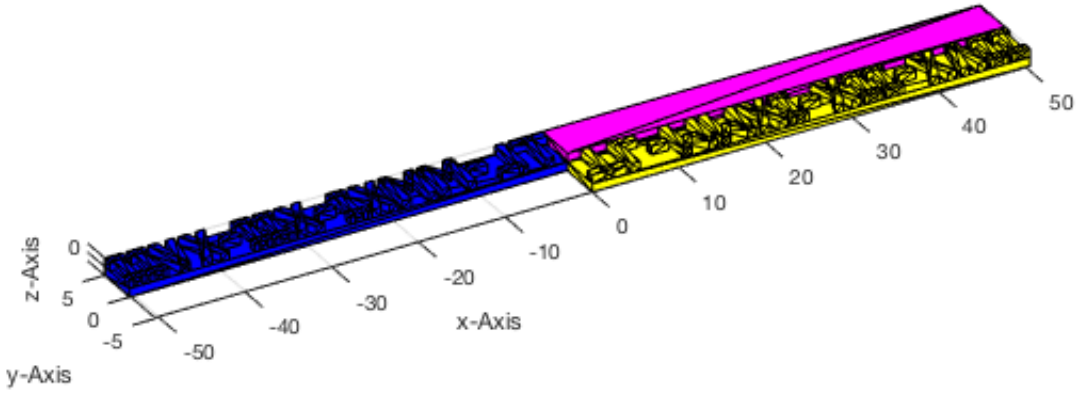
```
close all;
[VL,FL,d]=VLFLtext('TL-MMXI-XII-XVII'); VLFLplot(VL,FL,'r'); view(-30,30);
```



The functions for mirroring solid objects by manipulating the vertex list are the following:

- **VLswapX** mirrors the solid at the x-axis (y/z-plane).
- **VLswapY** mirrors the solid at the y-axis (x/z-plane).
- **VLswapZ** mirrors the solid at the z-axis (x/y-plane).

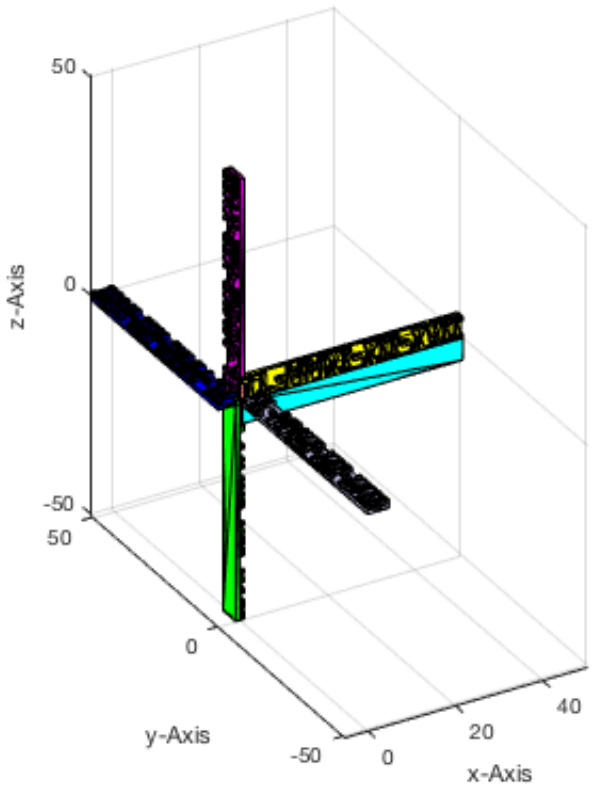
```
close all, view (-30,30); grid on;
VLFLplot(VLswapX(VL),FL,'b');       % mirror at x-axis
VLFLplot(VLswapY(VL),FL,'y');       % mirror at y-axis
VLFLplot(VLswapZ(VL),FL,'m');       % mirror at z-axis
```



The functions for turning solid objects by manipulating the vertex list are the following:

- **VLswapXY** turn the x-axis to the y-axis.
- **VLswapXZ** turn the x-axis to the z-axis.
- **VLswapYX** turn the y-axis to the x-axis.
- **VLswapYZ** turn the y-axis to the z-axis.
- **VLswapZX** turn the z-axis to the x-axis.
- **VLswapZY** turn the z-axis to the y-axis.

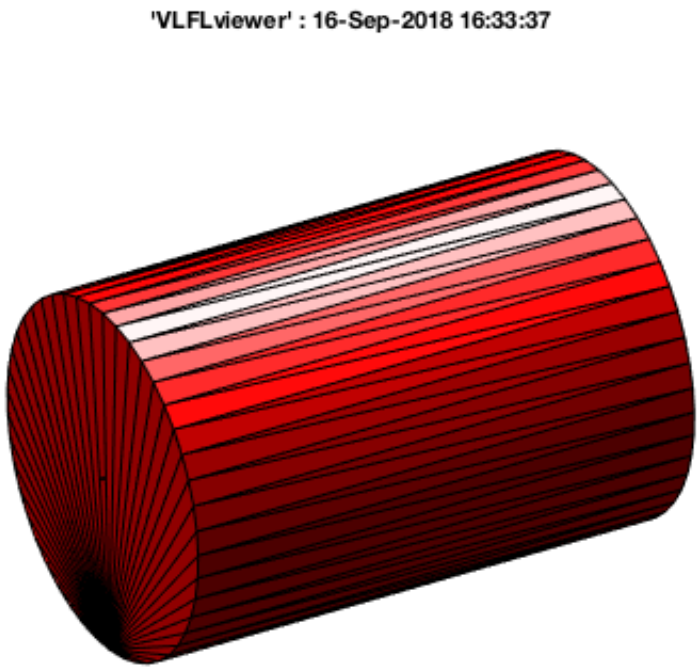
```
close all, view (-30,30); grid on
VLFLplot(VL,FL,'r');                % original solid
VLFLplot(VLswapXY(VL),FL,'b');      % turn the x-axis to the y-axis
VLFLplot(VLswapXZ(VL),FL,'m');      % turn the x-axis to the z-axis
VLFLplot(VLswapYZ(VL),FL,'y');      % turn the y-axis to the z-axis
VLFLplot(VLswapZY(VL),FL,'c');      % turn the z-axis to the y-axis
VLFLplot(VLswapZX(VL),FL,'g');      % turn the z-axis to the x-axis
VLFLplot(VLswapYX(VL),FL,'w');      % turn the y-axis to the x-axis
```



6. Spatial transformation of solids by manipulating the vertex lists (VL)

All solid objects consisting of vertices and facets can be moved and rotated by only manipulating the vertex list (VL). Since the facet list is an index list, the facet list (FL) is not affected by a transformation of the vertex list. The following example generates a cylinder and perform different postion and orientation transformations.

```
closeall;
VLFLviewer([]);
PL=PLcircle(10);                          % define a base-contour
[VL,FL]=VLFLofPLz (PL,30);                % extrude to a solid volume
VL=VLswapZX (VL);                         % swap X and X axis
VLFLplot(VL,FL); view (-30,30); grid on;  % plot as red cylinder
```



In detail, there are five basic transformation functions for manipulation a vertex list (VL)

- **VLtrans0** for translating the solid in the coordiante system origin.
- **VLtrans1** for translating the solid into quadrant 1.
- **VLtransP** for translating the solid using a translation vector.
- **VLtransR** for rotation the solid using a rotation matrix.
- **VLtransT** for transformating using an homogenous transformation matrix.

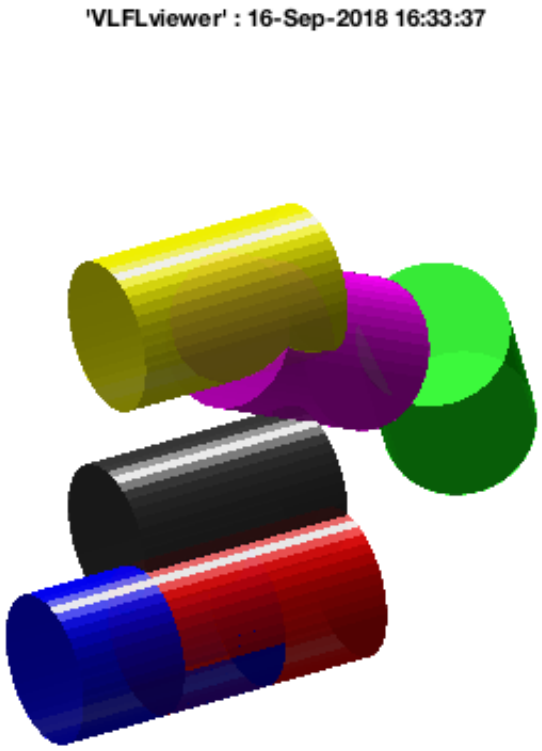
In addition to the already existing matlab functions rotx, roty, and rotz, two new functions are useful.

- **rot** for generating a 3x3 rotation matrix for x y z given in rad.
- **rotdeg** for generating a 3x3 rotation matrix for x y z given in degree.

```
VL=VLtrans0 (VL);                    % Transformation into the origin (blue)
VLFLplot(VL,FL,'b'); view (-30,30);

VL=VLtrans1 (VL);                     % Transformation into quadrant 1 (black)
VLFLplot(VL,FL,'k'); view (-30,30);

VL=VLtransP (VL,[0 ;0; 30]);          % Transformation upwards 30 mm (yellow)
VLFLplot(VL,FL,'y'); view (-30,30);

VL=VLtransR (VL,rotdeg(0,30,15));     % Rotate 30 degree around y and 15 around z (magenta)
VLFLplot(VL,FL,'m'); view (-30,30);

T=[rotdeg(0,30,15), [20;0;0];[0 0 0 1]] % define a homogenous transformation matrix (green)
```

```
VL=VLtransT (VL,T);                % Transformation using an HT matrix
VLFLplot(VL,FL,'g'); view (-30,30); grid on;
VLFLplotlight (1,0.9); grid on;
```

```
T =

    0.8365   -0.2241    0.5000   20.0000
    0.2588    0.9659         0         0
   -0.4830    0.1294    0.8660         0
         0         0         0    1.0000
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:33:38!
Executed 16-Sep-2018 16:33:40 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
```

```
matlab
simulink
==========================================================================================
==========
```

- Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-19
- Tim Lueth, executed and published on 64 Bit PC using Windows with Matlab 2014b on 2014-11-19

Published with MATLAB® R2018a

Tutorial 03: Closed 2D Contours and Boolean Operations in 2D

2014-11-19: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.3 required)
- 2. The contour polybool list (mapping toolbox)
- 3. Surface tesselation for contour polybool list (CPL)
- 4. Orientation of outer and inner polygons of a CPL
- 5. Boolean operations of contour polybool lists (CPL)
- 6. Converting a closed polybool list into a point list (PL) and an edge list (EL)
- 7. Extruding point list (PL) and edge list (EL) to a solid volume
- 8. Converting a point list and edge list into a closed polybool list
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

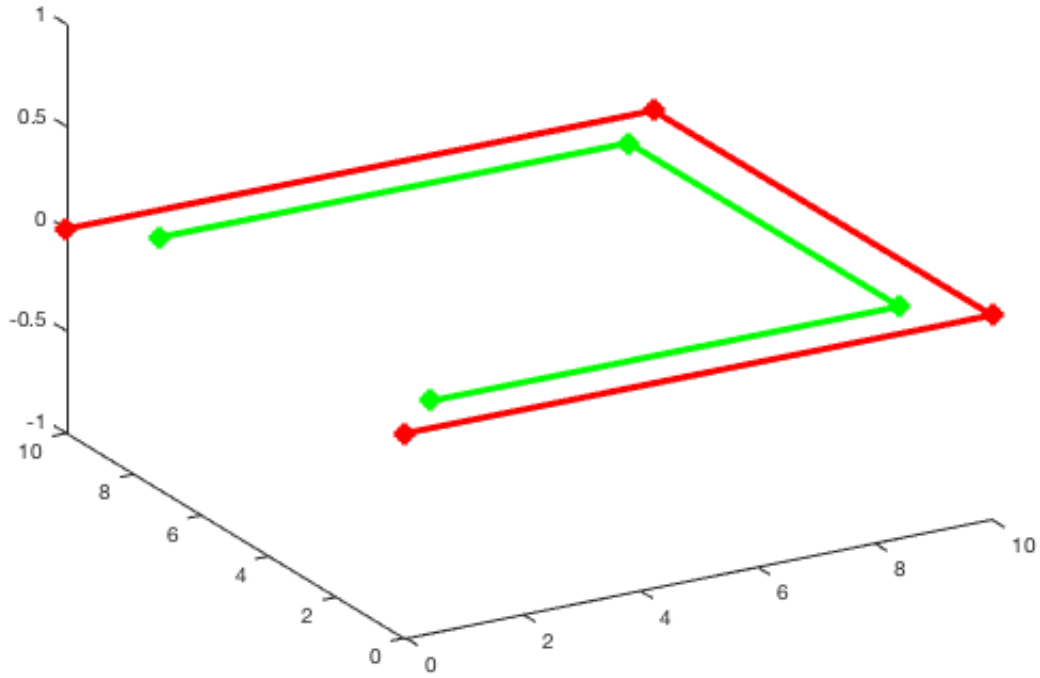
Motivation for this tutorial: (Originally SolidGeometry 1.3 required)

2. The contour polybool list (mapping toolbox)

This 3rd example deals with a different data structure for the description of 2D closed contour polygons: Contour Polybool List (CPL). The CPL is a nx2 x/y-coordinate point list [x y] similar to a point list (PL). Always, the first and last point of the list are considered as closed. In addition, it is possible to concatenate two point list after another. To separate the individual contours, a separator-point [NaN NaN] is inserted between them.

```
close all;
PLA=[0 0; 10 0; 10 10; 0 10], PLB=[1 1; 9 1; 9 9; 1 9]
PLplot (PLA,'r-*',3);PLplot (PLB,'g-*',3); view(-30,30);
```

```
PLA =

     0     0
    10     0
    10    10
     0    10


PLB =

     1     1
     9     1
     9     9
     1     9
```

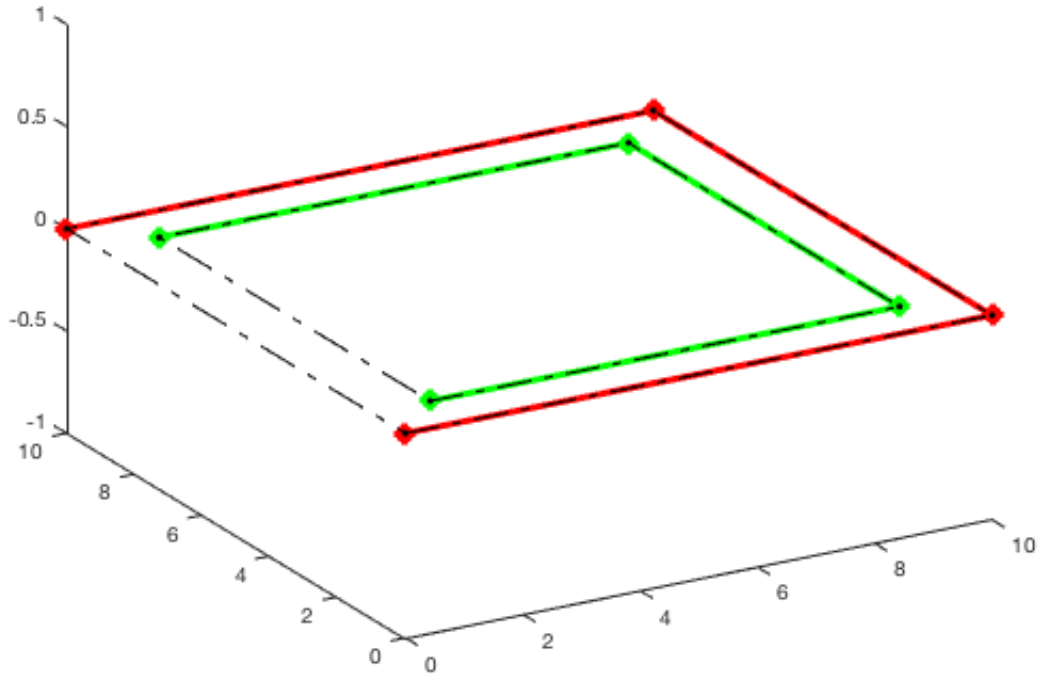


The concatenation generates then the closed polybool list (CPL). Two functions are helpful to draw a CPL and also to show the start point (black cube), the endpoint (black ring) and the direction of each edge of the CPL:

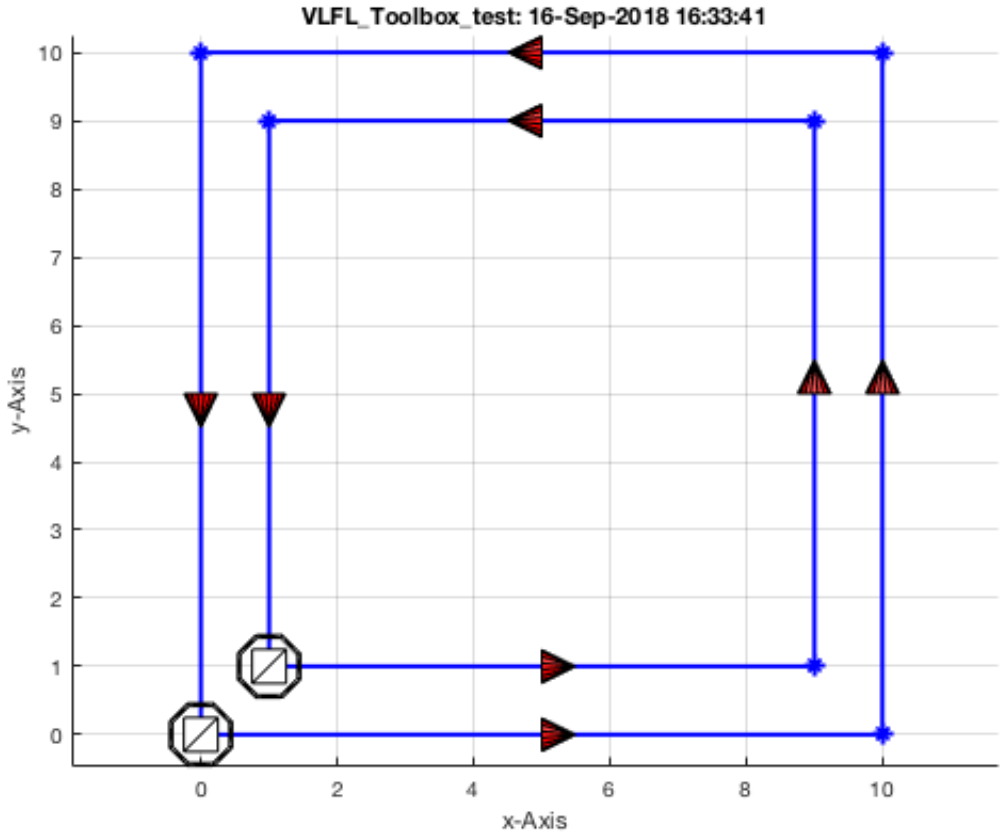
- **CPLplot** draws a closed contour polybool list (CPL).
- **PLELofCPL** draws a start point, end point and direction-arrows, when called without any output variable.

```
CPL=[PLA;NaN NaN;PLB],
CPLplot (CPL,'k.-.',1);
```

```
CPL =

     0     0
    10     0
    10    10
     0    10
   NaN   NaN
     1     1
     9     1
     9     9
     1     9
```




```
close all;
PLELofCPL (CPL);
```



3. Surface tesselation for contour polybool list (CPL)

A closed polygon list can be considered as bounding contour for a surface. In general, there exist different strategies, to tesselate a bounding contour, to get a triangle surface description. There is no optimal one. We can distinguish **simple strategies** or more advanced strategies such as **Delaunay-Triangulation** or **Row-Scanning-Triangulation**. In example 1 we used a simple strategy for closing convex polygons.

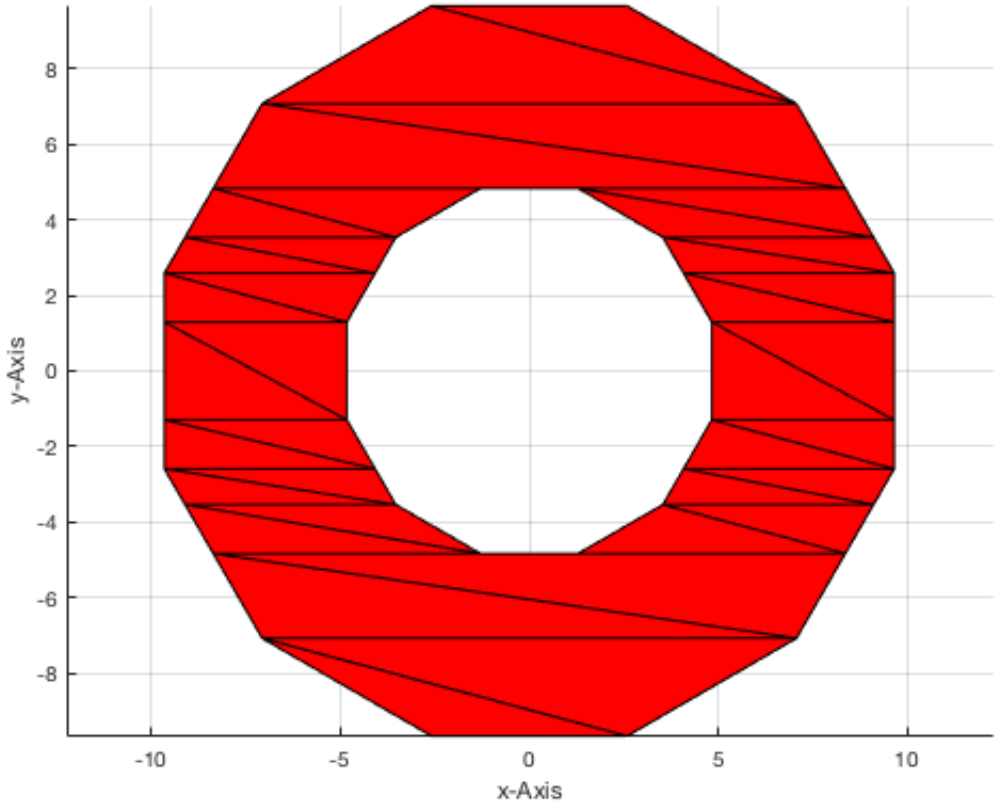
- **Row-Scanning-Triangulation** is able to handle all kinds of polygons (even enclosed), but the triangle facets are sometimes very small. Furthermore, the point contains redundant information.
- **Delaunay-Triangulation** is able to handle all kinds of polygons (even enclosed), but has problems with polygons that cross each other or share one point or more points, i.e. overlapping edges.

It is not unique to tesselate polygons, if they are enclosed or cross each other. There exist no general purpose solution. Nevertheless, in most cases, the Delaunay-Triangulation is preferrable, since this concept does work also in 3D. To generate a facet list (FL) for a CPL, there exist two functions. In case of crossing polygons or overlapping polygons, additional points have to be caluclated automatically, and therefore, the points in the point list can change. In this case, you will get a warning, but only in case of the Delaunay-triangulation. Conventionally, additional split/crossing points are added at the end of the point list, in case of the Delaunay-triangulation.

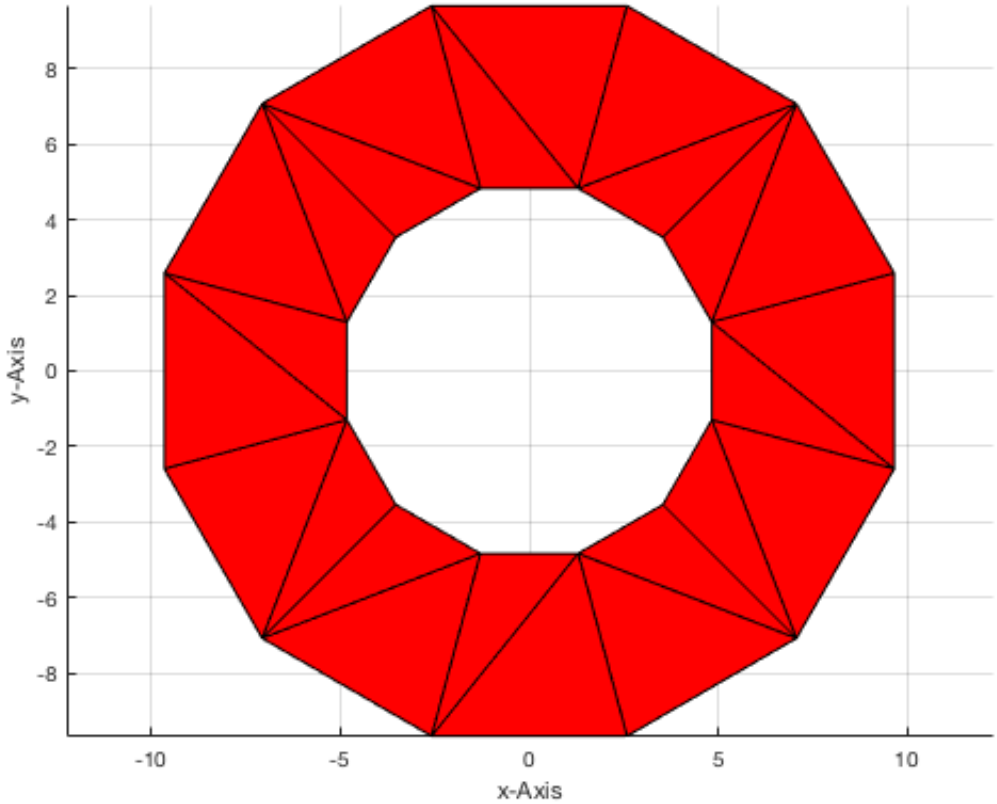
- **PLFLofCPLpoly** returns a facet tesselation by a simple y-coordinate row scanning (the points are ordered by increasing y, contour by contour, do not mix, but are redundant. Not as efficient as Delaunay, and not useful for 3D).
- **PLFLofCPLdelaunay** returns a facet tesselation by a Delaunay-triangulation (no crossings or joint points or joint edges are allowed, i.e create additional split points).

```
close all;
CPL=[PLcircle(10,12);NaN NaN;PLcircle(5,12)]
[PL,FL]=PLFLofCPLpoly(CPL); VLFLplot(PL,FL);
```

```
CPL =

    9.6593   -2.5882
    9.6593    2.5882
    7.0711    7.0711
    2.5882    9.6593
   -2.5882    9.6593
   -7.0711    7.0711
   -9.6593    2.5882
   -9.6593   -2.5882
   -7.0711   -7.0711
   -2.5882   -9.6593
    2.5882   -9.6593
    7.0711   -7.0711
       NaN       NaN
    4.8296   -1.2941
    4.8296    1.2941
    3.5355    3.5355
    1.2941    4.8296
   -1.2941    4.8296
   -3.5355    3.5355
   -4.8296    1.2941
   -4.8296   -1.2941
   -3.5355   -3.5355
   -1.2941   -4.8296
    1.2941   -4.8296
    3.5355   -3.5355
```



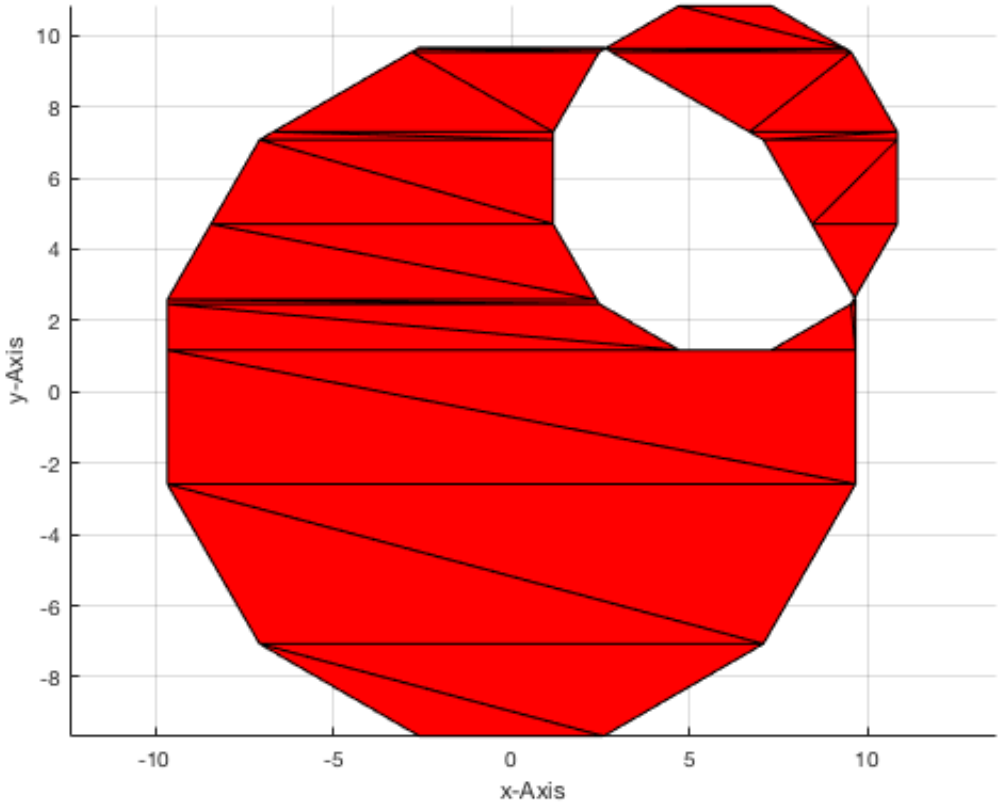

```
close all;
[PL,FL]=PLFLofCPLdelaunay(CPL); VLFLplot(PL,FL);
```

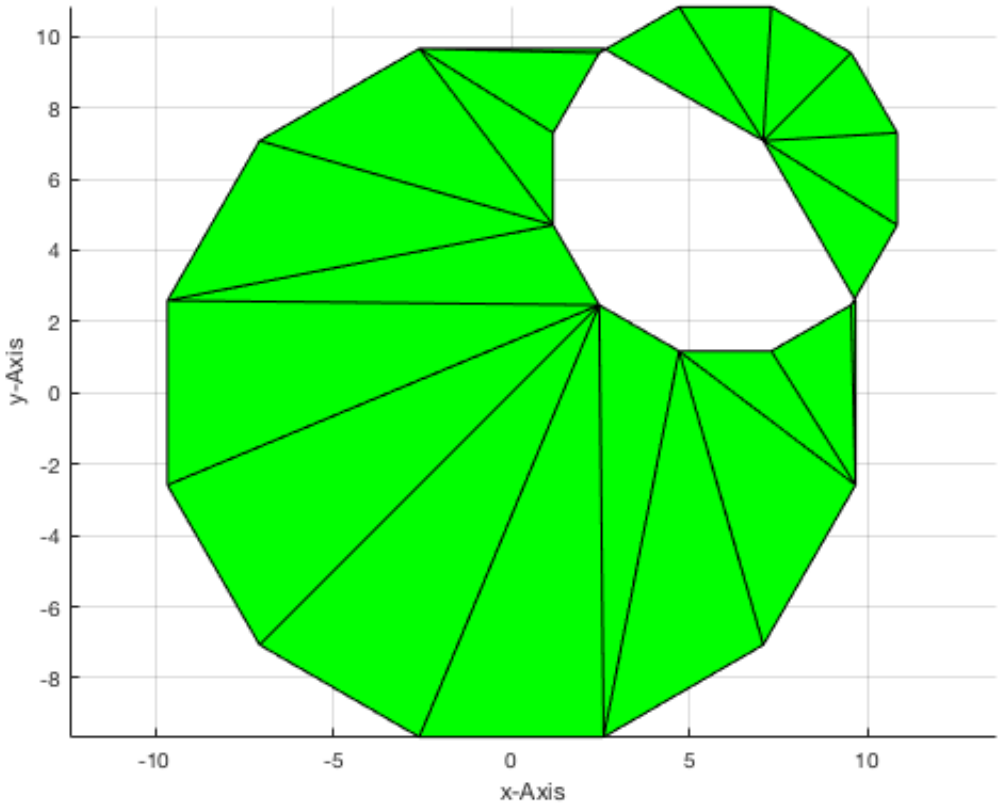


The next example shows the need for splitting contours:

```
close all
CPL=[PLcircle(10,12);NaN NaN;PLcircle(5,12)+6];
figure(1); [PL,FL]=PLFLofCPLpoly(CPL); VLFLplot(PL,FL,'r');
figure(2); [PL,FL]=PLFLofCPLdelaunay(CPL); VLFLplot(PL,FL,'g');
```

```
Warning: Intersecting edge constraints have been split, this may have added new
points into the triangulation.
```





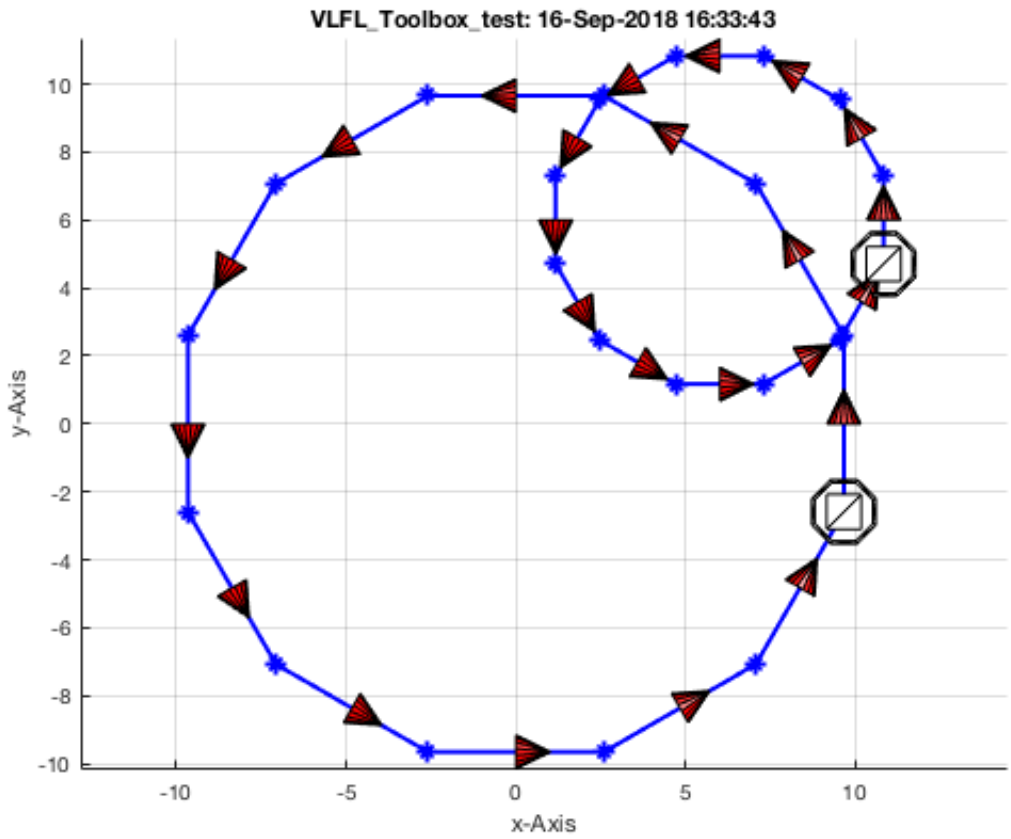
4. Orientation of outer and inner polygons of a CPL

In the mapping tool box, that supports the boolean operation of contours, there is a rule to use outer contours in clockwise (cw) directions and embedded contours always in the opposite direction, which means counter-clockwise (ccw) for the first level of embedding. Unfortunatly, this is exactly the other way around to the rules that are used for Delaunay representation and 3D surface description. So we have to be careful later when switching from CPL to surface description for 3D modelling. In 3D modelling, to distinguish outer contours and inner contours of a CPL, we use counter clockwise (ccw) polygons for outside and clockwise (cw) polygons for inside contours. At a later stage we want to generate walls extruded upwards on the contours. If the contour direction is defined correctly for 3D modelling, the facet orientation can be calcuated automatically from the contour direction.

- **PLELofCPL** shows the direction of the used contours.
- **flip(PL)** changes the direction of a point list.

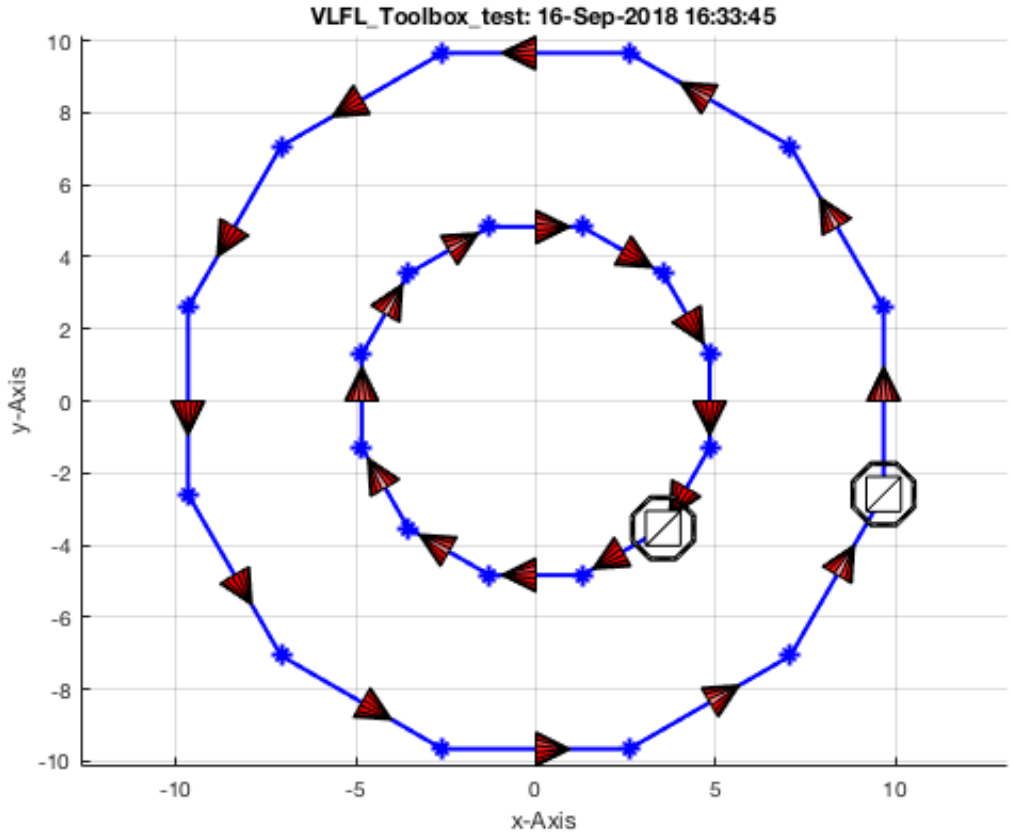
In the next figure, we see both polygons counter-clockwise:

```
PLELofCPL(CPL);
```



Now, we see the outer polygons counter-clockwise and the inner polygons clockwise

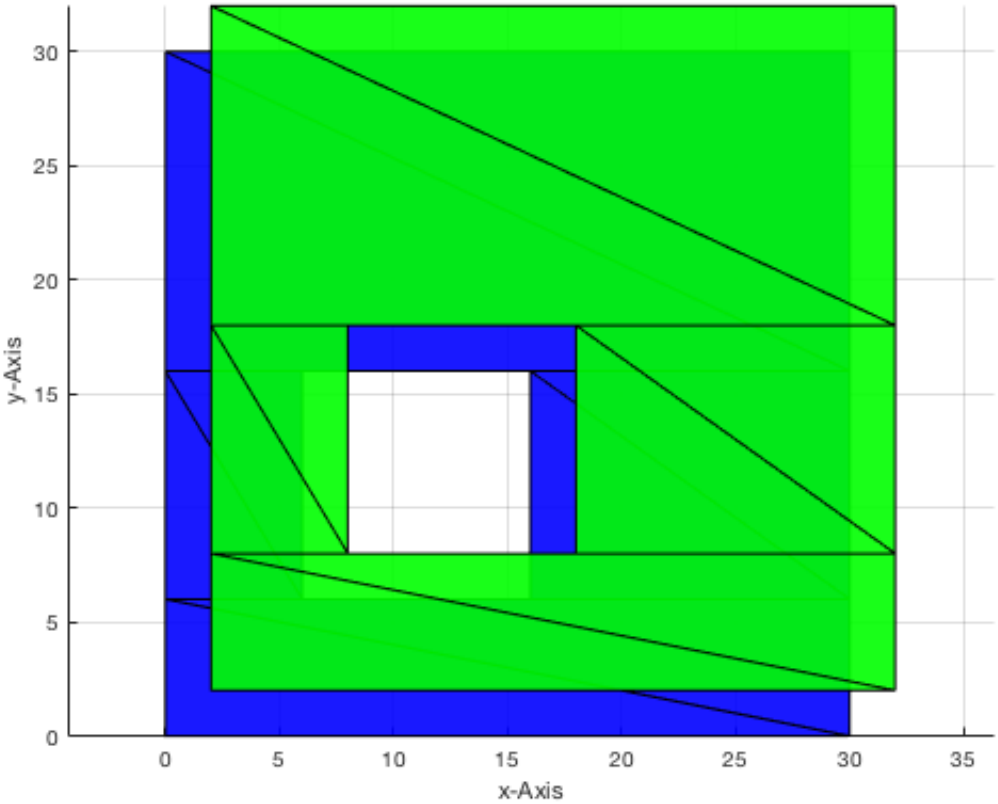
```
close all;
CPL=[PLcircle(10,12);NaN NaN;flip(PLcircle(5,12))];
PLELofCPL(CPL);
```



5. Boolean operations of contour polybool lists (CPL)

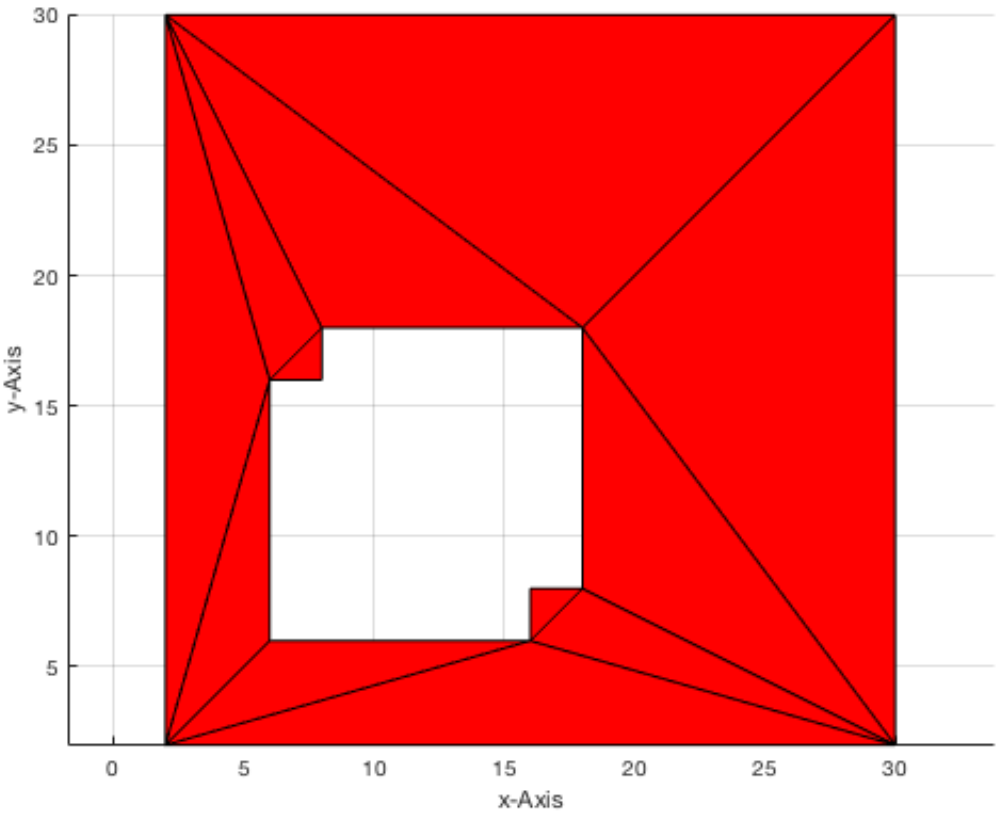
The main advantage of the CPL representation is currently the possibility to use the polybool functions of the mapping toolbox. Here, we show the use in embedded functions that help later to make the step forward to 3D modeling. We start with boolean operations of contour polybool lists (CPL).

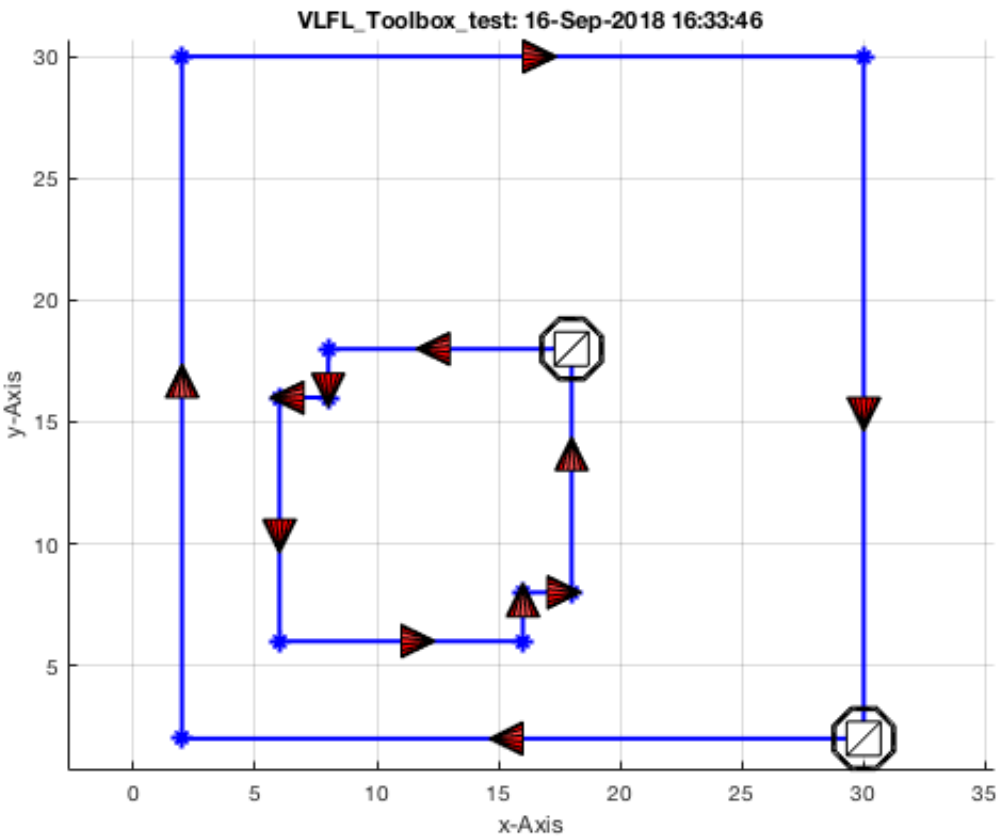
```
close all; figure;
CPL=[PLA*3;NaN NaN;(PLA)+6];
CPLA=CPL;    [PL,FL]=PLFLofCPLpoly(CPLA); VLFLplot(PL,FL,'b');
CPLB=CPL+2;  [PL,FL]=PLFLofCPLpoly(CPLB); VLFLplot(PL,FL,'g');
VLFLplotlight (0,0.9)
```



- **CPLpolybool('and',CPLA,CPLB)** delivers CPLA intersecting CPLB.

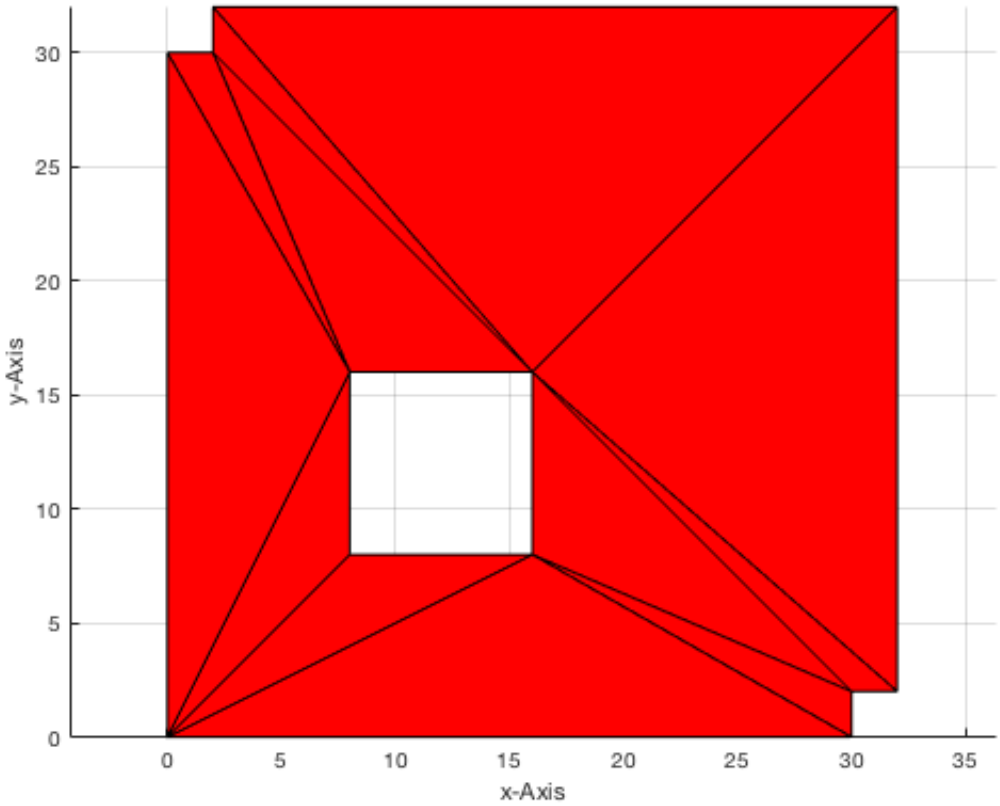
```
close all;
CPLN=CPLpolybool('and',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

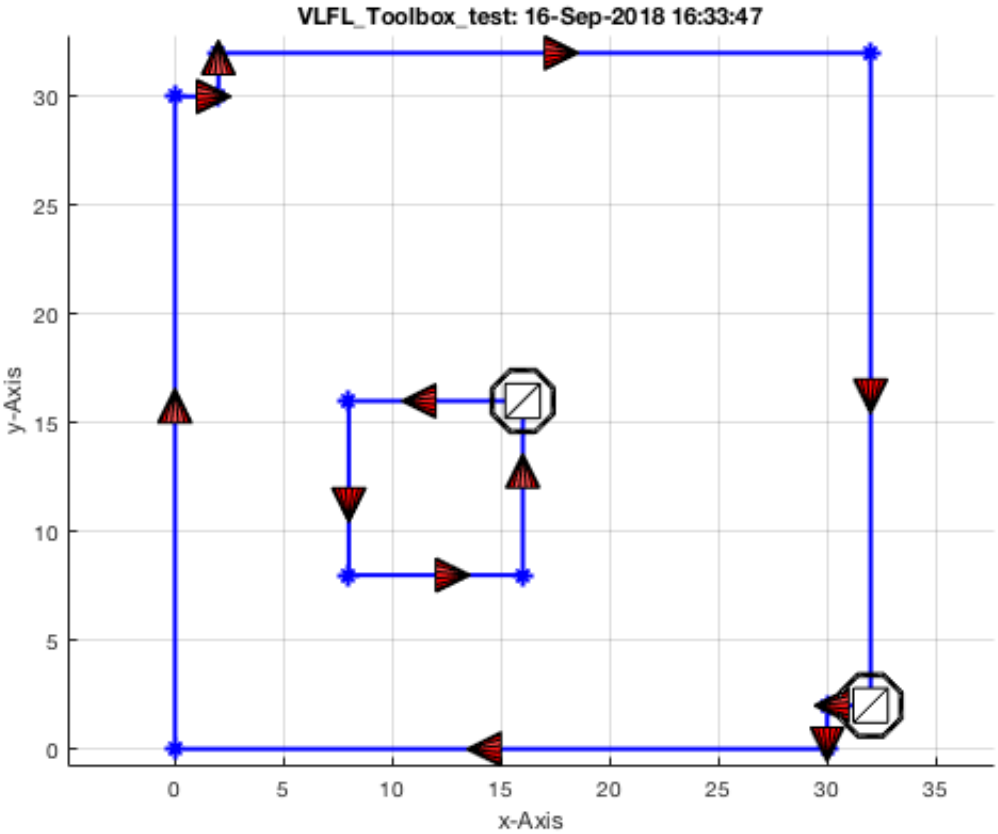




- **CPLpolybool('or',CPLA,CPLB)** delivers CPLA united with CPLB.

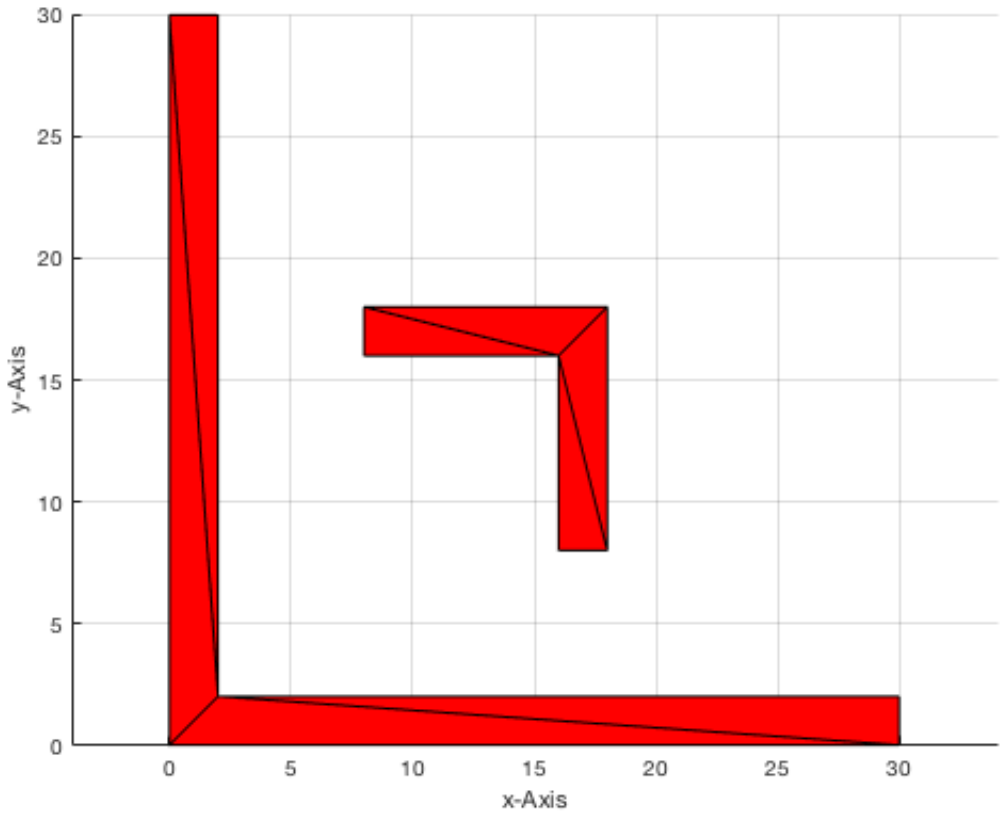
```
close all;
CPLN=CPLpolybool('or',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

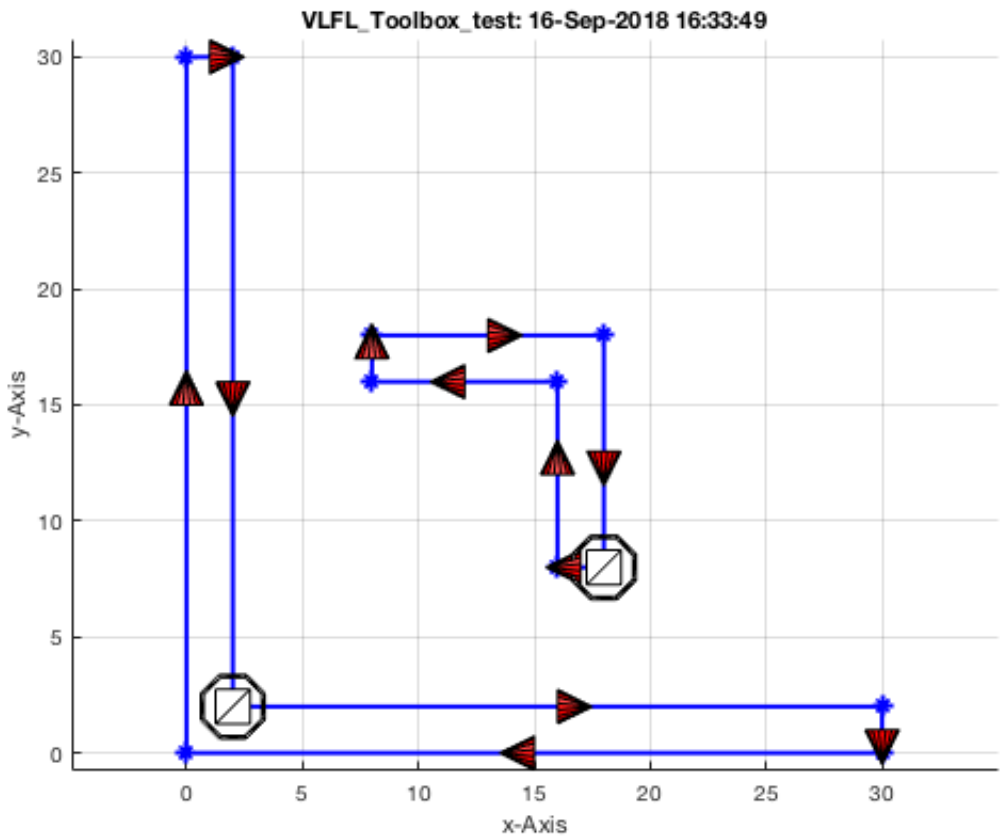




- **CPLpolybool('minus',CPLA,CPLB)** delivers CPLA minus CPLB.

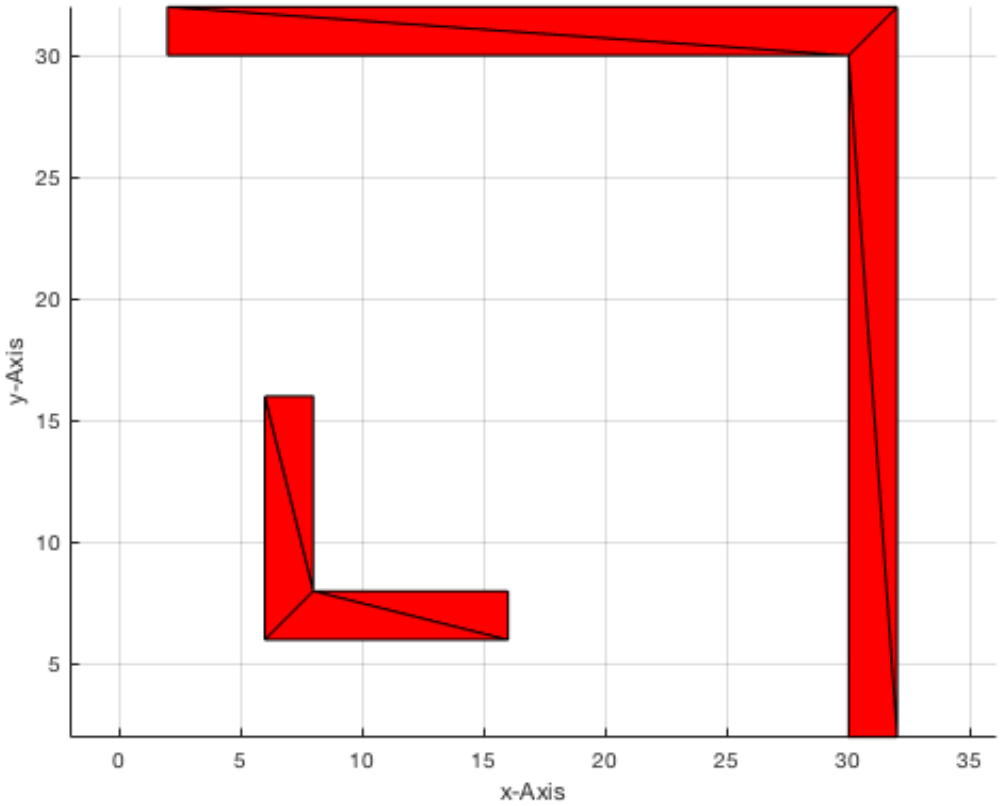
```
close all;
CPLN=CPLpolybool('minus',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

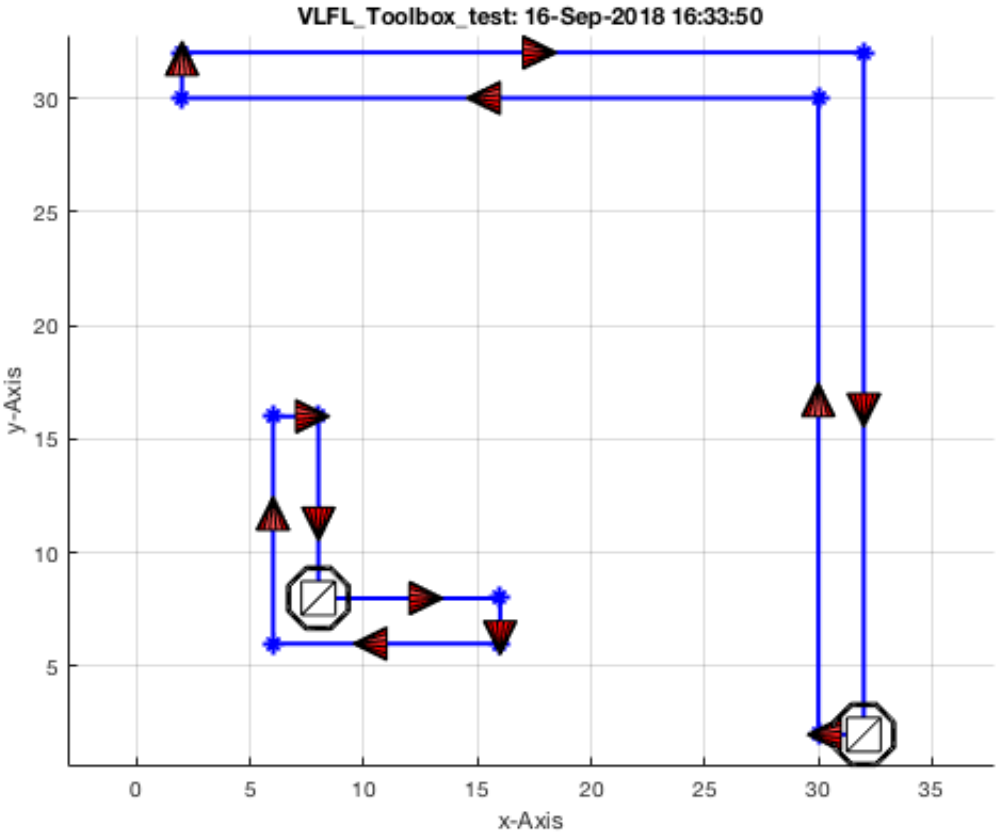




- **CPLpolybool('minus',CPLB,CPLA)** delivers CPLB minus CPLA.

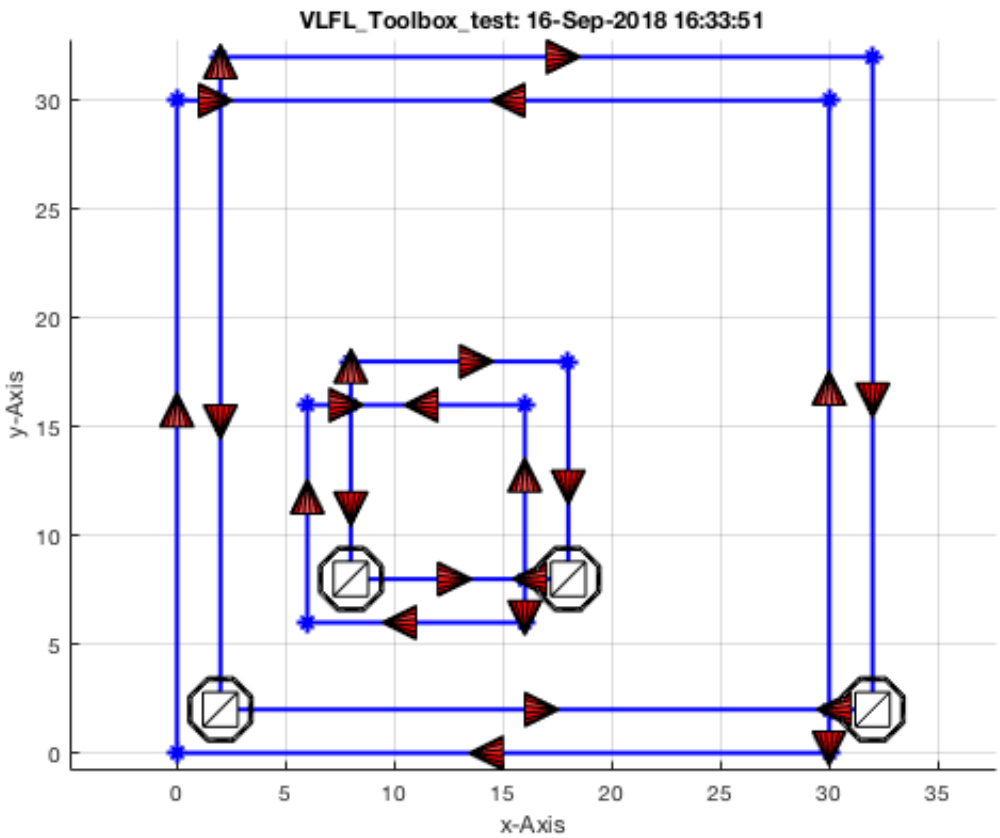
```
close all;
CPLN=CPLpolybool('minus',CPLB,CPLA);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```





- **CPLpolybool('xor',CPLA,CPLB)** delivers CPLA exclusiveor CPLB.

```
close all;
CPLN=CPLpolybool('xor',CPLA,CPLB);
[PL,FL]=PLFLofCPLpoly   (CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```



6. Converting a closed polybool list into a point list (PL) and an edge list (EL)

To generate an extruded 2½D solid from a CPL, it makes sense first to convert a CPL to a point list (PL) with an explicit description of the edges of the polygons as edge list (EL). Since the EL, as a result of CPLpolybool, has an inverted direction, ELflip is used to change the direction of the edges.

- **PLELofCPL** transforms the CPL into a point list (PL) and an edge list (EL).
- **ELflip** corrects the edge direction after CPLpolybool.

```
CPLN=CPLpolybool('minus',CPLA,CPLB)
[PL,EL]=PLELofCPL(CPLN), EL=ELflip(EL),
```

```
CPLN =

    18     8
    16     8
    16    16
     8    16
     8    18
    18    18
    18     8
   NaN   NaN
     2     2
    30     2
    30     0
     0     0
     0    30
     2    30
     2     2


PL =

    18     8
    16     8
    16    16
     8    16
     8    18
    18    18
     2     2
    30     2
    30     0
     0     0
     0    30
     2    30


EL =

     1     2
     2     3
     3     4
     4     5
     5     6
     6     1
     7     8
```

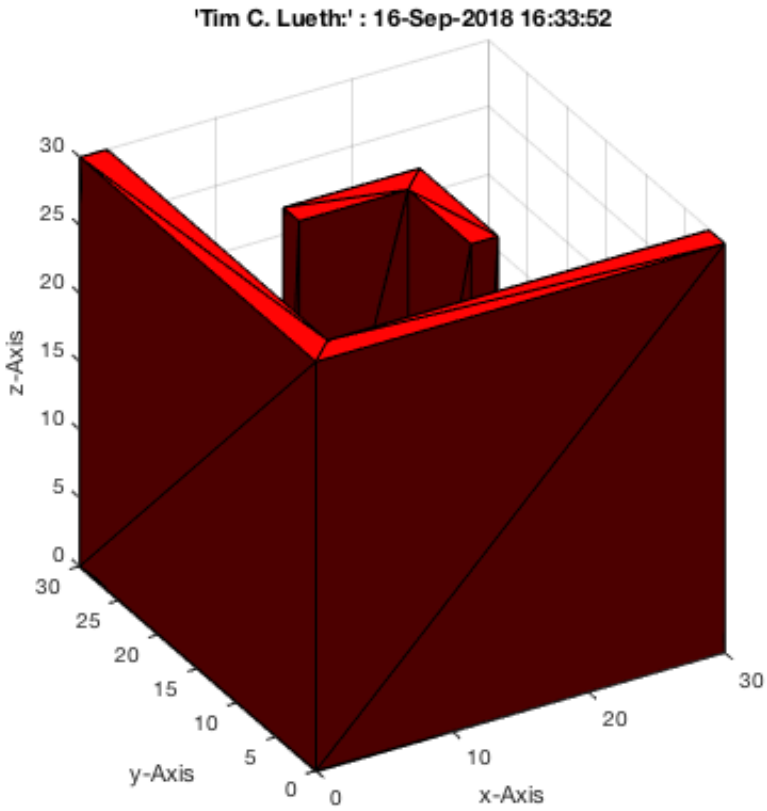
```
     8     9
     9    10
    10    11
    11    12
    12     7


EL =

     7    12
    12    11
    11    10
    10     9
     9     8
     8     7
     1     6
     6     5
     5     4
     4     3
     3     2
     2     1
```

7. Extruding point list (PL) and edge list (EL) to a solid volume

After a boolean operation there is often the wish to extrude the resulting base contour into a 3D solid volume. The function is explained later in more detail. Anyway,it is helpful to see in 3D a model that is the result of a 2D boolean operation.

- **VLFLofPLELz** extruding a point list (PL) and edge list (EL) into 3D.

```
close all; VLFLfigure; view(-30,30);
[VL,FL]=VLFLofPLELz(PL,EL,30); VLFLplots(VL,FL);
```



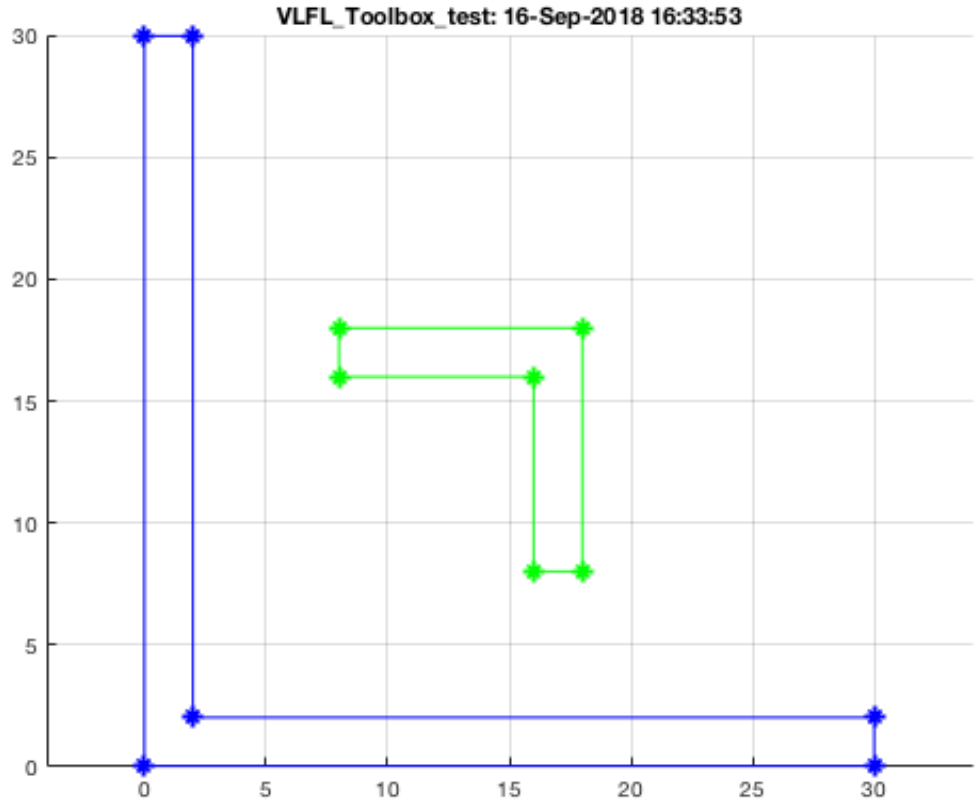
8. Converting a point list and edge list into a closed polybool list

Finally, as it was possible to convert a CPL into and point list and an edge list, there is a function for the opposite.

- **CPLofPLEL** converting a point list (PL) and an edge list (EL) into a closed polybool list.

```
CPLofPLEL(PL,EL)
```

```
ans =

     2     2
     2    30
     0    30
     0     0
    30     0
    30     2
     2     2
   NaN   NaN
    18     8
    18    18
     8    18
     8    16
    16    16
    16     8
    18     8
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:33:53!
Executed 16-Sep-2018 16:33:55 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-19*
- *Mattias Traeger, executed and published on 64 Bit PC using Windows with Matlab 2014b on 2014-11-20*

Published with MATLAB® R2018a

Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)

2014-11-21: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.6 required)
- 2. Moving and rotating point lists (PL) and closed polygon lists (CPL)
- 3. Simple extrusion of point lists (PL/CPL) to design 2½D solids
- 4. Simple Design of 2½D solids by boolean operators for point lists (PL/CPL)
- 5. Unite both contours and extrusion: CPL=CPLpolybool('or',PLA,PLB)
- 6. Intersect both contours: CPL=CPLpolybool('and',PLA,PLB)
- 7. Substract contour B from A: CPL=CPLpolybool('-',PLA,PLB)
- 8. Substract contour A from B: CPL=CPLpolybool('-',PLB,PLA)
- 9. Exclusive or of contour A and B: CPLpolybool('xor',PLB,PLA)
- 10. Checking the solid volumes for 3D printing
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)

- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 1.6 required)

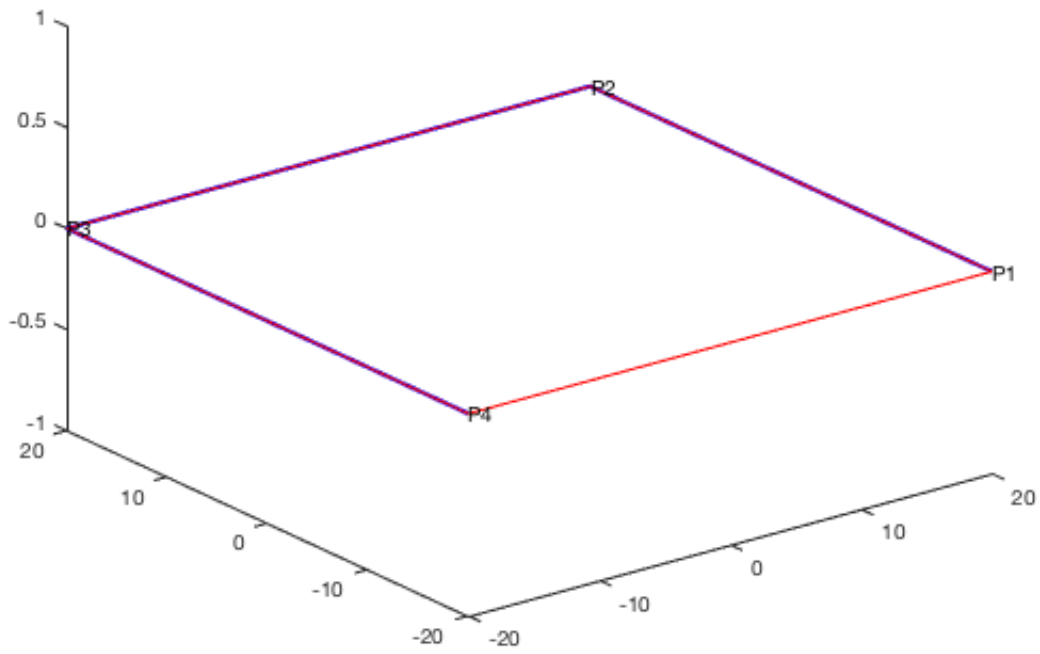
2. Moving and rotating point lists (PL) and closed polygon lists (CPL)

By using point lists (PL) and closed polybool lists (CPL) it is very convinient to design 2.5D objects. Here we see a first example to design a simple square by three function:

At the beginning we just plot simple point lists or closed polygon lists

- **PLplot** plots the point list as open contour
- **CPLplot** plots the point list as closed contour
- **textVL** plots descriptors at the points

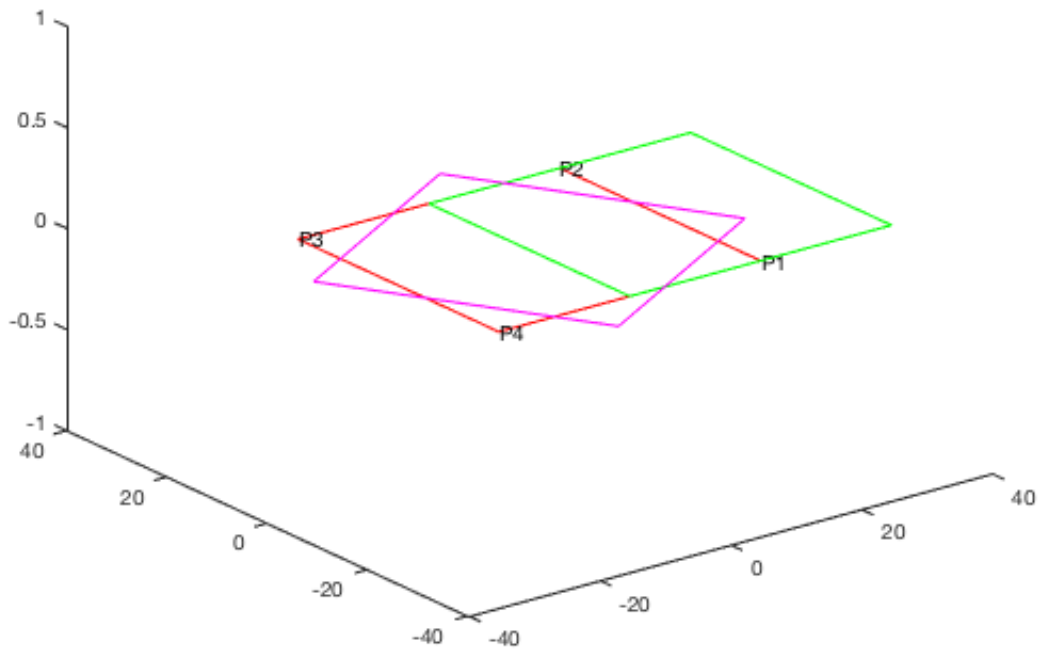
```
close all;
PLA=PLcircle(20*sqrt(2),4);            % Generate a circle with 4 point, i.e. square
PLplot(PLA,'b',2);                     % Plots the points in blue
CPLplot(PLA,'r');                      % Plots the closed polygon in red
textVL (PLA);                          % Plots point descriptors
```



Next we move the square and rotate the square

- **PLtransP** moves a point list (PL) or closed polygon list(CPL)
- **PLtransR** rotates a point list (PL) or closed polygon list(CPL)

```
close all;
CPLplot(PLA,'r');                        % Plots the closed polygon in red
textVL (PLA);                            % Plots point descriptors
CPLplot(PLtransP(PLA,[20  0]),'g');      % Plot the moved polygon in green
CPLplot(PLtransR(PLA,rot(pi/6)),'m');    % Plot the rotated polygon in magenta
```

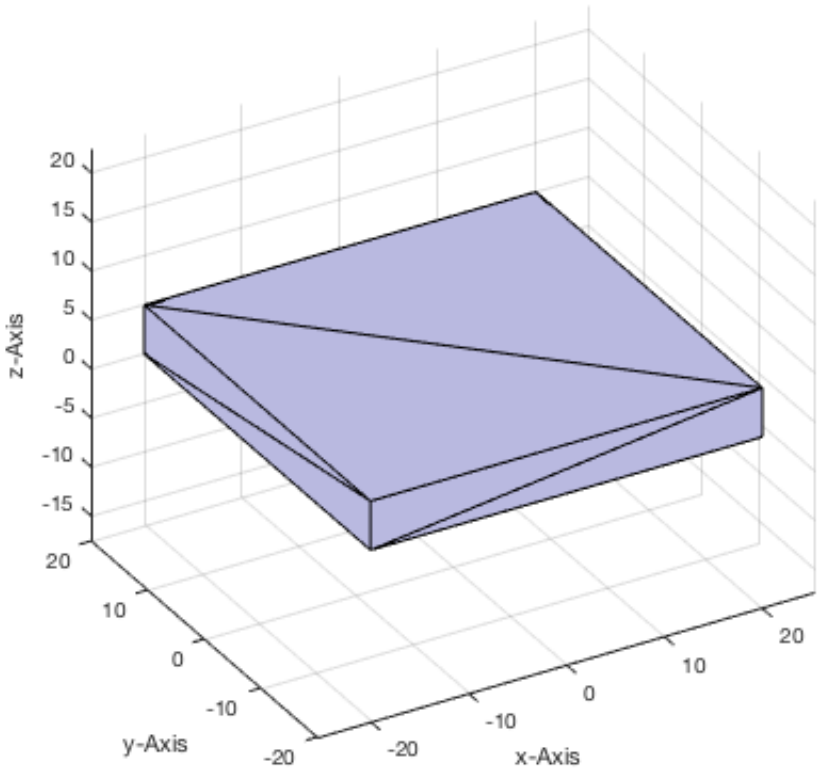


3. Simple extrusion of point lists (PL/CPL) to design 2½D solids

Next we extrude the square in 3D

- **VLFLofCPLz** extrudes point list (PL) or closed polygon list(CPL) in z

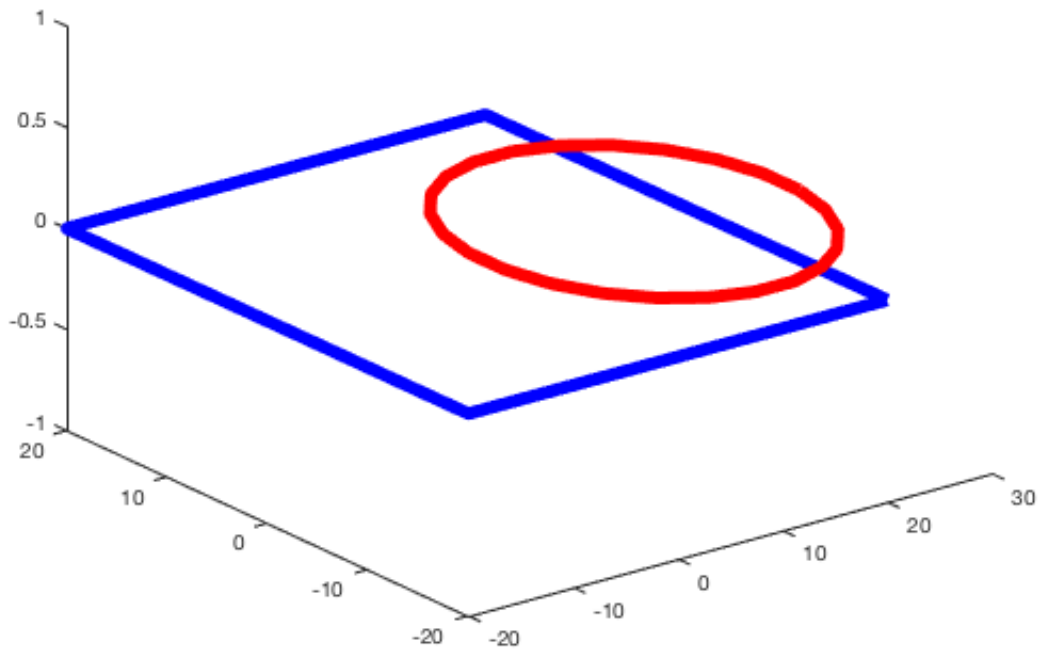
```
close all
[VL,FL]=VLFLofCPLz (PLA,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30);
```



4. Simple Design of 2½D solids by boolean operators for point lists (PL/CPL)

In this example we start with two point list, a square and an octaedron

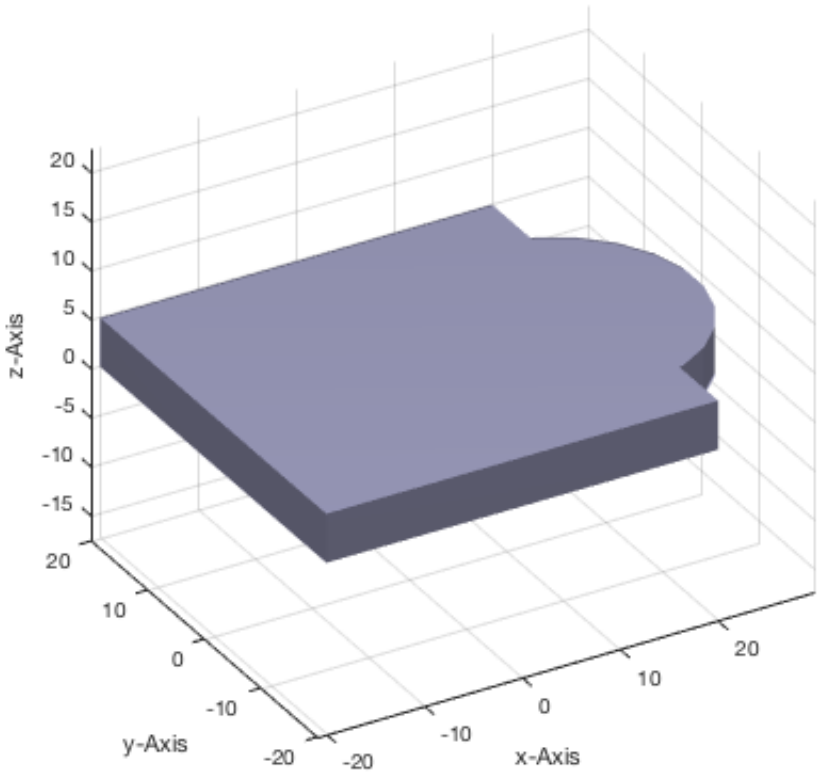
```
close all;
PLA=PLcircle(20*sqrt(2),4);
PLB=PLcircle(10*sqrt(2),24);  PLB=PLtransP(PLB,[15 0]);
CPLplot(PLA,'b',6); CPLplot(PLB,'r',6);
```



5. Unite both contours and extrusion: CPL=CPLpolybool('or',PLA,PLB)

```
close all
CPL=CPLpolybool('or',PLA,PLB); [VL,FL]=VLFLofCPLz(CPL,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30); VLFLplotlight (1);
VLFLwriteSTL (VL,FL,'A','EXP04-unite')
```

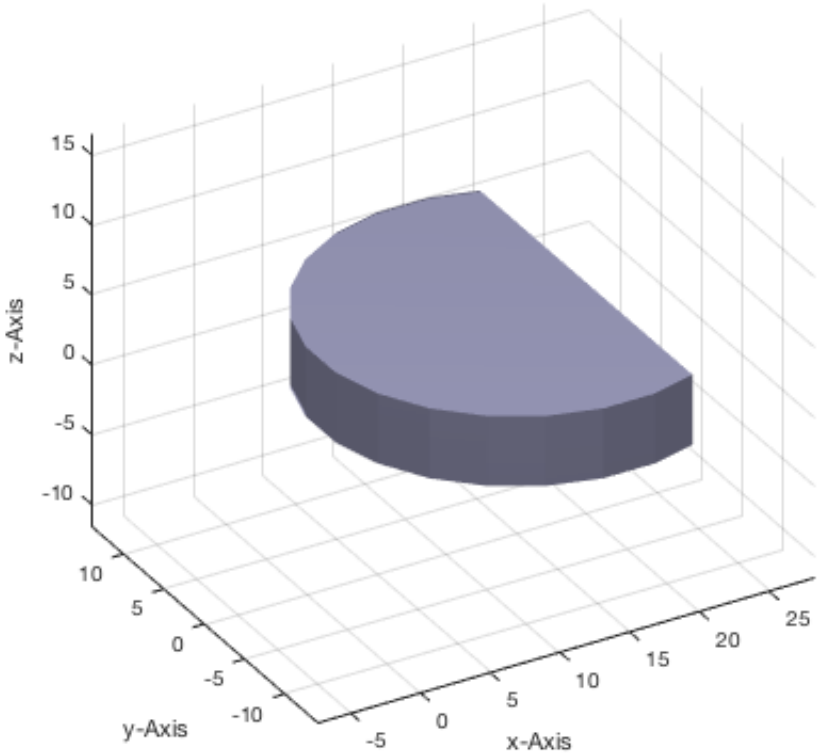
```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/A.STL in ASCII MODE
completed.
```



6. Intersect both contours: CPL=CPLpolybool('and',PLA,PLB)

```
close all
CPL=CPLpolybool('and',PLA,PLB); [VL,FL]=VLFLofCPLz(CPL,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30); VLFLplotlight (1);
VLFLwriteSTL (VL,FL,'A','EXP04-intersect')
```

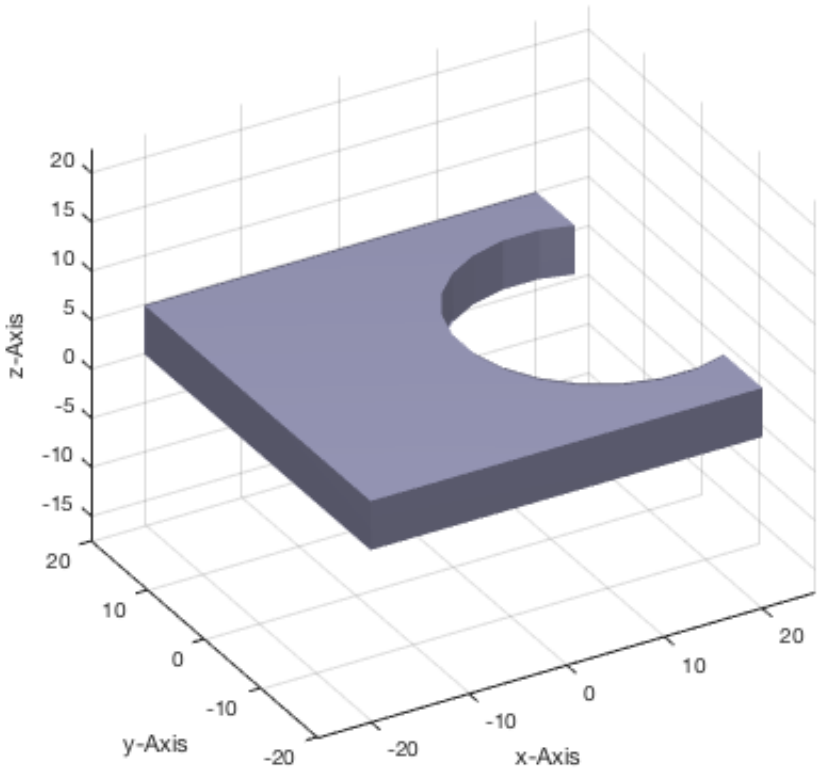
```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/A.STL in ASCII MODE
completed.
```



7. Substract contour B from A: CPL=CPLpolybool('-',PLA,PLB)

```
close all
CPL=CPLpolybool('-',PLA,PLB); [VL,FL]=VLFLofCPLz(CPL,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30); VLFLplotlight (1);
VLFLwriteSTL (VL,FL,'A','EXP04-AminusB')
```

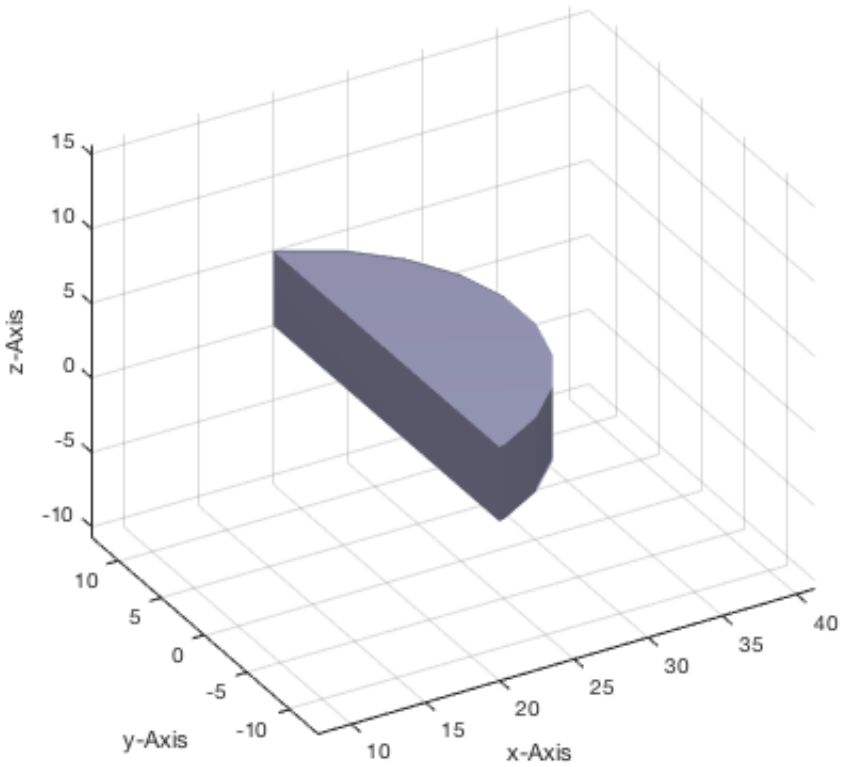
```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/A.STL in ASCII MODE
completed.
```



8. Substract contour A from B: CPL=CPLpolybool('-',PLB,PLA)

```
close all
CPL=CPLpolybool('-',PLB,PLA); [VL,FL]=VLFLofCPLz(CPL,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30); VLFLplotlight (1);
VLFLwriteSTL (VL,FL,'EXP04-AminusB')
```

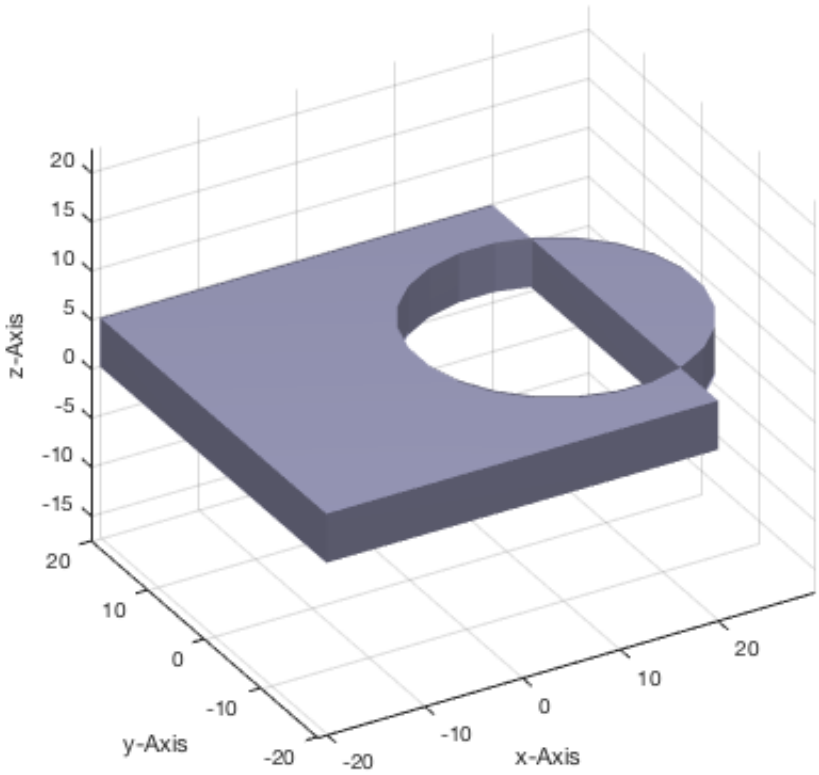
```
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/EXP04-AminusB.STL in ASCII MODE
completed.
```



9. Exclusive or of contour A and B: CPLpolybool('xor',PLB,PLA)

```
close all
CPL=CPLpolybool('xor',PLB,PLA); [VL,FL]=VLFLofCPLz(CPL,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30); VLFLplotlight (1);
VLFLwriteSTL (VL,FL,'EXP04-AxorB')
% VLFLviewer(VL,FL);
```

```
Warning: Duplicate data points have been detected and removed.
 The Triangulation indices and constraints are defined with respect to the
 unique set of points in DelaunayTri.
WRITING STL FILE /Users/timlueth/Desktop/Toolbox_test/EXP04-AxorB.STL in ASCII MODE
completed.
```



10. Checking the solid volumes for 3D printing

During the last extrusion we got a warning from a Delaunay-triangulation during the extrusion function VLFLofCPLz. This is typically a warning that somehow the final part cannot be printed with a 3D printing process such as FDM,SLS,3DP etc. Here in this case, the result of xor were two parts that touch each other at two edges. Such a part cannot be printed. The reason behind is called non-manifold edge problem. There are also problems with non manifold points and non-manifold facets.

```
VLFLchecker (VL,FL);
```

```
VLFLchecker: 60 vertices and 120 facets.
    0 FACET PROBLEMS DETECTED (ERRORS)
    0 VERTEX PROBLEMS DETECTED (OBSOLETE WARNING)
    4 EDGE PROBLEMS DETECTED (NON MANIFOLD WARNING)
    0 SOLID/EDGE PROBLEMS DETECTED (OPEN SOLID WARNING)
```

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:33:59!
```

```
Executed 16-Sep-2018 16:34:01 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-21*
- *Mattias Traeger, executed and published on 64 Bit PC using Windows with Matlab 2014b on 2014-11-21*

Published with MATLAB® R2018a

Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)

2014-11-22: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.7 required)
- 2. Creating, plotting, writing of the struct *Solid Geometry* (SG)
- 3. Spatial transformations of solid geometries and *sets of solid geometries*
- 4. Merging of solid geometries (SG) and sets of solid geometries
- 5. Non-manifold points, edges, and facets of solid geometries (SG)
- 6. Additive Design: Separate or penetrate solid geometries (SG)
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 1.7 required)

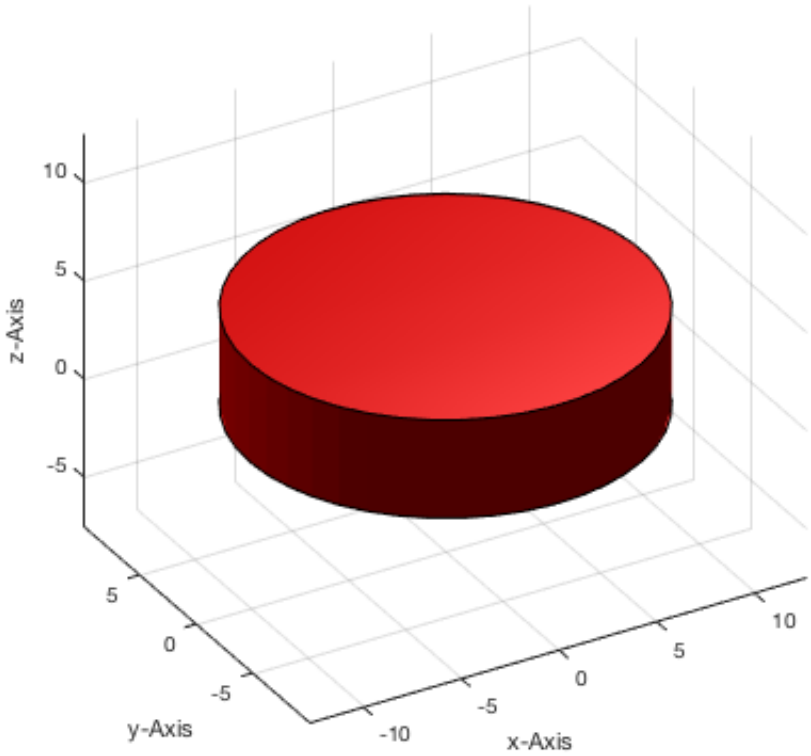
2. Creating, plotting, writing of the struct *Solid Geometry* (SG)

Even if it is useful to know that a vertex list (VL), and a facet list (FL) is required for 3D modeling, it is more convenient to use matlab structs for solid geometries (SG). Instead of writing VL or FL, we use SG.VL, SG.FL as variables. The advantage is, that each solid object is described by one struct that contains vertex list and facet list, but can contain other defined information (such as the underlying CPL, PL or EL) or it is open for your own defined information.

- **SGofCPLz** for extruding a solid object from a CPL similar to VLFLofCPLz.
- **SGplot** for plotting one or more solid objects similar VLFLplots.
- **SGchecker** for checking a solid object similar to VLFLchecker.
- **SGwriteSTL** for writing a solid object similar to VLFLwriteSTLb.
- **SGsize** for generating the bounding box of a solid geometry (SG).

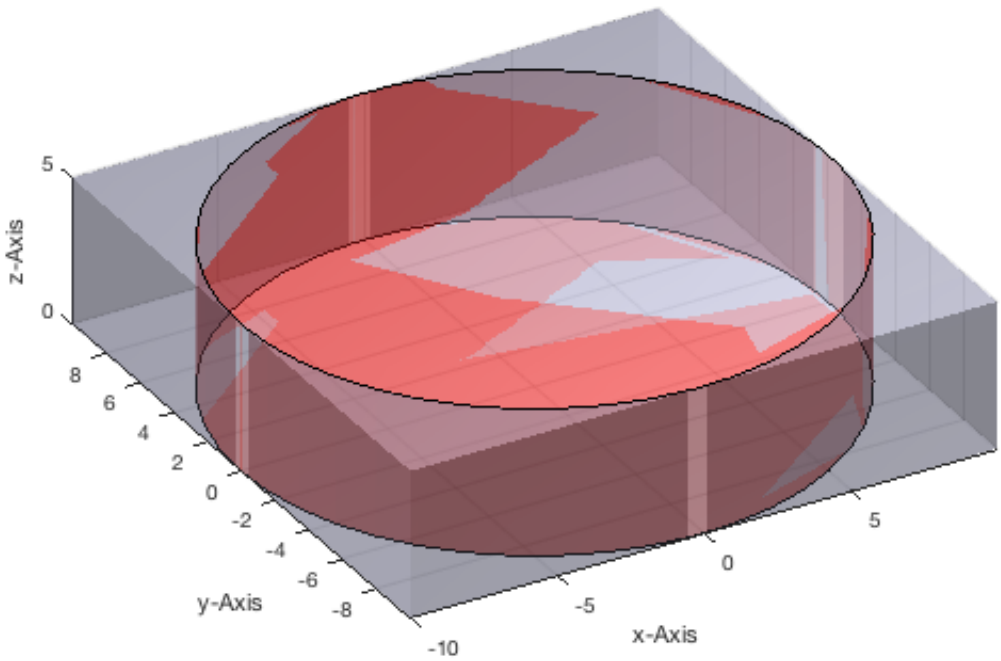
```
close all;
nSG=SGofCPLz(PLcircle(10),5)
SGplot(nSG,'r',1); VLFLplotlight(1); view (-30,30);
SGchecker(nSG);
SGwriteSTL (nSG,'EXP05-1');
```

```
nSG =

  struct with fields:

    CPL: [45×2 double]
     VL: [90×3 double]
     FL: [176×3 double]
     PL: [45×2 double]
     EL: [45×2 double]
```



Often it is useful to know the size of the bounding box of an object and to plot it.

```
bs=SGsize(nSG); [BB.VL,BB.FL]=VLFLofBB(bs); SGplot (BB,'w'); VLFLplotlight(1,0.4);
```

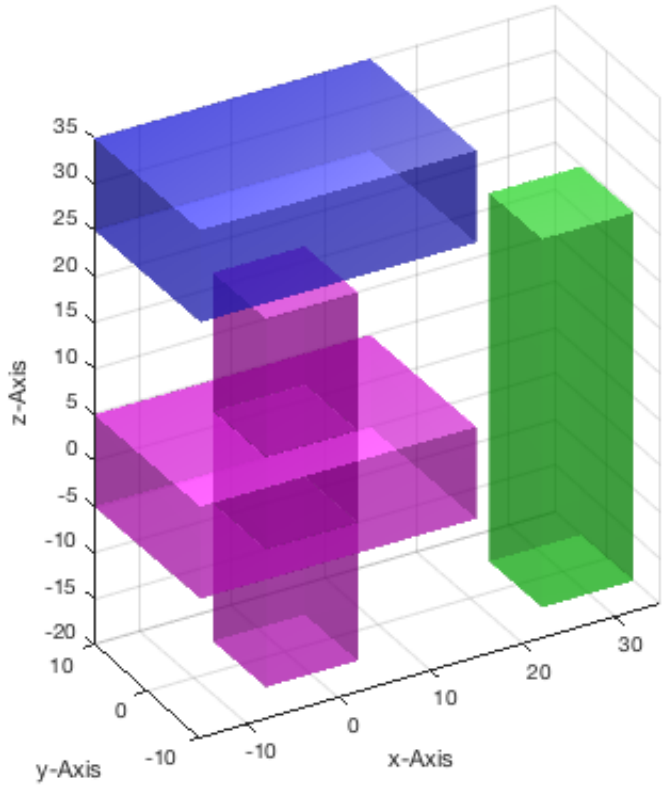


3. Spatial transformations of solid geometries and *sets of solid geometries*

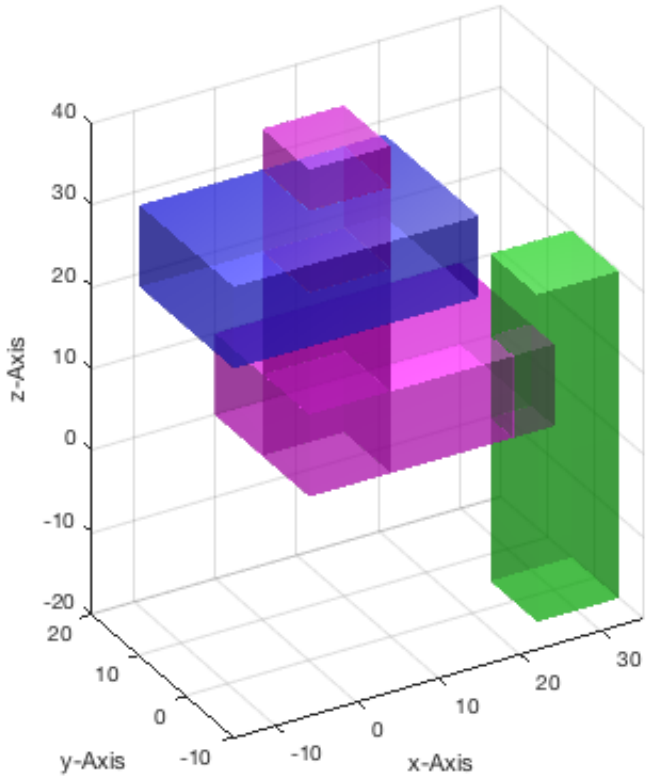
In contrast to manipulate an individual solid geometry as struct, it is often useful to manipulate or handle a set of solid geometries. For this purpose, we use the cell concept of Matlab. A=SGbox([30,20,10]); {A, A, A} is a set of three solid geometries that can be given as arguments of a function and can also be the output argument of a function.

- **SGbox** creates a simple box at the origin indimensions [x y z].
- **SGtransP** moves a solid geometry (SG) or a set of SG by a translation vector.
- **SGtransR** rotate a solid geometry (SG) or a set of SG by a rotation matrix .
- **SGtransT** transform a solid geometry (SG) or a set by a homogenous transformation matrix.
- **SGtrans0** moves a solid geometry (SG) or a set of SG into the coordinate systems origin.
- **SGtrans1** moves a solid geometry (SG) or a set of SG into quadrant 1.

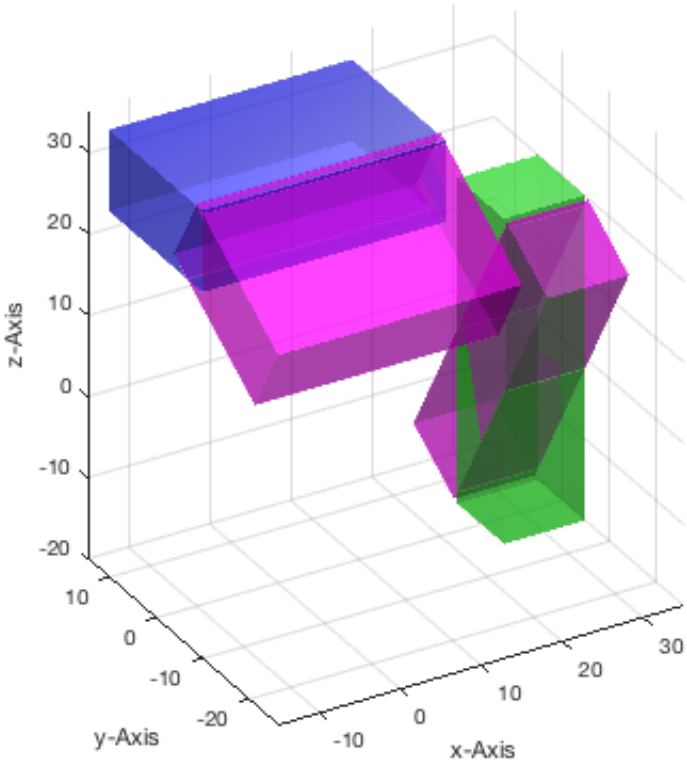
```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtrans0({A,B}),'m'); VLFLplotlight (1,0.5)
```



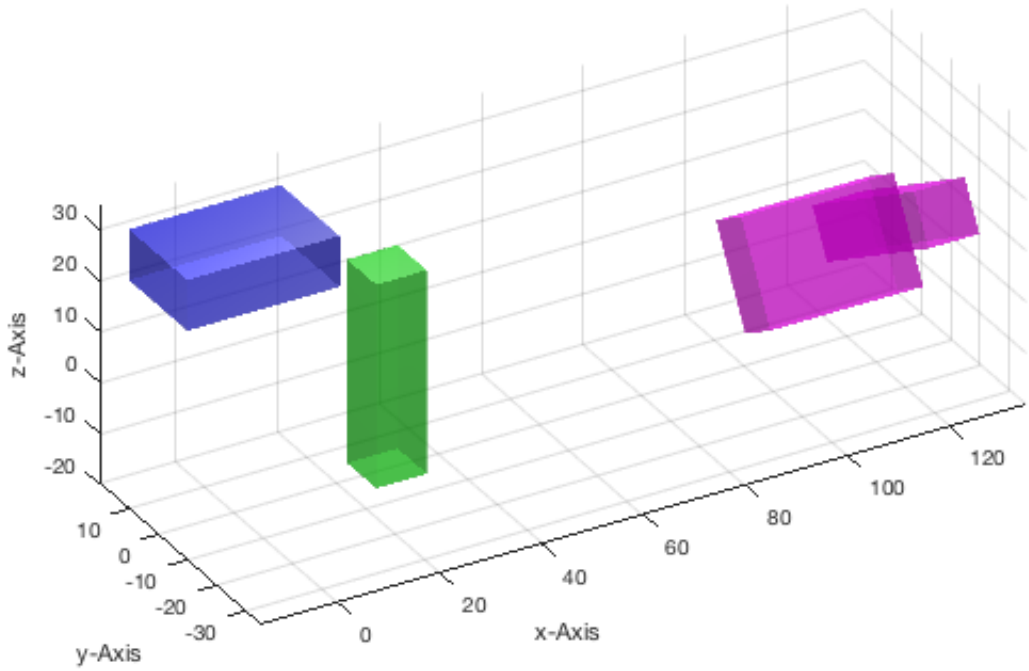

```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtrans1({A,B}),'m'); VLFLplotlight (1,0.5)
```




```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtransR({A,B},rot(pi/6,0,0)),'m'); VLFLplotlight (1,0.5)
```




```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtransT({A,B},[rot(pi/3,0,0),[100;0;0]]),'m'); VLFLplotlight (1,0.5)
```

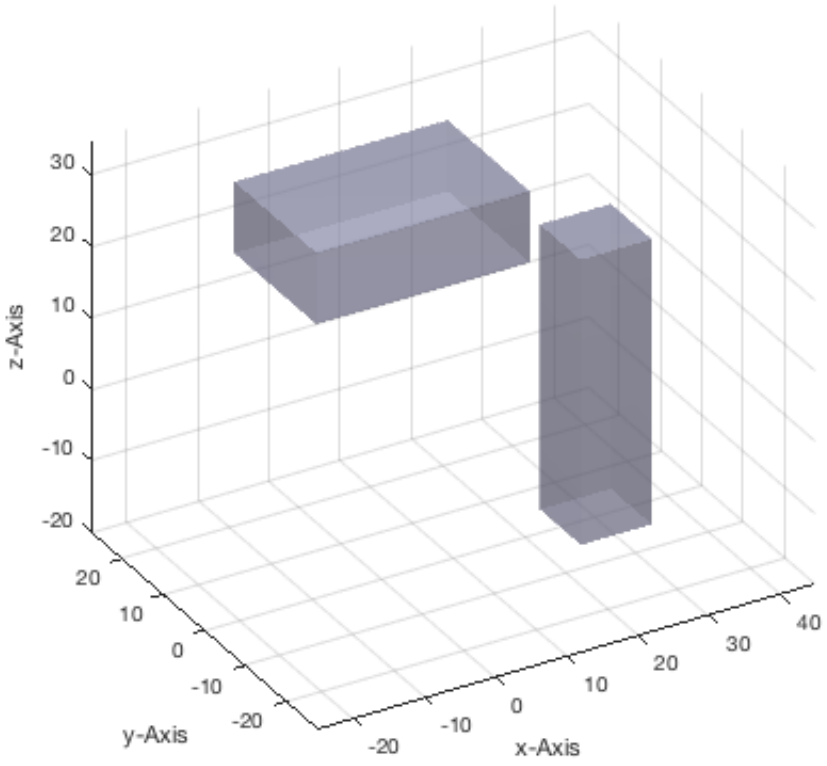


4. Merging of solid geometries (SG) and sets of solid geometries

In the previous example we often created and manipulated two solids. As explained, it is possible to handle several objects at the same time by using sets of elements. Nevertheless, in mosts cases, after some operations we want to merge several solid geometries into one single object. SGcat concatenates the vertex list and the facet list of a set og given solids into one list. Furthermore like in VLFLcat, doubled vertices are detected and removed. It is not possible anymore to separate the objects in general afterwards.

- **VLFLcat** merges two VL/FL into one VL/FL.
- **VLFLcat2** simply concatenates two VL/FL into one VL/FL.
- **SGcat** merges single solids or a set of solids into one solid object.

```
close all;
nSG=SGcat ({A,B}); SGplot (nSG,'w'); view (-30,30); VLFLplotlight (1,0.5)
SGwriteSTL (nSG,'EXP05-2');
```

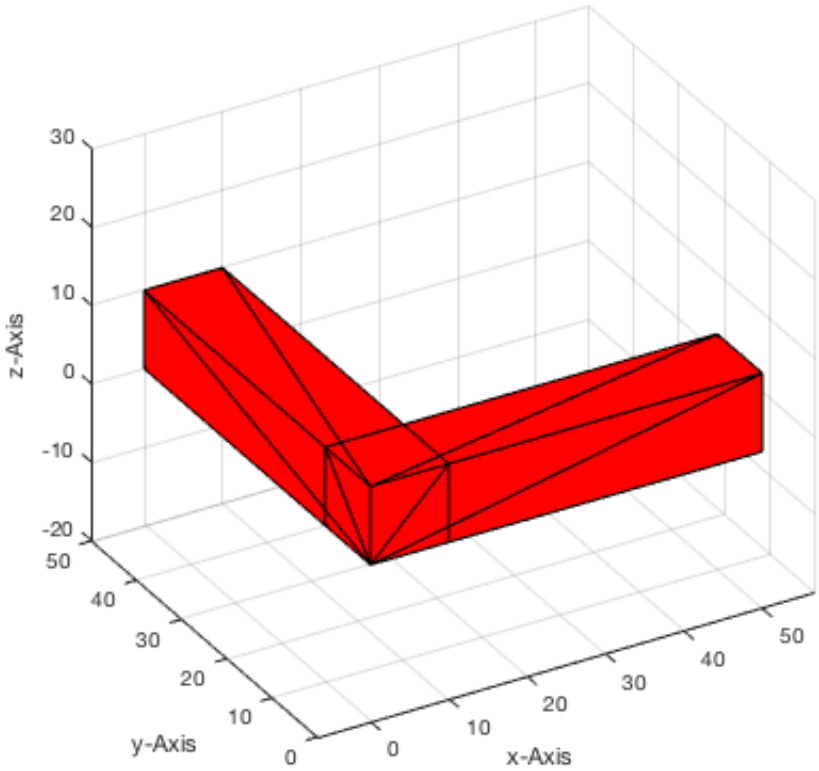


5. Non-manifold points, edges, and facets of solid geometries (SG)

By using functions such as SGcat or VLFLcat/VLcat2, it is very easy and efficient to create solid models and STL-files by simply attaching or penetrating individual solid objects. It is some kind of *additive design of solid objects* in 3D. For a 3D printing process, those additive designed objects are not a real problem, i.e. several independent parts are simply attached or penetrate each other. 3D contour printing of penetrating objects is automatically handled by the slicer, a piece of software that we get to know later.

Nevertheless, as soon as a vertex is used by two independent solids, an edge is used by two independent solids. In this case a slicer software will not be able to solve the manifold problem.

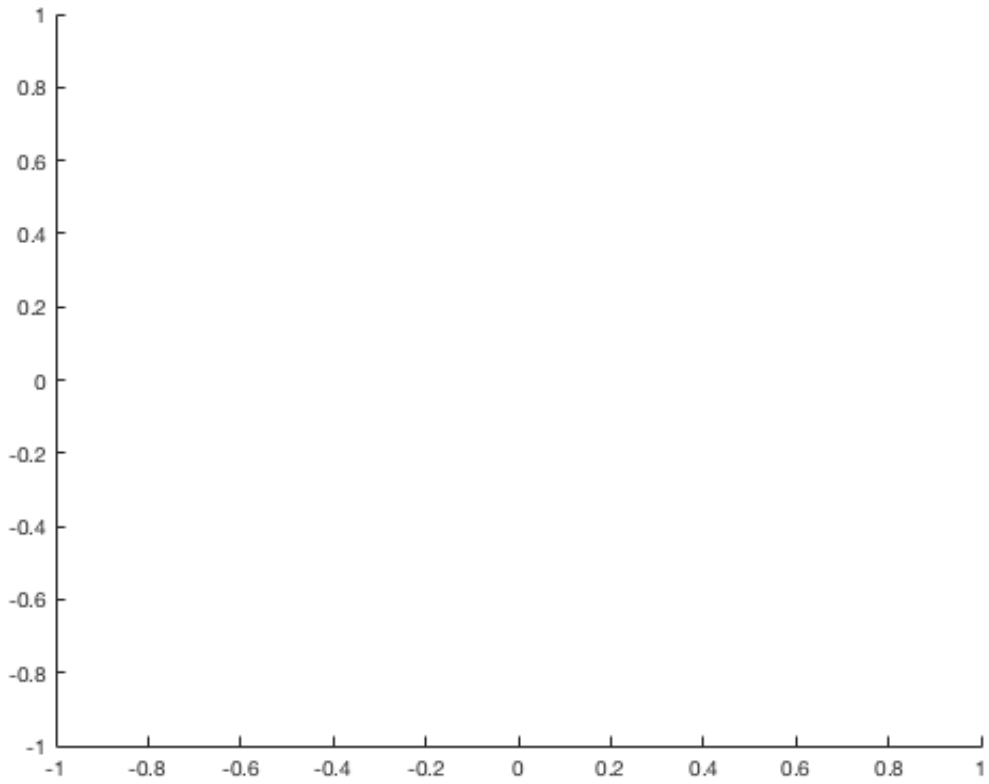
```
close all;
A=SGtrans1(SGbox([10,50,10])); B=SGtrans1(SGbox([50,10,10]));
nSG=SGcat({A,B});  SGplot (nSG); view (-30,30);
[VL,EL,PEL]=SGchecker (nSG);
if ~isempty(VL); VLELplots (VL(:,2:4),PEL,'m*-',4); VLFLplotlight (1,0.7); end
```



If we call SGchecker with a second argument ('plot'), we get a figure showing the non manifold objects that generate the conflict

```
close all; SGchecker (nSG,'plot');
```

```
2 edges [blue] are doubled, not removed
```



6. Additive Design: Separate or penetrate solid geometries (SG)

During additive solid geometry design, we will always get problems with non-manifold points or edges, as long as we try always to align objects point-to-point, edge-to-edge or face-to-face. As a basic rule, it is better to shorten or increase the length of a object slighly by a micrometer and do not align it with another face, edge, point. Even if the number of points of a solid geometry is increased by this strategy, the number of facets of this solid is decreased. Additive solid geoemtry design is therefore, not an inefficient but an efficient design methodology.

```
close all;
slot=1e-3;
A=SGtrans1(SGbox([10,50,10]));
B=SGtrans1(SGbox([50,10,10])); B=SGtransP(B,[slot;0;0]);
nSG=SGcat({A,B});
SGchecker (nSG,'plot');
```

The value for shifting the object about 1 micrometer is much lower than the manufacturing accuracy of the 3D printer. Anyway, if this would not be the case, then simply change it to 1 nanometer (1e-6) or one picometer (1e-9) if we consider a millimeter as default integer unit. It is also clear that we can automate the correction by simply splitting the objects and adding a random submicrometer value to the coordinates.

Another possibility is separating the objects instead of penetrating them. This will lead to the same solution to avoid non manifold edges. Nevertheless, some manufacturing preprocessors analyze STL-Files and detect objetcs that are separated and do not penetrate each other. These objects are then separated and repositioned in the 3D printing working volume to optimize the use of the print job's working volumen and material use.

The later presented function for relative spatial alignment of solid geometries will support a parameter for a gap between objects. Negative gap sizes correspond to a slightly penetration of the solid geometries.

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:34:07!
Executed 16-Sep-2018 16:34:09 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-23*
- *Tim Lueth, executed and published on 64 Bit PC using Windows with Matlab 2014b on YYYY-MMM-DD*

Published with MATLAB® R2018a

Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)

2014-11-24: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 1.8 required)
- 2. Relative spatial positioning of solid geometries (SG) using bounding boxes
- 3. Relative spatial alignment of solid geometries (SG) using bounding boxes
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages

- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 1.8 required)

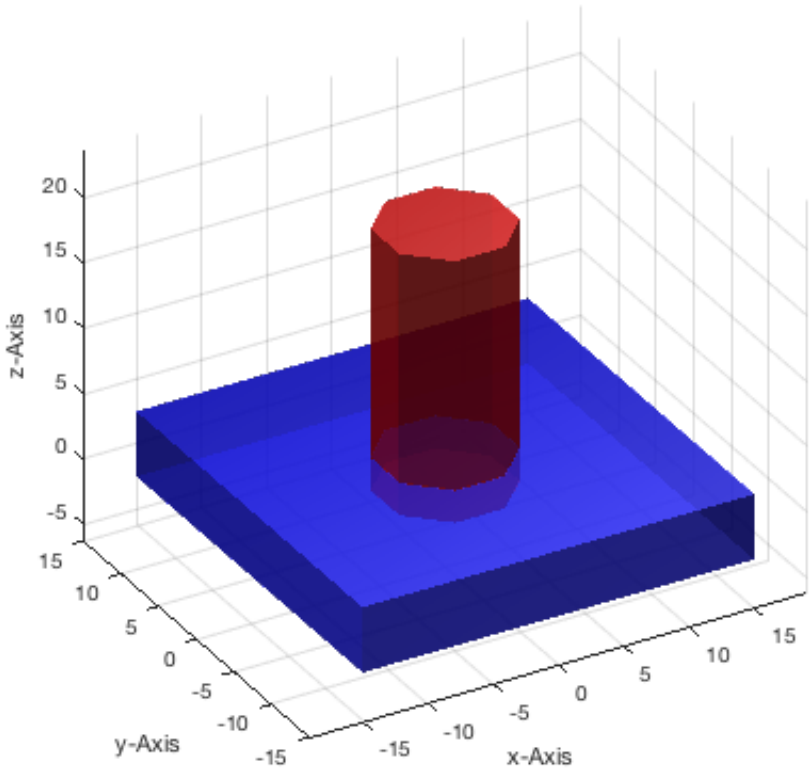
2. Relative spatial positioning of solid geometries (SG) using bounding boxes

Since it is quite convenient to use solid geometries (SG), there is a need for relative spatial positioning of these objects. For 'on top', 'under' (modifies the z-coordinates), 'in front', 'behind' (modifies the y-coordinates), 'left' and 'right' (modifies the x-coordinates), we have six different positioning functions that generate copies of the SG with just a changed vertex list. A third parameter of those functions is a gap, that can be defined. A positive gap value means a separation of thoses solids, a negative gap value means a penetration of those solids.

- **SGontop** positions a solid geometry 'A' on top of solid geometry 'B'
- **SGunder** positions a solid geometry 'A' under of solid geometry 'B'
- **SGinfront** positions a solid geometry 'A' in front of solid geometry 'B'
- **SGbehind** positions a solid geometry 'A' behind of solid geometry 'B'
- **SGleft** positions a solid geometry 'A' left of solid geometry 'B'
- **SGright** positions a solid geometry 'A' right of solid geometry 'B'

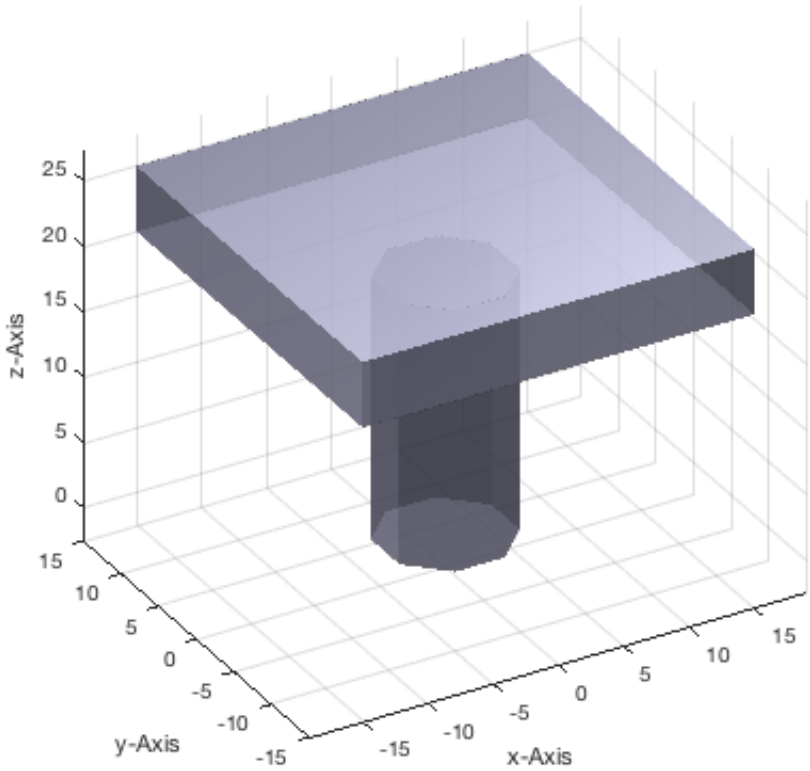
Define two solid geometrys 'A' and 'B'

```
close;
A=SGbox([30,30,5]); B=SGofCPLz(PLcircle(5,8),20);
SGplot (A,'b'); SGplot (B,'r'); VLFLplotlight(1,0.7); view (-30,30);
```



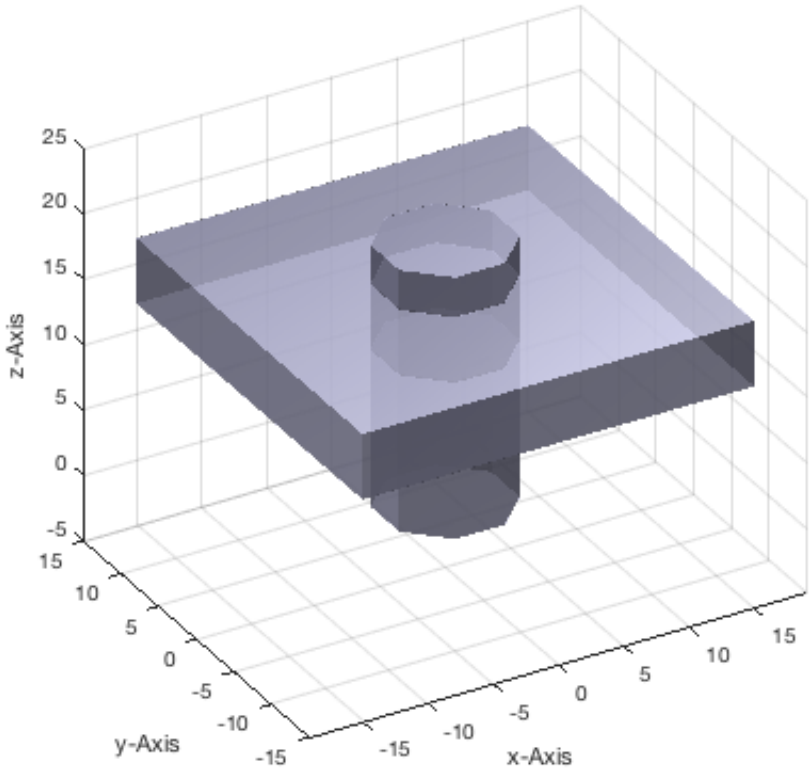
SGontop positions a solid geometry 'A' on top of solid geometry 'B'

```
close;
SG=SGcat(SGontop(A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



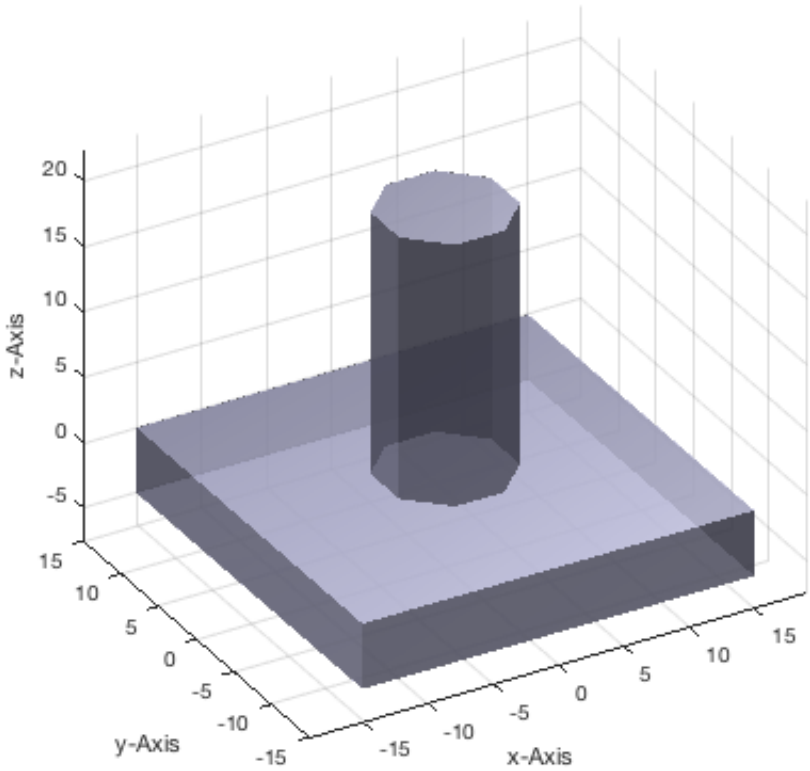
SGontop positions a solid geometry 'A' on top of solid geometry 'B' with a gap of -8

```
close all;
SG=SGcat(SGontop(A,B,-8),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



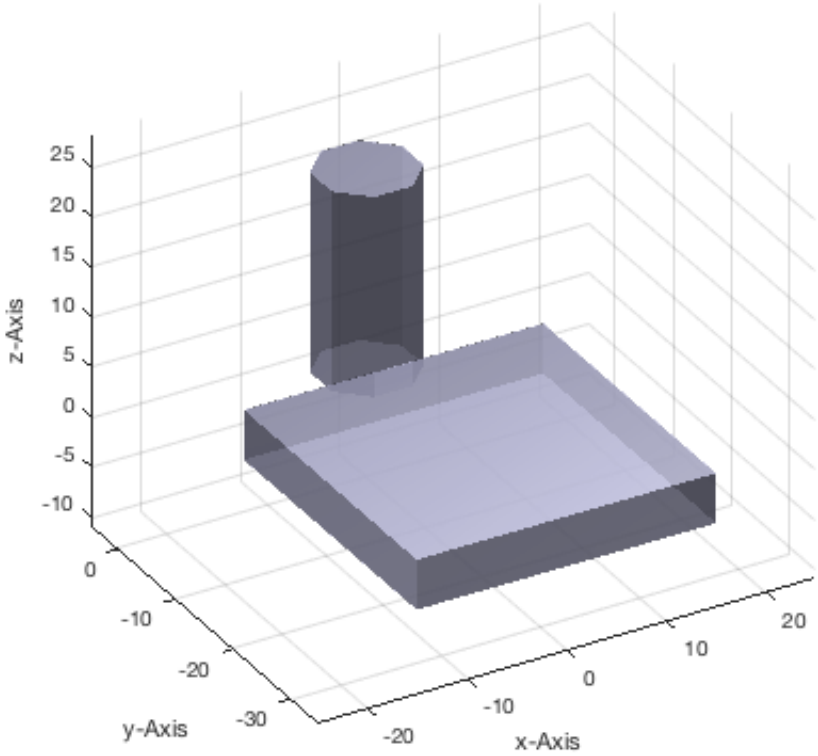
SGunder positions a solid geometry 'A' under of solid geometry 'B'

```
close all;
SG=SGcat(SGunder(A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



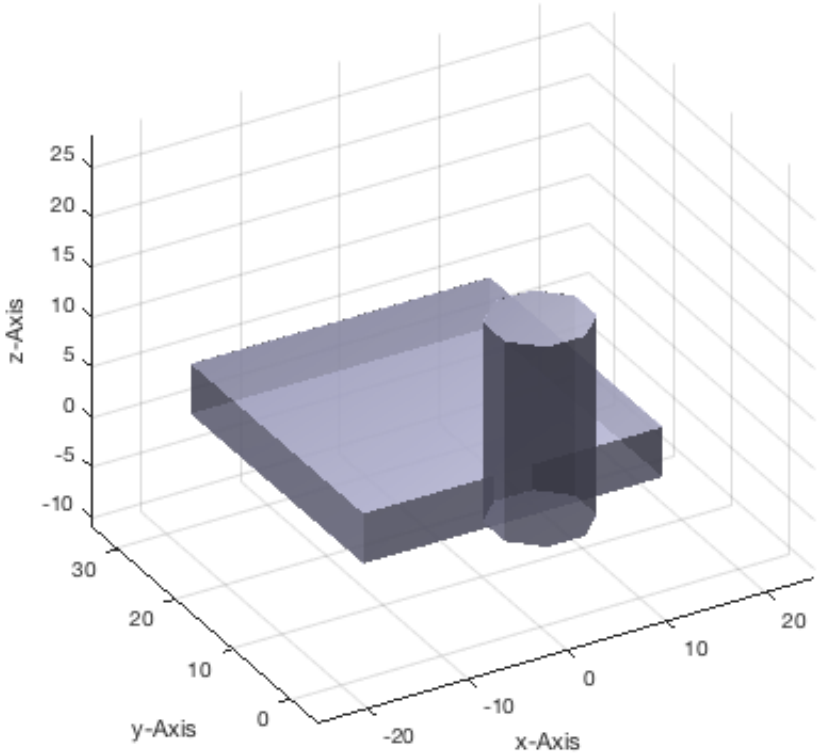
SGinfront positions a solid geometry 'A' in front of solid geometry 'B'

```
close all;
SG=SGcat(SGinfront (A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



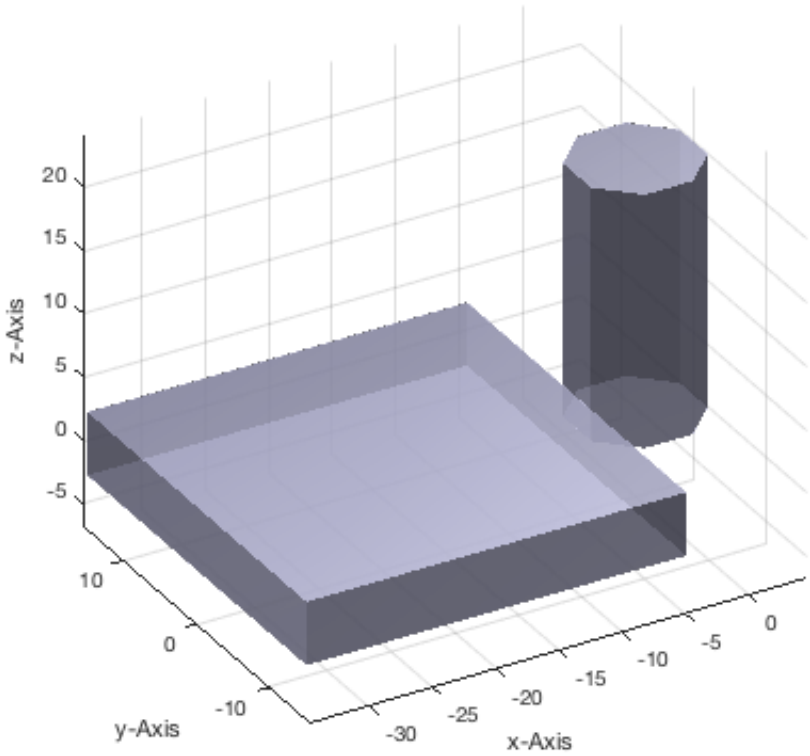
SGbehind positions a solid geometry 'A' behind of solid geometry 'B'

```
close all;
SG=SGcat(SGbehind (A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



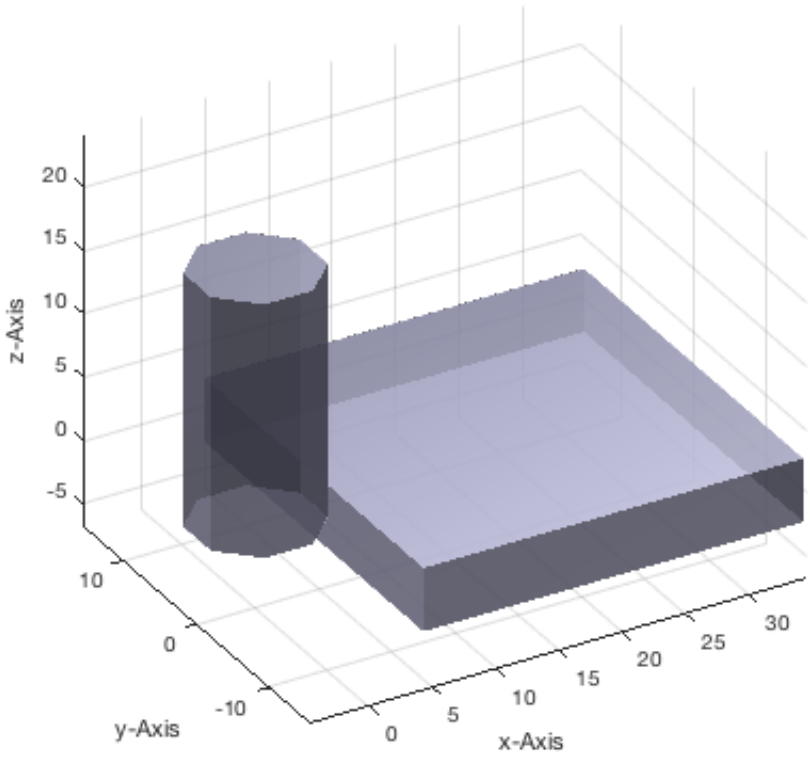
SGleft positions a solid geometry 'A' left of solid geometry 'B'

```
close all;
SG=SGcat(SGleft (A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



SGright positions a solid geometry 'A' right of solid geometry 'B'

```
close all;
SG=SGcat(SGright (A,B),B);
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



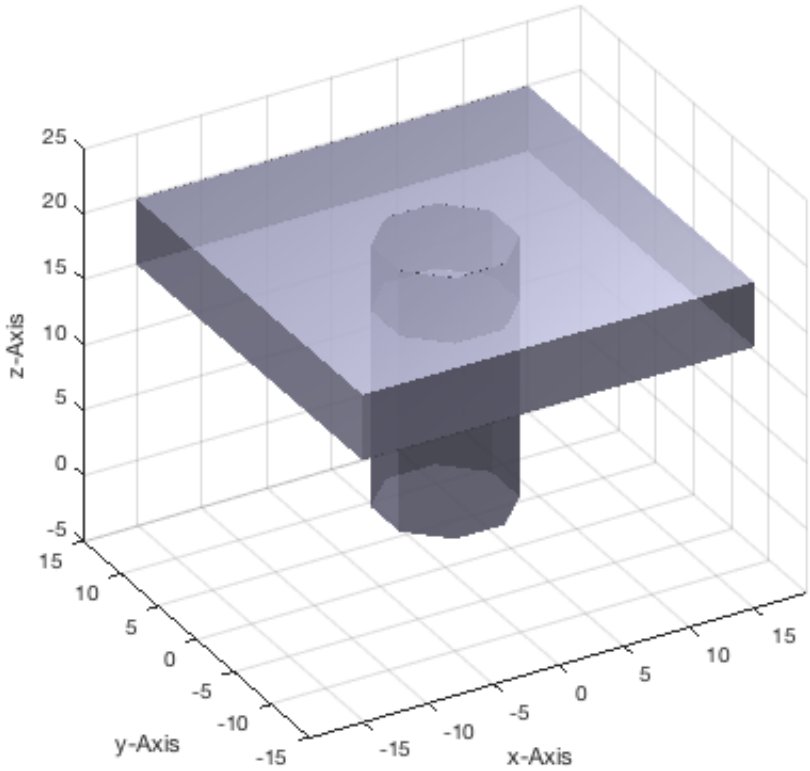
3. Relative spatial alignment of solid geometries (SG) using bounding boxes

Similar to the relative positioning, also the spatial alignment is helpful. For example solid A is aligned with solid B to achive the same 'top', 'bottom' (modifies the z-coordinates), 'front', 'back' (modifies the y-coordinates), 'left side' or 'right side' (modifies the x-coordinates).

- **SGaligntop** aligns the top of solid A with the top of solid B
- **SGalignbottom** aligns the bottom of solid A with the bottom of solid B
- **SGalignfront** aligns the front of solid A with the front of solid B
- **SGalignback** aligns the back of solid A with the back of solid B
- **SGalignleft** aligns the left side of solid A with the left side of solid B
- **SGalignright** aligns the right side of solid A with the right side of solid B

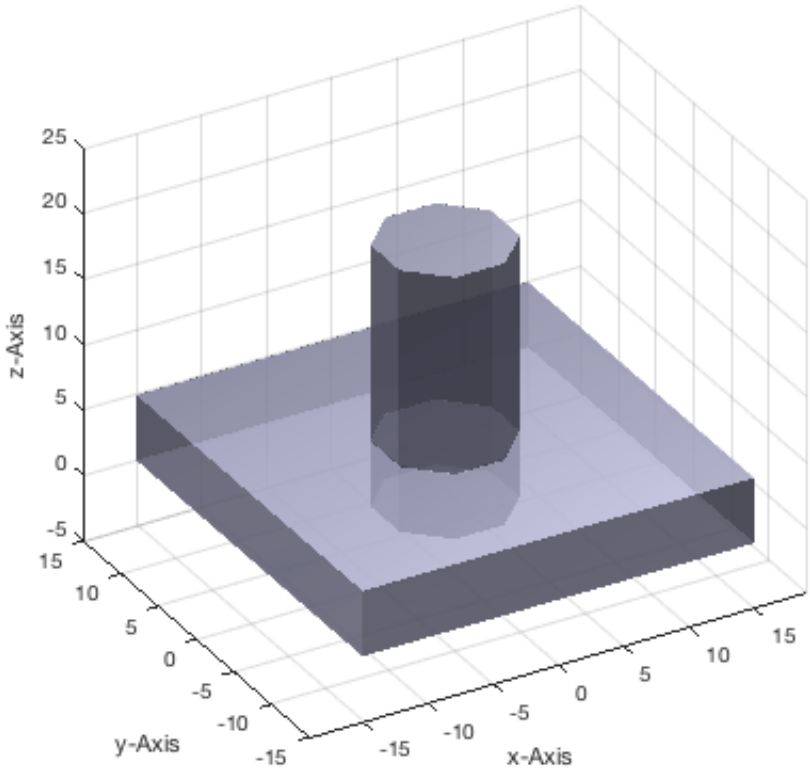
SGaligntop aligns the top of solid A with the top of solid B

```
close all;
SG=SGcat({SGaligntop(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



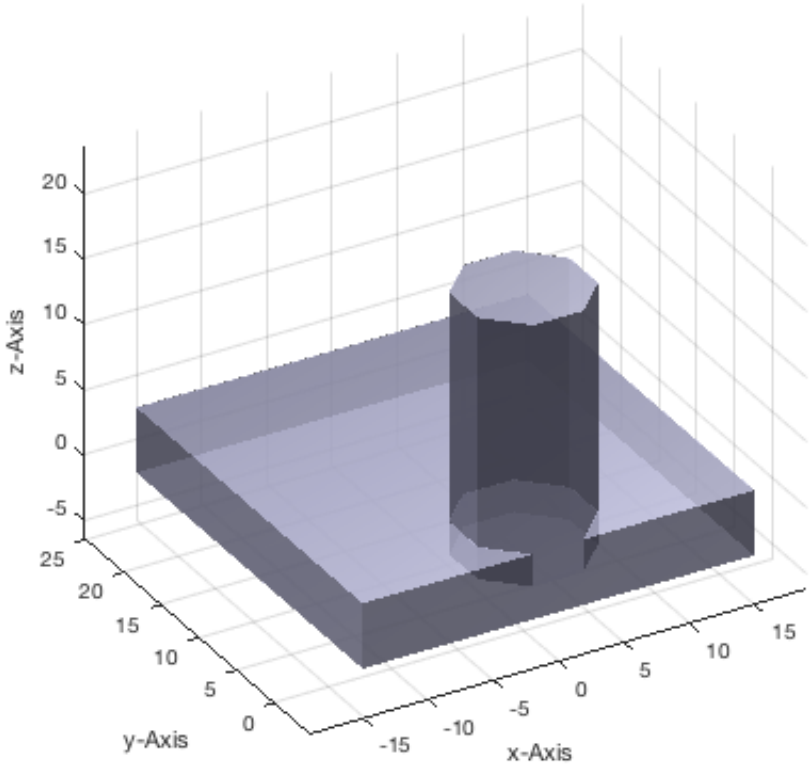
SGalignbottom aligns the bottom of solid A with the bottom of solid B

```
close all;
SG=SGcat({SGalignbottom(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



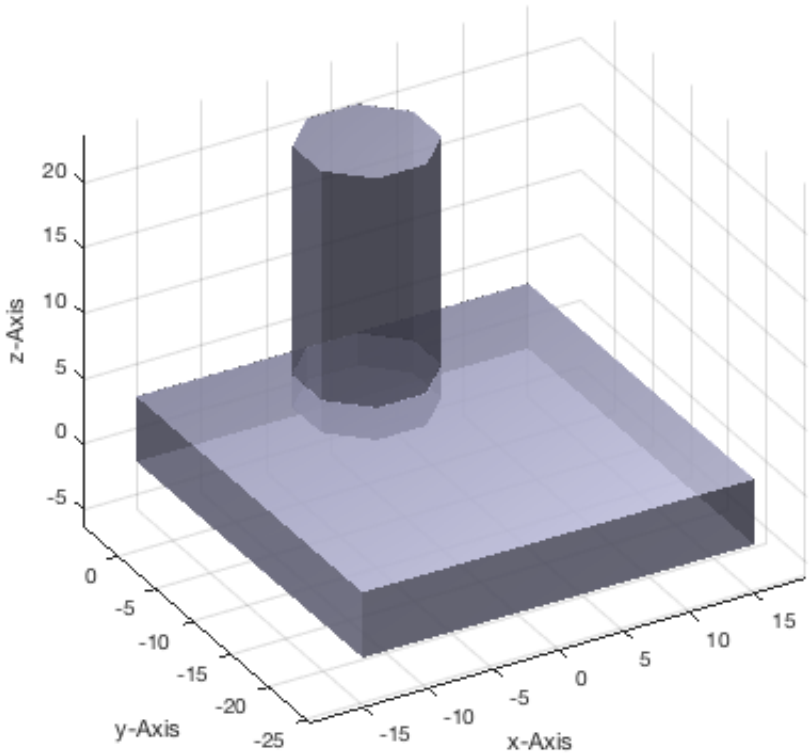
SGalignfront aligns the front of solid A with the front of solid B

```
close all;
SG=SGcat({SGalignfront(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



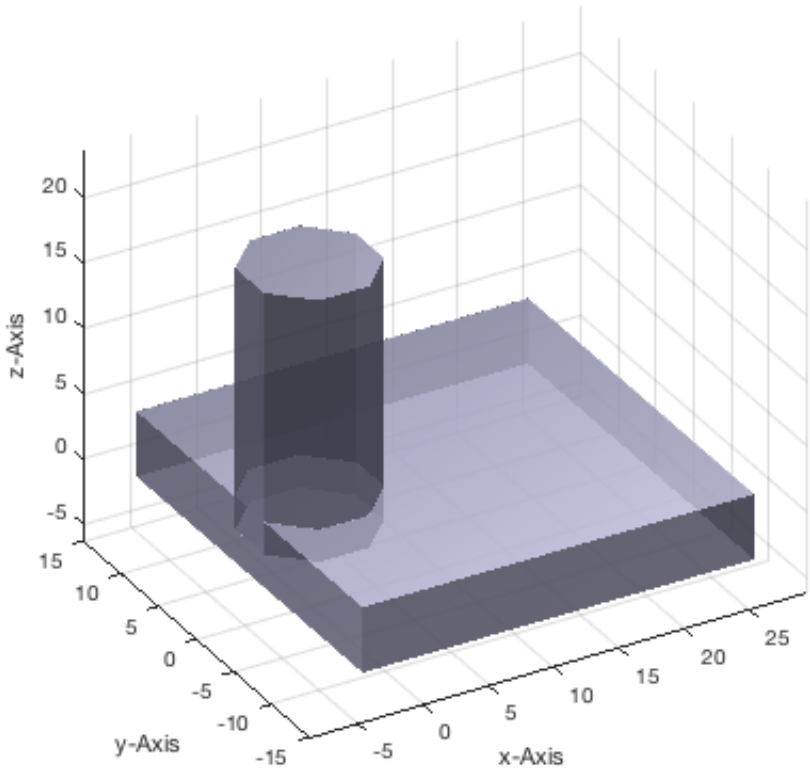
SGalignback aligns the back of solid A with the back of solid B

```
close all;
SG=SGcat({SGalignback(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



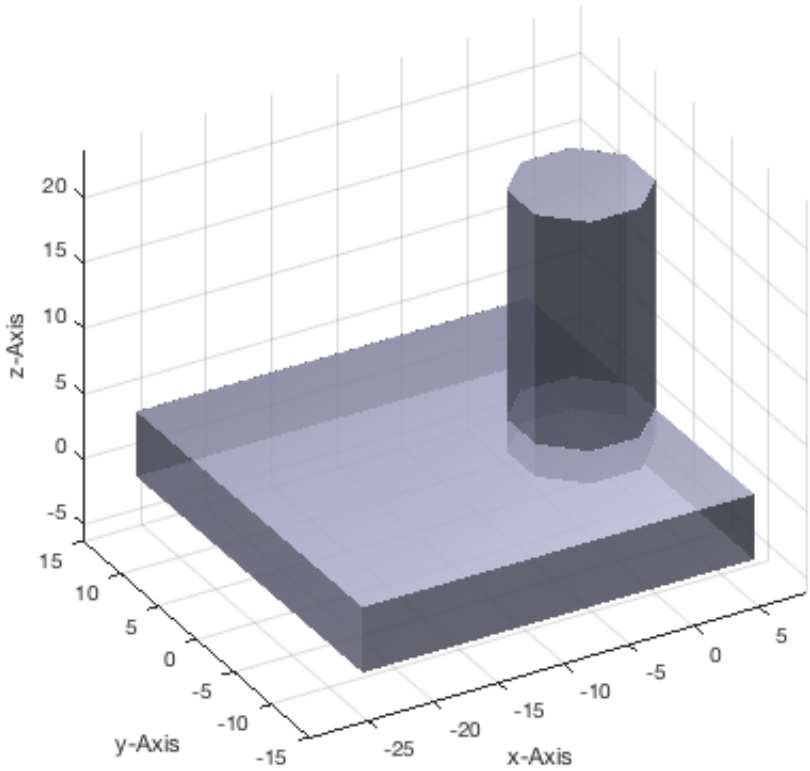
SGalignleft aligns the left side of solid A with the left side of solid B

```
close all;
SG=SGcat({SGalignleft(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



SGalignright aligns the right side of solid A with the right side of solid B

```
close all;
SG=SGcat({SGalignright(A,B),B});
SGplot (SG,'w'); VLFLplotlight(1,0.7); view (-30,30);
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:34:17!
Executed 16-Sep-2018 16:34:19 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-25*
- *Mattias Traeger, executed and published on 64 Bit PC using Windows with Matlab 2014b on YYYY-MMM-DD*

Published with MATLAB® R2018a

Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design

2014-11-26: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.0 required)
- 2. List of functions used in this example
- 3. Rotation of closed polygon lists (CPL)
- 4. Creating spheres by rotating half-circles
- 5. Creating embedded contours
- 6. Rotate Contours around the z-axis
- 7. Samples of 3D Design
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.0 required)

2. List of functions used in this example

As we learned in example 2 and 4, it is possible to extrude a planar point list (PL) or closed polygon (CPL) list into a 2.5D solid geometry. Now, we will rotate a CPL around th z-axis. In this case, we consider the CPL or the PL always as a x/z-list. Using closed polygon lists, we have to remember that before extruding them or rotating them it is necessary to guarantee that the outer contour has a counter-clockwise order (ccw).

In this example, some new functions are introduced:

- CPLplot to draw the closed polygon list in the x/y plane.
- PLELofCPL to draw the direction, starting point and end point.
- CPLuniteCPL to unite several CPL into one and adapt their original directions.
- SGofCPLrot to rotate a contour around the z-axis

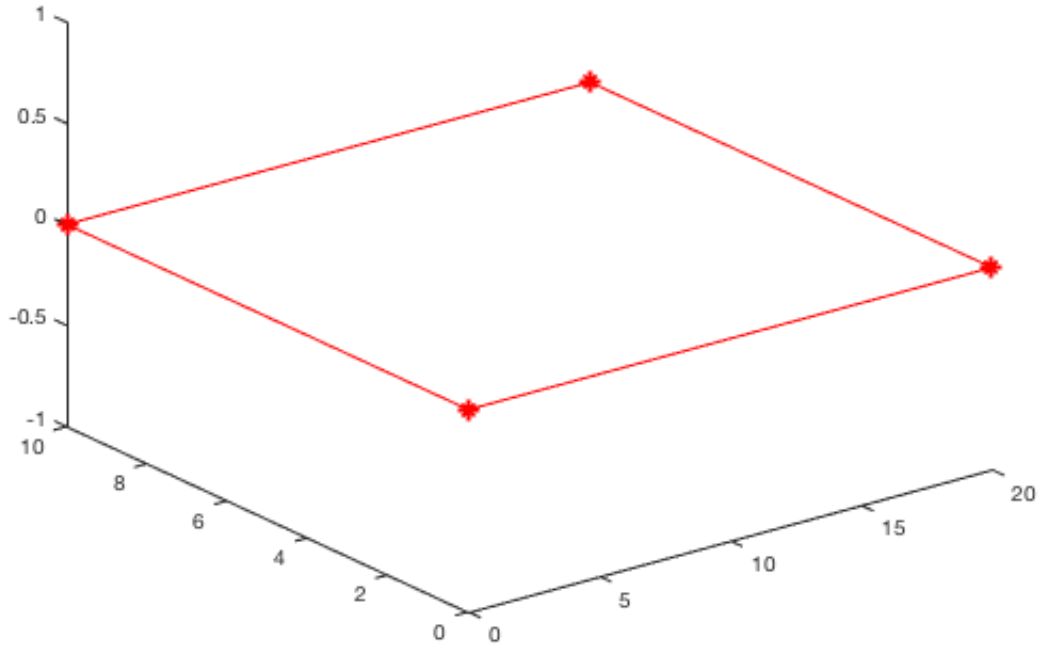
3. Rotation of closed polygon lists (CPL)

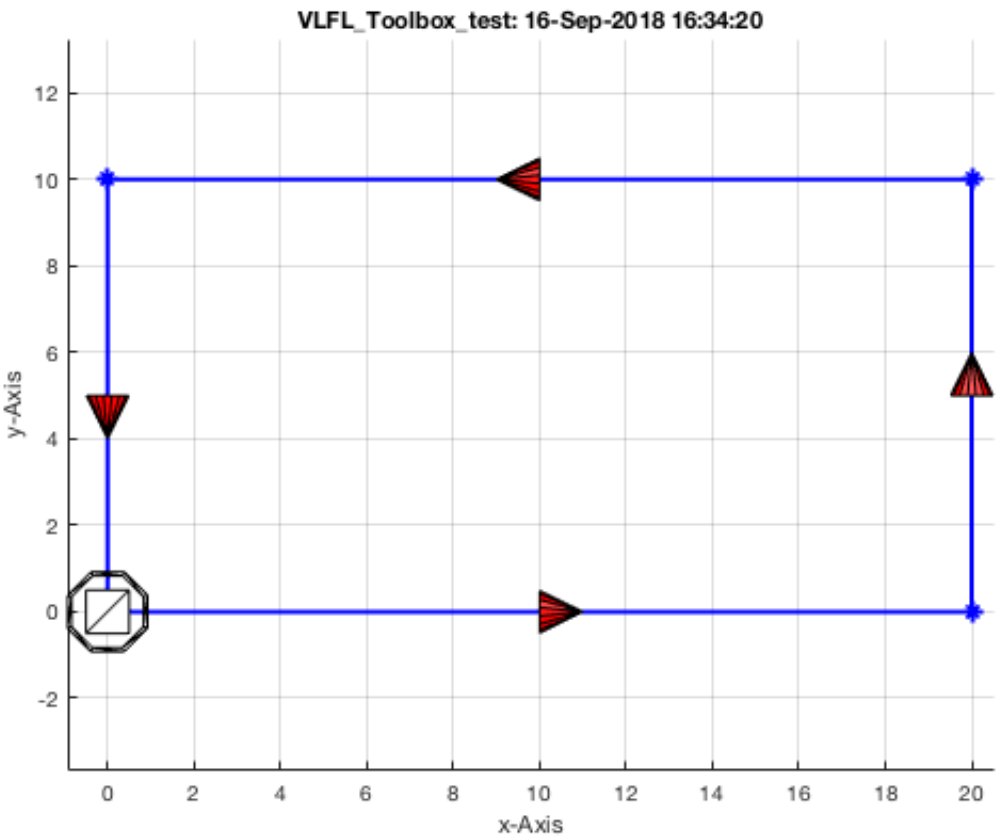
For the rotation of a simple contour we use the following functions

- **CPLplot** to draw the closed polygon list in the x/y plane.
- **PLELofCPL** to draw the direction, starting point and end point.
- **SGofCPLrot** to rotate a contour around the z-axis

Exercise: Create a simple point list that touches the y-axis

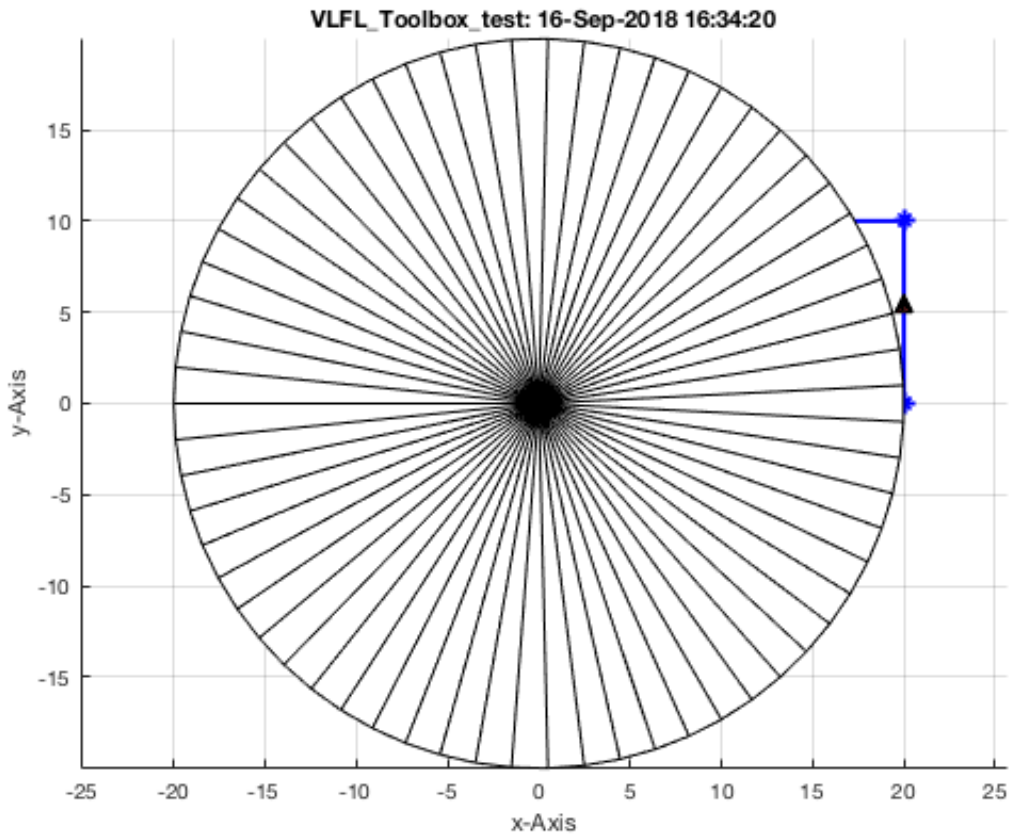
```
close all;
CPL=[0 0; 20 0; 20 10; 0 10];       % Create a simple rectangle (ccw)
CPLplot(CPL);                       % plot the rectangle
PLELofCPL(CPL);                     % show edges and directions
```





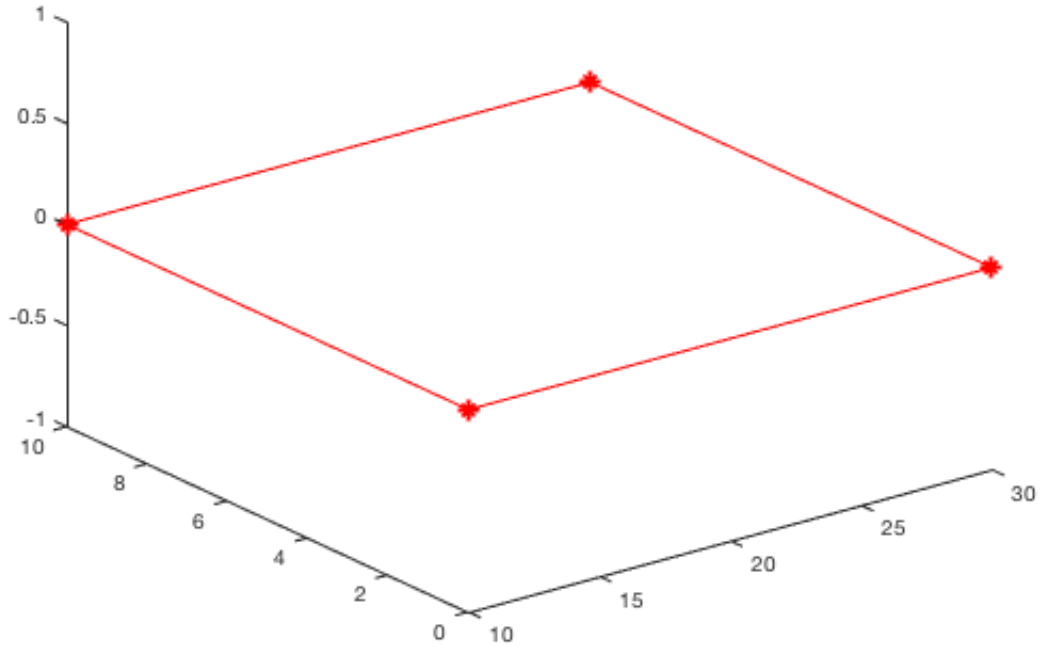
Exercise: Rotate the point list around the z-axis to create a cylinder

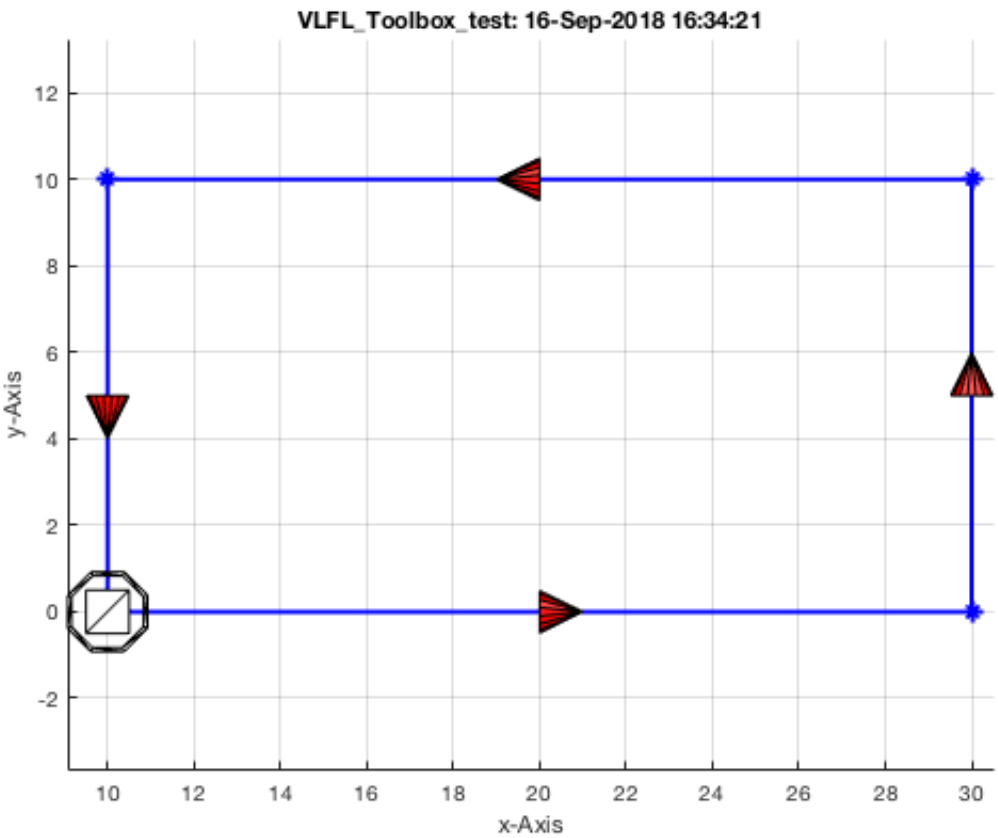
```
SG=SGofCPLrot(CPL);                 % Solid contour rotation
SGplot(SG);                         % show the solid
```



Exercise: Create a simple point list with distance to the y-axis

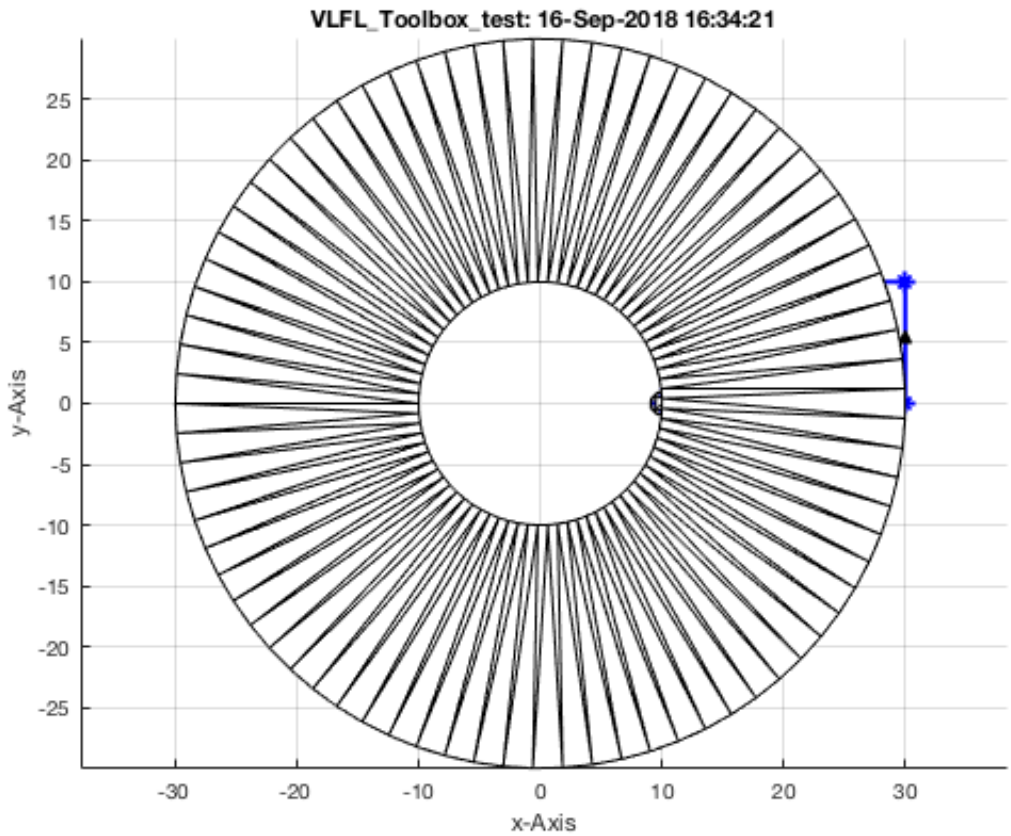
```
close all;
CPL=[0 0; 20 0; 20 10; 0 10];       % Create a simple rectangle (ccw)
CPL(:,1)=CPL(:,1)+10;               % shift by 1 on the x-axis
CPLplot(CPL);                       % plot the rectangle
PLELofCPL(CPL);                     % show edges and directions
```





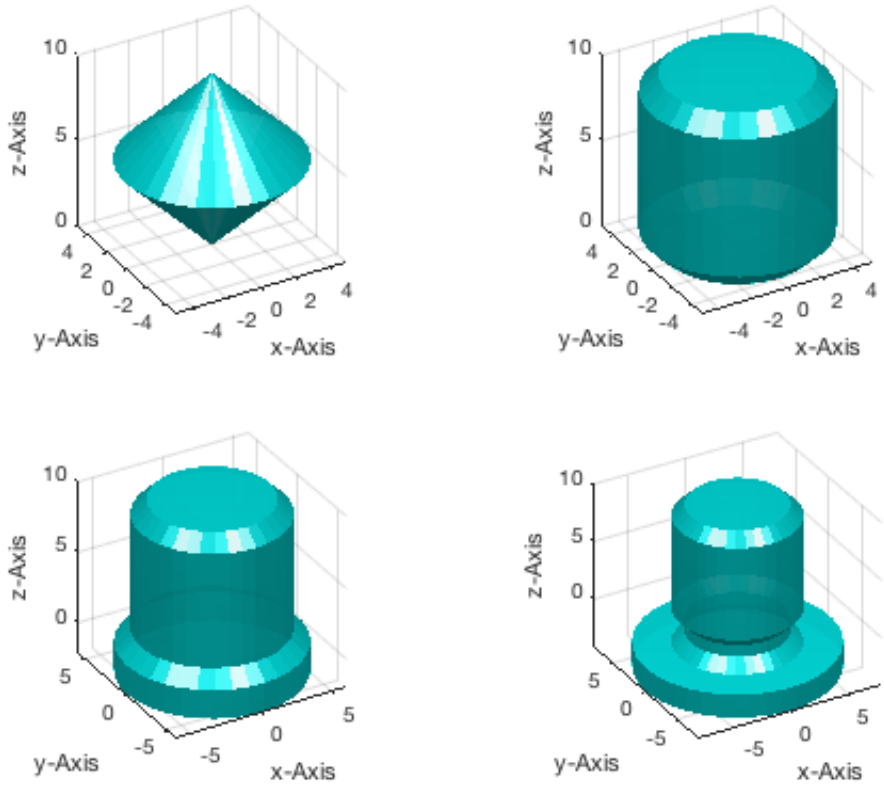
Exercise: Rotate the point list around the z-axis to create a hollow cylinder

```
SG=SGofCPLrot(CPL);                    % Solid contour rotation
SGplot(SG,'b');
```



Exercise: Some other examples for massiv rotational symetric solids

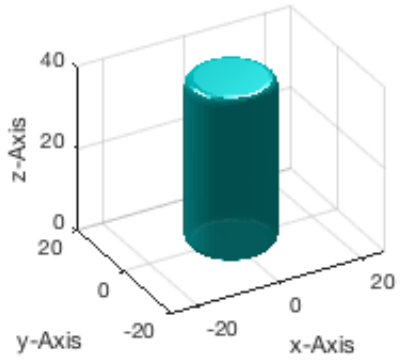
```
SG=SGofCPLrot([0 0; 5 5; 0 10]);                    % Solid contour rotation
subplot(2,2,1); view (-30,30); SGplot(SG,'c'); VLFLplotlight (1,0.9);
SG=SGofCPLrot([0 0; 4 0;  5 1; 5 9; 4 10; 0 10;]); % Solid contour rotation
subplot(2,2,2); view (-30,30); SGplot(SG,'c'); VLFLplotlight (1,0.9);
SG=SGofCPLrot([0 -2; 6 -2; 6 0;  5 1; 5 9; 4 10; 0 10;]); % Solid contour rotation
subplot(2,2,3); view (-30,30); SGplot(SG,'c'); VLFLplotlight (1,0.9);
SG=SGofCPLrot([0 -4; 8 -4; 8 -2; 5 -2; 4 -1; 4 0;  5 1; 5 9; 4 10; 0 10;]); % Solid contour
 rotation
subplot(2,2,4); view (-30,30); SGplot(SG,'c'); VLFLplotlight (1,0.9);
```

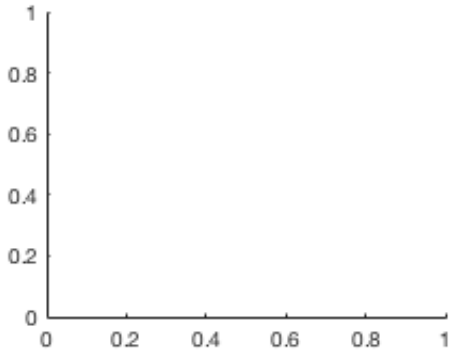


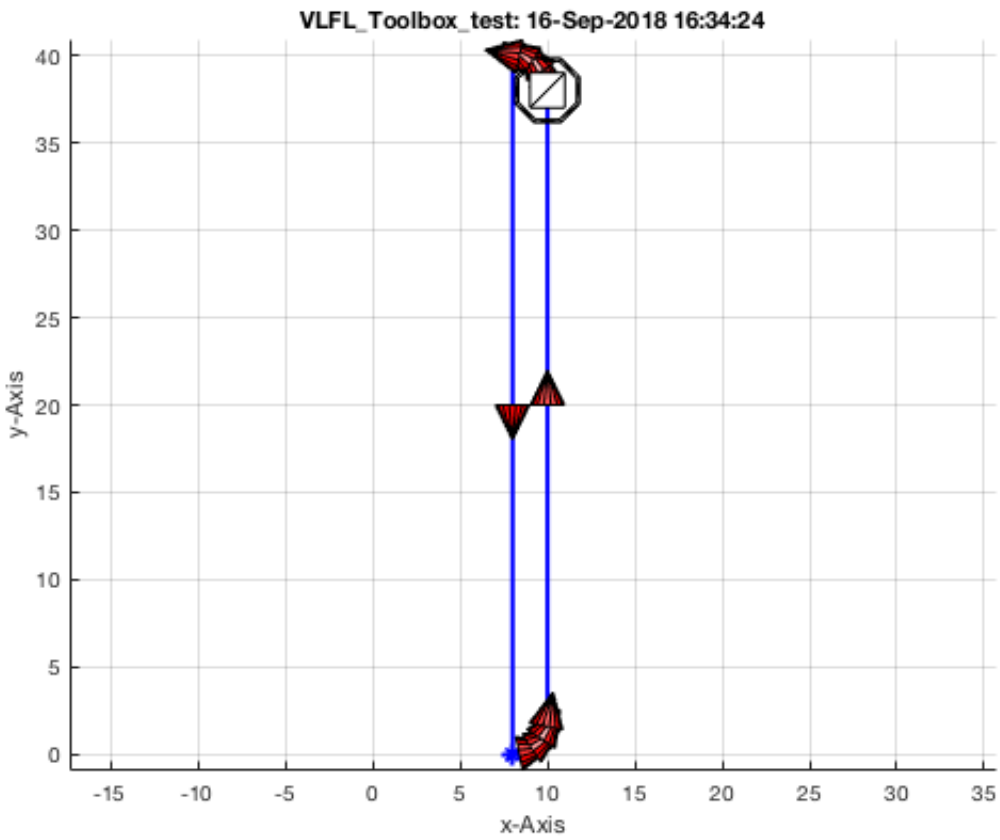
The warnings 'Removed n(m) facets' can be ignored. These warning appear if a part of the contour touches or crosses the x=0 line (y-axis).

Exercise: Creating a bold and a sleeve

```
closeall;
r=2; H=40; R=10;

PL=PLcircseg (r,[],0,pi/2);  CPL=PLtransP(PL,[R-r,H-r]);
PL=PLcircseg (r,[],-pi/2,0); CPL=[CPL;0 H; 0 0;PLtransP(PL,[R-r,r])];
SG=SGofCPLrot(CPL);                  % Solid contour rotation

subplot(2,2,1); SGplot(SG,'c'); VLFLplotlight (1,0.9); view (-30,30);
subplot(2,2,3); PLELofCPL (CPL);

PL=PLcircseg (r,[],0,pi/2);  CPL=PLtransP(PL,[R-r,H-r]);
PL=PLcircseg (r,[],-pi/2,0); CPL=[CPL;PLtransP(PL,[R-r,r])];
SG=SGofCPLrot(CPL);
subplot(2,2,2); SGplot(SG,'c'); VLFLplotlight (1,0.9); view (-30,30);
subplot(2,2,4); PLELofCPL (CPL);
```



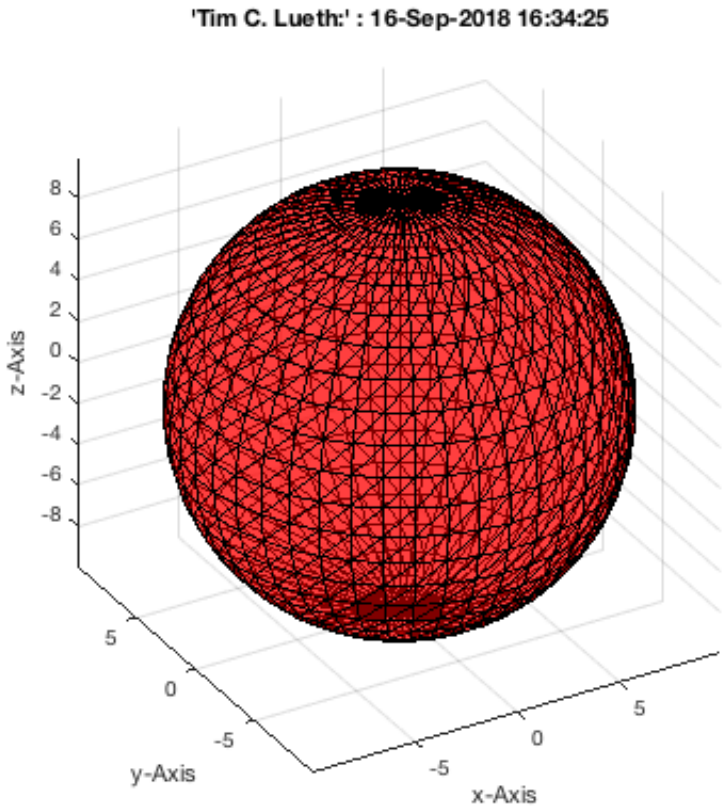




4. Creating spheres by rotating half-circles

Exercise: Creating a full sphere

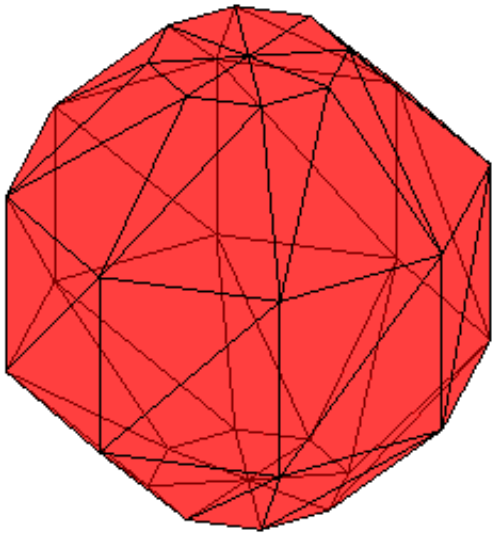
```
close all;
PL=PLcircle(10);

VLFLfigure; view(-30,30); grid on;
SG=SGofCPLrot(PL);
SGplot(SG); VLFLplotlight (0,0.5);
```



Exercise: Creating a 8 by 8 sphere

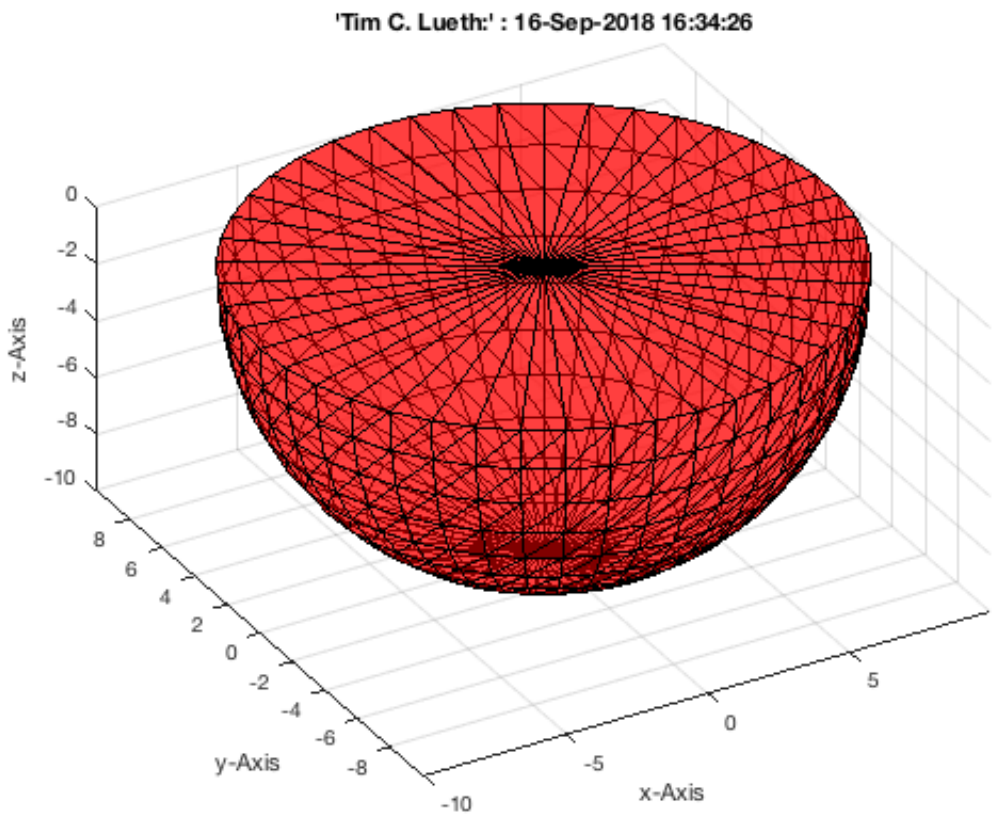
```
close all;
PL=PLcircle(10,8);

VLFLfigure; view(-30,30); grid on;
SG=SGofCPLrot(PL,8);
SGplot(SG); VLFLplotlight (0,0.5);
```

'Tim C. Lueth:' : 16-Sep-2018 16:34:25



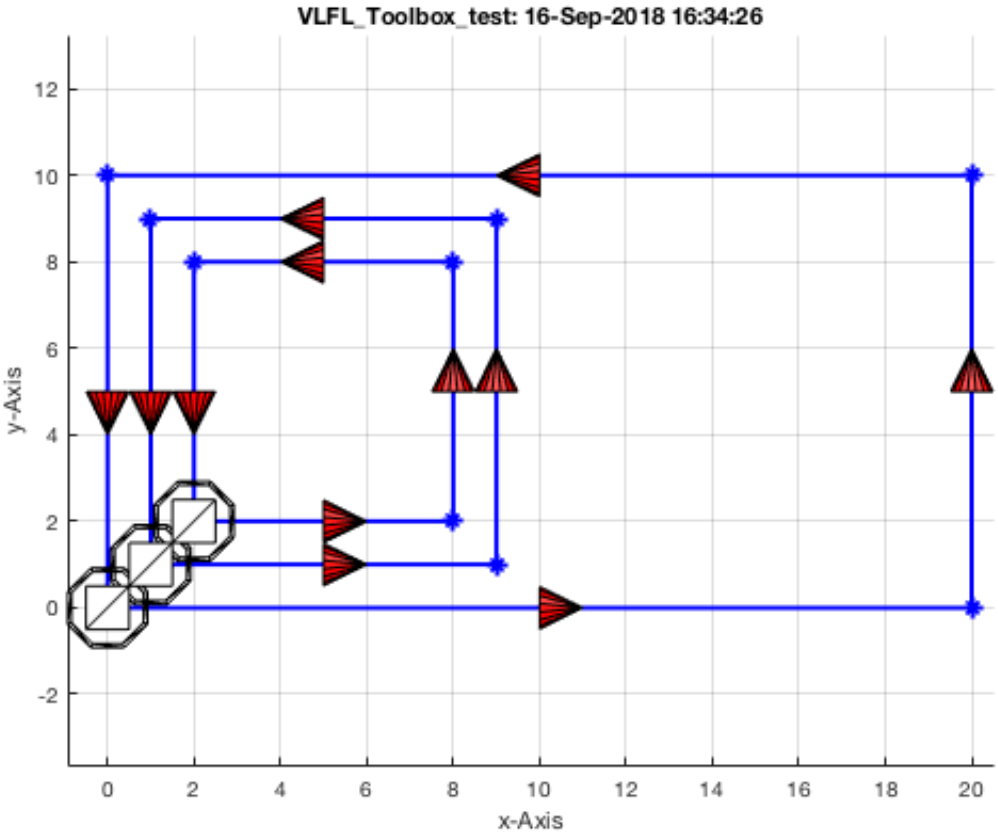
Exercise: Creating a half sphere

```
close all;
PL=[PLcircseg(10,[],-pi/2,0); 0 0];

VLFLfigure; view(-30,30); grid on;
SG=SGofCPLrot(PL);
SGplot(SG); VLFLplotlight (0,0.5);
```

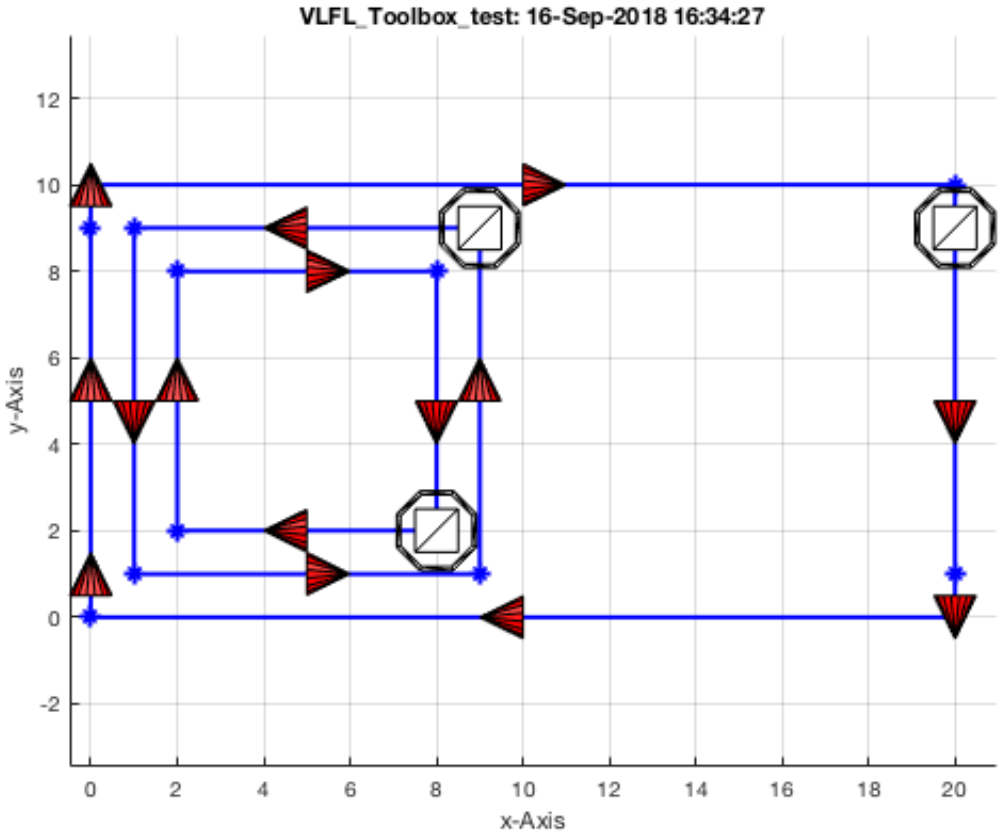


5. Creating embedded contours

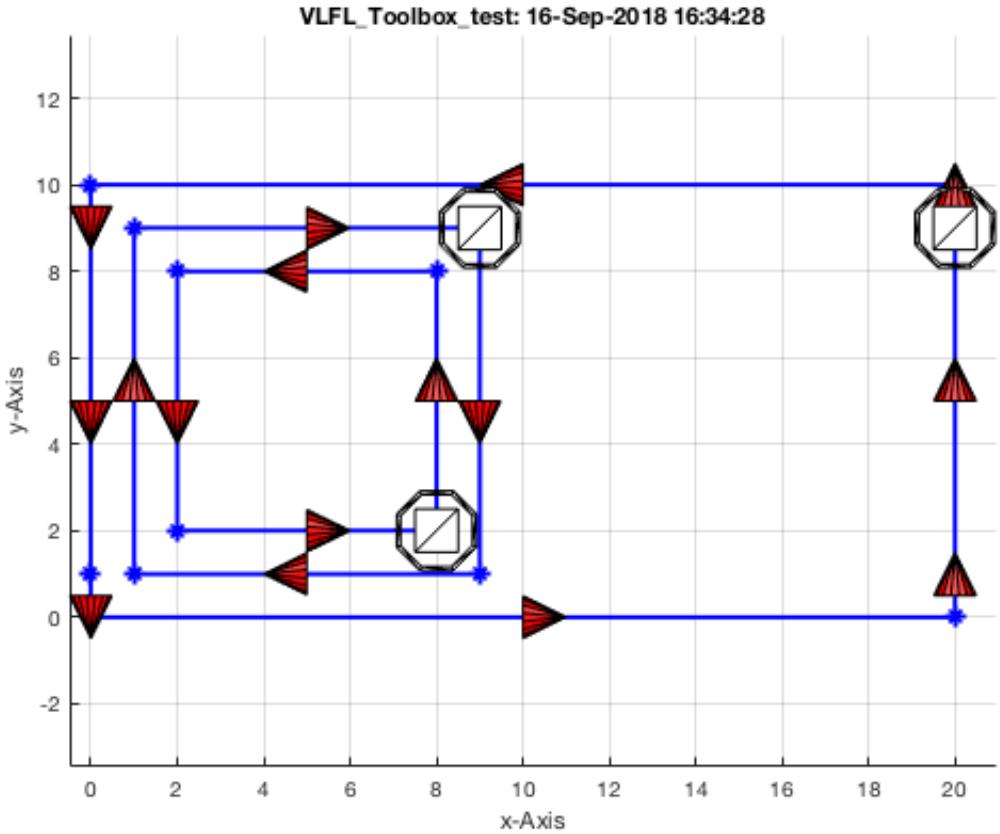
```
CPL=[0 0; 20 0; 20 10; 0 10; NaN NaN; 1 1; 9 1; 9 9; 1 9; NaN NaN; 2 2; 8 2; 8 8; 2 8 ];

close all;PLELofCPL(CPL);
```




```
CPL=CPLuniteCPL(CPL);
close all; PLELofCPL(CPL);
```

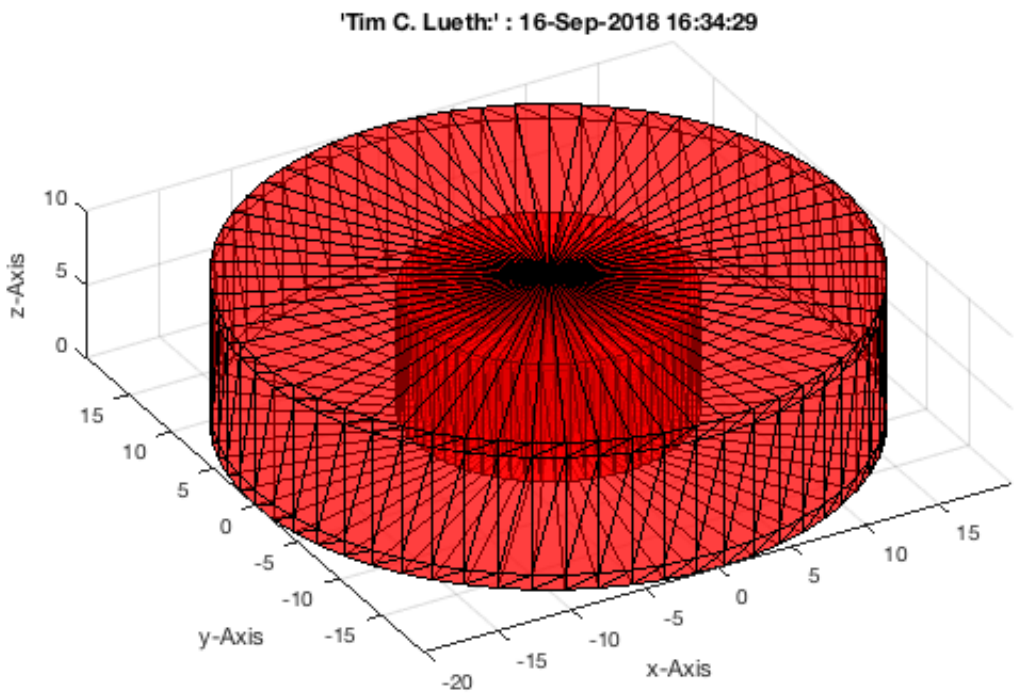



```
CPL=flip(CPL);
close all; PLELofCPL(CPL);
```



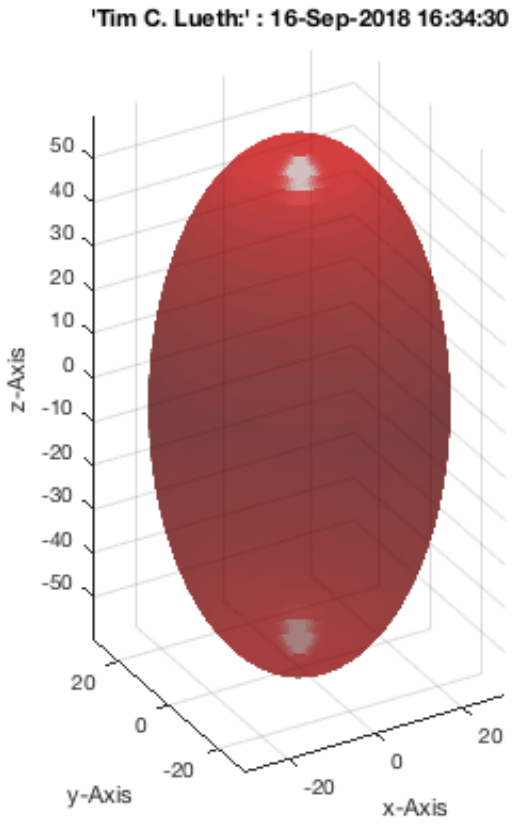
6. Rotate Contours around the z-axis

```
VLFLfigure; view(-30,30); grid on;
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (0,0.5);
```



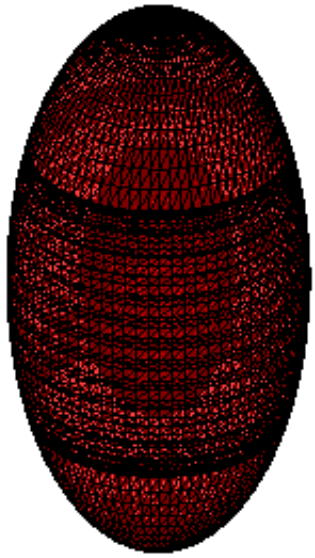

```
VLFLfigure; view(-30,30); grid on;
CPL(:,1)=CPL(:,1)+10;
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (1,0.5);
```

```
VLFLfigure; view(-30,30); grid on;
CPL=PLcircle(30); CPL(:,2)=CPL(:,2)*2;
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (1,0.5);
```

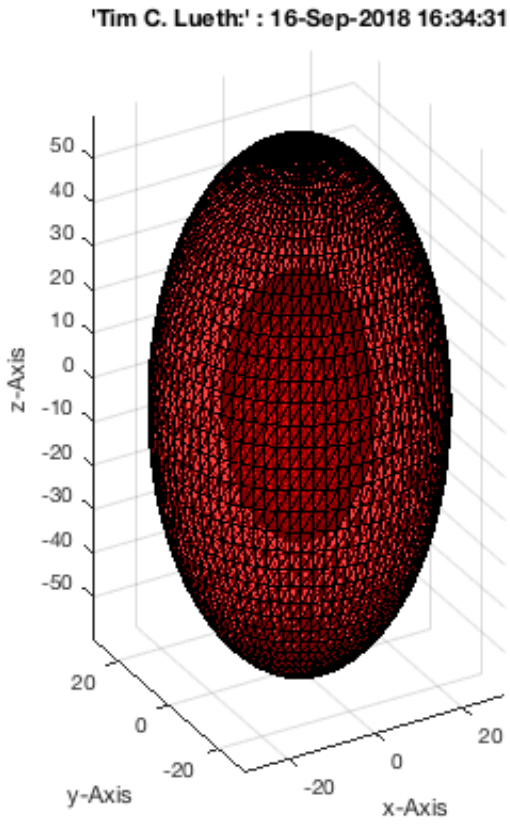



```
VLFLfigure; view(-30,30); grid on;
CPL=PLcircle(30); CPL(:,2)=CPL(:,2)*2;
CPL=[CPL;NaN NaN;CPL*0.5];
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (0,0.5);
```

'Tim C. Lueth:' : 16-Sep-2018 16:34:31



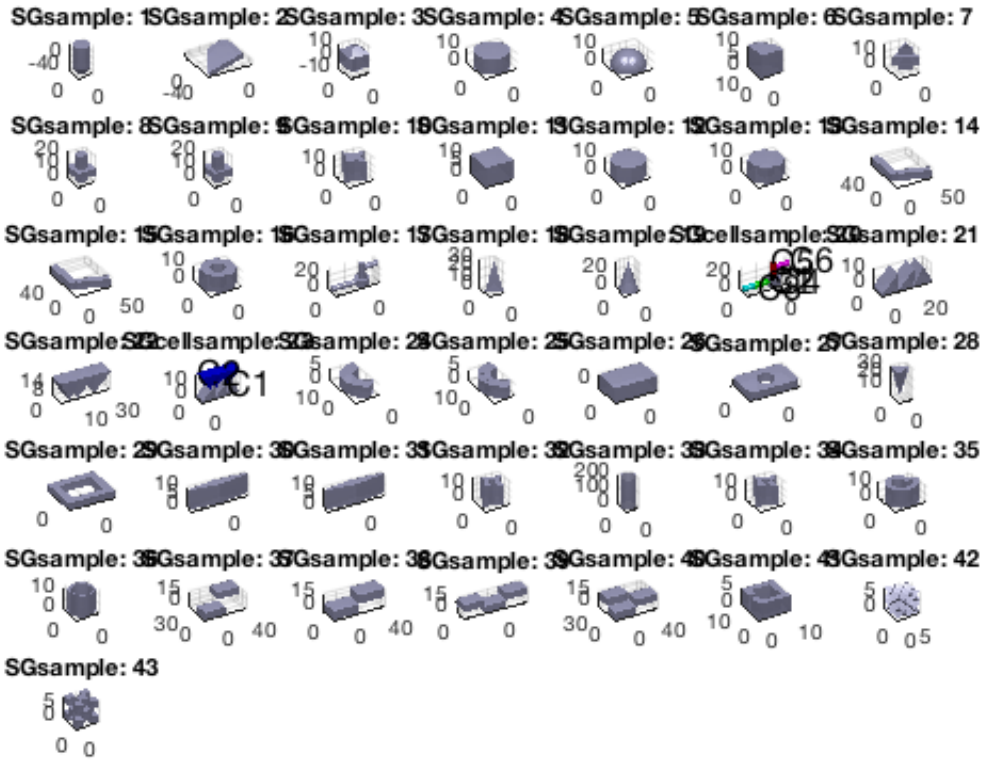
```
VLFLfigure; view(-30,30); grid on;
CPL=PLcircle(30); CPL(:,2)=CPL(:,2)*2;

SG=SGcat(SGofCPLrot(CPL),SGswap(SGofCPLrot(CPL*0.5)));
SGplot(SG); VLFLplotlight (0,0.5);
```



7. Samples of 3D Design

```
SGsample;
```

```
SGchecker "AXB":
SGchecker "A-B":
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:34:38!
Executed 16-Sep-2018 16:34:40 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
=================================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2014-11-30*
- *Mattias Traeger, executed and published on 64 Bit PC using Windows with Matlab 2014b on YYYY-MMM-DD*

Published with MATLAB® R2018a

Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries

2015-08-06: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.4 required)
- 2. Create a sample solid for this exercise
- 3. Analyze a slice plane through a solid geoemtry
- 4. Cutting and separating a solid geometries in two parts
- 5. Cutting as useful tool for the ending of complex shaped geoemtries
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model

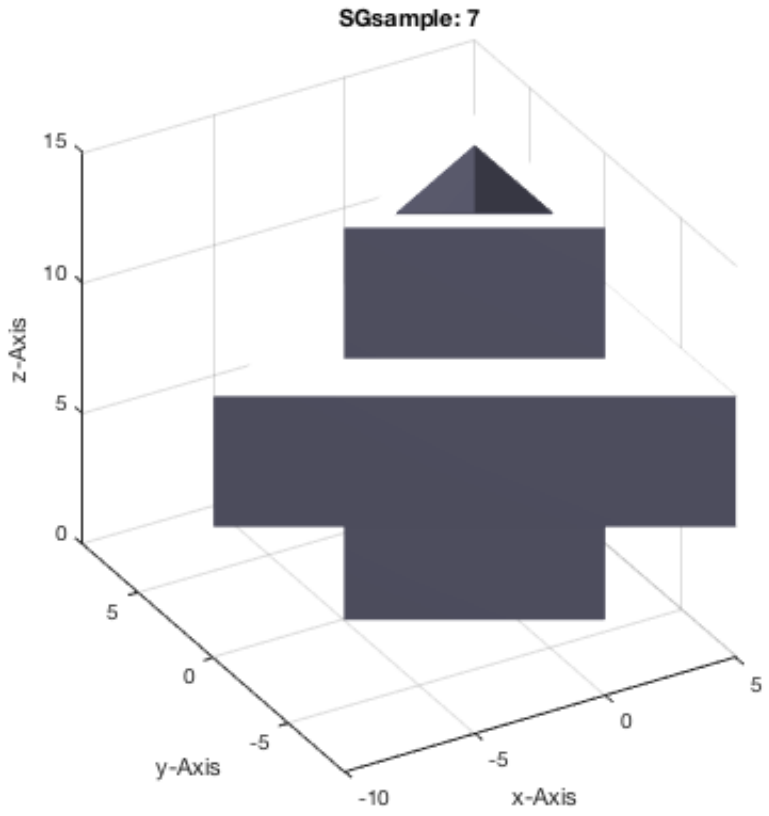
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.4 required)

2. Create a sample solid for this exercise

Using the function SGsample it is possible to create samples for an experiment, to see all of them or to select one.

```
close all
SGsample(7);
A=SGsample(7);
```

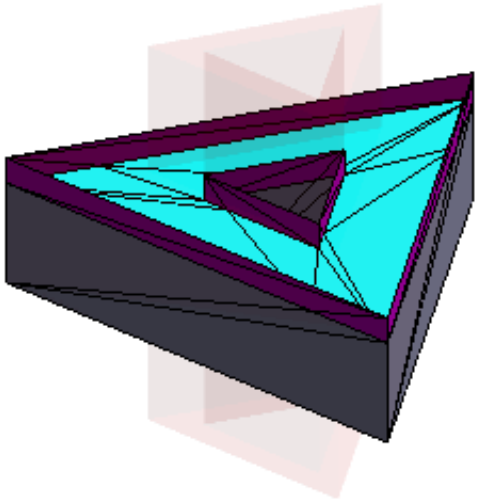


3. Analyze a slice plane through a solid geoemtry

Slicing at a specified z-coordinates is a more complex procedure than expected if several solids are processed that can penetrate each other. By slicing a single solid, the crossed facets/triangles are separated into 2 upper and lower parts that will lead to 2 lower and 1 upper facets or 1 lower and 2 upper facets depending on how many edges are above or under the cutting plane. For slicing we use the function **SGslicer**. Be aware that it is not possible to slice surfaces without crossing edges (i.e. surfaces in the z_max or z_min plane)

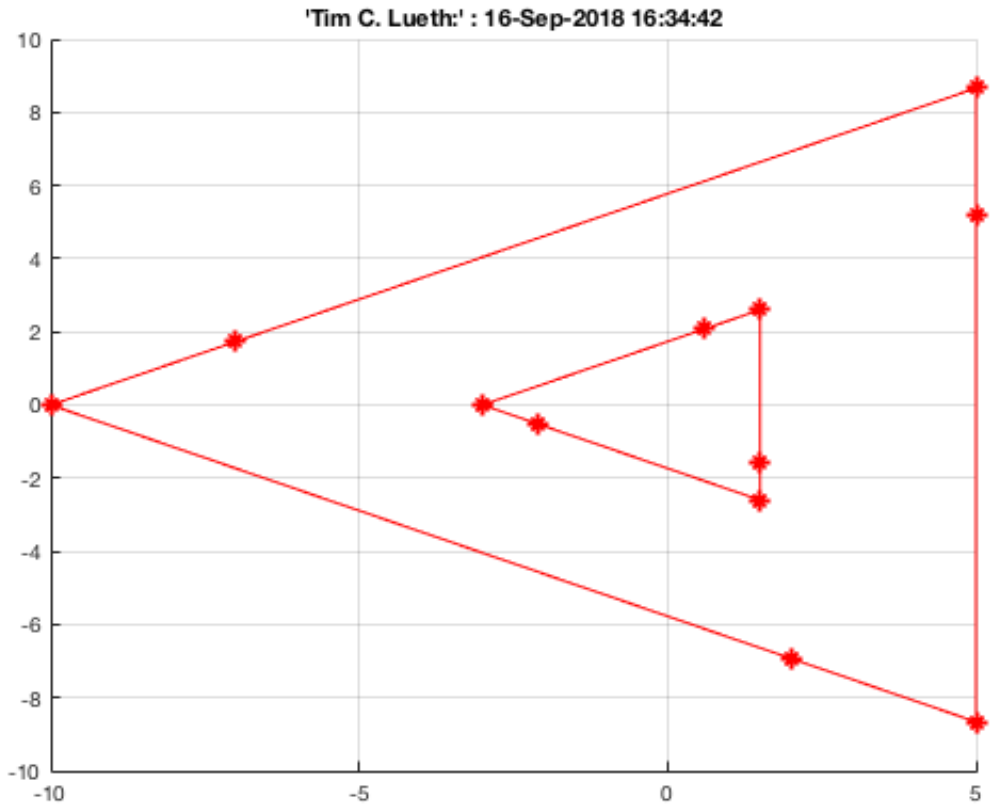
```
SGslicer (A,9);
view (10,30);
```

'Tim C. Lueth:' : 16-Sep-2018 16:34:41



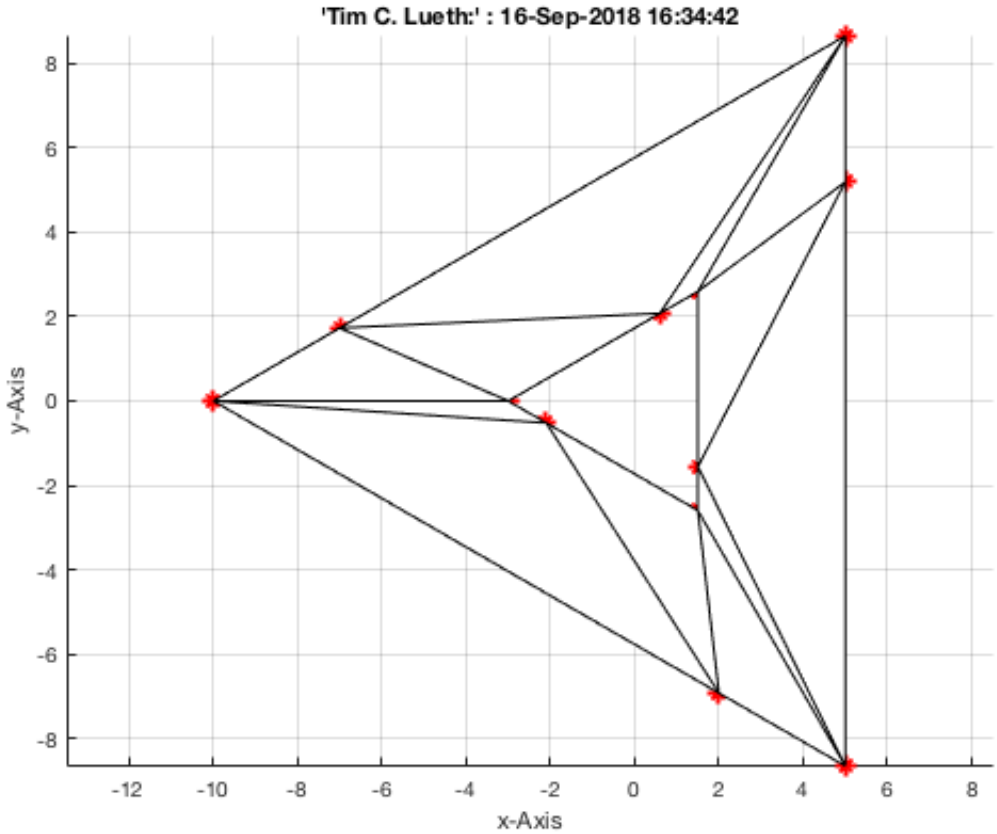
It is also possible just to show the cutting edges of the cutting contour

```
VLFLfigure;
TR2=SGslicer (A,9);
VLELplots(TR2.Points, TR2.Constraints);
```



The result of the slicing process is a delaunay triangulation of the cutting plane. It can be used as cover for closing the cutted solids.

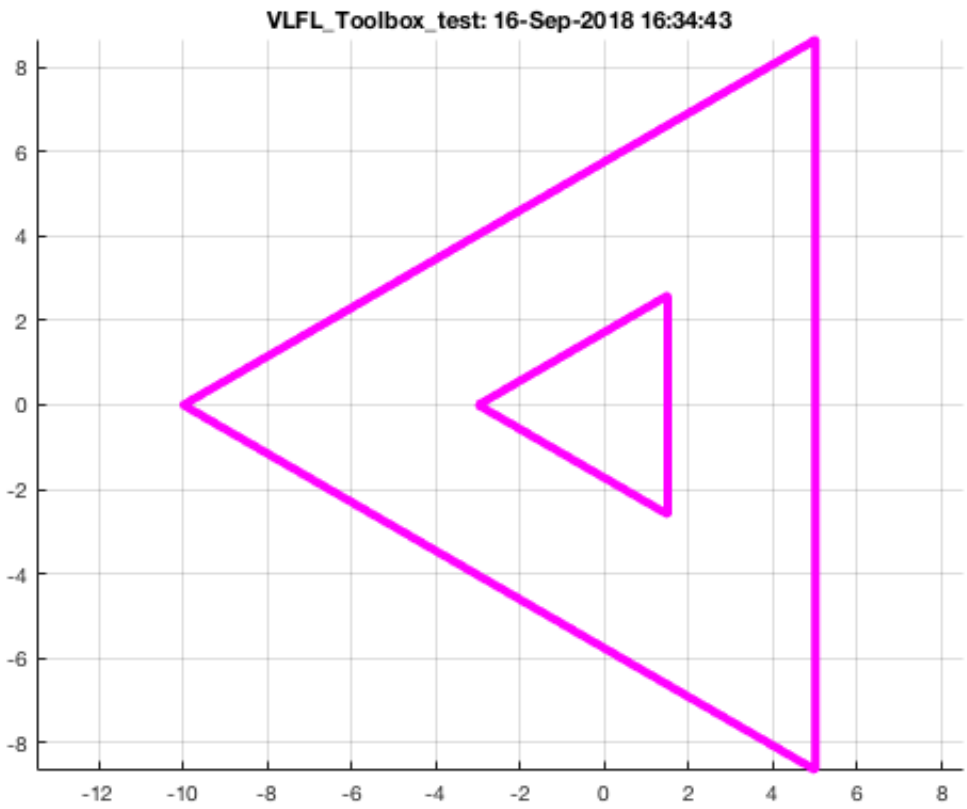
```
in=isInterior(TR2);
VLFLplots(TR2.Points, TR2.ConnectivityList(in,:),'c');
```



Often we want directly getting a closed contour of a slice.

```
CPLofSGslice(A,9); [CPL,warn]=CPLofSGslice(A,9); warn
```

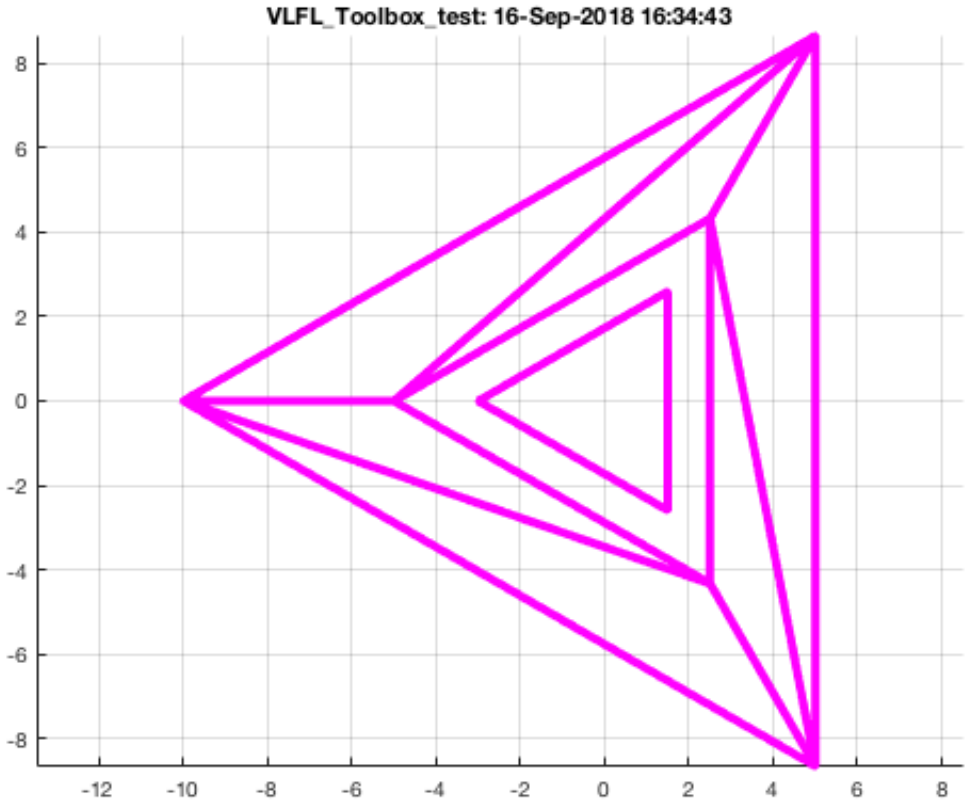
```
warn =

  logical

   0
```



The output parameter warns if a ambiguous cutting result exists

```
CPLofSGslice(A,10); [CPL,warn]=CPLofSGslice(A,10); warn
```

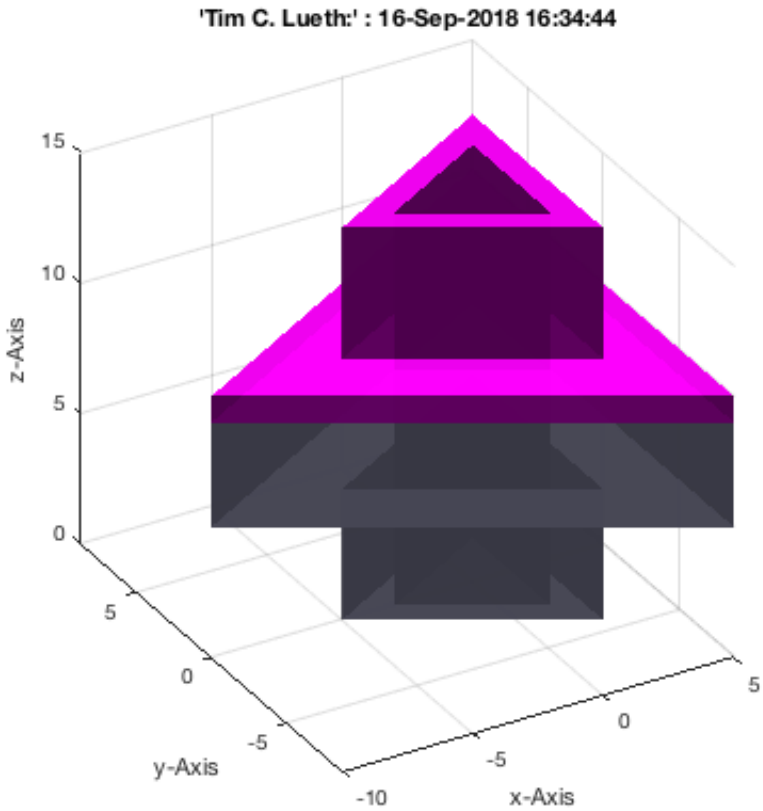
```
Warning: Crossing plane cannot be calculated error-free
Warning: Crossing plane cannot be calculated error-free

warn =

  logical

   1
```



4. Cutting and separating a solid geometries in two parts

By using the output of SGslicer it is possible to create an upper and lower part of an object or even by two cutting plane to cut a part out of a larger obect. This is done by the function **SGcut**.

```
VLFLfigure;
SGcut(A,9);
```

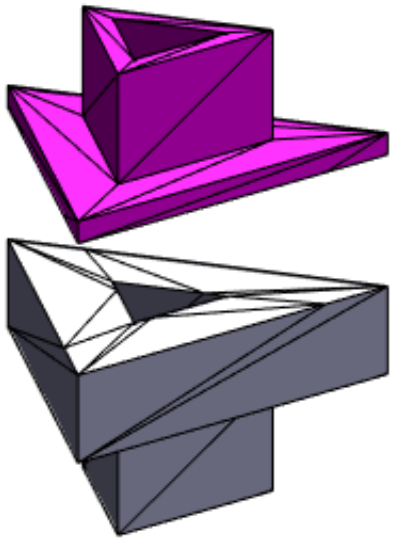


The next figure shows a separation of the two part by moving the upper part upwards.

```
[L,U]=SGcut(A,9)
VLFLfigure;
SGplot(SGtransP(L,[0 0 -3]),'w');
SGplot(SGtransP(U,[0 0 +3]),'m');
view (50,20);
```

```
L =

  struct with fields:

    VL: [27×3 double]
    FL: [54×3 double]


U =

  struct with fields:

    VL: [27×3 double]
    FL: [54×3 double]
```

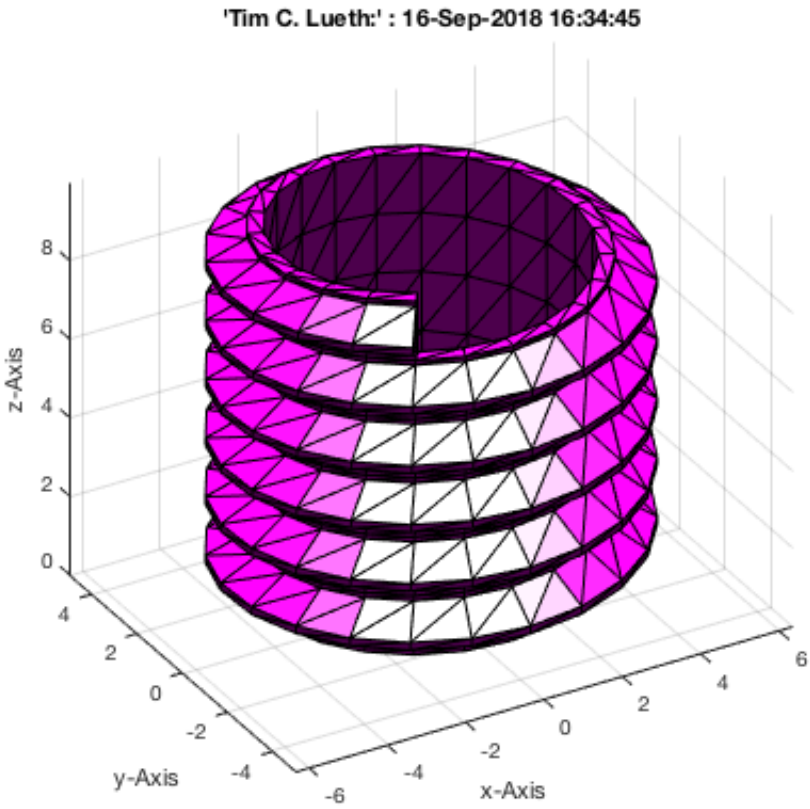
'Tim C. Lueth:' : 16-Sep-2018 16:34:44



5. Cutting as useful tool for the ending of complex shaped geoemtries

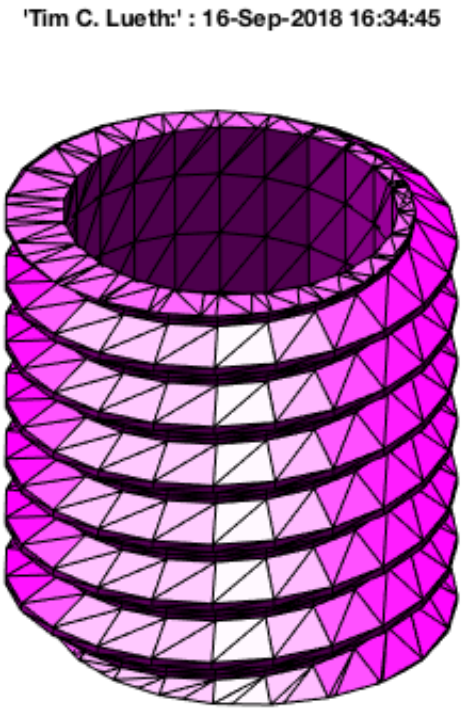
Some geometries such as screwnuts have specific geometries that have their origin in the manufacturing process of the threads. To create also similar shapes it is necessary to create a longer thread and to cut out the required length later:

```
VLFLfigure;
SGthread (10,10,[],[],'C'); view (-30,30);
% [A,b,c]=SGthread (10,10);
```



Now create a longer thread and cut out the required length later.

```
VLFLfigure;
A=SGthread (10,10+5+5,[],[],'C');
[~,B]=SGcut(A,[5.05 14.95]); B=SGtransP (B,[0 0 -5]);
SGplot(B,'m'); view (-30,30);
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:34:46!
Executed 16-Sep-2018 16:34:48 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-08*
- *Christina Friedrich, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-06-17*

Published with MATLAB® R2018a

Tutorial 09: Boolean Operations with Solid Geometries

2014-11-30: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.0 required)
- 3. Creating two solids for showing the boolean operations
- 4. Boolean operator: Substraction A-B or A\B
- 5. Boolean operator: Substraction A+B
- 6. Boolean operator: Substraction B\A
- 7. Boolean operator: A xor B
- 8. Analyzing the results and comparision with additive design.
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.0 required)

The implementation of the boolean operations based on STL geometries took the author several years. The reason was not only the complexity of the numerous special cases but also the numerical accuracy or resolution of the required geometrical calucations. So even if a normal position is cacluated with 12 digits accuracy, the cross product often has a just an accuracy of 6 digits. Unfortunatly crossing triangles with position errors of 6 digits can easily lead to phantom triangles, phantom edges wich either do not really exist or are just doubles of already existing lines. Since normally all edges must have a second edge with the opposite direction, doubled lines/egdes with the same direction make trouble. So to be successful with the boolean operations you should make sure that

1. No facet should be in the same plane as or overlap another facet or cross with almost parallel edges to the plane of another facet. This is always valid for one solid, but in case of a second solid for boolean operations, it is quite difficult to guarantee this.
2. It is fact that, the more boolean operations took place, to create a new solid, the more vertices and facets were created. The removement of dispensable vertices and facets is possible but is a boring non productive pieve of source code. So the motivation to programm such a procedure is not high.
3. No edge of a triangle should be in the sample plane or crossing but almost parallel to a plane of a facet.
4. It is fact that a normal user just want to use the boolean operator without thinking about those problems. The normal user will just be disappointed if the way to design a physical solid object finally fails because of the limitations of the crossing
5. Make definitly sure that after all boolean operations you use SGchecker to analyze the solid geoemtry do detect errors immedeatly.
6. May be the only solution is to use a fixed coordiate grid during all calculations to make shure that two vertices are either definitly separated or definitly the same. #

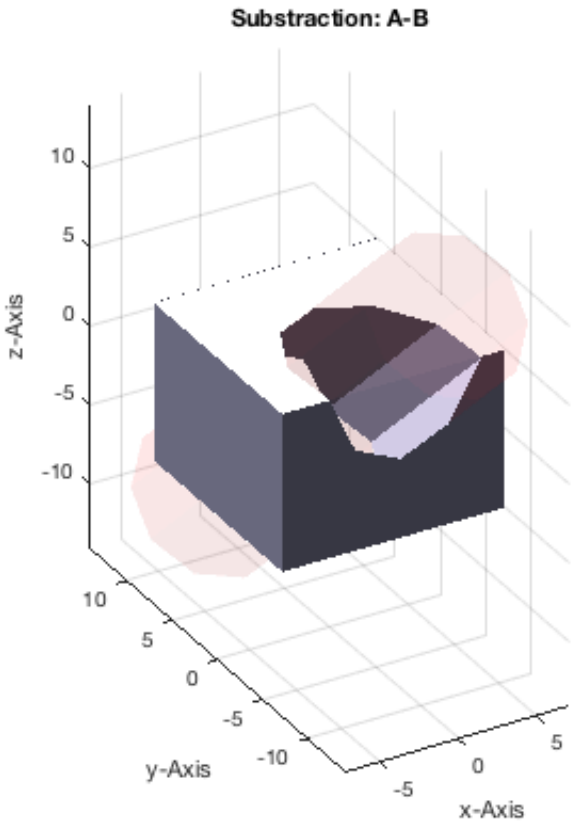
3. Creating two solids for showing the boolean operations

```
function VLFL_EXP09
```

```
VLFLfigure; view(-30,30); grid on;
```

```
A=SGofCPLz(PLcircle(10,4),10);  A=SGtrans0(A);
B=SGofCPLz(PLcircle(5,10),30);  B=SGtrans0(B); B=SGtransR(B,rotdeg(45,5,0));

SGplot(A,'b');
SGplot(B,'r');
VLFLplotlight (0,0.9);
A=SGstripfields(A)
B=SGstripfields(B)

SGbool ('-',A,B);
```

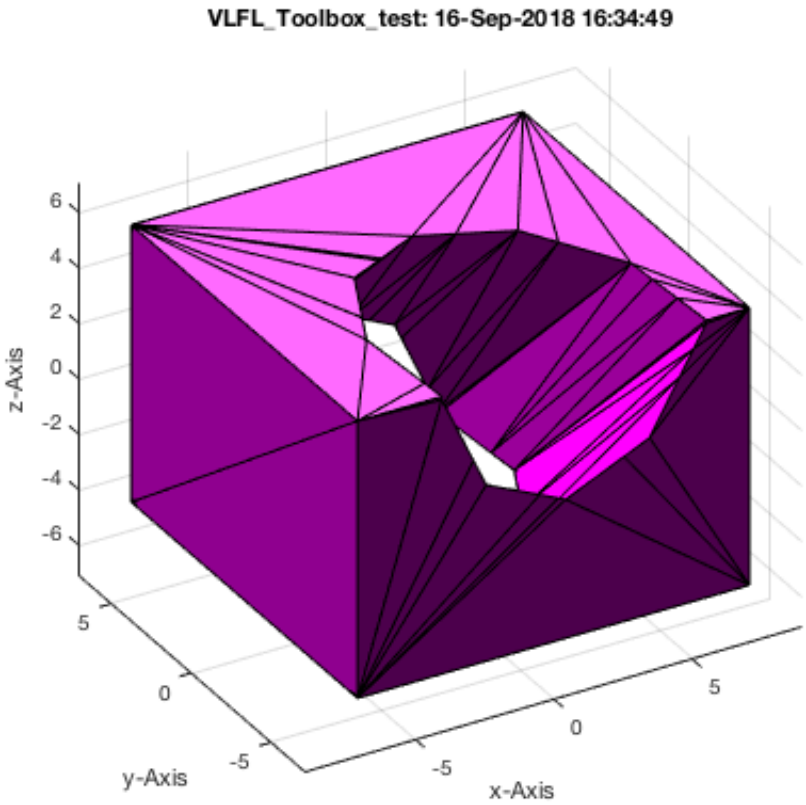
```
A =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]


B =

  struct with fields:

    VL: [20×3 double]
    FL: [36×3 double]

SGchecker "A-B":
```



4. Boolean operator: Substraction A-B or A\B

```
X=SGbool ('A',A,B);
SGfigure(X); view(-30,30);
```

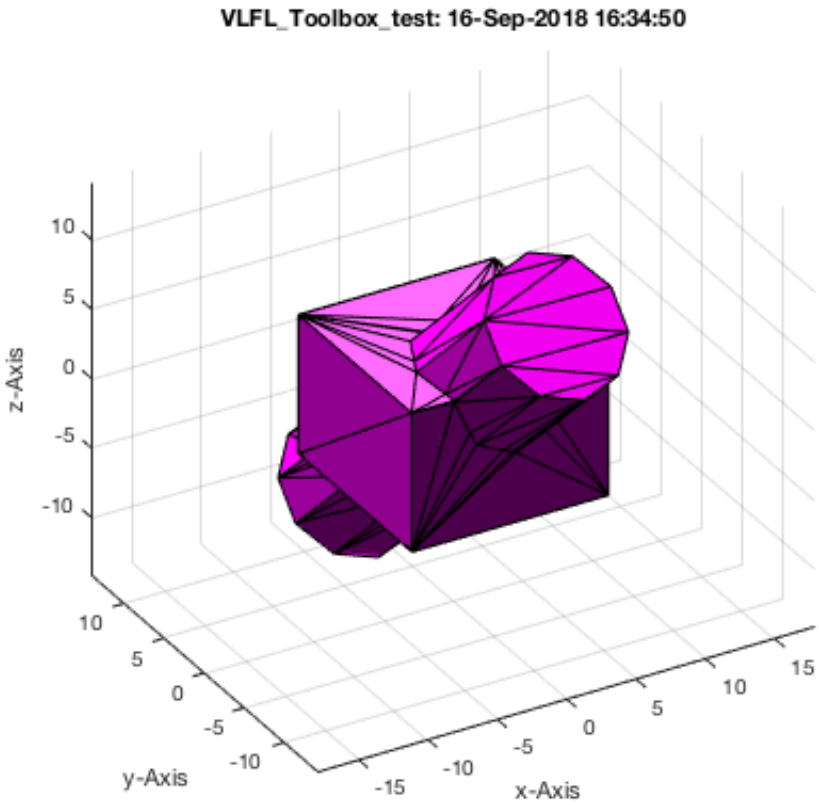
```
SGchecker "AAB":
```



5. Boolean operator: Substraction A+B

```
X=SGbool ('+',A,B);
SGfigure(X); view(-30,30);
```

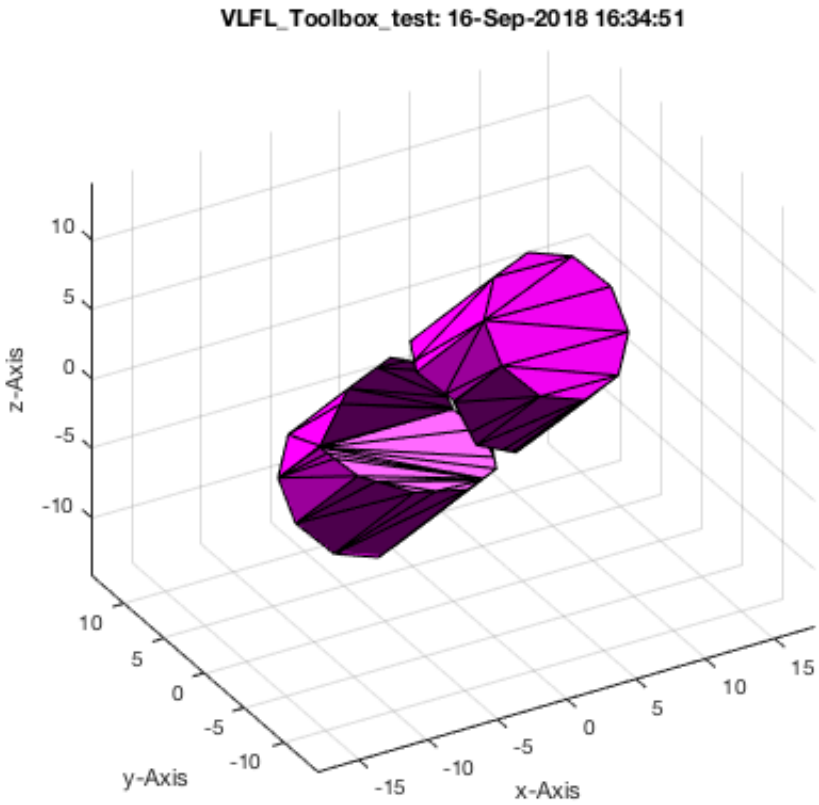
```
SGchecker "A+B":
```



6. Boolean operator: Substraction B\A

```
X=SGbool ('B',A,B);
SGfigure(X); view(-30,30);
```

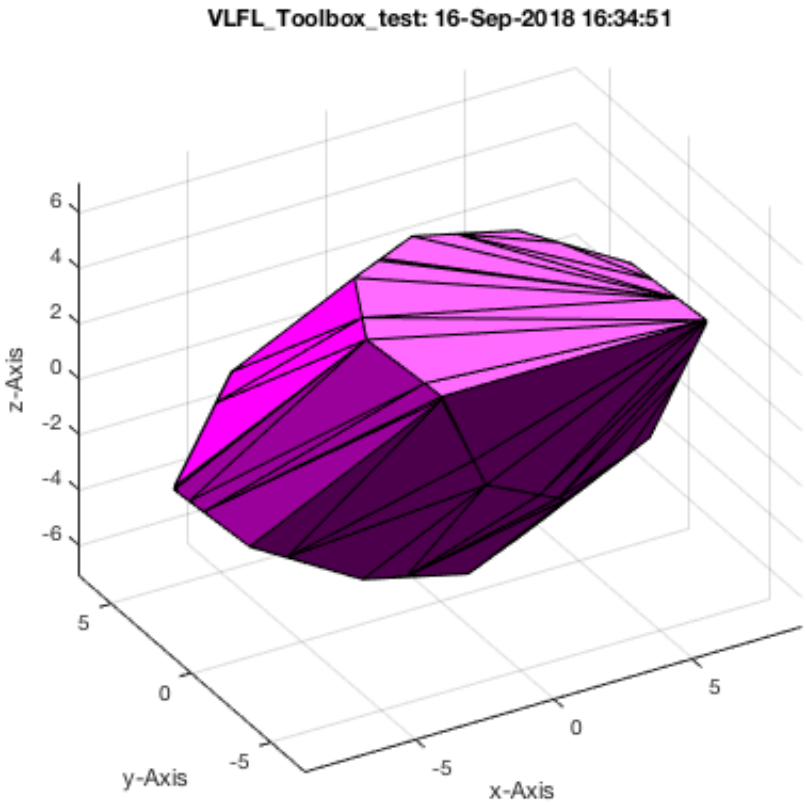
```
SGchecker "ABB":
```



7. Boolean operator: A xor B

```
X=SGbool ('x',A,B);
SGfigure(X); view(-30,30);
```

```
SGchecker "AxB":
```



8. Analyzing the results and comparision with additive design.

Analyzing the number of vertices and facets of the results of a boolean operation shows clearly that there are much more vertices and facets than the sum of the vertices and facets. In general it makes for STL more sense to add simple solids to a more complex by attaching them together by simply pushing them into another. In this case the final number of vertices and facets is the sum of the individual facets and vertices.

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:34:52!
Executed 16-Sep-2018 16:34:54 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-07*

- *Christina Friedrich, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-06-17*

Published with MATLAB® R2018a

Tutorial 10: Packaging of Sets of Solid Geometries (SG)

2015-08-06: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.4 required)
- 2. The four-bar-linkage kit as example for a set of multiple solids
- 3. Packaging a set of solid geometries in a volume
- 4. Using container/collections insted of itemizing the solids
- 5. Create boxes around the packed solids for the final 3D print job
- 6. Create the four-bar-linkage kit as print job
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

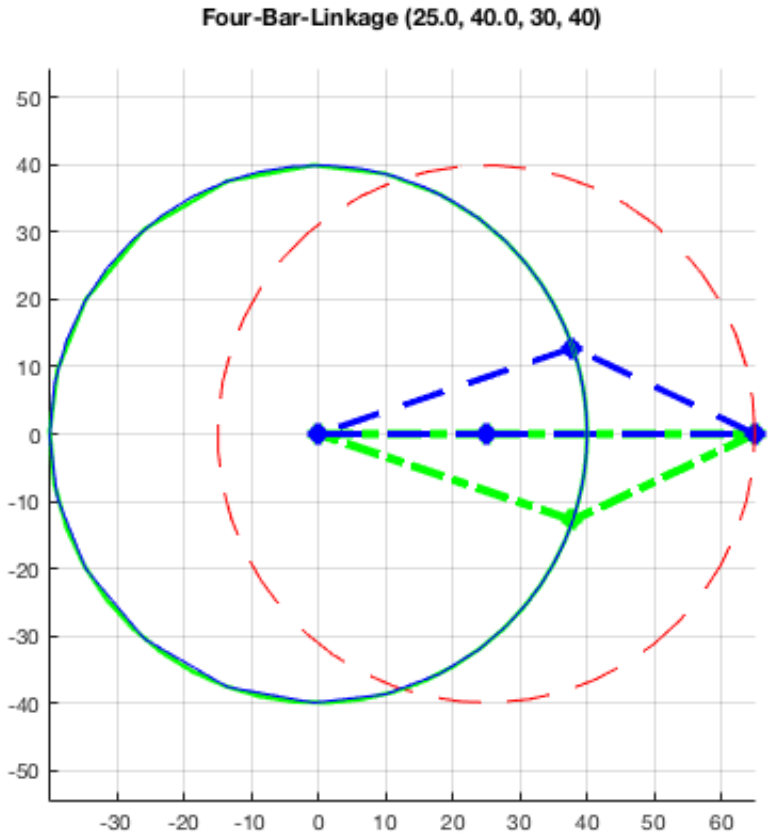
Motivation for this tutorial: (Originally SolidGeometry 2.4 required)

2. The four-bar-linkage kit as example for a set of multiple solids

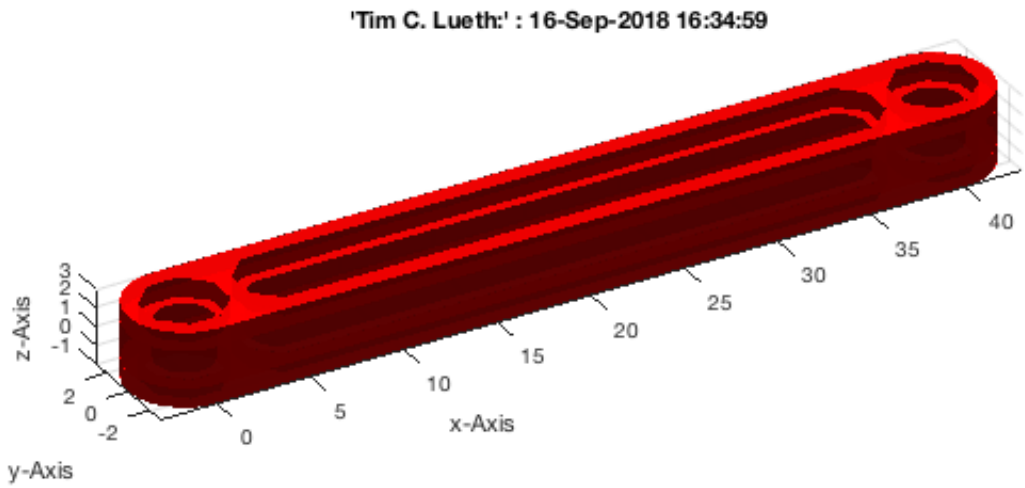
A very interesting mechanism in mechanics is the four-bar-linkage. It consists of four bars that are linked together by 4 rotatorial joints. Such a mechanism can be built by 4 different elememts

1. Bar: The basic mechanic link
2. Bolt: A simple bolt that allows rotation
3. Shaft: A simple shaft that transfer torque
4. Spacer: A simple element that is required to achieve parallel bars

```
close all;
fourBarLinkage (25,40,30,40);
```



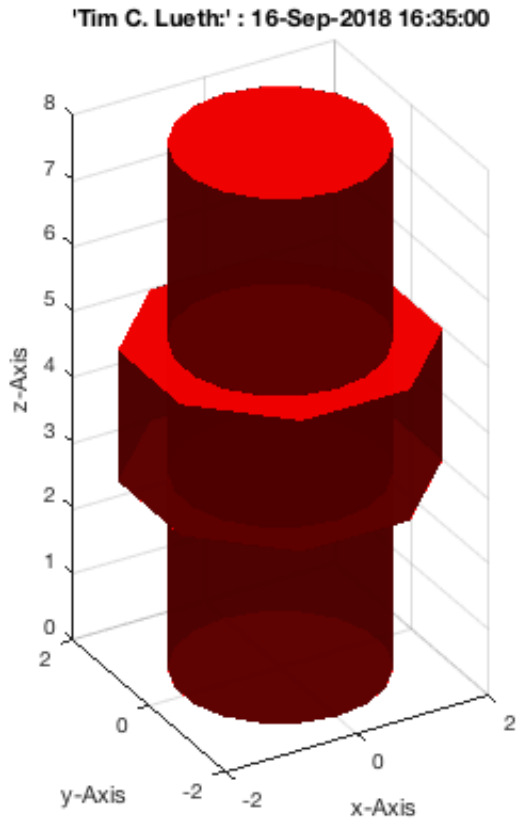

```
fourBarLinkageKit ('Bar',40);
```



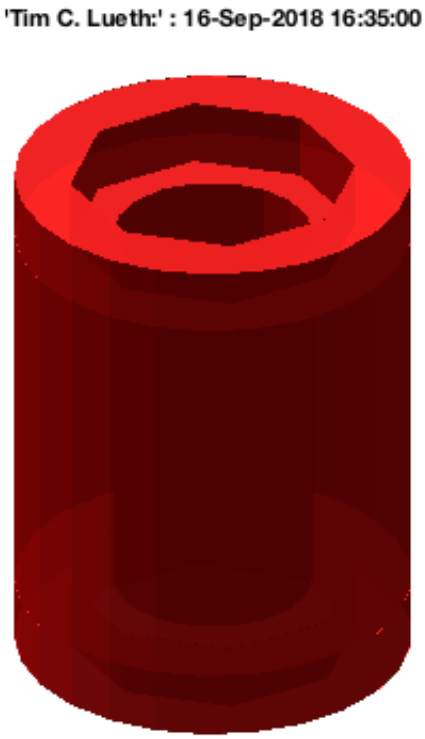

```
fourBarLinkageKit ('Bolt');
```

'Tim C. Lueth:' : 16-Sep-2018 16:34:59

```
fourBarLinkageKit ('Shaft');
```




```
fourBarLinkageKit ('Spacer');
```



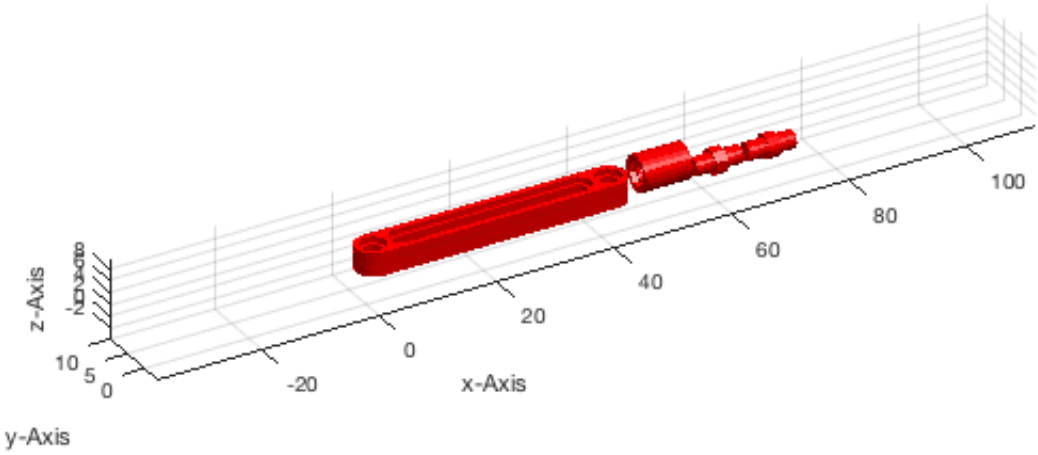
3. Packaging a set of solid geometries in a volume

For a four-bar-linkage we need 4 bars and 4 bolts and may be 2 spacer and 2 shafts. For this purpose there is one function

- **SGpacking** arranges several solid geometries side by side in a volume

```
close all;
A=fourBarLinkageKit ('Bar',40);
B=fourBarLinkageKit ('Bolt');
C=fourBarLinkageKit ('Shaft');
D=fourBarLinkageKit ('Spacer');
SG=SGpacking({A,B,C,D});
SGplot(SG); view (-30,30); VLFLplotlight (1,0.9); zoompatch;
```

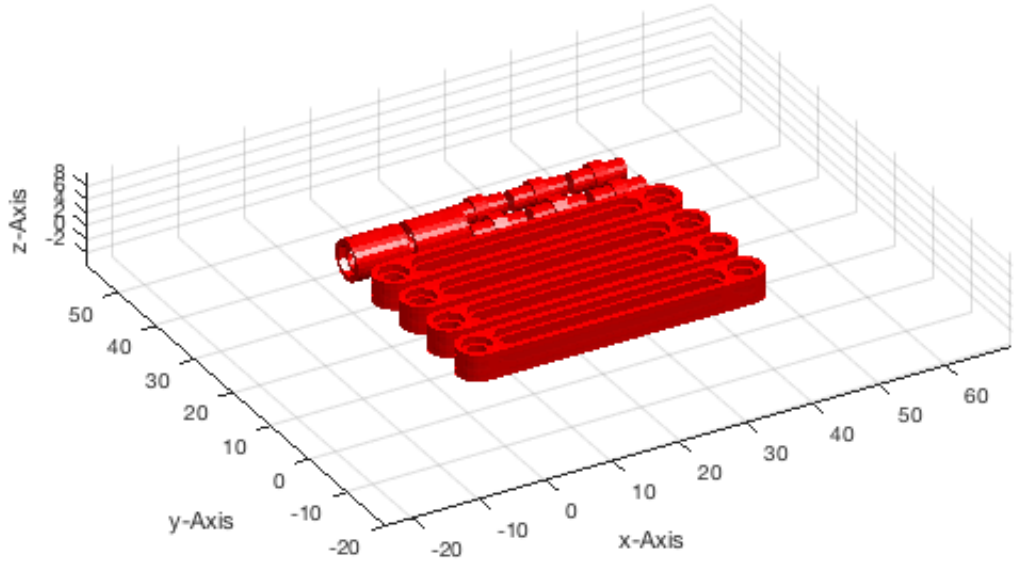
```
Packing 4 objects (h=24):
```



Similar it is possible to pack several objects of the same kind into the volume and also to define the dimensions of the packing volume. Typically the z-coordiante of the volume specification is unlimited or much bigger than the xy-coordinats.

```
close all;
SG=SGpacking({A,A,A,A,B,B,B,B,C,C,D,D},[50,50,1000]);
SGplot(SG); view (-30,30); VLFLplotlight (1,0.9); zoompatch;
```

```
Packing 12 objects (h=45):
```



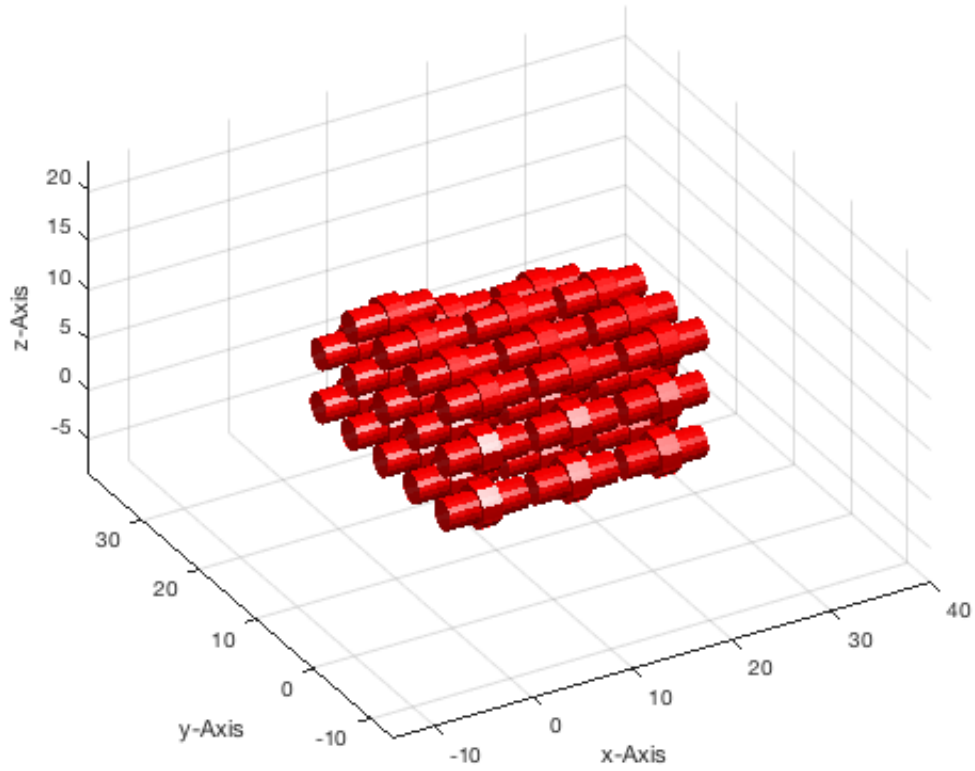
4. Using container/collections insted of itemizing the solids

In many cases we are not interested to list the items in the source code but to create a structure containing all objects we want to pack later Therefor, we need a data structure that allows to collect several solids into something like a container. This can be done by the following functions:

- **SGCaddSG** Add a single solid geometry to a collection
- **SGCaddSGn** Add multiple copies of a single solid geometry to a collection

```
close all
SGC=[];                                                 % Create a Solid Geometry Collection
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Bolt'),20);    % Add 20 bolt to the container
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Shaft'),20);   % Add 20 shafts to the container

SG=SGpacking(SGC,[30, 30 ,1000]);                    % SGpacking accepts also SGC structs
SGplot(SG); view (-30,30); VLFLplotlight (1,0.9); zoompatch;
```

```
Packing 40 objects (h=48): ....................
```

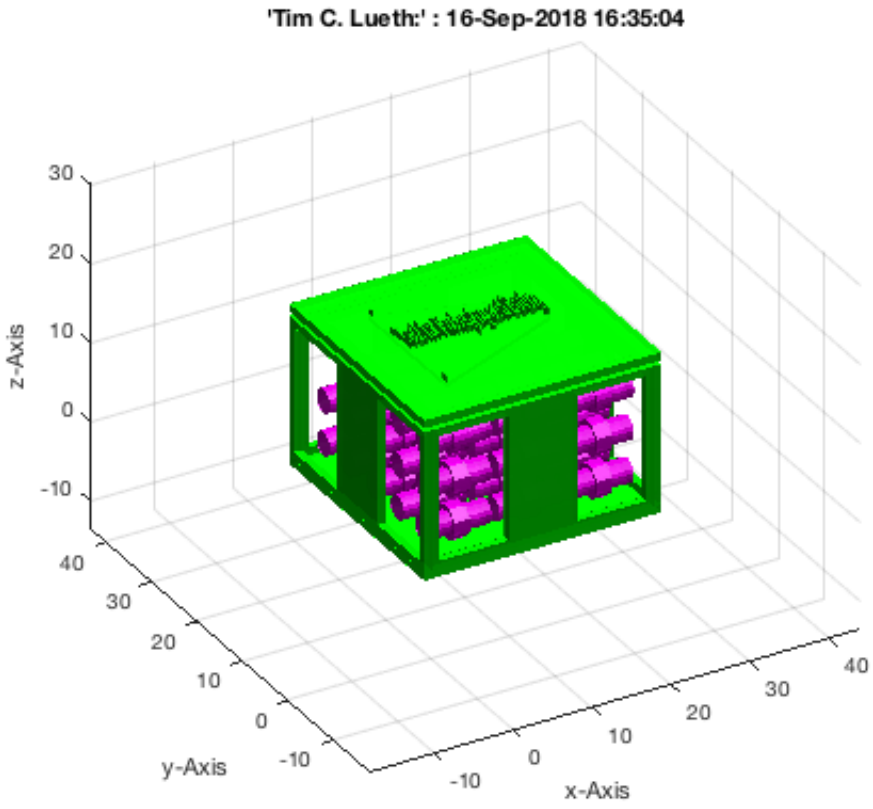


5. Create boxes around the packed solids for the final 3D print job

To handle the print job in a convinient way, it makes sense to create a box around the parts and also to write on top of the cover the content or the intended use of the box plus may by a date.

```
close all;
SGboxing(SG,[],[],'.\nTest for Packaging and Boxing\n.');
view (-30,30); VLFLplotlight (1,0.9); zoompatch;
```

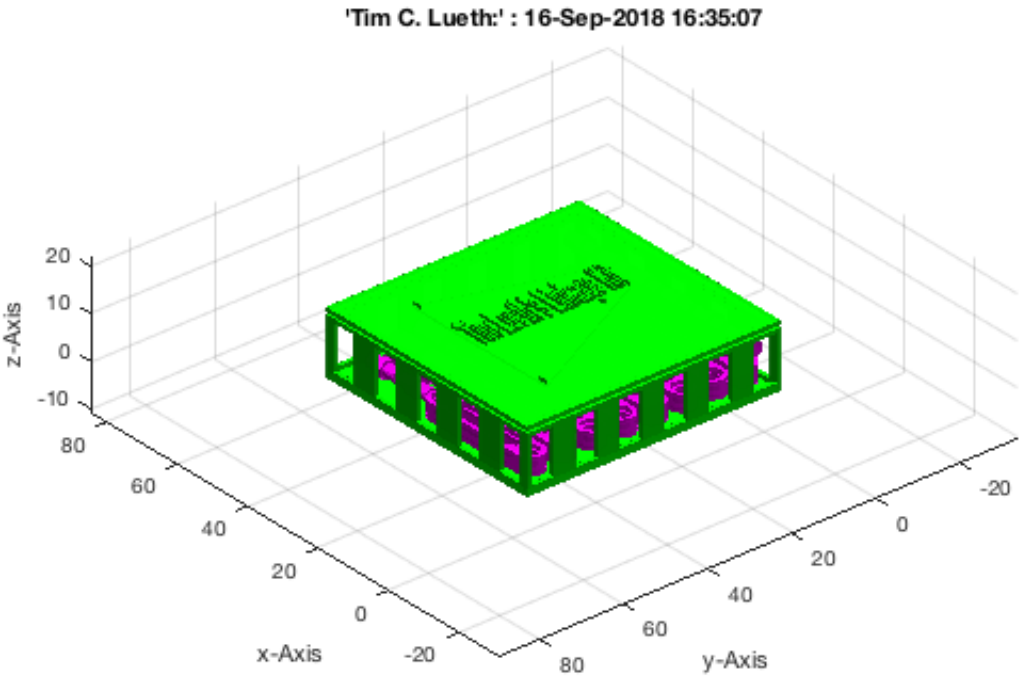
```
==>TEXT GENERATION..Analyze union areas for 60 facets:
Major union vectors: 6 found with maximum size of 1982.
Finally 2 union areas found with size > 100
Text attached to union Nr: 2
..finished!
```



6. Create the four-bar-linkage kit as print job

```
close all;
SGC=[];
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Bar',25),2);
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Bar',35),2);
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Bar',40),4);
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Bolt'),4);
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Shaft'),4);
SGC=SGCaddSGn(SGC,fourBarLinkageKit ('Spacer'),4);
SGA=SGpacking(SGC,[55, 60 ,100]);
SGB=SGboxing(SGA,[],[],'.\nTim Lueth''s Linkage Kit\n.');
VLFLfigure(SGA); SGplot(SGB,'g');
SG=SGcat(SGA,SGB); view (-130,30); VLFLplotlight (1,0.9); zoompatch;
SGwriteSTL(SG,'EXP10: Four-Bar-Linkage-Kit');
```

```
Packing 20 objects (h=58): ....................
==>TEXT GENERATION..Analyze union areas for 60 facets:
4 Dimension warnings
Major union vectors: 6 found with maximum size of 7323.
Finally 22 union areas found with size > 100
Text attached to union Nr: 2
..finished!
1000..2000..3000..4000..5000..6000..7000..8000..9000..10000..11000..12000..13000..
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:35:07!
Executed 16-Sep-2018 16:35:09 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-08*
- *Christina Friedrich, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-06-17*

Published with MATLAB® R2018a

Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models

2015-06-08: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.3 required)
- 2. Loading the 5 components of a 4DoF robot solid model
- 3. The concept of attaching coordinate frames as 4x4 honogenous transformation matrix
- 4. Attaching manually coordinate frames as 4x4 honogenous transformation matrix
- 5. Creating kinematic models consisting of named solids
- 6. Automatic creation of a chain
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.3 required)

2. Loading the 5 components of a 4DoF robot solid model

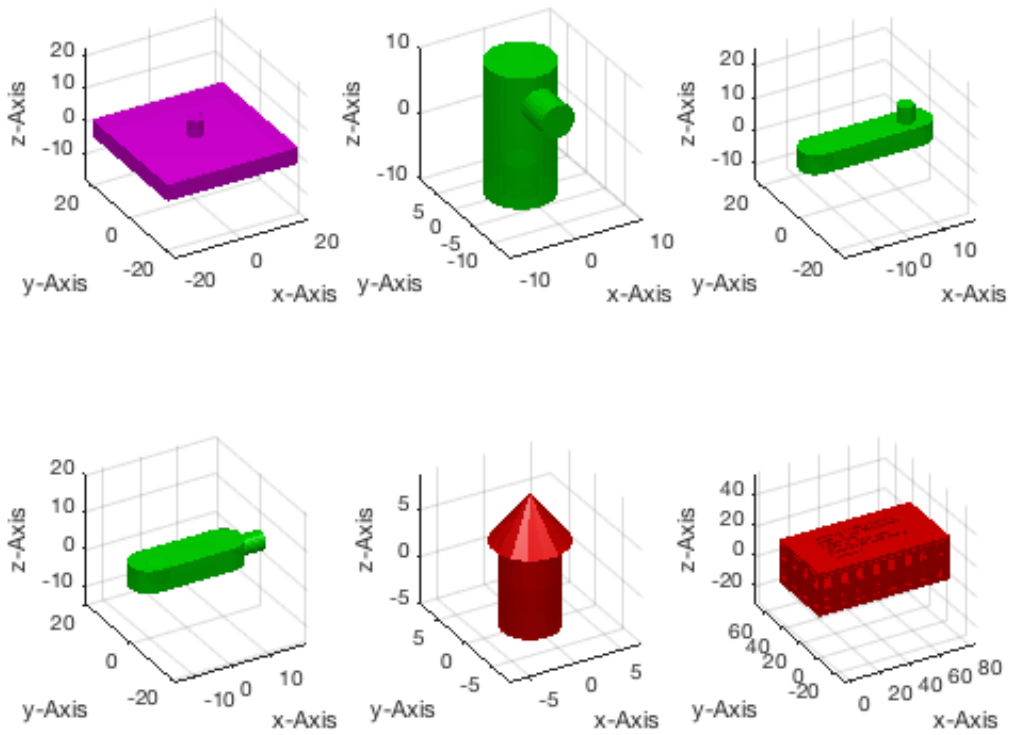
Before explaining how to create the parts of a robot kinematik we just load such components in. The command line load AIM_robot

```
load AIM_SGrobot
SG0=SGfixerrors(SG0,1e-3); SGchecker(SG0);
SG1=SGfixerrors(SG1,1e-3); SGchecker(SG1);
SG2=SGfixerrors(SG2,1e-3); SGchecker(SG2);
SG3=SGfixerrors(SG3,1e-3); SGchecker(SG3);
SG4=SGfixerrors(SG4,1e-3); SGchecker(SG4);
save ('AIM_SGrobot','SG0','SG1','SG2','SG3','SG4','SGrobot');
```

- returns a solid geometry SG0 which is a base plate with a rotatorial joint
- returns a solid geometry SG1 which is a link with a rotatorial joint
- returns a solid geometry SG2 which is a link with a rotatorial joint
- returns a solid geometry SG3 which is a link with a rotatorial joint
- returns a solid geometry SG4 which is a hand with a pointing tip
- returns a solid geometry SGrobot which can be written as STL-File an printed using a 3D printer.

```
VLFLfigure;
subplot(2,3,1); SGplot(SG0); view (-30,30); VLFLplotlight(1,0.9);
subplot(2,3,2); SGplot(SG1); view (-30,30); VLFLplotlight(1,0.9);
subplot(2,3,3); SGplot(SG2); view (-30,30); VLFLplotlight(1,0.9);
subplot(2,3,4); SGplot(SG3); view (-30,30); VLFLplotlight(1,0.9);
subplot(2,3,5); SGplot(SG4); view (-30,30); VLFLplotlight(1,0.9);
subplot(2,3,6); SGplot(SGrobot); view (-30,30); VLFLplotlight(1,0.9);
SGwriteSTL (SGrobot,'4-DOF Robot Set');
```

```
1000..2000..3000..4000..5000..6000..7000..8000..9000..10000..11000..12000..13000..14000..15
000..16000..17000..18000..19000..20000..21000..22000..23000..
```



3. The concept of attaching coordinate frames as 4x4 honogenous transformation matrix

If we analyze the structure of one of the components of the robot we see that we have now more than just the surface of the geometry.

```
SG0

% We see that beside vertices and facets (VL, FL) we have a color and a
% alpha value for transparency when plotting.
```

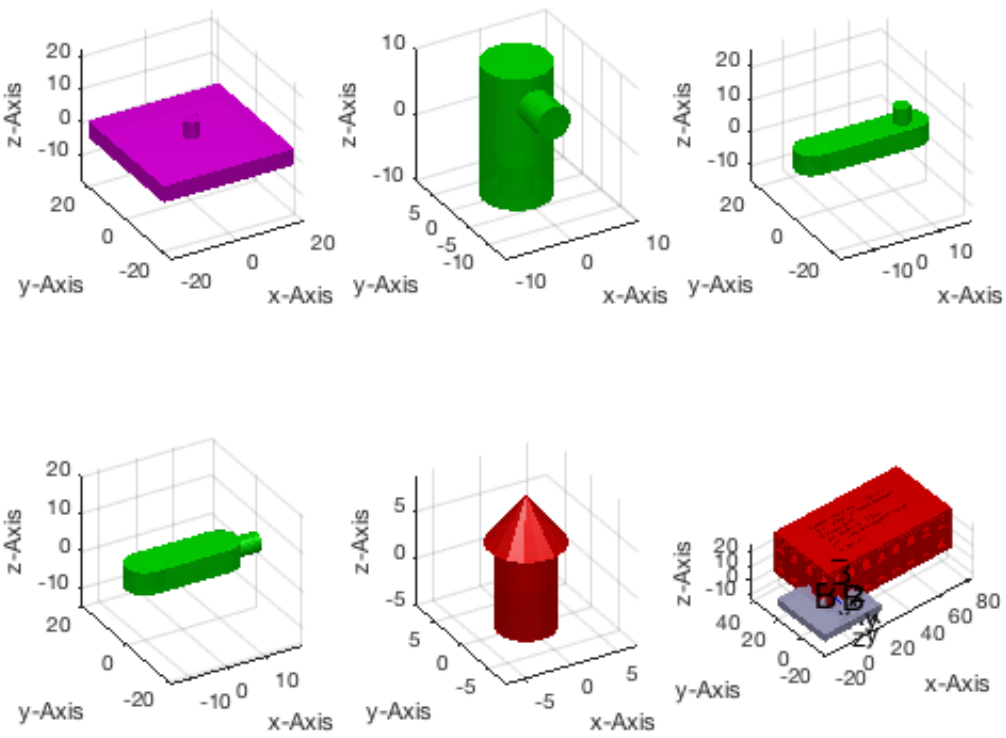
```
SG0 =

  struct with fields:

      VL: [79×3 double]
      FL: [154×3 double]
   alpha: 0.8000
   color: 'm'
   Tname: {'A'  'B'}
       T: {[4×4 double]  [4×4 double]}
    TFiL: {[2×1 double]  [21×1 double]}
```

- *Tname* is a cell list contain the ascii-string of the names of the coordinate frames
- *T* is the 4x4 homogenous transformation matrix related to the indexed name
- *TFiL* is an optional index of the facets that belong to the surface that defines the coordinate system To the homogenous transformation matrix out of the struct, the most convinient way is to use the function:

- **SGT** returns for a given solid and a given frame name the 4x4 matrix
- **SGT** draws the part and the frames and the defining facets if there is no output parameter

```
A=SGT(SG0,'A')
B=SGT(SG0,'B')

SGT(SG0);
view(-40,40);
```

```
A =

    1.0000         0         0         0
         0   -1.0000         0         0
    0.0000         0   -1.0000   -2.5000
         0         0         0    1.0000


B =

   -1.0000         0         0    0.0120
         0   -1.0000         0         0
         0         0    1.0000    7.5000
         0         0         0    1.0000
```



4. Attaching manually coordinate frames as 4x4 honogenous transformation matrix

Since there is no requirement to use the facets TFiL, T matrices and their name can easily added by a programm during the design phase. Nevertheless, there is also a need to add frames interactively. For that purpose there are two other functions to add or to remove frames.

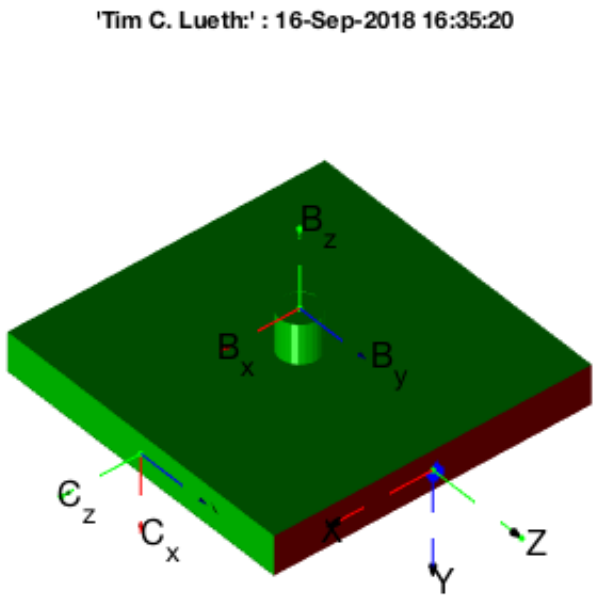
- **SGTremove** removes a named frame from the structure
- **SGTui** opens a figure and allows to generate a frame by touching a surface or point

To use SGTui you should a) first rotate the part on the screen until you see the surface where you like to set a frame, b) press enter and c) klick on the plane to set the frame. Now set two Frames 'C' and 'D'

```
A=SGTui(SG0,'C')
A=SGTui(A,'D')
view(-40,40);
```

```
A =

  struct with fields:

      VL: [79×3 double]
      FL: [154×3 double]
   alpha: 0.8000
   color: 'm'
   Tname: {'A'  'B'  'C'}
       T: {[4×4 double]  [4×4 double]  [4×4 double]}
```

```
      TFiL: {[2×1 double]  [21×1 double]  [2×1 double]}


A =

  struct with fields:

       VL: [79×3 double]
       FL: [154×3 double]
    alpha: 0.8000
    color: 'm'
    Tname: {'A'  'B'  'C'  'D'}
        T: {[4×4 double]  [4×4 double]  [4×4 double]  [4×4 double]}
     TFiL: {[2×1 double]  [21×1 double]  [2×1 double]  [2×1 double]}
```

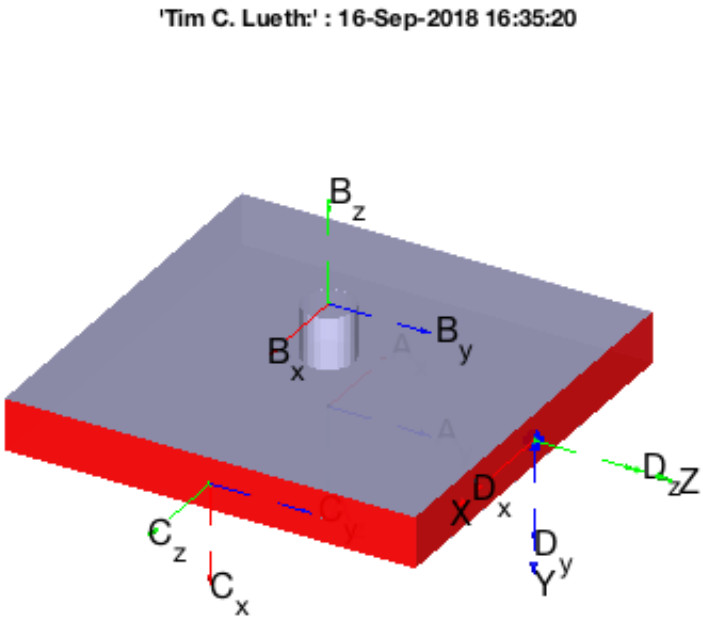


No we remove the first two frames 'A' and 'B'

```
A=SGTremove(A,'A')
A=SGTremove(A,'B')
SGT(A); view(-60,30);
```

```
A =

  struct with fields:

      VL: [79×3 double]
      FL: [154×3 double]
   alpha: 0.8000
   color: 'm'
   Tname: {'B'  'C'  'D'}
       T: {[4×4 double]  [4×4 double]  [4×4 double]}
    TFiL: {[21×1 double]  [2×1 double]  [2×1 double]}


A =

  struct with fields:

      VL: [79×3 double]
      FL: [154×3 double]
   alpha: 0.8000
   color: 'm'
```

```
Tname: {'C'  'D'}
    T: {[4×4 double]  [4×4 double]}
 TFiL: {[2×1 double]  [2×1 double]}
```

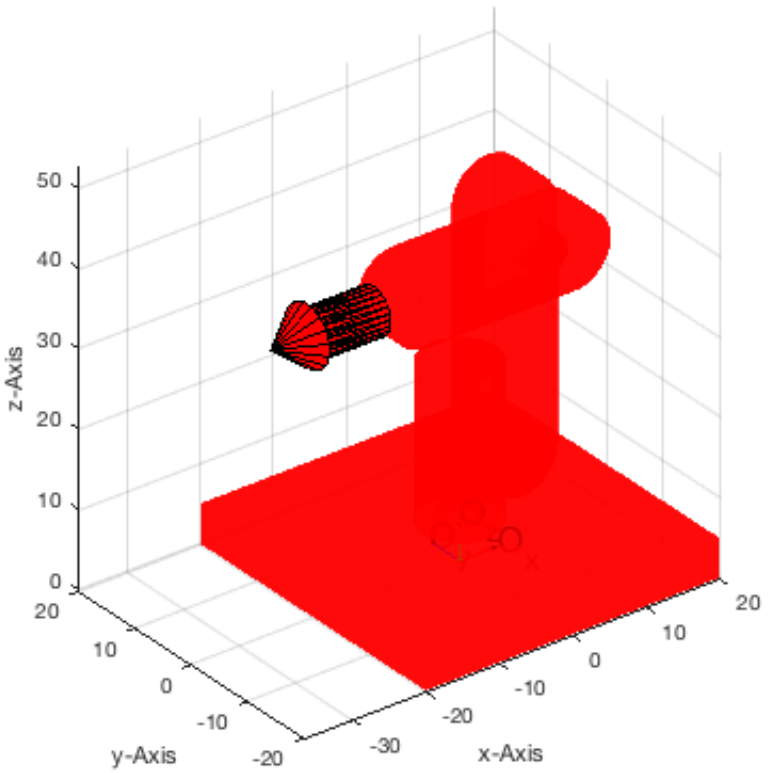


5. Creating kinematic models consisting of named solids

After being able to attach coodinate systems by frames to a solid, we can chain these solids by a string that describes which frames of the indiviudal objects are linked together. For this purpose we define a structure **KM (kinematik model)** that is a list of solids, followed by an ascii identifier and a transformation matrix for the origin of the solid. If the solids are chained, a function **KMchain** calculates those 3rd column transformation matrix to move and rotate the solid so that is fits to the given description of linked frames. **KMplot** shows the position of the individual solids in space.

```
% KM={SG0,'A',eye(4);SG1,'B',eye(4);SG2,'C',eye(4);SG3,'D',eye(4);SG4,'E',eye(4)}

KM.SG={SG0,SG1,SG2,SG3,SG4};
KM.Sname={'A','B','C','D','E'};
KM.BT={eye(4),eye(4),eye(4),eye(4),eye(4)};


KMchain(KM,'A.A-A.B-B.A-B.B-C.A-C.B-D.A-D.B-E.A-E.B-');
KM=KMchain(KM,'A.A-A.B-B.A-B.B-C.A-C.B-D.A-D.B-E.A-E.B-')
KMplot(KM);

% Now let us see how the 3rd column matrices describe the position of the
% solids in 3D space to create the robot model
KM.BT{:}
```

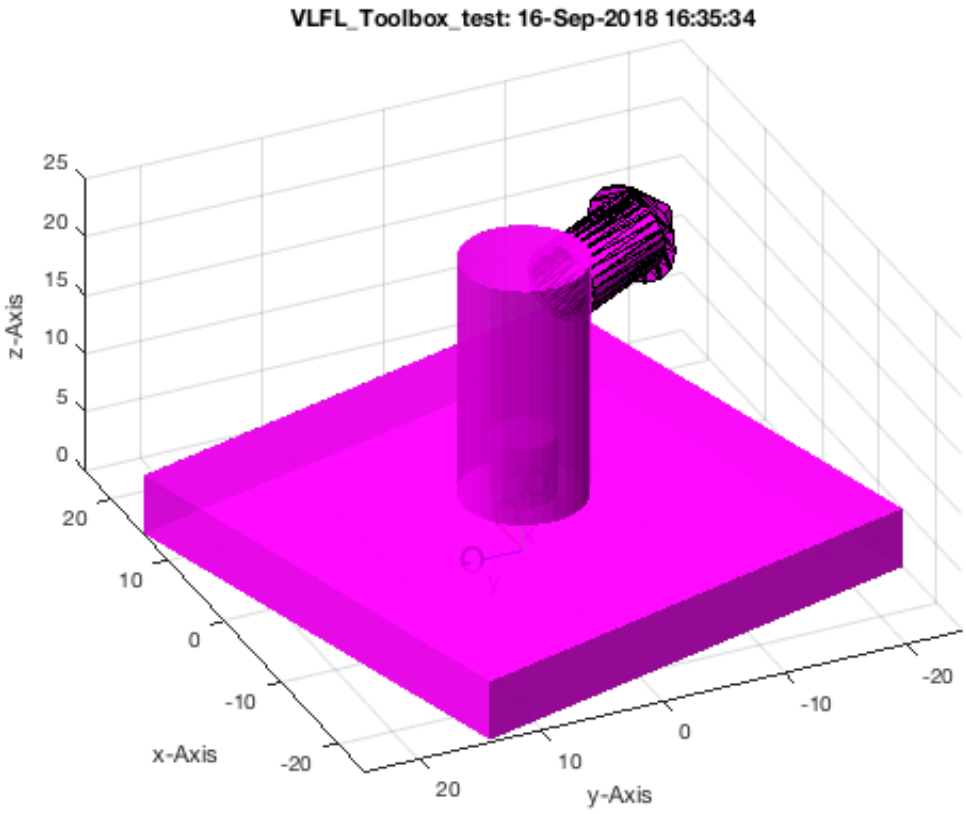
```
KM =

  struct with fields:

       SG: {1×5 cell}
    Sname: {'A'  'B'  'C'  'D'  'E'}
       BT: {1×5 cell}


ans =

   -0.0000    1.0000   -0.0000   -0.0000
   -1.0000   -0.0000         0         0
   -0.0000    0.0000    1.0000    2.5000
         0         0         0    1.0000


ans =

    1.0000    0.0000   -0.0000   -0.0120
   -0.0000    1.0000   -0.0000   -0.0120
         0    0.0000    1.0000   15.0000
         0         0         0    1.0000


ans =

   -0.0000   -1.0000    0.0000   -0.0000
    0.0000   -0.0000   -1.0000   -4.9880
    1.0000    0.0000   -0.0000   32.9590
         0         0         0    1.0000


ans =

   -1.0000         0    0.0000   -7.4530
    0.0000   -0.0000   -1.0000  -10.9880
    0.0000   -1.0000   -0.0000   45.4350
         0         0         0    1.0000


ans =

    0.0000   -0.0000   -1.0000  -27.4290
   -0.0000   -1.0000    0.0000  -13.9760
   -1.0000   -0.0000    0.0000   45.4470
         0         0         0    1.0000
```



6. Automatic creation of a chain

```
KMofSGs({SG0,SG1,SG4})
```

```
KMofSGs: No collisions found for tolerance: 0.10

ans =

  struct with fields:

       SG: {[1×1 struct]  [1×1 struct]  [1×1 struct]}
    Sname: {3×1 cell}
       BT: {3×1 cell}
       KC: {'A.A-A.B-B.A-B.B-C.A-C.B-'}
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:35:35!
Executed 16-Sep-2018 16:35:37 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==================================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-08*
- *Christina Friedrich, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-06-17*

Published with MATLAB® R2018a

Tutorial 12: Define Robot Kinematics and Detect Collisions

2015-08-09: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.4 required)
- 2. Loading the 5 components of a 4DoF robot solid model as last time
- 3. Automatic creation of a the robot
- 3. Collision in the joint by the resolution of the surfaces
- 4. Showing a different robot
- 5. Showing a self collision of a robot
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

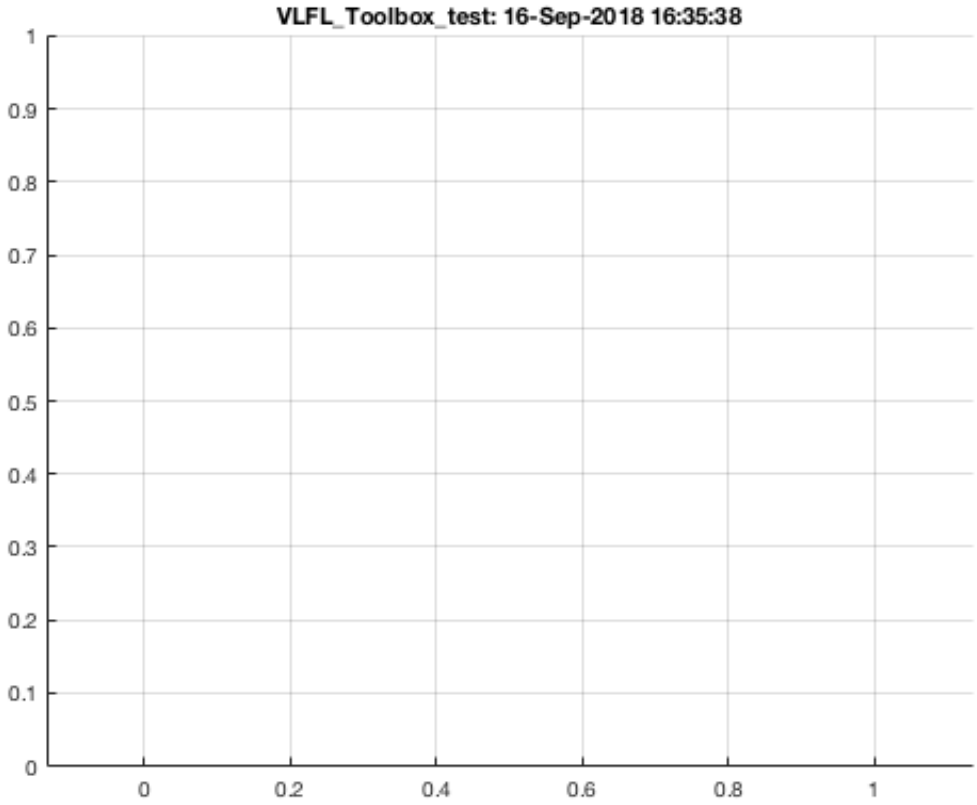
Motivation for this tutorial: (Originally SolidGeometry 2.4 required)

2. Loading the 5 components of a 4DoF robot solid model as last time

Before explaining how to create the parts of a robot kinematik we just load such components in. The command line load AIM_robot

```
function VLFL_EXP12
```

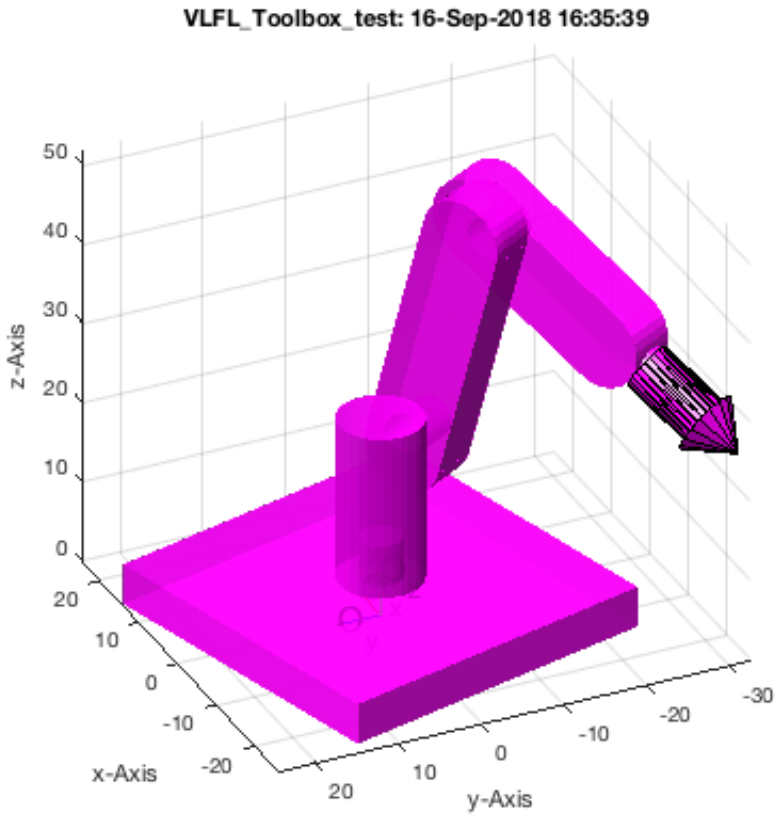
```
close all; SGfigure;
load ('AIM_SGrobot')
SG0=SGfixerrors(SG0,1e-3); SGchecker(SG0);
SG1=SGfixerrors(SG1,1e-3); SGchecker(SG1);
SG2=SGfixerrors(SG2,1e-3); SGchecker(SG2);
SG3=SGfixerrors(SG3,1e-3); SGchecker(SG3);
SG4=SGfixerrors(SG4,1e-3); SGchecker(SG4);
% save ('AIM_SGrobot','SG0','SG1','SG2','SG3','SG4','SGrobot');
VLFLplotlight(1,0.8);
```



3. Automatic creation of a the robot

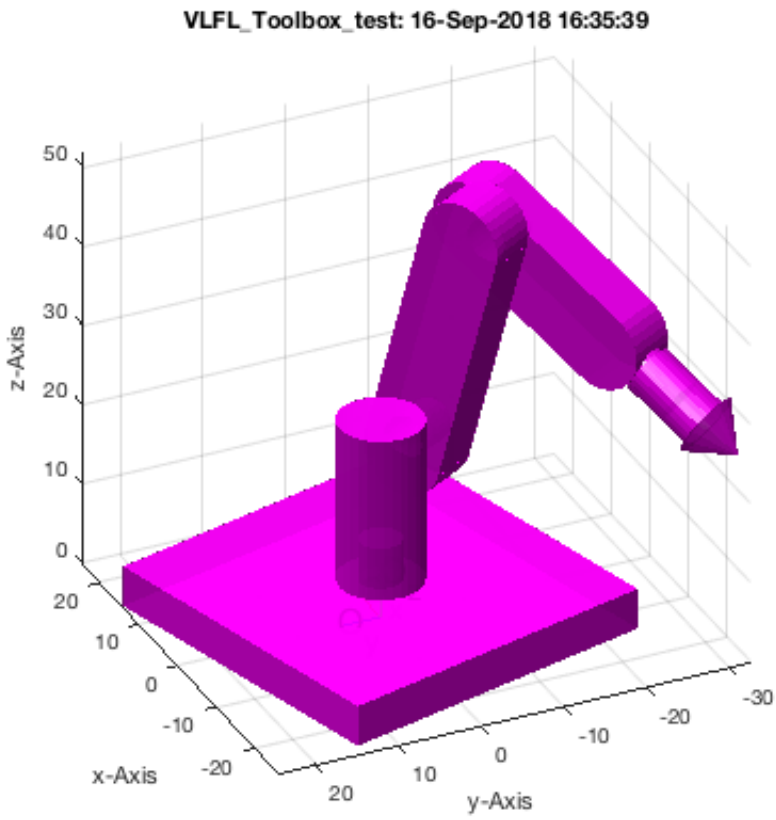
```
KMofSGs({SG0,SG1,SG2,SG3,SG4});
```

```
KMofSGs: No collisions found for tolerance: 0.10
```



3. Collision in the joint by the resolution of the surfaces

```
[KM,XVL]=KMofSGs({SG0,SG1,SG2,SG3,SG4},[],0.05);
KMplot(KM,'m'); VLFLplotlight (1,0.9);
```

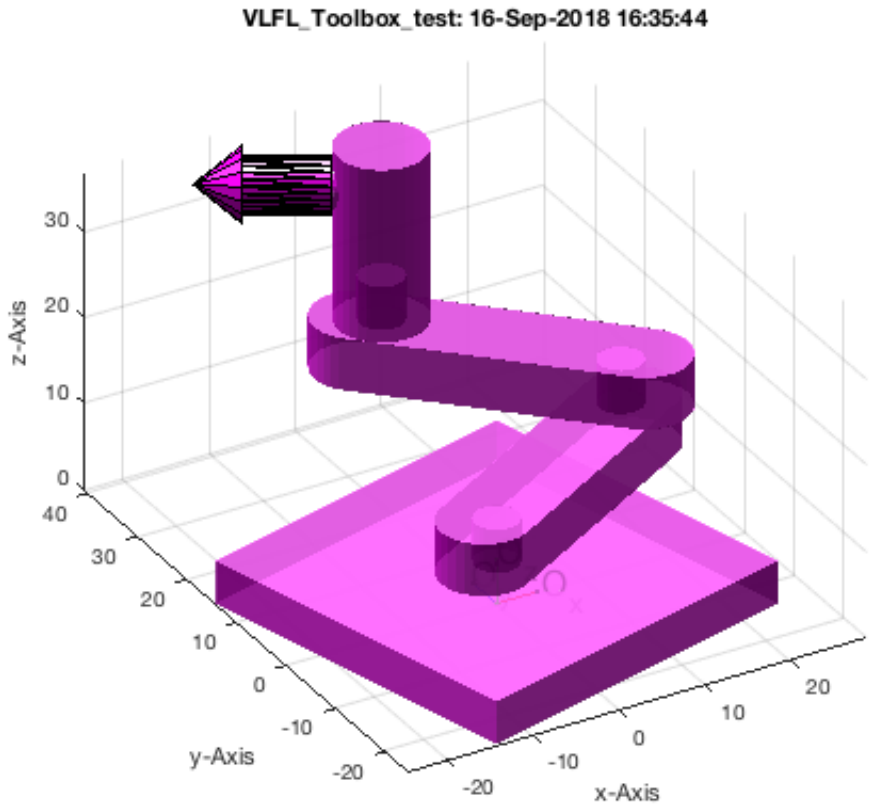



```
if ~isempty(XVL); zoompatch(XVL); VLplot(XVL,'k*',10);  end;
```

4. Showing a different robot

```
KMofSGs({SG0,SG2,SG2,SG1,SG4});
view(-30,30);
```

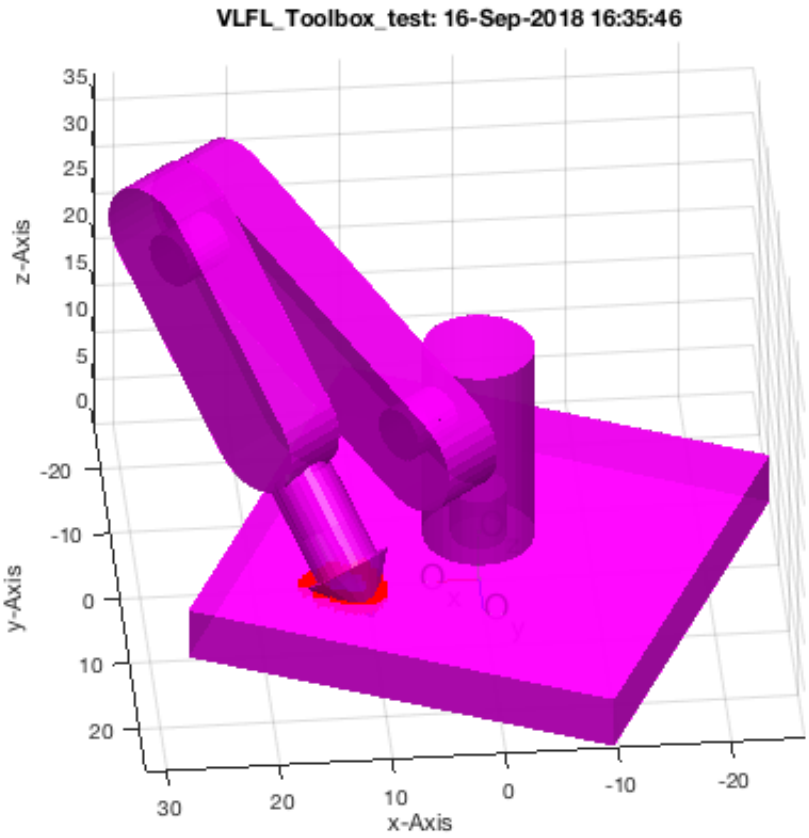
```
KMofSGs: No collisions found for tolerance: 0.10
```



5. Showing a self collision of a robot

```
KMofSGs({SG0,SG1,SG2,SG3,SG4},155);
view(-185,35); VLFLplotlight(1,0.8);
```

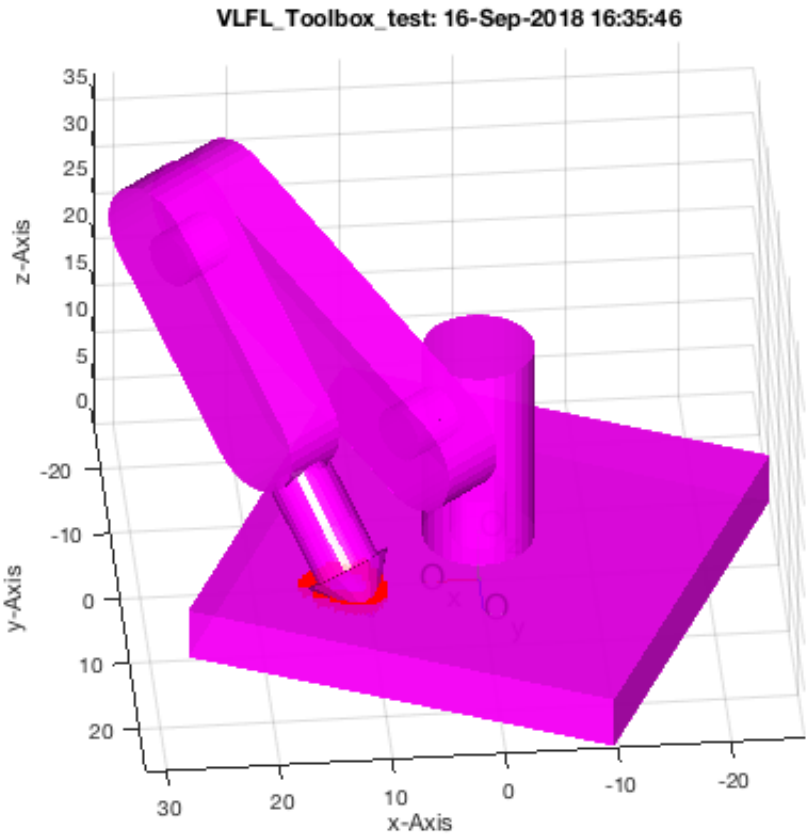
```
Warning in KMofSGs: 110 collisions found for tolerance: 0.10
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:35:47!
Executed 16-Sep-2018 16:35:49 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
=================================================================================================
==========
```



- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-08*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures

2015-09-11: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.5.1 required)
- 2. Analyzing mouting faces of flat surfaces
- 3. Analyzing mouting faces of spherical/freeform surfaces
- 4. Create corresponding surfaces parallel to mounting faces
- 5. Create solids using the parallel surfaces and a plate thickness
- 6. Finding the 2D CPL of a planar 3D Surface
- 7. Replace a solid block by covering plates
- 8. Replace a solid block by covering plates with punched contours
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.5.1 required)

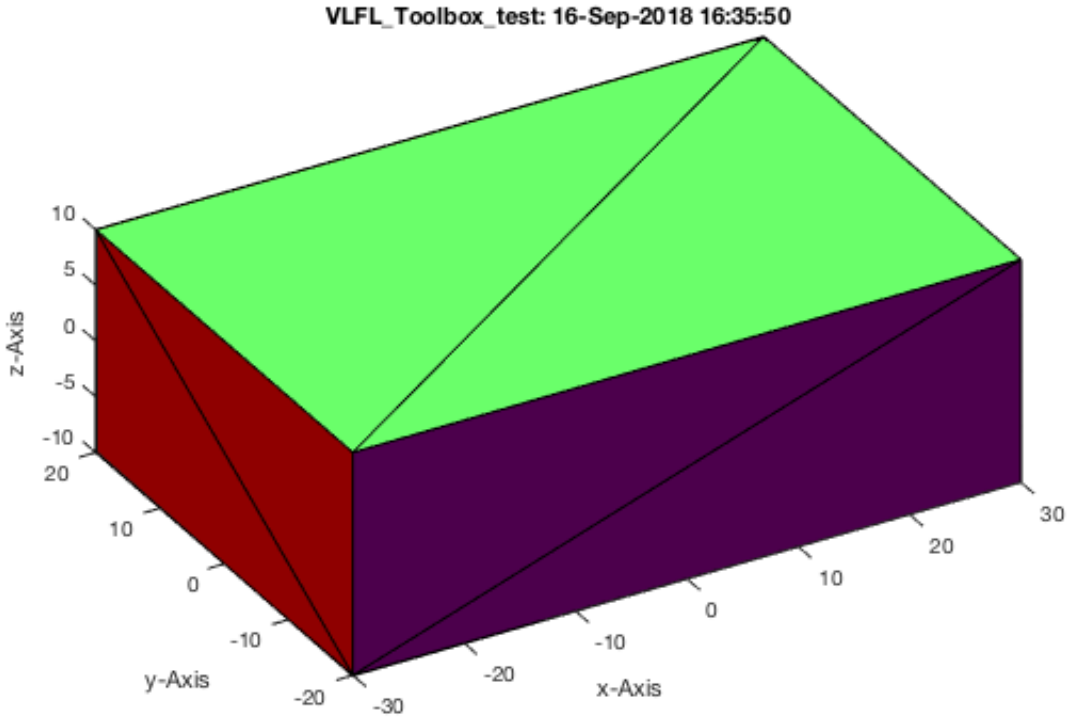
2. Analyzing mouting faces of flat surfaces

All planar faces of a solid can be considered as mounting faces for different design purpses. It is useful to calculate or to handle them using the following functions:

- MLofSG - creates the mounting faces and calculates normal vectors and sizes
- MLplot - plots the mounting faces in different colors

The following example shows the separation of a solid into a set of mouting faces wich are represented by a number and a correlation list between triangle faces and mounting faces.

```
close all; SGfigure; view (-30,30);
[ML,MA,SG]=MLofSG(SGbox([60,40,20]));
MLplot(SG);
```



- ML defines for all entries of FL the corresponding mounting face.
- In this example, 12 faces are ordered to 6 mounting faces

```
ML
```

```
ML =

     1
     1
     2
     2
     3
     3
     4
     4
     5
     5
     6
     6
```

- MA describes for each mounting face, the number, the size, and the normal vector.
- In this example, we see 6 faces with different normal vectors and sizes

```
MA
```

```
MA =

           1        4800           0           0          -1
           2        4800           0           0           1
           3        1600           1           0           0
           4        1600          -1           0           0
           5        2400           0          -1           0
           6        2400           0           1           0
```

- SG is a struct of VL and FL extended by ML and MA

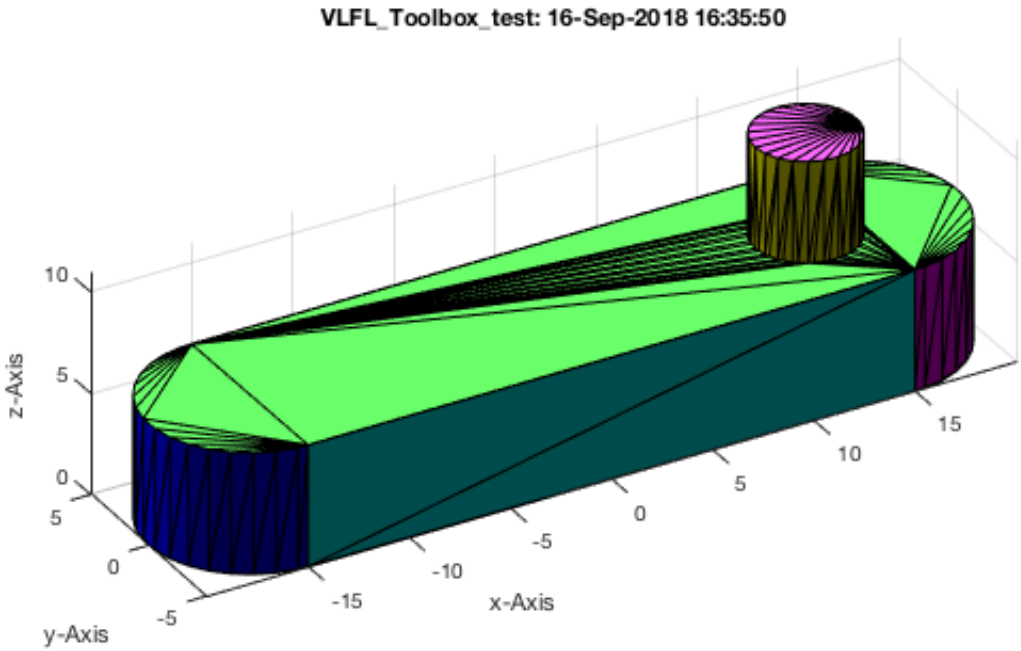
```
SG
```

```
SG =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]
    ML: [12×1 double]
    MA: [6×5 double]
```

3. Analyzing mouting faces of spherical/freeform surfaces

The concepts of mounting faces supports also spherical mounting faces. In case of spherical mounting faces the length of the normal vector is shorter than 1!. This information can be used to distinguish between planar and spherical or freeformed mounting faces

```
close all; SGfigure; view (-30,30);
load AIM_SGrobot
[ML,MA,SG]=MLofSG(SG2);
MLplot(SG);
```

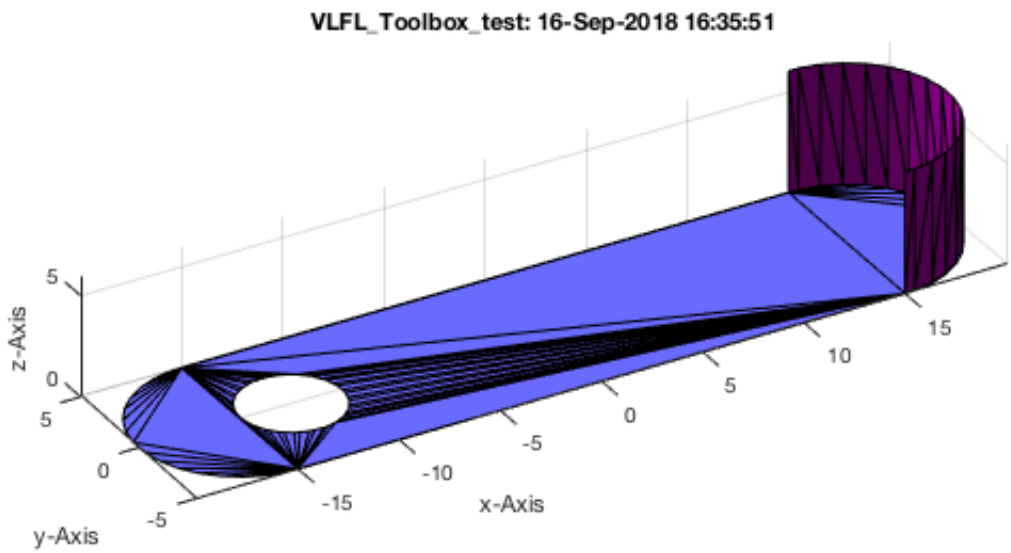


Now we plot only the two half-spherical mounting faces 5 and 1 of the robot link

```
close all; SGfigure; view (-30,30);
MA
MLplot(SG,5); % bended surface, since length of normal vector is less than 1
MLplot(SG,1); % planar surface since length of normal vector is 1
z1=MA(12,3:5) % normal vector of the cylindric surface 1
z2=MA(14,3:5) % normal vector of the cylindric surface 2
```

```
MA =

    1.0000  717.2978         0         0   -1.0000
    2.0000  717.2974         0         0    1.0000
    3.0000  360.0000         0   -1.0000         0
    4.0000    0.2880    1.0000         0         0
    5.0000  188.1982    0.5972         0         0
    6.0000    0.2880    1.0000         0         0
    7.0000  360.0000         0    1.0000         0
    8.0000    0.2880   -1.0000         0         0
    9.0000  188.1982   -0.5972         0         0
   10.0000    0.2880   -1.0000         0         0
   11.0000   38.7854         0         0   -1.0000
   12.0000  156.5960   -0.0001    0.0000    0.0000
   13.0000   38.7829         0         0    1.0000
   14.0000  156.5910    0.0000    0.0000    0.0000
```

```
z1 =

   1.0e-04 *

   -0.6543    0.2818    0.1521


z2 =

   1.0e-04 *

    0.1239    0.3370    0.0978
```

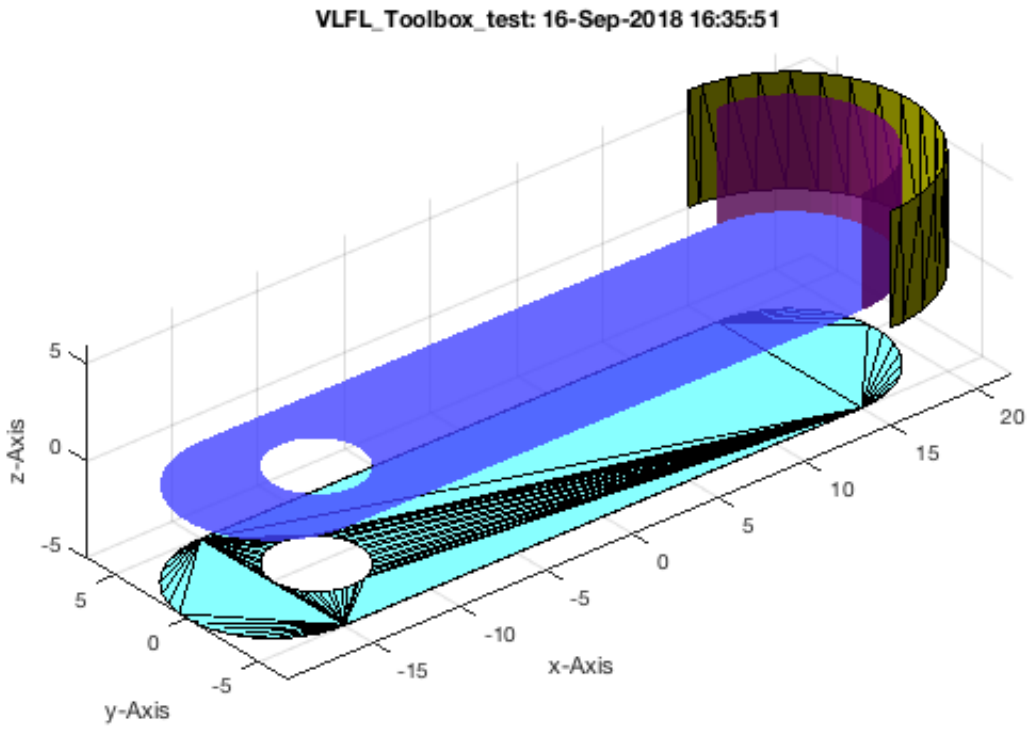


4. Create corresponding surfaces parallel to mounting faces

It is useful to create correspondig surface parallel to mounting faces, which can be smaller or larger than the original one. In the next example it is shown how to create a parallel surface in distance 5mm for a planar surface (#1) and a spherical surface (#5).

- VLtransN(VL,FL,shrink, distance) - helps to create corresonding surfaces

```
VLFLplotlight(1,0.8); view(-40,30);
[VL,~,~,FL]=VLtransN(SG.VL,SG.FL(ML==5,:),0,2);
VLFLplot(VL,FL,'y');

[VL,~,~,FL]=VLtransN(SG.VL,SG.FL(ML==1,:),0,5);
VLFLplot(VL,FL,'c');
```

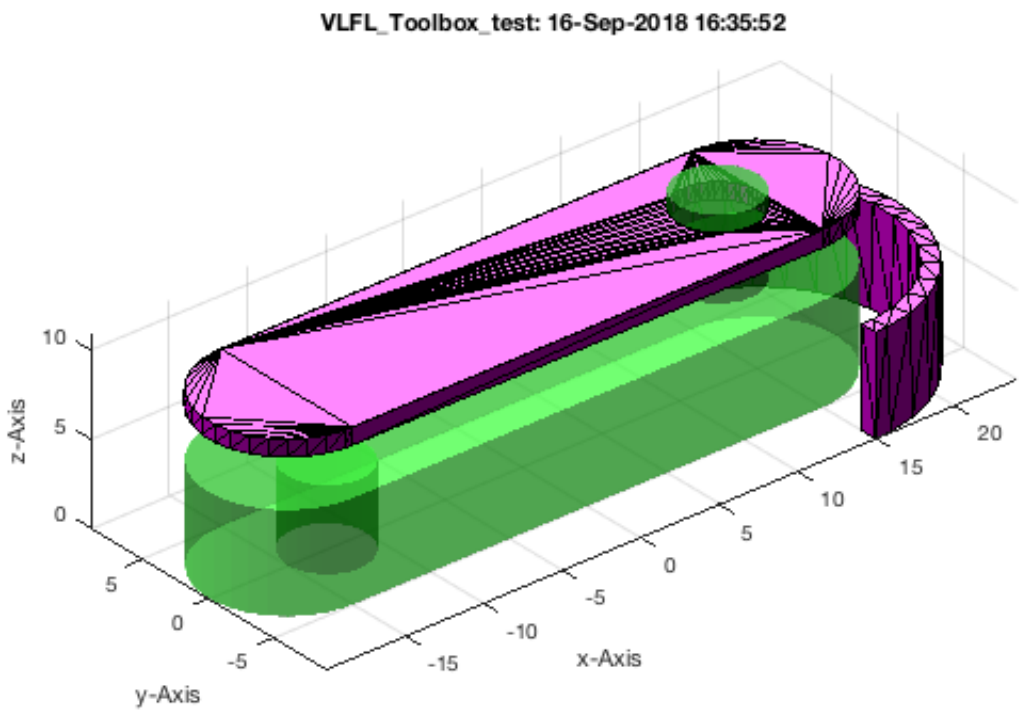


5. Create solids using the parallel surfaces and a plate thickness

Often we want to create a plate solid parallel to the mounting face.

- SGofSurface(VL,FL,thickness, distance, streching) - creates solids parallel to mounting faces.

```
close all; SGfigure; view (-30,30);
SGplot(SG); VLFLplotlight(1,0.5); view(-40,30);
SG2=SGofSurface(SG.VL,SG.FL(ML==2,:),1,3);
SGplot(SG2,'m');
SG2=SGofSurface(SG.VL,SG.FL(ML==5,:),1,3);
SGplot(SG2,'m');
```

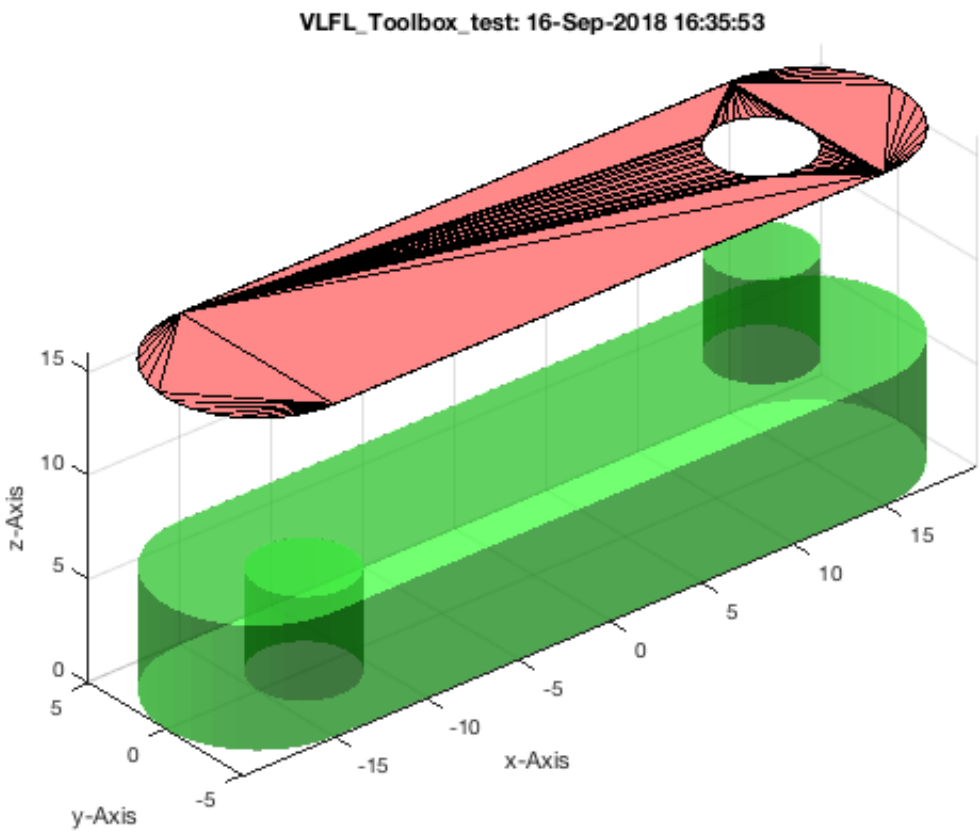


6. Finding the 2D CPL of a planar 3D Surface

Many procedures are based on the manipulation of CPL contours. Nevertheless not all planar surfacer are in the xy-plane. Therefor, there is a function that creates a CPL contour of a surface and returns also the transformation matrix for the back transformation.

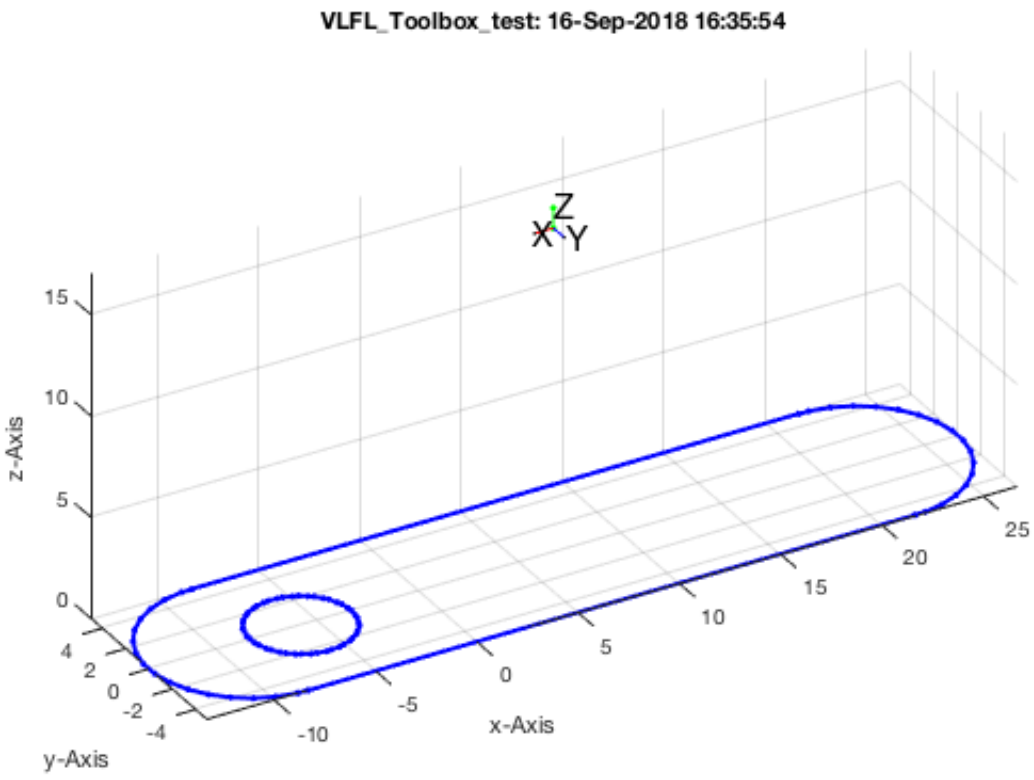
- [PL,T]=PLofVLFL(VL,FL) - returns a PL and a transformation matrix
- [CPL,T]=CPLofVLFL(VL,FL) - returns a CPL and a transformation matrix

```
close all; SGfigure; view (-30,30);
SGplot(SG); VLFLplotlight(1,0.5); view(-40,30);
[VL,~,~,FL]=VLtransN(SG.VL,SG.FL(ML==2,:),0,10);
VLFLplot(VL,FL);
```



Now show simply the isolated CPL of this mounting face

```
close all; SGfigure; view (-30,30); axis on; grid on;
[CPL,T]=CPLofVLFL(VL,FL);
CPLplot(CPL,'b.-',2);
plotT(T);
```

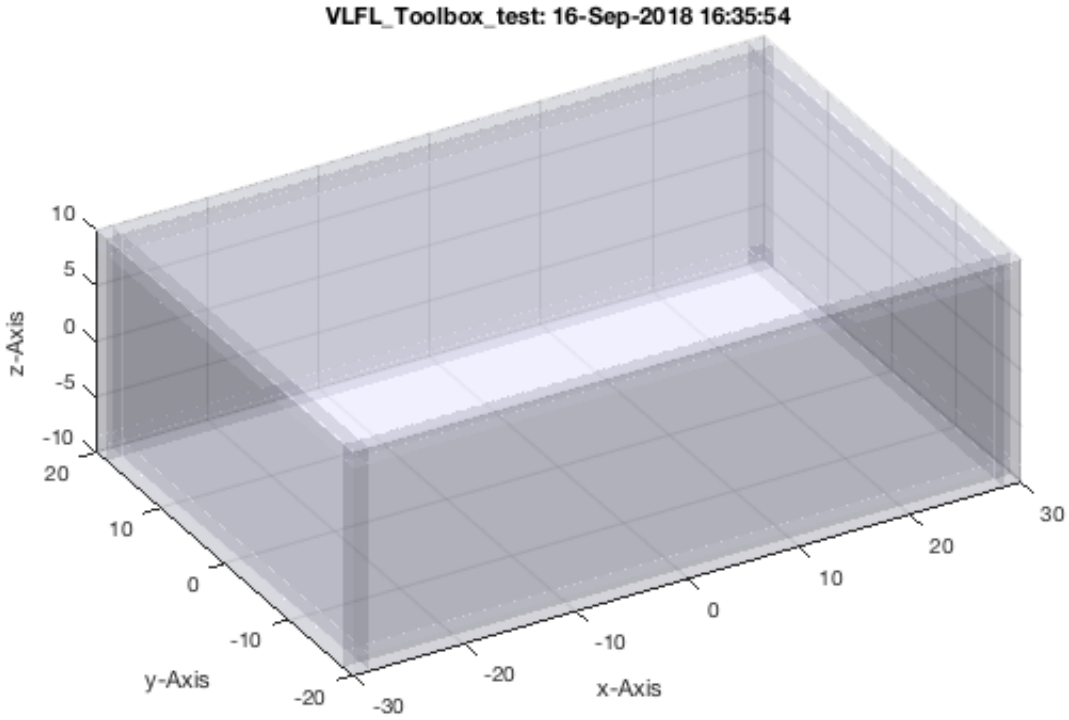


7. Replace a solid block by covering plates

By converting a solidblock into a hollow structure using covering plates, the weight an mass inertia of a solid is reduced.

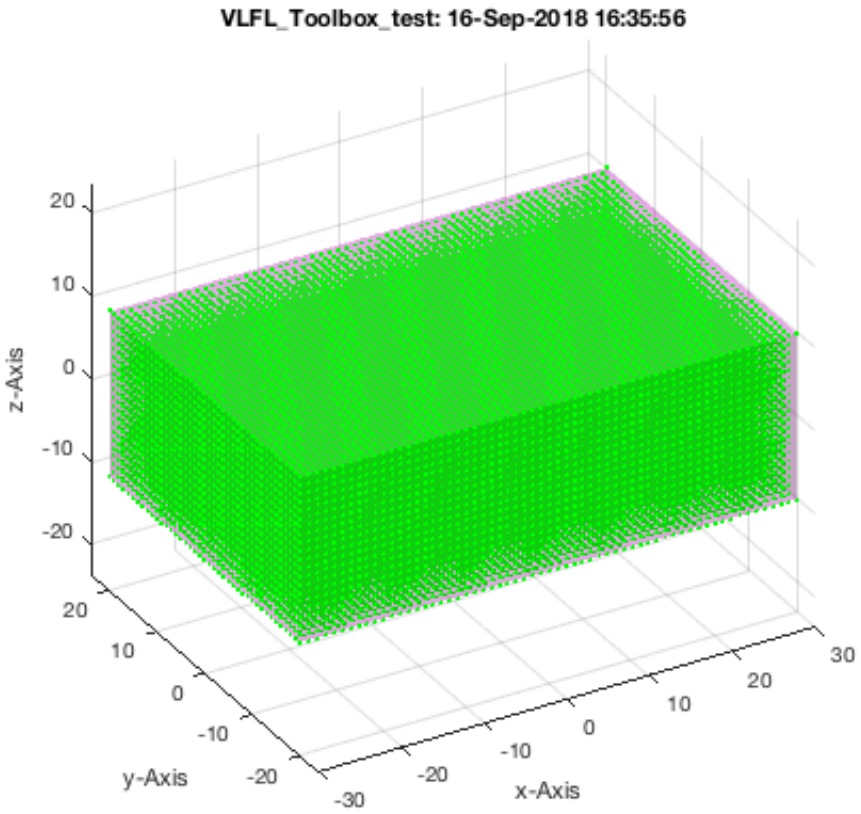
- SGplatesofSGML(SG,thickness) - convert a solid into a plate structure
- SGweight (SG,sepecific weight,resolution) - slowly calculates the weight

```
close all; SGfigure; view (-30,30);

SGN=SGplatesofSGML(SGbox([60,40,20]),1.5);
SGplot(SGN,'w'); VLFLplotlight(1,0.2);
```



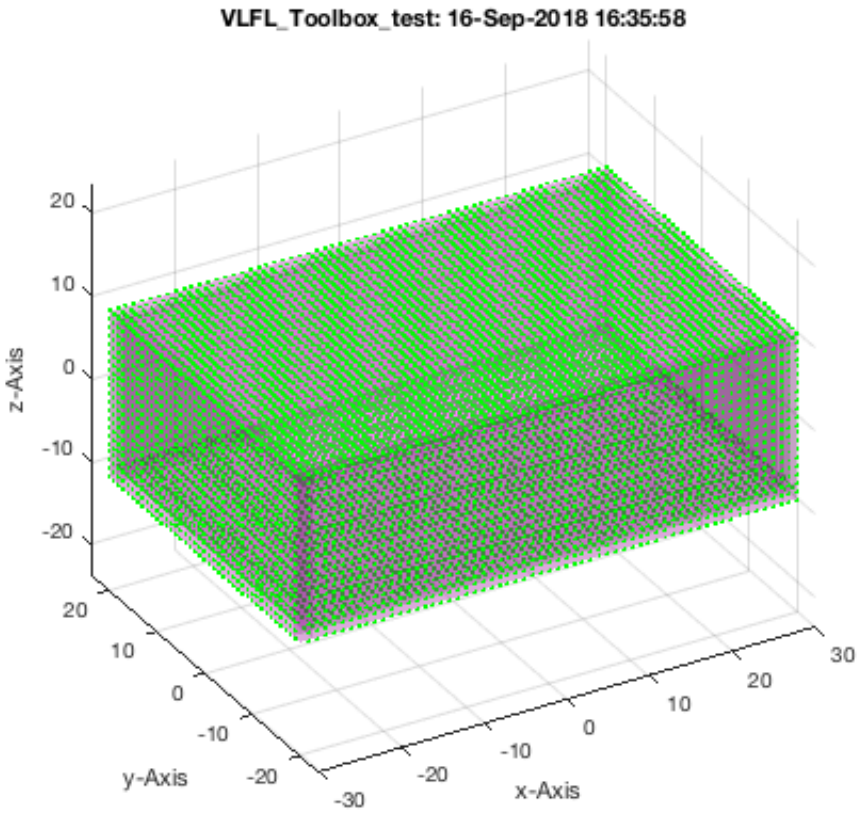

```
SGweight(SGbox([60,40,20]),[],1);
```

```
Using a resolution of 1.0 mm^3 (n=62307) and a specific weight of 1.15 milligramm per mm^3,
 the overall weight is ca. 58 gramm.
Elapsed time is 1.500283 seconds.
```




```
SGweight(SGN,[],1);
```

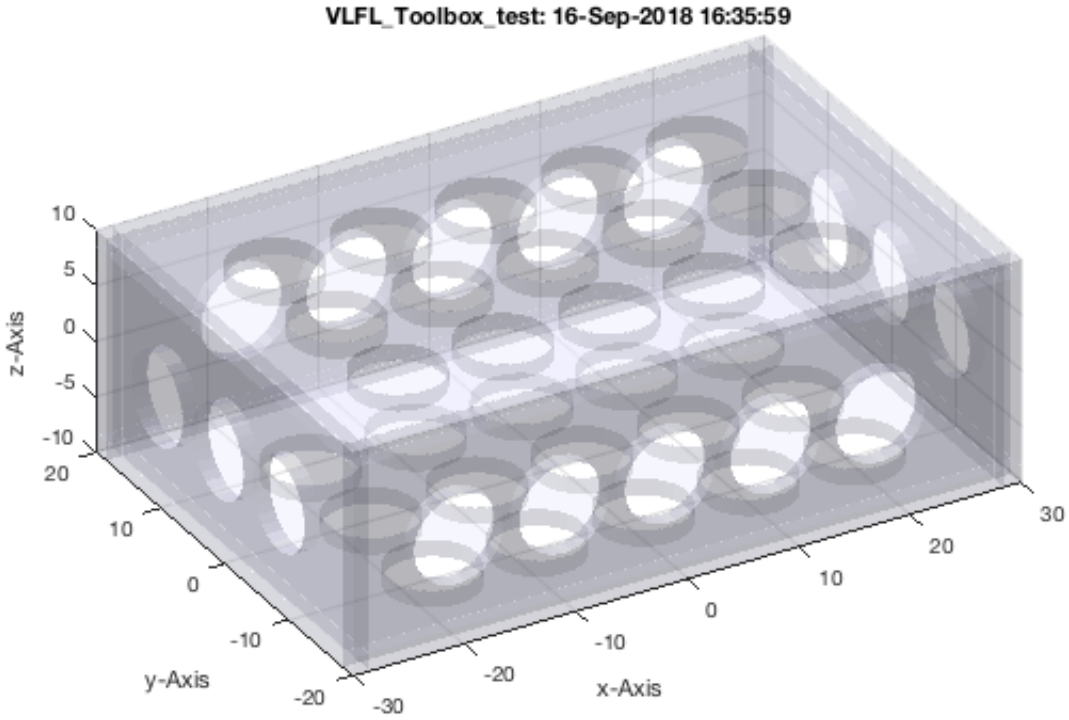
```
Using a resolution of 1.0 mm^3 (n=62307) and a specific weight of 1.15 milligramm per mm^3,
 the overall weight is ca. 11 gramm.
Elapsed time is 1.599883 seconds.
```



8. Replace a solid block by covering plates with punched contours

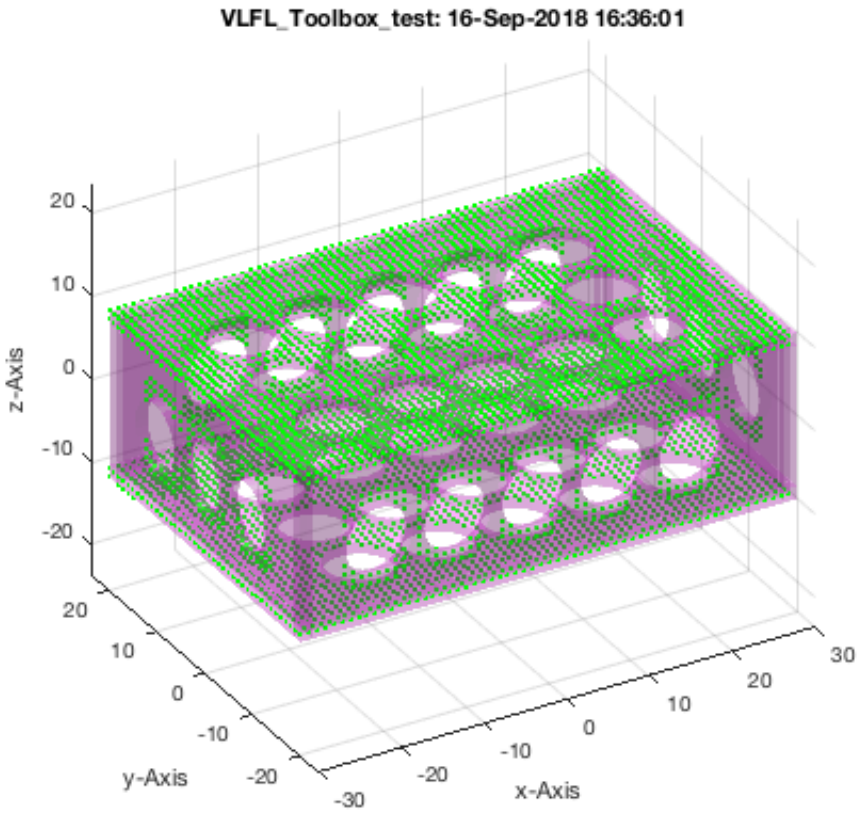
- SGplatesofSGML(SG,thickness,CPL) - convert a solid into a punched plate structure

```
close all; SGfigure; view (-30,30);
SGN=SGplatesofSGML(SGbox([60,40,20]),1.5,PLcircle(4));
SGplot(SGN,'w'); VLFLplotlight(1,0.2);
```




```
SGweight(SGN,[],1);
```

```
Using a resolution of 1.0 mm^3 (n=62307) and a specific weight of 1.15 milligramm per mm^3,
 the overall weight is ca. 7 gramm.
Elapsed time is 1.804665 seconds.
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:36:02!
Executed 16-Sep-2018 16:36:04 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-09-11*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)

2015-09-20: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.6.1 required)
- 2. Fill a contour with copies of pattern
- 3. Writing a contour as SVG-File for laser-cutting
- 4. Calculating the normal vectors of edges and points
- 5. Growing with same number of points
- 6. Growing with correct distance to edges
- 7. Rounded edges inside a contour
- 8. Sort CPLs around its center
- 9. Informations on contours inside of others
- 10. Order contours for the sequential plot with a laser cutter
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)

- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

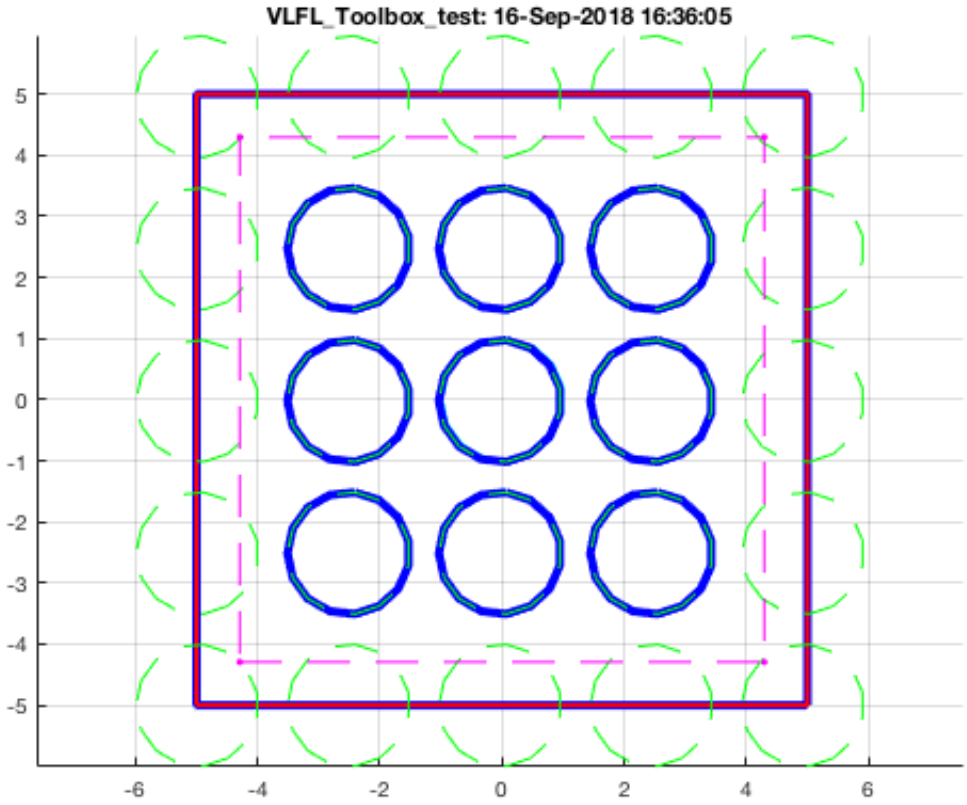
Motivation for this tutorial: (Originally SolidGeometry 2.6.1 required)

2. Fill a contour with copies of pattern

As it was shown already in the function SGplatesofSGML, it often makes sense to fill a contour with another pattern. This can be done by using one of the following functions:

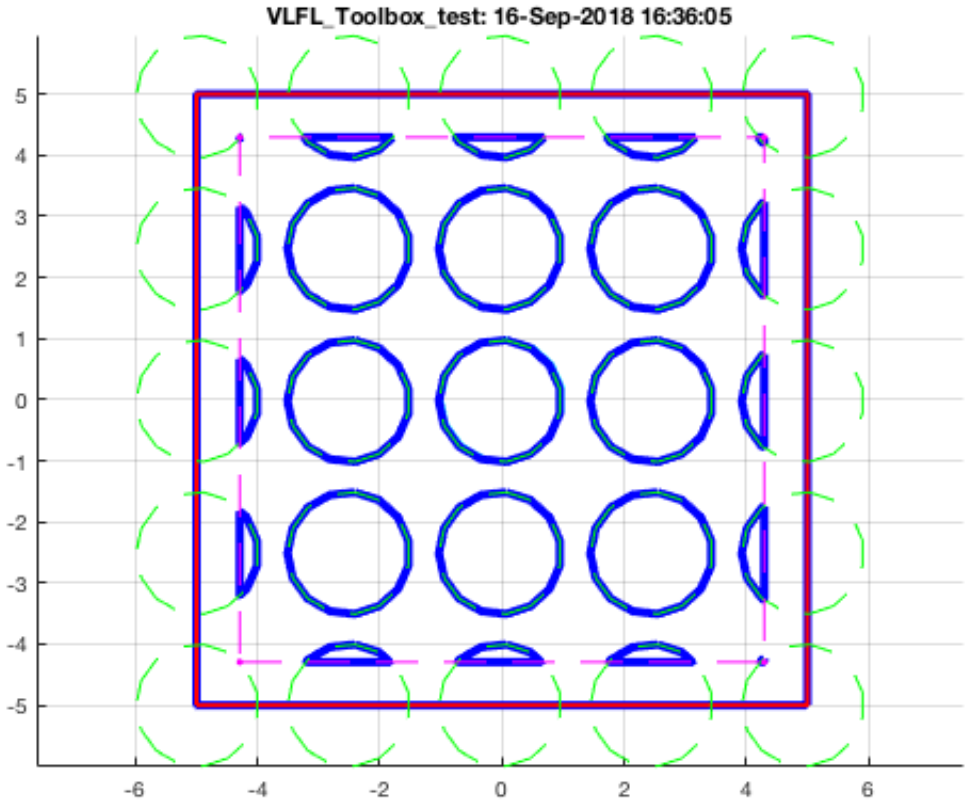
- **CPLfillPattern(CPLA, CPLB,w,d)** - fills a contour CPLA with copies of the pattern CPLB with a distance to the outer contour w and distance between the patterns of d

```
SGfigure; view(0,90); axis on;
CPLfillPattern(PLsquare(10,10),PLcircle(1),1);
```



This can also be done with cutted pattern instead of complete pattern

```
SGfigure; view(0,90); axis on;
CPLfillPattern(PLsquare(10,10),PLcircle(1),1,[],true);
```



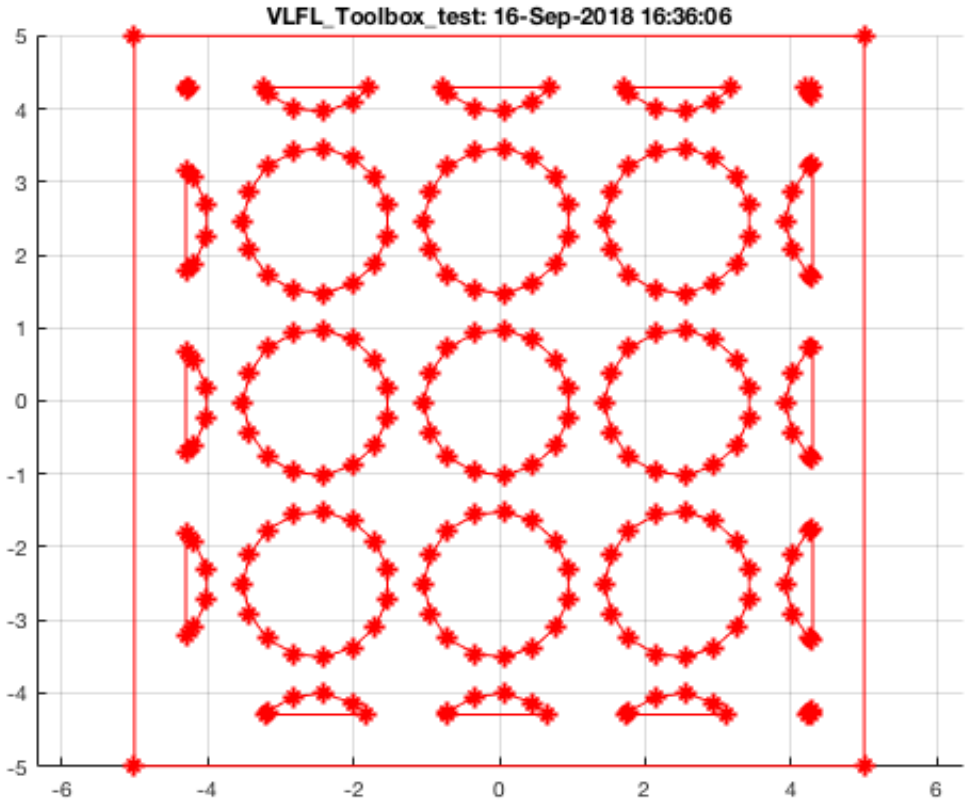
3. Writing a contour as SVG-File for laser-cutting

Especially for laser cutting or platting of contours, the SVG file-format is very popular. Handling of SVG Files is possible using the following functions:

- **CPLwriteSVG (CPL,Filename)** - writes the contours in a SVG-File

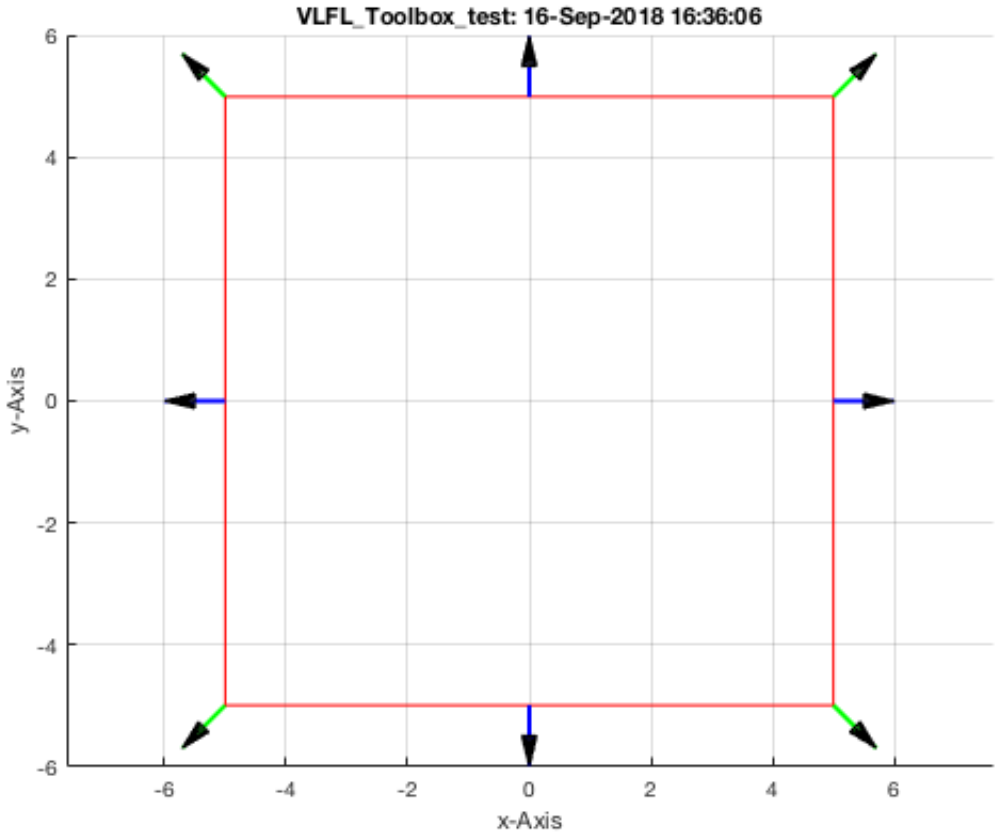
```
SGfigure; view(0,90); axis on;
A=CPLfillPattern(PLsquare(10,10),PLcircle(1),1,[],true);
CPLplot(A);
CPLwriteSVG(A,'VLFL_EXP14');
```

```
WRITING SVG FILE /Users/timlueth/Desktop/Toolbox_test/VLFL_EXP14.SVG in ASCII MODE
completed.
```



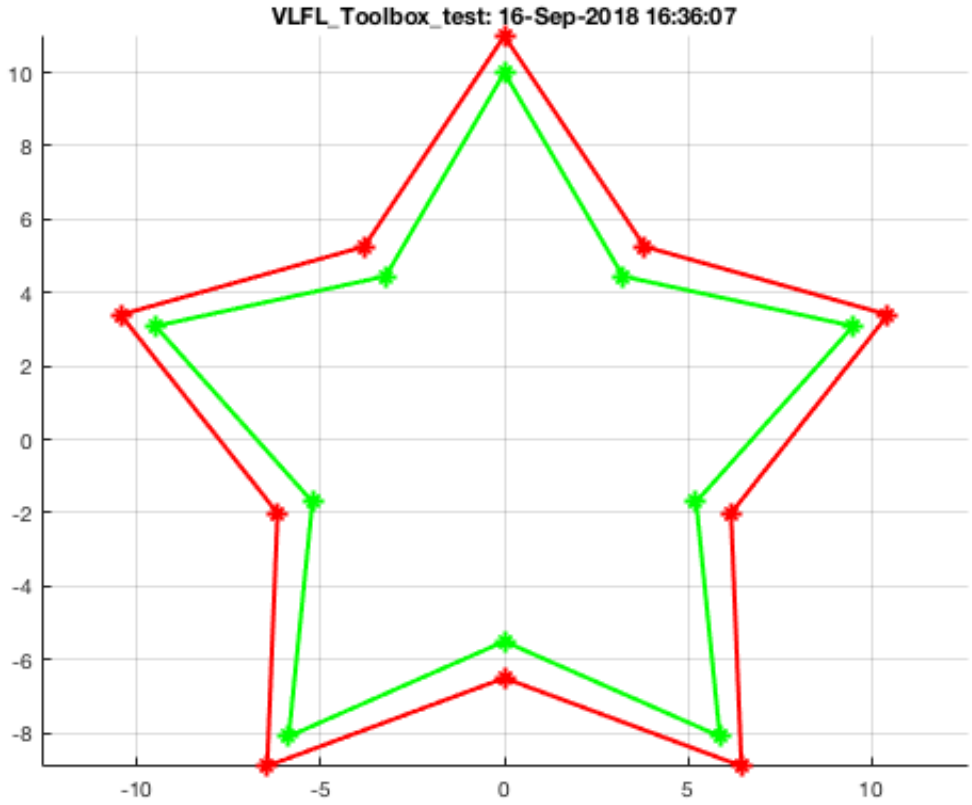
4. Calculating the normal vectors of edges and points

```
SGfigure; view(0,90);
CPLedgeNormal(PLsquare(10,10)); axis on;
```



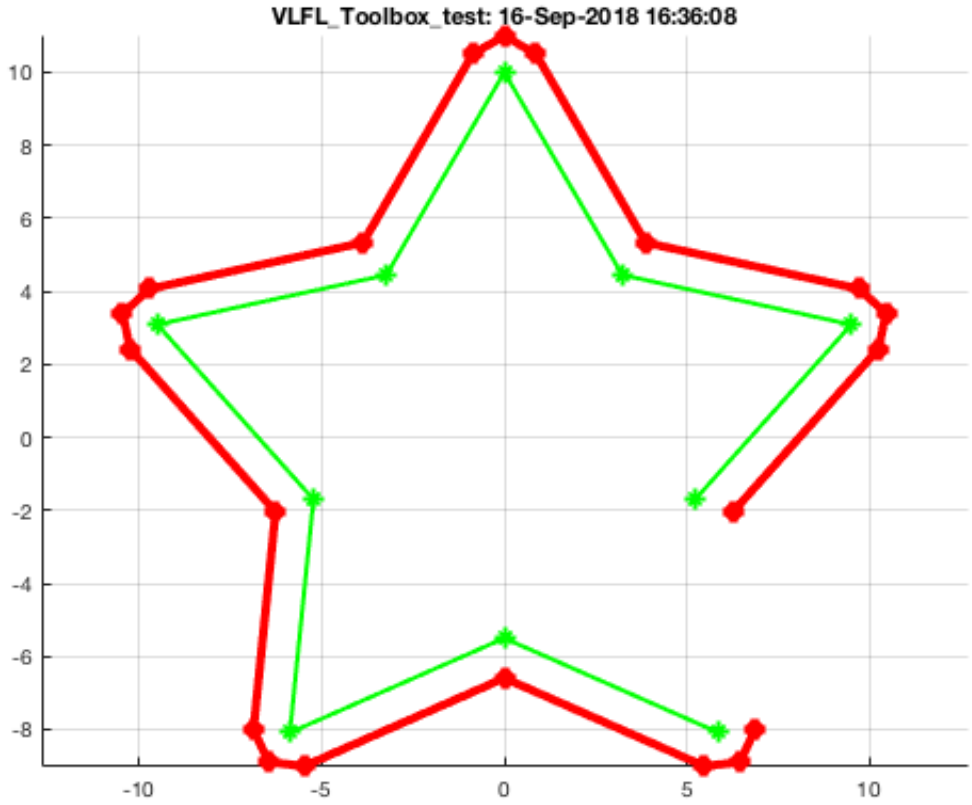
5. Growing with same number of points

```
SGfigure; view(0,90);
CPLgrow(PLstar(10,10),1); axis on;
```



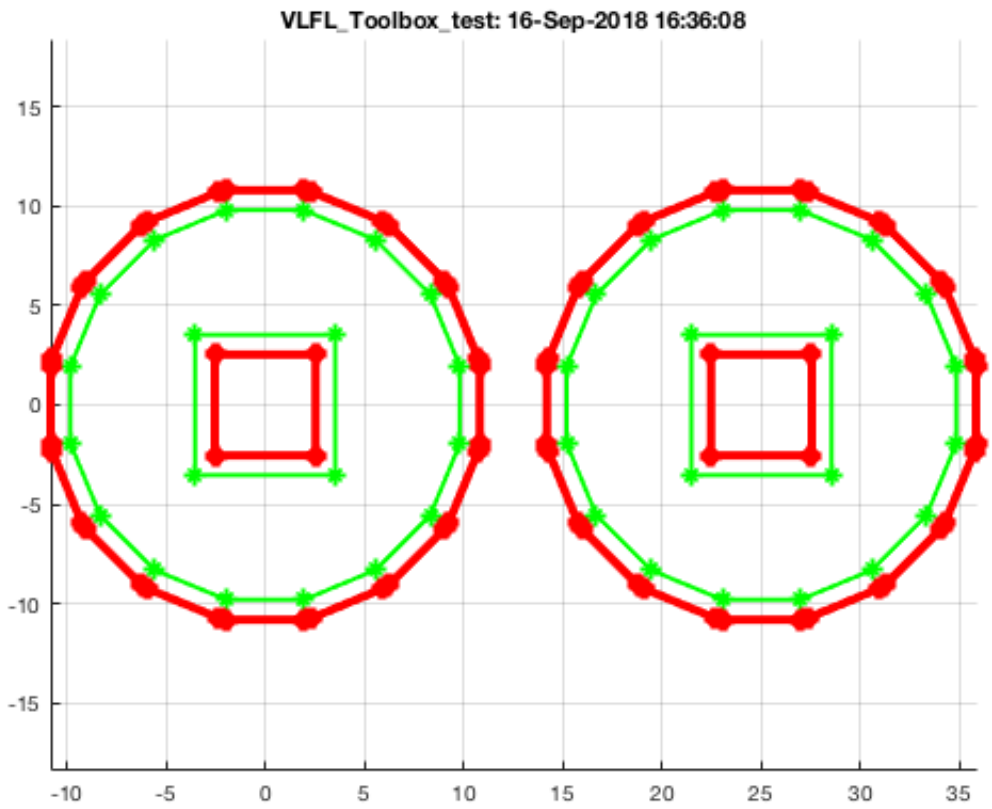
6. Growing with correct distance to edges

```
SGfigure; view(0,90); axis on;
CPLgrowEdge(PLstar(10,10),1);
```



Another example using CPLsample

```
SGfigure; view(0,90);
CPLgrowEdge(CPLsample(12),1); axis on;
```

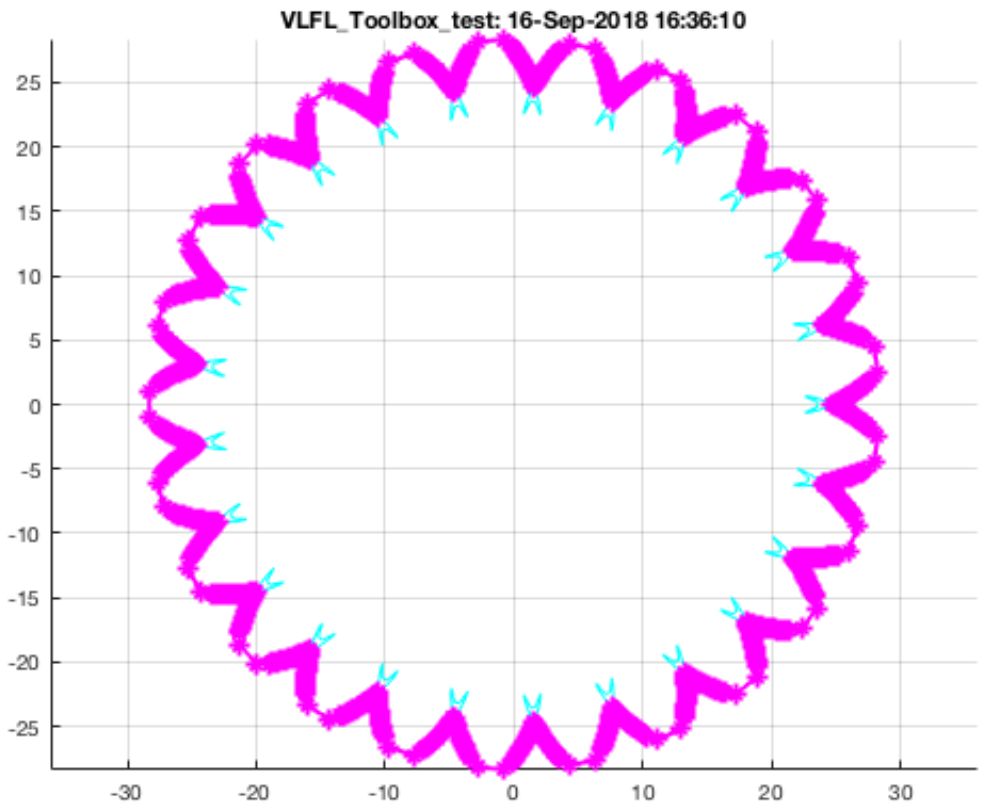


Growing may have no problems

```
SGfigure; view(0,90);
CPLgrow(CPLofPL(PLgearDIN(2,25)),0.5); axis on;

% *Growing may have problems*
SGfigure; view(0,90);
CPLgrow(CPLofPL(PLgearDIN(2,25)),1.5); axis on;

% *Growing problems can be solved using CPLoutercontour*
SGfigure; view(0,90);
CPLoutercontour(CPLgrow(CPLofPL(PLgearDIN(2,25)),1.5));
```

```
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
```

```
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
```

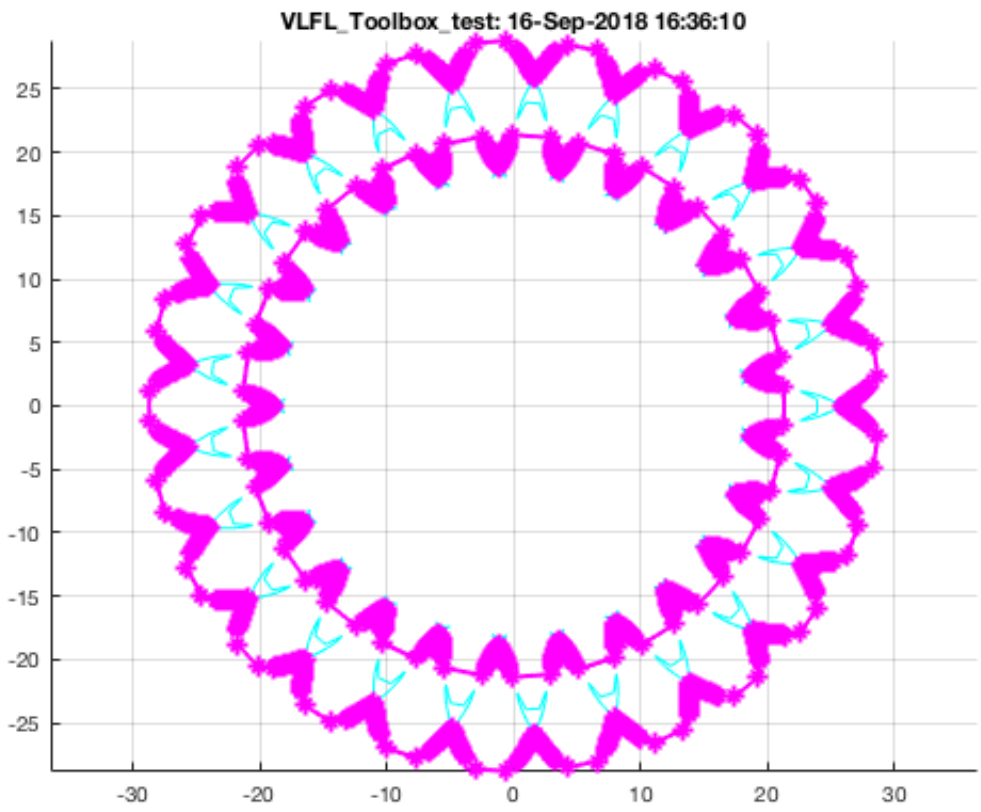


Another example using CPLoutercontour

```
SGfigure; view(0,90);
CPLoutercontour(CPLsample(25),1); axis on;
```

```
Modul: m=2.00
```

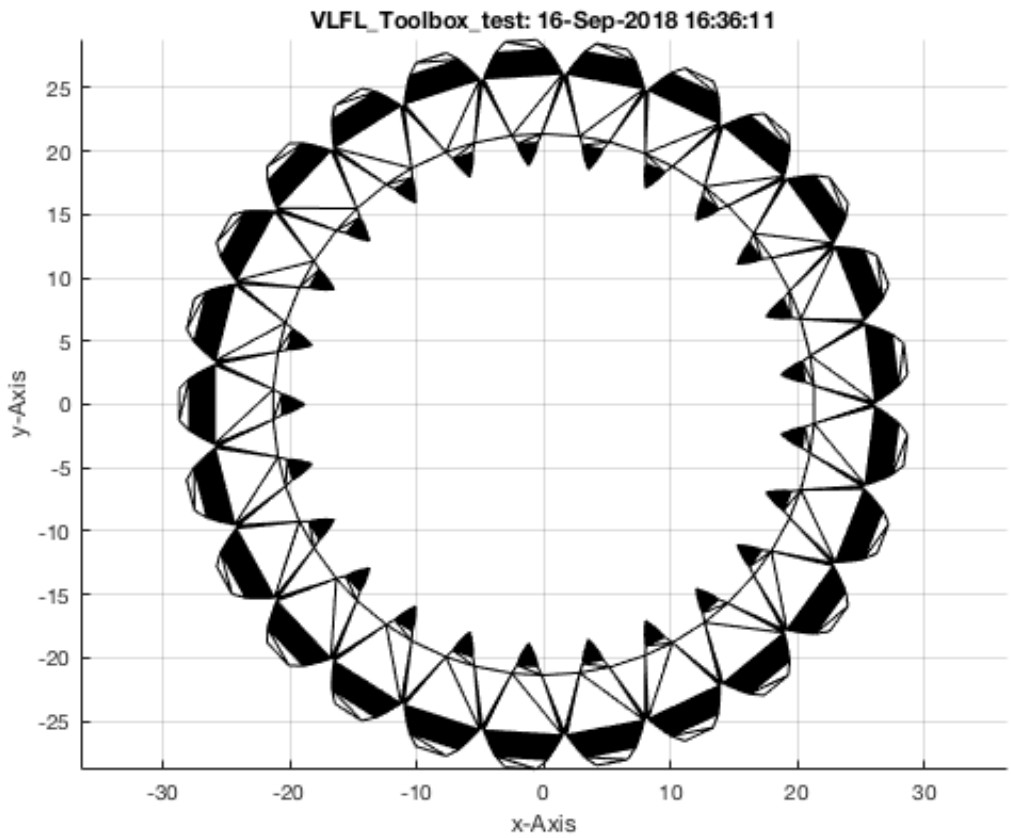
```
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=1.50
Teeth Nr: z=25
Rolling circle: d=37.50
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=41.25
Head diameter: dh=34.50
Tooth foot width: 3.63
Involute angle: 34.09
Tooth head width: 0.25
Difference tooth gap and tooth head width: 1.30
```




```
PLFLofCPLdelaunay(CPLoutercontour(CPLsample(25),1));
```

```
Modul: m=2.00
Teeth Nr: z=25
```

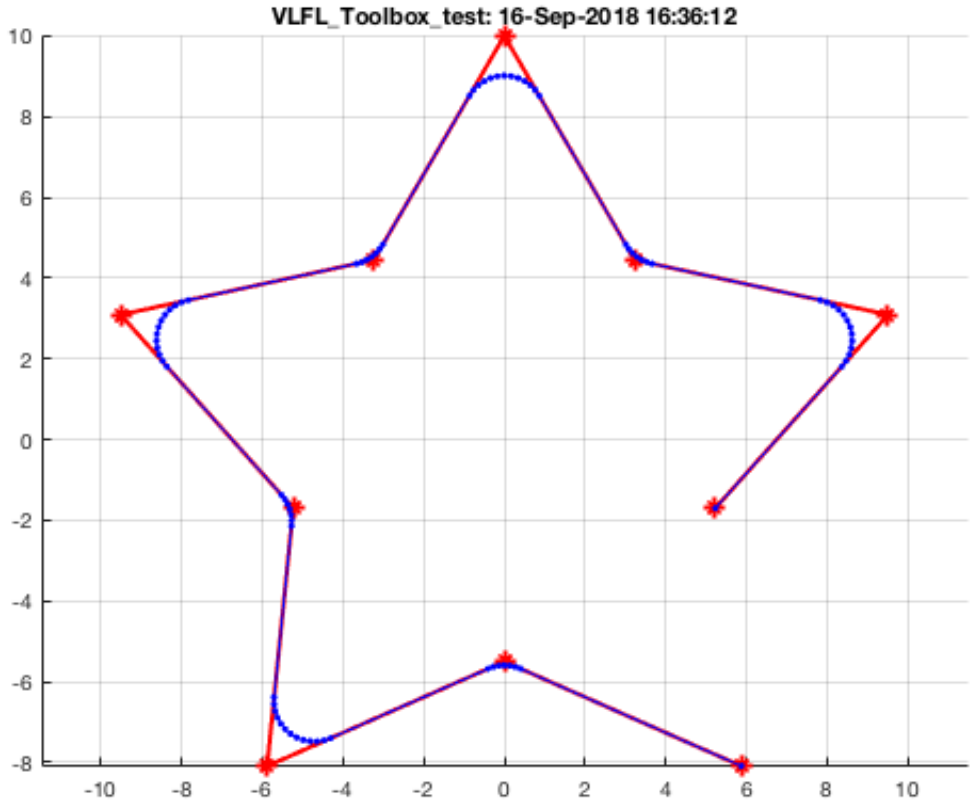
```
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=1.50
Teeth Nr: z=25
Rolling circle: d=37.50
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=41.25
Head diameter: dh=34.50
Tooth foot width: 3.63
Involute angle: 34.09
Tooth head width: 0.25
Difference tooth gap and tooth head width: 1.30
```



7. Rounded edges inside a contour

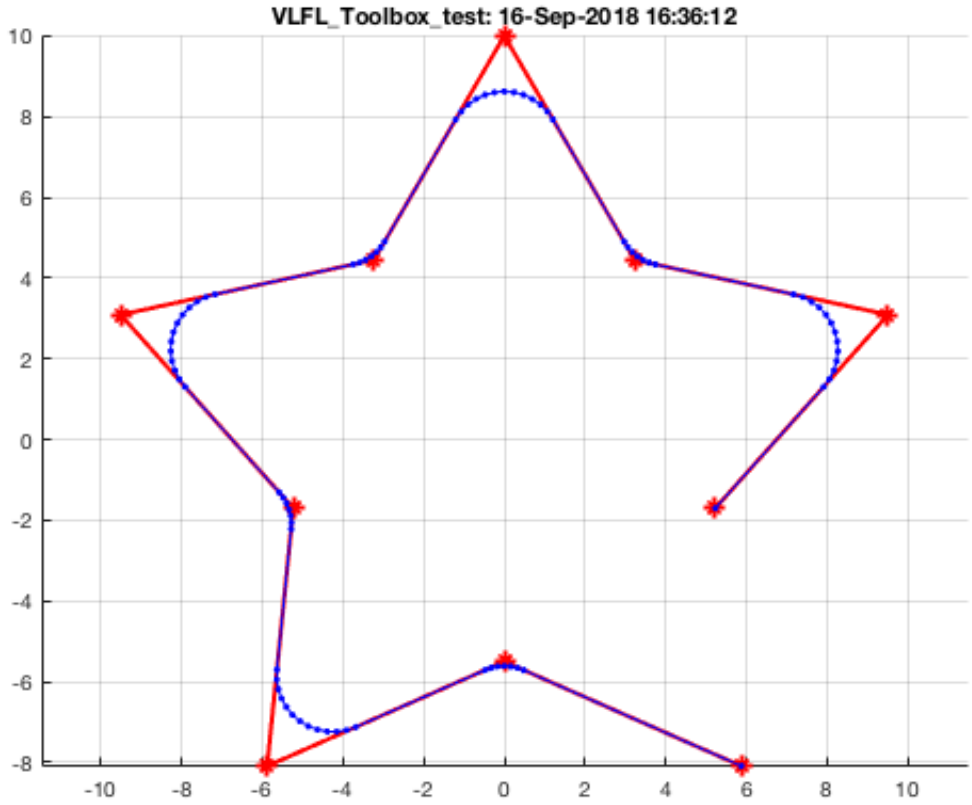
Another method to change the shape of a contour is to round the edges.

```
SGfigure; view(0,90);
PLradialEdges(PLstar(10,10));axis on;
```



Another example using radius=2

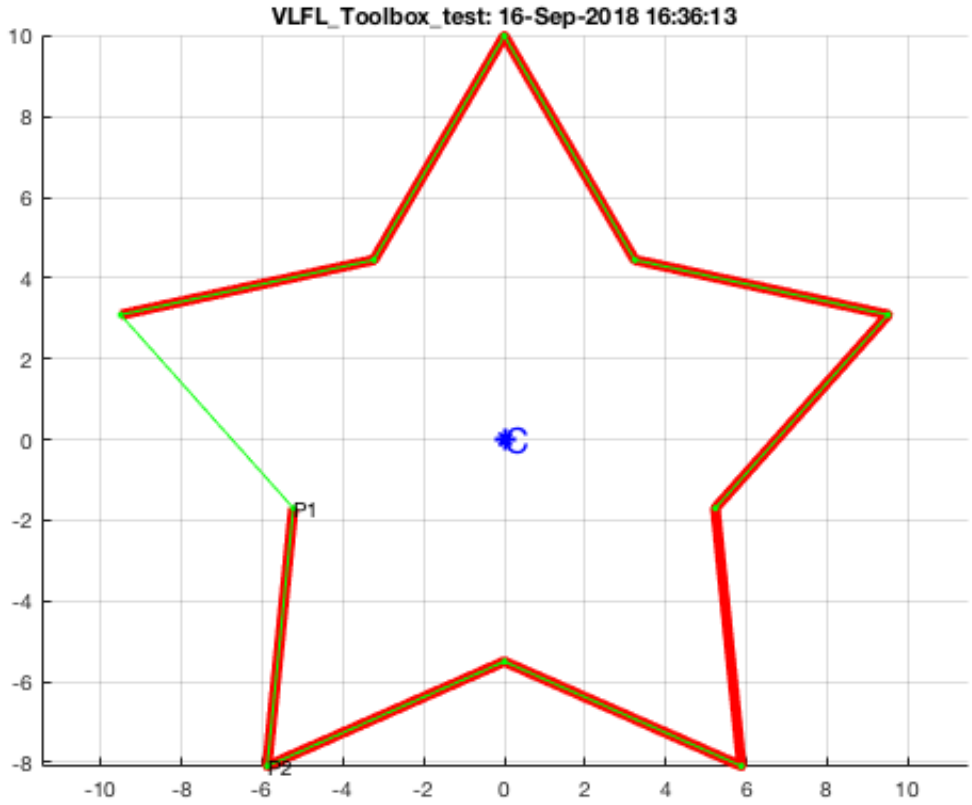
```
SGfigure; view(0,90); axis on;
PLradialEdges(PLstar(10,10),2); axis on;
```



8. Sort CPLs around its center

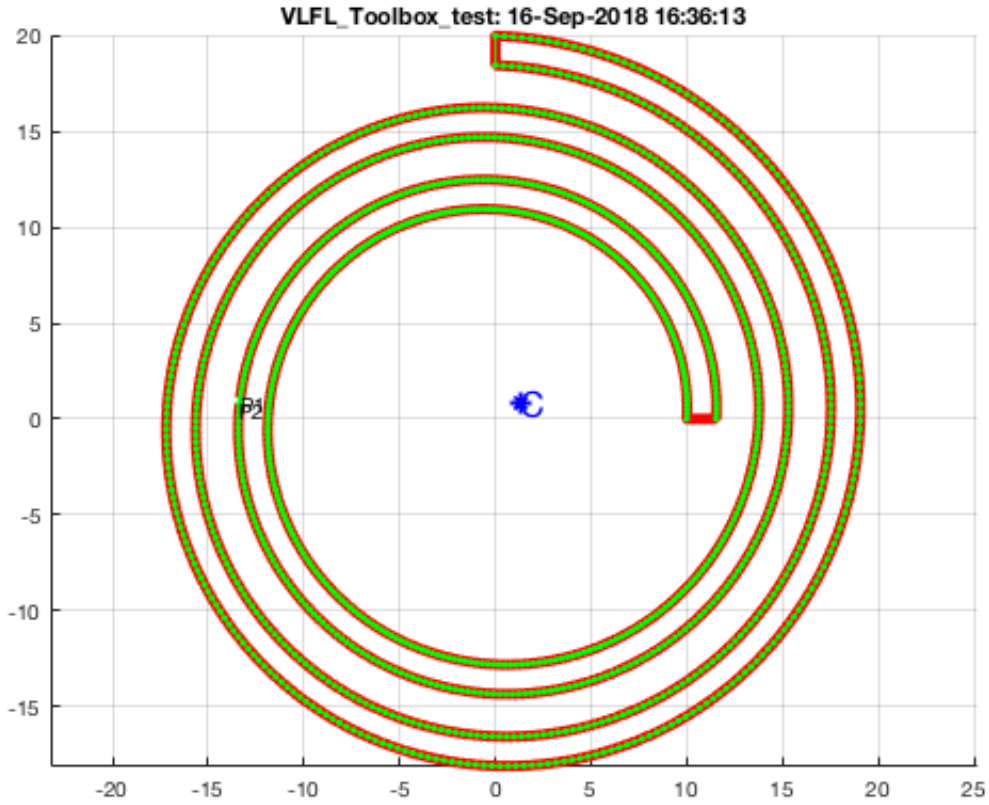
Find the minmal angle value of a star

```
SGfigure; view(0,90);
CPLsortC(PLstar(10,10),'min'); axis on;
```



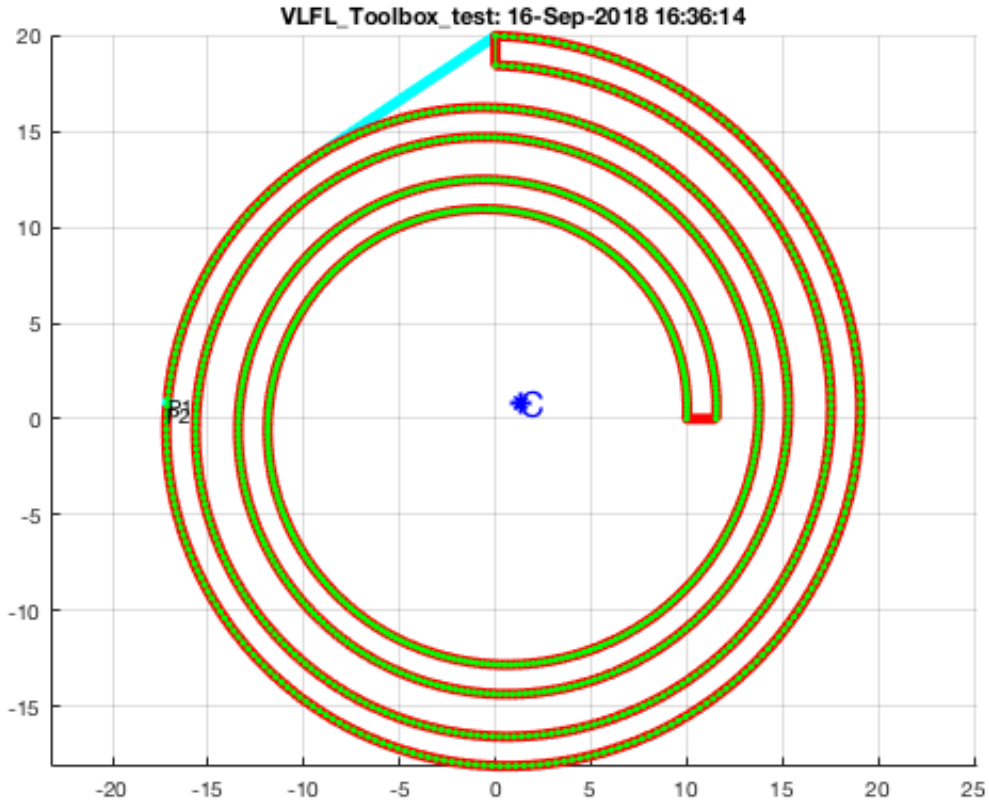
Find the minmal angle value of a spiral

```
SGfigure; view(0,90);
CPLsortC(CPLspiral(10,20,4*pi+pi/2),'min');
```



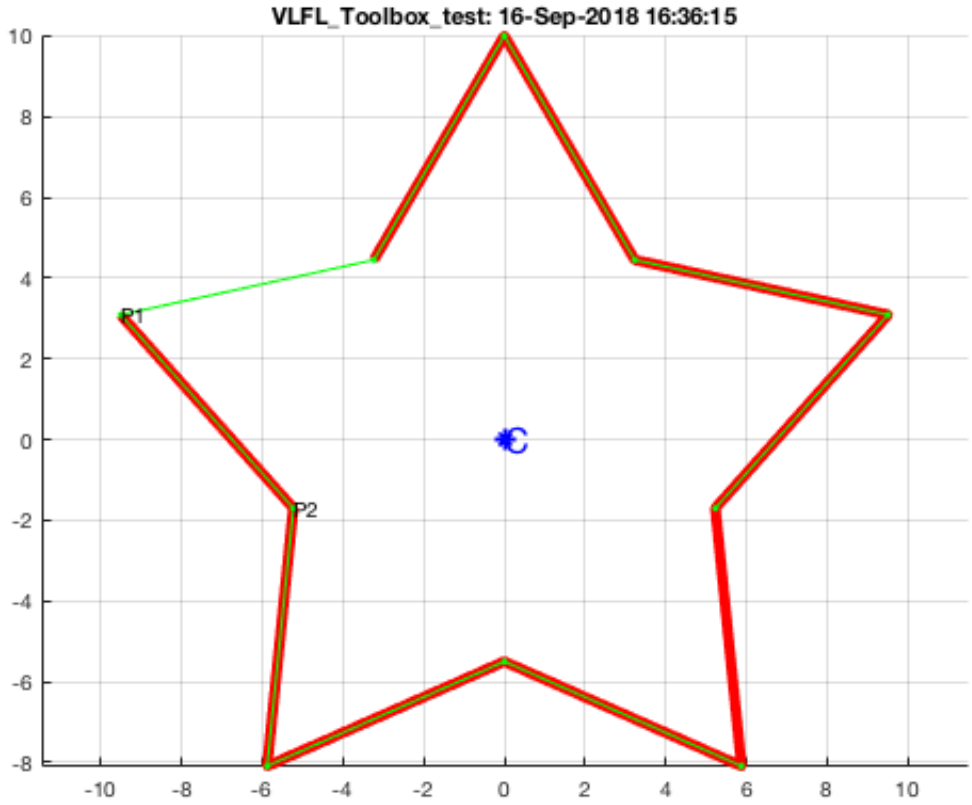
Find the minmal angle value of convex hull of a spiral

```
SGfigure; view(0,90);
CPLsortC(CPLspiral(10,20,4*pi+pi/2),'cmin');
```



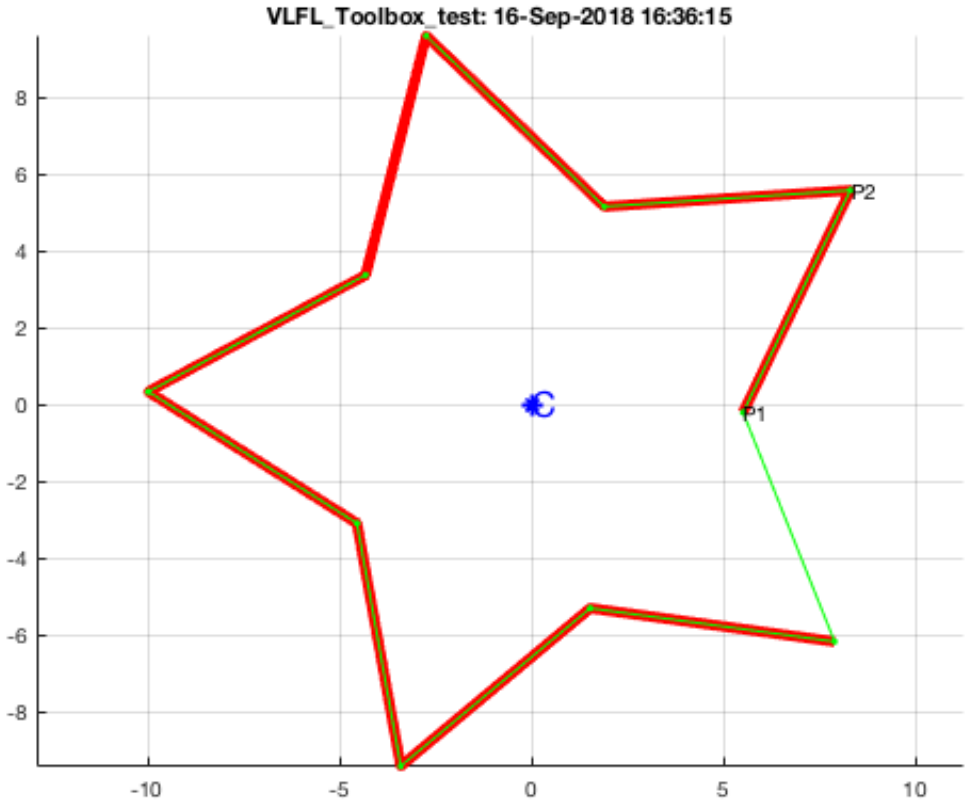
Find the maximum angle value of a star

```
SGfigure; view(0,90);
CPLsortC(PLstar(10,10),'max'); axis on;
```



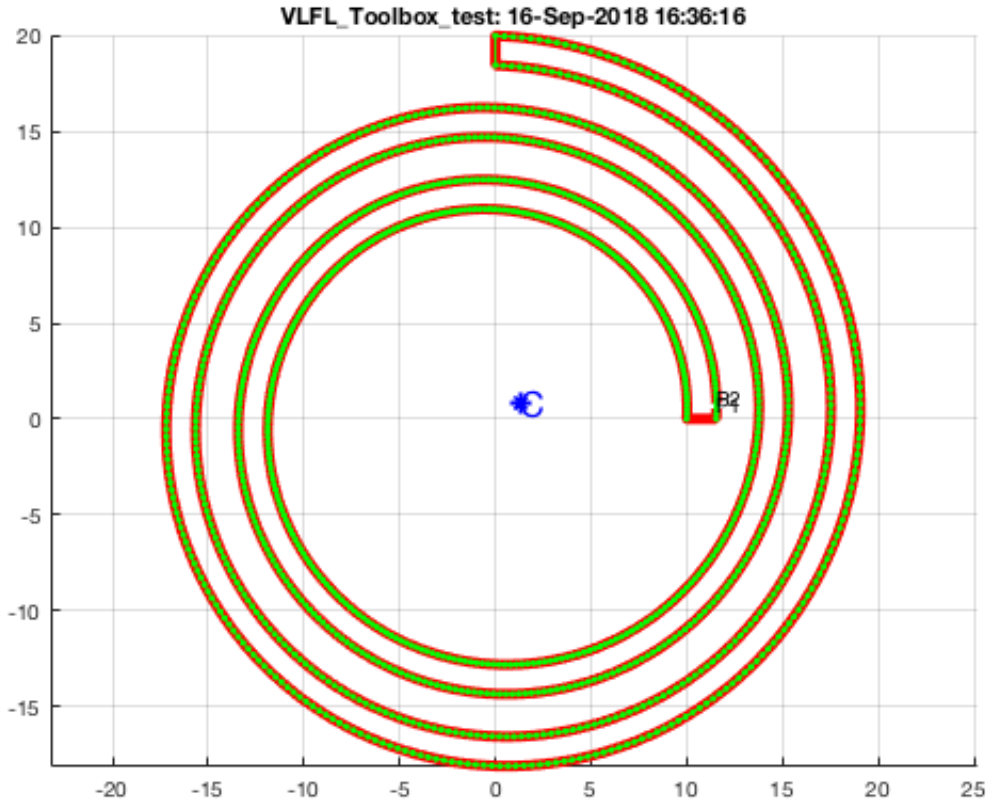
Find the angle value nearest to zero of a star

```
SGfigure; view(0,90);
CPLsortC(PLtrans(PLstar(10,10),rotdeg(160)),'zero'); axis on;
```



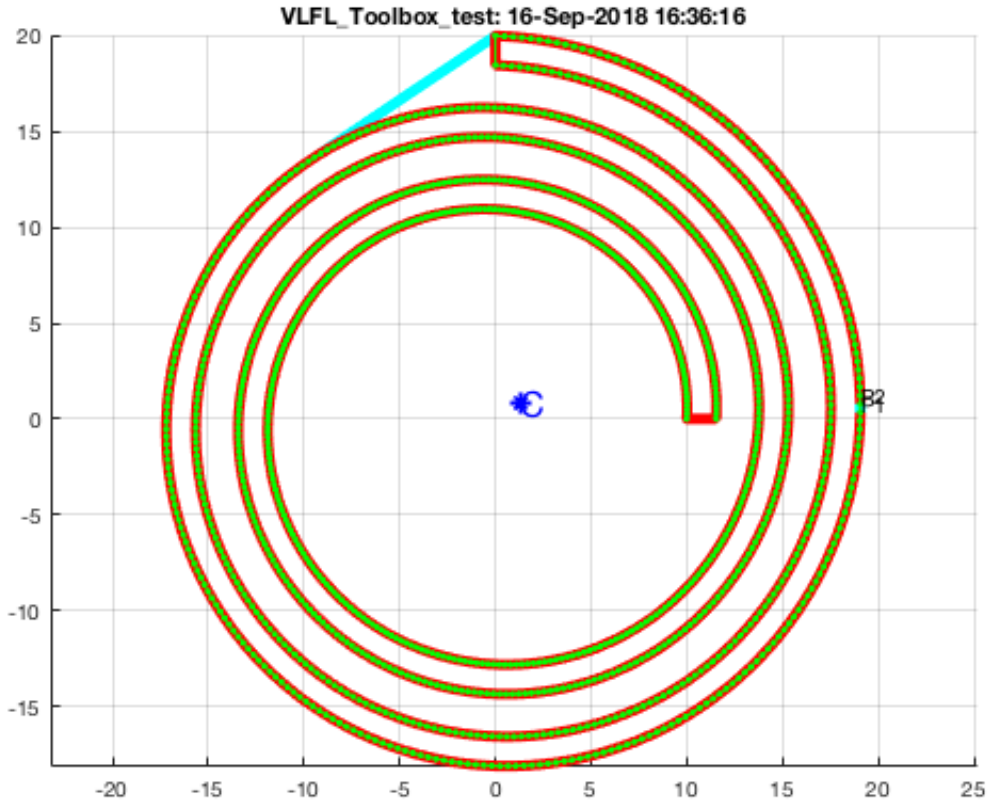
Find the angle value nearest to zero of a spiral

```
SGfigure; view(0,90);
CPLsortC(CPLspiral(10,20,4*pi+pi/2),'zero');
```



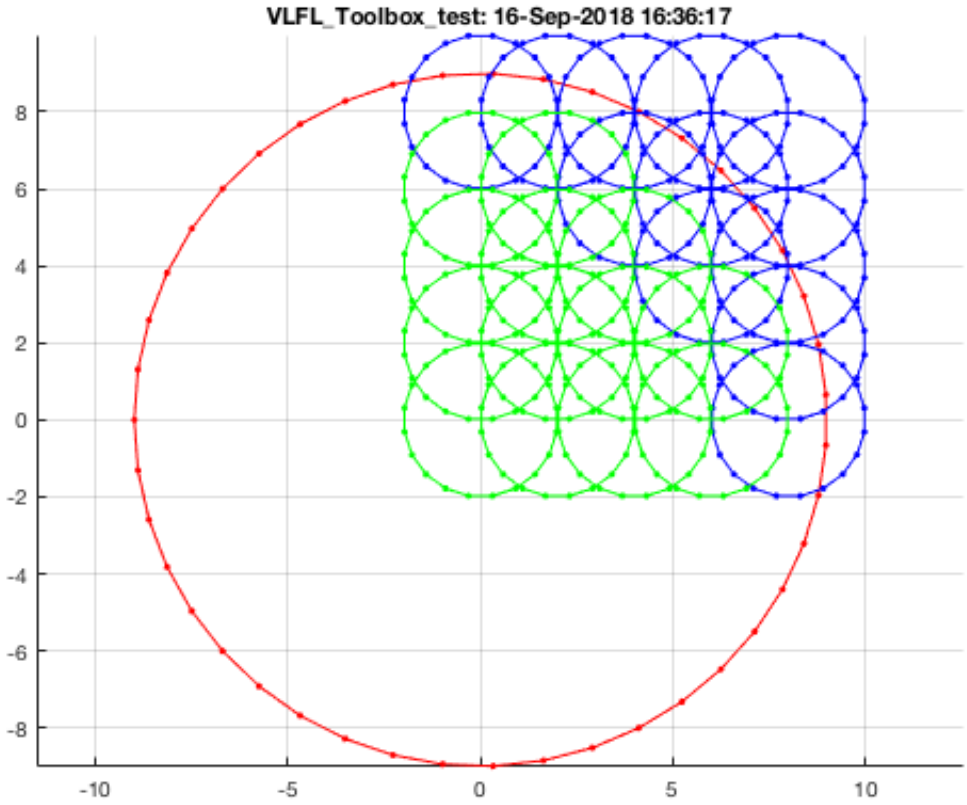
Find the angle value nearest to zero of convex hull of a spiral

```
SGfigure; view(0,90);
CPLsortC(CPLspiral(10,20,4*pi+pi/2),'czero');
```



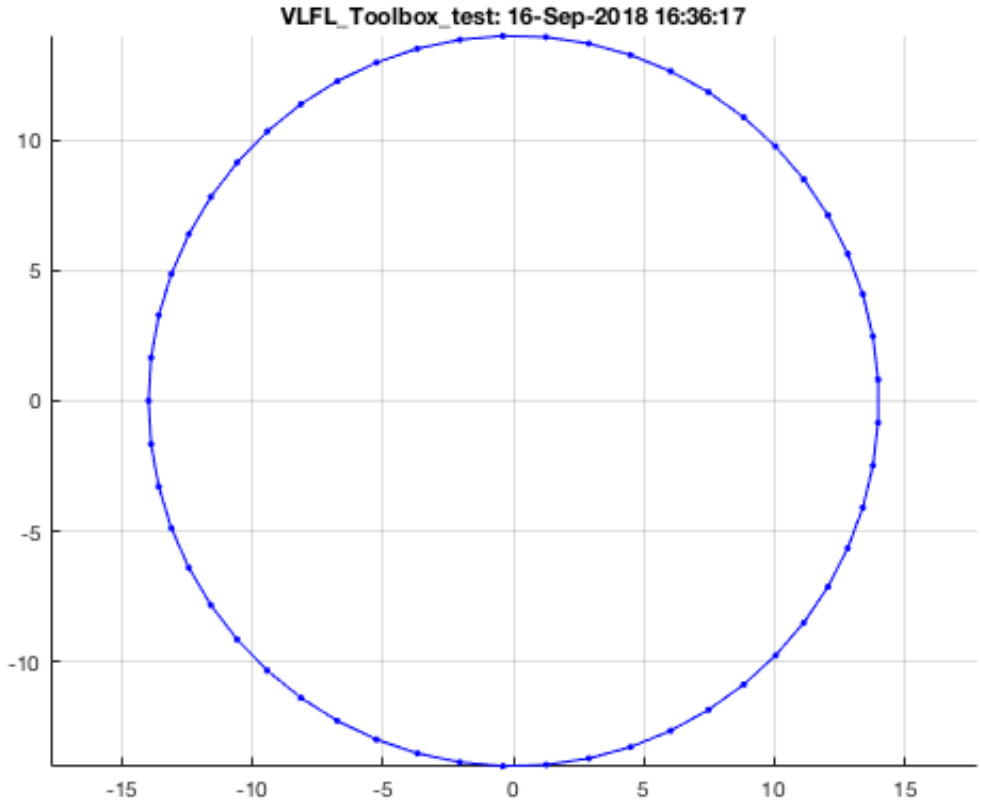
9. Informations on contours inside of others

```
SGfigure; view(0,90);
CPLinsideCPL(PLcircle(9),CPLcopypattern(PLcircle(2),[5 5],[2 2])); axis on;
```



Identical contours are not inside each other

```
CPLinsideCPL(PLcircle(14),CPLsortC(PLcircle(14)));
```



10. Order contours for the sequential plot with a laser cutter

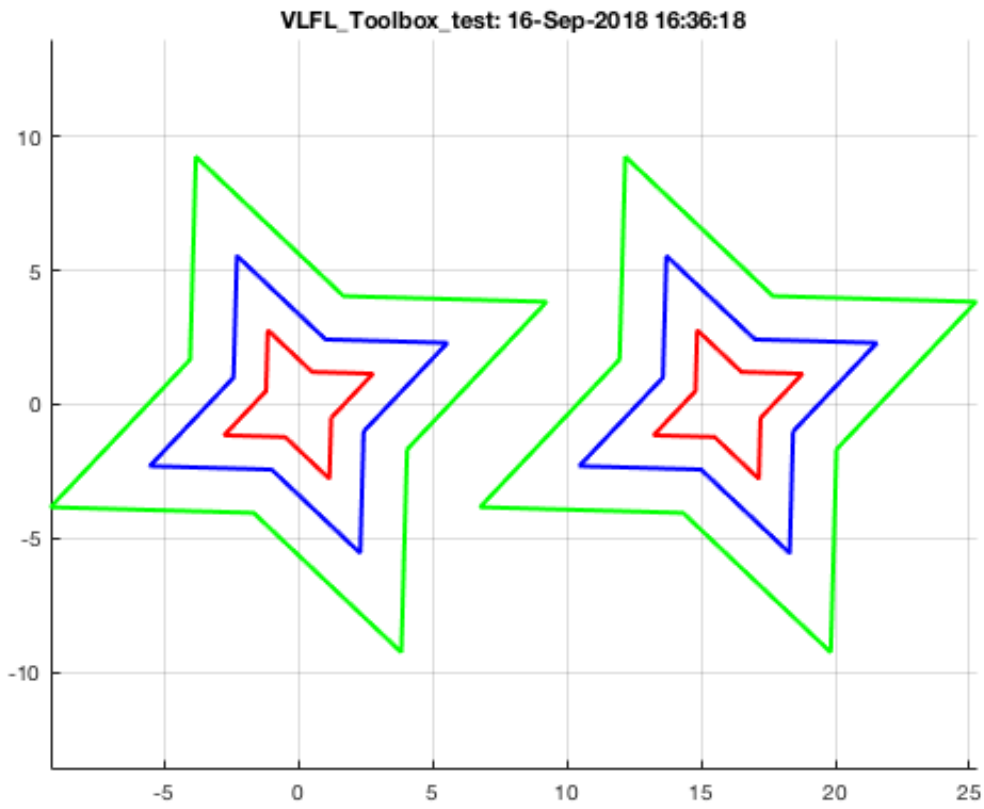
The "level" starts with zero runs from outer to inner. In case of a laser cutter it is necessary to cut the inner contours first.

- ***CPLwriteSVG*** - writes a CPL ans SVG on disk
- ***svgpolylineofCPL*** - plots an SVG file
- ***separateNaN*** - separates CPLs and CVLs
- ***selectNaN*** - creates a new set of selectex CPLs/CVLs
- ***CPLsortinout*** - sorts contours to inner and outer

```
SGfigure; view(0,90);
[ci,CC]=CPLsortinout(CPLsample(14))
CPLsortinout(CPLsample(14));
```

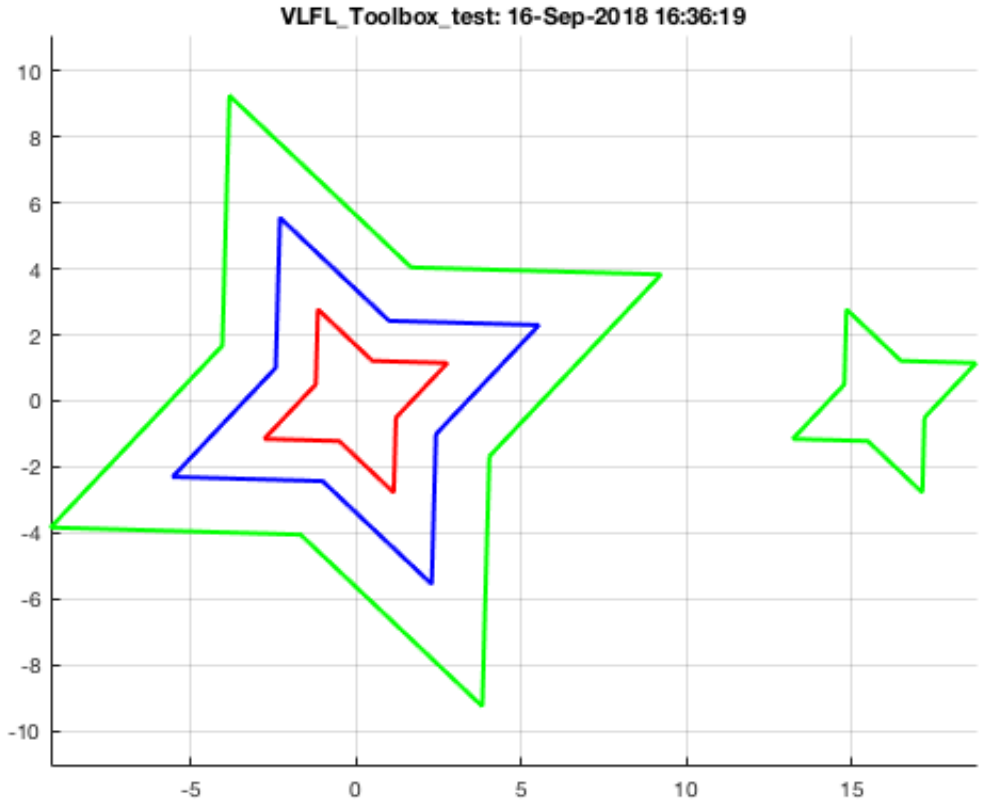
```
ci =

     0
     1
     2
     0
     1
     2


CC =
```

```
   NaN     1     1    -1    -1    -1
    -1   NaN     1    -1    -1    -1
    -1    -1   NaN    -1    -1    -1
    -1    -1    -1   NaN     1     1
    -1    -1    -1    -1   NaN     1
    -1    -1    -1    -1    -1   NaN
```



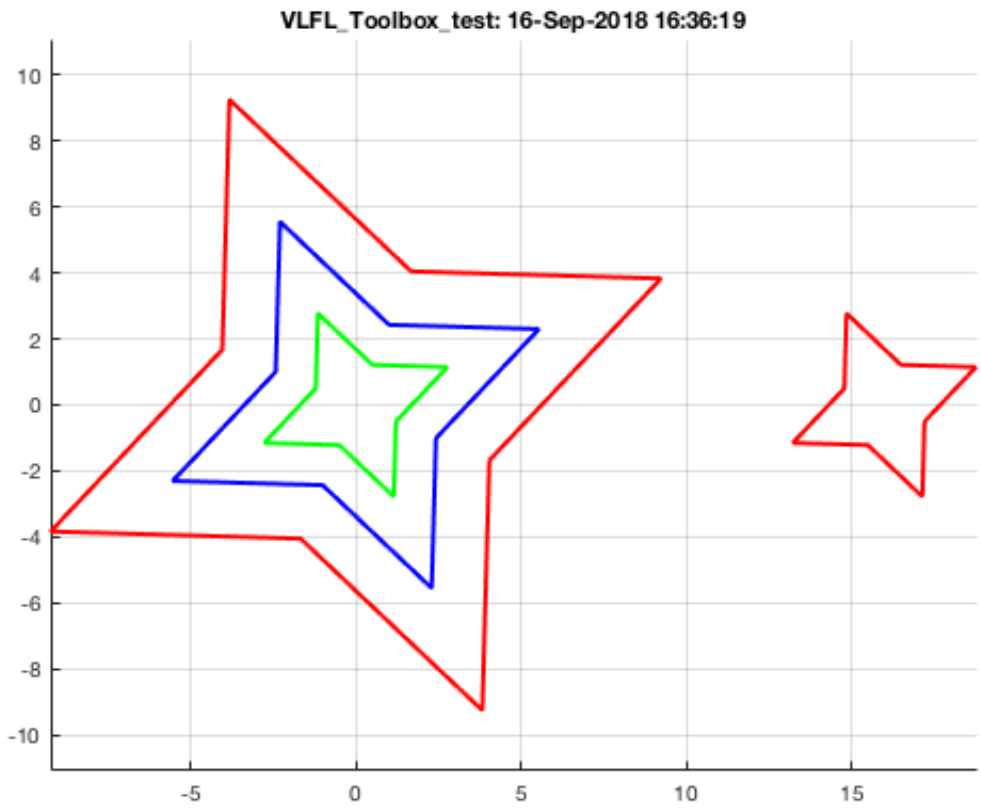
Show a selction from inner to outer for laser cutting

```
CPLsortinout(selectNaN(CPLsample(14),[1,2,3,6]));
```



Now change the order direction from outer to inner

```
CPLsortinout(selectNaN(CPLsample(14),[1,2,3,6]),false);
```



Now write is als a cutter file

```
CPLwriteSVG(CPLsample(14),'VLFL_EXP14_cutter','',true);
```

```
WRITING SVG FILE /Users/timlueth/Desktop/Toolbox_test/VLFL_EXP14_cutter.SVG in ASCII MODE
completed.
```

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:36:19!
Executed 16-Sep-2018 16:36:21 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
=================================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-09-20*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 15: Create a Solid by 2 Closed Polygons

2015-10-03: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.7 required)
- 2. Basics on the creation of a solid between two planar contours in different height
- 3. Solid surface generation and the importance of start point of the contour (turning)
- 4. Solid surface generation and the importance of point assignment strategy
- 5. Solid surface generation for polygons with a small number of points
- 6. Two identical contours with (strictly) monotonic increasing point-center angle
- 7. Two contours of the same shape with different number of points
- 8. Two contours of the similar shape with different number of points
- 9. Two contours of the similar shape with different sum of boundary bending angle sum
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines

- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.7 required)

2. Basics on the creation of a solid between two planar contours in different height

The creation of a solid based on two CPL (each containing exactly ONE closed polygon) with a z-difference have to be solved by different method depending on the contour.

In general, the inner angle sum of a closed polygon that has no overlaps is exactly 360 degree, i.e. 2pi.

So, for convex polygons there is absolutely no problem by **stepping forward related to the strictly monotonic increasing angle sum**, after **finding a suitable start point** of both polygons (which is a challege itself).

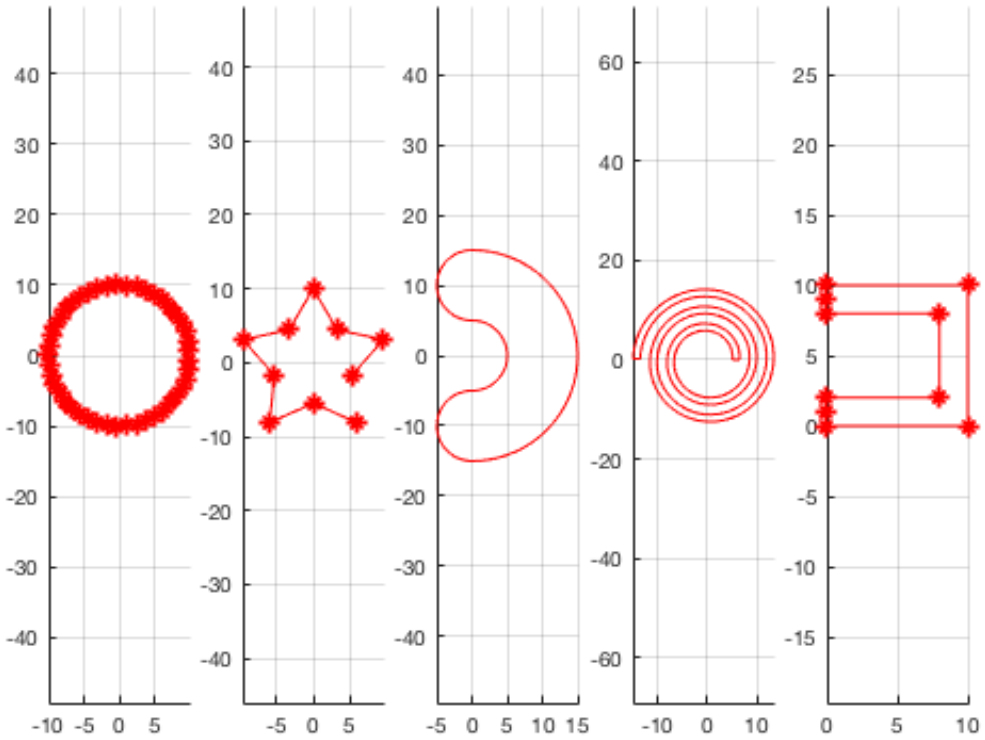
Even **some concave shaped polygons**, such as stars, **have at least monotonic increasing angle sum** and can be connected by this strategy.

Serious challanges exist if we have non monotonic increasing angle sums such as in u-shaped kidneys, or spirals. Here the challange is not only the assignment of points of one contour to a corresponing point on the second, but also how to deal with the starting point if the conturs are rotated.

A challenge is also that the result changes with the order of the input arguments: SGof2CPLz(PLA,PLB) is not the same as (PLB,PLA);

```
SGfigure; view(0,90);
PLU0=[0 0;10 0; 10 10; 0 10; 0 0];
PLU1=[0 0;10 0; 10 10; 0 10; 0 8; 8 8; 8 2; 0 2; 0 0];
PLU2=[0 0;10 0; 10 10; 0 10; 0 9; 0 8; 8 8; 8 2; 0 2; 0 1; 0 0];
subplot(1,5,1); PL=PLcircle(10);PLplot(PL);
view(0,90); grid on; axis equal;
subplot(1,5,2); PL=PLstar(10,10); PLplot(PL);
view(0,90); grid on; axis equal;
```

```
subplot(1,5,3); PL=PLkidney(5,15,pi); CPLplot(PL,'r-');
view(0,90); grid on; axis equal;
subplot(1,5,4); PL=CPLspiral(5,15,5*pi); CPLplot(PL,'r-');
view(0,90); grid on; axis equal;
subplot(1,5,5); PL=CPLspiral(5,15,5*pi); CPLplot(PLU2,'r*-');
view(0,90); grid on; axis equal;
```

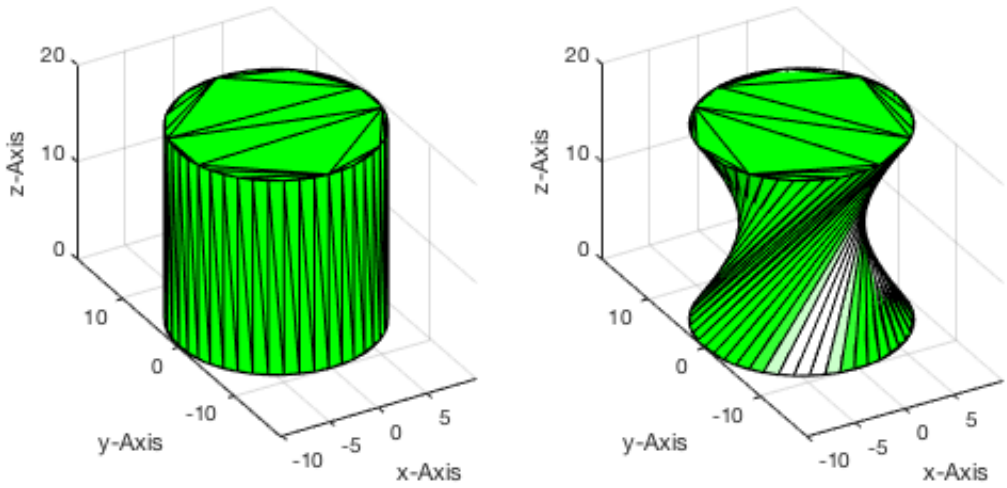


3. Solid surface generation and the importance of start point of the contour (turning)

If two identical contours are assigned, the turning angle around the center is of importance. Already a small turning angle, known or unkown, results in a different shape. For solid generation we use the function:

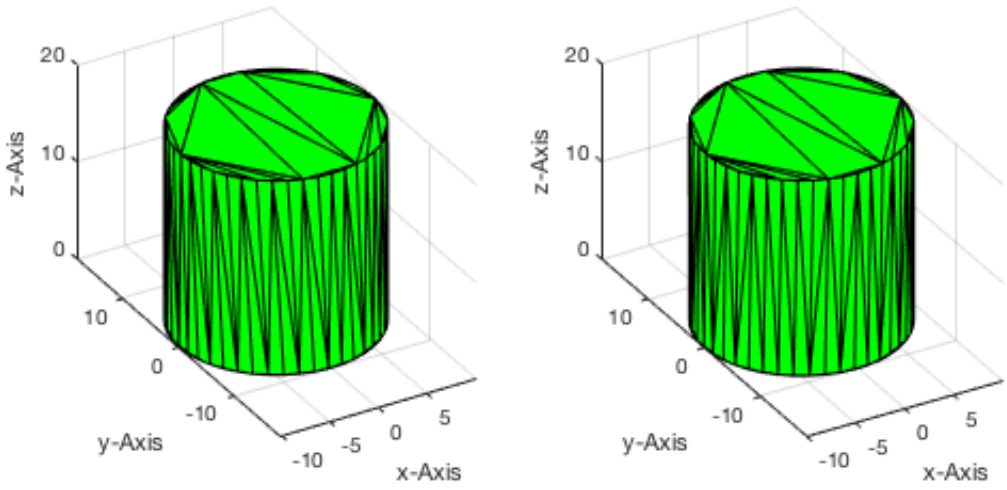
SGof2CPLz(CPLA,CPLB,z) - creates a solid between two plane contours in height 0 and z

```
SGfigure; view(-30,30);
subplot(1,2,1);
SG=SGof2CPLz(PLcircle(10),PLcircle(10),20); SGplot(SG,'g'); view(-30,30);
subplot(1,2,2);
SG=SGof2CPLz(PLcircle(10),PLtransR(PLcircle(10),rotdeg(120)),20,[],'none');
SGplot(SG,'g'); view(-30,30);
```



One input paramter of SGof2CPLz allows to select the turning angle adjustment to 'none', 'rot', or 'miny'. Please read the documentation of 'czero'=rot (minimal angle near zero of the convex hull) of CPLsortC and PLminyx (Point with the minimal y and minmal x value) to understand 'miny'. In addition it is also possible directly to give the assignment of the first points directly by an 1x2 circshift value [1 1] == 'none'

```
SGfigure; view(-30,30);
subplot(1,2,1);
SG=SGof2CPLz(PLcircle(10),PLtransR(PLcircle(10),rotdeg(90)),20,[],'rot');
SGplot(SG,'g'); view(-30,30);
subplot(1,2,2);
SG=SGof2CPLz(PLcircle(10),PLtransR(PLcircle(10),rotdeg(90)),20,[],'miny');
SGplot(SG,'g'); view(-30,30);
```



Turning adjustment 'miny' is the default method for turning both contours! Even if the contour is turned for point assignment, the final order of the points is the same as the original order! This is important for generating tubes based on this function.

4. Solid surface generation and the importance of point assignment strategy

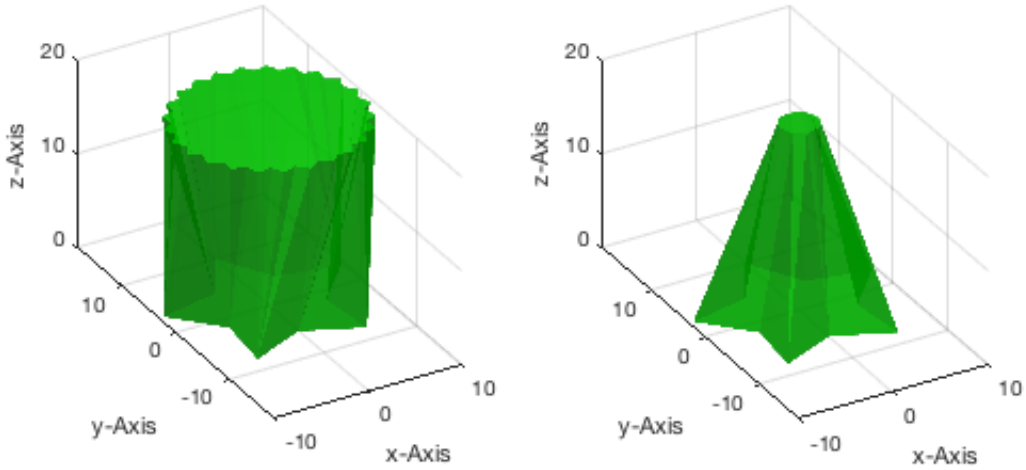
Currently, there are four strategies for the contour point assignment during contour connections:
SGof2CPLZ(PLA,PLB,z,assignment,turning);

- **'number' assignment** based on point index / sum(points) - works well for identical contours with different point numbers. Use in combination with both 'miny' or 'rot'.
- **'length' assignment** based on edge length / sum(edge length) - works well for identical contours (shrinked,grown, different sampling). Use in combination with 'miny' or 'rot', the later especially if the number of points is very large.
- **'angle' assignment** based on abs(edge angle) / sum(abs(edge angle)) - required if the sum(abs(edge angle)) differs remarkable. Use in combination with 'rot' and not with 'miny'.
- **'center' assignment** based on angle between center and point - should not be used anymore.

- **In most cases 'length' and 'miny' works well**, wich are the default values
- **For heavy curved contours use 'angle' and 'rot'**

```
SGfigure; view(-30,30);
subplot(1,2,1);
SG=SGof2CPLz(PLstar(10,10),PLstar(10,50),20); SGplot(SG,'g'); view(-30,30);
VLFLplotlight(1,0.7);
subplot(1,2,2);
SG=SGof2CPLz(PLstar(10,10),PLcircle(2),20); SGplot(SG,'g'); view(-30,30);
```

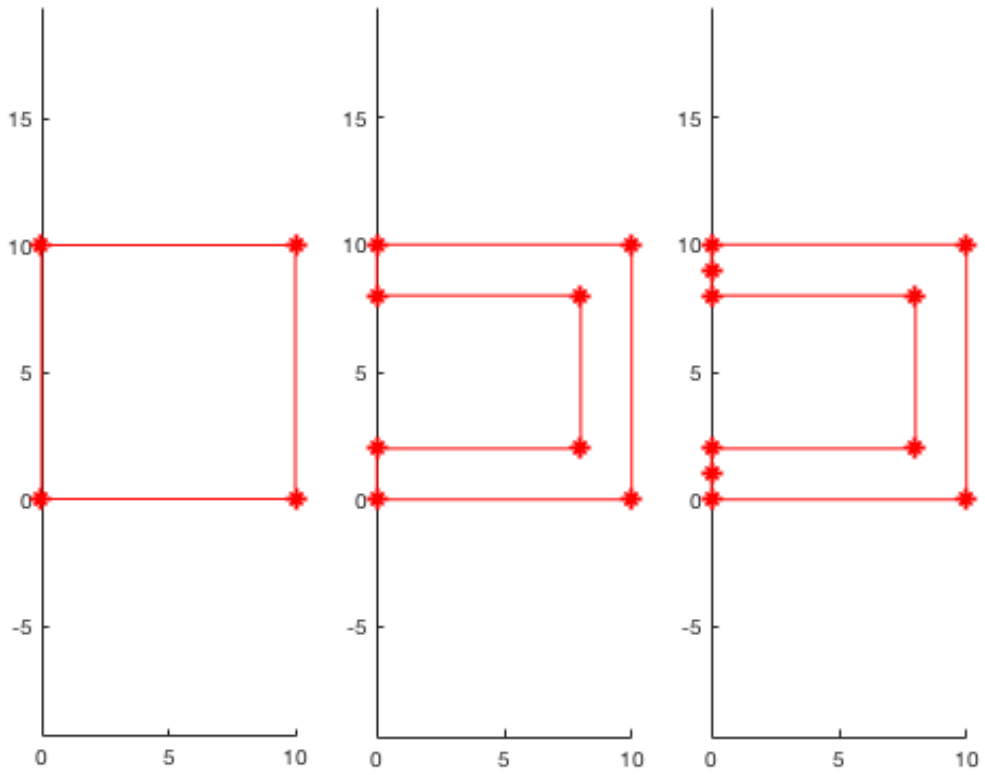
```
VLFLplotlight(1,0.7);
```



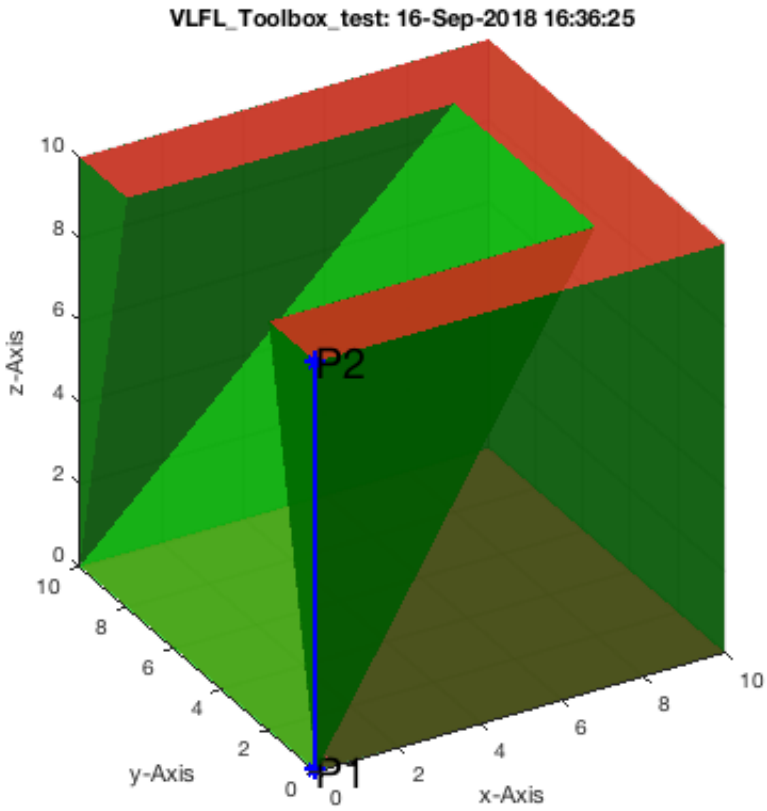
5. Solid surface generation for polygons with a small number of points

For polygon of only a few point, typically, the user has clear expectations about the final result of the solid.

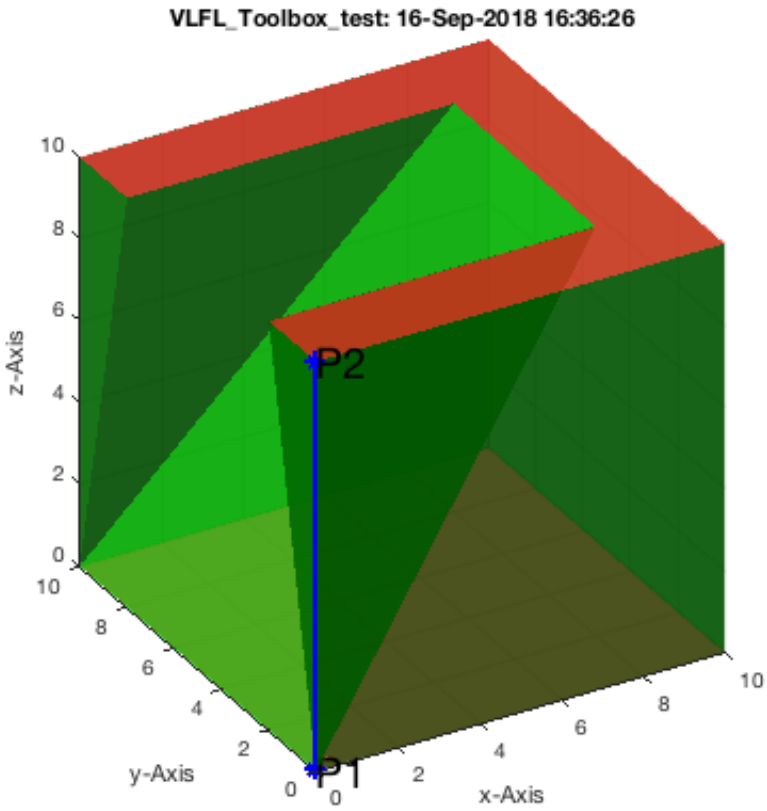
```
SGfigure;
PLU0=[0 0;10 0; 10 10; 0 10; 0 0];
PLU1=[0 0;10 0; 10 10; 0 10; 0 8; 8 8; 8 2; 0 2; 0 0];
PLU2=[0 0;10 0; 10 10; 0 10; 0 9; 0 8; 8 8; 8 2; 0 2; 0 1; 0 0];
subplot(1,3,1); CPLplot(PLU0);  view(0,90); axis equal;
subplot(1,3,2); CPLplot(PLU1);  view(0,90); axis equal
subplot(1,3,3); CPLplot(PLU2);  view(0,90); axis equal
```



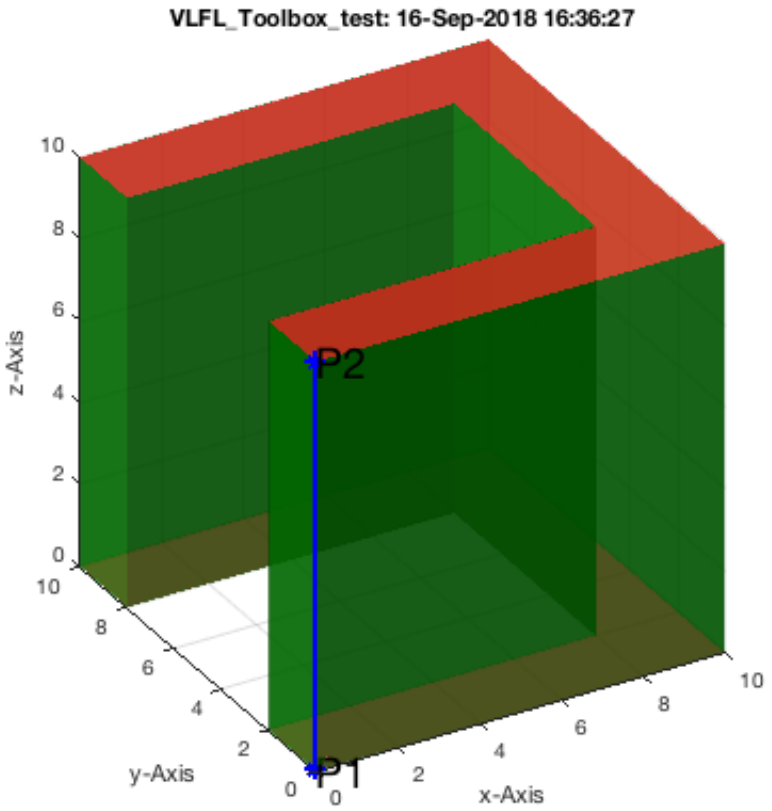

```
SGof2CPLz(PLU0,PLU2,10); VLFLplotlight(1,0.7);
```




```
SGof2CPLz(PLU0,PLU1,10); VLFLplotlight(1,0.7);
```



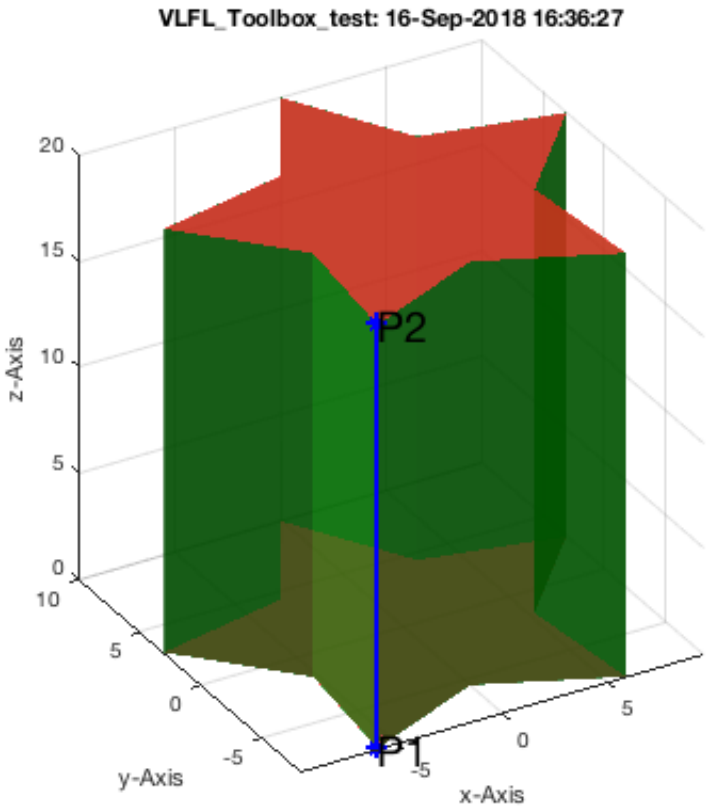

```
SGof2CPLz(PLU1,PLU2,10); VLFLplotlight(1,0.7);
```



6. Two identical contours with (strictly) monotonic increasing point-center angle

In case of two identical polygons that are just shifted or rotated, the default values 'length' and 'miny' work almost always perfectly.

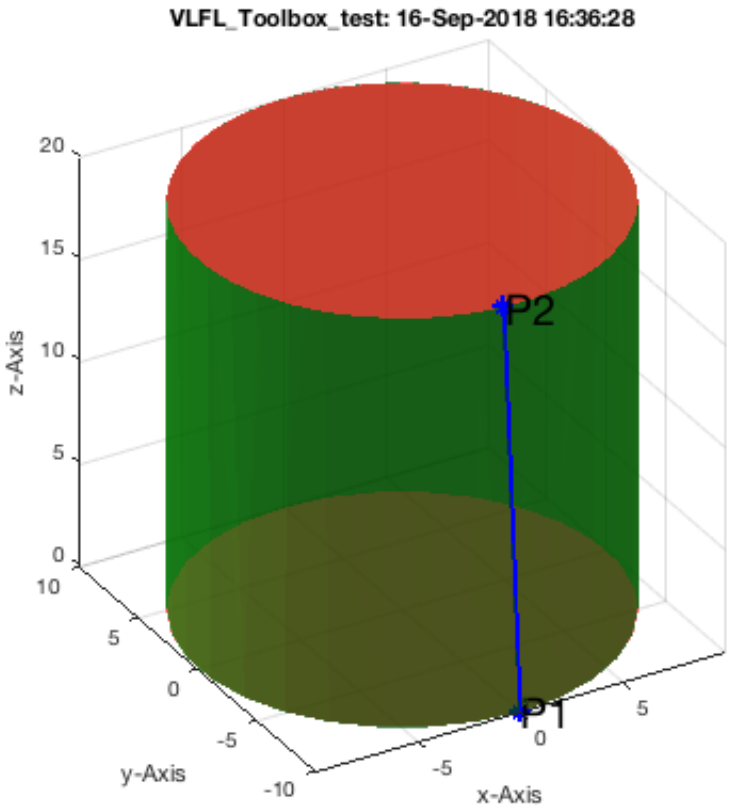
```
SGof2CPLz(PLstar(10,10),PLstar(10,10),20); VLFLplotlight(1,0.7);
```



7. Two contours of the same shape with different number of points

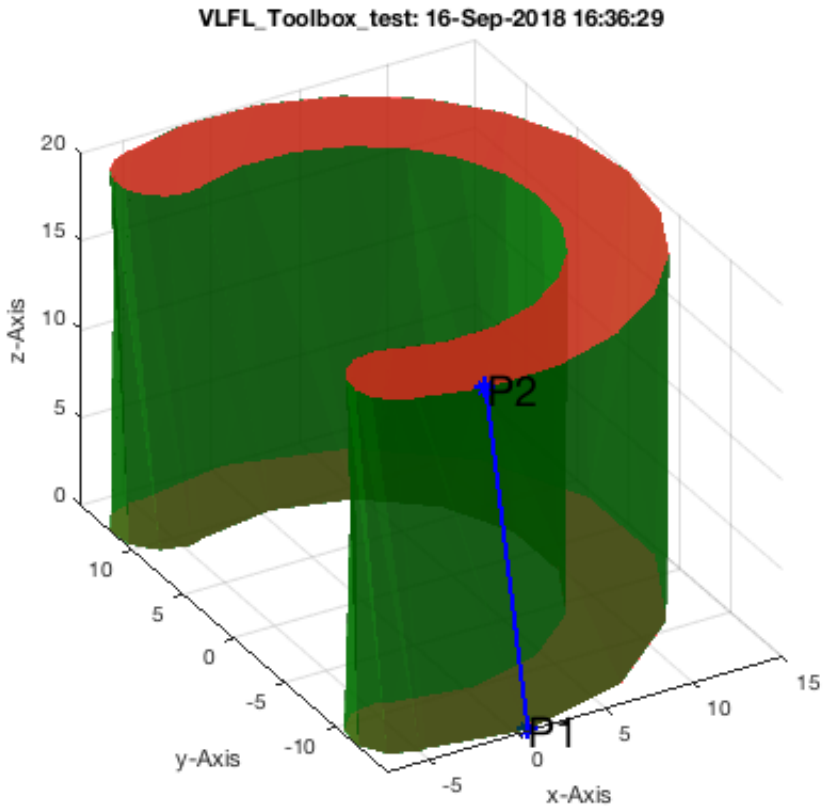
In case that there are two identically shaped contours but with a different number of points, 'number' would be a solution but the default values 'length' and 'miny' work almost always perfectly. Even in the case of u-shaped contour.

```
SGof2CPLz(PLcircle(10,30),PLcircle(10,40),20);
VLFLplotlight(1,0.7);
```




```
SGof2CPLz(PLkidney(10,15,pi/0.8,10),PLkidney(10,15,pi/0.8,15),20);
```

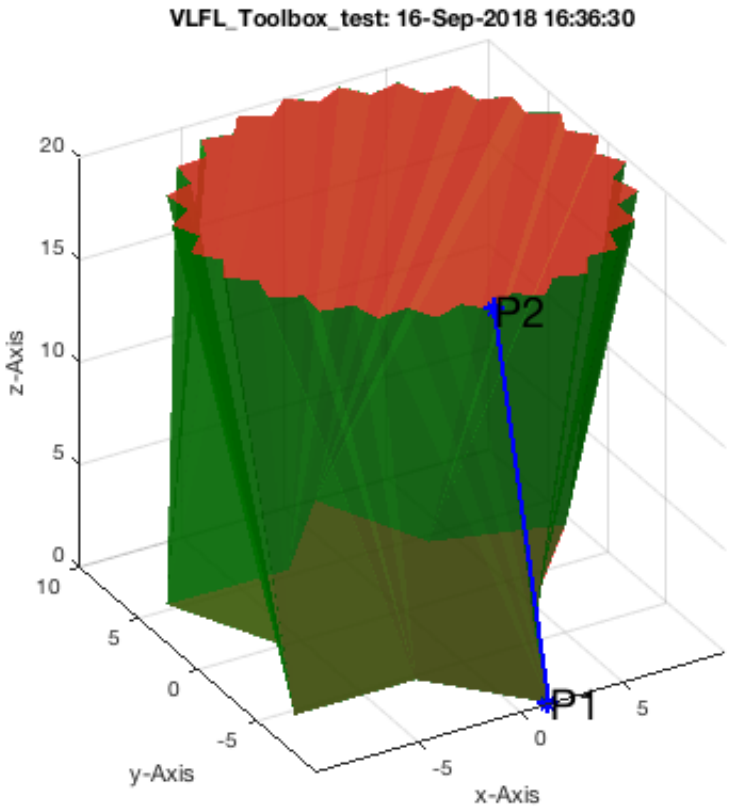
```
VLFLplotlight(1,0.7);
```



8. Two contours of the similar shape with different number of points

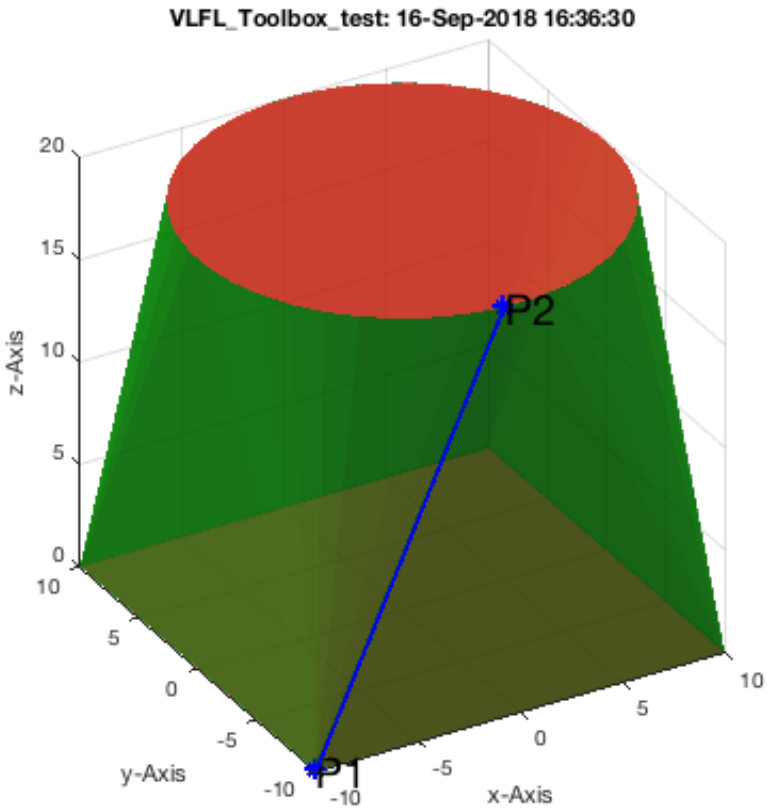
In case that there are two similar not indentical contours and with a different number of points, again the default values 'length' and 'miny' work almost always perfectly. Even in the case of u-shaped contour.

```
SGof2CPLz(PLstar(10,11),PLstar(10,50),20); VLFLplotlight(1,0.7);
```

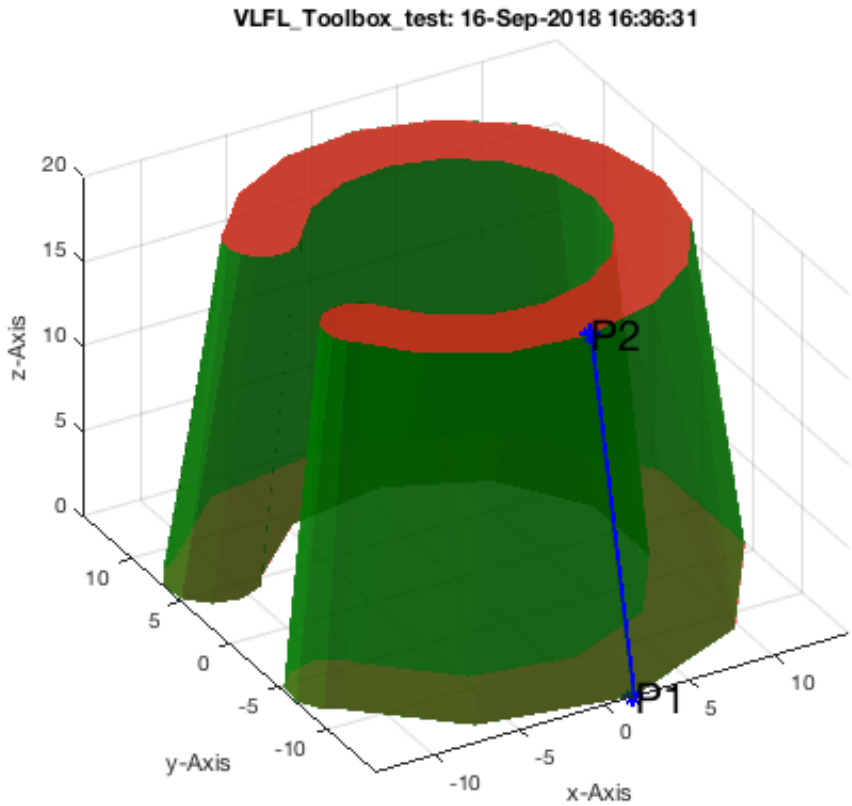



```
SGof2CPLz(PLcircle(10*sqrt(2),4),PLcircle(10,40),20);
```

```
VLFLplotlight(1,0.7);
```




```
SGof2CPLz(PLkidney(10,15,pi/0.6,10),PLkidney(8,12,pi/0.6,15),20);
VLFLplotlight(1,0.7);
```

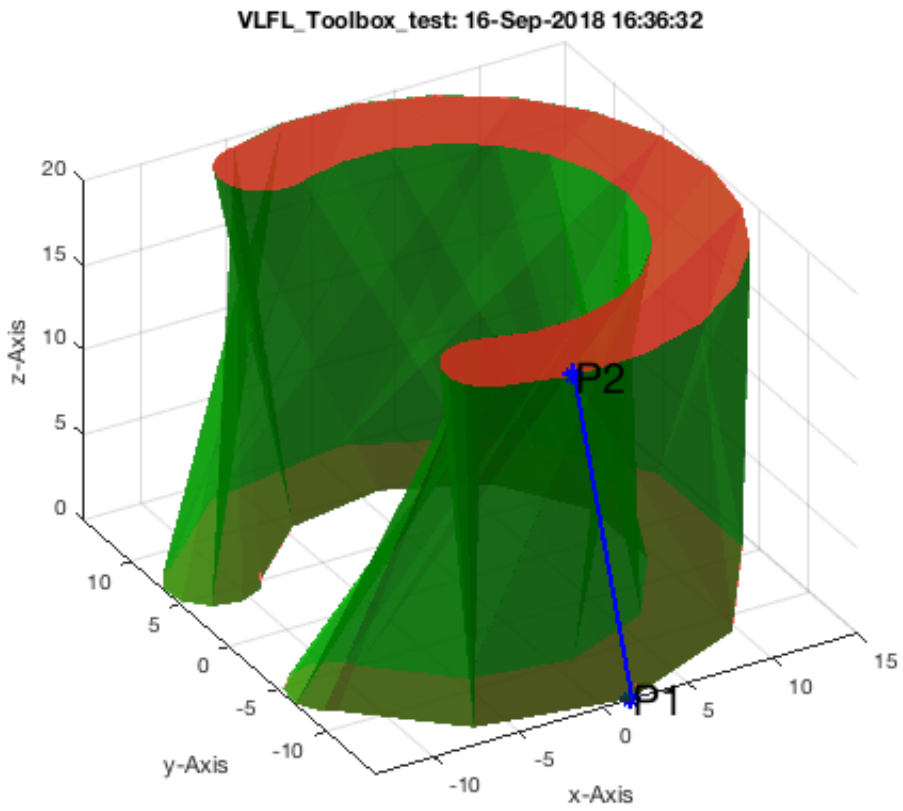


In this case we see that the kidney has different size but the same bending angle of pi/.6.

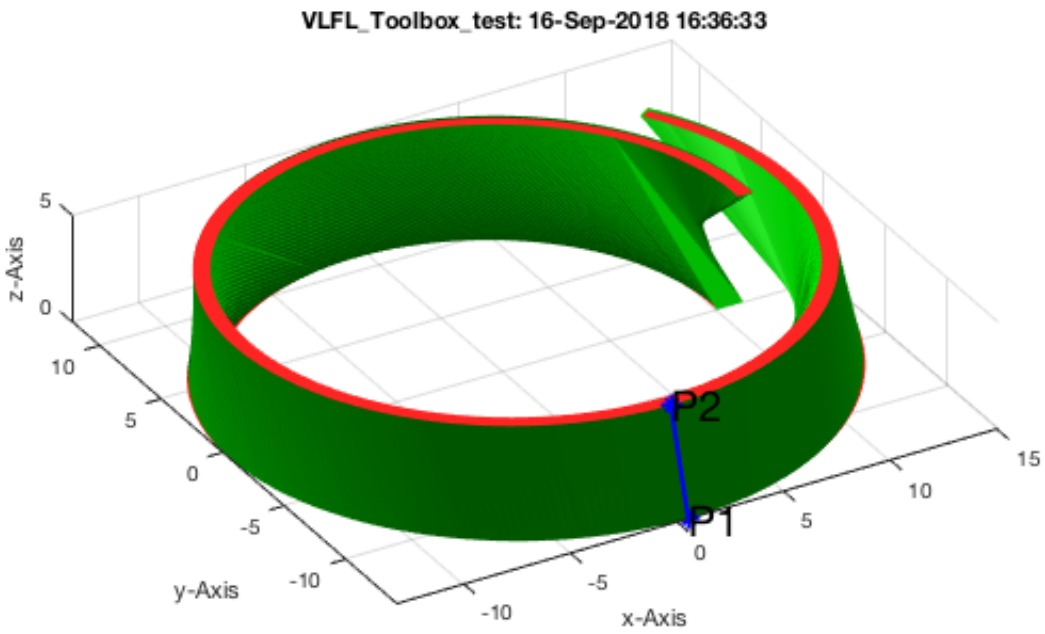
9. Two contours of the similar shape with different sum of boundary bending angle sum

In case that the boundary bending angle is greated than 2*pi (360 degree), the assignment by length and the turning strategy of miny is not the best anymore.

```
SGof2CPLz(PLkidney(10,15,pi/0.6,10),PLkidney(10,15,pi/0.8,15),20);
VLFLplotlight(1,0.7);
```

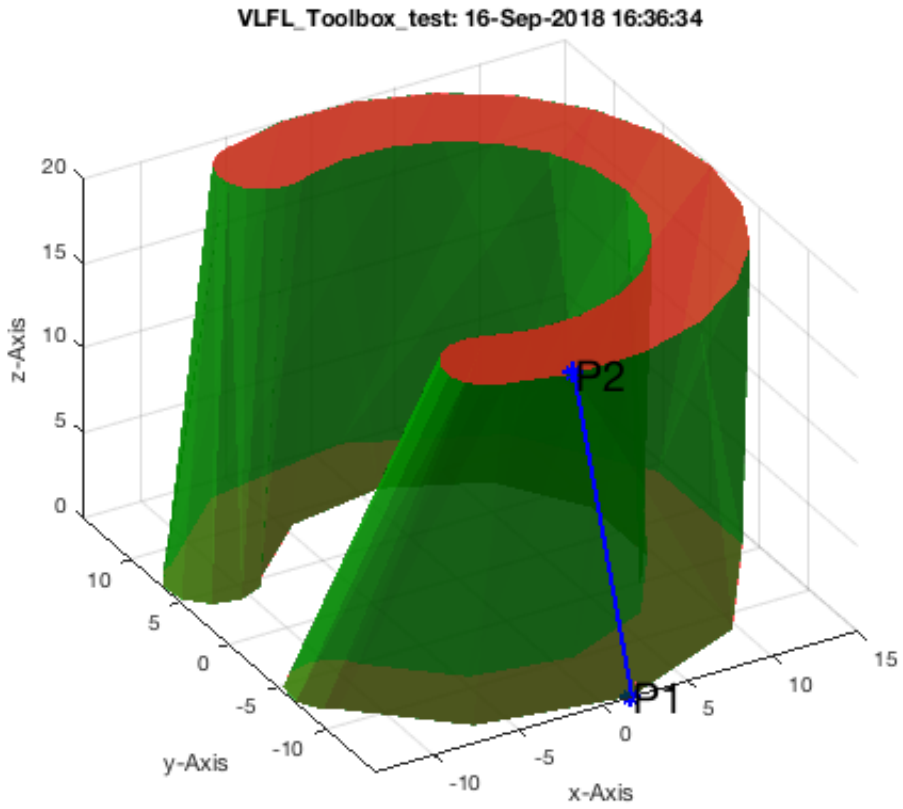



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5);
VLFLplotlight (1,1);
```

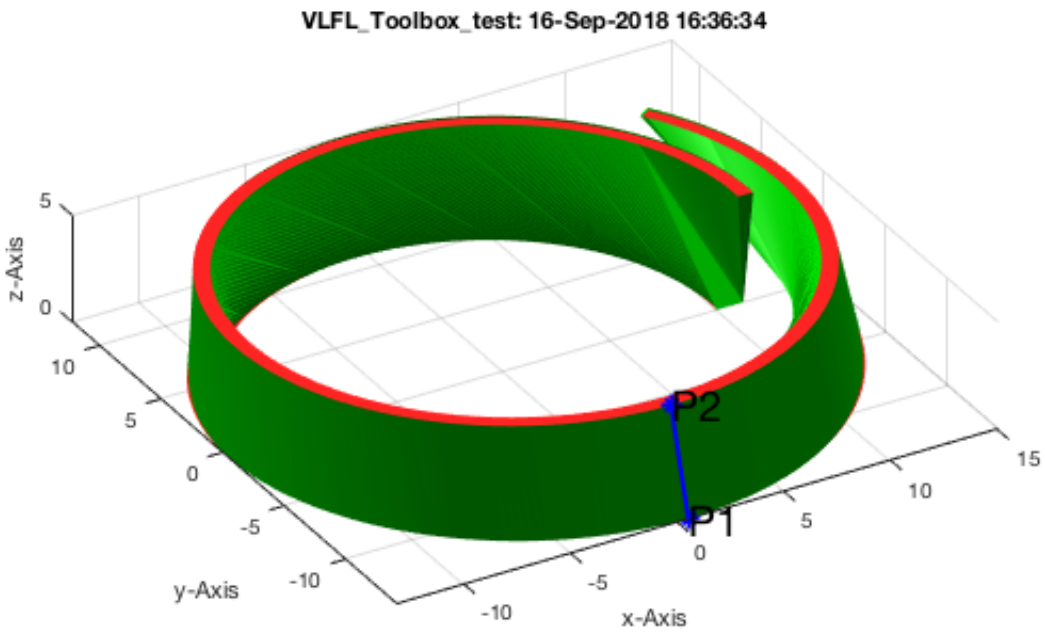


In this case we see malformed solids.

```
SGof2CPLz(PLkidney(10,15,pi/0.6,10),PLkidney(10,15,pi/0.8,15),20,'angle');
VLFLplotlight(1,0.7);
```

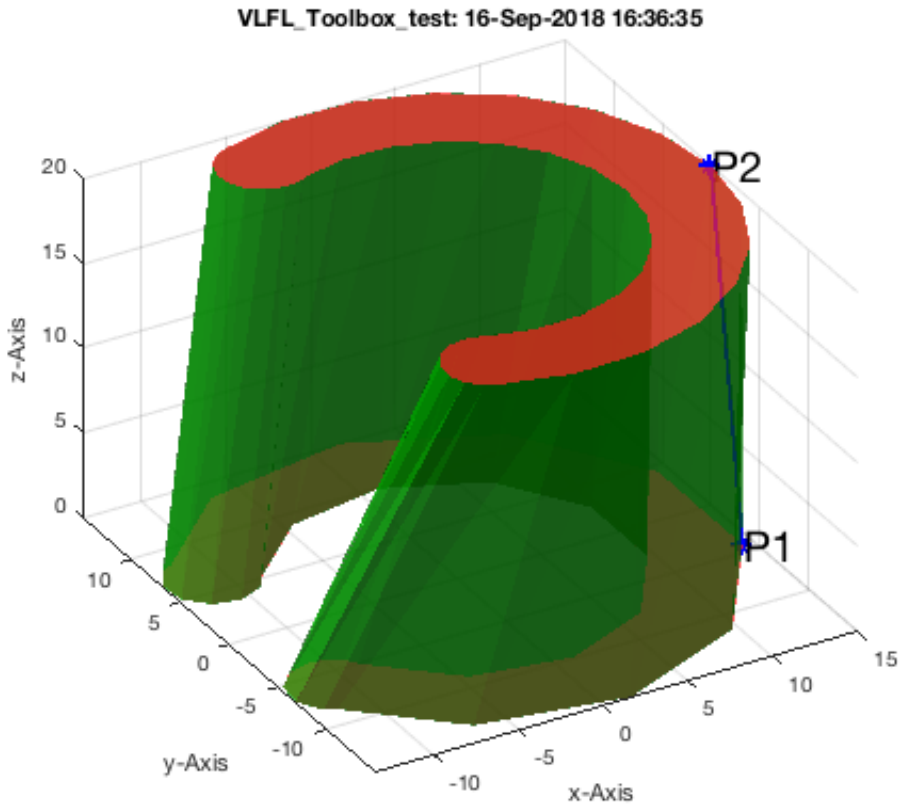



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5,'angle');
VLFLplotlight (1,1);
```

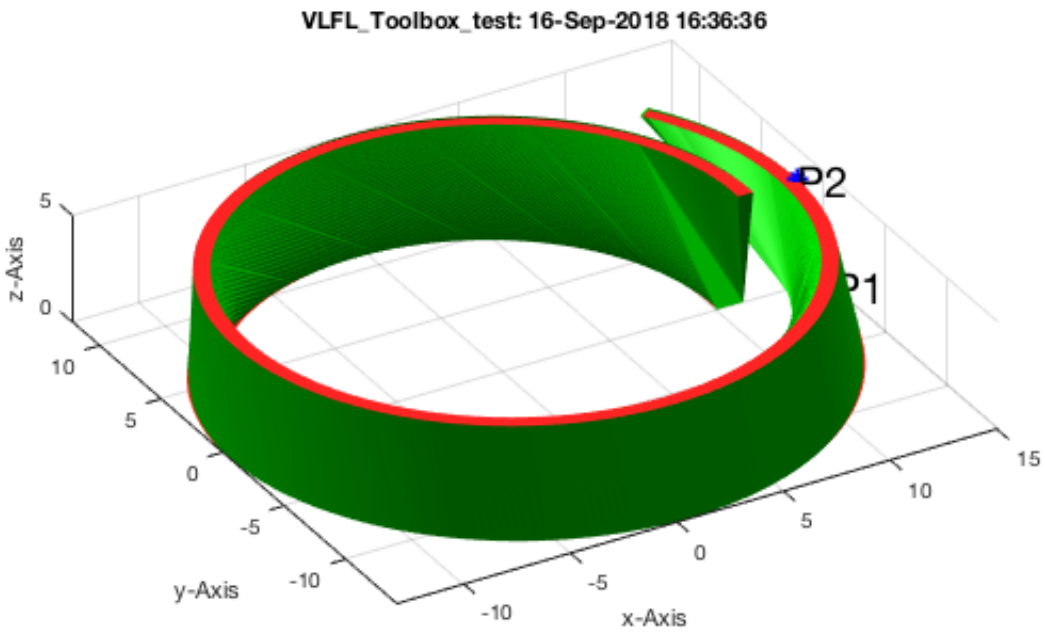


By using 'angle' instead of 'length', the solids are more what we expected.

```
SGof2CPLz(PLkidney(10,15,pi/0.6,10),PLkidney(10,15,pi/0.8,15),20,'angle','rot');
VLFLplotlight(1,0.7);
```

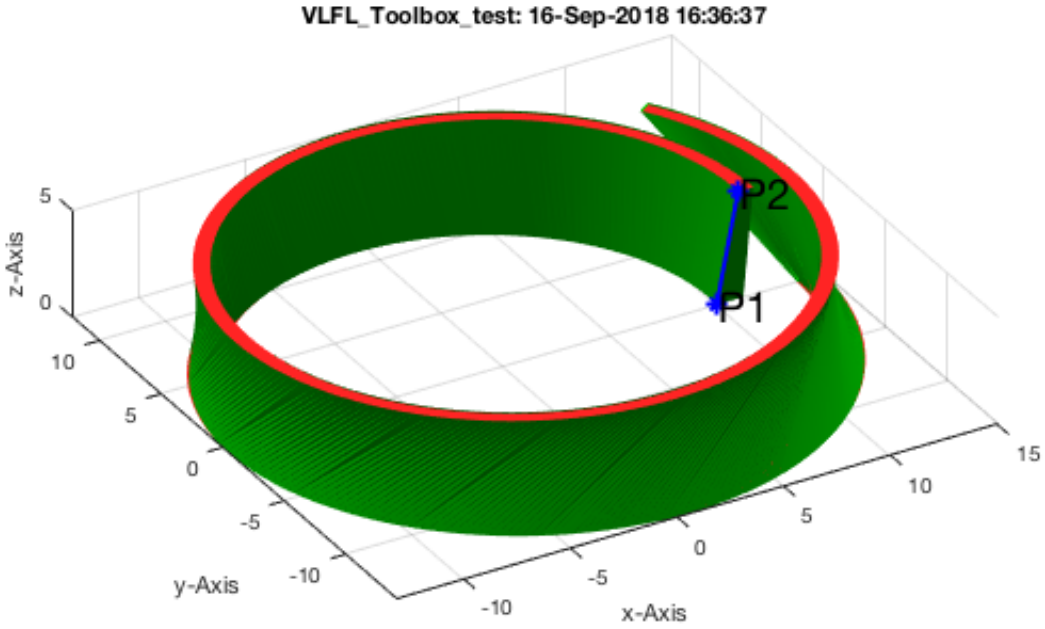



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5,'angle','rot');
VLFLplotlight (1,1);
```



By using 'angle' instead of 'length' AND an explicitly given 1st point assignment, the best result can be achieved.

```
PLA=CPLspiral(10,15,2*pi); PLB=CPLspiral(11,14,2.2*pi);
SGof2CPLz(PLA,PLB,5,'angle',[size(PLA,1) size(PLB,1)]); VLFLplotlight (1,1);
```



By using 'angle' instead of 'length' AND 'rot' instead of 'miny', the solids are almost what we expected.

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:36:37!
Executed 16-Sep-2018 16:36:39 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simulink
=================================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-10-03*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 16: Create Tube-Style Solids by Succeeding Polygons

2015-10-04: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 2.7 required)
- 2. Tube generation by repeating identical CPL along a 3D path
- 3. Tube generation using a 3D path with Bezier-curves or radial edges
- 4. Creating solids by closed polygons in different height: z-coordinate
- 5. Creating a sphere with minmal number of points
- 6. Creating a spherical joint
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements

- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 2.7 required)

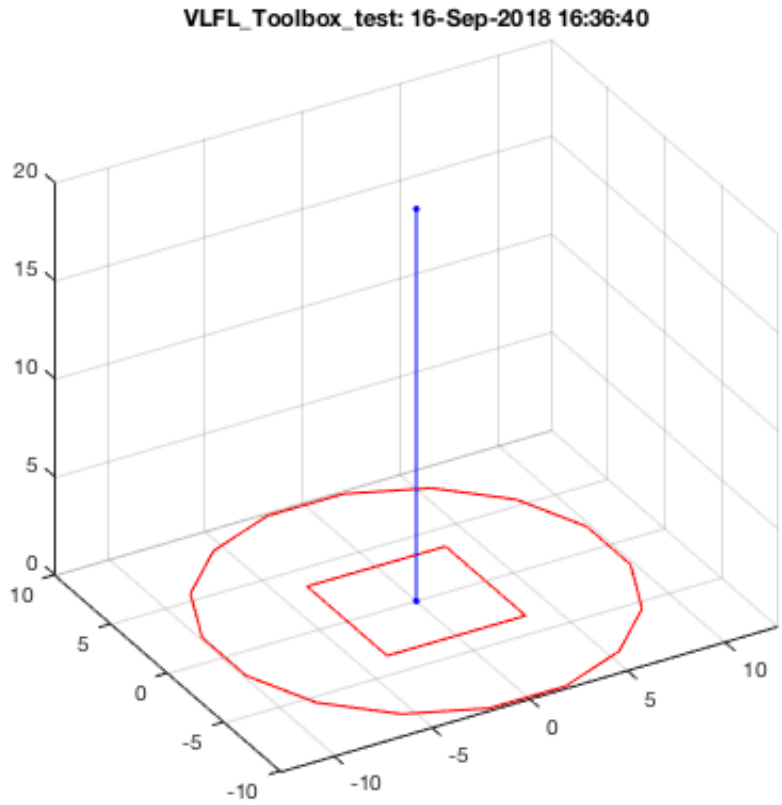
2. Tube generation by repeating identical CPL along a 3D path

For robotics design,very often we have a wish to extrude a CPL not only in an orthognal z direction to the xy-plane, but in an any desired direction even along a path in 3D space. Intutively we expect a result, but this is not easy to achieve automatically. Anyway, for those tasks we have two functions:

- SGcontourtube - repeats a CPL along a path in 3D
- FLofCVL

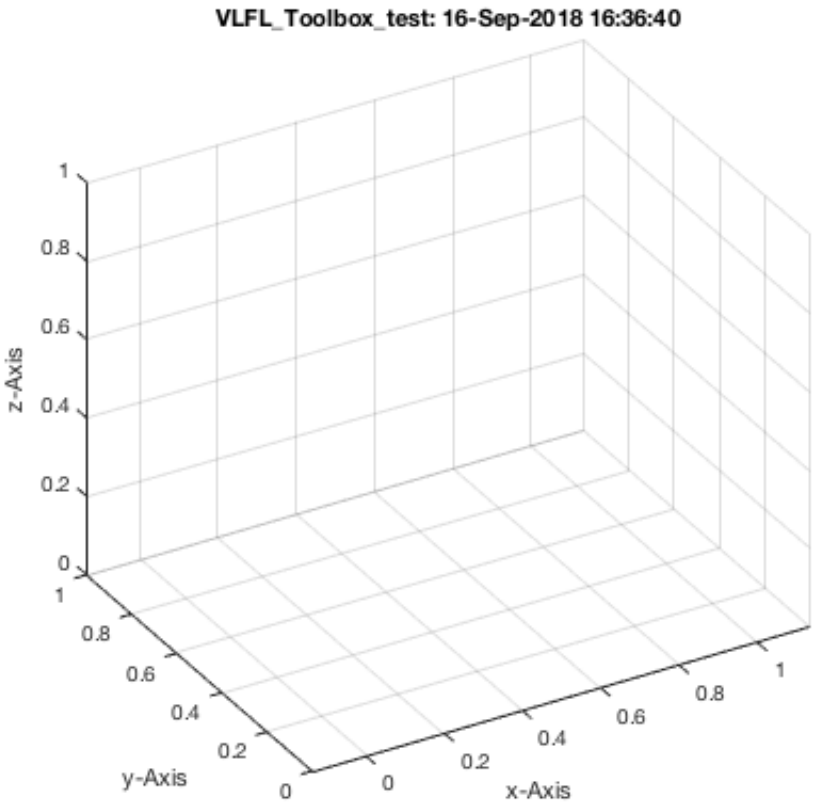
Let us start with a planar contour in the x/y-plane, and an orthogonal path

```
SGfigure; axis on ; grid on;
CPL=CPLsample(8);
CPLplot(CPL,'r-');

VL=[0 0 0; 0 0 20];
VLplot(VL,'b.-'); view(-30,30);
```



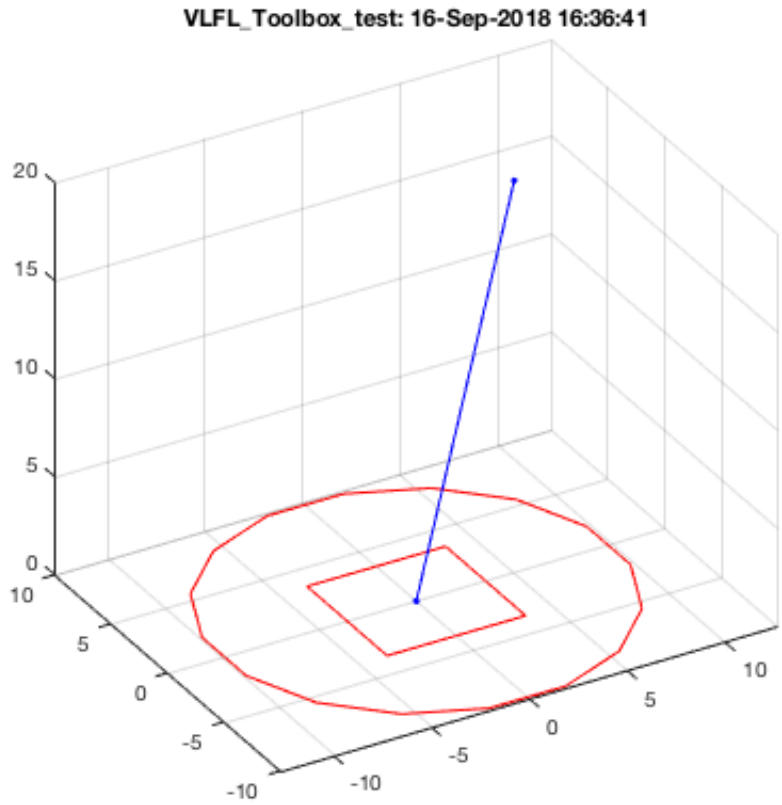
The final result looks as expected

```
SG=SGcontourtube(CPL,VL); SGfigure(SG); VLFLplotlight(1,1);view (-30,30);
```



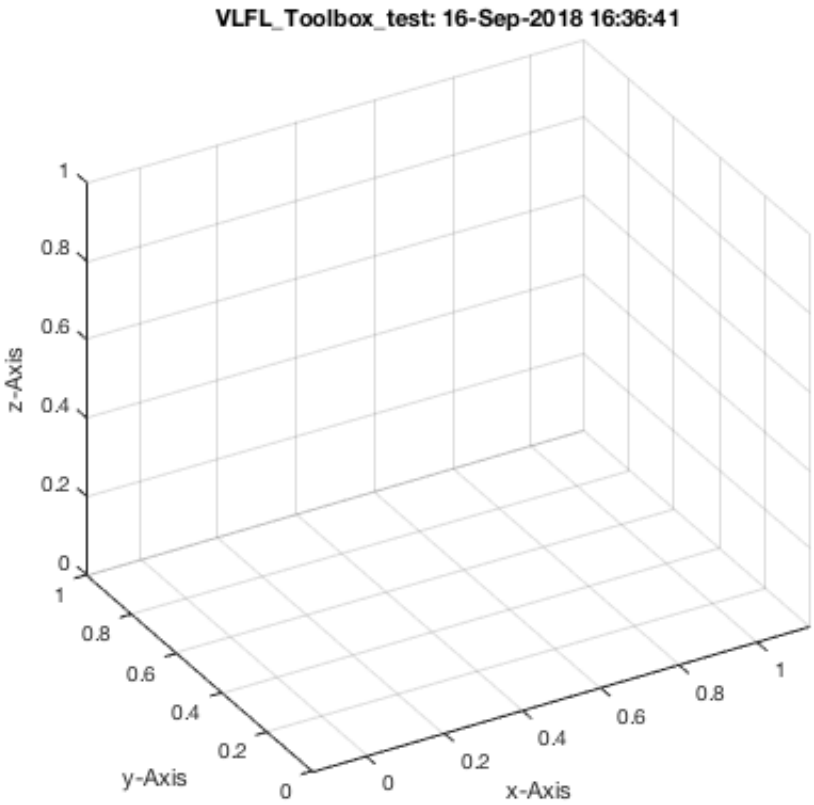
Now we change the path a little

```
SGfigure;  axis on ; grid on;
CPLplot(CPL,'r-');
VL=[0 0 0; 5 0 20];
VLplot(VL,'b.-'); view(-30,30);
```



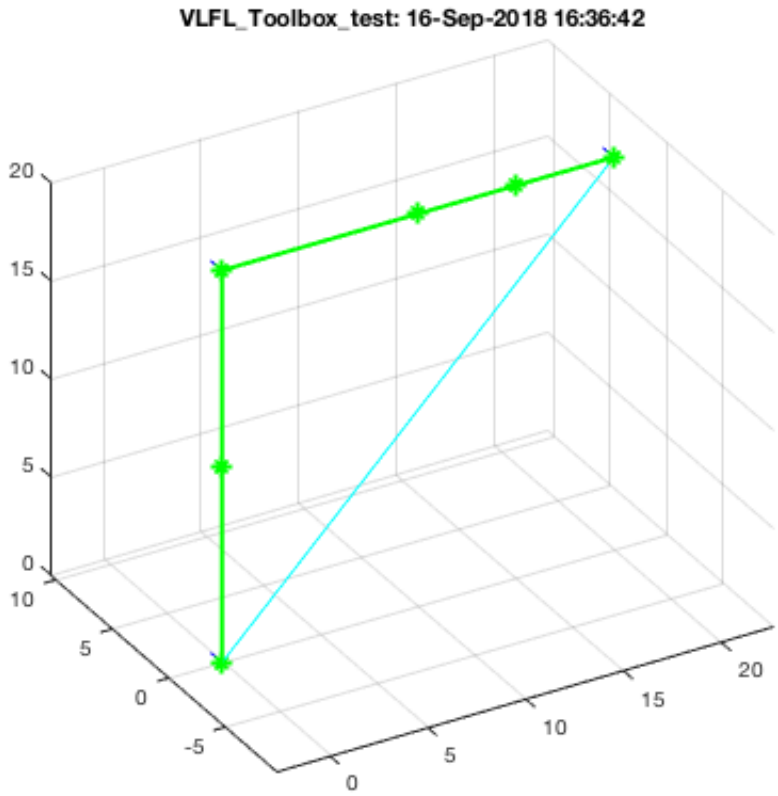
The final result may not look as expected

```
SG=SGcontourtube(CPL,VL);
SGfigure; axis on ; grid on;
SGplot(SG,'m');
VLFLplotlight(1,1);view (-30,30);
```



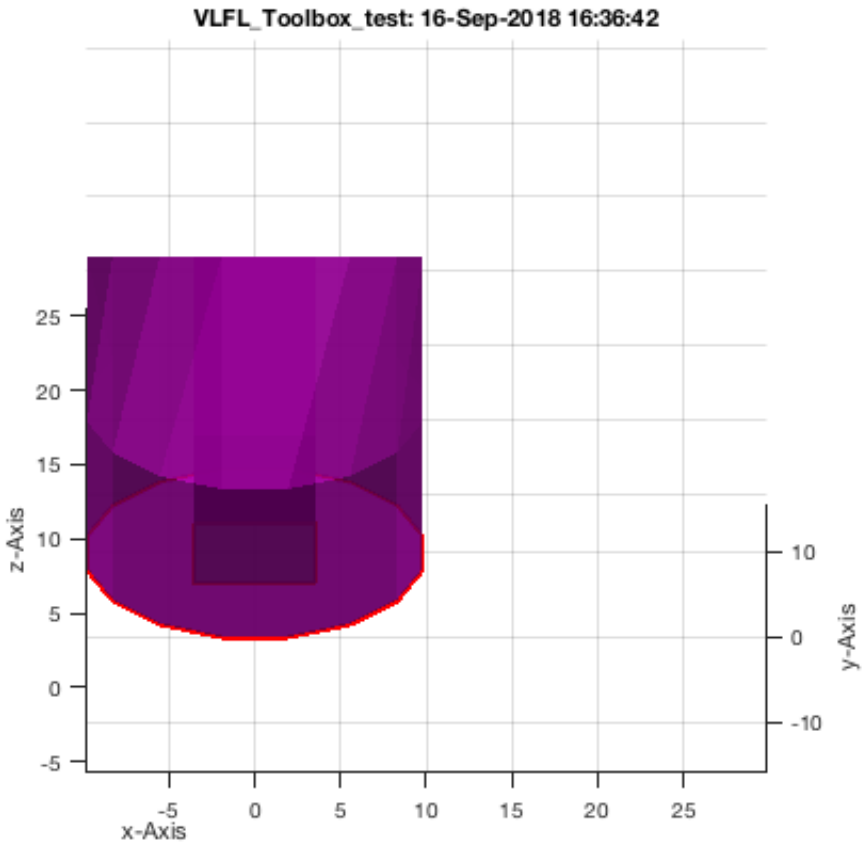
Now we change the path, still a planar one and anylyze the angle situation The cynan colored path explain that the vertex list has to be closed to analyze the first an last angle. Furthermore we see that more than one point has no defined orthogonal vector since it sits on a straight line

```
SGfigure; CPLplot(CPL,'r-');  axis on ; grid on;
VL=[0 0 0; 0 0 10; 0 0 20; 10 0 20; 15 0 20; 20 0 20];
VLangle(VL);
```



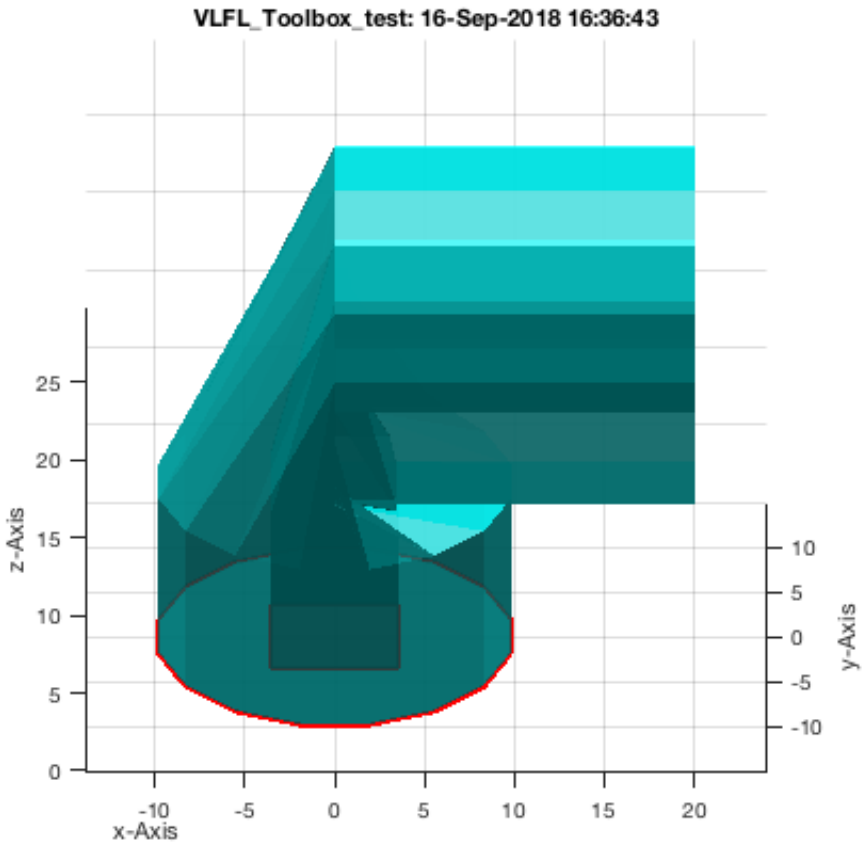
The result is not may be we expected

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGcontourtube(CPL,VL); SGplot(SG,'m'); VLFLplotlight(1,0.8);view (0,30);
```

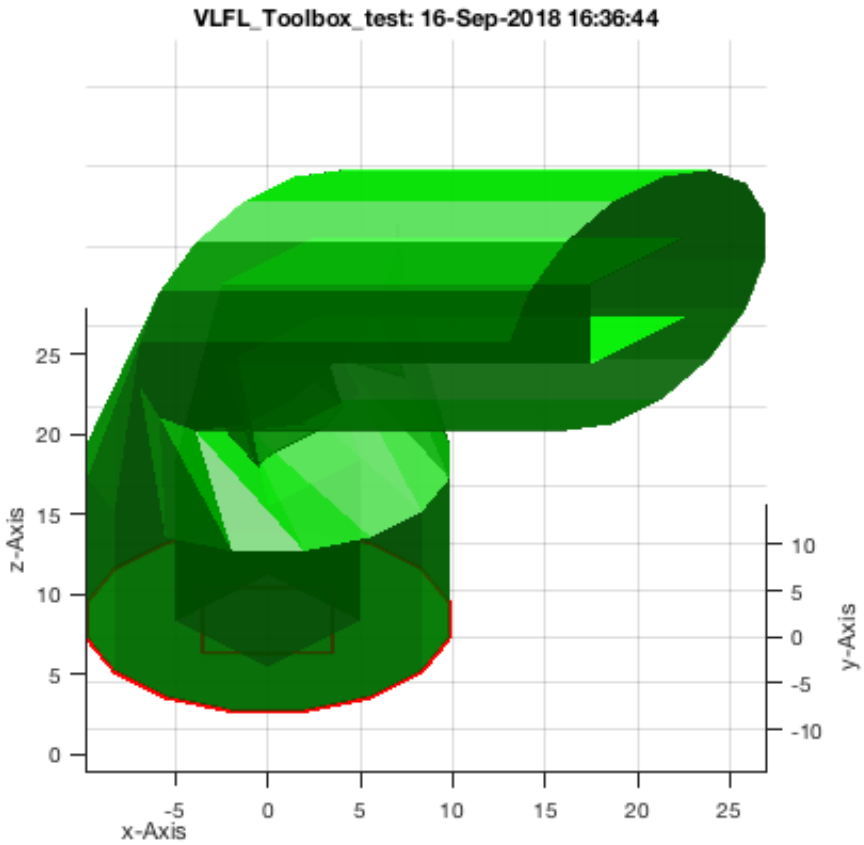


The result can be adjusted by defining the ex-vector at the start point

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGcontourtube(CPL,VL,[0 1 0]); SGplot(SG,'c'); VLFLplotlight(1,0.8);view (0,30);
```



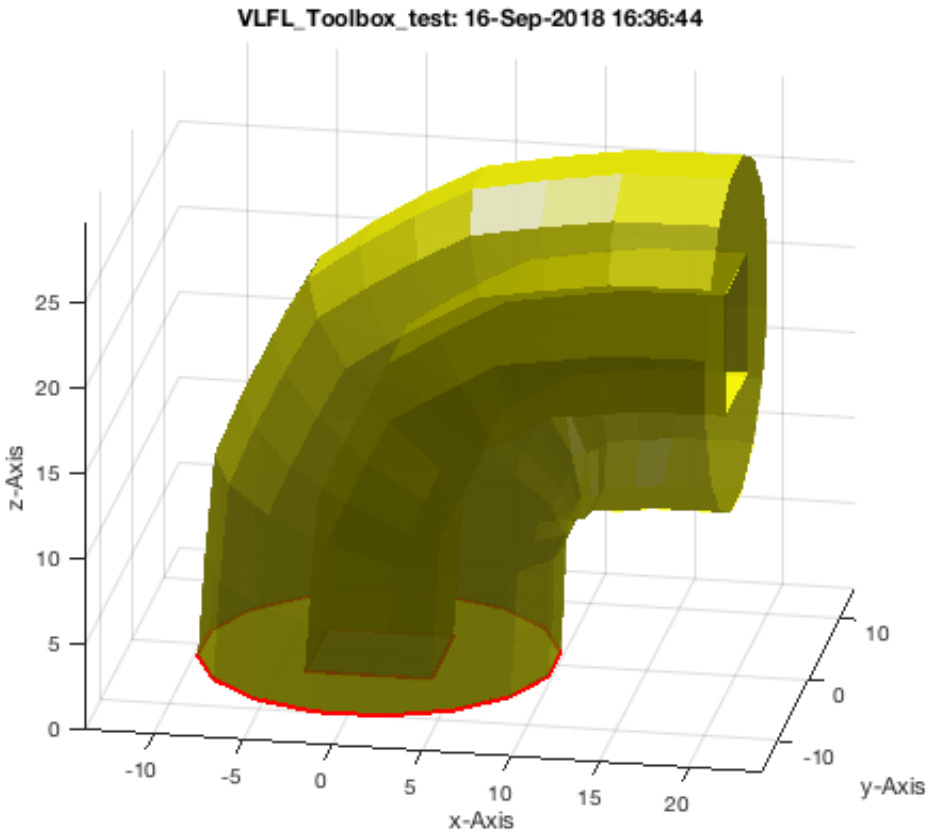

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGcontourtube(CPL,VL,[1 1 0]); SGplot(SG,'g'); VLFLplotlight(1,0.8);view (0,30);
```



One lazy approach is delivered by SGofCPLCVLR without a given radius

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGofCPLCVLR(CPL,VL); SGplot(SG,'y'); VLFLplotlight(1,0.8);view (10,20);
```

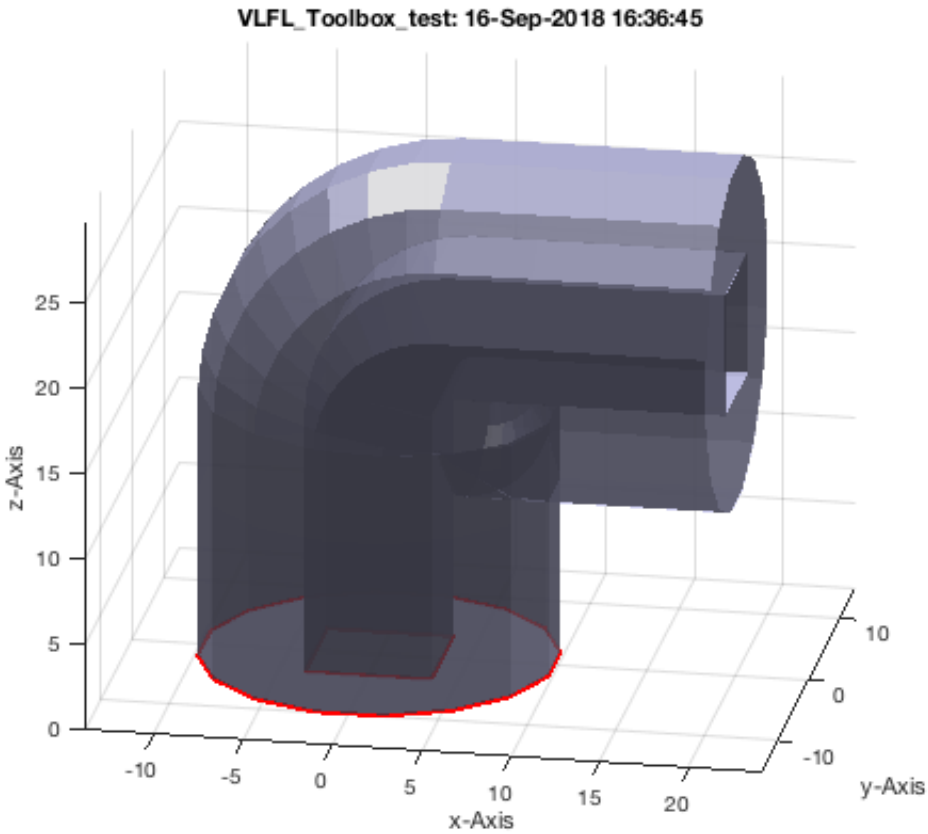
```
Warning: 1st Euler angle does not fit to vertex list path direction
Warning: Last Euler angle does not fit to vertex list path direction
```



One lazy approach is delivered by SGofCPLCVLR including a radius 5

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGofCPLCVLR(CPL,VL,5); SGplot(SG,'w'); VLFLplotlight(1,0.8);view (10,20);
```

```
VLradialEdges: Radius 5.00 reduced to 4.76
```

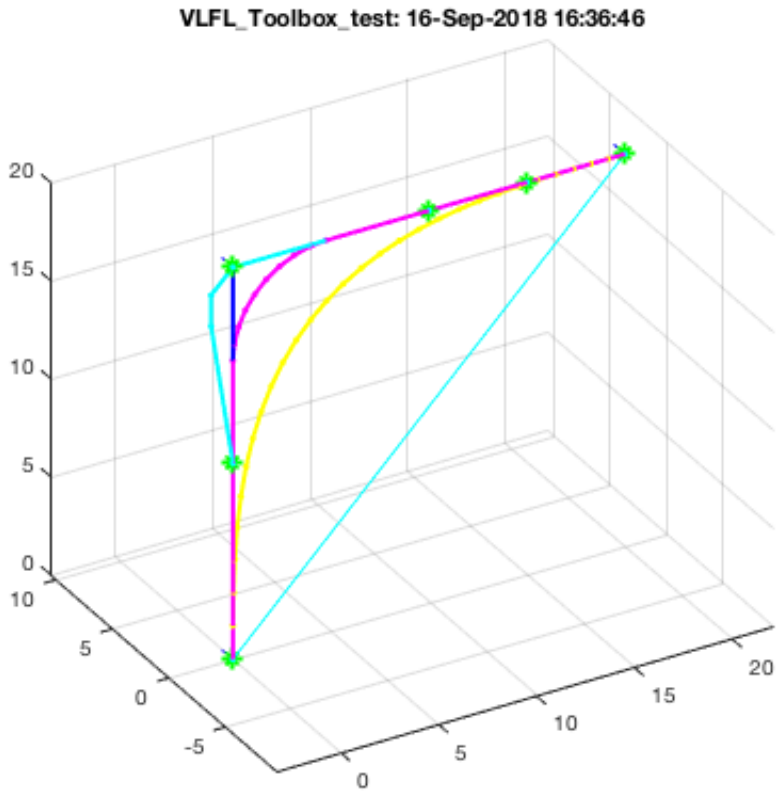


3. Tube generation using a 3D path with Bezier-curves or radial edges

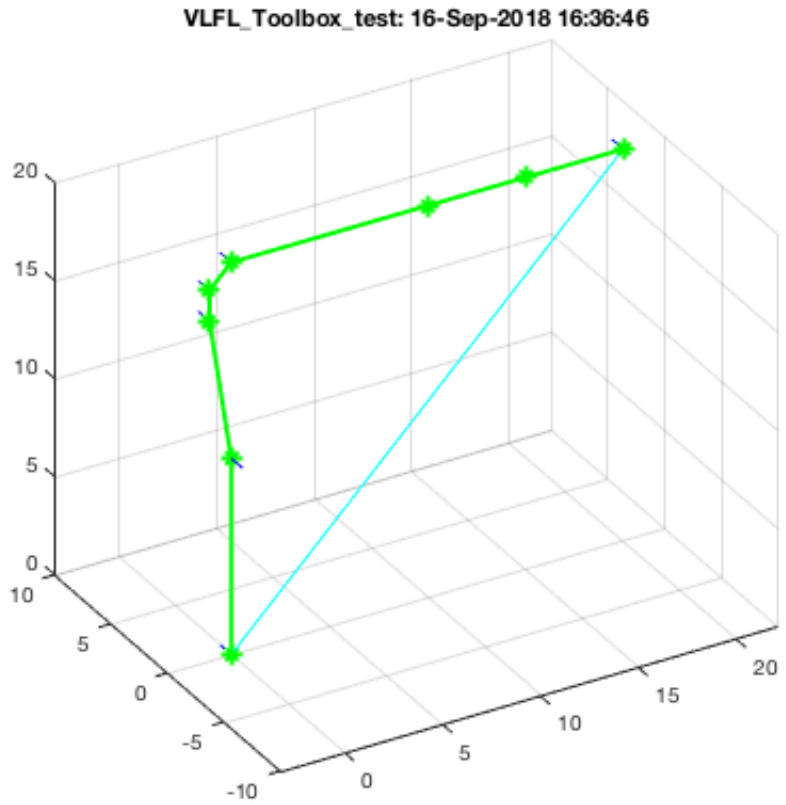
Create a 2D closed polygon line to by copied in 3D space

```
SGfigure; CPLplot(CPL,'r-');  axis on ; grid on;
VL=[0 0 0; 0 0 10; 0 0 20; 10 0 20; 15 0 20; 20 0 20];
VLangle(VL);

VLB=VLBezierC(VL,30);
VLR=VLRadiusC(VL,pi/4,2);
VLr=VLradialEdges(VL,5);
VLplot(VL,'b.-',2); view (-30,30);
VLplot(VLB,'y.-',2); view (-30,30);
VLplot(VLR,'c.-',2); view (-30,30);
VLplot(VLr,'m.-',2); view (-30,30);
```

```
VLradialEdges: Radius 5.00 reduced to 4.76
```

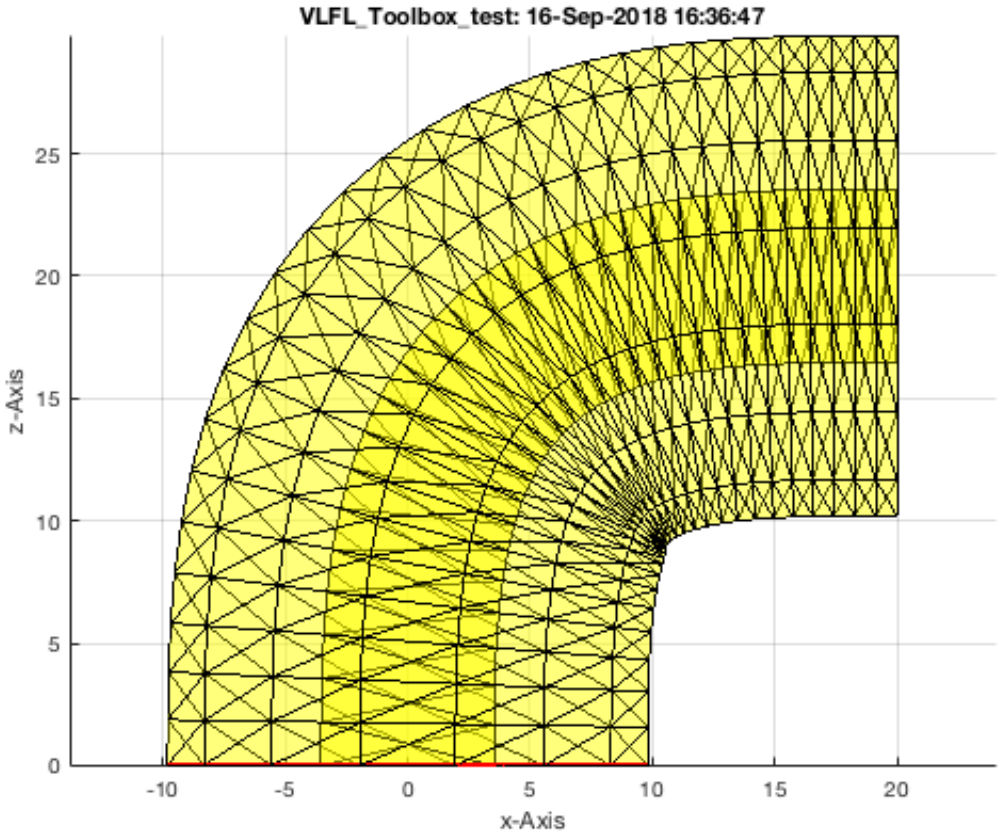

VLFL_Toolbox_test: 16-Sep-2018 16:36:46
20
15
10
5
0
10
5
0
-5
0
5
10
15
20

```
VLangle(VLR);
```



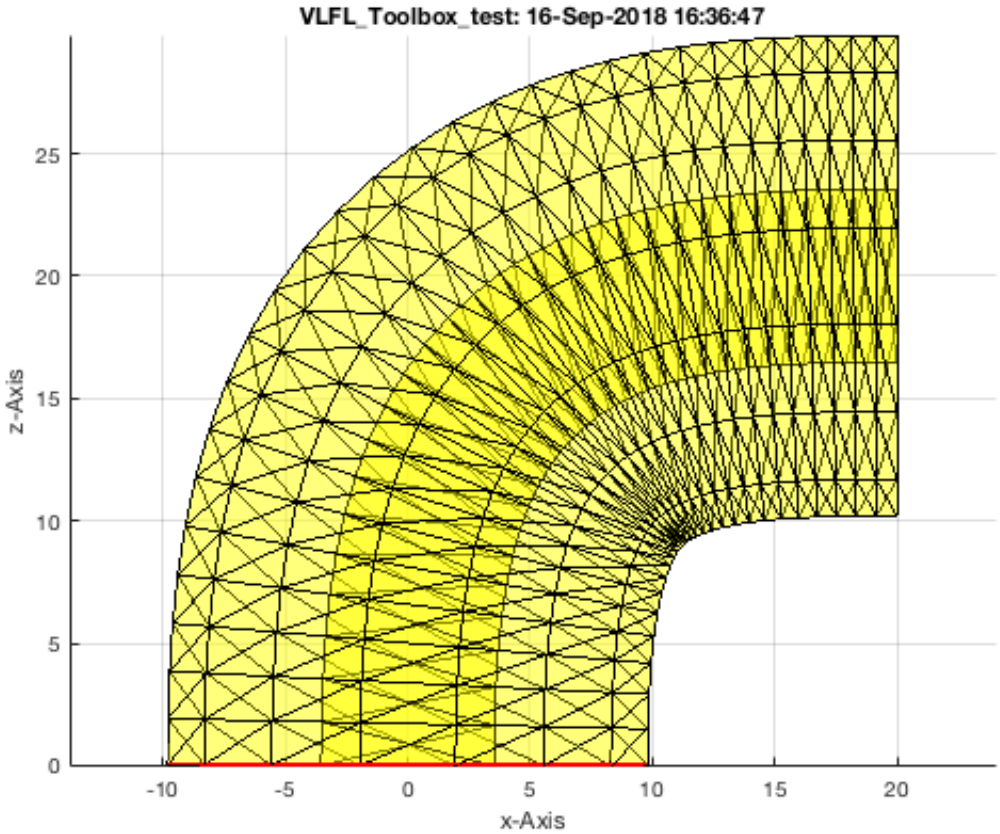
The Bezier-curve tube

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGcontourtube(CPL,VLB); SGplot(SG,'y'); VLFLplotlight(0,0.3);view (0,0);
```



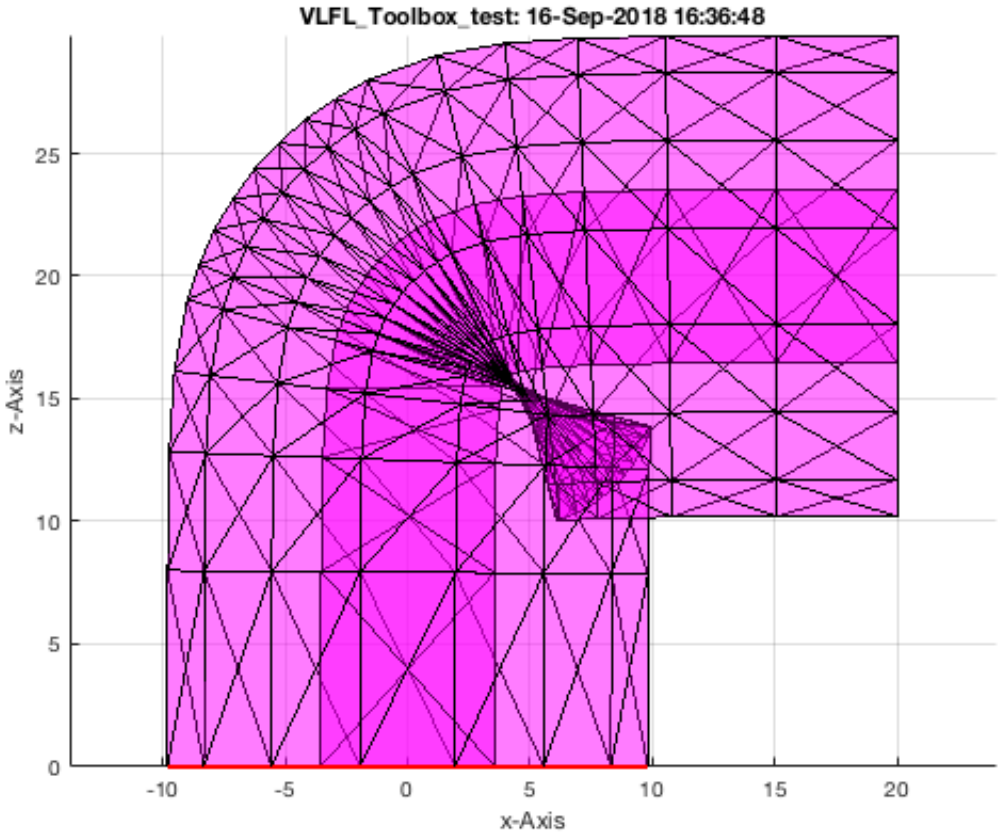
The Bezier-curve tube

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGofCPLCVLR(CPL,VLB); SGplot(SG,'y'); VLFLplotlight(0,0.3);view (0,0);
```



The Radial-curve tube

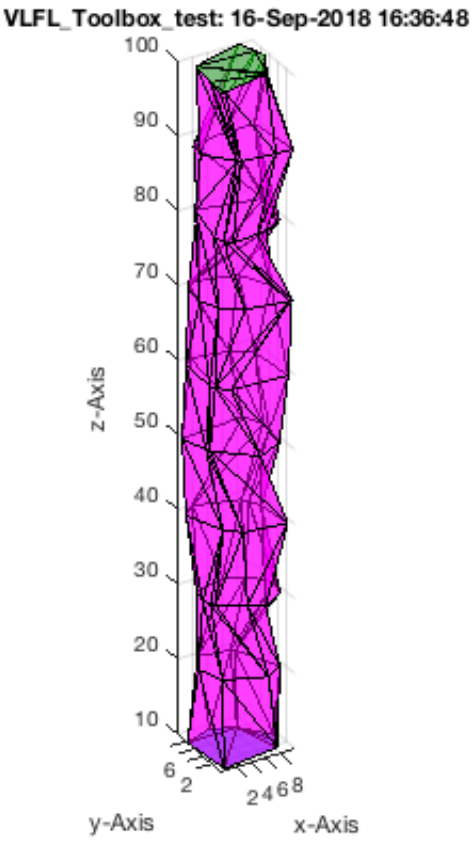
```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGofCPLCVLR(CPL,VLr); SGplot(SG,'m'); VLFLplotlight(0,0.3);view (0,0);
```



4. Creating solids by closed polygons in different height: z-coordinate

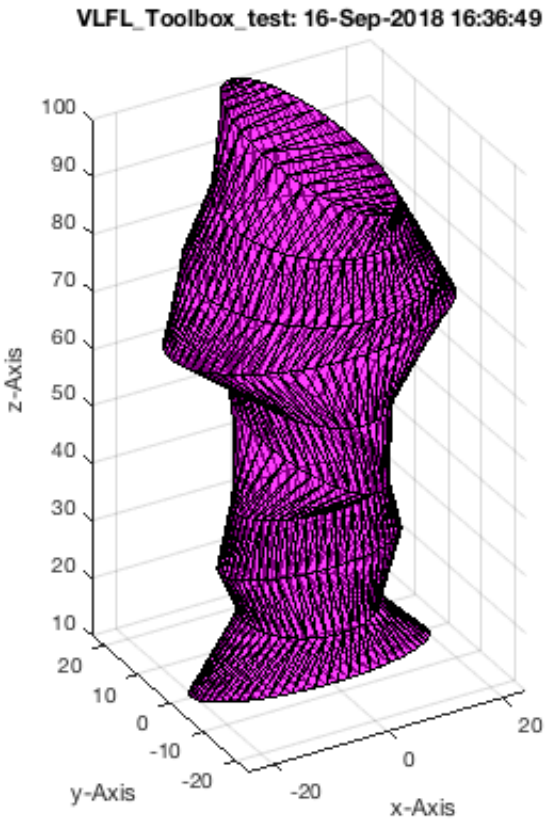
The connection of contours in different z-values works currently only with ONE contour per z-value

```
SGfigure; view(-30,30); axis on; grid on;
VL=[]; for i=1:10; VL=[VL;VLaddz(PLconvexhull(10*rand(20,2)),i*10)]; end;
[FLB,FLW,FLT]=FLofCVL(VL);
VLFLplot(VL,FLB,'b'); VLFLplot(VL,FLW,'m'); VLFLplot(VL,FLT,'g'); VLFLplotlight(0,0.5)
```



Same us but this time with ellipoids

```
SGfigure; view(-30,30); axis on; grid on;
VL=[]; for i=1:10; VL=[VL;VLaddz(PLcircle(5+20*rand,[],[],5+20*rand),i*10)]; end;
[FLB,FLW,FLT]=FLofCVL(VL); FL=[FLB;FLW;FLT];
VLFLplot(VL,FL,'m'); VLFLplotlight(0,0.5)
```



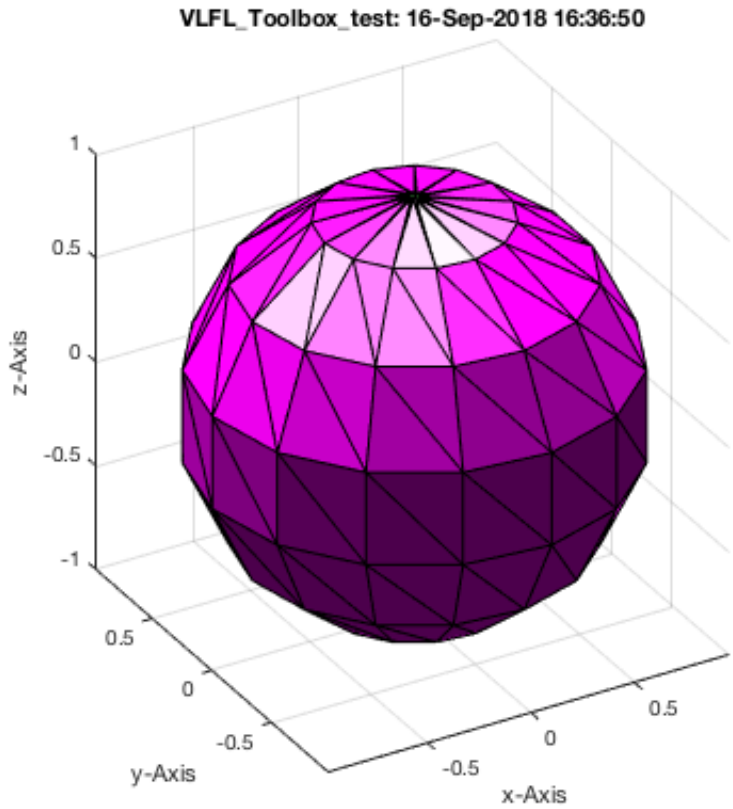
5. Creating a sphere with minmal number of points

For the creation of spherical joints, we need sphered shaped geometries. Those spheres consist of circular point lists in different z-height. The number of points of each polygon, the number of polygons an the z-resolution depend on the size of the sphere.

A sphere with just 1mm radius and a resolution of 50µm (default) has only hundreds of facets.

```
SGsphere(1)
```

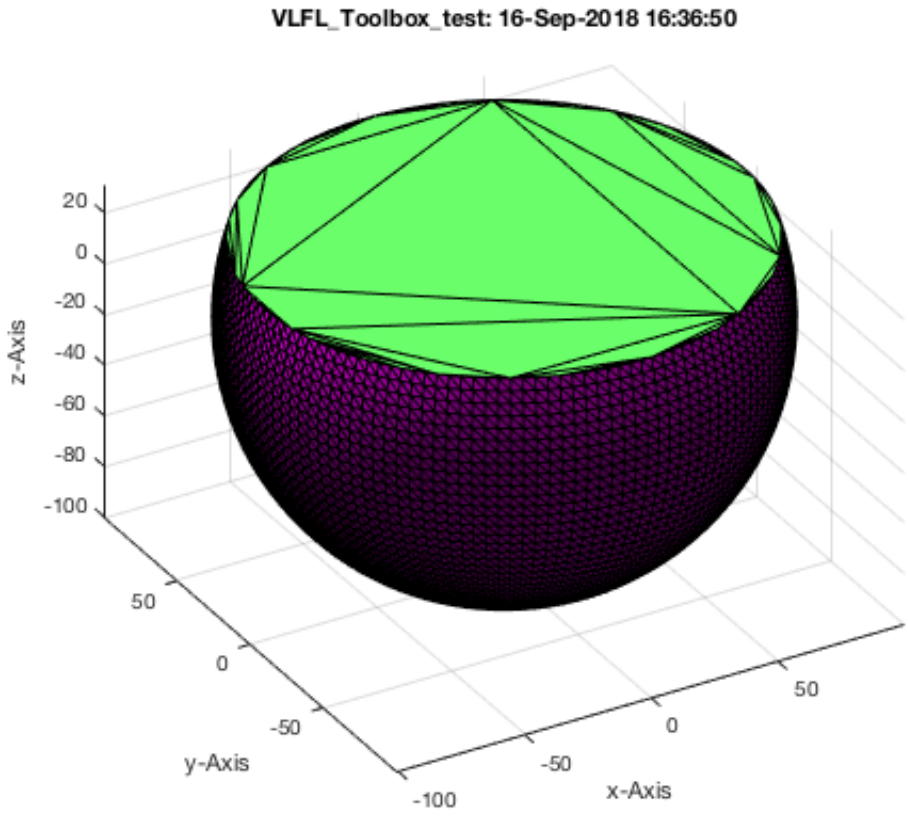
```
ans =

  struct with fields:

    VL: [120×3 double]
    FL: [236×3 double]
```



Asphere with 100mm radius and a resolution of 50µm (default) has then thausands of facets.

```
SGsphere(100,[],pi/10)
```

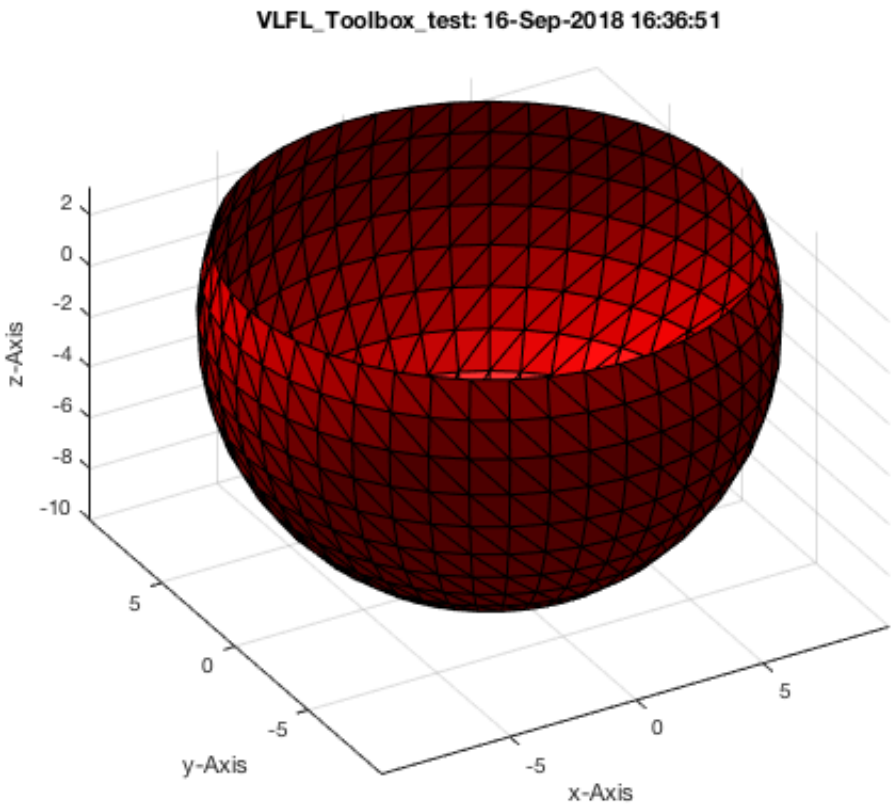
```
ans =

  struct with fields:

    VL: [6063×3 double]
    FL: [12122×3 double]
```



6. Creating a spherical joint

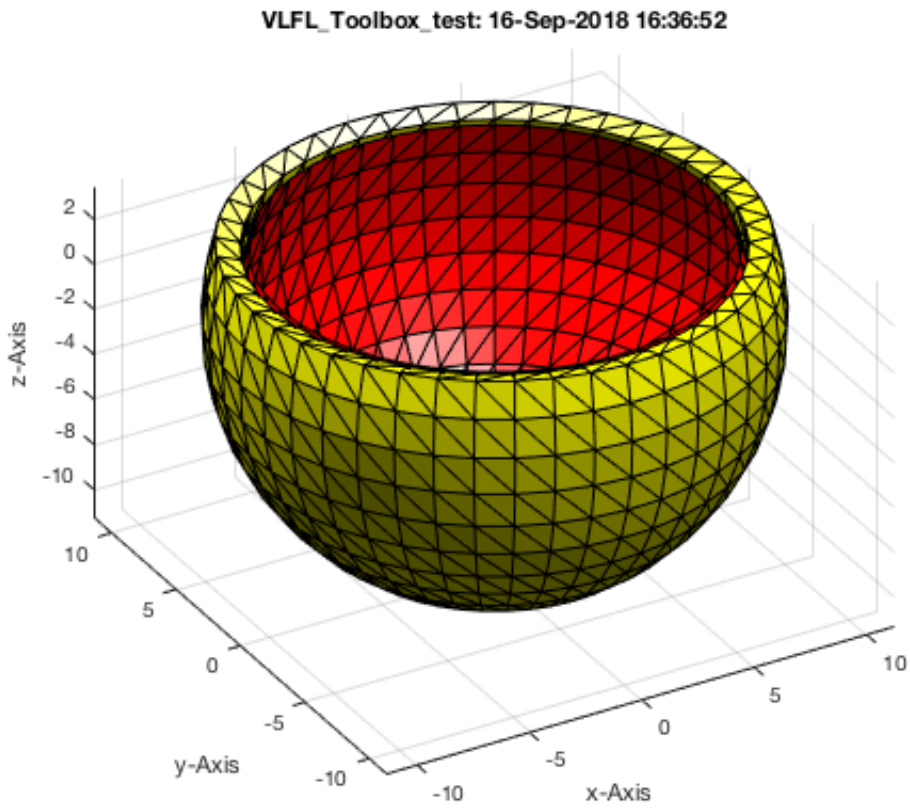
First we have to create a sphere and separate the spherical surface

```
SGfigure; view(-30,30);
[~,~,SG]=MLofSG(SGsphere(10,[],pi/10));
VLFLplot(SG.VL,SG.FL(SG.ML(:,1)==1,:));
```



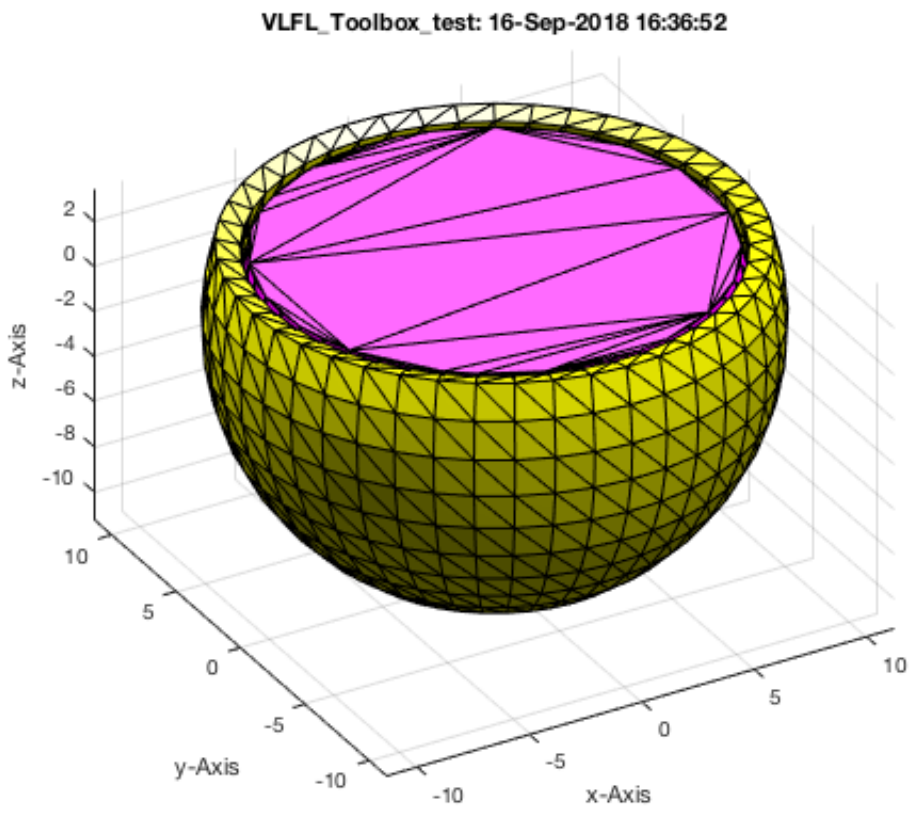
Now create the surface for the joint from the spherical surfacewith tickness 1 as bearing

```
SGofSurface(SG.VL,SG.FL(SG.ML(:,1)==1,:),1);
```



Now fill in the sphere ball as joint

```
SGplot(SG,'m');
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:36:53!
Executed 16-Sep-2018 16:36:55 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-10-12*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 17: Filling and Bending of Polygons and Solids

2017-03-29: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.7 required)
- 1. Creating Closed Polygon Line
- 2. Converting CPL into PL EL
- 3. Adding and removing points on the contour
- 4. Adding and removing points inside of the contour
- 5. Calculate Grid Points
- 6. Bending of contours
- 7. Bending of closed contour surfaces
- 8. Bending of solid geometries
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines

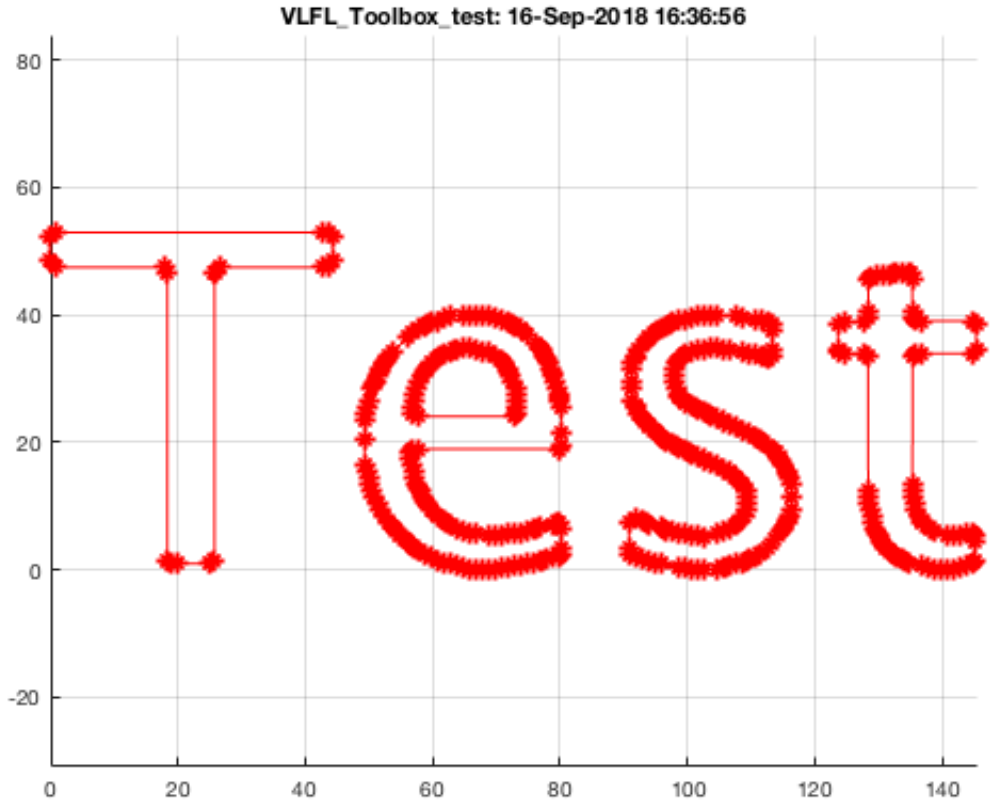
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.7 required)

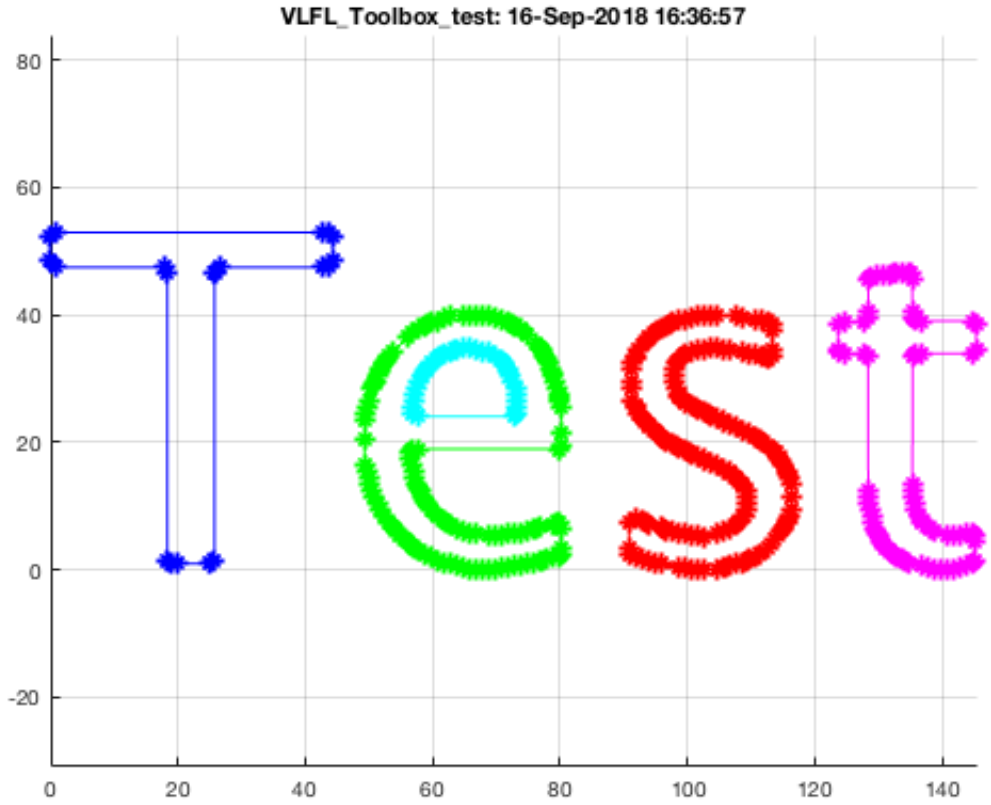
For robotics design,very often we have a wish to extrude a CPL not only in an orthognal z direction to the xy-plane, but in an any desired direction even along a path in 3D space. Intutively we expect a result, but this is not easy to achieve automatically. Anyway, for those tasks we have two functions:

1. Creating Closed Polygon Line

```
CPL=CPLoftext('Test');
SGfigure; view(0,90); CPLplot(CPL);
```

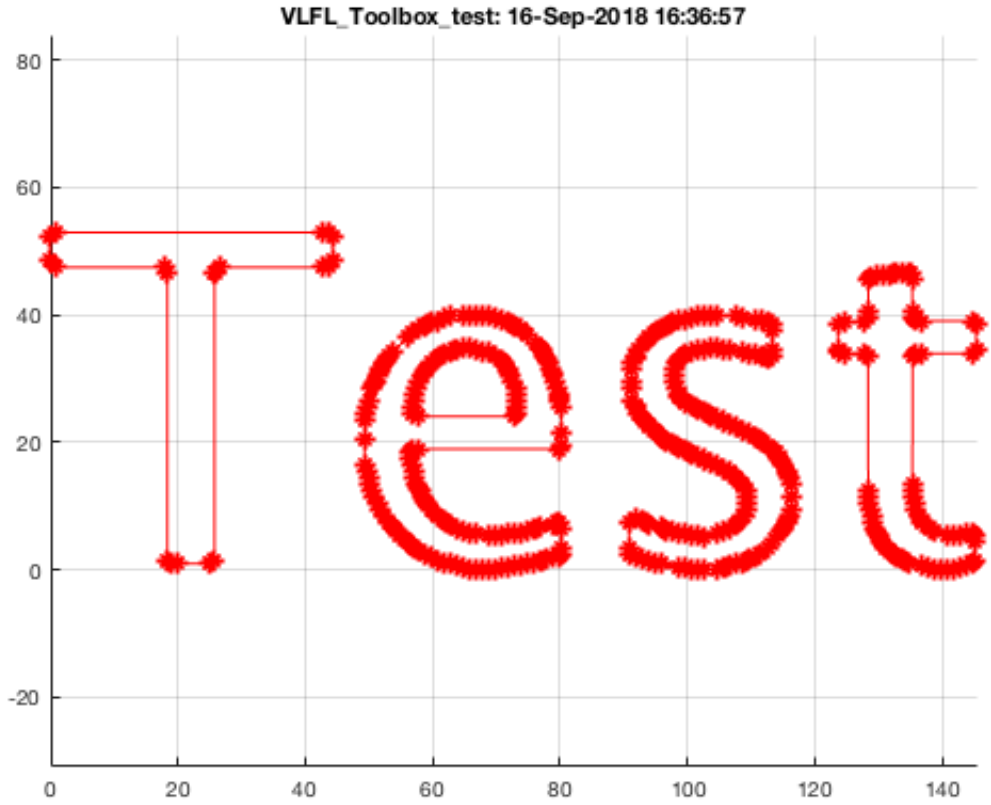



```
SGfigure; view(0,90); CVLplot(CPL);
```

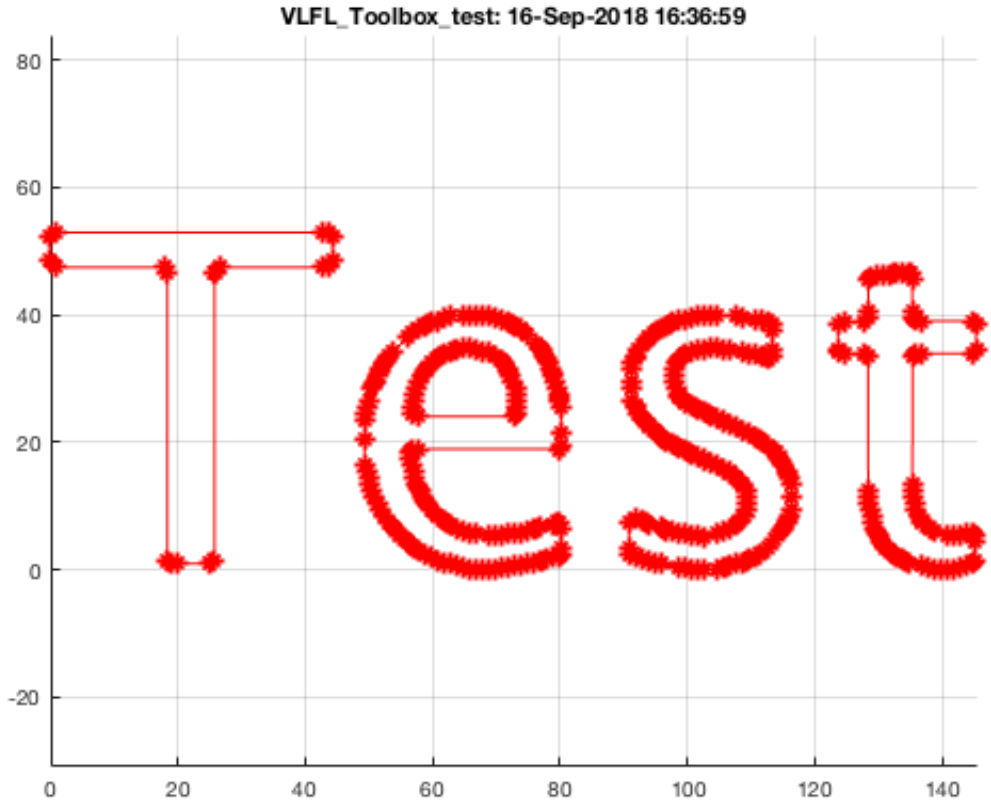


2. Converting CPL into PL EL

```
[PL,EL]=PLELofCPL(CPL);
SGfigure; view(0,90); PLELplot(PL,EL);
```

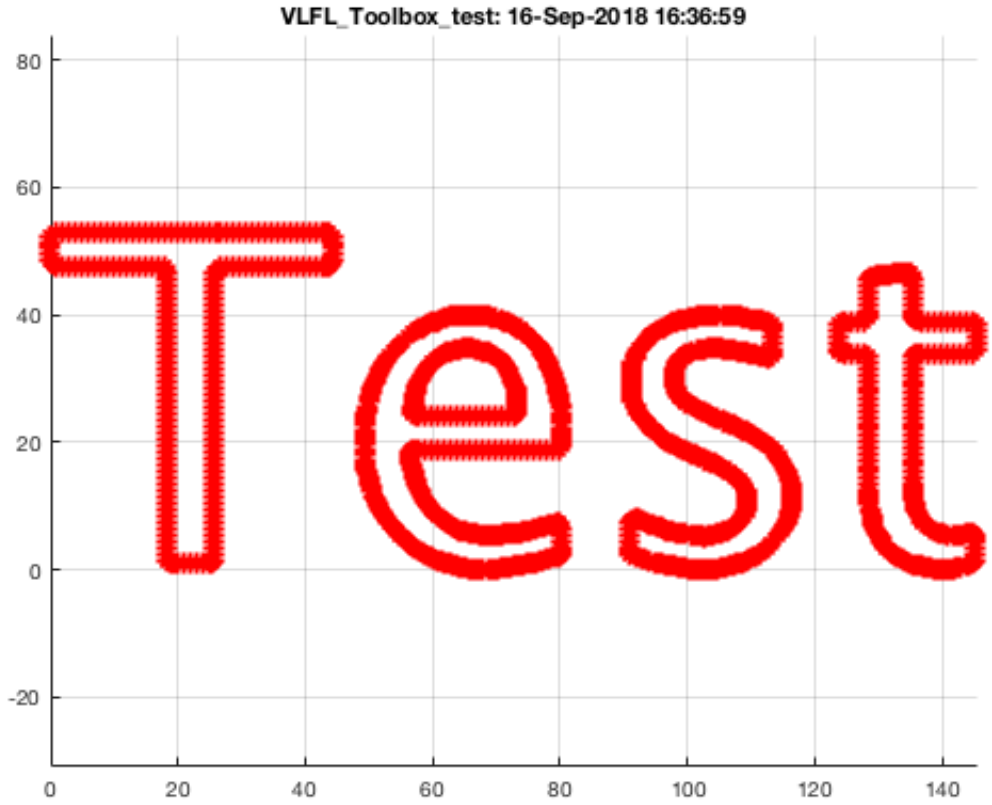



```
CPL=CPLofPLEL(PL,EL);
SGfigure; view(0,90); CPLplot(CPL);
```

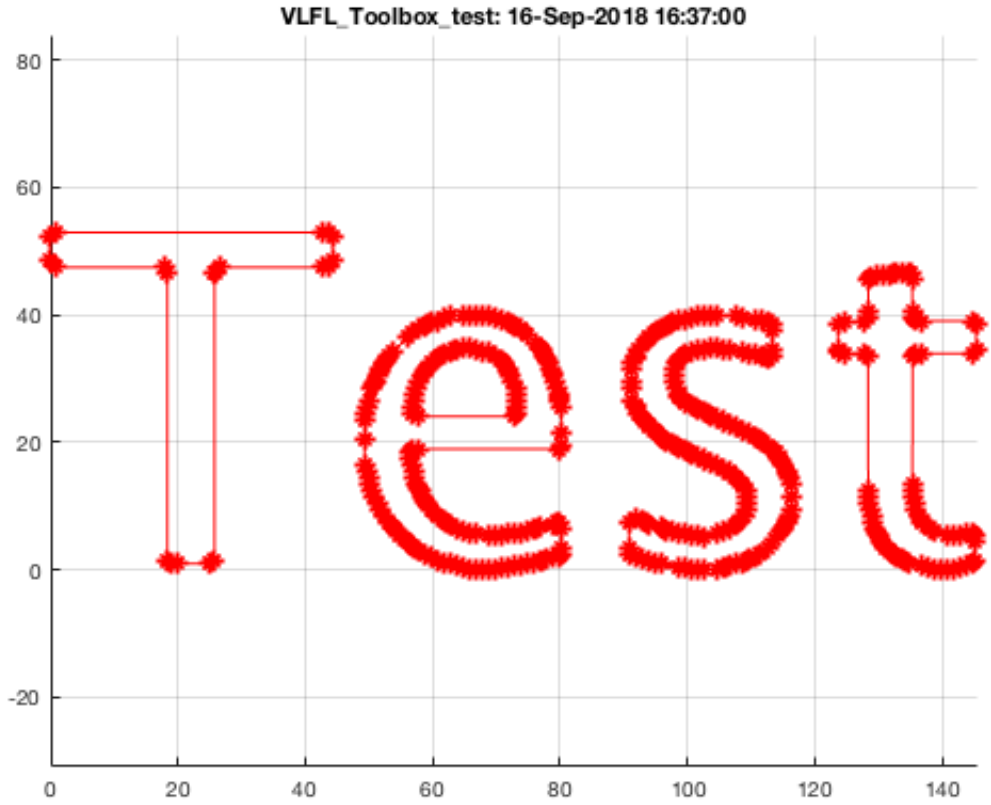


3. Adding and removing points on the contour

```
CPLN=CPLaddauxpoints(CPL,1);
SGfigure; view(0,90); CPLplot(CPLN);
```

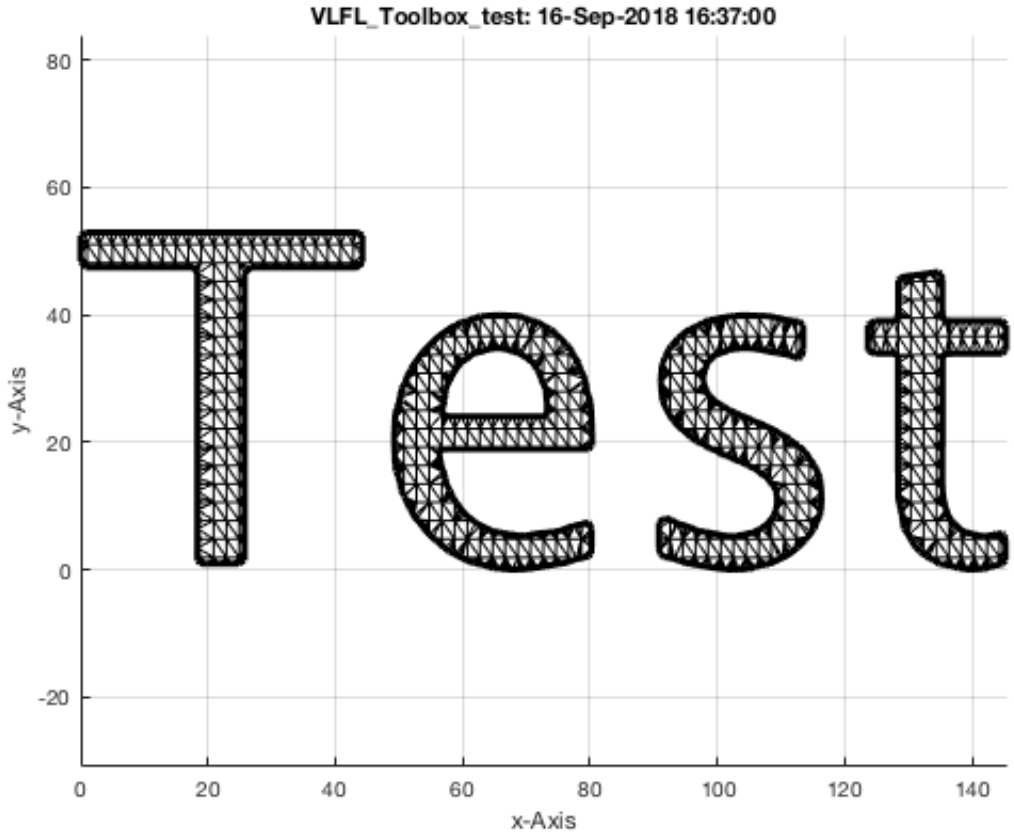



```
CPLB=CPLremstraight(CPL);
SGfigure; view(0,90); CPLplot(CPLB);
```



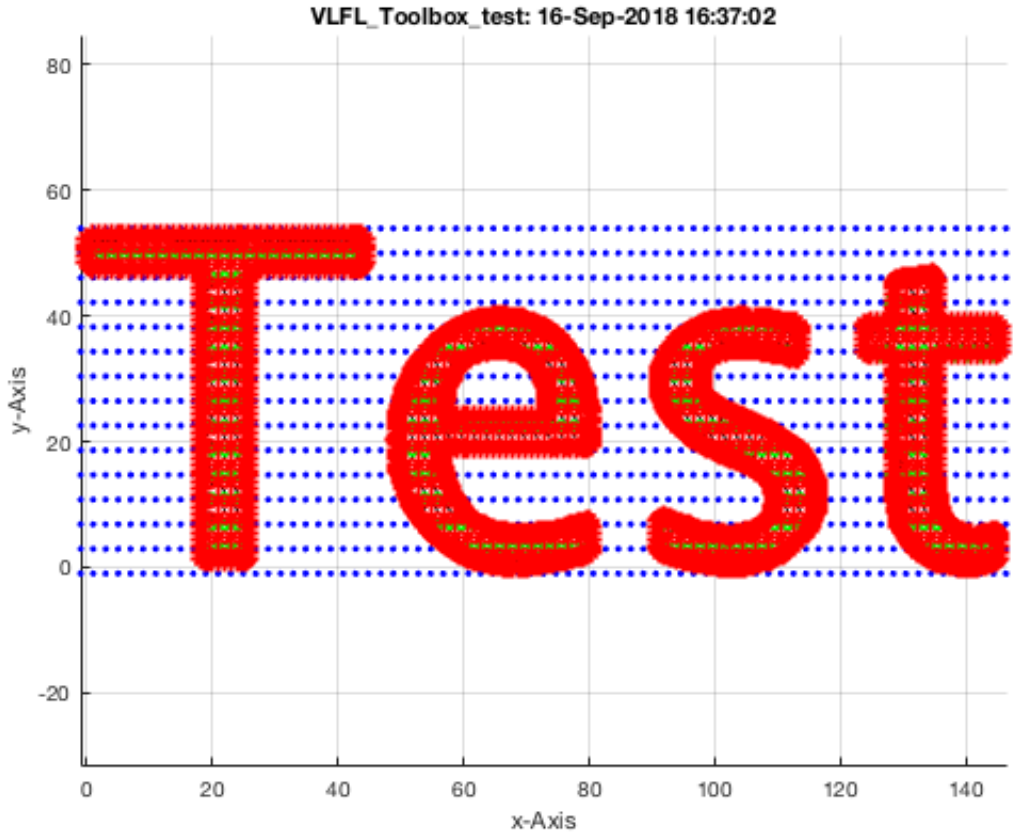
4. Adding and removing points inside of the contour

```
[PL,FL,EL]=PLFLofCPLdelaunayGrid(CPL,1,2,3);
SGfigure; view(0,90); VLFLplot(PL,FL); VLELplots(PL,EL,'k',3);
```



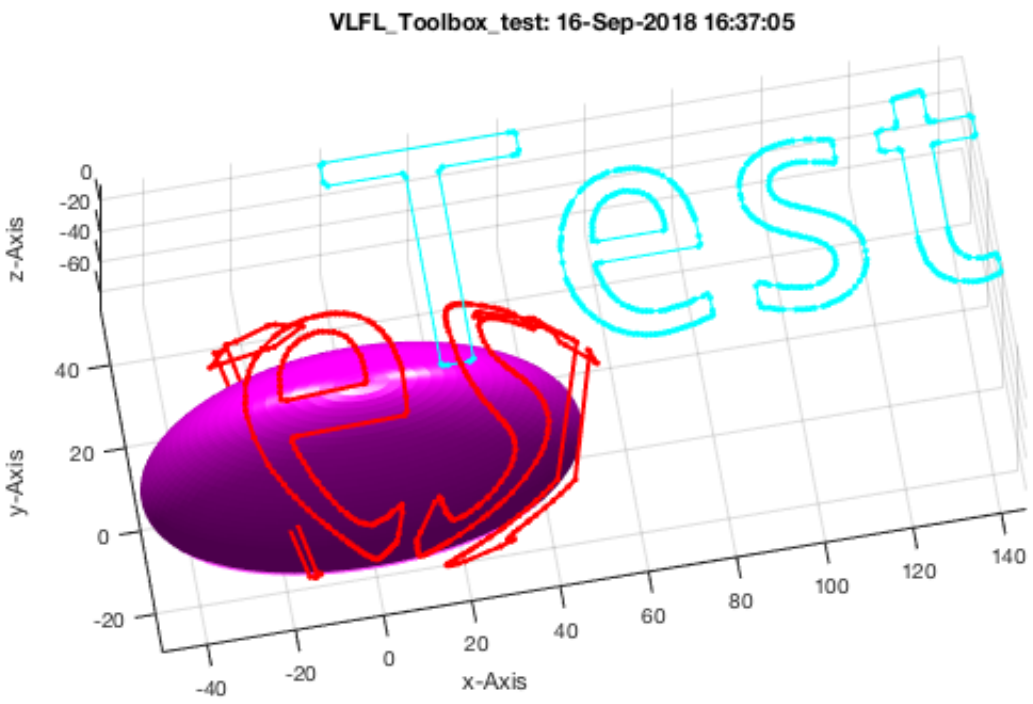
5. Calculate Grid Points

```
GPL=GPLauxgridpointsPLEL(PL,EL,2,4);
insidePLELdelaunay(PL,EL,GPL);
```

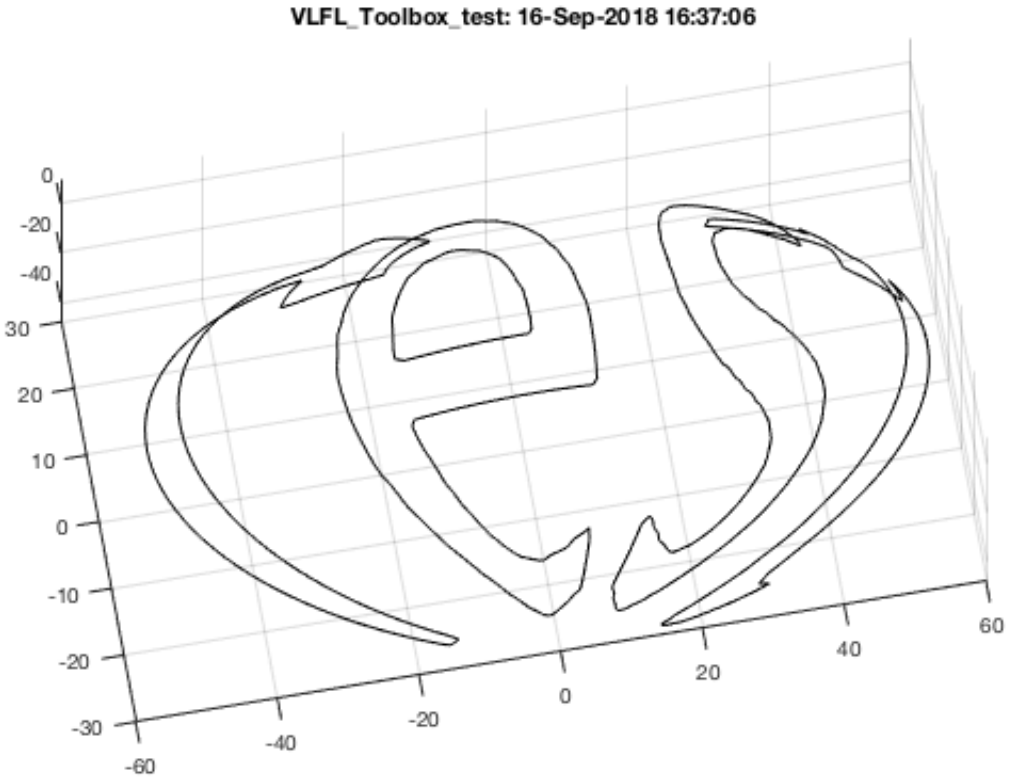


6. Bending of contours

```
% Without auxiliary points

BPL=PLbending(CPL,50,10,20);
PLbending(CPL,50,10,20);

% With auxiliary points
```

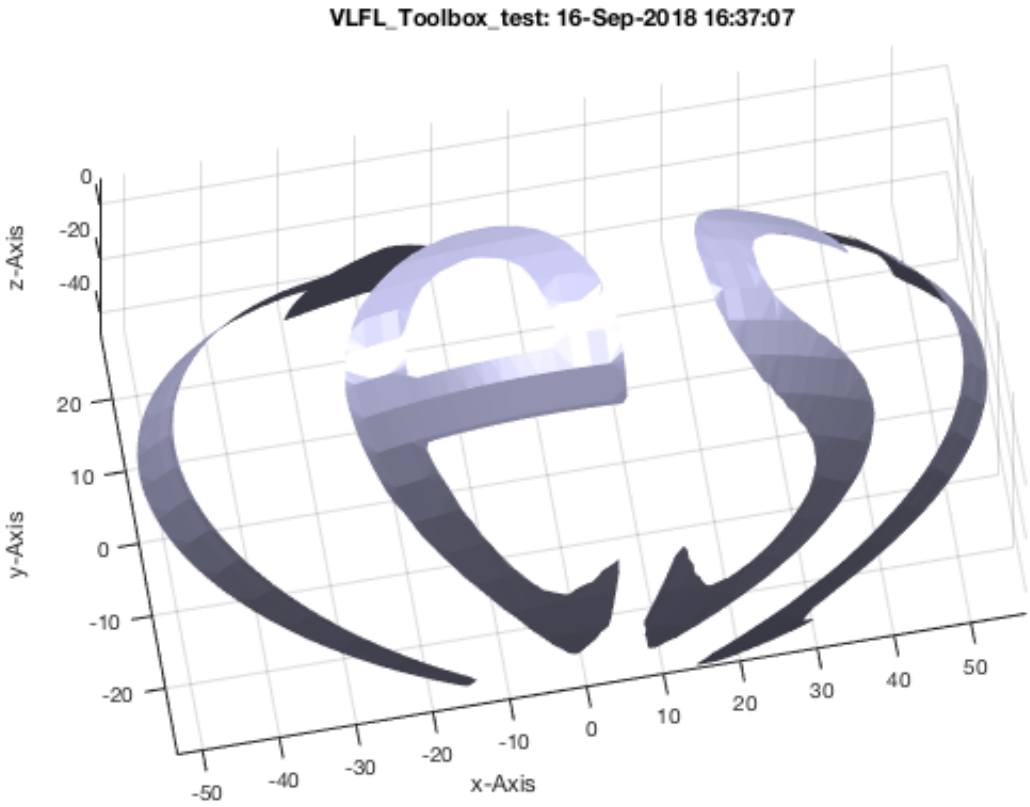



```
[PL,FL,EL]=PLFLofCPLdelaunayGrid(CPL,1,2,3);
NPL=PLbending(PL,50,10,20);
SGfigure; view(-10,70); VLELplot(NPL,EL,'k-');
```



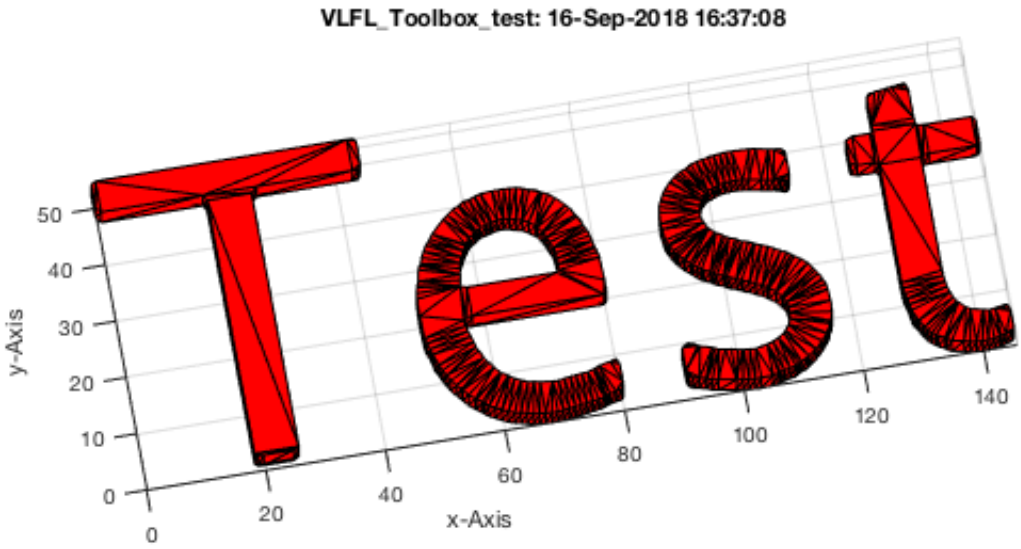
7. Bending of closed contour surfaces

```
SGfigure; view(-10,70); VLFLplot(NPL,FL,'k-'); VLFLplotlight(1,1);
```

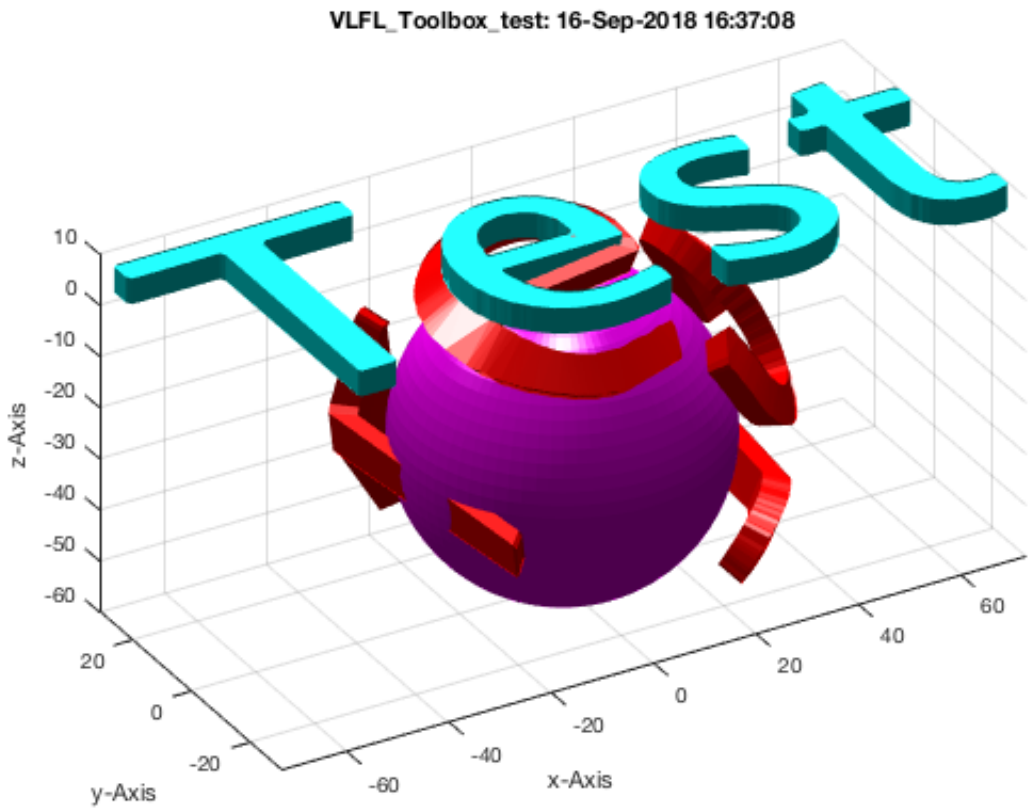


8. Bending of solid geometries

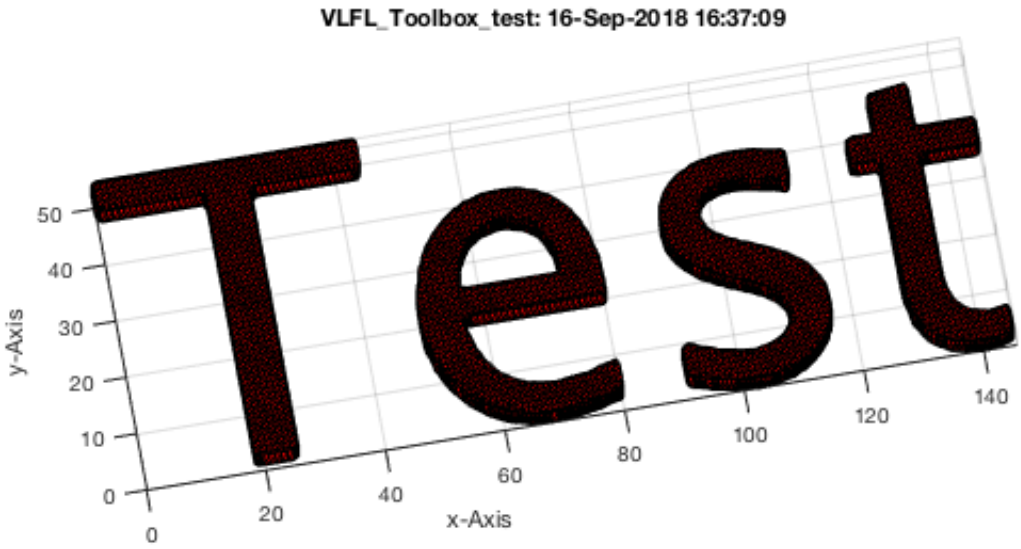
```
% Without auxiliary points
SG=SGofCPLzdelaunayGrid (CPL,5);
SGfigure; view(-10,70); SGplot(SG); VLFLplotlight(0,1);
```



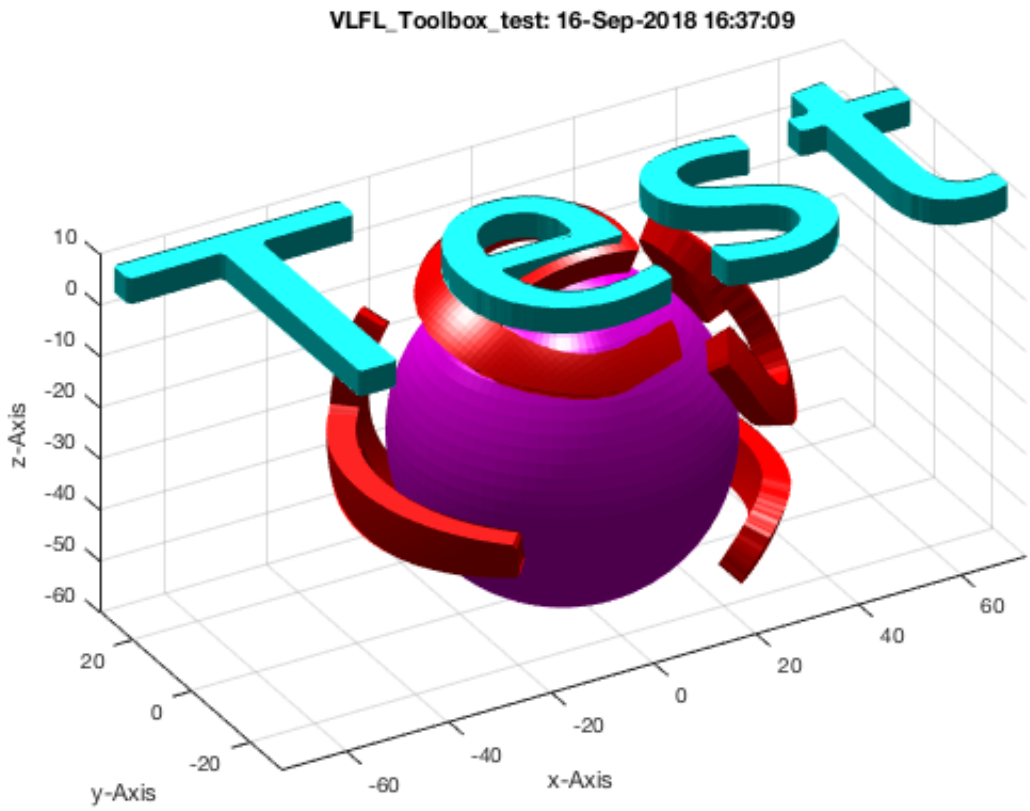

```
SGbending(SG,30,5,30); VLFLplotlight(1,1);
```




```
% With auxiliary points
SG=SGofCPLzdelaunayGrid (CPL,5,1,1,1);
SGfigure; view(-10,70); SGplot(SG); VLFLplotlight(0,1);
```




```
SGbending(SG,30,5,30); VLFLplotlight(1,1);
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:37:10!
Executed 16-Sep-2018 16:37:12 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016b on 2017-03-29*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)

2017-04-04: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.8 required)
- 1. Show all separated surfaces that are part of a Solid
- 2.Select some of the surfaces
- 3. Show the size of the surfaces as histogram
- 4. Show just a single solid
- 5. Shrink all convex parts
- 6. Print all surfaces in different STL files
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

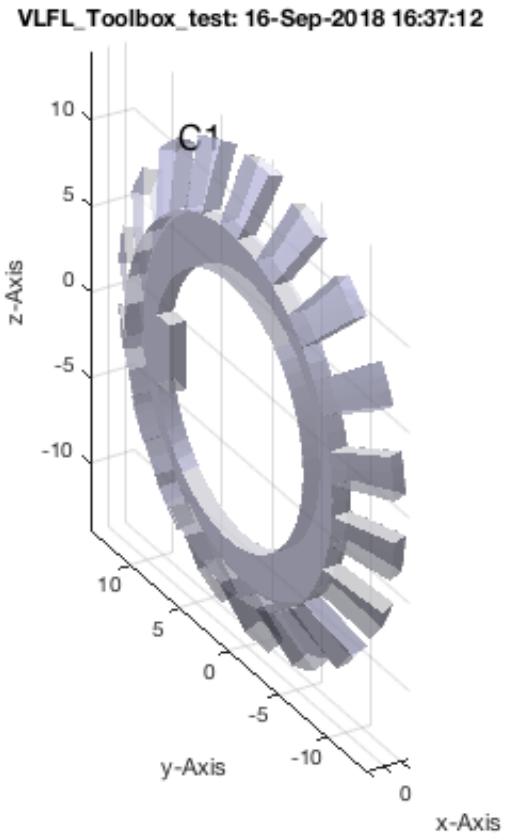
Motivation for this tutorial: (Originally SolidGeometry 3.8 required)

Often CSG modelers are used for mechanism construction and the subsequent STL export. This tutorial will show you how to use those STL files after reading them in as SG

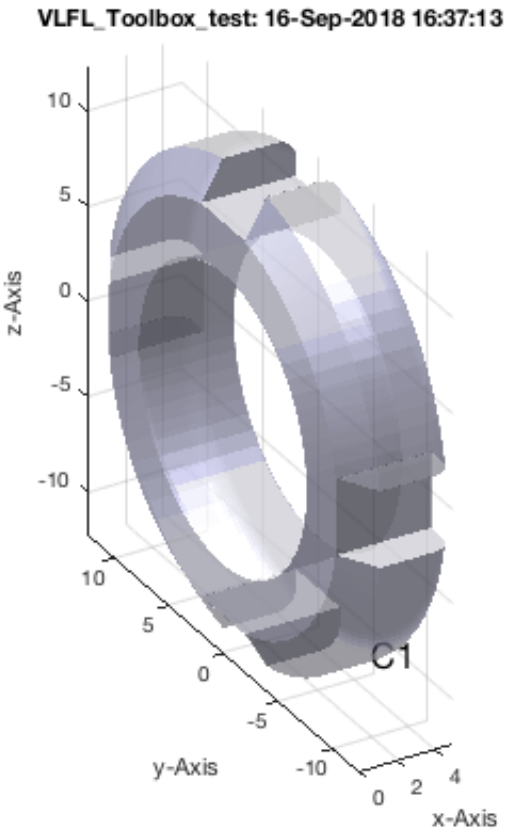
The mat-File 'FZG_Welle.mat' contains already read STL files of the TUM FZG institute. There are 6 Solids that contain overall 18 separate surfaces of a bearing for an axle. You can either load the data from the WWW page of the Technical University of Munich or after download use the load command.

```
% loadweb ('FZG_Welle.mat',true) % load the data from the TUM Mimed Page
load ('FZG_Welle.mat'); % load the data from the matlab path
FZG={SG1,SG2,SG3,SG4,SG5,SG6};
```

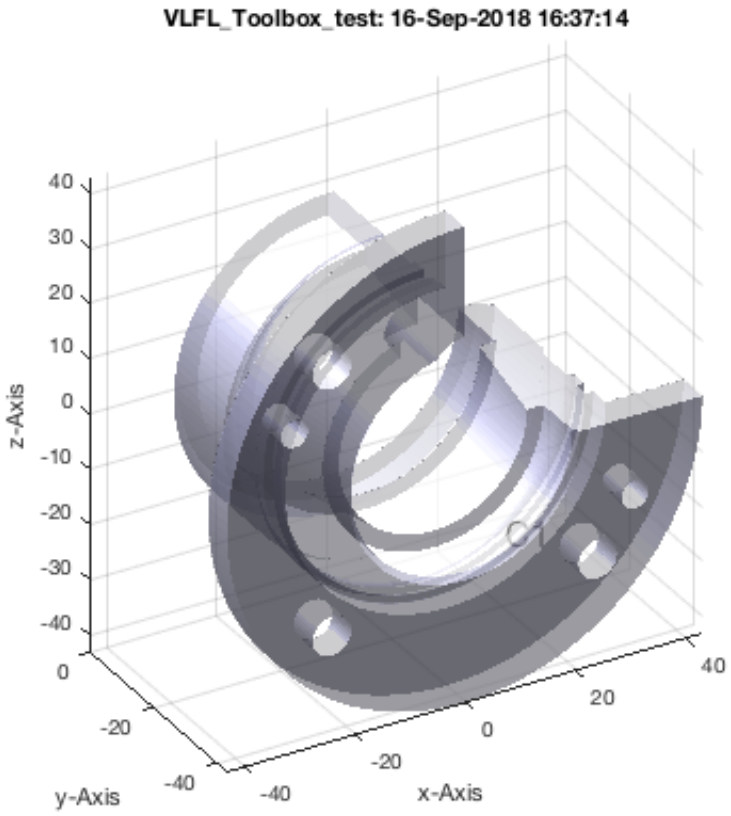
```
SGfigure; SGsurfaceplot(SG1); view(-30,30);
```



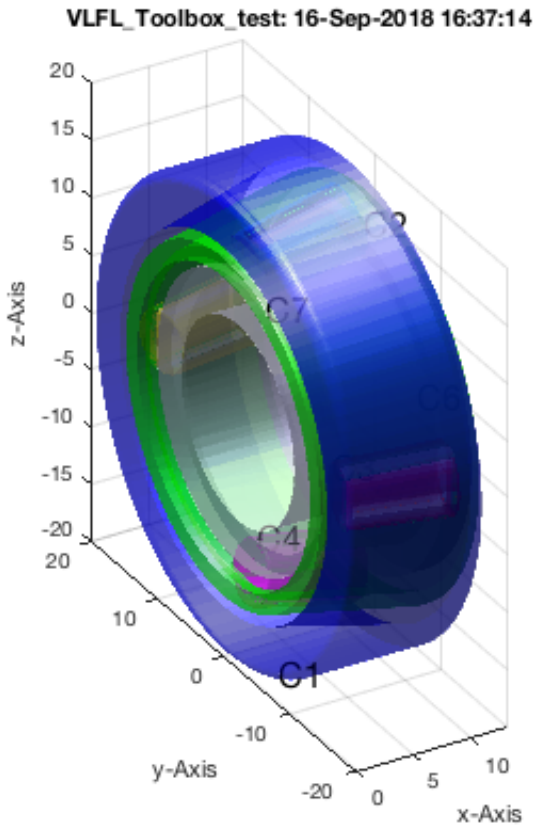

```
SGfigure; SGsurfaceplot(SG2); view(-30,30);
```



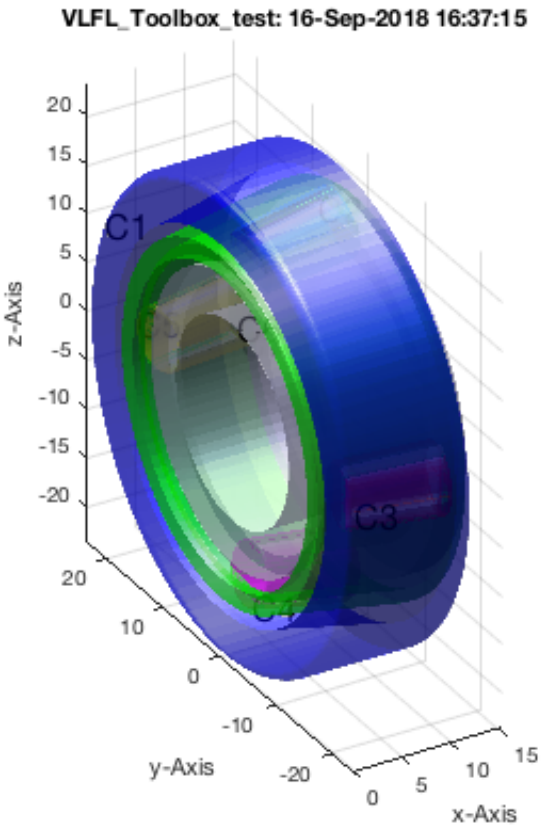

```
SGfigure; SGsurfaceplot(SG3); view(-30,30);
```



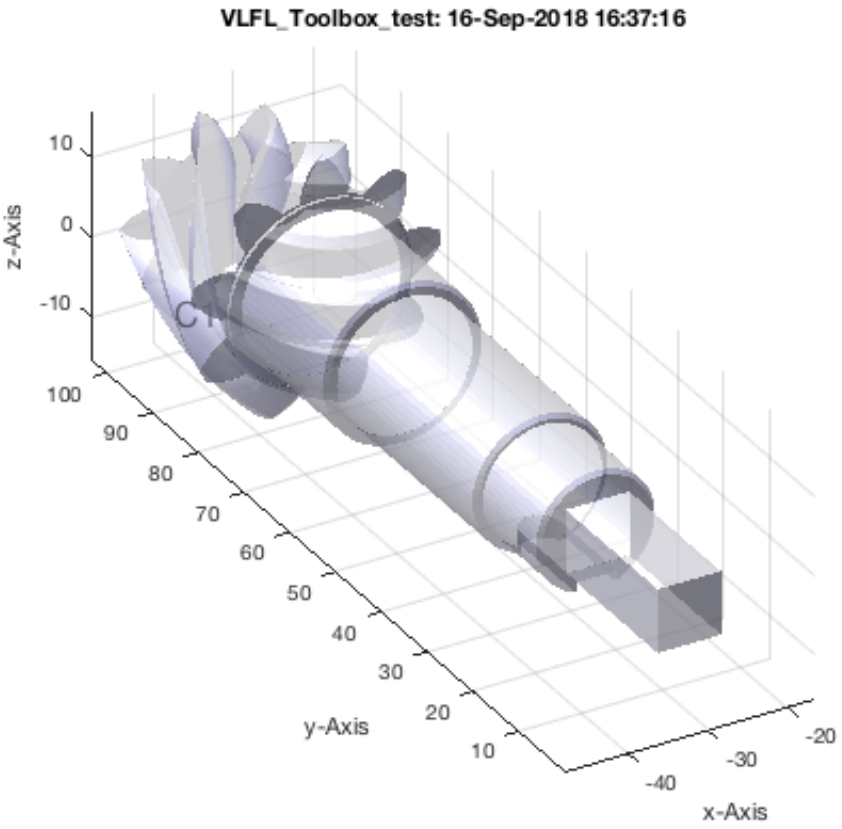

```
SGfigure; SGsurfaceplot(SG4); view(-30,30);
```




```
SGfigure; SGsurfaceplot(SG5); view(-30,30);
```

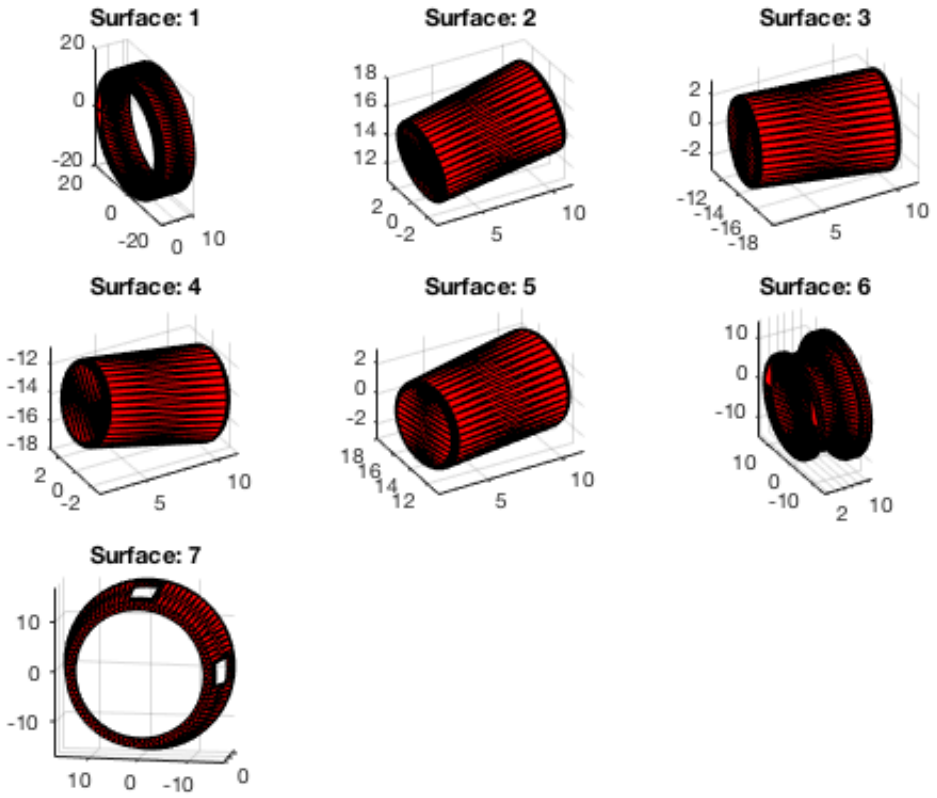



```
SGfigure; SGsurfaceplot(SG6); view(-30,30);
```



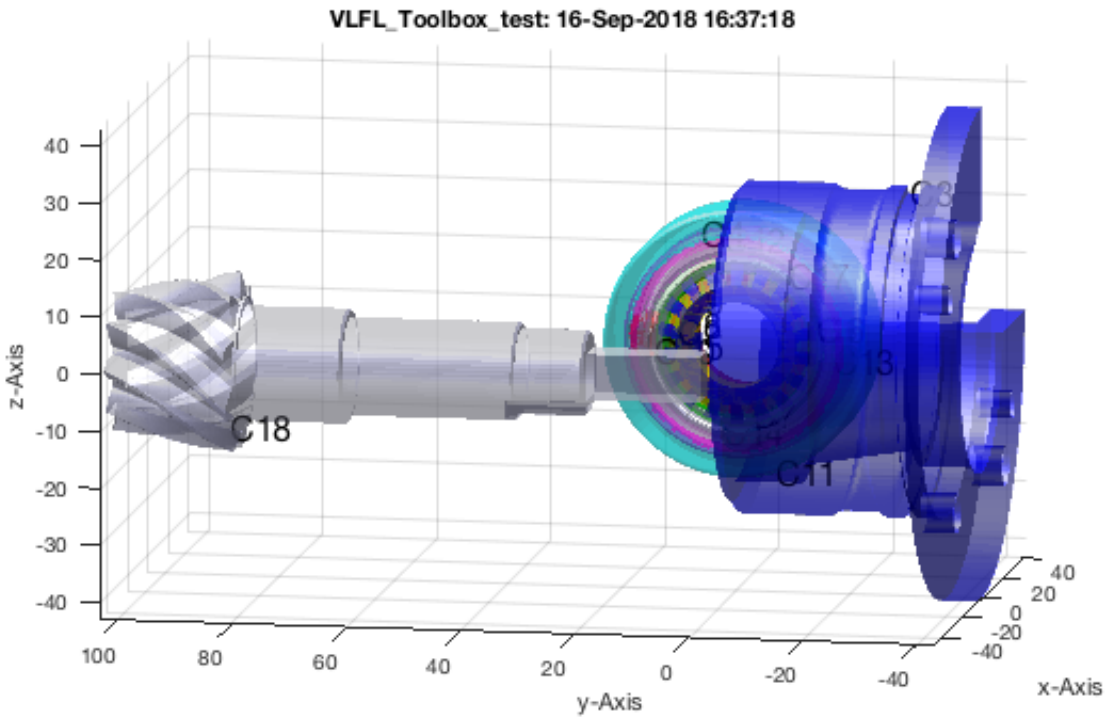
1. Show all separated surfaces that are part of a Solid

```
SGseparate(SG4); view(-80,10);
```



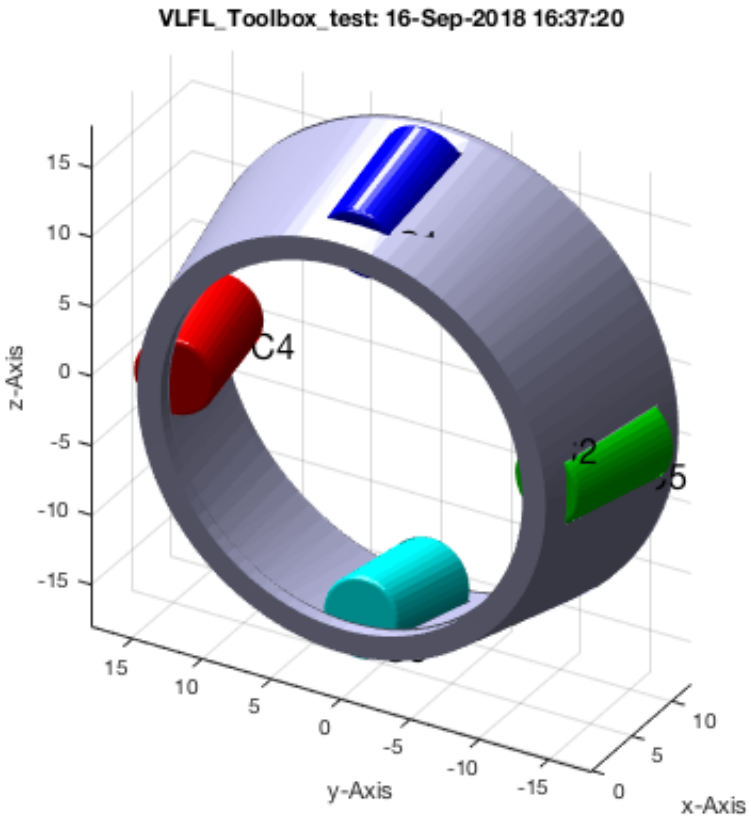
Show all surfaces in different colors

```
SGsurfaces(FZG); view(-80,10);
```



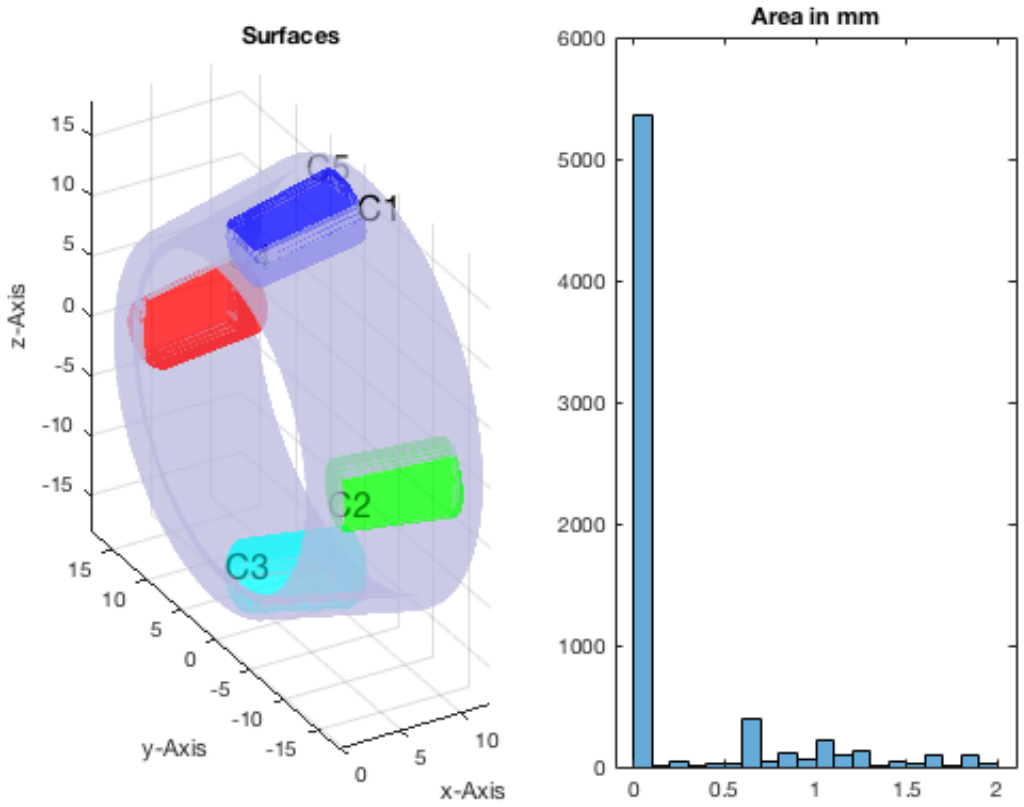
2.Select some of the surfaces

```
SGsurfaces(SG4,[2 3 4 5 7]); view(-60,30); VLFLplotlight(1,1);
```



3. Show the size of the surfaces as histogram

```
SGsurfacehistogram(SG4,[2 3 4 5 7]);
```



4. Show just a single solid

```
B=SGsurfaces(SG4)
```

```
B =

  7×1 cell array

    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
```

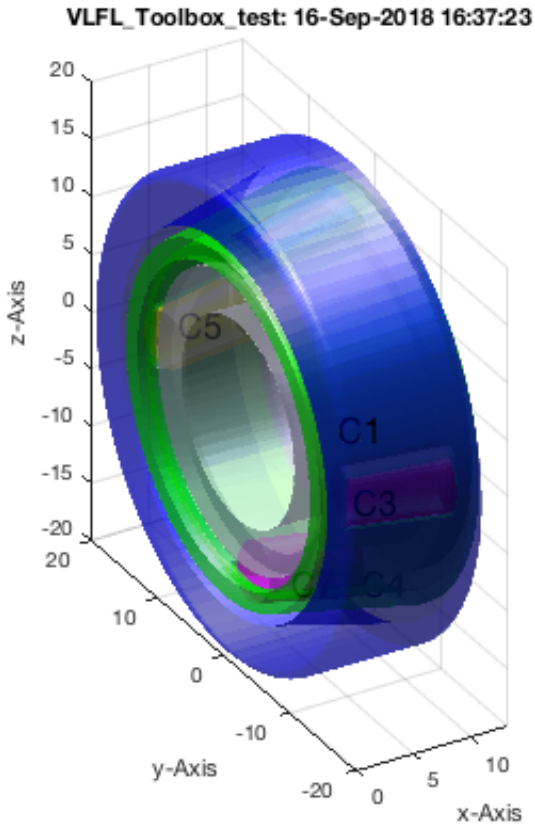
5. Shrink all convex parts

Now reduce al convex solids by 0.3 mm SG=SGreadSTL('30204-a.stl')

```
B=SGsurfaces(SG4)
for i=1:length(B)
    if SGisconvex(B{i})
        B{i}=SGgrow(B{i},-0.3);
%         B{i}=SGofVLdelaunay(B{i}.VL); % Just to show convex solids
```

```
    end
end


SGsurfaces(B);
```

```
B =

  7×1 cell array

    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
    {1×1 struct}
```



6. Print all surfaces in different STL files

```
SGwriteMultipleSTL(B)
```

```
SGwritemultipleSTL: Writing 7 STL files in /Users/timlueth/Desktop/Toolbox_test/EXP-2018-09
-16/
```

Show the written files on disk

```
dir ([desktopdir expname])  %
```

```
.                        EXP-2018-09-16_0003.stl  EXP-2018-09-16_0007.stl
..                       EXP-2018-09-16_0004.stl
EXP-2018-09-16_0001.stl  EXP-2018-09-16_0005.stl
EXP-2018-09-16_0002.stl  EXP-2018-09-16_0006.stl
```

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:37:24!
Executed 16-Sep-2018 16:37:26 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016b on 2017-03-29*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 19: Creating drawing templates and dimensioning from polygon lines

2017-04-23: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.8 required)
- Motivation
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links

- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.8 required)

```
%
% function VLFL_EXP19
```

Motivation

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:37:26!
Executed 16-Sep-2018 16:37:28 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016b on 2017-03-29*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

2016-11-19: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.0 required)
- 2. Creating a new SimMechanics System
- 3. Create two links with length 50 and 80 and one or two mounting holes
- 4. Create SimMechanics models for the four links in different colors
- 5. Create SimMechanics models for the four joint and connect them with the links
- 6. Connect the base frame of link1 to the world coordinate system
- 7. Run the Simulation of the Simulink/SimMechanics diagram for 1 second
- 8. Create a Video of the Simualatin for 5 seconds
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.0 required)

2. Creating a new SimMechanics System

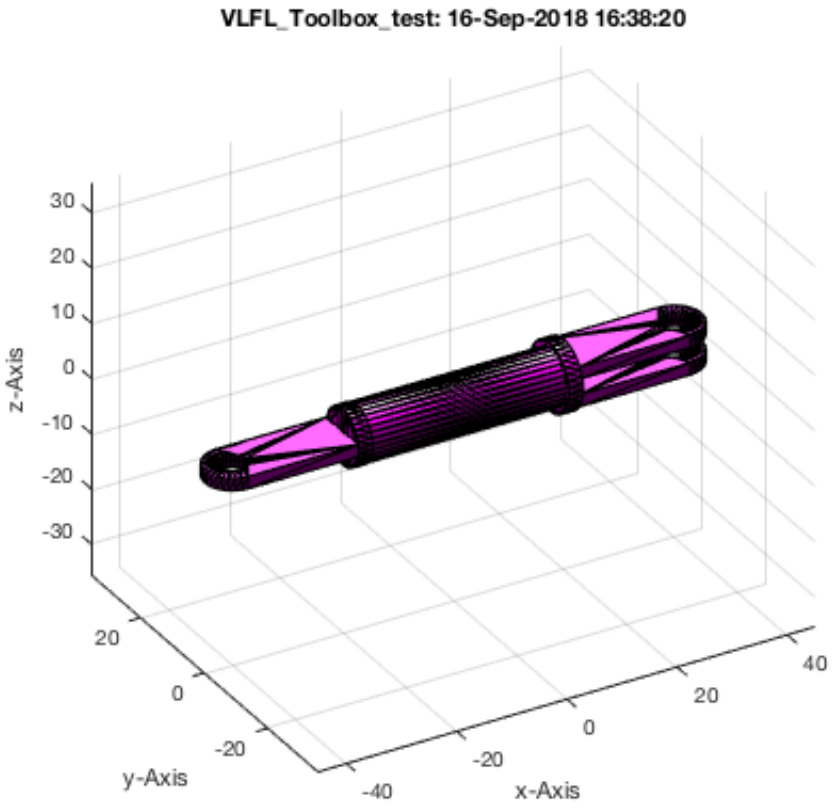
```
smbNewSystem ('SG_LIB_EXP_20');           % Creates the mechansim diagramm
smbDrawNow;
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_20/'
```

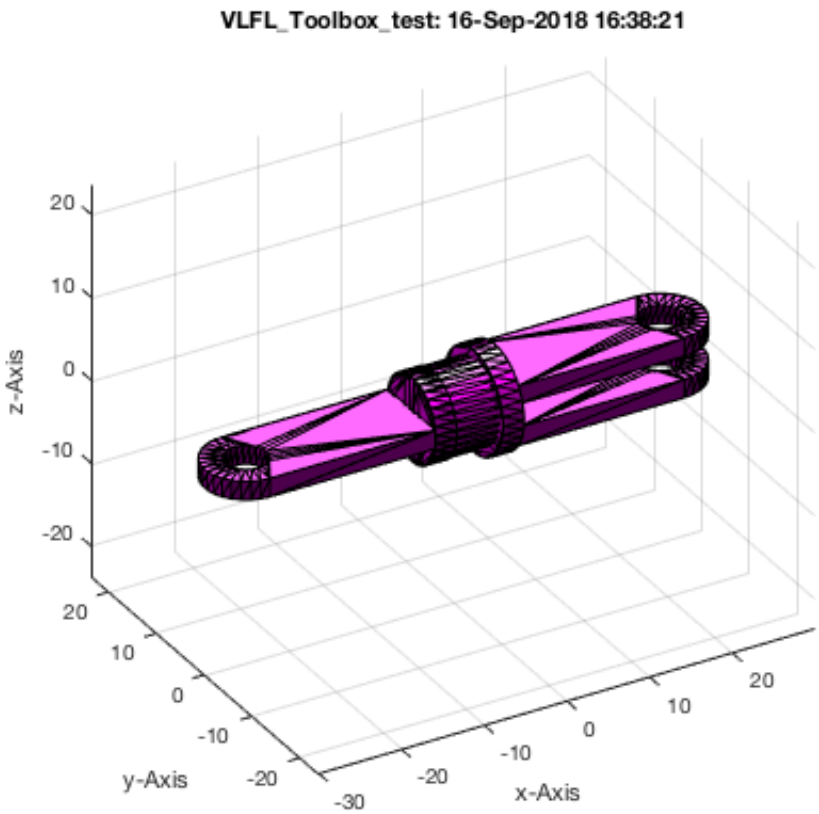
3. Create two links with length 50 and 80 and one or two mounting holes

```
SG1=SGmodelLink(80,'',1,2);               % Creates a long rod with flange
SG2=SGmodelLink(50,'',1,2);               % Creates a short rod with flange
```

```
SGfigure(SG1); view(-30,30);
```




```
SGfigure(SG2); view(-30,30);
```

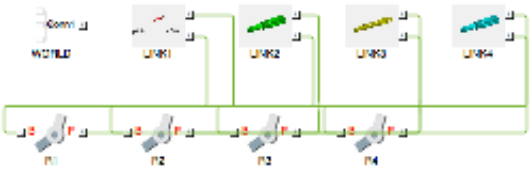


4. Create SimMechanics models for the four links in different colors

```
smbCreateSG (SG1,'LINK1','r');              % Add long rod as LINK1
smbCreateSG (SG2,'LINK2','g');              % Add short rod as LINK2
smbCreateSG (SG1,'LINK3','y');              % Add long rod as LINK3
smbCreateSG (SG2,'LINK4','c');              % Add short rod as LINK4
smbDrawNow;
```

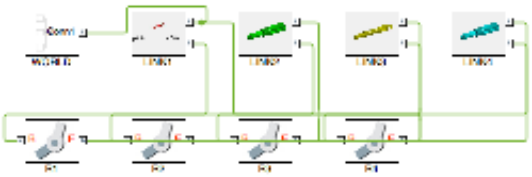
5. Create SimMechanics models for the four joint and connect them with the links

```
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
smbDrawNow;
```



6. Connect the base frame of link1 to the world coordinate system

```
smbCreateConnection('WORLD.ORIGIN','LINK1.B'); % Connect Linkage to World Frame
smbDrawNow;
```



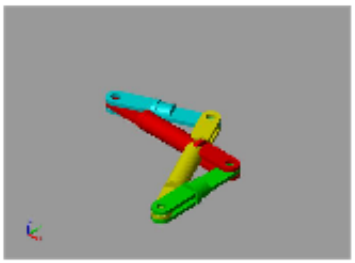
7. Run the Simulation of the Simulink/SimMechanics diagram for 1 second

```
smbSimulate(1);          % Simulate for 1 second
```

8. Create a Video of the Simualatin for 5 seconds

```
smbVideoSimulation(5);  % Show a 5 seconds video
```

```
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:38:56!
Executed 16-Sep-2018 16:38:58 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016 on 2016-12-09*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 22: Adding Simulink Signals to Record Frame Movements

2016-12-18: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.1 required)
- 2. Creating a new SimMechanics System
- 3. Create two links with length 50 and 80 and one or two mounting holes
- 4. Create SimMechanics models for the four links and four joints in different colors
- 5. Create a video of the movements
- 6. Analyze the simulation for 3 Seconds
- 7. Create Simulink signals for all the frames of the four links
- 8. Simulate and record those signals too
- Final remarks on toolbox version and execution date

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.1 required)

2. Creating a new SimMechanics System

```
smbNewSystem ('SG_LIB_EXP_22');           % Creates the mechansim diagramm
smbDrawNow;
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_22/'
```

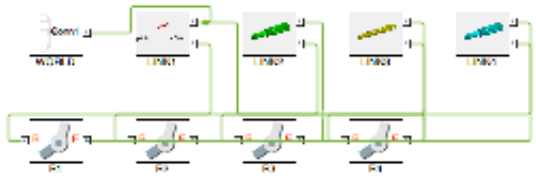
3. Create two links with length 50 and 80 and one or two mounting holes

```
SG1=SGmodelLink(80,'',1,2);               % Creates a long rod with flange
SG2=SGmodelLink(50,'',1,2);               % Creates a short rod with flange
```

4. Create SimMechanics models for the four links and four joints in different colors

```
smbCreateSG (SG1,'LINK1','r');              % Add long rod as LINK1
smbCreateSG (SG2,'LINK2','g');              % Add short rod as LINK2
smbCreateSG (SG1,'LINK3','y');              % Add long rod as LINK3
smbCreateSG (SG2,'LINK4','c');              % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
```

```
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
smbCreateConnection('WORLD.ORIGIN','LINK1.B'); % Connect Linkage to World Frame
smbDrawNow;
```



5. Create a video of the movements

smbVideoSimulation(3); % Show a 3 seconds video

6. Analyze the simulation for 3 Seconds

The result of a simulation is a strucutre that contains SimMultiBody states (xout) and recorded Simulink signals (sim). If there are no Simulink signals, sout is empty.

```
simOut=smbSimulate(3)
```

```
simOut =

  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   tout: [241x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]

      SimulationMetadata: [1x1 Simulink.SimulationMetadata]
            ErrorMessage: [0x0 char]
```

The states contain the parameter = angles/velocity of the joints

```
xout = simOut.get('xout')
```

```
xout =

Simulink.SimulationData.Dataset 'xout' with 8 elements

                        Name                    BlockPath
                        ______________________  _________________
    1  [1x1 State]      SG_LIB_EXP_22.R1.Rz.q   SG_LIB_EXP_22/R1
    2  [1x1 State]      SG_LIB_EXP_22.R1.Rz.w   SG_LIB_EXP_22/R1
```

```
  3  [1x1 State]      SG_LIB_EXP_22.R2.Rz.q  SG_LIB_EXP_22/R2
  4  [1x1 State]      SG_LIB_EXP_22.R2.Rz.w  SG_LIB_EXP_22/R2
  5  [1x1 State]      SG_LIB_EXP_22.R3.Rz.q  SG_LIB_EXP_22/R3
  6  [1x1 State]      SG_LIB_EXP_22.R3.Rz.w  SG_LIB_EXP_22/R3
  7  [1x1 State]      SG_LIB_EXP_22.R4.Rz.q  SG_LIB_EXP_22/R4
  8  [1x1 State]      SG_LIB_EXP_22.R4.Rz.w  SG_LIB_EXP_22/R4

- Use braces { } to access, modify, or add elements using index.
```

There is no Simulink signals yet

```
sout = simOut.get('sout')
```

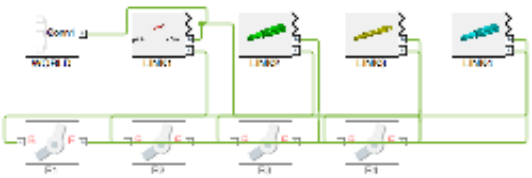
```
sout =

     []
```

7. Create Simulink signals for all the frames of the four links

```
smbAddFrameSensor ('LINK1.RF');
smbAddFrameSensor ('LINK2.RF');
smbAddFrameSensor ('LINK3.RF');
smbAddFrameSensor ('LINK4.RF');
```

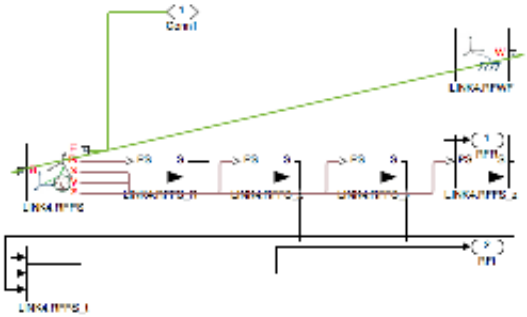
Now, all links have simulink signals and signal output for R and T of the reference frame

```
smbDrawNow;
```



The model of link4 is extendend by a transformation sensor

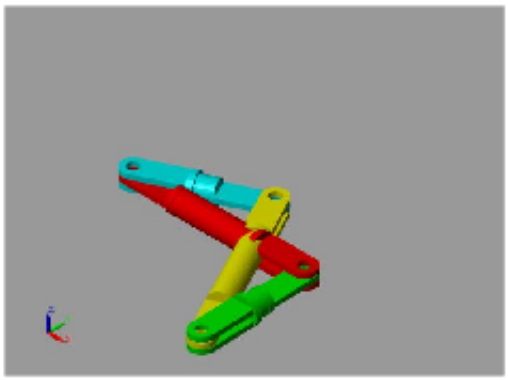
```
smbDrawNow ('LINK4.RF_T');
```



8. Simulate and record those signals too

```
simOut=smbSimulate(3)
smbVideoSimulation(3);
```

```
simOut =

  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   sout: [1x1 Simulink.SimulationData.Dataset]
                   tout: [241x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]

.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
```



The states contain the parameter = angles/velocity of the joints

```
xout = simOut.get('xout')
```

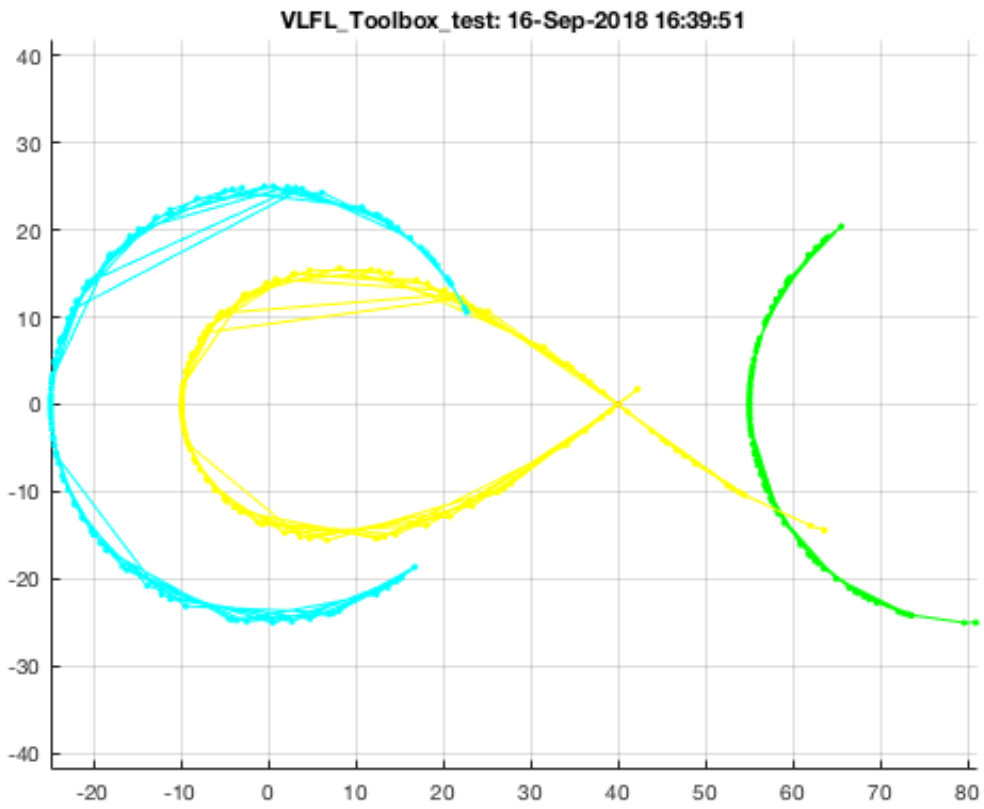
```
xout =

Simulink.SimulationData.Dataset 'xout' with 8 elements

                        Name                   BlockPath
                        _______________________ _________________
    1  [1x1 State]      SG_LIB_EXP_22.R1.Rz.q  SG_LIB_EXP_22/R1
    2  [1x1 State]      SG_LIB_EXP_22.R1.Rz.w  SG_LIB_EXP_22/R1
    3  [1x1 State]      SG_LIB_EXP_22.R2.Rz.q  SG_LIB_EXP_22/R2
    4  [1x1 State]      SG_LIB_EXP_22.R2.Rz.w  SG_LIB_EXP_22/R2
    5  [1x1 State]      SG_LIB_EXP_22.R3.Rz.q  SG_LIB_EXP_22/R3
    6  [1x1 State]      SG_LIB_EXP_22.R3.Rz.w  SG_LIB_EXP_22/R3
    7  [1x1 State]      SG_LIB_EXP_22.R4.Rz.q  SG_LIB_EXP_22/R4
    8  [1x1 State]      SG_LIB_EXP_22.R4.Rz.w  SG_LIB_EXP_22/R4

  - Use braces { } to access, modify, or add elements using index.
```

TheSimulink signals are related to the reference rotation and translation

```
sout = simOut.get('sout')
T1=smbTofSimOut(simOut,'LINK1.RF'); VL1=squeeze(T1(1:3,4,:))';
T2=smbTofSimOut(simOut,'LINK2.RF'); VL2=squeeze(T2(1:3,4,:))';
T3=smbTofSimOut(simOut,'LINK3.RF'); VL3=squeeze(T3(1:3,4,:))';
T4=smbTofSimOut(simOut,'LINK4.RF'); VL4=squeeze(T4(1:3,4,:))';
SGfigure; axis on; view(0,90); grid on;
VLplot(VL1,'r.-');
VLplot(VL2,'g.-');
VLplot(VL3,'y.-');
VLplot(VL4,'c.-');
drawnow;
```

```
sout =
```

```
Simulink.SimulationData.Dataset 'sout' with 8 elements

                          Name                                BlockPath
                          ____________________________________  ____________________________________

    1  [1x1 Signal]       SG_LIB_EXP_22/LINK1/LINK1.RF_T.RFR  SG_LIB_EXP_22/LINK1/LINK1.RF_T
    2  [1x1 Signal]       SG_LIB_EXP_22/LINK1/LINK1.RF_T.RFt  SG_LIB_EXP_22/LINK1/LINK1.RF_T
    3  [1x1 Signal]       SG_LIB_EXP_22/LINK2/LINK2.RF_T.RFR  SG_LIB_EXP_22/LINK2/LINK2.RF_T
    4  [1x1 Signal]       SG_LIB_EXP_22/LINK2/LINK2.RF_T.RFt  SG_LIB_EXP_22/LINK2/LINK2.RF_T
    5  [1x1 Signal]       SG_LIB_EXP_22/LINK3/LINK3.RF_T.RFR  SG_LIB_EXP_22/LINK3/LINK3.RF_T
    6  [1x1 Signal]       SG_LIB_EXP_22/LINK3/LINK3.RF_T.RFt  SG_LIB_EXP_22/LINK3/LINK3.RF_T
    7  [1x1 Signal]       SG_LIB_EXP_22/LINK4/LINK4.RF_T.RFR  SG_LIB_EXP_22/LINK4/LINK4.RF_T
    8  [1x1 Signal]       SG_LIB_EXP_22/LINK4/LINK4.RF_T.RFt  SG_LIB_EXP_22/LINK4/LINK4.RF_T

  - Use braces { } to access, modify, or add elements using index.
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:39:51!
```

```
Executed 16-Sep-2018 16:39:53 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

- *Tim Lueth, tested and compiled on OSX 10.11.6 with Matlab 2016b on 2016-12-18*
- _____, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-xx-xx_

Published with MATLAB® R2018a

Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model

2016-12-19: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.1 required)
- 2. Open a system and create several fixed nodes and attach revolute joints
- 3. Create a cylindric joint from two solids an attach it to revolute joint
- 4. Attach two frame sensor to record the movement of the falling cylinder
- 5. Show the Simulation
- 6. Now create a solid between the revolute joint and cylindric joint
- 7. Now connect the new solid in the model
- 8. Show the Simulation: The Mechnism has no Movement anymore
- 9. Now Create a Solid Model of Movement Status at Time = 0.1 Seconds
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)

- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

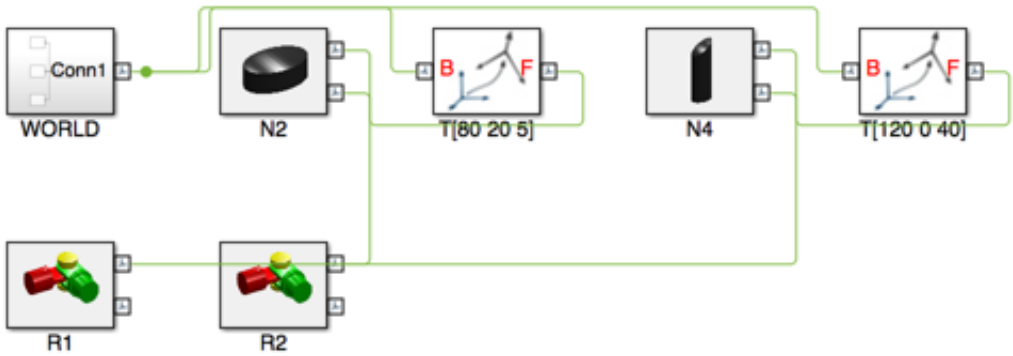
Motivation for this tutorial: (Originally SolidGeometry 3.1 required)

2. Open a system and create several fixed nodes and attach revolute joints

```
function VLFL_EXP23
```

```
smbNewSystem ('SG_LIB_EXP_23');

smbCreateSGNode ([80  20  5],'N2');
smbCreateSGNode ([120 0 40],'N4','',rot(0,-pi/8,0));
A=SGmodelJoint('R',pi/2);
smbCreateSGJoint('R','R1', A,'N4.F');
smbCreateSGJoint('R','R2',A,'N2.F');
smbDrawNow;
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/'
```

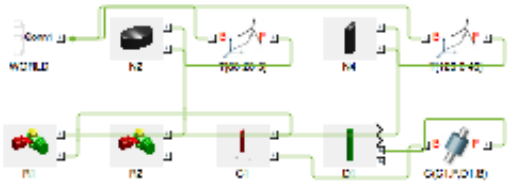


3. Create a cylindric joint from two solids an attach it to revolute joint

```
Ro=5;
Ri=3;
slot=0.3;

C1=SGofCPLz([PLcircle(Ro);NaN NaN;PLcircle(Ri+slot)],30);
% C1=SGTset(C1,'B',TofSG(C1,'bottom','roty',pi));
C1=SGTset(C1,'B',TofSG(C1,'incenter','right',-1,'roty',pi/2));
C1=SGTset(C1,'F',TofSG(C1,'bottom'));
smbCreateSG(C1,'C1','r','R1_M');
D1=SGofCPLz(PLcircle(3),30);
D1=SGTset(D1,'B',TofSG(D1,'incenter'));
D1=SGTset(D1,'F',TofSG(D1,'top'));
```

4. Attach two frame sensor to record the movement of the falling cylinder

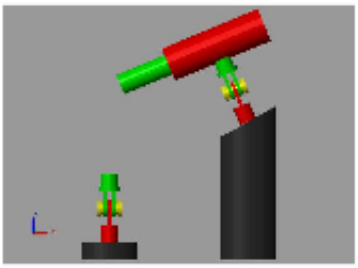
```
smbCreateSG(D1,'D1','g');
smbCreateConnection('C1.F','D1.B','C');
smbAddFrameSensor('R2_M.F');
smbAddFrameSensor('D1.F');
smbDrawNow;
```



5. Show the Simulation

```
simOut=smbSimulate(0.1);
smbVideoSimulation(1);
```

```
..
```



6. Now create a solid between the revolute joint and cylindric joint

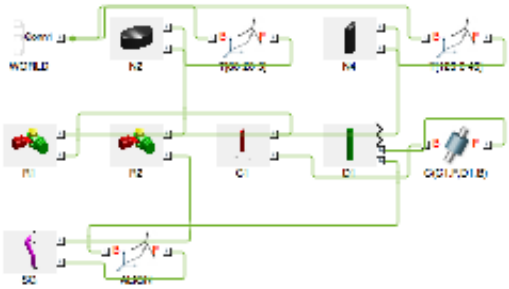
```
[T,ta]=smbTofSimOut(simOut,'R2_M.F'); T1=squeeze(T(:,:,1));
[T,tb]=smbTofSimOut(simOut,'D1.F');   T2=squeeze(T(:,:,1));
SG=SGof2T(T1,T2*TofR(rot(0,pi,0)),'',4); % Radius 4
SGTplot(SG);
```

```
VLradialEdges: Radius 4.00 reduced to 2.73
```

7. Now connect the new solid in the model

```
smbCreateSG(SG,'SG','m');
```

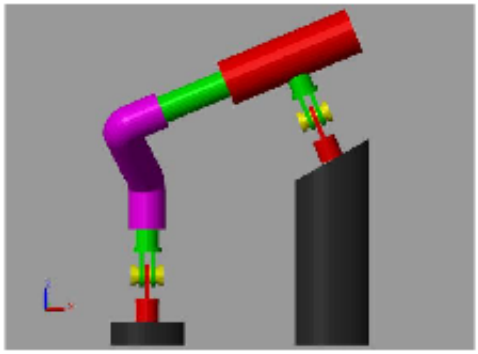
```
smbCreateConnection('R2_M','SG.B');
smbCreateConnection('D1.F','SG.F','align');
smbDrawNow;
```



8. Show the Simulation: The Mechnism has no Movement anymore

```
smbVideoSimulation(1);
```

```
..
```



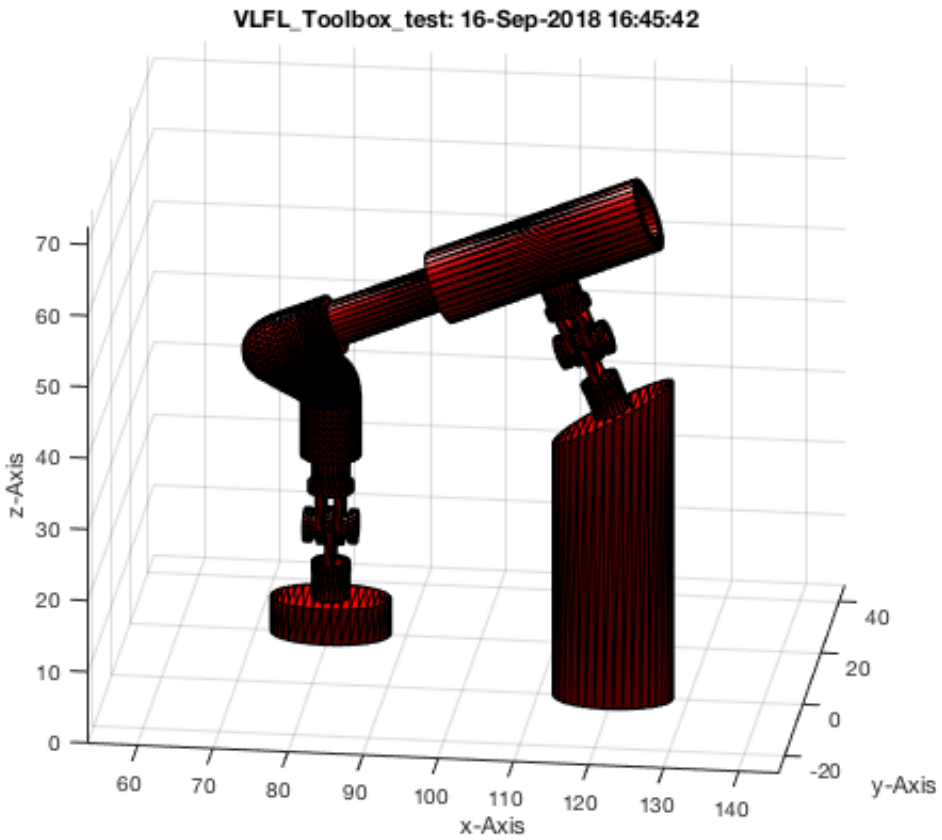
9. Now Create a Solid Model of Movement Status at Time = 0.1 Seconds

```
SG=smbFullModelSimulation(0.1);
SGfigure; SGplot(SG); view (7,20);
```

```
CREATING A FULL SOLID-MOVEMENT SIMULATION-MODEL 'SG_LIB_EXP_23' THAT RUNS At LEAST 0.10 SEC
ONDS
==========================================================================================
========
Adding frame sensors for all solids of the model
Add frame sensors for 'C1.SG'
Add frame sensors for 'D1.SG'
Add frame sensors for 'N2.SG'
Add frame sensors for 'N4.SG'
```

```
Add frame sensors for 'R1.FIX1.SG'
Add frame sensors for 'R1_M.SG'
Add frame sensors for 'R1_S.SG'
Add frame sensors for 'R2.FIX1.SG'
Add frame sensors for 'R2_M.SG'
Add frame sensors for 'R2_S.SG'
Add frame sensors for 'SG.SG'
==========================================================================================
========

simOut =

  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   sout: [1x1 Simulink.SimulationData.Dataset]
                   tout: [51x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]

LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp_C1
.stl
Header:
Number of facets: 232
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_D1.stl
Header:
Number of facets: 96
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_N2.stl
Header:
Number of facets: 156
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_N4.stl
Header:
Number of facets: 156
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R1.FIX1.stl
Header:
Number of facets: 240
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R1_M.stl
Header:
Number of facets: 456
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R1_S.stl
Header:
Number of facets: 448
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R2.FIX1.stl
Header:
Number of facets: 240
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R2_M.stl
Header:
Number of facets: 456
```

```
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_R2_S.stl
Header:
Number of facets: 448
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_23/sbm_temp
_SG.stl
Header:
Number of facets: 3824
0..
CREATED A SOLID GEOMETRY OF THE FULL SIMULATION-MODEL 'SG_LIB_EXP_23' AT TIME: 0.10 SECONDS
==========================================================================================
========
```



Write the STL file on disk for 3D printing

```
SGwriteSTL(SG);
```

Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:45:42!
```

```
Executed 16-Sep-2018 16:45:45 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
========================================= Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
=================================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 24: Automatic Creation of a Joint Limitations

2016-12-25: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.2 required)
- 2. Open a system and create several fixed nodes and attach revolute joints
- 3. Create a cylindric joint from two solids an attach it to revolute joint
- 4. Attach two frame sensor to record the movement of the falling cylinder
- 5. Show the Simulation
- 6. Install additional block funktion for joint restrictions
- 7. Create a stopp joint and copy all connections of an existing joint
- 8. Create a stopp joint and replace an existing joint
- 9. Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

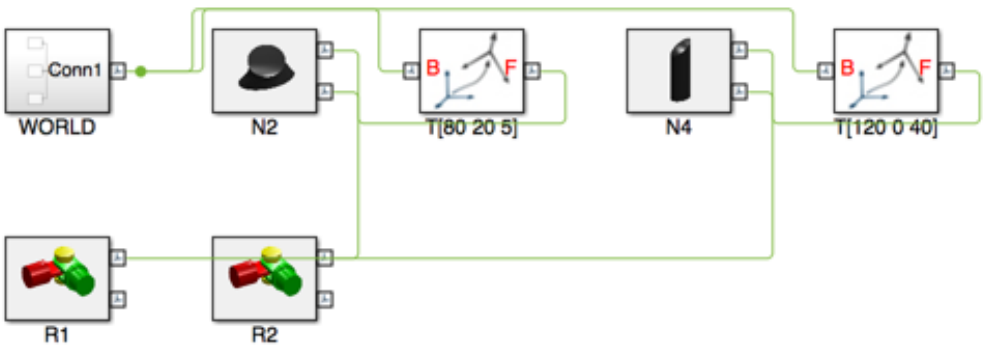
Motivation for this tutorial: (Originally SolidGeometry 3.2 required)

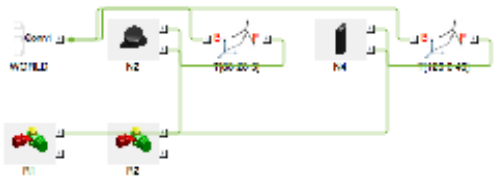
2. Open a system and create several fixed nodes and attach revolute joints

function VLFL_EXP24

```
smbsys='SG_LIB_EXP_24';
smbNewSystem (smbsys);

smbCreateSGNode ([80  20  5],'N2','',rot(0,0,pi/3));
smbCreateSGNode ([120 0 40],'N4','',rot(0,-pi/8,0));
A=SGmodelJoint('R',pi/2);
smbCreateSGJoint('R','R1', A,'N4.F');
smbCreateSGJoint('R','R2',A,'N2.F');
smbDrawNow;
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/'
```



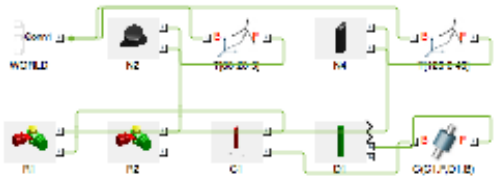


3. Create a cylindric joint from two solids an attach it to revolute joint

```
Ro=5;
Ri=3;
slot=0.3;

C1=SGofCPLz([PLcircle(Ro);NaN NaN;PLcircle(Ri+slot)],30);
% C1=SGTset(C1,'B',TofSG(C1,'bottom','roty',pi));
C1=SGTset(C1,'B',TofSG(C1,'incenter','right',-1,'roty',pi/2));
C1=SGTset(C1,'F',TofSG(C1,'bottom'));
smbCreateSG(C1,'C1','r','R1_M');
D1=SGofCPLz(PLcircle(3),30);
D1=SGTset(D1,'B',TofSG(D1,'incenter'));
D1=SGTset(D1,'F',TofSG(D1,'top'));
```

4. Attach two frame sensor to record the movement of the falling cylinder

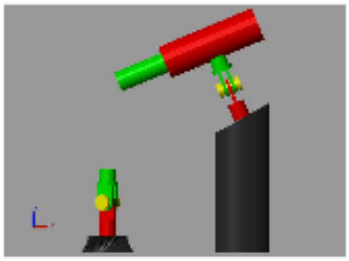
```
smbCreateSG(D1,'D1','g');
smbCreateConnection('C1.F','D1.B','C');
smbAddFrameSensor('R2_M.F');
smbAddFrameSensor('D1.F');
smbDrawNow;
```



5. Show the Simulation

```
simOut=smbSimulate(0.1);
smbVideoSimulation(4);
```

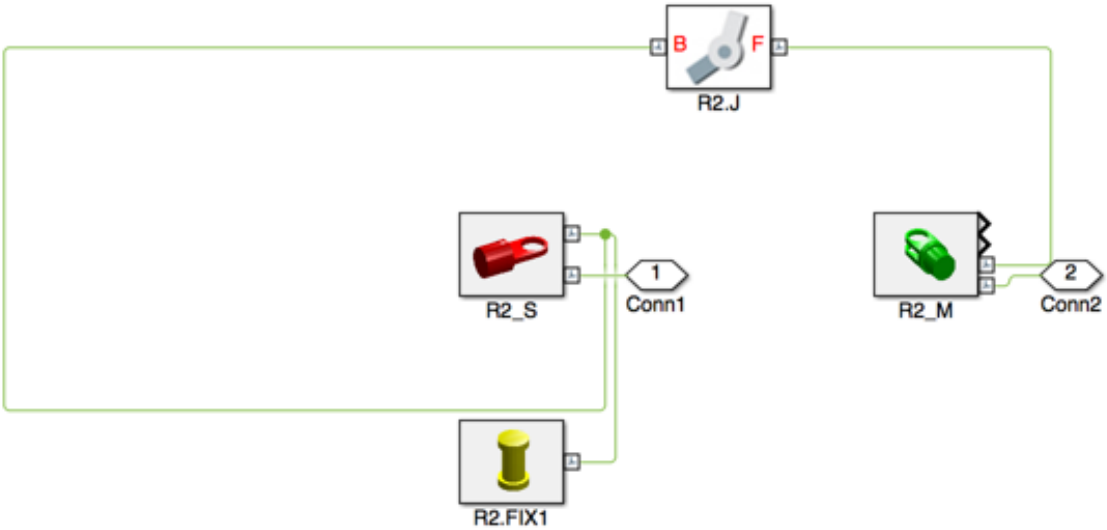
...

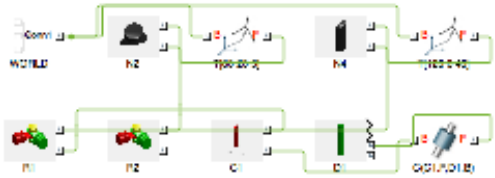


6. Install additional block funktion for joint restrictions

```
smbPSLibInstall
open_system(smbPSBlockname);
open_system(smbsys,'tab');
open_system(smbWhich('R2'),'tab');smbDrawNow;
```

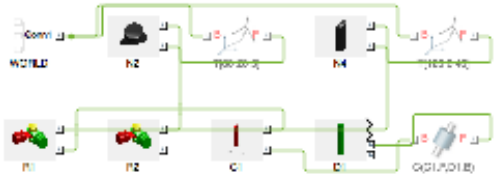
```
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_frict
ion_rot.ssc
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_frict
ion_rot.svg
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hards
top_rot.ssc
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hards
top_rot.svg
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_frict
ion_trans.ssc
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_frict
ion_trans.svg
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hards
top_trans.ssc
Create /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hards
top_trans.svg
Generating Simulink library 'mechPS_Tim_Lueth_lib' in the current directory '/Users/timlueth/D
esktop/Toolbox_test/tmp_SG_LIB_EXP_24' ...
```





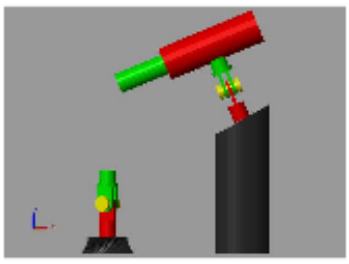
7. Create a stopp joint and copy all connections of an existing joint

```
smbCreateStopJointR ('R2stop.J',[-pi/2 +pi/2]);
smbCopyConnections  ('R2.J','R2stop.J');
smbDrawNow;
```



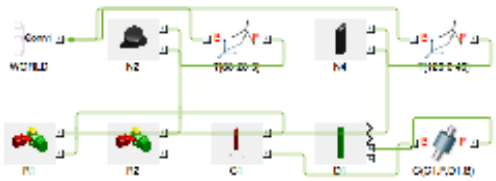
```
smbVideoSimulation(4);
```

...

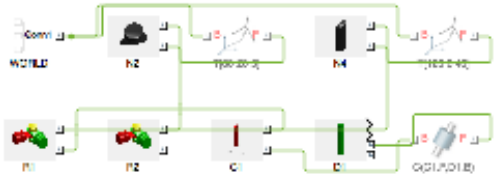


8. Create a stopp joint and replace an existing joint

```
delete_block(smbWhich('R2stop.J'));
smbDeleteUnconnectedLines;
smbDrawNow;
```



```
smbCreateStopJointR ('R2new.J',[-pi/2 +pi/2]);
smbCopyConnections  ('R2.J','R2new.J','replace');
smbDrawNow;
```



9. Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:46:36!
Executed 16-Sep-2018 16:46:38 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on a M
ACI64
======================================== Used Matlab products: ================================
=======
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
=======
```

Published with MATLAB® R2018a

Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages

2017-01-01: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.2 required)
- 3. Create a video clip for a text title
- 4. Create an end title video clip
- 5. Create a text page title for a video
- 6. Now create a SimMultiBody fourbar linkage
- 7. Create a video simulation and creates header and titles
- Now we create four small video clips in the desktopdir
- 8. Create Video Headers and Explaination
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages

- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.2 required)

The creation of videos of the simulated multi body system, created with the SimMultiBody (2nd Generation) is essential for documentation of the results. Nevertheless, without video titles and end titles and some pages with text description, the videos cannot fulfill their purpose. Therefor it is essential also, to create text for explaining content and make comments on authors and creation date. Therefor we will show some examples:

- imageVideoFrames(xy,ptime); cuts images out of a video a defined positions
- imageVideoTitle(xy,STitles,cols,ptime), creates a 2 second title page video
- imageVideoEndtitle(xy,ETitles,cols), creates a 1 second end title page video
- imageVideoTextPage(xy,ETitles,cols), creates a 2 second text page video
- imageVideoWrite (v,I,t), creates a video by repeating an image t frames
- videoCopyFrames(v,vr), copies a video content into another video (no sound)
- videoCopyCutMovies (WName,RName,style), complex cutting function

3. Create a video clip for a text title

It is possible to start by defining the size of the video titles. If the function is called without am output parameter, automatically a video clip is created with this image

```
I=imageVideoTitle([640 480],{'Video Titel','$date'});
imshow(I.cdata);
```

It is also possible to name an existing video or to select it during function execution to define the size from an existing video. In addition, the background color and text color can be defined (in future also font name and font size), and furthermore times for creating a snapshot that becomes part of the title page. I=imageVideoTitle('',{'Video Titel','SubTitle','Author','$date'},['w' 'r'],[0 1 3]);

```
close all; figure; imshow(I.cdata);
```

Calling the function without an output parameter creates a small video clip in the desktopdir. This can later be used to add the original video including text pages, title page and end title page to a video clip that has sound. Matlab in 2016b does not support sound videos on MAC, only on PC platforms. imageVideoTitle('',{'Video Titel','SubTitle','Author','$date'},['w' 'r'],[0 1 3]);

4. Create an end title video clip

In similar manner, it is possible to define end titles. The creation date is added automatically. Please use the title page if you want to clarify the result was achieved earlier.

```
I=imageVideoEndtitle([640 480],{'Technical University of Munich','','www.tum.de'});
imshow(I.cdata);
imageVideoEndtitle([640 480],{'Technical University of Munich','','www.tum.de'}); % write video clip
```

```
Creating a new video file: '/Users/timlueth/Desktop/Toolbox_test/imageVideoEndtitle.avi'
```

5. Create a text page title for a video

There are several reasons for adding text pages including latex equations too. This is also possible by a toolbox function. Again, the call without an output parameter would create a video clip.

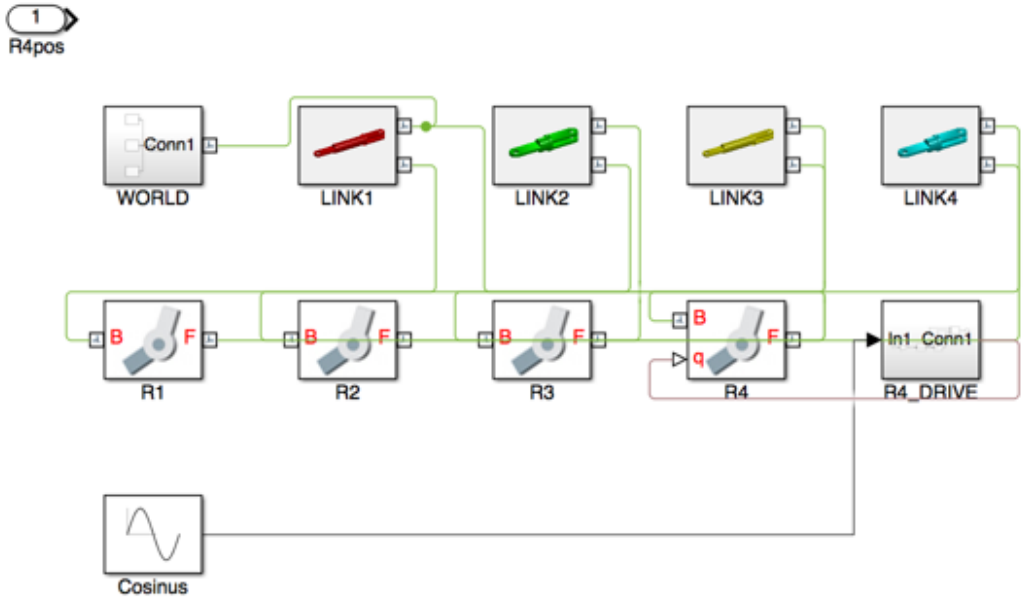
```
I=imageVideoTextPage([640 480],...
['It is also possible to name an existing video or to select it during function '...
'execution to define the size from an existing video. In addition, the '...
'background color and text color can be defined (in future also font name and '...
'font size), and furthermore times for creating a snapshot that becomes part of '...
'the title page.', char(13), '©']);
imshow(I.cdata);
```

It is also possible to name an existing video or to select it during function execution to define the size from an existing video. In addition, the background color and text color can be defined (in future also font name and font size), and furthermore times for creating a snapshot that becomes part of the title page.
©

6. Now create a SimMultiBody fourbar linkage

```
smbNewSystem ('SG_LIB_EXP_25');           % Creates the mechansim diagramm
SG1=SGmodelLink(80,'',1,2);               % Creates a long rod with flange
SG2=SGmodelLink(50,'',1,2);               % Creates a short rod with flange
smbCreateSG (SG1,'LINK1','r');              % Add long rod as LINK1
smbCreateSG (SG2,'LINK2','g');              % Add short rod as LINK2
smbCreateSG (SG1,'LINK3','y');              % Add long rod as LINK3
smbCreateSG (SG2,'LINK4','c');              % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
smbCreateConnection('WORLD.ORIGIN','LINK1.B'); % Connect Linkage to World Frame
smbCreateDrive ('R4');                         % Convert Joint R4 into a Drive
smbCreateSineWave ('Cosinus','R4_DRIVE/1');    % Connect a Sinus Generator to Drive
smbDrawNow;
```

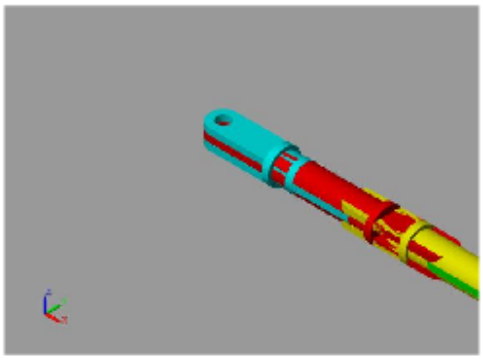
```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_25/'
```



7. Create a video simulation and creates header and titles

```
[I,FN]=smbVideoSimulation (10);
```

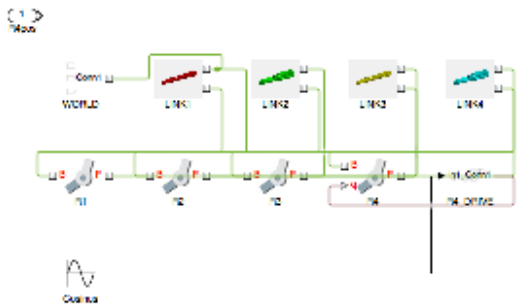
```
........
```



Now we create four small video clips in the desktopdir

```
imageVideoTitle(FN,{'SG-Lib Tutorial #25','Creating Videos Titles for SimMultiBody Videos','Tim C. Lu
eth','$date'},'',[2 5]);
imageVideoEndtitle(FN,{'Technical University of Munich','','www.tum.de'});
imageVideoTextPage(FN,{...
['This video was created by using Mathwork''s SimMultiBody environment using the '...
'SG-Library of Tim C. Lueth. The fourbar linkage in the simulation has the '...
'following dimensions:©', char(13), ' '...
'L1= ', sprintf('%.2f mm',80), char(13), ' '...
'L2= ', sprintf('%.2f mm',50),char(13), ' '...
'L3= ', sprintf('%.2f mm',80),char(13), ' '...
'L4= ', sprintf('%.2f mm',50),char(13), '']});
imageVideoImagePage(FN,smbDrawNow);
```

```
Creating a new video file: '/Users/timlueth/Desktop/Toolbox_test/imageVideoTitle.avi'
Creating a new video file: '/Users/timlueth/Desktop/Toolbox_test/imageVideoEndtitle.avi'
Creating a new video file: '/Users/timlueth/Desktop/Toolbox_test/imageVideoTextPage.avi'
Creating a new video file: '/Users/timlueth/Desktop/Toolbox_test/imageVideoImagePage.avi'
```



8. Create Video Headers and Explaination

```
[I,FN]=smbVideoSimulation (10,'Video SG_LIB_EXP_25');
IT=imageVideoTitle(FN,{'SG-Lib Tutorial #25','Creating Videos Titles for SimMultiBody Videos','Tim C.
 Lueth','$date'},'',[2 5]);
IE=imageVideoEndtitle(FN,{'Technical University of Munich','','www.tum.de'});
ID=imageVideoTextPage(FN,{...
['This video was programmatically created by using Mathwork''s SimMultiBody environment using the '..
.
'SG-Library of Tim C. Lueth. The fourbar linkage in the simulation has the '...
'following dimensions:©', char(13), ' '...
'L1= ', sprintf('%.2f mm',80), char(13), ' '...
'L2= ', sprintf('%.2f mm',50),char(13), ' '...
'L3= ', sprintf('%.2f mm',80),char(13), ' '...
'L4= ', sprintf('%.2f mm',50),char(13), '']});
IM=imageVideoImagePage(FN,smbDrawNow);
videoWriteClipMovie(smbFilename('Video comp SG_LIB_EXP_25.avi'),IT,2,ID,5,IM,5,smbFilename('Video SG_
LIB_EXP_25.avi'),IE,1);
```

```
........Creating a new video file (NO SOUND/2016b): '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_
EXP_25/Video comp SG_LIB_EXP_25.avi'
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```

Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:47:50!
Executed 16-Sep-2018 16:47:52 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on a MACI64
======================================== Used Matlab products: ========================================
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==================================================================================================
```

Published with MATLAB® R2018a

Tutorial 26: Create Mechanisms using Universal Planar Links

2017-01-20: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.3 required)
- Motivation for this tutorial
- 1. Create a SimMultiBody System for a Fourbar-Linkage
- 2. Now Create a Specific Configuration (Pose) and Write a STL-Files
- 3. Now Analyze the Stucture and Group the Solids to Parts
- 4. Now Arrange all Parts for Printing as Separated Solids
- 5. Now Write the Separated Parts into Different STL Files
- 6. Create a Video of the Linkage Simulation
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.3 required)

```
% function VLFL_EXP26
```

Motivation for this tutorial

A mechanism consists of two basic elements: a) joints and b) links that connect these joints. In the "automatic construction" of mechanisms, it is helpful to limit one of the two elements. This has already been used in the previous tutorials. In this tutorial a new procedure is presented. They are "universal planar links". These consist of a simple joint member and two halves of a rotary joint. If two links are connected to each other at one of the end points, the two halves of the joints are connected to an axis of rotation due to a spatial overlap, and a swivel joint is automatically formed. If a member is not connected, an axis of rotation is still retained there. Each axis of rotation can be connected with "knobs or drive mechanisms relative to the joint and its angular range can be restricted, the links can be connected in fixed planes, allowing a collision-free movement considering the links as well as the consideration of drive elements. This tutorial now shows you how to use the universal planar links in a simple example.

```
% clear all;
```

1. Create a SimMultiBody System for a Fourbar-Linkage

```
smbNewSystem ('SG_LIB_EXP_26')   % Creates the mechansim diagramm

L1=75;
L2=60;
L3=50;
L4=50;

L1=75; A=SGmodelLink2(L1,0,1,'BL,FL'); A.col='r';
L2=60; B=SGmodelLink2(L2,0,1);         B.col='g';
L3=50; C=SGmodelLink2(L3,0,-1);        C.col='y';
```

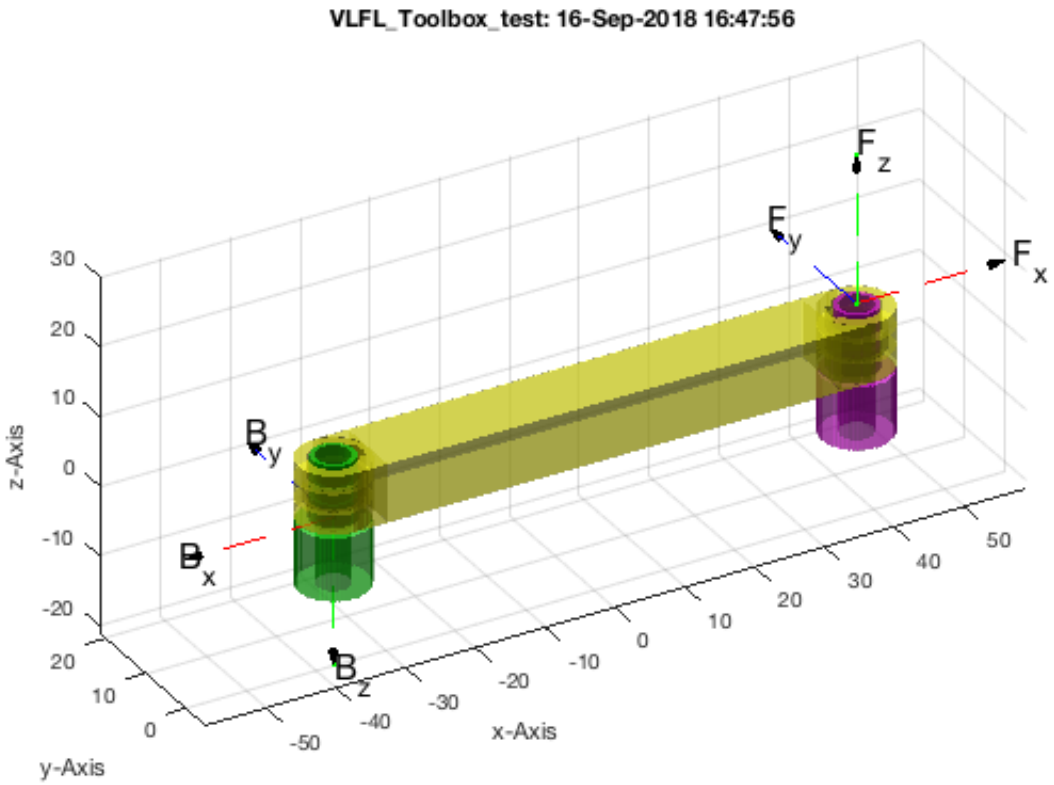
```
L4=50;D=SGmodelLink2(L4,0,-1);          D.col='m';
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_26/'
```

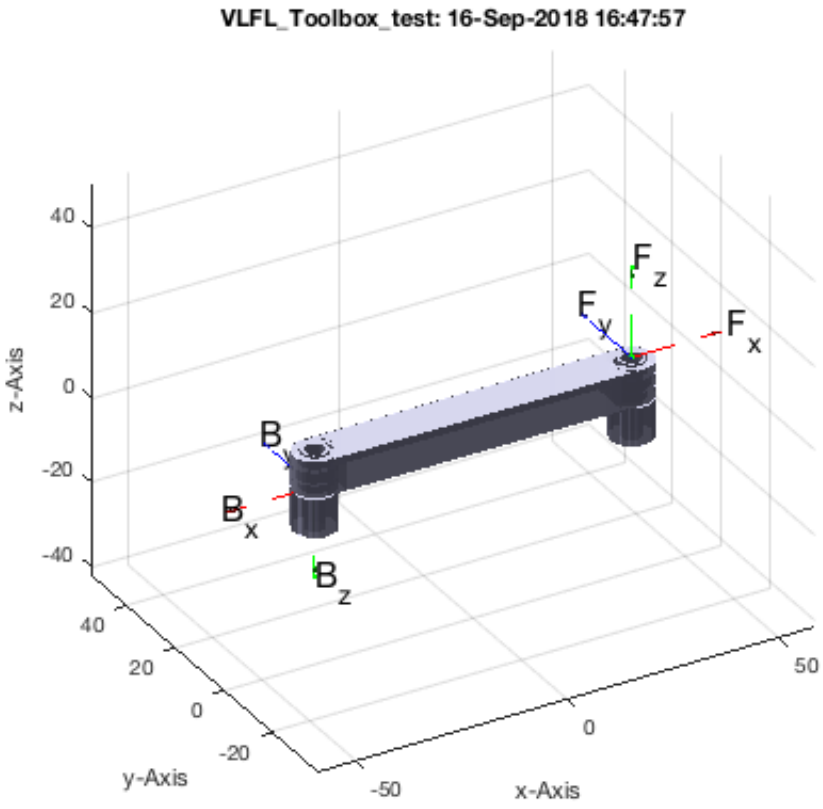
Show the components of the link

```
SGanalyzeGroupParts(A); SGTframeplot(A);
```

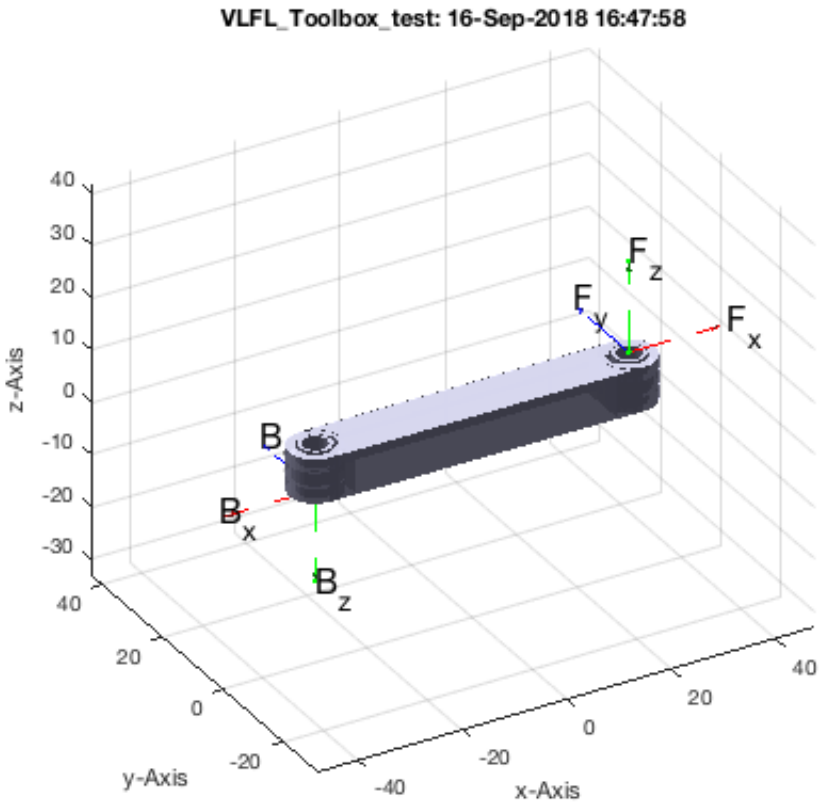
```
8% 12% 16% 20% 24% 28% 32% 36% 40% 44% 48% 52% 56% 60% 64% 68% 72% 76% 80% 84% 88% 92% 96%
100%
SGanalyzeGroupParts: 3 separated parts found.
```



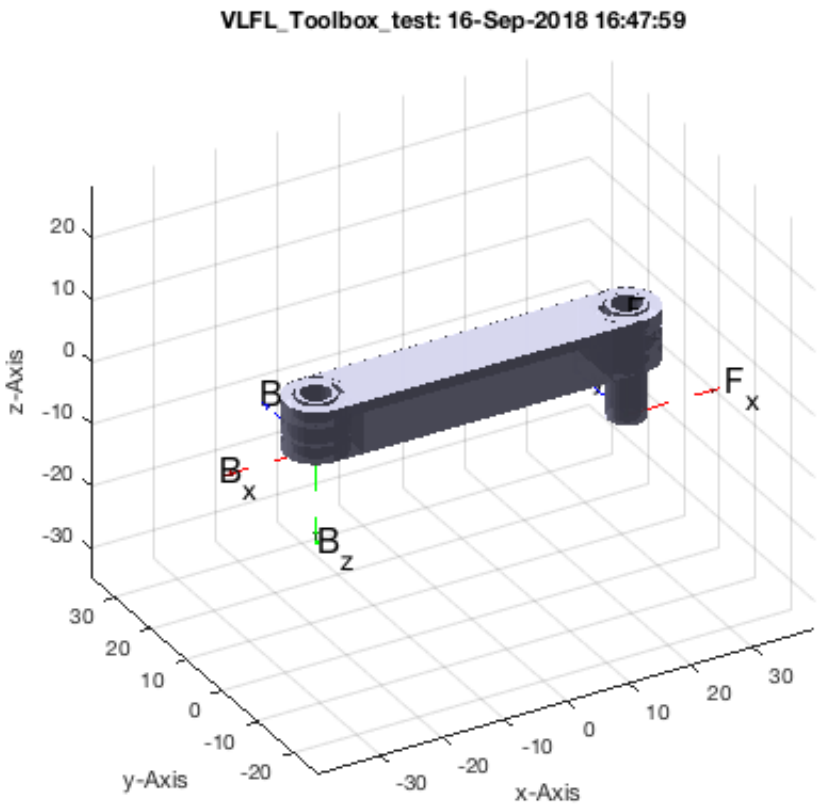

```
SGfigure; SGTplot(A); view(-30,30);
```



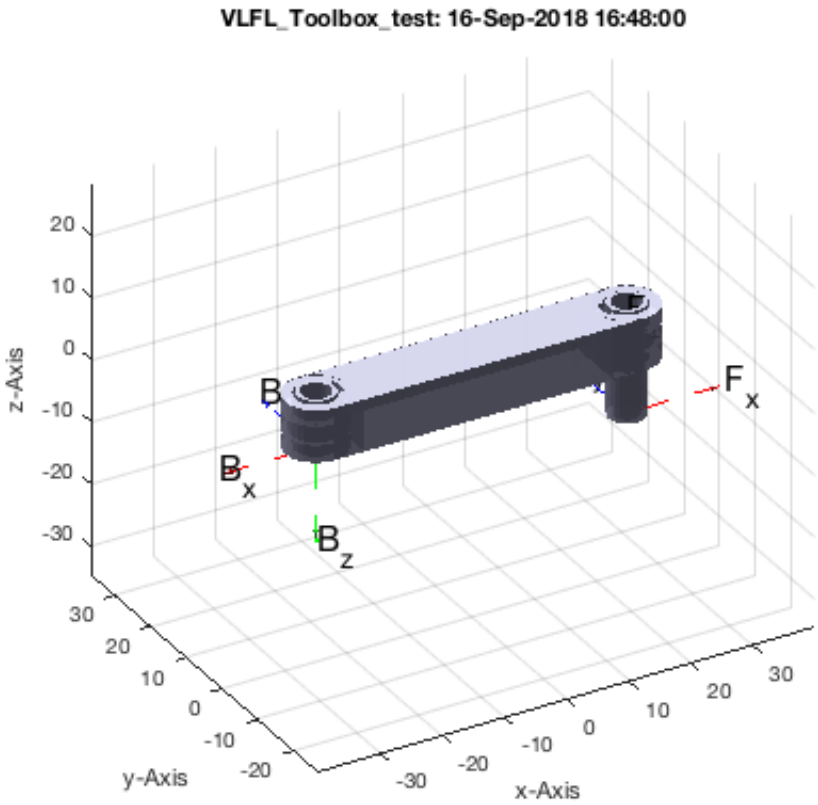

```
SGfigure; SGTplot(B); view(-30,30);
```



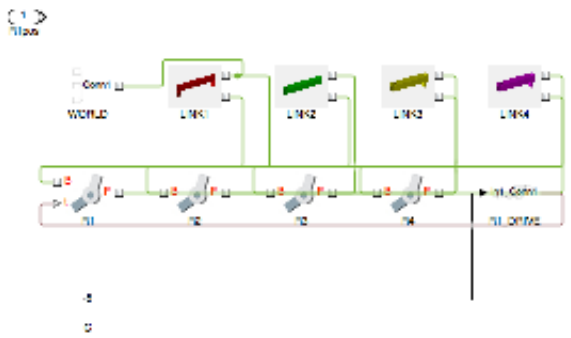

```
SGfigure; SGTplot(C); view(-30,30);
```




```
SGfigure; SGTplot(D); view(-30,30);
```




```
smbCreateSG (A,'LINK1','r');                % Add long rod as LINK1
smbCreateSG (B,'LINK2','g');                % Add short rod as LINK2
smbCreateSG (C,'LINK3','y');                % Add long rod as LINK3
smbCreateSG (D,'LINK4','m');                % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint

smbCreateConnection('WORLD.ORIGIN','LINK1.B'); % Connect Linkage to World Frame
smbCreateDrive ('R1');
smbSetJointInputTorque('R1');
smbCreateBlockConst('C','R1_DRIVE/1',-5)
ID=smbDrawNow;
smbSimulate(4);
```

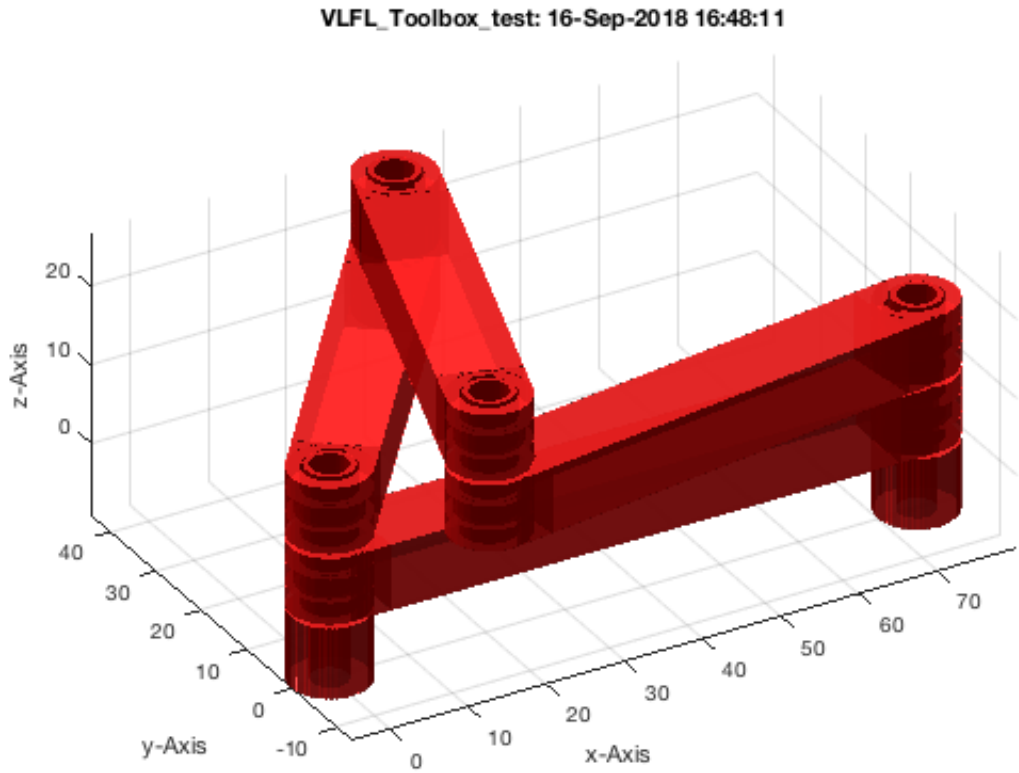


2. Now Create a Specific Configuration (Pose) and Write a STL-Files

```
SG=smbFullModelSimulation(5);
% SG=SGmagnifyVL(SG,'',[100 100 100]);
SGwriteSTL(SG,smbFilename('Universal Planar Link'));
```

```
CREATING A FULL SOLID-MOVEMENT SIMULATION-MODEL 'SG_LIB_EXP_26' THAT RUNS At LEAST 5.00 SEC
ONDS
==========================================================================================
========
Adding frame sensors for all solids of the model
Add frame sensors for 'LINK1.SG'
Add frame sensors for 'LINK2.SG'
Add frame sensors for 'LINK3.SG'
Add frame sensors for 'LINK4.SG'
==========================================================================================
========

simOut =

  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   sout: [1x1 Simulink.SimulationData.Dataset]
                   tout: [1000x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]

LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_26/sbm_temp_LI
NK1.stl
Header:
Number of facets: 3756
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_26/sbm_temp
_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_26/sbm_temp
_LINK3.stl
Header:
```

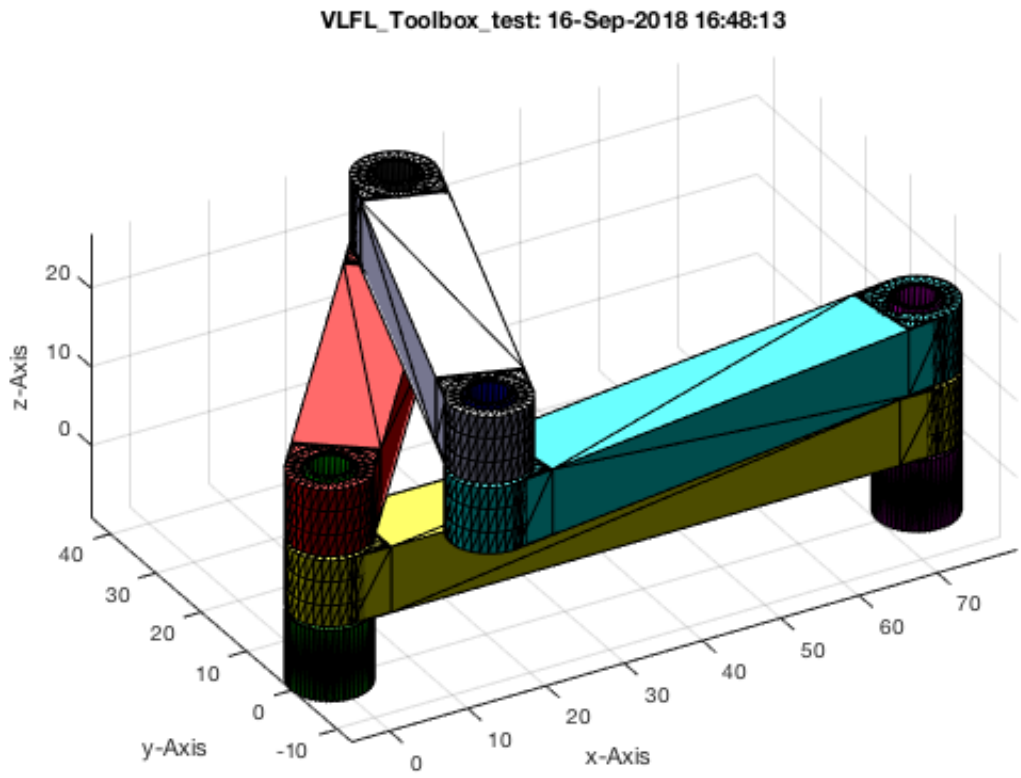
```
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_26/sbm_temp
_LINK4.stl
Header:
Number of facets: 2584
0..
CREATED A SOLID GEOMETRY OF THE FULL SIMULATION-MODEL 'SG_LIB_EXP_26' AT TIME: 5.00 SECONDS
==========================================================================================
========
1000..2000..3000..4000..5000..6000..7000..8000..9000..10000..11000..
```



3. Now Analyze the Stucture and Group the Solids to Parts

```
SGN=SG;
SGfigure; view(-30,30); SGplot(SG,'m');  SG=SGanalyzeGroupParts(SG); SGplot(SG);
```

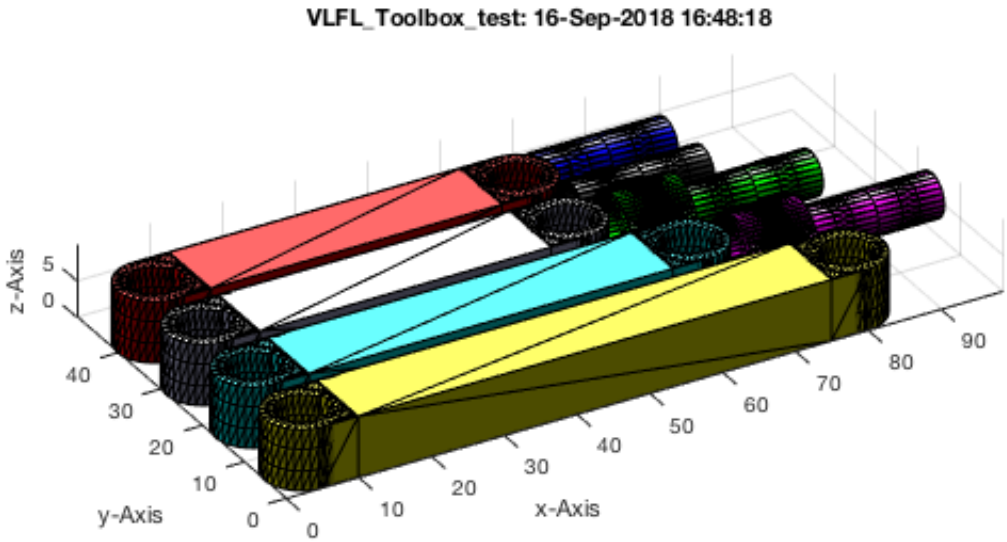
```
4% 8% 12% 16% 20% 24% 28% 32% 36% 40% 44% 48% 52% 56% 60% 64% 68% 72% 76% 80% 84% 88% 92% 9
6% 100%
```



4. Now Arrange all Parts for Printing as Separated Solids

```
[~,SG]=SGpacking(SG); SGfigure; view(-30,30); SGplot(SG);
```

```
Packing 8 objects (h=66):
```



5. Now Write the Separated Parts into Different STL Files

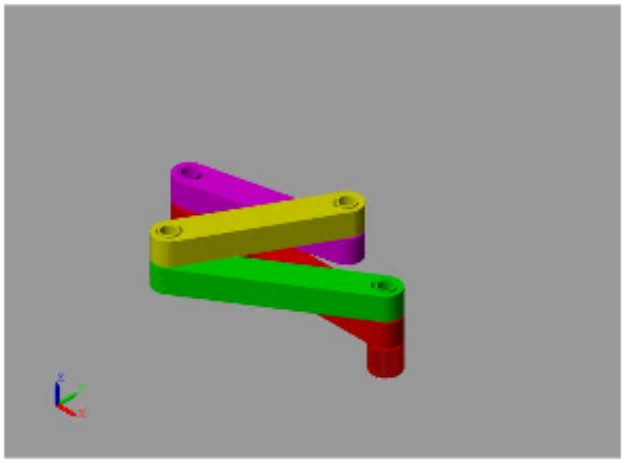
```
SGwriteSeparatedSTL(SG);
```

```
SGwriteSeparatedSTL: Writing 8 STL files in /Users/timlueth/Desktop/Toolbox_test/EXP-2018-0
9-16/
```

6. Create a Video of the Linkage Simulation

```
[I1,FN]=smbVideoSimulation (4);    % Simulate for 1 second
IT=imageVideoTitle(FN,{'SG-Lib Tutorial #26','Universal Planar Links','Tim C. Lueth','$date
'},'',[0.1 0.2 0.3]);
IE=imageVideoEndtitle(FN);
videoWriteClipMovie(smbFilename('Universal Planar Links SimMultiBody.avi'),IT,2,ID,1,FN,IE,
1);
imshow(I1);
```

```
...Creating a new video file (NO SOUND/2016b): '/Users/timlueth/Desktop/Toolbox_test/tmp_SG
_LIB_EXP_26/Universal Planar Links SimMultiBody.avi'
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:48:34!
Executed 16-Sep-2018 16:48:36 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing

2017-01-05: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.3 required)
- 3. Show a Two Pose Problem
- 4. Find a General Solutions for the Two Pose Problem
- 5. Find a special Solution for the 4Bar-Linkage wit A0 and B0 on same level
- 6. Create a SimMultiyBody System for the calculated solution
- 7. Show the Video of the Simulation
- 8. Now Create the Solid Geoemtry at time 0.78 seconds
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.3 required)

```
function VLFL_EXP27
```

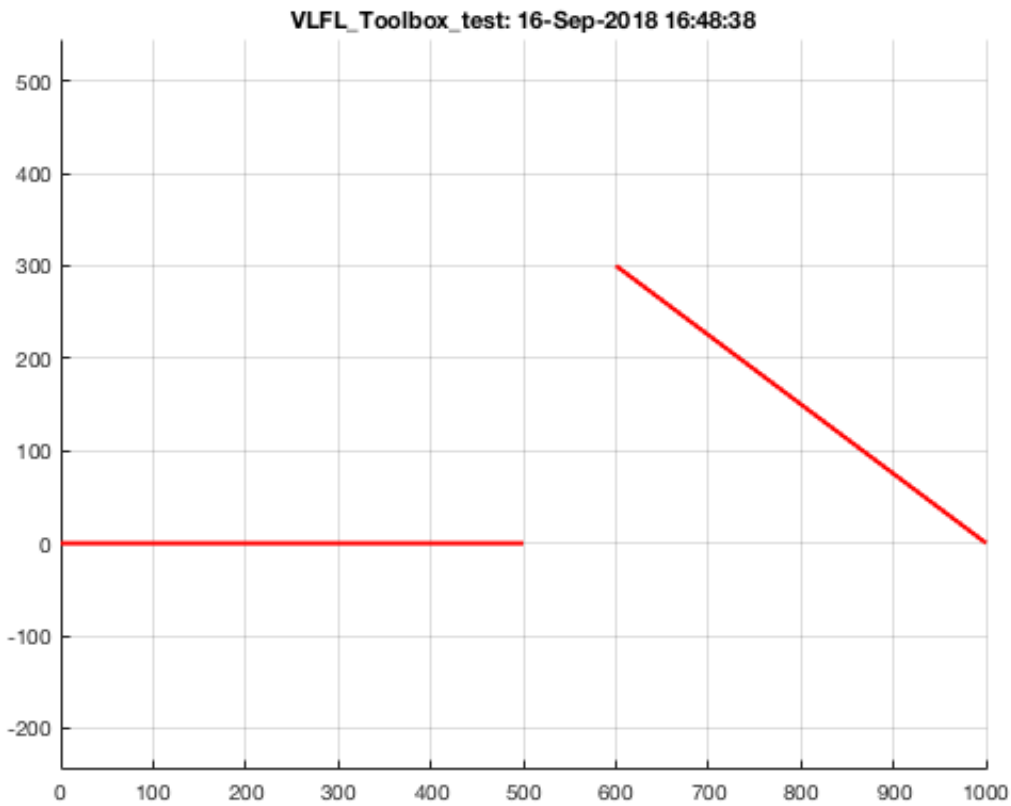
```
smbNewSystem ('SG_LIB_EXP_27');           % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_27/'
```

3. Show a Two Pose Problem

Poses are described by the start point and end point of a link. The first point of the coupling defines a point, the second point also the direction. In the simplest case, two poses for the design of a four-bar are given.

```
d=[0 0];
C1=[0 0];
D1=[500 0];
C2=[600 300];
D2=[1000 0];

SGfigure;
PLplot([C1;D1],'r-',2);
PLplot([C2;D2],'r-',2);
```



4. Find a General Solutions for the Two Pose Problem

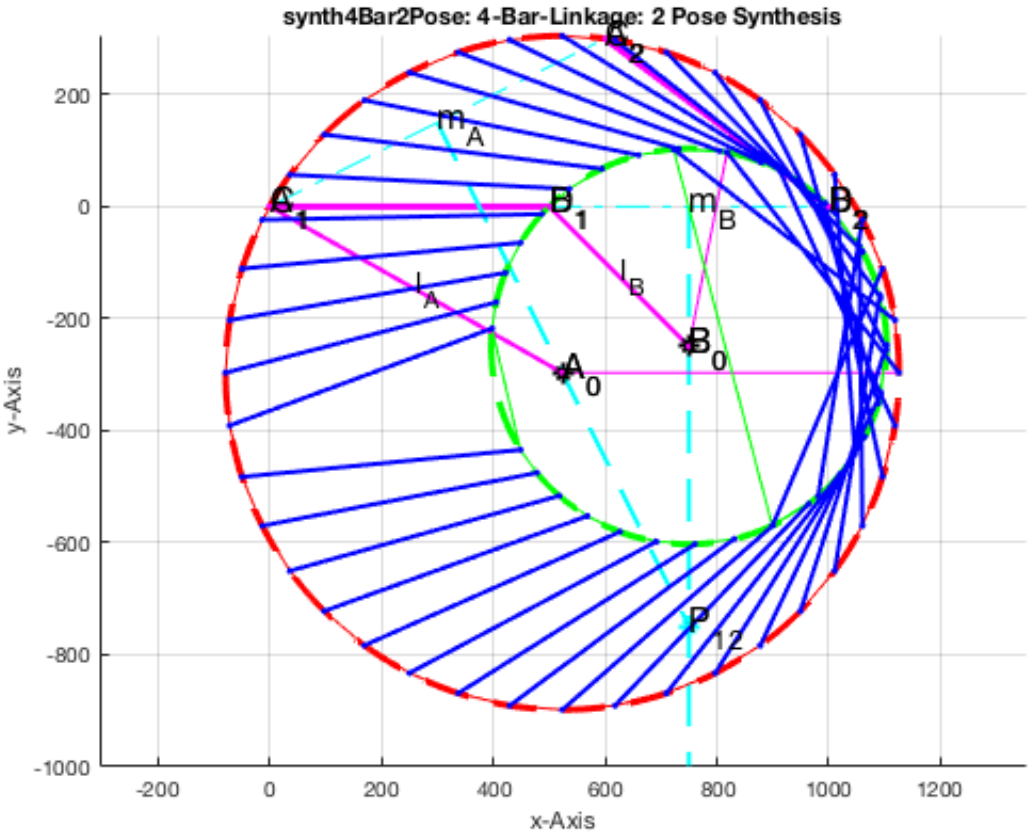
As a solution, there are two straight lines on each of which the frame point A0 or the frame point B0 may be located. The intersection point of these two lines is the pole point P12. In a special case, both frame points are located at this point and a triangle is formed from the four-bar. In any case, the rack points can be displaced such that other secondary conditions can also be fulfilled.

```
imageFigureMovie('record');
synth4Bar2Pose(C1,D1,C2,D2,d);
[~,FN2]=imageFigureMovie('write',smbFilename('synth4Bar2Pose.avi'));
```

```
ans =

  Line with properties:

              Color: [0 0 1]
          LineStyle: '-'
          LineWidth: 3
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [0 500]
              YData: [0 0]
              ZData: [0 0]

  Use GET to show all properties
```

```
ans =

  Line with properties:

              Color: [0 0 1]
          LineStyle: '-'
          LineWidth: 3
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [600 1000]
              YData: [300 0]
              ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 3
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [0 500]
              YData: [0 0]
              ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 3
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [600 1000]
              YData: [300 0]
              ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:
```

```
              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 2
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [523.6068 0]
              YData: [-297.2136 0]
              ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 2
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [750 500]
              YData: [-250 0]
              ZData: [0 0]

  Use GET to show all properties

imageFigureSaveMovie: Writing figure movie with 42 frames in file: /Users/timlueth/Desktop/
Toolbox_test/tmp_SG_LIB_EXP_27/synth4Bar2Pose.avi.
```



5. Find a special Solution for the 4Bar-Linkage wit A0 and B0 on same level

If A0 and B0 are to lie on the same plane, B0+k*(P12-B0) must correspond to the Y coordinate of the Y coordinate of A0 in the y coordinate

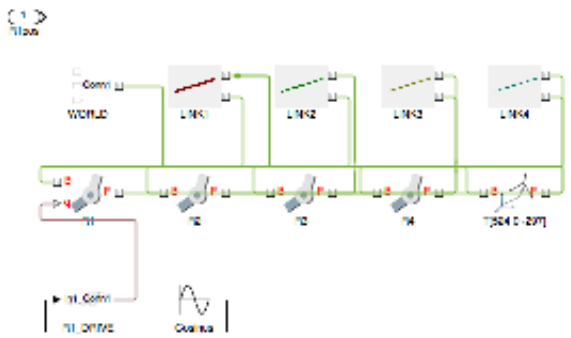
```
[A0,B0,A1,B1,P12]=synth4Bar2Pose(C1,D1,C2,D2,d);
db=P12-B0
k=(A0(2)-B0(2))/db(2)
B0=B0+k*db
A0
L1=norm(B0-A0)
L2=norm(B1-B0)
L3=norm(A1-B1)
L4=norm(A0-A1)
```

```
db =

    0.0000 -500.0000


k =

    0.0944


B0 =
```

```
  750.0000 -297.2136


A0 =

  523.6068 -297.2136


L1 =

  226.3932


L2 =

  388.3760


L3 =

   500


L4 =

  602.0797
```

6. Create a SimMultiyBody System for the calculated solution

```
A=SGmodelLink(L1,'',1,2); A=SGmodelLink2(L1,0,1);
B=SGmodelLink(L2,'',1,2); B=SGmodelLink2(L2,0,1);
C=SGmodelLink(L3,'',1,2); C=SGmodelLink2(L3,0,-1);
D=SGmodelLink(L4,'',1,2); D=SGmodelLink2(L4,0,-1);

smbCreateSG (A,'LINK1','r');              % Add long rod as LINK1
smbCreateSG (B,'LINK2','g');              % Add short rod as LINK2
smbCreateSG (C,'LINK3','y');              % Add long rod as LINK3
smbCreateSG (D,'LINK4','c');              % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
smbCreateConnection('WORLD.ORIGIN','LINK1.B',TofP([A0(1) 0 A0(2)])); % Connect Linkage to W
orld Frame
smbCreateDrive ('R1');
smbCreateSineWave ('Cosinus','R1_DRIVE/1');
ID=smbDrawNow;
```



7. Show the Video of the Simulation

```
[I1,FN]=smbVideoSimulation (4);    % Simulate for 1 second
IT=imageVideoTitle(FN,{'SG-Lib Tutorial #27','2 Pose Syntheses','Tim C. Lueth','$date'},'',
[0.78 1.33]);
IE=imageVideoEndtitle(FN);
videoWriteClipMovie(smbFilename('2 Pose Syntheses SimMultiBody.avi'),IT,2,FN2,ID,1,FN,IE,1)
;
imshow(I1);
```

```
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00020857922471329357. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 7.4102226475251900E-019 for 1 consecutive times at time
2.0857922471329400E-004.  Solver will continue simulation with the step size
restricted to 7.4102226475251900E-019 and using an effective relative error
tolerance of 8.4742961193161300E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00020857922471329943. Simulink will continue to simulate with warnings.
Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 7.4102226475252200E-019 for 2 consecutive times at time
2.0857922471329400E-004.  Solver will continue simulation with the step size
restricted to 7.4102226475252200E-019 and using an effective relative error
tolerance of 1.2481106134613500E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00020857922521788663. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 7.4102226654519400E-019 for 1 consecutive times at time
2.0857922521788700E-004.  Solver will continue simulation with the step size
restricted to 7.4102226654519400E-019 and using an effective relative error
tolerance of 1.1695417848713900E-002, which is greater than the specified
```

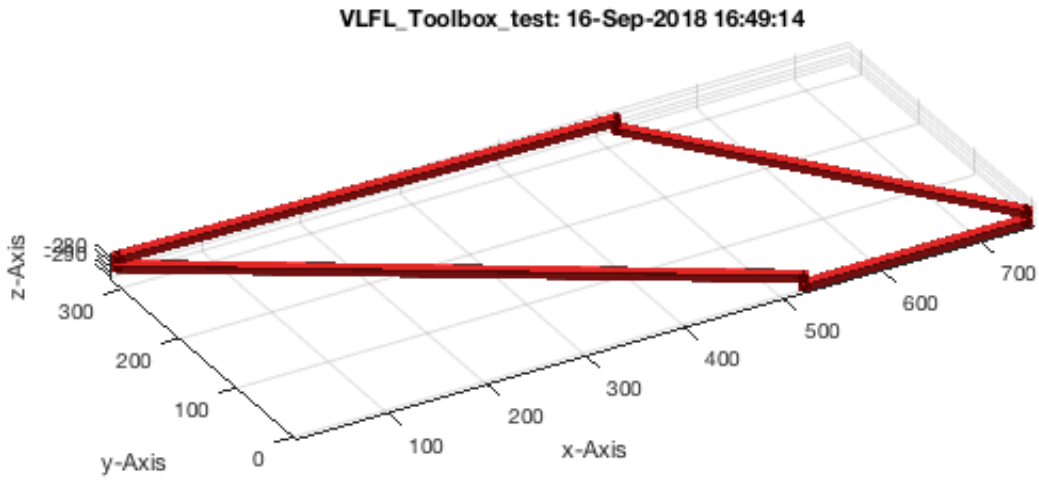
```
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00020857922521788736. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 7.4102226654519600E-019 for 2 consecutive times at time
2.0857922521788700E-004.  Solver will continue simulation with the step size
restricted to 7.4102226654519600E-019 and using an effective relative error
tolerance of 1.0469936968904200E-002, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00020857922521788809. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 7.4102226654519900E-019 for 3 consecutive times at time
2.0857922521788800E-004.  Solver will continue simulation with the step size
restricted to 7.4102226654519900E-019 and using an effective relative error
tolerance of 1.2481106164805300E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
...Creating a new video file (NO SOUND/2016b): '/Users/timlueth/Desktop/Toolbox_test/tmp_SG
_LIB_EXP_27/2 Pose Syntheses SimMultiBody.avi'
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100% 5% 10% 15%
20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```

8. Now Create the Solid Geoemtry at time 0.78 seconds

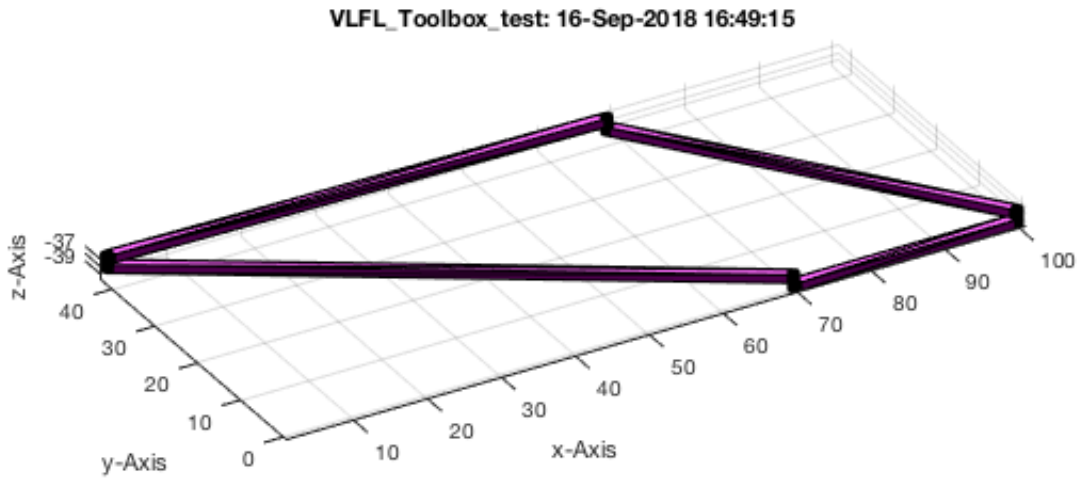
```
SG=smbFullModelSimulation(0.78);
SG=SGmagnifyVL(SG,'',[100 100 100]);
SGwriteSTL(SG,smbFilename('2-Pose-Synth'));
```

```
CREATING A FULL SOLID-MOVEMENT SIMULATION-MODEL 'SG_LIB_EXP_27' THAT RUNS At LEAST 0.78 SEC
ONDS
==========================================================================================
========
Adding frame sensors for all solids of the model
Add frame sensors for 'LINK1.SG'
Add frame sensors for 'LINK2.SG'
Add frame sensors for 'LINK3.SG'
Add frame sensors for 'LINK4.SG'
==========================================================================================
========
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00015184117587706779. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 5.3944822254360900E-019 for 1 consecutive times at time
1.5184117587706800E-004.  Solver will continue simulation with the step size
restricted to 5.3944822254360900E-019 and using an effective relative error
tolerance of 4.3035671698391400E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00015184117697708603. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 5.3944822645165900E-019 for 1 consecutive times at time
1.5184117697708600E-004.  Solver will continue simulation with the step size
restricted to 5.3944822645165900E-019 and using an effective relative error
tolerance of 6.1589696648945900E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00015184117697708658. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 5.3944822645166100E-019 for 2 consecutive times at time
1.5184117697708700E-004.  Solver will continue simulation with the step size
restricted to 5.3944822645166100E-019 and using an effective relative error
tolerance of 8.7521093125381700E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.
Warning: Solver is encountering difficulty in simulating model 'SG_LIB_EXP_27'
at time 0.00015184117697708712. Simulink will continue to simulate with
warnings. Please check the model for errors.
Warning: Solver was unable to reduce the step size without violating minimum
step size of 5.3944822645166300E-019 for 3 consecutive times at time
1.5184117697708700E-004.  Solver will continue simulation with the step size
restricted to 5.3944822645166300E-019 and using an effective relative error
tolerance of 1.2266067701044700E-003, which is greater than the specified
relative error tolerance of 1.0000000000000000E-003. This usually may be caused
by the high stiffness of the system. Please check the system or increase the
solver Number of consecutive min steps violation parameter.

simOut =
```

```
  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   sout: [1x1 Simulink.SimulationData.Dataset]
                   tout: [189x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]

     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]

LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_27/sbm_temp_LI
NK1.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_27/sbm_temp
_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_27/sbm_temp
_LINK3.stl
Header:
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_27/sbm_temp
_LINK4.stl
Header:
Number of facets: 2584
0..
CREATED A SOLID GEOMETRY OF THE FULL SIMULATION-MODEL 'SG_LIB_EXP_27' AT TIME: 0.78 SECONDS
==========================================================================================
========
```




```
SGfigure; view(-30,30); SGplot(SG,'m'); % SGanalyzeGroupParts(SG);
```



Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:49:16!
Executed 16-Sep-2018 16:49:18 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing

2017-01-08: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.3 required)
- 3. Show a Two Pose Problem
- 4. Find a General Solutions for the Three Pose Problem
- 5. Find a special Solution for the 4Bar-Linkage wit A0 and B0 on same level
- 6. Create a SimMultiyBody System for the calculated solution
- 7. Show the Video of the Simulation
- 8. Now Create the Solid Geoemtry at time 0.78 seconds
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.3 required)

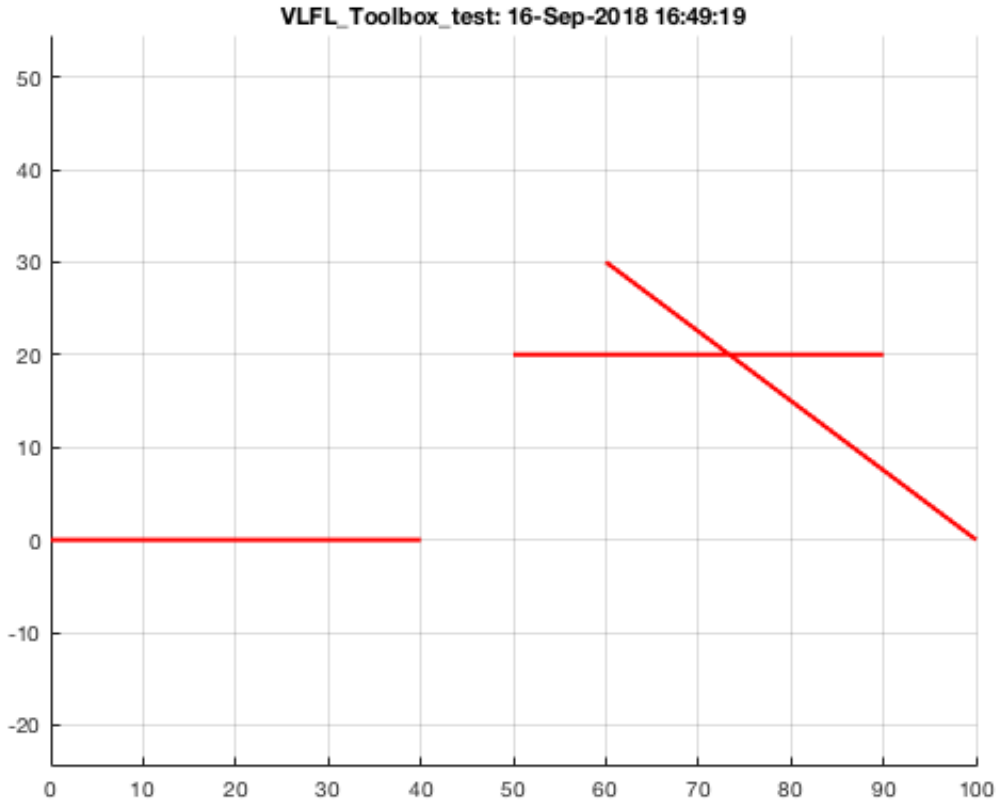
```
smbNewSystem ('SG_LIB_EXP_28',[+5 +5 +5]);          % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_28/'
```

3. Show a Two Pose Problem

Poses are described by the start point and end point of a link. The first point of the coupling defines a point, the second point also the direction. In the simplest case, two poses for the design of a four-bar are given.

```
d=[0 0];
C1=[ 0 0];
D1=[40 0];
C2=[60 30];
D2=[100 0];
C3=[90 20];
D3=[50 20];

SGfigure;
PLplot([C1;D1],'r-',2);
PLplot([C2;D2],'r-',2);
PLplot([C3;D3],'r-',2);
```



4. Find a General Solutions for the Three Pose Problem

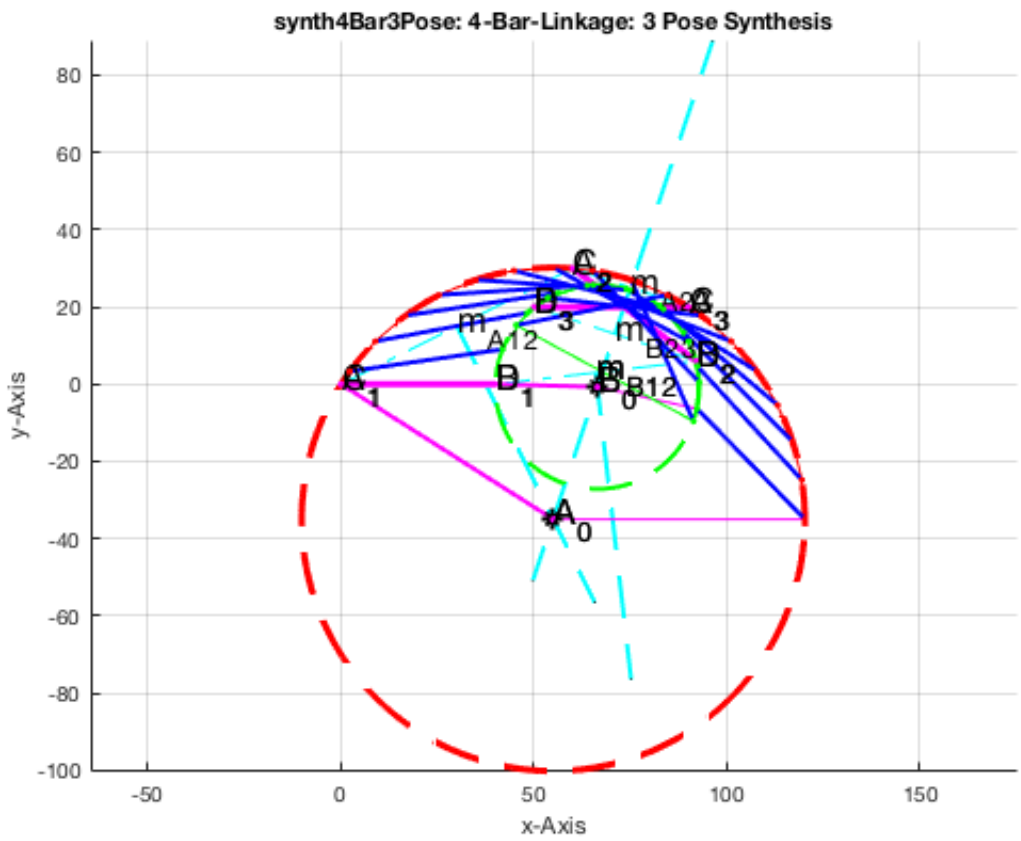
As a solution, there are two straight lines on each of which the frame point A0 or the frame point B0 may be located. The intersection point of these two lines is the pole point P12. In a special case, both frame points are located at this point and a triangle is formed from the four-bar. In any case, the rack points can be displaced such that other secondary conditions can also be fulfilled.

```
l=500

imageFigureMovie('record');
synth4Bar3Pose(C1,D1,C2,D2,C3,D3,d);
% exp_2017_01_08(C1,D1,C2,D2,C3,D3,l,d);
[~,FN2]=imageFigureMovie('write',smbFilename('synth4Bar3Pose.avi'));
```

```
l =

   500


ans =

  Line with properties:

              Color: [0 0 1]
          LineStyle: '-'
          LineWidth: 3
             Marker: '.'
```

```
          MarkerSize: 6
     MarkerFaceColor: 'none'
               XData: [0 40]
               YData: [0 0]
               ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

             Color: [0 0 1]
         LineStyle: '-'
         LineWidth: 3
            Marker: '.'
          MarkerSize: 6
     MarkerFaceColor: 'none'
               XData: [60 92]
               YData: [30 6]
               ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

             Color: [0 0 1]
         LineStyle: '-'
         LineWidth: 3
            Marker: '.'
          MarkerSize: 6
     MarkerFaceColor: 'none'
               XData: [90 50]
               YData: [20 20]
               ZData: [0 0]

  Use GET to show all properties


ans =

  Line with properties:

             Color: [1 0 1]
         LineStyle: '-'
         LineWidth: 2
            Marker: '.'
          MarkerSize: 6
     MarkerFaceColor: 'none'
               XData: [55.0000 0]
               YData: [-35.0000 0]
               ZData: [0 0]
```

```
  Use GET to show all properties


ans =

  Line with properties:

              Color: [1 0 1]
          LineStyle: '-'
          LineWidth: 2
             Marker: '.'
         MarkerSize: 6
    MarkerFaceColor: 'none'
              XData: [66.4286 40]
              YData: [-0.7143 0]
              ZData: [0 0]

  Use GET to show all properties

imageFigureSaveMovie: Writing figure movie with 42 frames in file: /Users/timlueth/Desktop/
Toolbox_test/tmp_SG_LIB_EXP_28/synth4Bar3Pose.avi.
```



5. Find a special Solution for the 4Bar-Linkage wit A0 and B0 on same level

If A0 and B0 are to lie on the same plane, B0+k*(P12-B0) must correspond to the Y coordinate of the Y coordinate of A0 in the y coordinate

```
[A0,B0,A1,B1]=synth4Bar3Pose(C1,D1,C2,D2,C3,D3,d);
```

```
% [A0,B0,A1,B1]=exp_2017_01_08(C1,D1,C2,D2,C3,D3,l,d);
A0
B0

L1=norm(B0-A0)
L2=norm(B1-B0)
L3=norm(A1-B1)
L4=norm(A0-A1)

% L1=500; L2=400; L3=500; L4=400

L=[L1 L2 L3 L4]
LMax=find((L==max(L))); LMax=LMax(1);
LMin=find((L==min(L))); LMin=LMin(1);
l1=L(LMin)+L(LMax);
l2=sum(L)-l1;
l3=l1-l2
```

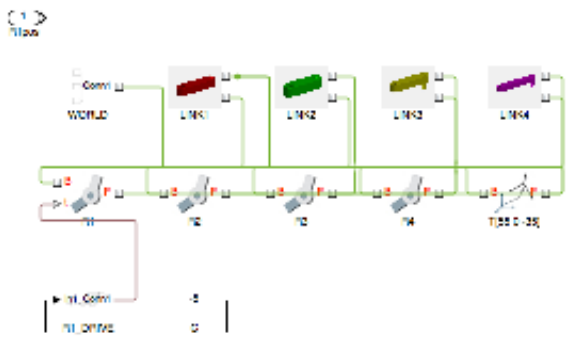
```
A0 =

   55.0000  -35.0000


B0 =

   66.4286   -0.7143


L1 =

   36.1403


L2 =

   26.4382


L3 =

    40


L4 =

   65.1920


L =

   36.1403   26.4382   40.0000   65.1920


l3 =
```

```
    15.4899
```

6. Create a SimMultiyBody System for the calculated solution

```
A=SGmodelLink(L1,'',1,2); A=SGmodelLink2(L1,0,1);
B=SGmodelLink(L2,'',1,2); B=SGmodelLink2(L2,0,1);
C=SGmodelLink(L3,'',1,2); C=SGmodelLink2(L3,0,-1);
D=SGmodelLink(L4,'',1,2); D=SGmodelLink2(L4,0,-1);

smbCreateSG (A,'LINK1','r');              % Add long rod as LINK1
smbCreateSG (B,'LINK2','g');              % Add short rod as LINK2
smbCreateSG (C,'LINK3','y');              % Add long rod as LINK3
smbCreateSG (D,'LINK4','m');              % Add short rod as LINK4
smbCreateJoint ('R','R1','LINK1.F','LINK2.B'); % Add a RR Joint
smbCreateJoint ('R','R2','LINK2.F','LINK3.B'); % Add a RR Joint
smbCreateJoint ('R','R3','LINK3.F','LINK4.B'); % Add a RR Joint
smbCreateJoint ('R','R4','LINK4.F','LINK1.B'); % Add a RR Joint
phi=atan2(B0(2)-A0(2),B0(1)-A0(1))
% phi=0;
smbCreateConnection('WORLD.ORIGIN','LINK1.B',TofR(rot(0,0,phi),[A0(1) 0 A0(2)])); % Connect
 Linkage to World Frame
smbCreateDrive ('R1');
smbSetJointInputTorque('R1');
smbCreateBlockConst('C','R1_DRIVE/1',-5)
ID=smbDrawNow;
smbSimulate(4);
```

```
phi =

    1.2490
```

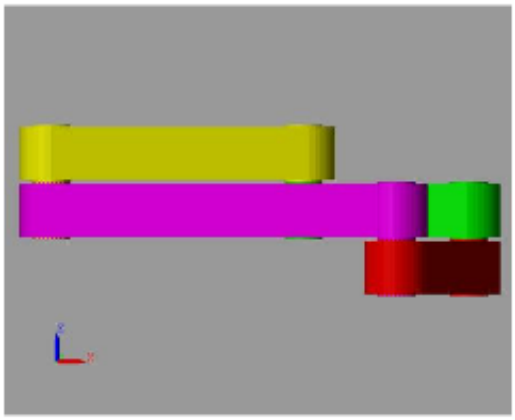


7. Show the Video of the Simulation

```
[I1,FN]=smbVideoSimulation (4);    % Simulate for 1 second
IT=imageVideoTitle(FN,{'SG-Lib Tutorial #27','3 Pose Syntheses','Tim C. Lueth','$date'},'',
[0.44 0.74 1.14]);
```

```
IE=imageVideoEndtitle(FN);
videoWriteClipMovie(smbFilename('3 Pose Syntheses SimMultiBody.avi'),IT,2,FN2,ID,1,FN,IE,1)
;
imshow(I1);
```

```
....Creating a new video file (NO SOUND/2016b): '/Users/timlueth/Desktop/Toolbox_test/tmp_S
G_LIB_EXP_28/3 Pose Syntheses SimMultiBody.avi'
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100% 5% 10% 15%
20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```

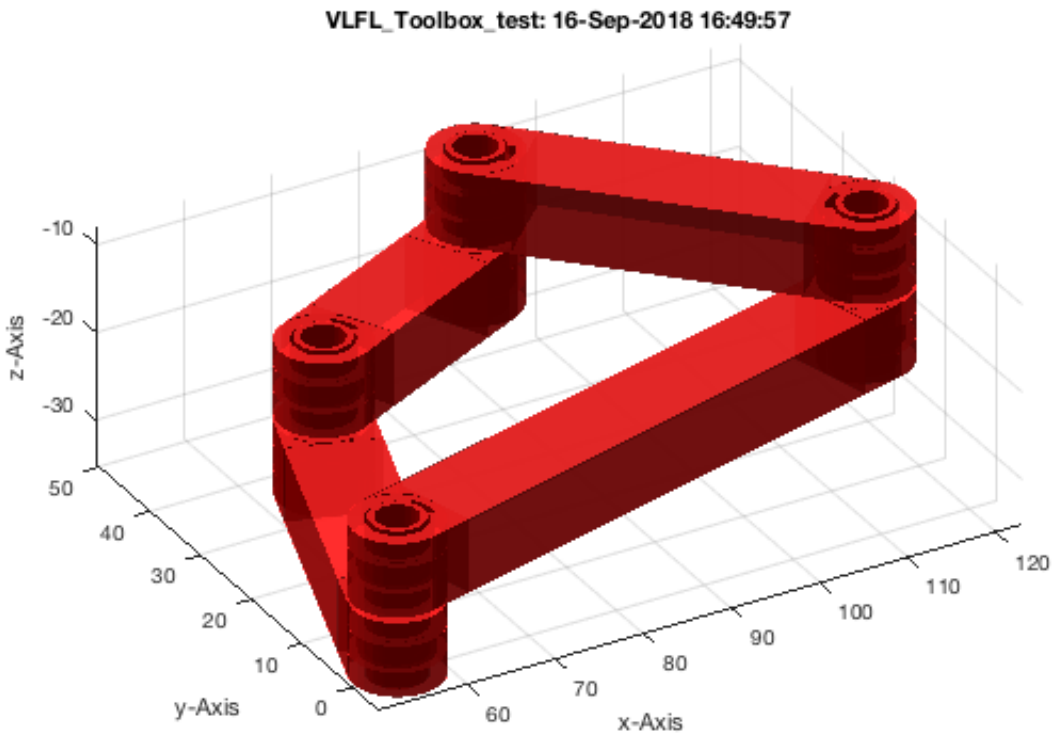


8. Now Create the Solid Geoemtry at time 0.78 seconds

```
SG=smbFullModelSimulation(0.74);
% SG=SGmagnifyVL(SG,'',[100 100 100]);
SGwriteSTL(SG,smbFilename('3-Pose-Synth'));
```

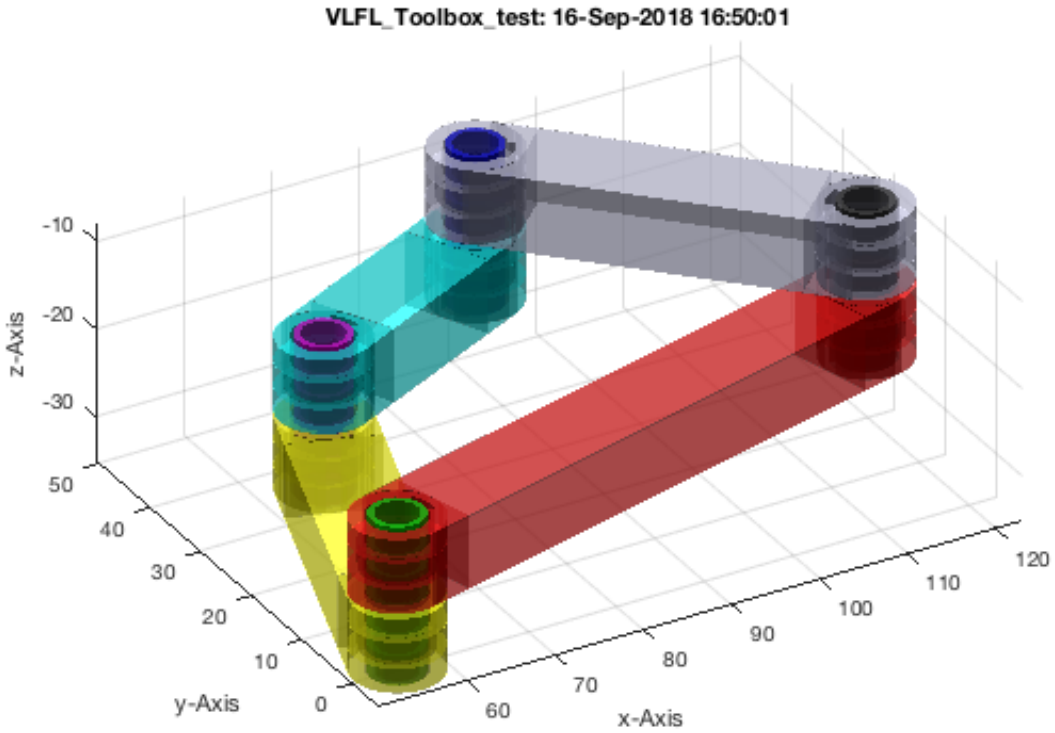
```
CREATING A FULL SOLID-MOVEMENT SIMULATION-MODEL 'SG_LIB_EXP_28' THAT RUNS At LEAST 0.74 SEC
ONDS
==========================================================================================
========
Adding frame sensors for all solids of the model
Add frame sensors for 'LINK1.SG'
Add frame sensors for 'LINK2.SG'
Add frame sensors for 'LINK3.SG'
Add frame sensors for 'LINK4.SG'
==========================================================================================
========

simOut =

  Simulink.SimulationOutput:
                 simlog: [1x1 simscape.logging.Node]
                   sout: [1x1 Simulink.SimulationData.Dataset]
                   tout: [220x1 double]
                   xout: [1x1 Simulink.SimulationData.Dataset]
```

```
     SimulationMetadata: [1x1 Simulink.SimulationMetadata]
           ErrorMessage: [0x0 char]

LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_28/sbm_temp_LI
NK1.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_28/sbm_temp
_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_28/sbm_temp
_LINK3.stl
Header:
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_28/sbm_temp
_LINK4.stl
Header:
Number of facets: 2584
0..
CREATED A SOLID GEOMETRY OF THE FULL SIMULATION-MODEL 'SG_LIB_EXP_28' AT TIME: 0.74 SECONDS
==========================================================================================
========
```




```
SGfigure; view(-30,30); SGplot(SG,'m');  SGanalyzeGroupParts(SG);
```

```
4% 8% 12% 16% 20% 24% 28% 32% 36% 40% 44% 48% 52% 56% 60% 64% 68% 72% 76% 80% 84% 88% 92% 9
6% 100%
SGanalyzeGroupParts: 8 separated parts found.
```



Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:50:02!
Executed 16-Sep-2018 16:50:04 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 29: Create a multi body simulation using several mass points

2017-03-17: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.6 required)
- Motivation for this tutorial
- 1. Create a SimMultiBody system for a Mass - Spring - Damper - System
- 2 Create four mass points
- 2 Create six springs between the masses
- 3. Connect the mass - spring - damping system to the world coordinate system
- 4. Show the Simulation
- 6. Create a Video of the Linkage Simulation
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.6 required)

```
% function VLFL_EXP29
```

Motivation for this tutorial

Showing a finite element mass spring system

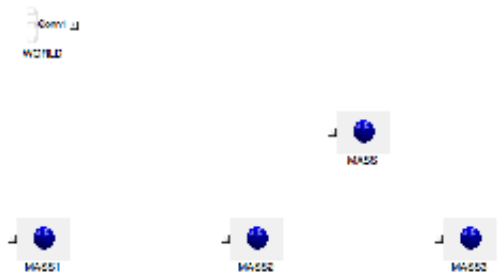
1. Create a SimMultiBody system for a Mass - Spring - Damper - System

```
smbNewSystem ('SG_LIB_EXP_29',[0 0 -9.81])   % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LIB_EXP_29/'
```

2 Create four mass points

```
smbCreateSGMass;
smbCreateSGMass;
smbCreateSGMass;
smbCreateSGMass;
smbDrawNow;
```

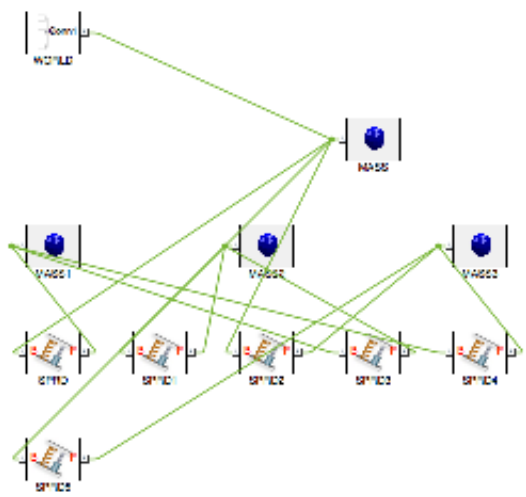


2 Create six springs between the masses

```
smbCreateSpring('MASS','MASS1');
smbCreateSpring('MASS','MASS2');
smbCreateSpring('MASS','MASS3');
smbCreateSpring('MASS1','MASS2');
smbCreateSpring('MASS1','MASS3');
smbCreateSpring('MASS2','MASS3');
```

3. Connect the mass - spring - damping system to the world coordinate system

```
smbAddLine( 'WORLD/RConn1', 'MASS/LConn1');
ID=smbDrawNow;
```



4. Show the Simulation

6. Create a Video of the Linkage Simulation

```
[I1,vname]=smbVideoSimulation (4);     % Simulate for 1 second
IT=imageVideoTitle(vname,{'SG-Lib Tutorial #29','Mass-Spring-Nets','Tim C. Lueth','$date'},
'',[0 4]);
```

```
IE=imageVideoEndtitle(vname);
videoWriteClipMovie(smbFilename('SG-Lib Tutorial #29-Mass-Spring-Nets.avi'),IT,2,ID,1,vname
,IE,1);
imshow(I1);
```

```
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
.Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
Creating a new video file (NO SOUND/2016b): '/Users/timlueth/Desktop/Toolbox_test/tmp_SG_LI
B_EXP_29/SG-Lib Tutorial #29-Mass-Spring-Nets.avi'
Warning: A value of class "matlab.internal.video.PluginManager" was indexed with
no subscripts specified. Currently the result of this operation is the indexed
value itself, but in a future release, it will be an error.
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
```

```
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:50:25!
Executed 16-Sep-2018 16:50:27 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.

2017-02-10: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.4 required)
- 1. Plotting points (PL) and vertices (3D)
- 2. Plotting lines
- slplot([0 0],[10 0],'r-',3,1,1);
- 3. Plotting angles
- 4. Plotting coordinate
- 8. Adding text to the drawings
- 9. Helpful generic polygons for
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.4 required)

In this tutorial, features are introduced that allow you to create drawings for the descriptive geometry using Matlab. The goal is to display lines, planes, spaces and polygons, surfaces and surface-bounded volumes and to label them mathematically (Tex-style). some functions are based in individual points such as:

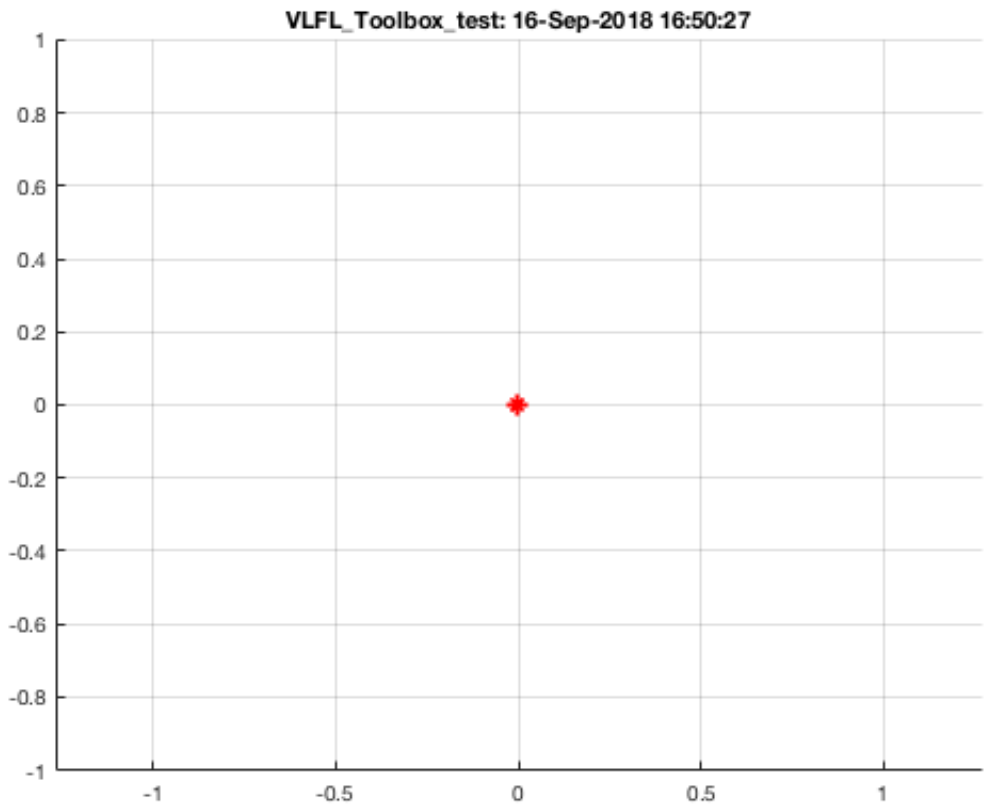
- pplot - plot a point in defined color, shape and size
- lplot - plot a line between two points with color, width, tip, start point end point
- aplot - plot an angle at a point using a line and a second line or angle
- slplot - plot a staight line using a start point and direction vector
- tplot;
- tfplot; some functions are based on point lists (PL) or vertex lists (VL), such as: PLplot, VLplot,

1. Plotting points (PL) and vertices (3D)

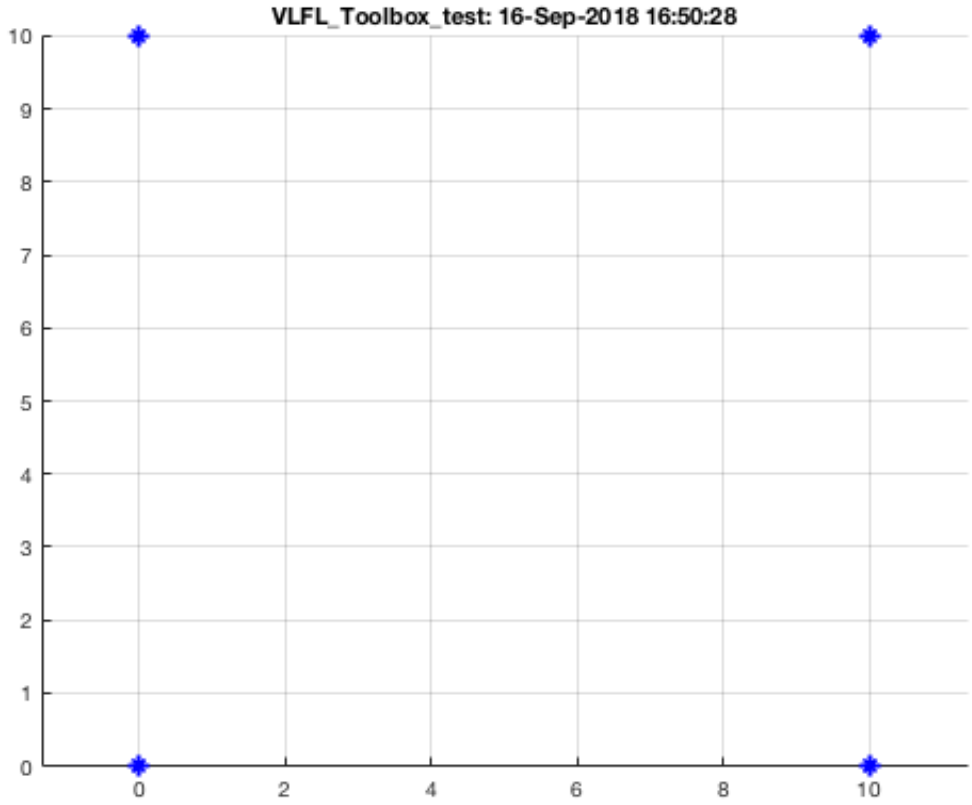
```
p=[0 0]       % row style
p=[0;0]       % column style
v=[0 0 0]   % row style
v=[0; 0; 0]   % column style


SGfigure; pplot(p);
```

```
p =

     0     0


p =
```

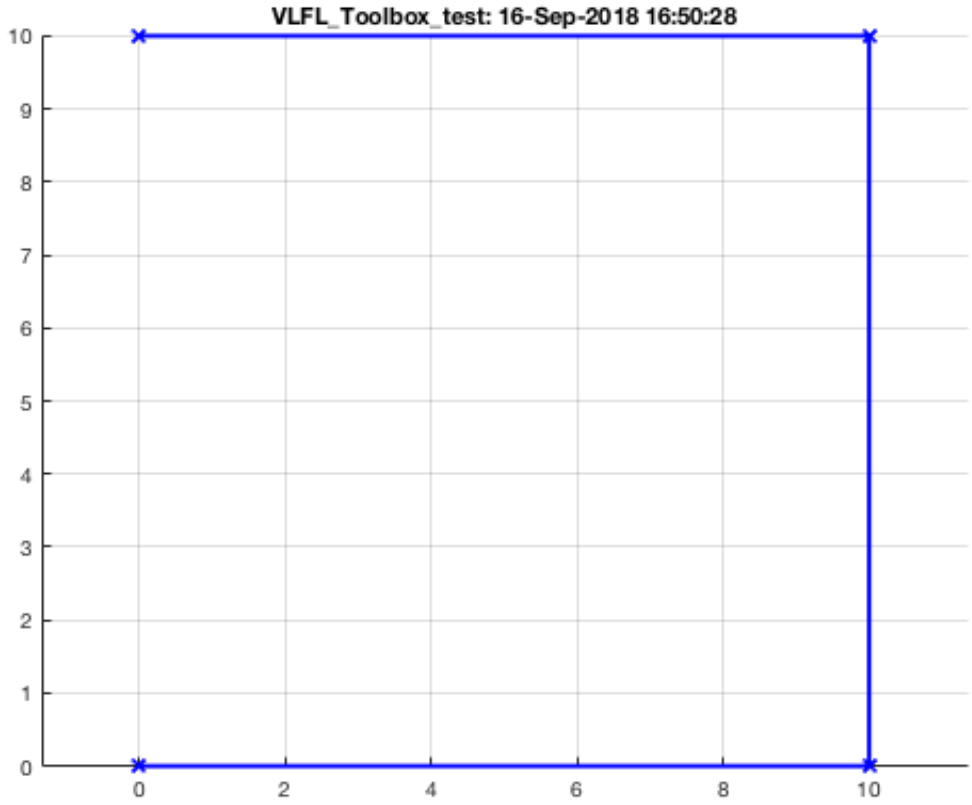
```
     0
     0


v =

     0     0     0


v =

     0
     0
     0
```



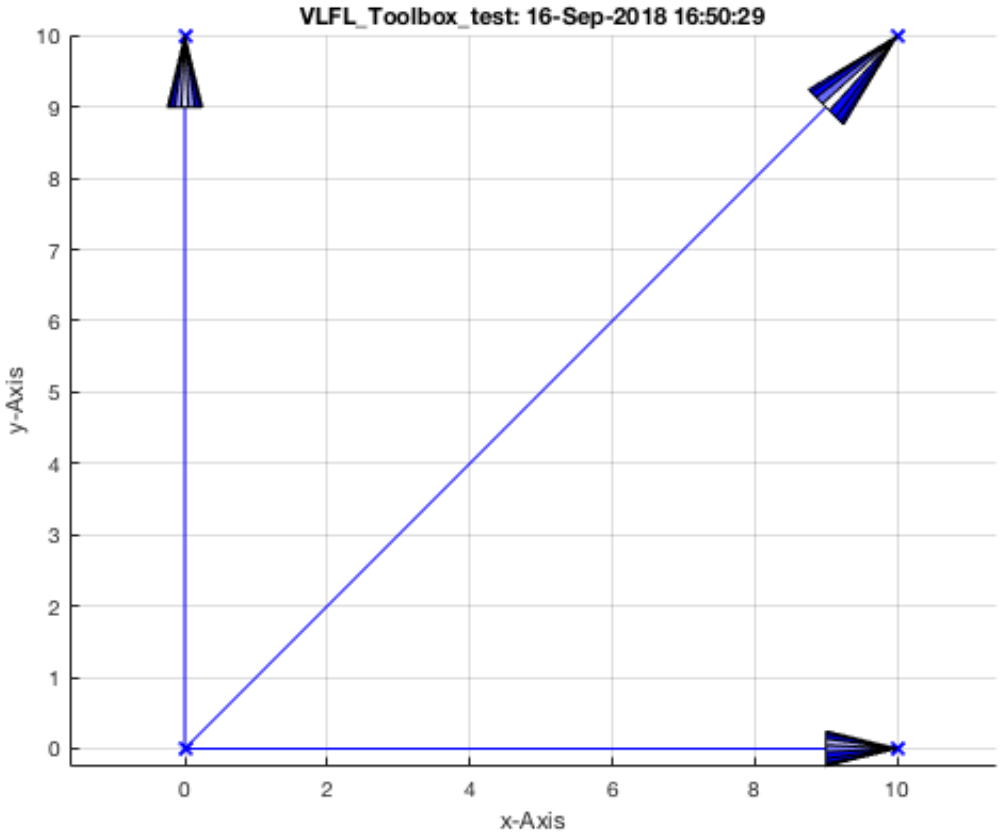

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'b*',2);            % point plot of point list
```



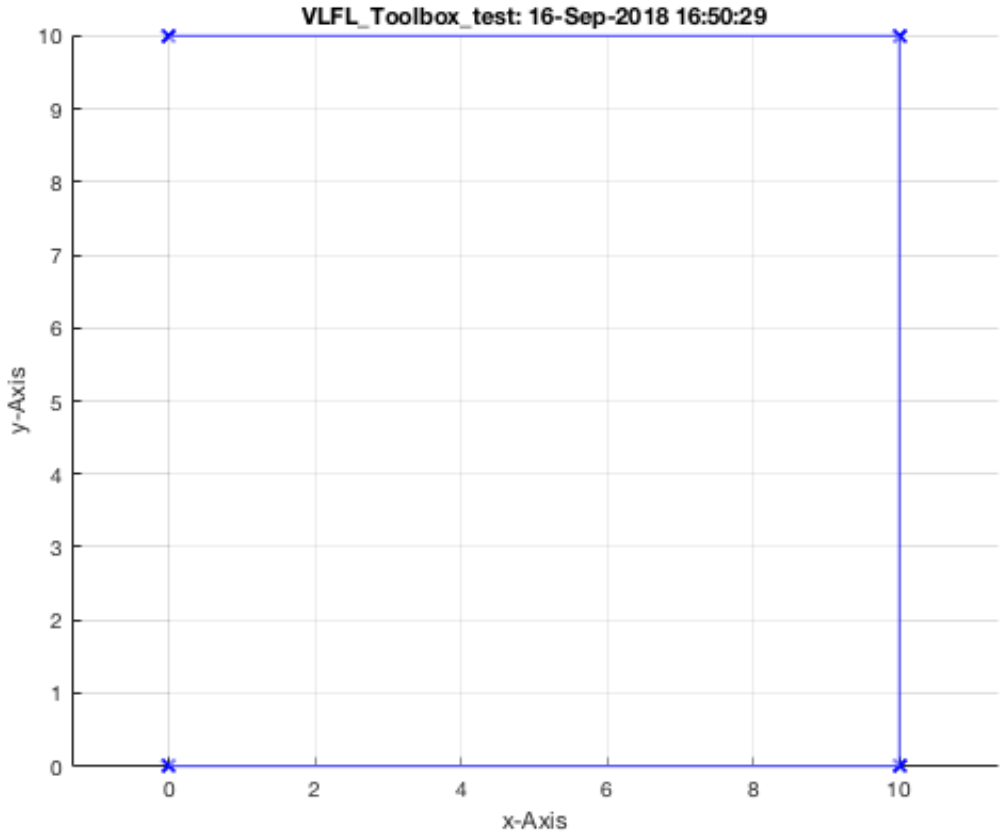

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',2);          % Line plot of point list
```



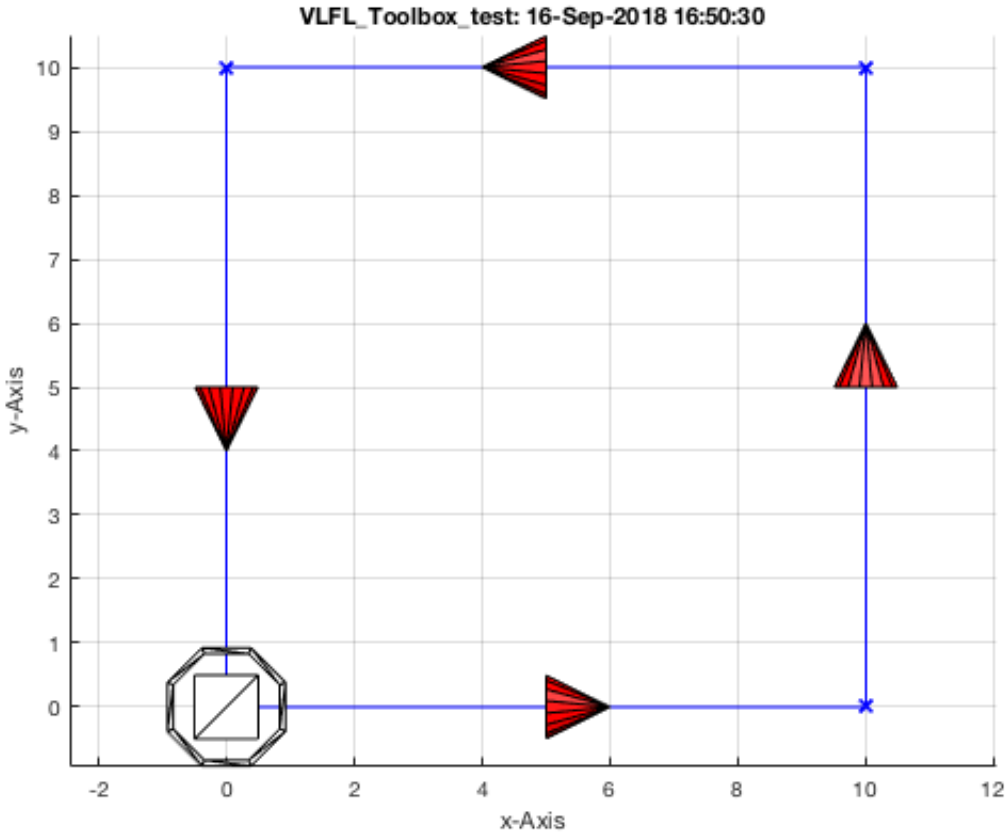

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',1,1);          % Vector plot of point list
```



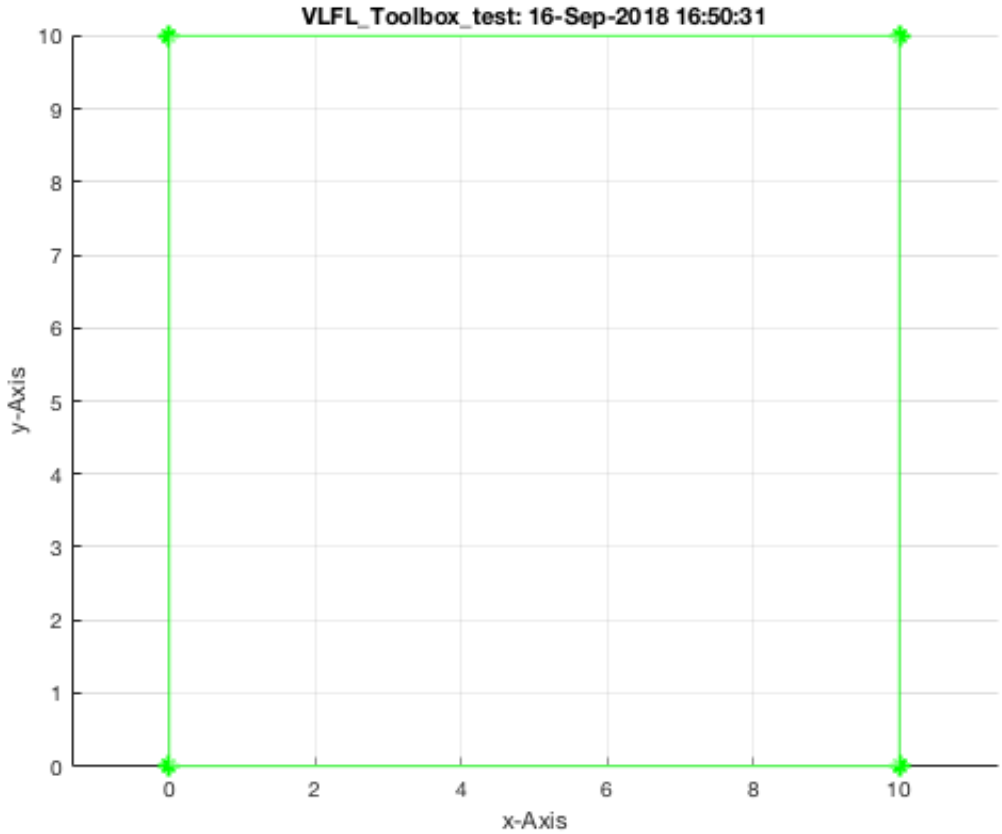

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',1,'',0.5);    % Surface enlosed by point list
```




```
SGfigure; CPLplot([0 0;10 0;10 10; 0 10],'bx-',1,1,1,1);    % Plotting closed polygon
```

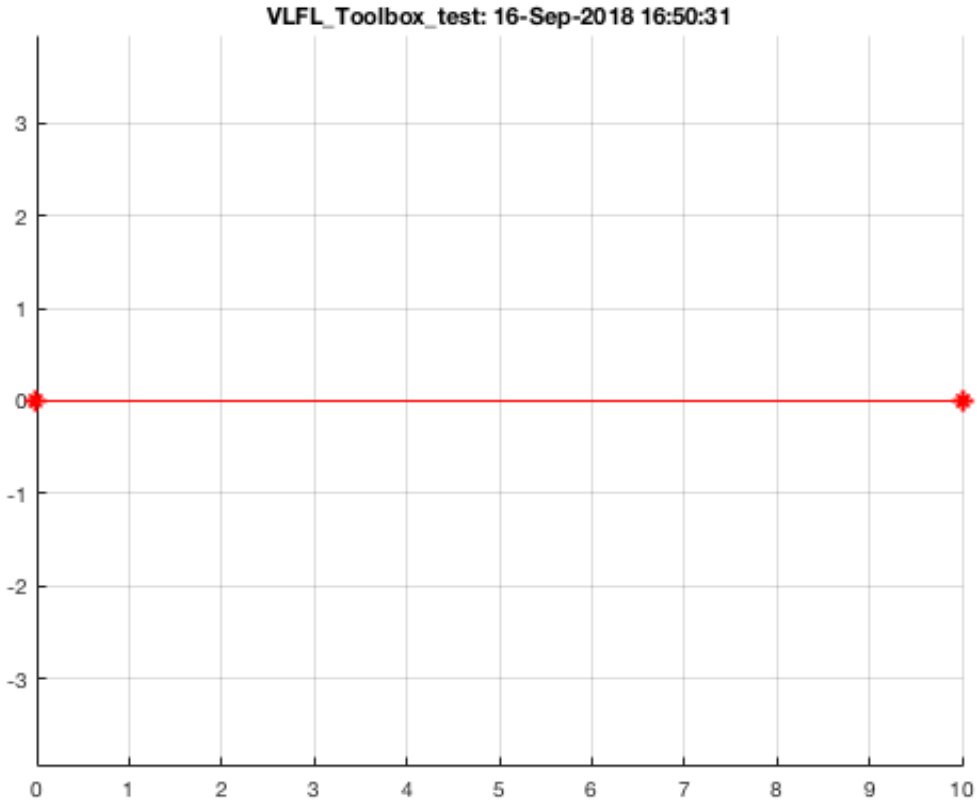



```
SGfigure; CPLfaceplot([0 0;10 0;10 10; 0 10],'g*-',1,0.5);  % Plotting closed polygon surfa
ces
```

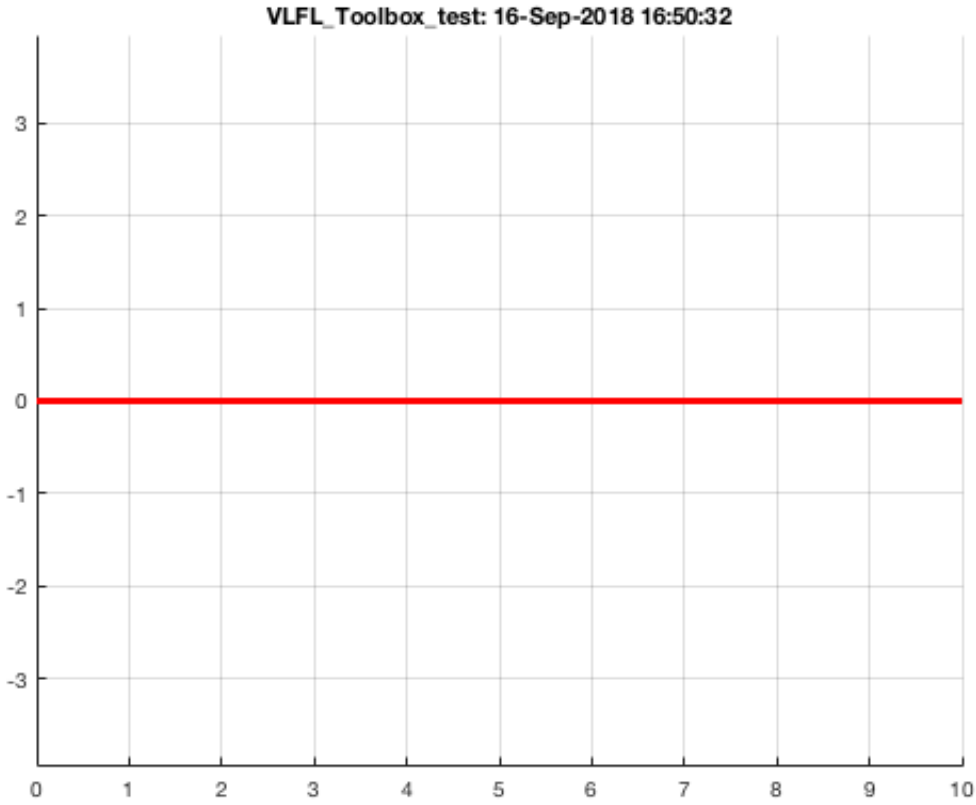


2. Plotting lines

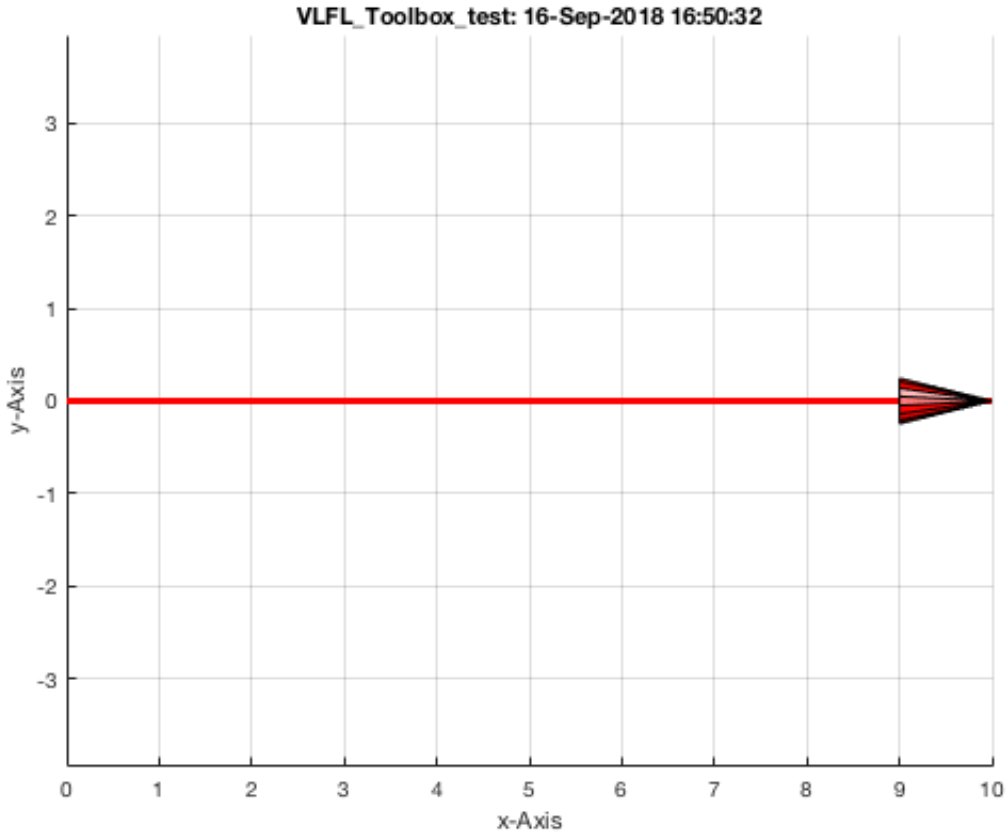
```
SGfigure; lplot([0 0],[10 0],'r*-');
```



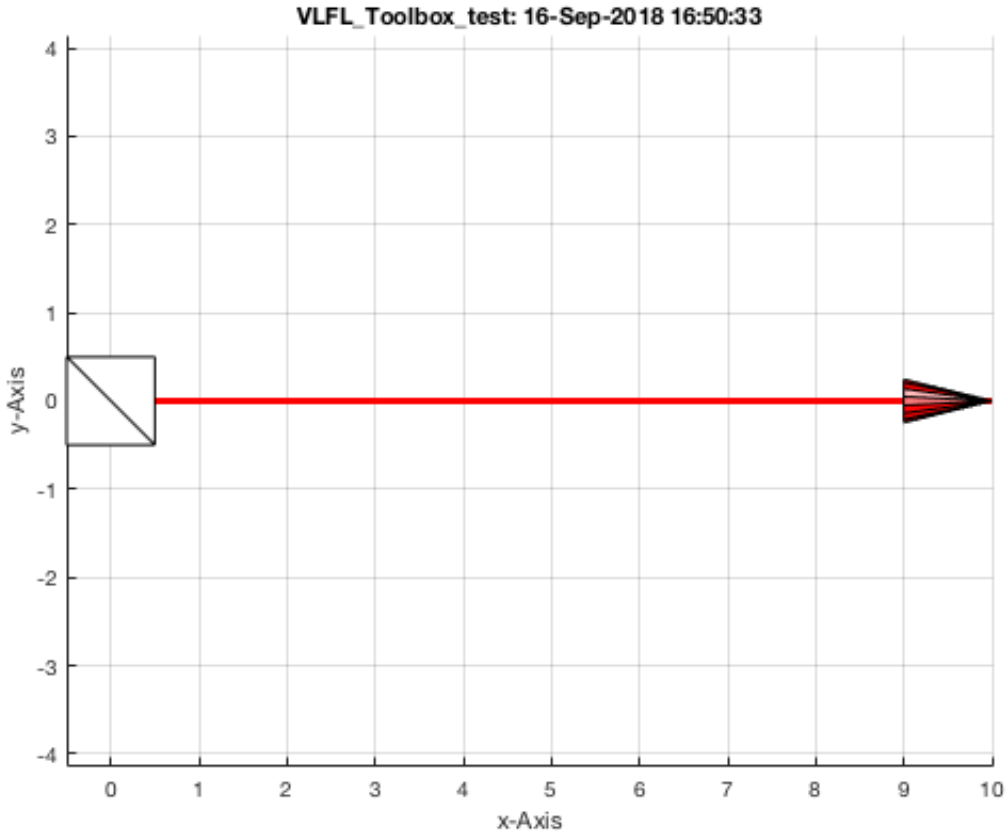

```
SGfigure; lplot([0 0],[10 0],'r-',3);
```




```
SGfigure; lplot([0 0],[10 0],'r-',3,1);
```

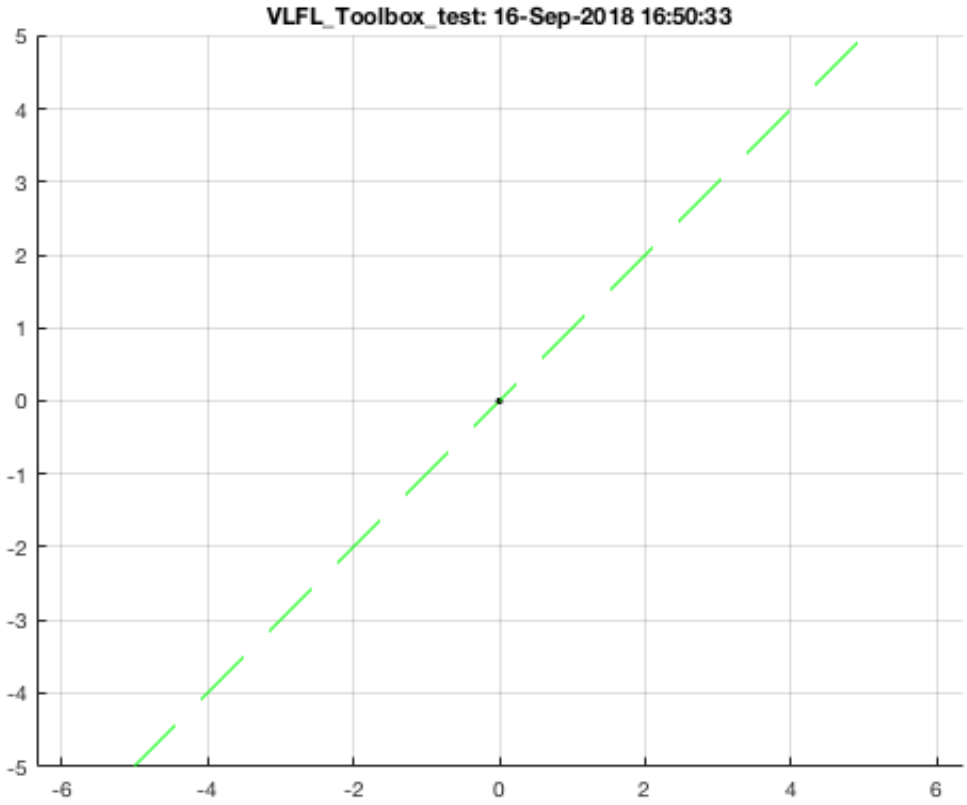



```
SGfigure; lplot([0 0],[10 0],'r-',3,1,1);
```



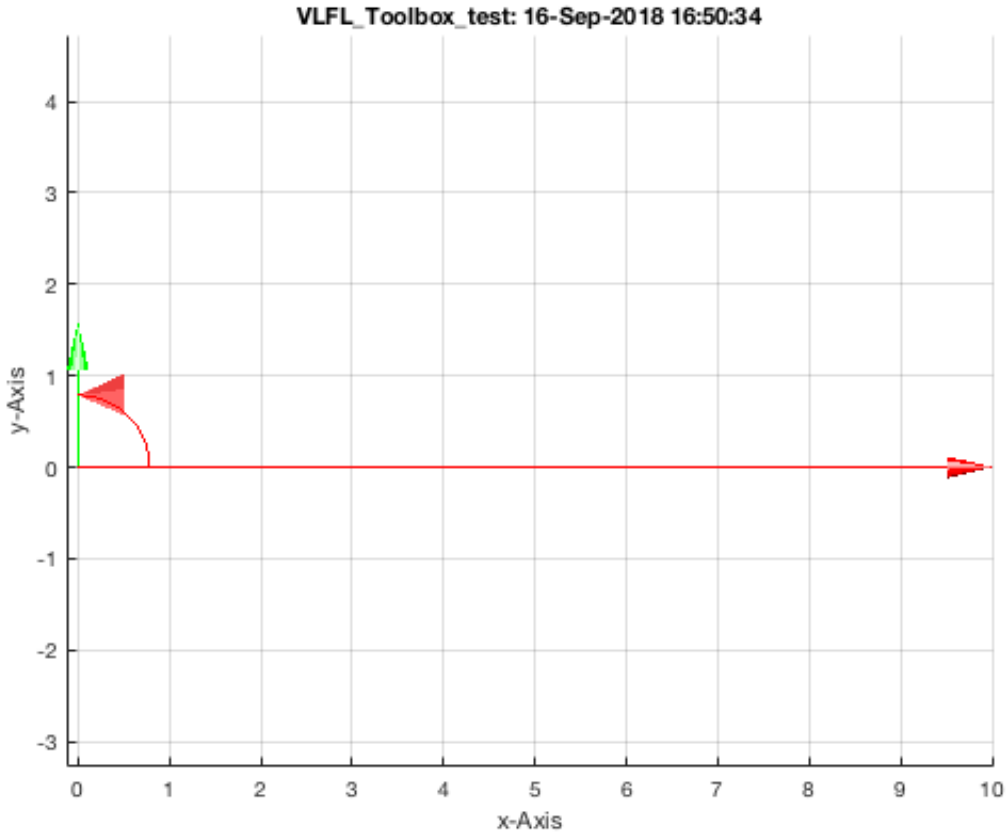
slplot([0 0],[10 0],'r-',3,1,1);

```
SGfigure; slplot([0 0 0],[1 1 1],'color','g--')
```



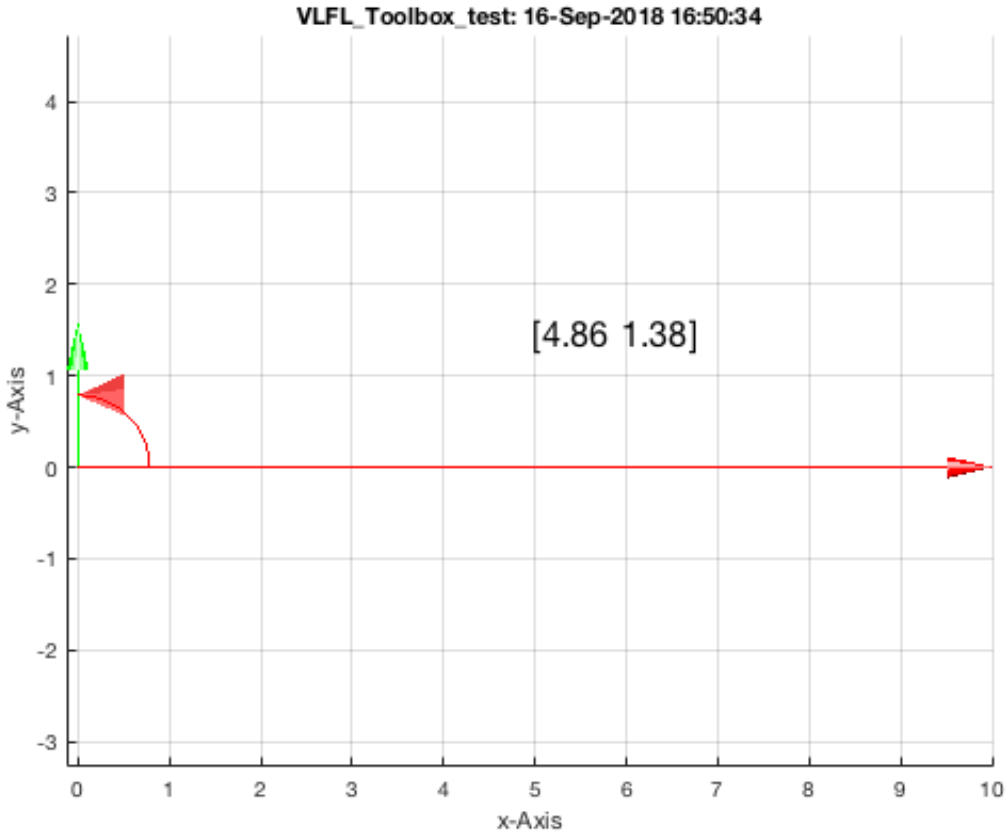
3. Plotting angles

```
SGfigure; aplot([0 0],[10 0],pi/2);
```



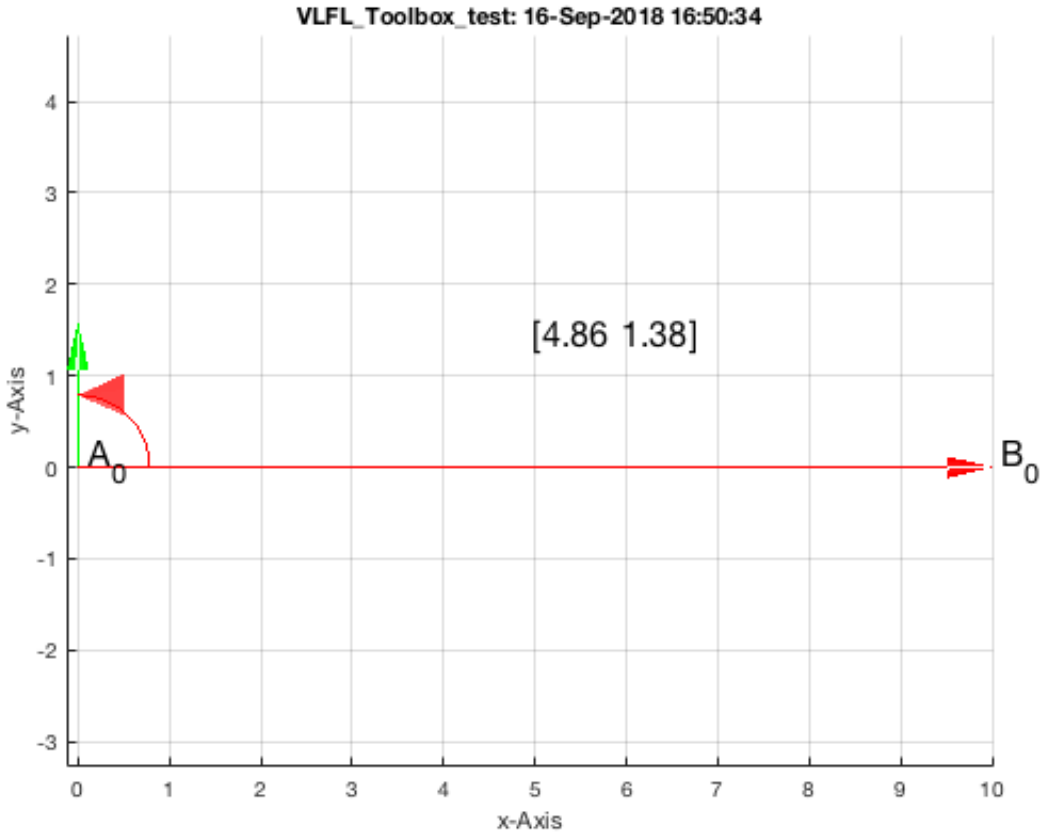
4. Plotting coordinate

```
p=ginput(1); textP (p,sprintf('[%.2f %.2f]',p));
```



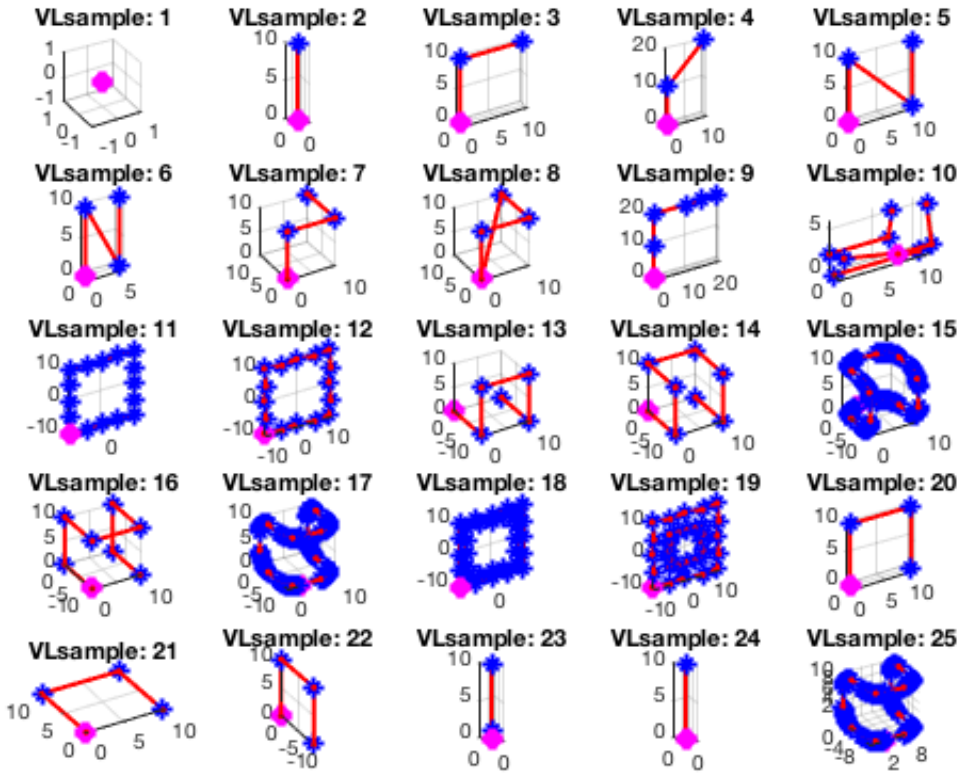
8. Adding text to the drawings

```
textP ([0 0],'A0'); textP ([10 0],'B0');
```

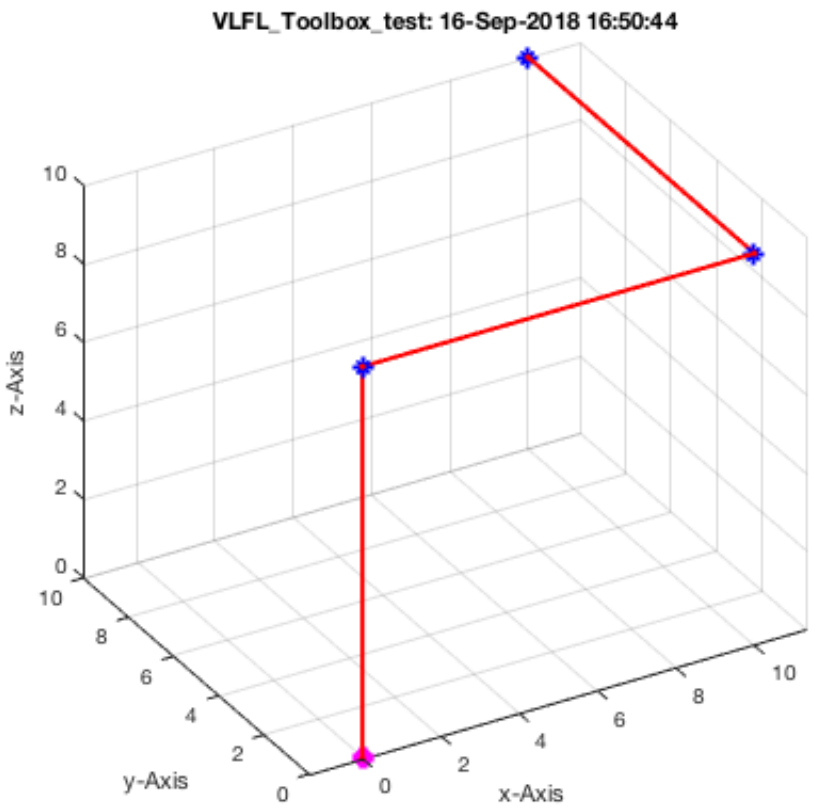


9. Helpful generic polygons for

```
SGfigure; VLsample;
```



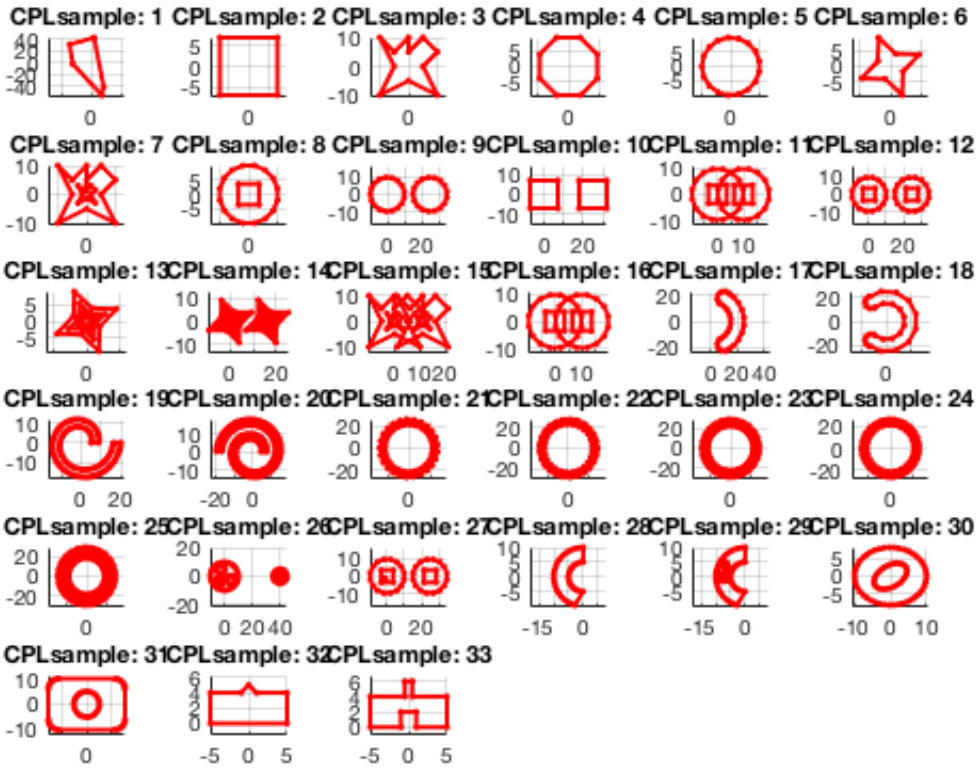

```
SGfigure; VLsample(7);
```


VLFL_Toolbox_test: 16-Sep-2018 16:50:44
z-Axis
y-Axis
x-Axis
0
2
4
6
8
10

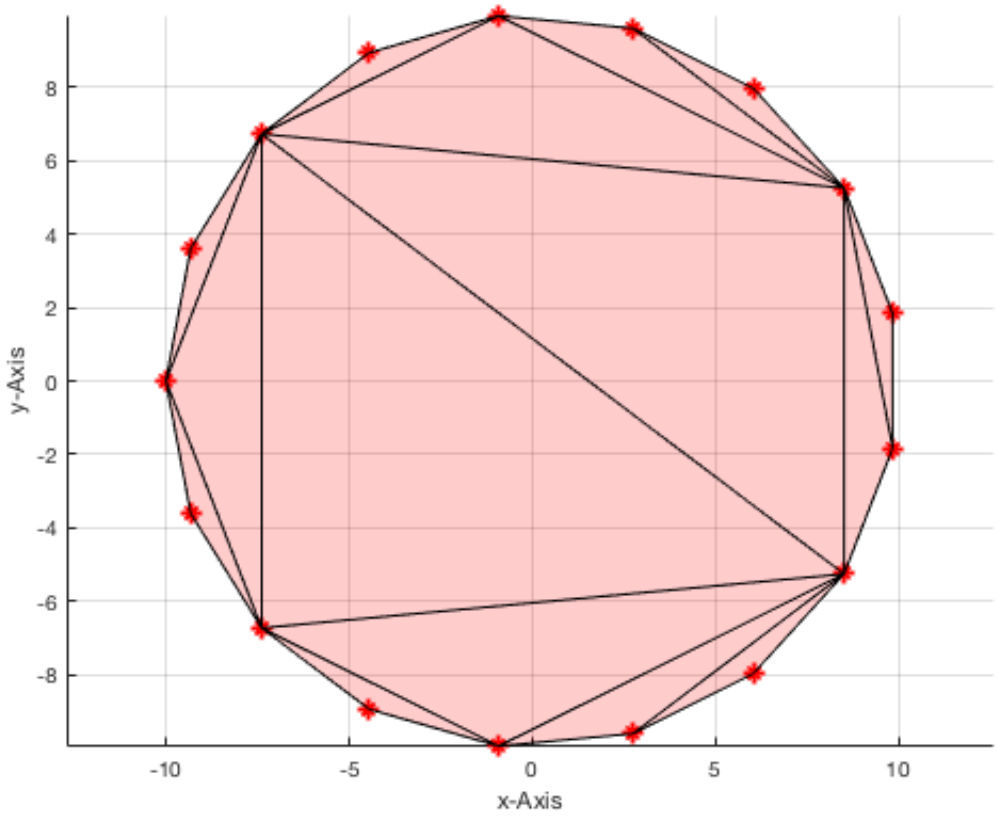
```
SGfigure; CPLsample;
```

```
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=55.00
Head diameter: dh=46.00
Tooth foot width: 4.84
Involute angle: 34.09
Tooth head width: 0.34
Difference tooth gap and tooth head width: 1.74
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=2.00
Teeth Nr: z=25
Rolling circle: d=50.00
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=45.00
Head diameter: dh=54.00
Tooth foot width: 3.96
Involute angle: 38.01
```

```
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=1.50
Teeth Nr: z=25
Rolling circle: d=37.50
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=41.25
Head diameter: dh=34.50
Tooth foot width: 3.63
Involute angle: 34.09
Tooth head width: 0.25
Difference tooth gap and tooth head width: 1.30
```




```
SGfigure; CPLsample(5);
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:50:48!
Executed 16-Sep-2018 16:50:50 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids

2017-02-19: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.4 required)
- 1. Reading DICOM models as voxel model from disk and resize voxel models (VM)
- 2. Solid skull bone reconstruction using SGofVMdelaunay
- 3. Solid skull bone reconstruction using SGofVMisosurface
- 4. Solid skull bone reconstruction using SGofVMmarchcub
- 5. Reduce the numbers Facets to 300.000 facets
- 6 Show the Voxel model in quadrant 1-2-4 and surface model in quadrant 3
- 7. Create a surface model and convert it into a Voxel model
- 8. Plot the surface model in 4 quadrant plot
- 9. Select Point in 3D
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids

- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.4 required)

- VMreaddicomdir - reads in a voxel model
- VMresize - resizes of a voxel model
- SGofVMdelaunay - creates a surface model using delaunay (slow)
- SGofVMmarchcube - creates a surface model using marching cube (fast)
- SGcut - cuts surface models
- CPLofSGslice - creates a slice contour at a specific height/direction
- PLFLofCPLdelauny - tesselates the facets of a CPL using delaunay
- PLFLofCPLpoly - tesselates the facets of a CPL using mapping toolbox
- VMplot - plots a voxel mode
- VMplotslide - plots a voxel mode for slider navigation
- VMimage - plots a voxel image
- VMmontage - montage of voxel
- VMpseudo3D - creates a pesudo 3D image
- VMuidicom - select and read a voxel model
- VMreaddicom - read a dicom file

1. Reading DICOM models as voxel model from disk and resize voxel models (VM)

```
% load AIM_Patientmodel.mat % Does work world-wide=
```

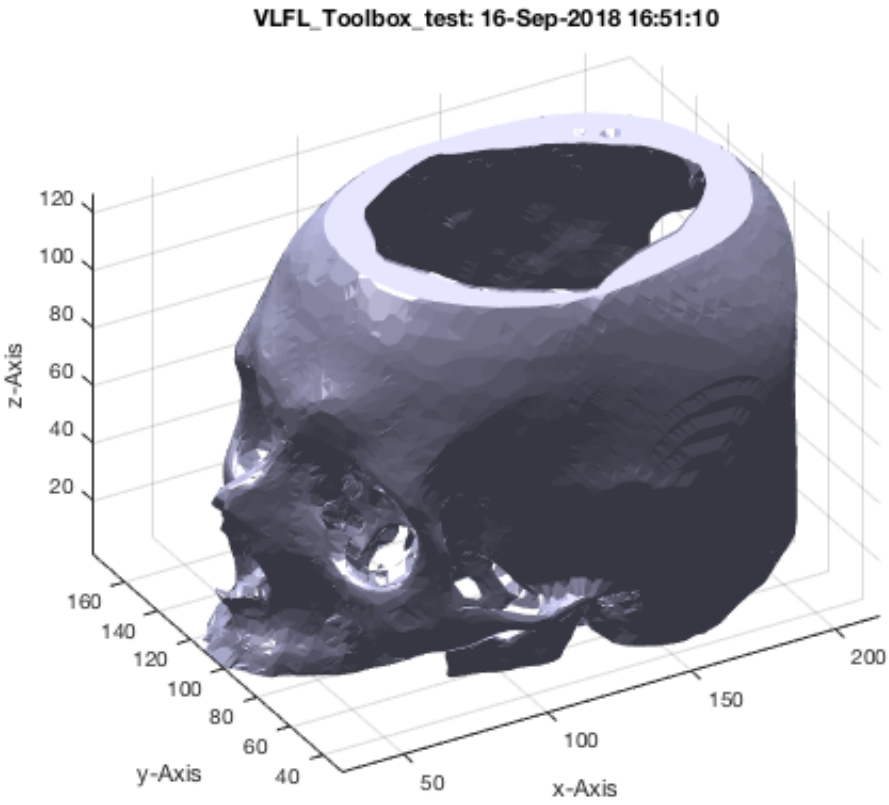
```
[V,vs]=VMreaddicomdir('/Volumes/LUETH-WIN/WIN AIM Matlab Libraries/VLFL-Lib/AIM_DICOMFILES'
);
vs
[a,as]=VMresize(V,[0.5 0.5 0.5],vs);
as
[a,as]=VMresize(V,vs,vs);
as
```

```
VMreaddicomdir: Analyzing 129 entries in path: /Volumes/LUETH-WIN/WIN AIM Matlab Libraries/
VLFL-Lib/AIM_DICOMFILES
.......................................................................................
.....................................
Stack of 126 DICOM images read from disk.

vs =

    0.4219    0.4219    1.0000

VMresize: Resize voxel image [512 512 126] to [256 256 63] with voxel size [0.84mm 0.84mm 2
.00mm]

as =

    0.8438    0.8438    2.0000

VMresize: Resize voxel image [512 512 126] to [216 216 126] with voxel size [1.00mm 1.00mm
1.00mm]

as =

     1     1     1
```

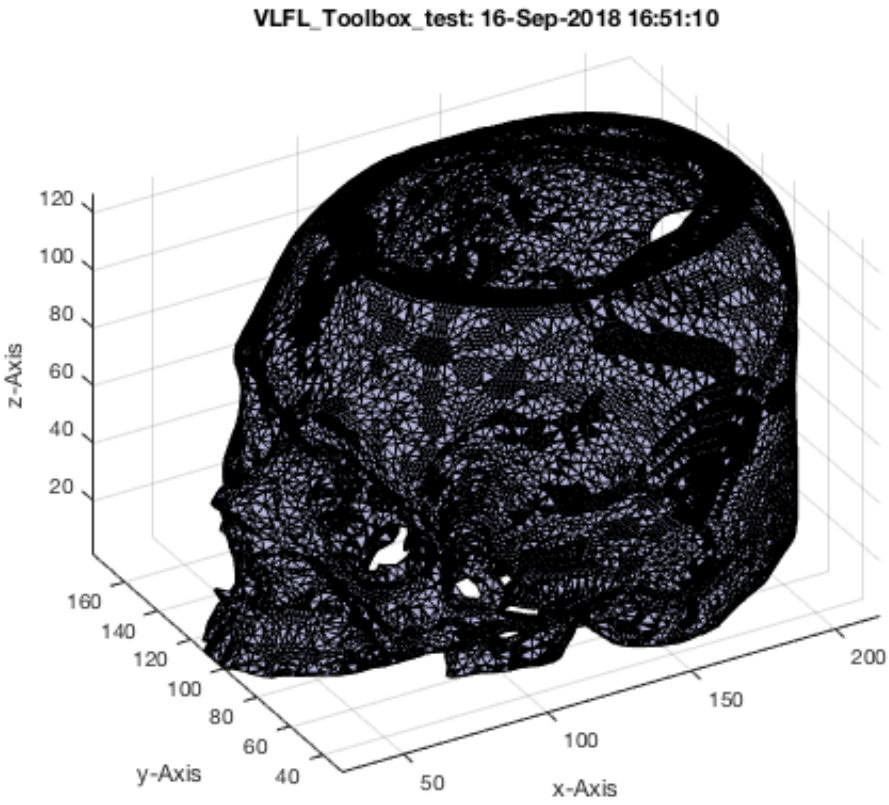
2. Solid skull bone reconstruction using SGofVMdelaunay

```
SG1=SGofVMdelaunay(a>1400,as);       % Takes about 30 seconds
SGfigure; VLFLplotlight (1,1); view(-30,30);
SGplot(SG1,'w');VLFLplotlight(1,1)
```

```
Elapsed time is 4.151191 seconds.
Elapsed time is 6.005731 seconds.
```

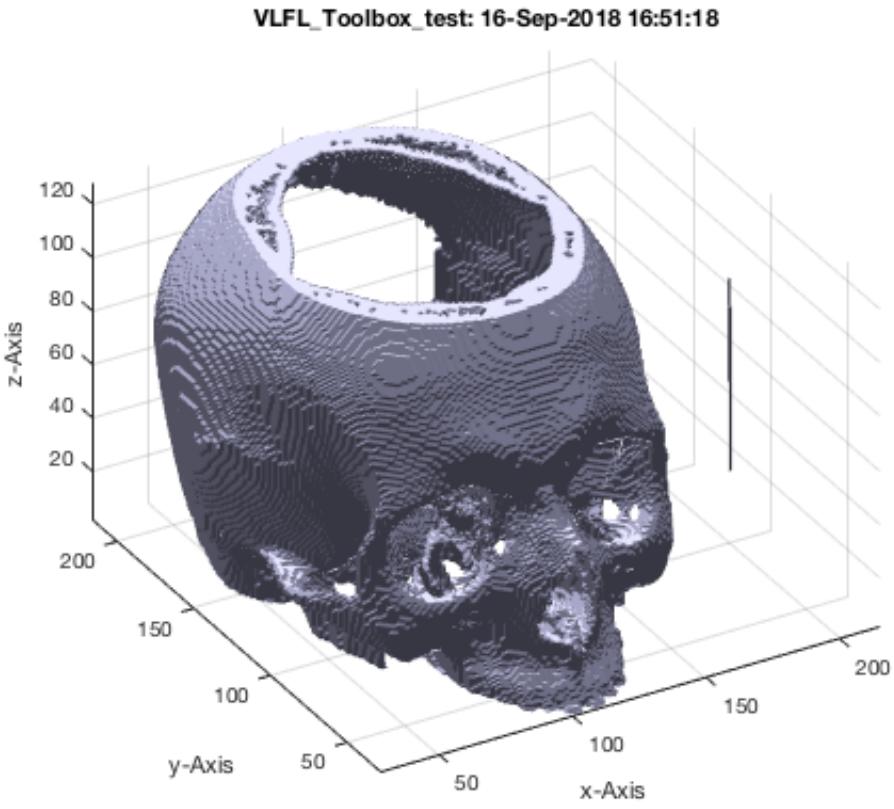



```
VLFLplotlight(0,1);
```

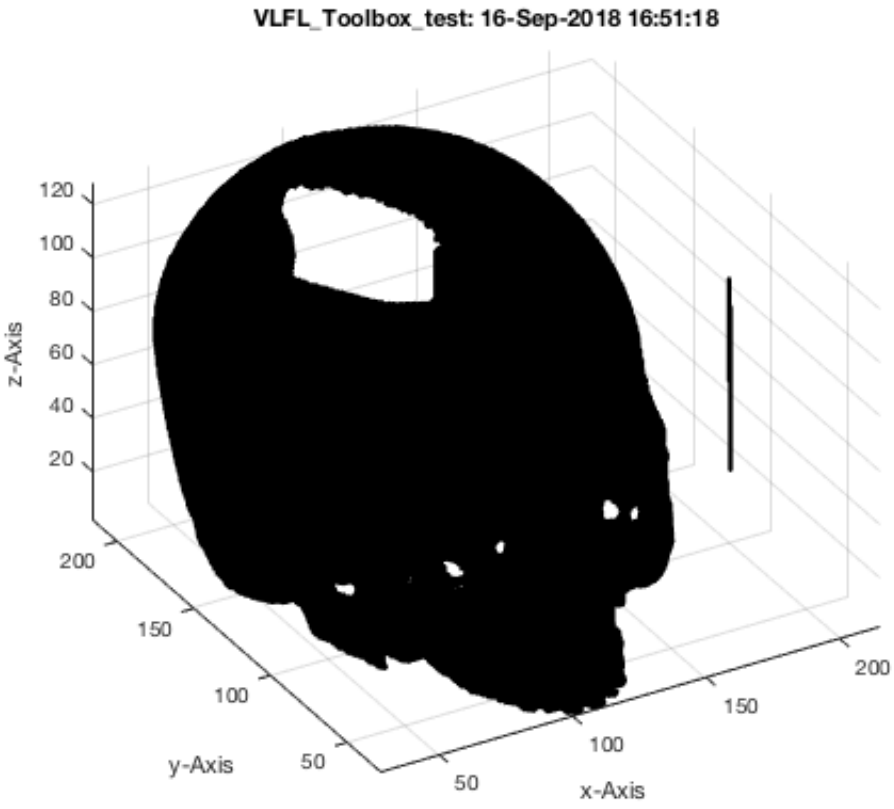


3. Solid skull bone reconstruction using SGofVMisosurface

```
SG2=SGofVMisosurface(a>1400,as);       % Takes about 7 seconds
SGfigure; VLFLplotlight (1,1); view(-30,30);
SGplot(SG2,'w');VLFLplotlight(1,1)
```

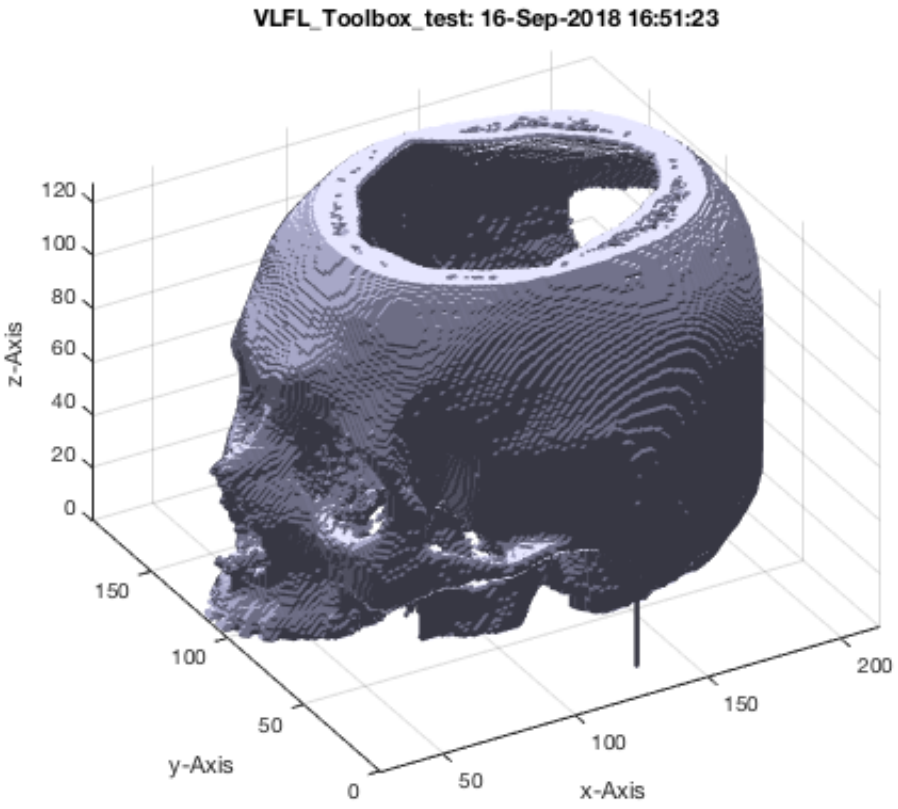



```
VLFLplotlight(0,1);
```

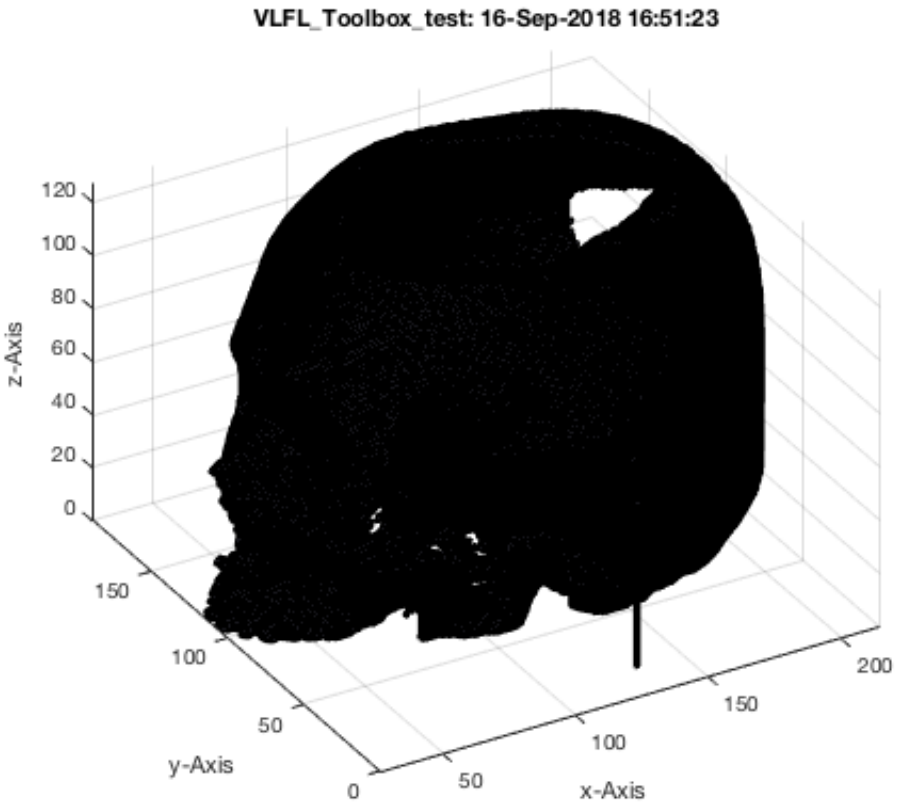


4. Solid skull bone reconstruction using SGofVMmarchcub

```
SG3=SGofVMmarchcube(a>1400,as);       % Takes about 2 seconds
SGfigure; VLFLplotlight (1,1); view(-30,30);
SGplot(SG3,'w');VLFLplotlight(1,1)
```

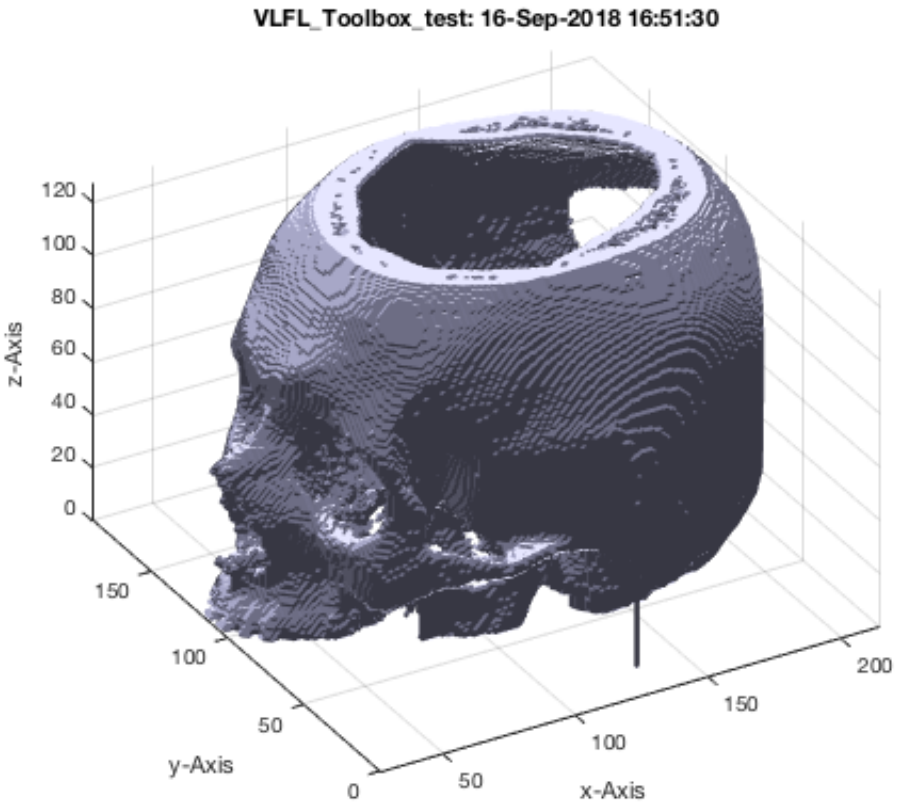



```
VLFLplotlight(0,1);
```

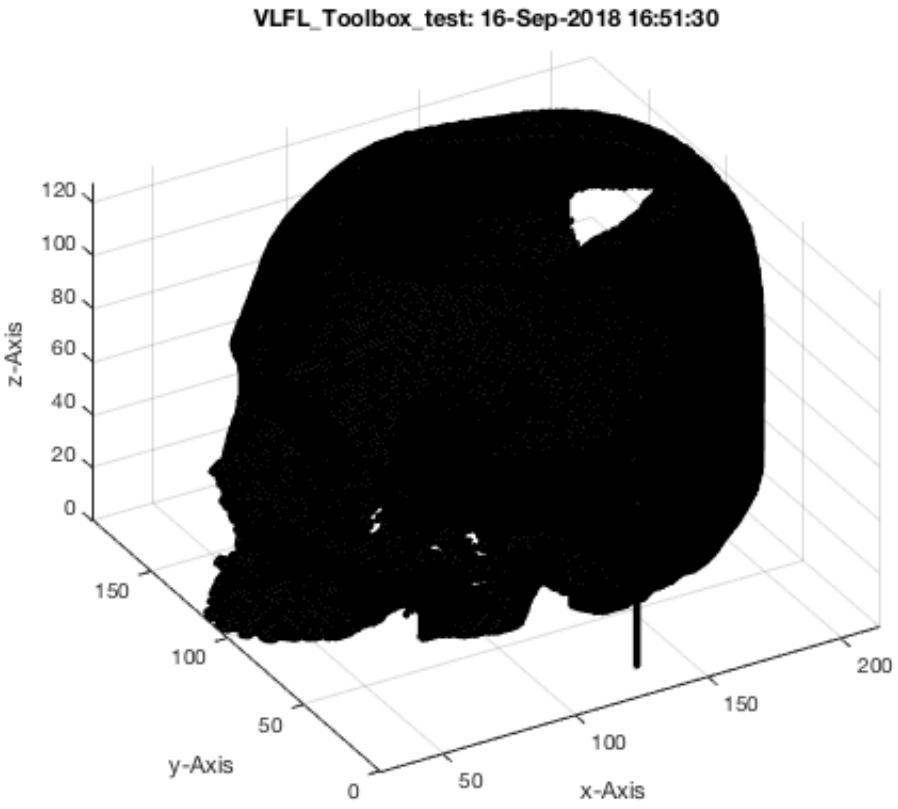


5. Reduce the numbers Facets to 300.000 facets

```
SG4=SGreduceVLFL(SG3,300000); % Takes about 2 seconds
SGfigure; VLFLplotlight (1,1); view(-30,30);
SGplot(SG3,'w');VLFLplotlight(1,1)
```

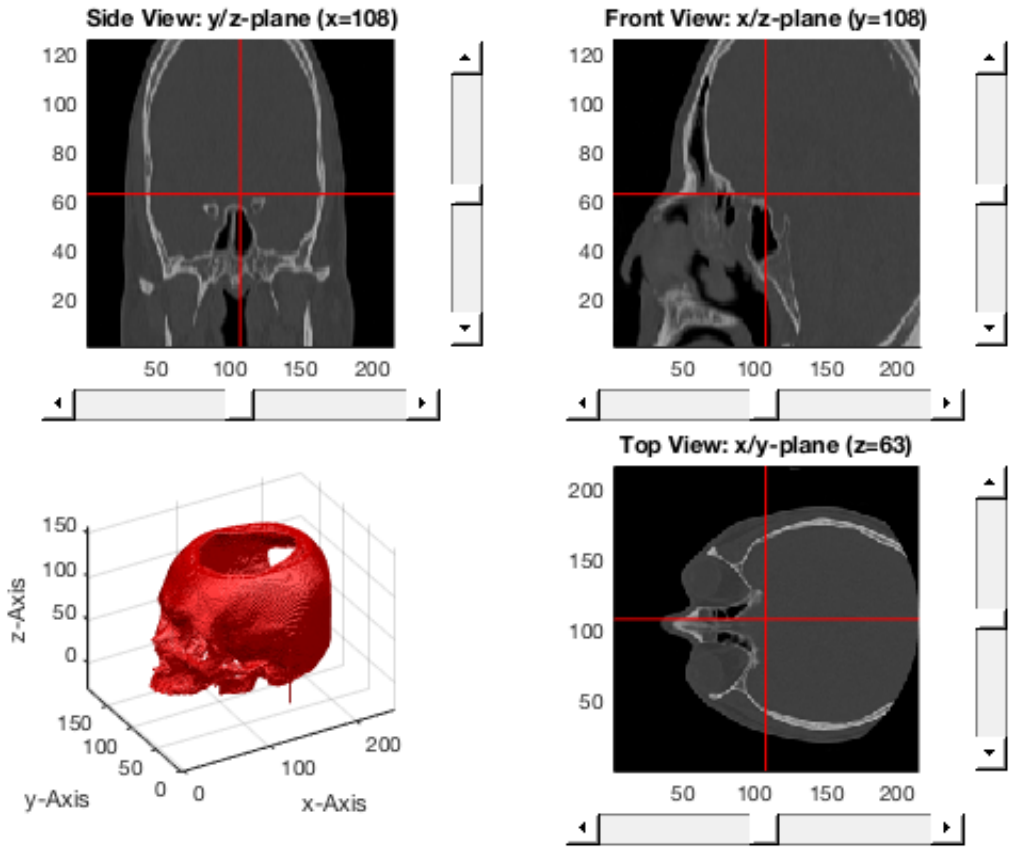



```
VLFLplotlight(0,1);
```



6 Show the Voxel model in quadrant 1-2-4 and surface model in quadrant 3

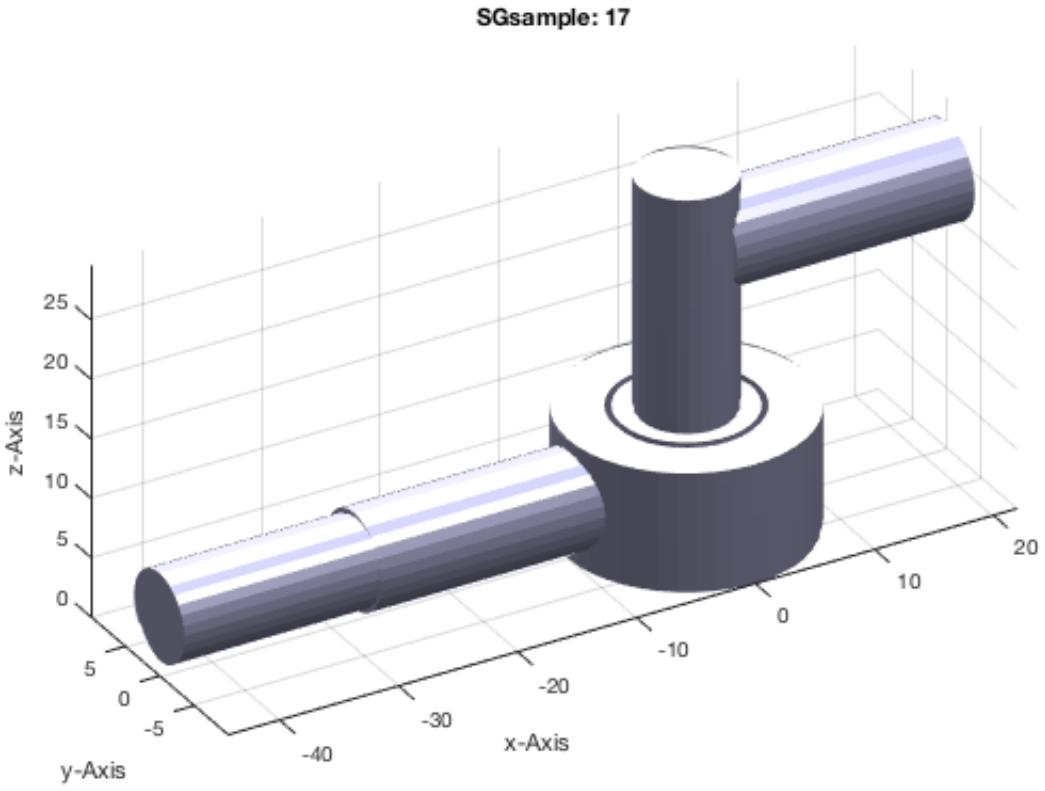
```
VMplot(a,'',SG4)
```



7. Create a surface model and convert it into a Voxel model

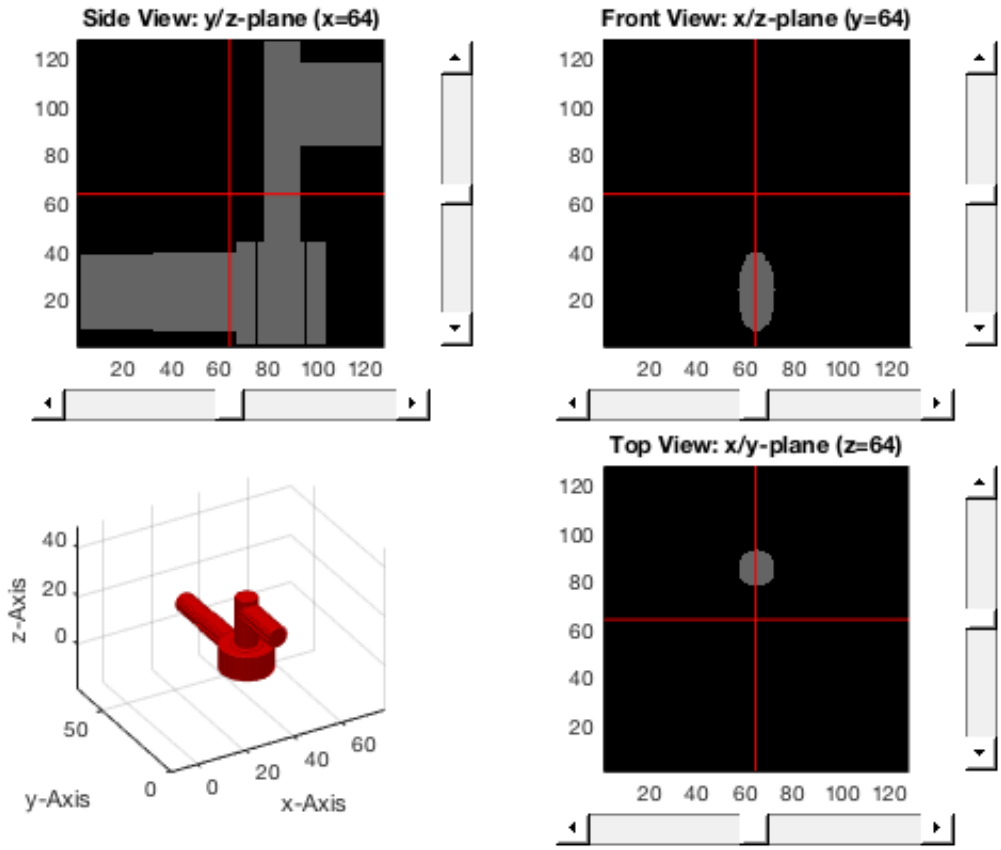
```
SGsample(17); VLFLplotlight(1,1);

[VM,ms,SG3]=VMofSG(SGsample(17),[128 128 128],true);
whos VM
ms
SG3
```

```
VMofSG: 5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
  Name         Size                    Bytes  Class     Attributes

  VM         128x128x128             16777216  double


ms =

    0.5309    0.5309    0.2358


SG3 =

  struct with fields:

    VL: [20614×3 double]
    FL: [60580×3 double]
```



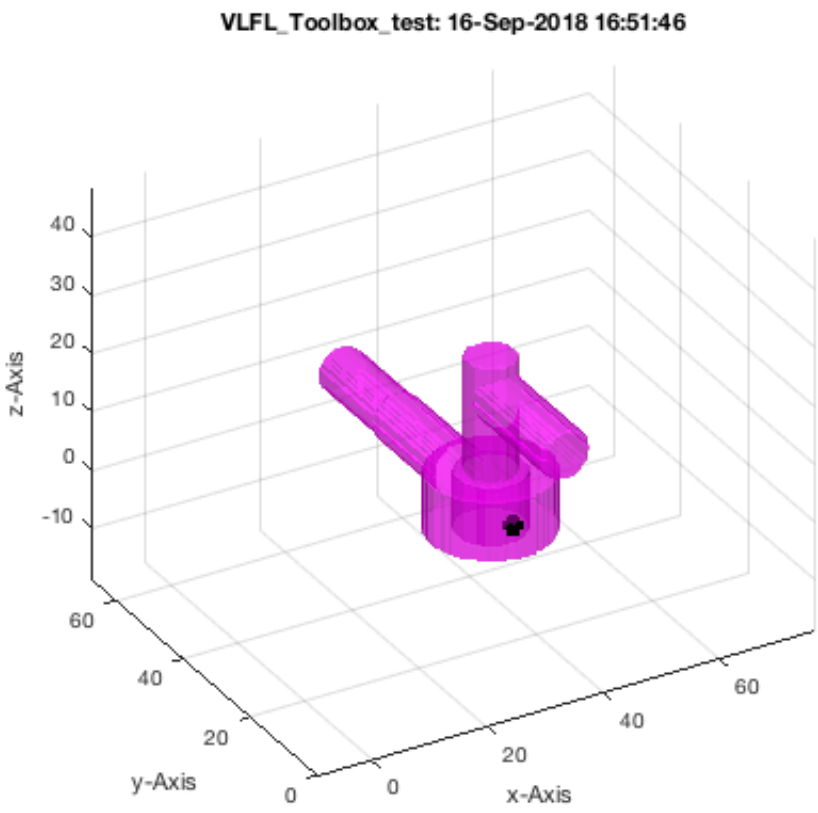
8. Plot the surface model in 4 quadrant plot

```
VMplot(VM,'',SG3)
```



9. Select Point in 3D

```
SGfigure(SG3); view(-30,30); VLFLplotlight (1,.5);
ginput(1); p=select3d; pplot(p,'k*',4); rotate3d on
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:52:40!
Executed 16-Sep-2018 16:52:42 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==================================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 32: Exchanging Data with a FileMaker Database

2017-03-01: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.5 required)
- List of Supported functions
- 1. Getting some help information on the interface
- 2. Open a Database and establishes a connection
- 3. Getting Informations on FileMaker Database Fields
- 4. Retrieving Information
- Closing the Database Connection
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.5 required)

List of Supported functions

- FMhelp - Returning some help text on the function set
- FMinitJDBC - called automatically by FMopen; expecting "fmjdbc.jar" in the search path
- FMopen - to open a database with user name and password
- FMgetFieldTabs - to get informations on the Database
- FMsqlQuery - to send requests or data to the Databse

1. Getting some help information on the interface

```
FMhelp
FMinitJDBC('fmjdbc.jar') % installs the Filemaker JDBC Driver
```

```
ans =

    '  FMhelp - returns a help text for the FileMaker-Matlab interface
       (by Tim Lueth, FileMaker, 2017-FEB-28)

       Tobias Lüddemann did start the connection of FileMaker and Matlab using
       2012b and FileMaker Pro 11. There is a document dated 2014-11-27 at TUM
       MIMED.
       Tim Lueth capsulated the JDBC FileMaker interface using Matlab 2016b
       and FileMaker 13 starting February 2017. The solution described here
       works with Filemaker 13 and later.

       The Matlab Database Toolbox is required. You need a license for that.
       The xDBC Drivers for Filemaker can be downloaded from the Filemaker
       WWW-Site for your Filemaker Version
       The JDBC Driver "fmjdbc.jar" is part of this package.
```

```
        This driver file has to be added to the javaclasspath (which is done by
        the fnctn FMinitJDBC)

        For connecting to the Filemaker App you have to:
        SWITCH ON FILESHARING for ALL Users (Filemaker & Database)
        SWITCH ON ODBC-JDBC-Sharing: for ALL Users (Filemaker & Database)

        Lueth's fnctns to support the connection to Filemaker are:
        FMhelp - This fnctn
        FMinitJDBC - Opens the Driver "fmjdbc.jar"
        FMopen - to open a database with user name and password
        FMgetFieldTabs - to get informations on the Database
        FMsqlQuery - to send requests or data to the Databse
        ...there are some fnctns all starting with capital letters "FM"

         (Status of: 2017-03-01)

        See also: FMhelp, FMinitJDBC, FMopen, FMgetFieldTabs, FMsqlQuery

        LITERATURE:
        Filemaker (2013): "SQL-Referenzhandbuch FM 13",
        https://fmhelp.filemaker.com/docs/13/de/fm13_sql_reference.pdf

        FMhelp

        EXAMPLE: How to use the library after copying "fmjdbc.jar" in a search
        path directory:
        FMinitJDBC('fmjdbc.jar')
        conn=FMopen('Basename.fmp12','user','passw')
        FMgetFieldTabs(conn)
        FMsqlQuery(conn,'SELECT * FROM FileMaker_Tables')

     '

Java class installed: '/Volumes/LUETH-WIN/WIN AIM Matlab Libraries/VLFL-Lib/fmjdbc.jar'
1. Make sure that Filemaker Application's Sharing is ON for ALL USER.
2. Make sure that Filemaker Database's Sharing is ON for ALL USER.
3. Make sure that Filemaker ODBC/JDBC's Sharing is ON for ALL USER.
```

2. Open a Database and establishes a connection

```
conn=FMopen('FileMakerTestBase.fmp12')
```

```
conn =

  connection with properties:

                 DataSource: ''
                   UserName: 'Admin'
                     Driver: 'com.filemaker.jdbc.Driver'
                        URL: 'jdbc:filemaker://localhos ...'
                    Message: ''
                       Type: 'JDBC Connection Object'
  Database Properties:
```

```
                 AutoCommit: 'on'
                   ReadOnly: 'off'
               LoginTimeout: 0
      MaxDatabaseConnections: 0

  Catalog and Schema Information:

            DefaultCatalog: ''
                  Catalogs: {}
                   Schemas: {}

  Database and Driver Information:

        DatabaseProductName: 'FileMaker'
      DatabaseProductVersion: 'ProAdvanced 14.0.3'
                 DriverName: 'FileMaker JDBC'
              DriverVersion: 'FileMaker 13.0.12 JDBC3 ( ...'
```

3. Getting Informations on FileMaker Database Fields

```
FMgetFieldTabs(conn)
```

```
ans =

  7×1 cell array

    {'Doppelte Themen'    }
    {'Frage zu Fragencode'}
    {'FragenCode'         }
    {'Prüfungsfragen'     }
    {'SAME FragenCode'    }
    {'Same Thema'         }
    {'Vorlesungsinhalte'  }


ans =

  7×5 cell array

  Columns 1 through 3

    {'Doppelte Themen'  }    {[1065092]}    {'Vorlesungsinhalte'}
    {'Frage zu Fragen…'}    {[1065098]}    {'Prüfungsfragen'   }
    {'FragenCode'       }    {[1065094]}    {'Vorlesungsinhalte'}
    {'Prüfungsfragen'   }    {[1065089]}    {'Prüfungsfragen'   }
    {'SAME FragenCode'  }    {[1065099]}    {'Prüfungsfragen'   }
    {'Same Thema'       }    {[1065093]}    {'Prüfungsfragen'   }
    {'Vorlesungsinhalte'}    {[1065091]}    {'Vorlesungsinhalte'}

  Columns 4 through 5

    {'FileMakerTestBa…'}    {[ 26]}
```

```
    {'FileMakerTestBa…'}    {[105]}
    {'FileMakerTestBa…'}    {[ 26]}
    {'FileMakerTestBa…'}    {[105]}
    {'FileMakerTestBa…'}    {[105]}
    {'FileMakerTestBa…'}    {[105]}
    {'FileMakerTestBa…'}    {[ 26]}


ans =

  169×7 cell array

  Columns 1 through 3

    {'Doppelte Themen'  }    {'Dopplestundennu…'}    {'decimal'          }
    {'Doppelte Themen'  }    {'Themennummern'    }    {'decimal[30]'      }
    {'Doppelte Themen'  }    {'Themen'           }    {'varchar[30]'      }
    {'Doppelte Themen'  }    {'REL Doppelstund…'}    {'decimal'          }
    {'Doppelte Themen'  }    {'Anzahl der Themen'}    {'varchar'          }
    {'Doppelte Themen'  }    {'CHECK Themennum…'}    {'varchar'          }
    {'Doppelte Themen'  }    {'VAR Fragencode'   }    {'global decimal'   }
    {'Doppelte Themen'  }    {'CALC Fragen zu …'}    {'decimal'          }
    {'Doppelte Themen'  }    {'Fragenrecords'    }    {'decimal[30]'      }
    {'Doppelte Themen'  }    {'REL Fragenrecord' }    {'decimal'          }
    {'Doppelte Themen'  }    {'REL Doppelstund…'}    {'varchar'          }
    {'Frage zu Fragen…'}    {'REC Creation Date'}    {'date'             }
    {'Frage zu Fragen…'}    {'REC Nr'           }    {'decimal'          }
    {'Frage zu Fragen…'}    {'REC Change Date'  }    {'date'             }
    {'Frage zu Fragen…'}    {'REC Change Time'  }    {'time'             }
    {'Frage zu Fragen…'}    {'REC Creation Time'}    {'time'             }
    {'Frage zu Fragen…'}    {'REC Creation Au…'}    {'varchar'          }
    {'Frage zu Fragen…'}    {'REC Change Author'}    {'varchar'          }
    {'Frage zu Fragen…'}    {'Fragen'           }    {'varchar[10]'      }
    {'Frage zu Fragen…'}    {'Fragenskizzen'    }    {'binary[10]'       }
    {'Frage zu Fragen…'}    {'CALC Themenkomp…'}    {'varchar'          }
    {'Frage zu Fragen…'}    {'CALC Fragennumm…'}    {'decimal[10]'      }
    {'Frage zu Fragen…'}    {'VAR i'            }    {'global decimal'   }
    {'Frage zu Fragen…'}    {'VAR n'            }    {'global decimal'   }
    {'Frage zu Fragen…'}    {'Doppelstunden'    }    {'global varchar[…'}
    {'Frage zu Fragen…'}    {'VAR j'            }    {'global decimal'   }
    {'Frage zu Fragen…'}    {'Vorlesungstitel'  }    {'global varchar'   }
    {'Frage zu Fragen…'}    {'CALC Doppelstun…'}    {'global decimal[…'}
    {'Frage zu Fragen…'}    {'Nummer der Dopp…'}    {'decimal'          }
    {'Frage zu Fragen…'}    {'Nummer des Themas'}    {'decimal'          }
    {'Frage zu Fragen…'}    {'REL Thema'        }    {'varchar'          }
    {'Frage zu Fragen…'}    {'CALC Vorlesungs…'}    {'decimal'          }
    {'Frage zu Fragen…'}    {'CALC Thema'       }    {'decimal'          }
    {'Frage zu Fragen…'}    {'CALC Fragekode'   }    {'decimal'          }
    {'Frage zu Fragen…'}    {'CALC Anzahl der…'}    {'decimal'          }
    {'Frage zu Fragen…'}    {'VAR k'            }    {'global decimal'   }
    {'Frage zu Fragen…'}    {'CALC Themen und…'}    {'varchar'          }
    {'Frage zu Fragen…'}    {'Antworten'        }    {'varchar[10]'      }
    {'Frage zu Fragen…'}    {'CALC Folie Title' }    {'decimal'          }
    {'Frage zu Fragen…'}    {'CALC Folie Unte…'}    {'decimal'          }
    {'Frage zu Fragen…'}    {'CHECK Doppelt'    }    {'varchar'          }
    {'Frage zu Fragen…'}    {'STAT TEST'        }    {'decimal'          }
    {'Frage zu Fragen…'}    {'Fragen als Bild'  }    {'binary[10]'       }
```

```
{'Frage zu Fragen…'}    {'Uebungstext'      }    {'varchar'          }
{'Frage zu Fragen…'}    {'Einfache Beschr…'}    {'binary'           }
{'FragenCode'       }    {'Dopplestundennu…'}    {'decimal'          }
{'FragenCode'       }    {'Themennummern'    }    {'decimal[30]'      }
{'FragenCode'       }    {'Themen'           }    {'varchar[30]'      }
{'FragenCode'       }    {'REL Doppelstund…'}    {'decimal'          }
{'FragenCode'       }    {'Anzahl der Themen'}    {'varchar'          }
{'FragenCode'       }    {'CHECK Themennum…'}    {'varchar'          }
{'FragenCode'       }    {'VAR Fragencode'   }    {'global decimal'   }
{'FragenCode'       }    {'CALC Fragen zu …'}    {'decimal'          }
{'FragenCode'       }    {'Fragenrecords'    }    {'decimal[30]'      }
{'FragenCode'       }    {'REL Fragenrecord' }    {'decimal'          }
{'FragenCode'       }    {'REL Doppelstund…'}    {'varchar'          }
{'Prüfungsfragen'   }    {'REC Creation Date'}    {'date'             }
{'Prüfungsfragen'   }    {'REC Nr'           }    {'decimal'          }
{'Prüfungsfragen'   }    {'REC Change Date'  }    {'date'             }
{'Prüfungsfragen'   }    {'REC Change Time'  }    {'time'             }
{'Prüfungsfragen'   }    {'REC Creation Time'}    {'time'             }
{'Prüfungsfragen'   }    {'REC Creation Au…'}    {'varchar'          }
{'Prüfungsfragen'   }    {'REC Change Author'}    {'varchar'          }
{'Prüfungsfragen'   }    {'Fragen'           }    {'varchar[10]'      }
{'Prüfungsfragen'   }    {'Fragenskizzen'    }    {'binary[10]'       }
{'Prüfungsfragen'   }    {'CALC Themenkomp…'}    {'varchar'          }
{'Prüfungsfragen'   }    {'CALC Fragennumm…'}    {'decimal[10]'      }
{'Prüfungsfragen'   }    {'VAR i'            }    {'global decimal'   }
{'Prüfungsfragen'   }    {'VAR n'            }    {'global decimal'   }
{'Prüfungsfragen'   }    {'Doppelstunden'    }    {'global varchar[…'}
{'Prüfungsfragen'   }    {'VAR j'            }    {'global decimal'   }
{'Prüfungsfragen'   }    {'Vorlesungstitel'  }    {'global varchar'   }
{'Prüfungsfragen'   }    {'CALC Doppelstun…'}    {'global decimal[…'}
{'Prüfungsfragen'   }    {'Nummer der Dopp…'}    {'decimal'          }
{'Prüfungsfragen'   }    {'Nummer des Themas'}    {'decimal'          }
{'Prüfungsfragen'   }    {'REL Thema'        }    {'varchar'          }
{'Prüfungsfragen'   }    {'CALC Vorlesungs…'}    {'decimal'          }
{'Prüfungsfragen'   }    {'CALC Thema'       }    {'decimal'          }
{'Prüfungsfragen'   }    {'CALC Fragekode'   }    {'decimal'          }
{'Prüfungsfragen'   }    {'CALC Anzahl der…'}    {'decimal'          }
{'Prüfungsfragen'   }    {'VAR k'            }    {'global decimal'   }
{'Prüfungsfragen'   }    {'CALC Themen und…'}    {'varchar'          }
{'Prüfungsfragen'   }    {'Antworten'        }    {'varchar[10]'      }
{'Prüfungsfragen'   }    {'CALC Folie Title' }    {'decimal'          }
{'Prüfungsfragen'   }    {'CALC Folie Unte…'}    {'decimal'          }
{'Prüfungsfragen'   }    {'CHECK Doppelt'    }    {'varchar'          }
{'Prüfungsfragen'   }    {'STAT TEST'        }    {'decimal'          }
{'Prüfungsfragen'   }    {'Fragen als Bild'  }    {'binary[10]'       }
{'Prüfungsfragen'   }    {'Uebungstext'      }    {'varchar'          }
{'Prüfungsfragen'   }    {'Einfache Beschr…'}    {'binary'           }
{'SAME FragenCode'  }    {'REC Creation Date'}    {'date'             }
{'SAME FragenCode'  }    {'REC Nr'           }    {'decimal'          }
{'SAME FragenCode'  }    {'REC Change Date'  }    {'date'             }
{'SAME FragenCode'  }    {'REC Change Time'  }    {'time'             }
{'SAME FragenCode'  }    {'REC Creation Time'}    {'time'             }
{'SAME FragenCode'  }    {'REC Creation Au…'}    {'varchar'          }
{'SAME FragenCode'  }    {'REC Change Author'}    {'varchar'          }
{'SAME FragenCode'  }    {'Fragen'           }    {'varchar[10]'      }
{'SAME FragenCode'  }    {'Fragenskizzen'    }    {'binary[10]'       }
{'SAME FragenCode'  }    {'CALC Themenkomp…'}    {'varchar'          }
```

```
{'SAME FragenCode'  }    {'CALC Fragennumm…'}    {'decimal[10]'      }
{'SAME FragenCode'  }    {'VAR i'            }    {'global decimal'   }
{'SAME FragenCode'  }    {'VAR n'            }    {'global decimal'   }
{'SAME FragenCode'  }    {'Doppelstunden'    }    {'global varchar[…'}
{'SAME FragenCode'  }    {'VAR j'            }    {'global decimal'   }
{'SAME FragenCode'  }    {'Vorlesungstitel'  }    {'global varchar'   }
{'SAME FragenCode'  }    {'CALC Doppelstun…'}    {'global decimal[…'}
{'SAME FragenCode'  }    {'Nummer der Dopp…'}    {'decimal'          }
{'SAME FragenCode'  }    {'Nummer des Themas'}    {'decimal'          }
{'SAME FragenCode'  }    {'REL Thema'        }    {'varchar'          }
{'SAME FragenCode'  }    {'CALC Vorlesungs…'}    {'decimal'          }
{'SAME FragenCode'  }    {'CALC Thema'       }    {'decimal'          }
{'SAME FragenCode'  }    {'CALC Fragekode'   }    {'decimal'          }
{'SAME FragenCode'  }    {'CALC Anzahl der…'}    {'decimal'          }
{'SAME FragenCode'  }    {'VAR k'            }    {'global decimal'   }
{'SAME FragenCode'  }    {'CALC Themen und…'}    {'varchar'          }
{'SAME FragenCode'  }    {'Antworten'        }    {'varchar[10]'      }
{'SAME FragenCode'  }    {'CALC Folie Title' }    {'decimal'          }
{'SAME FragenCode'  }    {'CALC Folie Unte…'}    {'decimal'          }
{'SAME FragenCode'  }    {'CHECK Doppelt'    }    {'varchar'          }
{'SAME FragenCode'  }    {'STAT TEST'        }    {'decimal'          }
{'SAME FragenCode'  }    {'Fragen als Bild'  }    {'binary[10]'       }
{'SAME FragenCode'  }    {'Uebungstext'      }    {'varchar'          }
{'SAME FragenCode'  }    {'Einfache Beschr…'}    {'binary'           }
{'Same Thema'       }    {'REC Creation Date'}    {'date'             }
{'Same Thema'       }    {'REC Nr'           }    {'decimal'          }
{'Same Thema'       }    {'REC Change Date'  }    {'date'             }
{'Same Thema'       }    {'REC Change Time'  }    {'time'             }
{'Same Thema'       }    {'REC Creation Time'}    {'time'             }
{'Same Thema'       }    {'REC Creation Au…'}    {'varchar'          }
{'Same Thema'       }    {'REC Change Author'}    {'varchar'          }
{'Same Thema'       }    {'Fragen'           }    {'varchar[10]'      }
{'Same Thema'       }    {'Fragenskizzen'    }    {'binary[10]'       }
{'Same Thema'       }    {'CALC Themenkomp…'}    {'varchar'          }
{'Same Thema'       }    {'CALC Fragennumm…'}    {'decimal[10]'      }
{'Same Thema'       }    {'VAR i'            }    {'global decimal'   }
{'Same Thema'       }    {'VAR n'            }    {'global decimal'   }
{'Same Thema'       }    {'Doppelstunden'    }    {'global varchar[…'}
{'Same Thema'       }    {'VAR j'            }    {'global decimal'   }
{'Same Thema'       }    {'Vorlesungstitel'  }    {'global varchar'   }
{'Same Thema'       }    {'CALC Doppelstun…'}    {'global decimal[…'}
{'Same Thema'       }    {'Nummer der Dopp…'}    {'decimal'          }
{'Same Thema'       }    {'Nummer des Themas'}    {'decimal'          }
{'Same Thema'       }    {'REL Thema'        }    {'varchar'          }
{'Same Thema'       }    {'CALC Vorlesungs…'}    {'decimal'          }
{'Same Thema'       }    {'CALC Thema'       }    {'decimal'          }
{'Same Thema'       }    {'CALC Fragekode'   }    {'decimal'          }
{'Same Thema'       }    {'CALC Anzahl der…'}    {'decimal'          }
{'Same Thema'       }    {'VAR k'            }    {'global decimal'   }
{'Same Thema'       }    {'CALC Themen und…'}    {'varchar'          }
{'Same Thema'       }    {'Antworten'        }    {'varchar[10]'      }
{'Same Thema'       }    {'CALC Folie Title' }    {'decimal'          }
{'Same Thema'       }    {'CALC Folie Unte…'}    {'decimal'          }
{'Same Thema'       }    {'CHECK Doppelt'    }    {'varchar'          }
{'Same Thema'       }    {'STAT TEST'        }    {'decimal'          }
{'Same Thema'       }    {'Fragen als Bild'  }    {'binary[10]'       }
{'Same Thema'       }    {'Uebungstext'      }    {'varchar'          }
```

```
  {'Same Thema'       }    {'Einfache Beschr…'}    {'binary'           }
  {'Vorlesungsinhalte'}    {'Dopplestundennu…'}    {'decimal'          }
  {'Vorlesungsinhalte'}    {'Themennummern'    }    {'decimal[30]'      }
  {'Vorlesungsinhalte'}    {'Themen'           }    {'varchar[30]'      }
  {'Vorlesungsinhalte'}    {'REL Doppelstund…'}    {'decimal'          }
  {'Vorlesungsinhalte'}    {'Anzahl der Themen'}    {'varchar'          }
  {'Vorlesungsinhalte'}    {'CHECK Themennum…'}    {'varchar'          }
  {'Vorlesungsinhalte'}    {'VAR Fragencode'   }    {'global decimal'   }
  {'Vorlesungsinhalte'}    {'CALC Fragen zu …'}    {'decimal'          }
  {'Vorlesungsinhalte'}    {'Fragenrecords'    }    {'decimal[30]'      }
  {'Vorlesungsinhalte'}    {'REL Fragenrecord' }    {'decimal'          }
  {'Vorlesungsinhalte'}    {'REL Doppelstund…'}    {'varchar'          }

Columns 4 through 7

  {[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
  {[ 2]}    {'Normal'    }    {[30]}    {[ 5]}
  {[ 3]}    {'Normal'    }    {[30]}    {[ 3]}
  {[ 4]}    {'Calculated'}    {[ 1]}    {[ 5]}
  {[ 5]}    {'Calculated'}    {[ 1]}    {[ 3]}
  {[ 6]}    {'Calculated'}    {[ 1]}    {[ 0]}
  {[ 7]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[ 8]}    {'Calculated'}    {[ 1]}    {[ 0]}
  {[ 9]}    {'Normal'    }    {[30]}    {[ 1]}
  {[12]}    {'Calculated'}    {[ 1]}    {[ 0]}
  {[13]}    {'Calculated'}    {[ 1]}    {[ 7]}
  {[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
  {[ 2]}    {'Normal'    }    {[ 1]}    {[ 2]}
  {[ 3]}    {'Normal'    }    {[ 1]}    {[ 3]}
  {[ 5]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[ 6]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[ 7]}    {'Normal'    }    {[ 1]}    {[ 2]}
  {[ 8]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[10]}    {'Normal'    }    {[10]}    {[ 1]}
  {[11]}    {'Normal'    }    {[10]}    {[ 1]}
  {[14]}    {'Calculated'}    {[ 1]}    {[ 6]}
  {[15]}    {'Normal'    }    {[10]}    {[ 5]}
  {[16]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[17]}    {'Normal'    }    {[ 1]}    {[ 3]}
  {[18]}    {'Normal'    }    {[15]}    {[ 6]}
  {[20]}    {'Normal'    }    {[ 1]}    {[ 2]}
  {[21]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[23]}    {'Normal'    }    {[15]}    {[ 9]}
  {[24]}    {'Normal'    }    {[ 1]}    {[ 1]}
  {[27]}    {'Normal'    }    {[ 1]}    {[ 0]}
  {[28]}    {'Calculated'}    {[ 1]}    {[ 6]}
  {[29]}    {'Calculated'}    {[ 1]}    {[ 9]}
  {[30]}    {'Calculated'}    {[ 1]}    {[18]}
  {[31]}    {'Calculated'}    {[ 1]}    {[ 0]}
  {[32]}    {'Calculated'}    {[ 1]}    {[ 0]}
  {[33]}    {'Normal'    }    {[ 1]}    {[ 3]}
  {[35]}    {'Calculated'}    {[ 1]}    {[ 1]}
  {[36]}    {'Normal'    }    {[10]}    {[ 2]}
  {[38]}    {'Calculated'}    {[ 1]}    {[28]}
  {[39]}    {'Calculated'}    {[ 1]}    {[12]}
  {[40]}    {'Calculated'}    {[ 1]}    {[ 4]}
  {[42]}    {'Summary'   }    {[ 1]}    {[ 1]}
```

```
{[43]}    {'Normal'    }    {[10]}    {[ 2]}
{[45]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[46]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 2]}    {'Normal'    }    {[30]}    {[ 5]}
{[ 3]}    {'Normal'    }    {[30]}    {[ 3]}
{[ 4]}    {'Calculated'}    {[ 1]}    {[ 5]}
{[ 5]}    {'Calculated'}    {[ 1]}    {[ 3]}
{[ 6]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[ 7]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 8]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[ 9]}    {'Normal'    }    {[30]}    {[ 1]}
{[12]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[13]}    {'Calculated'}    {[ 1]}    {[ 7]}
{[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 2]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 3]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[ 5]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 6]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 7]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 8]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[10]}    {'Normal'    }    {[10]}    {[ 1]}
{[11]}    {'Normal'    }    {[10]}    {[ 1]}
{[14]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[15]}    {'Normal'    }    {[10]}    {[ 5]}
{[16]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[17]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[18]}    {'Normal'    }    {[15]}    {[ 6]}
{[20]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[21]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[23]}    {'Normal'    }    {[15]}    {[ 9]}
{[24]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[27]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[28]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[29]}    {'Calculated'}    {[ 1]}    {[ 9]}
{[30]}    {'Calculated'}    {[ 1]}    {[18]}
{[31]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[32]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[33]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[35]}    {'Calculated'}    {[ 1]}    {[ 1]}
{[36]}    {'Normal'    }    {[10]}    {[ 2]}
{[38]}    {'Calculated'}    {[ 1]}    {[28]}
{[39]}    {'Calculated'}    {[ 1]}    {[12]}
{[40]}    {'Calculated'}    {[ 1]}    {[ 4]}
{[42]}    {'Summary'   }    {[ 1]}    {[ 1]}
{[43]}    {'Normal'    }    {[10]}    {[ 2]}
{[45]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[46]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 2]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 3]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[ 5]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 6]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 7]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 8]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[10]}    {'Normal'    }    {[10]}    {[ 1]}
{[11]}    {'Normal'    }    {[10]}    {[ 1]}
```

```
{[14]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[15]}    {'Normal'    }    {[10]}    {[ 5]}
{[16]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[17]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[18]}    {'Normal'    }    {[15]}    {[ 6]}
{[20]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[21]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[23]}    {'Normal'    }    {[15]}    {[ 9]}
{[24]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[27]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[28]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[29]}    {'Calculated'}    {[ 1]}    {[ 9]}
{[30]}    {'Calculated'}    {[ 1]}    {[18]}
{[31]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[32]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[33]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[35]}    {'Calculated'}    {[ 1]}    {[ 1]}
{[36]}    {'Normal'    }    {[10]}    {[ 2]}
{[38]}    {'Calculated'}    {[ 1]}    {[28]}
{[39]}    {'Calculated'}    {[ 1]}    {[12]}
{[40]}    {'Calculated'}    {[ 1]}    {[ 4]}
{[42]}    {'Summary'   }    {[ 1]}    {[ 1]}
{[43]}    {'Normal'    }    {[10]}    {[ 2]}
{[45]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[46]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 2]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 3]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[ 5]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 6]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[ 7]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[ 8]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[10]}    {'Normal'    }    {[10]}    {[ 1]}
{[11]}    {'Normal'    }    {[10]}    {[ 1]}
{[14]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[15]}    {'Normal'    }    {[10]}    {[ 5]}
{[16]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[17]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[18]}    {'Normal'    }    {[15]}    {[ 6]}
{[20]}    {'Normal'    }    {[ 1]}    {[ 2]}
{[21]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[23]}    {'Normal'    }    {[15]}    {[ 9]}
{[24]}    {'Normal'    }    {[ 1]}    {[ 1]}
{[27]}    {'Normal'    }    {[ 1]}    {[ 0]}
{[28]}    {'Calculated'}    {[ 1]}    {[ 6]}
{[29]}    {'Calculated'}    {[ 1]}    {[ 9]}
{[30]}    {'Calculated'}    {[ 1]}    {[18]}
{[31]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[32]}    {'Calculated'}    {[ 1]}    {[ 0]}
{[33]}    {'Normal'    }    {[ 1]}    {[ 3]}
{[35]}    {'Calculated'}    {[ 1]}    {[ 1]}
{[36]}    {'Normal'    }    {[10]}    {[ 2]}
{[38]}    {'Calculated'}    {[ 1]}    {[28]}
{[39]}    {'Calculated'}    {[ 1]}    {[12]}
{[40]}    {'Calculated'}    {[ 1]}    {[ 4]}
{[42]}    {'Summary'   }    {[ 1]}    {[ 1]}
{[43]}    {'Normal'    }    {[10]}    {[ 2]}
```

```
    {[45]}    {'Normal'    }    {[ 1]}    {[ 0]}
    {[46]}    {'Normal'    }    {[ 1]}    {[ 0]}
    {[ 1]}    {'Normal'    }    {[ 1]}    {[ 2]}
    {[ 2]}    {'Normal'    }    {[30]}    {[ 5]}
    {[ 3]}    {'Normal'    }    {[30]}    {[ 3]}
    {[ 4]}    {'Calculated'}    {[ 1]}    {[ 5]}
    {[ 5]}    {'Calculated'}    {[ 1]}    {[ 3]}
    {[ 6]}    {'Calculated'}    {[ 1]}    {[ 0]}
    {[ 7]}    {'Normal'    }    {[ 1]}    {[ 1]}
    {[ 8]}    {'Calculated'}    {[ 1]}    {[ 0]}
    {[ 9]}    {'Normal'    }    {[30]}    {[ 1]}
    {[12]}    {'Calculated'}    {[ 1]}    {[ 0]}
    {[13]}    {'Calculated'}    {[ 1]}    {[ 7]}
```

4. Retrieving Information

```
FMsqlQuery(conn ,'Select "Fragen" from "Prüfungsfragen"')
```

```
ans =

  324×1 cell array

    {'Nennen Sie die 14 Dokumentationsschritte im Goldstandard der ingenieurwissenschaftlic
hen Forschung?'
                                                        }
    {'Beschreiben Sie die unterschiedlichen Aufgaben der Biologie, Pharmazie, Humanbiologie
, medizinischer Physik, Humanmedizin und  Zahnmedizin'
                                                        }
    {'Nennen Sie mehrere Lehrberufe und akademische Berufe in der Zahnheilkunde!'

                                                        }
    {'Welches sind die drei Säulen eines Klinikums und wie heißen die Leitungsfunktionen?'

                                                        }
    {'Welche Aufgabe hat ein Wellness-Hotel oder Kurhotel? Wer besucht diese? Wer bezahlt d
ie Leistung? Welche Berufe sind dort anzutreffen?'
                                                        }
    {'Was ist die kassenärztliche Vereinigung und wen vertritt diese?'

                                                        }
    {'Was gehört in den Stand der Technik?'

                                                        }
    {'Was gehört in die Beschreibung von Nachteilen?'

                                                        }
    {'Wie gliedert man eine Aufgabestellung'

                                                        }
    {'Wie beschreibt man Vorteile einer technischen Innovation ?'

                                                        }
    {'Was ist einen statische Beschreibung eines technischen Systems mit seinen Schnittstel
```

len?'
}
{'Was ist eine dynamische Beschreibung einer Innovation?'

}
{'Was beschreiben die Unterscheidungsmerkmale.'

}
{'Wozu dient die Beschreibung des Labor- und Geräteaufbaus'

}
{'Welche Aufgabe hat die Beschreibung des Messverfahren?'

}
{'Welche Aufgabe hat die Beschreibung des Experiments.'

}
{'Welche Beschreibung ist notwendig, um die Messwerte in ein Ergebnis umzuformen?'

}
{'Welche Aufgabe hat die Zusammenfassung der Ergebnisse?'

}
{'Warum müssen Sie Ihre Ergebnisse publizieren'

}
{'Was ist der Citation Index Factor und wer gibt diesen heraus'

}
{'Welchen Stellenwert haben die IEEE die ASME und der VDI, BMT, DGBMT'

}
{'Was ist ein Patent und wofür wird es verliehen. Was ist ein Erfinder und was ist ein Anmelder. Wem gehört das Patent?'
}
{'Was geschieht bei der Patentprüfung und was kostet diese beim Patentamt.'

}
{'Wo und wie können Sie eine Patentrecherche selbst durchführen'

}
{'Was ist ein Gebrauchsmuster und wie lange haben Sie damit Rechtsschutz'

}
{'Dürfen Sie die Patentidee zuvor jemanden mitteilen oder diese publizieren'

}
{'Wann können Sie eine Firma verklagen, wenn diese ihr Patent verletzt?'

}
{'Beschreiben Sie welche Inhalte in die Problemstellung einer normalen technischen Dokumentation gehören'
}
{'Welches sind die einzelnen Schritte von einer Patentanmeldung bis hin zu der Anmeldung der Patente auf vielen Ländern der Welt'
}
{'Gibt es Richtlinien, wie gute wissenschaftliche Praxis aussieht und wer definiert die

```
se Richtlinien?'
                                                    }
    {'Was geschieht bei der Registrierung von 2 Koordinatensystemen bzw. was wird dabei ber
echnet?'
                                                    }
    {'Beschreiben Sie das RMS-Maß für die Genauigkeitsangabe einer Registrierungsmatrix, di
e aus den Punktepaaren berechnen können.'
                                                    }
    {'Welche Effekte führen zu einem Positionsmessfehler bei der Verwendung von Navigations
systemen?'
                                                    }
    {'Welches ist die Aufgabe bei der Registrierung von Patient und Bilddaten? Welche Koord
inatensysteme werden dabei registriert?'
                                                    }
    {'Nennen Sie 6 Regeln für die Auswahl von Landmarken am Patienten/Messobjekt?'

                                                    }
    {'Nennen Sie das Gesetz mit Nummer, dass die Anwendung spezieller  Normen und Regeln vo
rschreibt, die für Medizinprodukte gelten?'
                                                    }
    {'Welche Formen der Bildgebung kennen Sie?'

                                                    }
    {'Welche Bildmodalitäten kennen Sie?'

                                                    }
    {'Nennen Sie Beispiele für eindimensionale Bilddaten?'

                                                    }
    {'Nennen Sie 7 Geräte der Funktionsdiagnostik?'

                                                    }
    {'Zeichnen Sie die schematische Darstellung einer Röntgenquelle?'

                                                    }
    {'Nennen Sie zwei aktuelle Prinzipien zur Aufzeichnung von Röntgenstrahlung?'

                                                    }
    {'Beschreiben Sie die Funktion eines C-Bogens?'

                                                    }
    {'Wie funktioniert die Verwischungstomographie?'

                                                    }
    {'Wie funktioniert eine Algebraische Rekonstruktion?'

                                                    }
    {'Was ist der Unterschied zwischen einer 3D-CT Aufnahme und einer Cone-Beam-CT Aufname
aus technischer Sicht?'
                                                    }
    {'Wie funktioniert das CT?'

                                                    }
    {'Was bedeutet Angiographie?'

                                                    }
    {'Auf welche Art und Weise werden Kern-Spins ausgerichtet?'
```

```
                                                    }
    {'Wie kann man die Feldstärken messen?'

                                                    }
    {'Wie funktioniert das MRT?'

                                                    }
    {'Was bedeutet funktionales MRT?'

                                                    }
    {'Was bedeutet IR Tomographie?'

                                                    }
    {'Auf welcher Basis funktionieren Gamma-Sensoren?'

                                                    }
    {'Auf welcher Basis funktioniert Ultraschallerzeugung?'

                                                    }
    {'Was ist der A-Mode und wird das Gerät dann betrieben?'

                                                    }
    {'Was ist ein Phased Array?'

                                                    }
    {'Wie funktioniert ein Mikroskop?'

                                                    }
    {'Wie funktioniert ein Endoskop?'

                                                    }
    {'Welche Bedeutung haben Fluoreszenz-Marker?'

                                                    }
    {'Wie wird ein typisches 2D-Bild auf einer Festplatte abgelegt?'

                                                    }
    {'Wie wird typischerweise ein 3D-Bild abgelegt? '

                                                    }
    {'Wie wird typischerweise ein Video abgelegt?'

                                                    }
    {'Was bedeutet DICOM 2.0?'

                                                    }
    {'Welche Komponenten definieren ein Gerät und auf welche Norm bezieht sich diese Defini
tion?'
                                                    }
    {'Welche Aufgabe hat das Betriebssystem Windows XP?'

                                                    }
    {'Welche Aufgaben hat ein Echtzeitbetriebssystem und können dort mehrere Programme quas
iparallel ablaufen?'
                                                    }
    {'Wie werden Eingabegeräte angeschlossen. Nennen Sie Bussysteme (je 2)'
```

```
                                                    }
    {'Wie können unterschiedliche Programme/Prozesse/Tasks untereinander Daten austauschen?
'
                                                    }
    {'Unterstützt XP den Zugriff auf Datenträger?'

                                                    }
    {'Wie erzeugt man eine feste Taktfrequenz für das Aufrufen/Abarbeiten von Gerätesteueru
ngen bzw. Regelschleifen?'
                                                    }
    {'Wie ist ein Bildspeicher organisiert und wie kommen die Bilddaten auf das Display (DR
AM/VRAM)'
                                                    }
    {'Welche Formen der 3D-Darstellung kennen Sie?'

                                                    }
    {'Zeichnen Sie in ein schematisches Volumenbild V(u,v,w) die Schnittbilder Iwp, Iup. Iv
p ein, die für das definierte Voxel v=(up,vp,wp) festgelegt sind?'
                                                    }
    {'Zeichnen Sie ein schematisches Bild, wie die Tiefeninformation aus dem Volumenbild be
rechnet werden kann?'
                                                    }
    {'Zeichnen Sie schematisch das Konzept der Mischung von 3D-Bildern mit Schnittbildern u
nd Aufsichtbildern. Wie sie ein Mischbild frontal aus.'
                                                    }
    {'Beschreiben Sie eine Koordinaten-Transformation mit der DH-Matrix. Was steht in der 4
x4 Matrix?'
                                                    }
    {'Geben Sie die drei Einheitsvektoren des kartesischen Koordinatensystems an'

                                                    }
    {'Geben Sie formal die Translation von Punkt p1 nach Punkt p2 an. Geben Sie auch die Tr
anslation von p2 nach p1 an. Worin unterscheiden sich die beiden Translationen?'
                                                    }
    {'Beschreiben Sie die Rotationsmatrix um den Winkel 0.'

                                                    }
    {'Wie kehrt man eine Rotations um?'

                                                    }
    {'Wie werden Translationen verkettet? Gilt dabei das Kommutativgesetz?'

                                                    }
    {'Wodurch wird ein Koordinatensystem beschrieben?'

                                                    }
    {'Was sind homogene Koordinaten und wodurch unterscheiden Sie sich von normalen Koordin
aten?'
                                                    }
    {'Wie lautet die Inverse einer homogenen Transformationsmatrix?'

                                                    }
    {'Wie werden homogene Transformationsmatrizen verkettet?'

                                                    }
    {'Wieviele Möglichkeiten der Winkeldarstellung einer Rotationsmatrix sind für die Euler
```

-Winkel möglich?'
}
{'Geben Sie eine Vektornorm an?'

}
{'Geben Sie das Skalarprodukt zweier Vektoren a, b (mit Koordinaten) an?'

}
{'Geben Sie das Kreuzprodukt zwischen zwei Vektoren an? Warum heißt es Kreuzprodukt?'

}
{'Was ist ein stereotaktischer Rahmen und wie wird dieser benutzt?'

}
{'Zeichnen Sie die minimalen Komponenten, die für ein Navigationssystem erforderlich si nd. Zeichnen Sie ein, welche Kommunikationsverbindungen es geben muss.'
}
{'Wozu dient die Kamera und wie häufig misst diese die Position der Messmarken?'

}
{'Wie funktionieren aktive Messmarken und Zeilenkameras? Wie viele Zeilenkameras sind n otwendig?'
}
{'Welche Bildschirme sind bei klinischen Navigationssystemen bekannt? Nennen Aufgaben f ür kleine, mittlere und große Bildschirme?'
}
{'Geben Sie die 6 Schritte der bildgestützten Navigation an.'

}
{'Was bedeutet Echtzeit-Navigation und worin unterscheidet Sie sich grundsätzlich von d er CT/MRT-gestützten Navigation? Welche Anforderungen werden an die Bildgebung gestellt? Ne nnen Sie ein Beispiel für die Bildgebung?' }
{'Aus welchen 7 Schritten besteht die planungsbasierte Navigation? Worin unterscheidet Sie sich von der normalen CT/MRT-bildgestützten Navigation?'
}
{'Bei der Planung von dentalen Implantaten sollen zwei Kriterien erfüllt werden: Welche s sind diese beiden Kriterien?'
}
{'Was ist ein Navigationsbogen und welche drei Aufgaben erfüllt dieser Bogen? '

}
{'Was geschieht nach dem Einlesen der Schichtbilder am Bildschirm?'

}
{'Welche Schritte werden bei der Planung von Implantaten durchgeführt?'

}
{'Wie wird der Messmarken-Tracker an dem Handstück befestigt?'

}
{'Welche Bedeutung hat die Lage des Patienten auf dem Tisch. Welche Rolle spielt sitzen d oder liegend? '
}
{'Was geschieht bei der Kalibration des Instruments?'

}
{'Auf welche Weise erfolgt die Patientenregistrierung mit der Bissschiene und dem Navig

ationsbogen?'
}
{'Was genau sieht man am Fadenkreuz und warum ist die Anzeige so wichtig?'

}
{'Geben Sie die homogene Transformationsmatrix für die Pose des Patienten im Kamerakoor dinatensystem an. Welche Information enthalten die ersten drei Zeilen der ersten drei Spalt en der Matrix?'
}
{'Welches elektrisches Interface benutzt eine IR-Stereomesskamera? Welche Interface ben utzen Videokameras? Was ist in die Kamera integriert?'
}
{'Zeichnen Sie eine Kamera und einen Messmarkentracker mit 3 Punkten!'

}
{'Zeichnen Sie eine Tabelle zur Definition der Geometrien von Messmarken-Tracker?'

}
{'Welche 4 Formen der Befestigung von Patiententracker am Patienten gibt es? '

}
{'Was geschieht bei dem Einmessen von Trackergeometrien?'

}
{'Welche Form hat der Arbeitsraum eine IR-Stereomesskamera?'

}
{'Wie funktionieren elektromagnetische Messsysteme?'

}
{'null'

}
{'null'

}
{'null'

}
{'Wie werden Materialien klassifiziert (Werkstoffkunde)?'

}
{'Beschreiben Sie den Aufbau eines OP-Tisch mit Lafette und seine Verwendung im OP.'

}
{'Unterscheiden Sie bitte qualitative Aussagen von quantitativen Aussagen. Welche Aussa gen sind leichter nachprüfbar und objektiver?'
}
{'Was bedeuten die Begriffe: Eigenschaft, Messgröße, Wert und Einheit?'

}
{'Können Sie aus einer neuen Innovativen Idee die daraus vermutlich entstehenden Vortei le ableiten? Wie hoch ist dieser Aufwand? Was passiert wenn Sie Vorteile entdecken, die Sie nicht direkt vorhersehbar sind?'
}
{'Wie sieht der zeitliche Zusammenhang aus zwischen der „Erwartung von Vorteilen“ und d er Planung der Messung der „Vorteile“?'
}
{'Warum spielt der Zusammenhang zwischen einem Messwert und einer daraus abgeleiteten E

igenschaftsgröße eine wichtige Rolle?'
}
{'Dürfen Sie ein Gerät bauen, damit Messwerte aufnehmen, daraus eine Formel entwickeln und dann behaupten, die Formel beschriebe das Gerät? Wie macht man es richtig?'
}
{'Nennen Sie ein Beispiel für ein Hypothese, die einen Gerätevorteil beschreibt'

}
{'Skizzieren Sie ein Experiment Protokoll'

}
{'Was alles gehört zur Dokumentation eines durchgeführten Experiments?'

}
{'Warum archivieren Sie Messwerte und Messkörper?'

}
{'Wie lauten die Formen für Mittelwert und mitlaufenden Mittelwert. Wie berechnet man einen laufenden Mittelwert über eine begrenzte Anzahl von Stichprobenwerten?'
}
{'Welche Vorteile hat es, alle Messungen grundsätzlich mit einem Computer aufzuzeichnen?'
}
{'Wozu dient der f-Test?'

}
{'Was ist der t-Test?'

}
{'Welches sind die 5 Verfahren für systematische Belegung von Erkenntnissen?'

}
{'Welche drei Aussagen können bei Evidence-Based-Research getroffen werden?'

}
{'Welche Einrichtung überwacht in Deutschland die Wirksamkeit neuer Medizinprodukte? Welche Aufgabe hat die Behörde'
}
{'Warum müssen klinische Studien durchgeführt werden?'

}
{'Was ist ein Prüfplan'

}
{'Wie viele Phasen kennt eine klinische Studie ?'

}
{'Wie lautet die Norm zur Durchführung von klinischen Studien ?'

}
{'Was ist eine Norm und warum wurden diese entwickelt?'

}
{'Auf welche Art und Weise können Sie eine Idee der Allgemeinheit zugänglich machen und wie groß sind die Chancen, dass Ihre Idee die Welt verändern wird?'
}
{'Beschreiben Sie das V-Modell in Entwicklung und Dokumentation? '

```
                                                    }
    {'Welche Dokumente müssen Sie mindestens besitzen, wenn Sie ein neues Gerät als Medizin
produkt zulassen wollen?'
                                                    }
    {'Wozu dient eine Funktionsbeschreibung und wer soll diese in erster Linie lesen?'

                                                    }
    {'Wozu dient die Systembeschreibung und wer soll diese in erster Linie verstehen können
?'
                                                    }
    {'Was ist eine Bauteileliste?'

                                                    }
    {'Was ist eine Gebrauchsanweisung und welche 2 wichtigen Dinge werden durch sie definie
rt?'
                                                    }
    {'Was ist ein Montagehandbuch und welche 2 Dinge werden durch dieses definiert?'

                                                    }
    {'Was ist eine Risikoanalyse warum wird dies durchgeführt? Wie lautet die Formel für da
s Risiko'
                                                    }
    {'Was ist ein Verifikationsplan und was ist eine Verifikation?'

                                                    }
    {'Was ist eine Validierung und was ist ein Validierungsprotokoll?'

                                                    }
    {'Was bedeutet dauerhafte Produktüberwachung?'

                                                    }
    {'Welche Zertifikate müssen Sie unbedingt einholen, wenn Sie Medizinprodukte herstellen
?'
                                                    }
    {'Was ist die CE-Kennzeichnung und müssen Medizinprodukte diese haben'

                                                    }
    {'Beschreiben Sie in kurzen Worten, wie man ein Medizingerät zulässt. Geben Sie zeitlic
he Schritte an?'
                                                    }
    {'Was ist der Unterschied zwischen dem Zulassungsverfahren in Europa und den USA und we
lche Bedeutung haben klinische Tests dafür?'
                                                    }
    {'Was bedeutet Qualitätsmanagement?'

                                                    }
    {'Was ist die ISO 900o bzw. ISO 9001?'

                                                    }
    {'Was ist die ISO 13485 und was ist der Unterschied zur ISO 9001?'

                                                    }
    {'Welches sind die wichtigsten Dinge über die sie sich klar werden müssen?'

                                                    }
    {'Was ist eine technische Unbedenklichkeit und was müssen Sie dafür tun?'
```

}
{'Vor einem klinischen Einsatz müssen mehrere Dinge unbedingt vorliegen. Welches sind diese Dinge, wenn Sie keine klinische Studie planen sondern das Gerät evaluieren wollen?'
}
{'Welche Keramiken kommen zum Einsatz. nennen Sie 3 Keramiken für 3 unterschiedliche Einsatzgebiete?'
}
{'Welche Metalle kommen in der Instrumententechnik zum Einsatz. Nennen Sie 5 Metalle mit Eigenschaften (Werte, Einheit)?'
}
{'Nennen Sie 7 Gruppen von Kunststoffen, die in der Medizintechnik zum Einsatz kommen. Jeweils 2 Duroplaste, Thermoplaste, 2 Silikone. Geben sie jeweils 3 Eigenschaften mit Wert und Einheit an.' }
{'Was sind biokompatible Materialien?'

}
{'Was sind sterilisierbare Materialien?'

}
{'Welche Materialien sind implantierbar?'

}
{'Was sind bioabsorbierbare Materialien? Welche Einheiten und Werte kennzeichnen sie?'

}
{'Welche Materialien werden als Nahtmaterial eingesetzt?'

}
{'Was sind die Anforderungen an Gehäuse, die dauerhaft im Körper verbleiben?'

}
{'Nennen Sie 3 typische Materialien für Schläuche und Katheter, die im Körper bzw. für Bluttransport eingesetzt werden können (Name, Material, Anwendung, Eigenschaften)?'
}
{'Nennen Sie je 2 Kunststoffe mit Eigenschaften, Werten und Einheiten die für sterilisierbare und biokompatible Zwecke verwendet werden können.'
}
{'Welche Kunststoffe werden als Steriler Überzug eingesetzt und welche 3 Eigenschaften müssen sie erfüllen?'
}
{'Welche Materialien werden zur Züchtung von Zellen eingesetzt? Nennen Sie 3 Materialien für unterschiedliche Anwendungen?'
}
{'Welche Oberflächenbeschichtungsverfahren werden eingesetzt zur Erzielung welcher Eigenschaften? Nennen Sie 5 Eigenschaften mit Einheiten und Werten.'
}
{'Nennen Sie 3 Werkstoffe zur Herstellung autoklavierbarer Instrumente.'

}
{'Nennen Sie 5 Materialien, die in Zahnfüllen und als Zahnimplantate zum Einsatz kommen, einschließlich deren Materialeigenschaften (EWE).'
}
{'Aus welchem Material sind Trokkare (jeweils ein Beispiel für Metall und Kunststoff)?'

}
{'Aus welchem Material sind Stents (Material, Eigenschaft, Wert, Einheit)?'

}
{'Welche Elemente werden für Seeds verwendet und welche Halbwertszeiten haben diese im Vergleich zu Afterloading quellen?'
}
{'Nennen Sie 3 Materialien aus denen Grafts hergestellt werden mit ihren Eigenschaften (EWE)?'
}
{'Nennen Sie 3 Knochenersatzmaterialien (mit EWE). Je eines für die Anwendung als Spritzmaterial, Einlegeplatte und als absorbierbares Material?'
}
{'Nennen Sie 3 Knochenzementsorten mit ihre Eigenschaften, Werten und Einheiten.'

}
{'Welche Materialzertikate benötigen Instrumentenwerkstoffen, Hautkontaktwerkstoffe, Temporär im Körper Werkstoffe, bioabsorbierbare Werkstoffe, dauerhaft implantierbare Werkstoffe?'
}
{'Welche Magnete kommen in MRT Systemen sowie in den Sterotaxis-Systemen (NIOBE) vor? Welche Materialen und welche Feldstärken weisen diese Magnete auf? Welche spezifische Gewicht haben diese Magnete?'
}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

```
                                                                    }
        {'null'

                                                                    }
        {'null'

                                                                    }
        {'null'
```

```
}
{'null'

}
{'null'

}
{'null'
```

```
                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'Welches sind die typischen Antriebsformen von Bohrmaschinen im OP-Saal. Welche Versor
gung wird im OP dafür benötigt (Einheit und Wert)?'
                                                    }
    {'Welche Drehzahl und welches Drehmoment erreichen elektrische Fräser in der Chirurgie?
'
                                                    }
    {'Was ist der Unterschied zwischen einem Koagulator und einem HF-Messer und einem Shave
r? Welches dieser Instrumente hat eine Absaugung?'
                                                    }
    {'Was ist Lithotripsie und in welchen klinischen Anwendungen kommt es zum tragen? Wie f
unktioniert das Gerät? Mit welchen Frequenzen und welche Kräften arbeitet das Gerät? Wie we
rden die „Ergebnisse“ abtransportiert?'            }
    {'Wie funktioniert ein Wasserstrahlskalpell?'

                                                    }
    {'Wie funktioniert ein Laser zum Knochenabtrag?'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'

                                                    }
    {'null'
```

```
                                                    }
        {'null'

                                                    }
        {'null'

                                                    }
        {'null'
```

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'null'

}
{'Was ist eine Landmarkenkupplung? Welche zwei Komponenten werden dabei zusammengesteck
t?'

}
{'Geben Sie an, wie Sie aus einem Volumenbild I (x,y,z) eine orthogonale Schnittebene
formal angeben können? '

}
{'Geben Sie an, wie Sie aus einem Volumenbild I (x,y,z) eine orthogonale Schnittebene
formal angeben können? '

}
{'Welche Koordinatensysteme sind für die Kalibrierung von Instrumenten erforderlich? Ze
ichnen Sie eine Skizze mit allen Koordinatensystemen'

}
{'Welche Komponenten sind zur Registrierung starrer Instrumente über einen Punkt erford
erlich? Was ist das Ergebnis der Registrierung eines starren Instruments. Wie lautet die Ma
trix?'

}
{'Welche Komponenten sind zur Kalibration der Pose starrer Instrumente erforderlich? W
as ist das Ergebnis der Registrierung eines starren Instruments. Wie lautet die Matrix?'

}
{'Welche Komponenten sind zur Registrierung starrer Werkzeugspitzen oder Werkzeuglängen

über einen Punkt erforderlich? Was ist das Ergebnis der Registrierung eines Bohrers? Wie wird die Matrix berechnet?' }
{'Welche zwei Möglichkeiten bestehen zur Realisierung einer Instrumententracker-Kupplung? Welche ist patentiert?'
}
{'Welches Risiko und welche Gefahr besteht grundsätzlich bei der Registrierung von Instrumenten und nach der Registrierung des Instruments bei der Behandlung?'
}
{'Welches Risiko und welche Gefahr besteht grundsätzlich bei der Registrierung und nach der Registierung von Patient und Bild?'
}
{'Wie viele Menschen leben gegenwärtig in Deutschland? Wie viele sind davon a)Kinder und Jugendliche, b) aktuell erwerbstätige, c) Rentner, d) erwerbslose oder erwerbsunfähige oder eingeschränkt erwerbsfähig?' }
{'Was bedeutet Lab-on-a-Chip? Was wird mit diesen Systemen gemessen?'

}
{'Was ist ein Drug-Delivery-System?'

}
{'Welches Verhalten von Menschen hält den Menschen lange körperlich und geistig gesund?'
}
{'Was bedeuten die Begriffe Diagnose, Anamnese, Befundung und von wem werden diese Tätigkeiten ausgeführt?'
}
{'Was macht eine künstliche Niere? Wie groß ist dieses Gerät? Wieviele Menschen sind davon betroffen? Was passiert wenn keine Niere gespendet wird und das Gerät defekt ist? Seit wann kann man Nieren spenden? Wachsen Nieren nach?'}
{'Welche Aufgaben hat das Gebiet des Clinical Engineering in der Medizintechnik?'

}
{'Nennen Sie bildgebende Systeme in der Medizin und gegen Sie an, wieviele Dimensionen die Bilder haben?'
}
{'Wie kommunizieren bisher Geräte in der Medizintechnik? '

}
{'Nennen Sie die häufigsten Datenaustauschformate in Kliniken? Welche Geschwindigkeiten bietet Ethernet?'
}
{'Was ist ein Datenlogger?'

}
{'Was ist RFID Technologie?'

}
{'Womit beschäftigt sich das Tissue Engineering?'

}
{'In welches Gebiet gehört die Strahlentherapie?'

}
{'Welches sind die 13 Inhalte der Vorlesung?'

}
{'Warum muss der Auftraggeber einer klinischen Studie bekannt sein?'

```
                                                    }
    {'Geben sie 4 Standardwerke der Medizintechnik an.'

                                                    }
    {'Wo werden in der Medizintechnik spezielle Textilien eingesetzt?'

                                                    }
    {'Geben Sie eine formale Schreibweise für ein Grauwertbild mit 256 Intensitätsstufen so
wie n=768 Zeilen und m=1024 Spalten an: I (r,s)=.......'
                                                    }
    {'Geben Sie eine formale Schreibweise für ein Voxelbild mit nx=512, ny=512, nz=200 Voxe
ln an, bei einer Voxelgröße von bx=by=bz= 1 mm und 4096 Graustufen. Die Matrix hat die Indi
zes (u,v,w) und beginnt bei (0,0,0)'                }
    {'Was ist eine automatische Registrierung und was wird dabei registriert?'

                                                    }
    {'Was ist eine automatische Registrierung und was wird dabei berechnet?'

                                                    }
    {'null'

                                                    }
    {'Welche Formen der Bildgebung kennen Sie?'

                                                    }
    {'null'

                                                    }
```

Closing the Database Connection

```
FMclose(conn);
```

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:52:45!
Executed 16-Sep-2018 16:52:47 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
```

```
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 33: Using a Round-Robin realtime multi-tasking system

2017-03-05: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-07

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.5 required)
- List of supported functions
- Intended use of RRrun or RRshell (both are the same)
- 1. Starting the shell
- 2. Adding realtime tasks and define the stop time
- 3. Adding realtime tasks and change the cycle time
- 4. Adding realtime tasks and change the cycle time and kill the task
- 5. Run a plotting task and save the task list by using "save"
- 6. Run the saved task a second time by using "load"
- 7. Execute a command file
- 8. Edit a command file
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids

- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Collection of Ideas for Tutorials
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

Motivation for this tutorial: (Originally SolidGeometry 3.5 required)

Matlab can be converted relatively simply to a realtime-multi-tasking environment according to the round-robin method. The following conditions must apply: A) There is a fixed time base for all tasks B) All tasks are called up one after the other in the fixed time clock

The environment presented in this tutorial has the following properties:

- A) Shell character = While the real-time environment is running, Matlab commands can still be typed as before.
- B) All variables of the real-time environment can be modified directly.

List of supported functions

The input-interpreter routine of the real-time environment has additional commands that superimpose comparable commands on the Matlab command line.

- ***HELP*** - shows a help text
- ***EXIT*** or ***QUIT*** - ends the environment
- ***SHOWLIST*** or ***TASKS*** - shows all tasks
- ***KILLTASKS*** or ***KILLALL*** - removes all tasks
- ***ADD*** - appends a task at the task list
- ***KILLLAST*** - removes the last task
- ***BREAK*** or ***STOP*** - stops the realtime execution
- ***STEP*** - runs the realtime loop only ones

- ***GO*** or ***START*** or ***CONT*** - starts the realtime loop
- ***KILL*** Tasknumber - removes a task with that number
- ***SAVE*** - saves the current task list on disk
- ***LOAD*** - loads the current task list on disk
- ***EXE*** or ***EXECUTE*** filename - executes a command line file
- ***edit*** filename edits a command line file
- ***whos*** shows the variables

Intended use of RRrun or RRshell (both are the same)

- ***RRrun*** - starts the Shell
- ***RRrun*** commandstring (lines are spearated by \r)

1. Starting the shell

The following commands could be typed in absolute in the same way as part of the command string. So please try to type them also manually instead of using them als input parameter of RRrun There is no other chance to create a publishable document as to describe them as input parameter.

```
RRrun 'quit'    % Quit immediately
```

```
RRkeyboardLine =

    'quit'

====LOOP STARTS 16-Sep-2018 16:52:48 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
TERMINATED by User
================LOOP ENDS 16-Sep-2018 16:52:48 USING 0.11 SECONDS =========================
```

```
RRrun 'RRstop=RRcputime+1;'  % Quit after 1 second
```

```
RRkeyboardLine =

    'RRstop=RRcputime+1;'

====LOOP STARTS 16-Sep-2018 16:52:48 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
================LOOP ENDS 16-Sep-2018 16:52:49 USING 1.20 SECONDS =========================
```

```
RRrun 'LIST \r QUIT' % show the tasks and quit
```

```
RRkeyboardLine =
```

```
    'LIST       QUIT'

====LOOP STARTS 16-Sep-2018 16:52:50 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
RRtasklist =

  struct with fields:

       t0: 0.1000
    twarn: 0.1000
    tstop: 1
      cnt: 0
    tlist: []


TERMINATED by User
================LOOP ENDS 16-Sep-2018 16:52:50 USING 0.20 SECONDS =========================
```

```
RRrun 'whos \r QUIT' % show the variables and quit
```

```
RRkeyboardLine =

    'whos       QUIT'

====LOOP STARTS 16-Sep-2018 16:52:50 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>  Name                Size            Bytes  Class               Attributes

  RRbreak             1x1                 1  logical
  RRcurs              0x0                 0  double              persistent
  RRdelay             1x1                 8  double              global
  RRkeyboardLine      1x6                12  char                global
  RRlastcommand       1x4                 8  char                global
  RRlastexectime      1x1                 8  double
  RRlasttime          1x1                 8  double              global
  RRmaxtime           1x1                 8  double              global
  RRpause             1x1                 8  double
  RRprompt            1x5                10  char                global
  RRstart             1x1                 8  double              global
  RRstop              1x1                 8  double              global
  RRtasklist          1x1               912  struct              global
  RRwindow            1x1                 8  matlab.ui.Figure    global
  RRwindowNr          1x1                 8  double              global
  cmd                 1x4                 8  char
  remain              0x0                 0  char
  token               1x4                 8  char
  varargin            1x1               136  cell



TERMINATED by User
================LOOP ENDS 16-Sep-2018 16:52:50 USING 0.20 SECONDS =========================
```

```
RRrun ('fprintf(''%.2f\n'',RRcputime) \rQUIT') % show the cputime and quit
```

```
RRkeyboardLine =

    'fprintf('%.2f\n',RRcputime)      QUIT'

====LOOP STARTS 16-Sep-2018 16:52:50 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>2.25


TERMINATED by User
================LOOP ENDS 16-Sep-2018 16:52:50 USING 0.20 SECONDS =========================
```

2. Adding realtime tasks and define the stop time

```
RRrun 'ADD fprintf(''%.2f\n'',RRcputime) \r RRstop=RRcputime+1;' % show the cputime every c
ycle and quit after 1 second
```

```
RRkeyboardLine =

    'ADD fprintf('%.2f\n',RRcputime)       RRstop=RRcputime+1;'

====LOOP STARTS 16-Sep-2018 16:52:50 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>2.64

2.74
2.84
2.94
3.04
3.14
3.24
3.34
3.44
3.54
3.64
3.74
================LOOP ENDS 16-Sep-2018 16:52:52 USING 1.30 SECONDS =========================
```

3. Adding realtime tasks and change the cycle time

```
RRrun 'ADD fprintf(''%.2f\n'',RRcputime) \r RRtasklist.t0=0.05 \r RRstop=RRcputime+1;'
```

```
RRkeyboardLine =

    'ADD fprintf('%.2f\n',RRcputime)       RRtasklist.t0=0.05       RRstop=RRcputime+1;'
```

```
====LOOP STARTS 16-Sep-2018 16:52:52 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>4.04

RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {' fprintf('%.2f\n',RRcputime)'  [Inf]  [1]}


4.09

4.14
4.19
4.24
4.29
4.34
4.39
4.44
4.49
4.54
4.59
4.64
4.69
4.74
4.79
4.84
4.89
4.94
4.99
5.04
5.09
5.14
================LOOP ENDS 16-Sep-2018 16:52:53 USING 1.30 SECONDS =========================
```

4. Adding realtime tasks and change the cycle time and kill the task

```
RRrun 'ADD fprintf(''%.2f\n'',RRcputime) \r RRtasklist.t0=0.05 \r KILLLAST \r RRstop=RRcput
ime+1;'
```

```
RRkeyboardLine =

    'ADD fprintf('%.2f\n',RRcputime)       RRtasklist.t0=0.05       KILLLAST       RRstop=R
Rcputime+1;'

====LOOP STARTS 16-Sep-2018 16:52:53 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>5.44
```

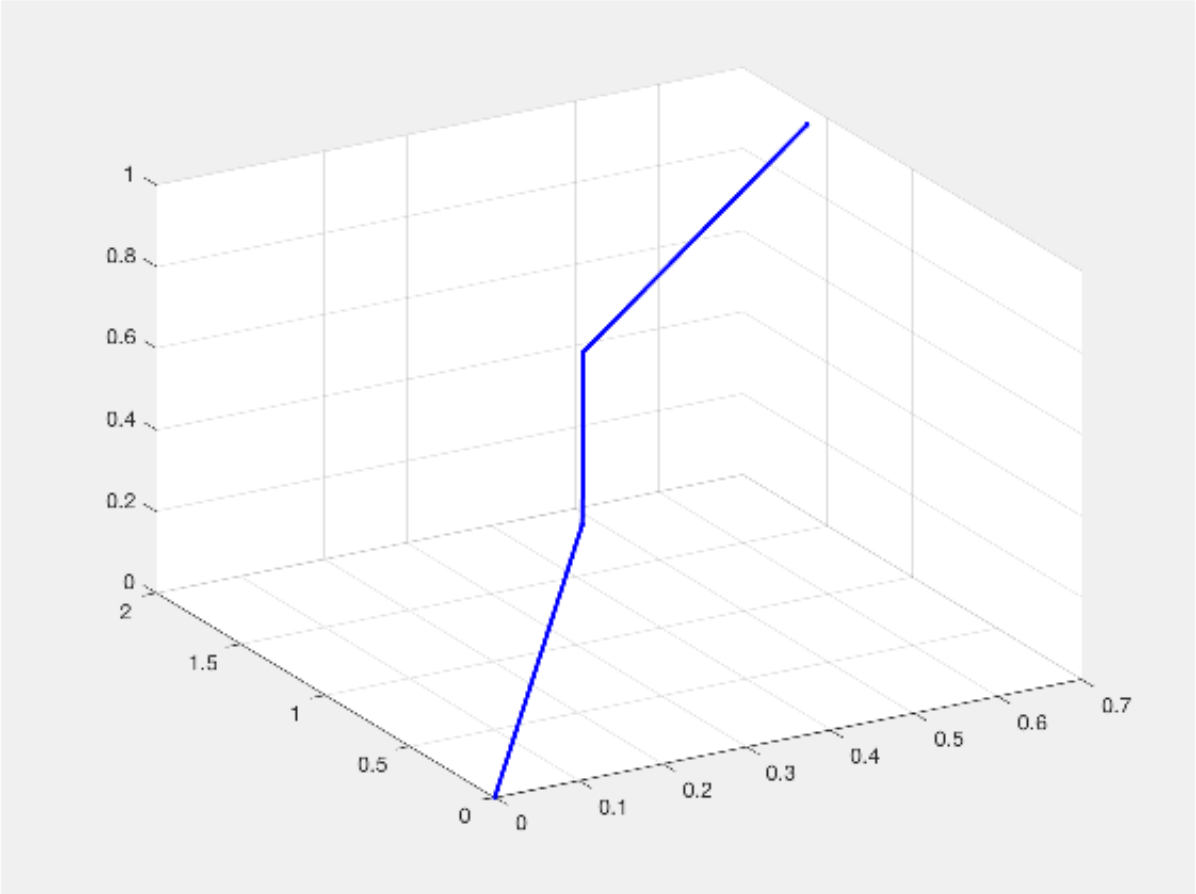
```
RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {' fprintf('%.2f\n',RRcputime)'  [Inf]  [1]}


5.49

RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {0×3 cell}


================LOOP ENDS 16-Sep-2018 16:52:55 USING 1.35 SECONDS =========================
```

5. Run a plotting task and save the task list by using "save"

```
RRrun 'KILLALL \r global PL; PL=[0 0 0]; \r  ADD global PL; PL=[PL; PL(end,:)+rand(1,3)]; \
r ADD global PL; delete(gca); VLplot(PL,''b.-'',2); view(-30,30); grid on; \r save \r copyp
lot \r RRstop=RRcputime+3;'
```

```
RRkeyboardLine =

    'KILLALL       global PL; PL=[0 0 0];        ADD global PL; PL=[PL; PL(end,:)+rand(1,3)
];       ADD global PL; delete(gca); VLplot(PL,'b.-',2); view(-30,30); grid on;       save
      copyplot       RRstop=RRcputime+3;'

====LOOP STARTS 16-Sep-2018 16:52:55 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
RRtasklist =

  struct with fields:

       t0: 0.1000
    twarn: 0.1000
    tstop: 1
      cnt: 0
    tlist: []
```

```
RRtasklist saved in RRtasklist.mat
  Name            Size            Bytes  Class     Attributes

  RRtasklist      1x1              1832  struct    global



RRrun: realtime condition broken by 308 milliseconds

================LOOP ENDS 16-Sep-2018 16:52:59 USING 4.14 SECONDS ========================
```

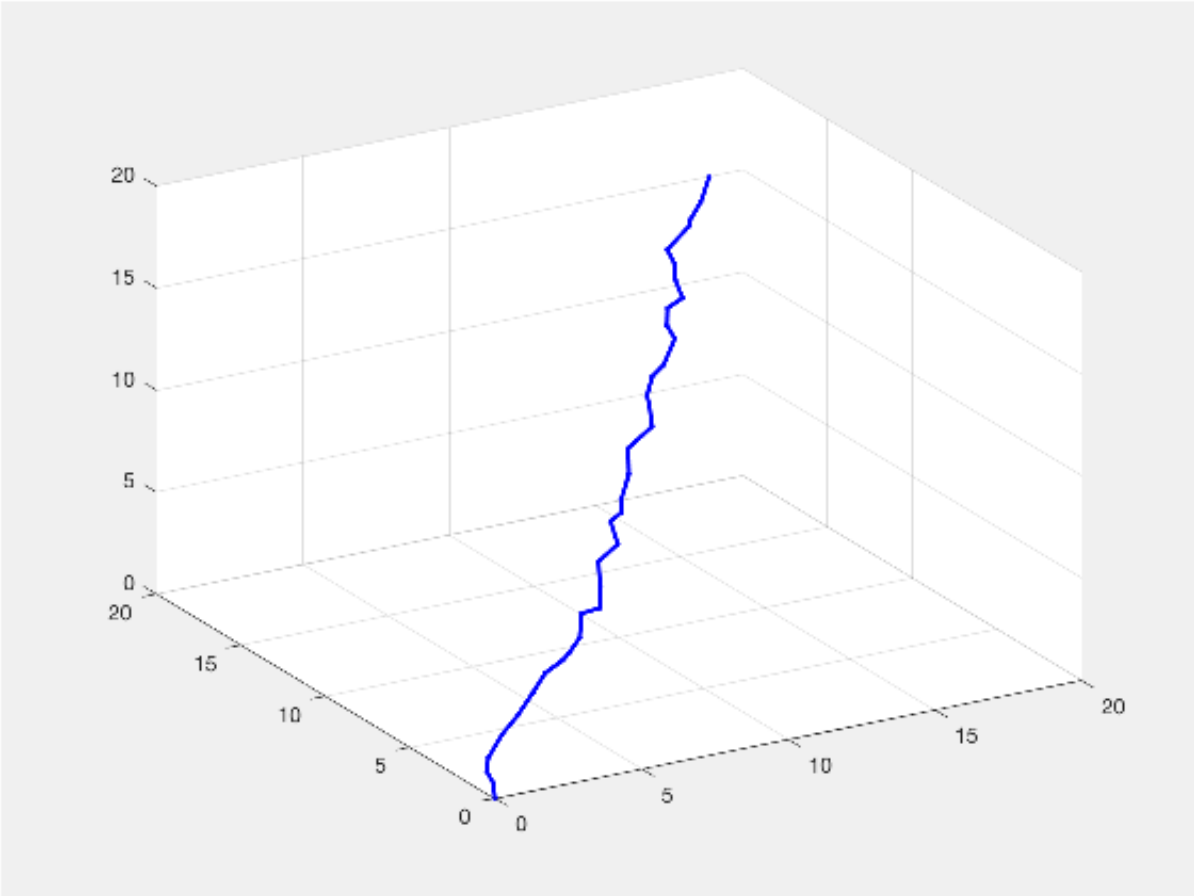


6. Run the saved task a second time by using "load"

```
RRrun 'load \r start \r copyplot \r RRstop=RRcputime+3;'
```

```
RRkeyboardLine =

    'load       start       copyplot       RRstop=RRcputime+3;'

====LOOP STARTS 16-Sep-2018 16:52:59 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
New RRtasklist loaded from RRtasklist.mat
```

```
RRrun: realtime condition broken by 119 milliseconds

================LOOP ENDS 16-Sep-2018 16:53:03 USING 3.65 SECONDS ========================
```

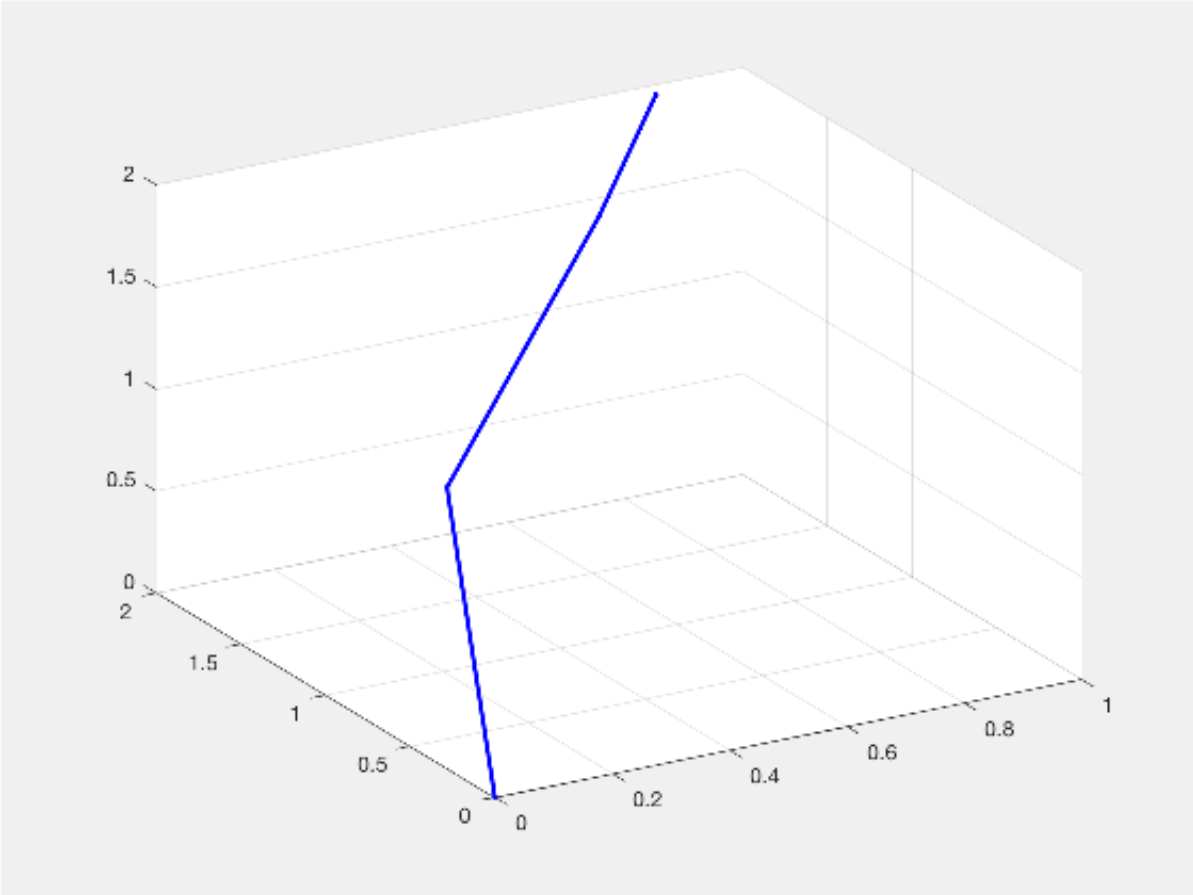


7. Execute a command file

```
RRrun 'execute RRrun_testcommands.txt'
```

```
RRkeyboardLine =

    'execute RRrun_testcommands.txt'

====LOOP STARTS 16-Sep-2018 16:53:03 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========
=>> END =======
RRrun>>
fname =

    'RRrun_testcommands.txt'

Warning: Inputs must be character vectors, cell arrays of character vectors, or
string arrays.
```

```
RRtasklist =

  struct with fields:

       t0: 0.1000
    twarn: 0.1000
    tstop: 1
      cnt: 0
    tlist: []



RRtasklist =

  struct with fields:

       t0: 0.1000
    twarn: 0.1000
    tstop: 1
      cnt: 4
    tlist: {2×3 cell}


ans =

  2×3 cell array

    {' global PL; PL=[…'}    {[Inf]}    {[1]}
    {' global PL; dele…'}    {[Inf]}    {[3]}

RRtasklist currently stopped. Use "START" or "STEP" to start task execution.

Elapsed time is 0.008233 seconds.



RRrun: realtime condition broken by 108 milliseconds

RRrun>>================LOOP ENDS 16-Sep-2018 16:53:07 USING 3.94 SECONDS ==================
=======
```



8. Edit a command file

```
edit 'RRrun_testcommands.txt'
```

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:53:08!
Executed 16-Sep-2018 16:53:10 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
```

```
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction

2017-05-15: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 3.8 required)
- 1. Create a number of random points around the center
- 2. Create an X-ray image by using the camera parameter of Matlab
- 3. Find the marker points in the image
- turn the coordinate
- Some knowledge on corresponding axis
- 3. Calculate the Point Position of a X-Ray Camera
- 5. Comparision of point lists created by numerical projection or projection image reconstruction
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 3.8 required)

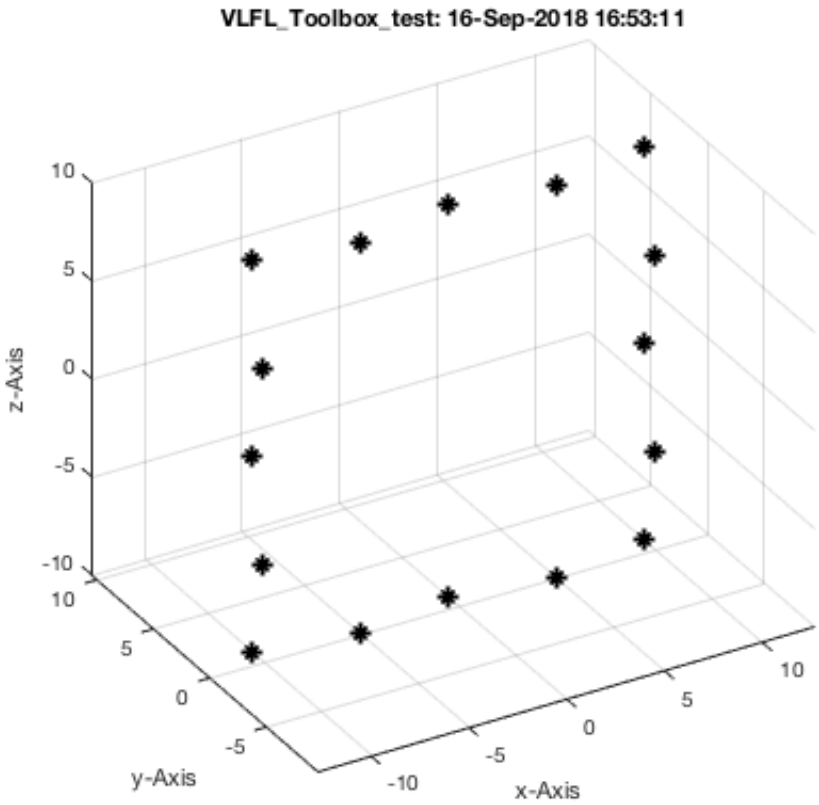
Many surgical procedures in orthopedics are not based on three-dimensional CT or MRI image data but on C-arm images. These C-arm images are 2D projection images of a spatial region of the patient. In this, the most important strategies for the conversion of volume images to projection images are presented. It is also explained how the position of the X-ray camera can be calculated from projection images, if one knows the exact location of objects in space and the 2D image. The research area is also called Camera Calibration.

ATTENTION >>> The Publishermode changes the aspect ratio of figures, therefor it is strongly recommended to copy lines from this tutorial instead of just executing the publishabe example

1. Create a number of random points around the center

The following commands could be typed in absolute in the same way as part

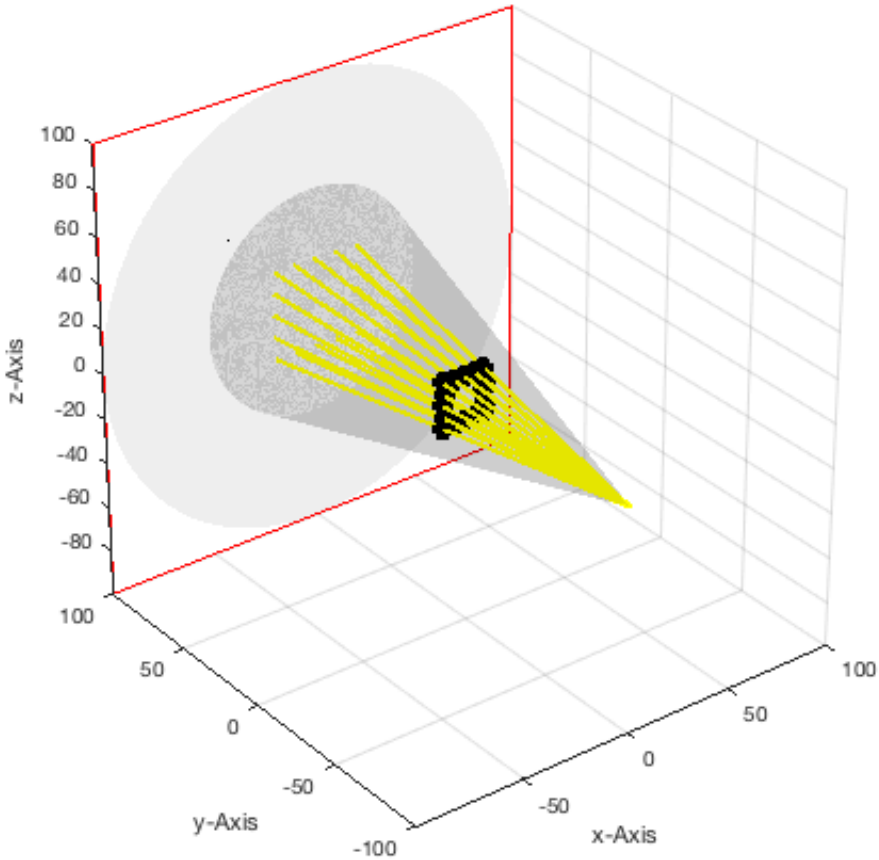
```
SGfigure; view(-30,30); xlabel 'x-Axis', ylabel 'y-Axis', zlabel 'z-Axis';
VL=50*rand(10,3)-25; VL(:,2)=1*rand(10,1)';
% VL=VLsample(9)
VL=VLsample(11); VL=VLtransT(VL,TofR(rot(-pi/20,0,pi/20)));
VL=VLsample(12);
VLplot(VL,'k*');
```



2. Create an X-ray image by using the camera parameter of Matlab

The x-ray source is at position [0 100 0]; The target is at [0 +100 0] The screen has a size of 100x100 The scaling factor is 4, i.e. the pixel size if 025 x 0.25 mm

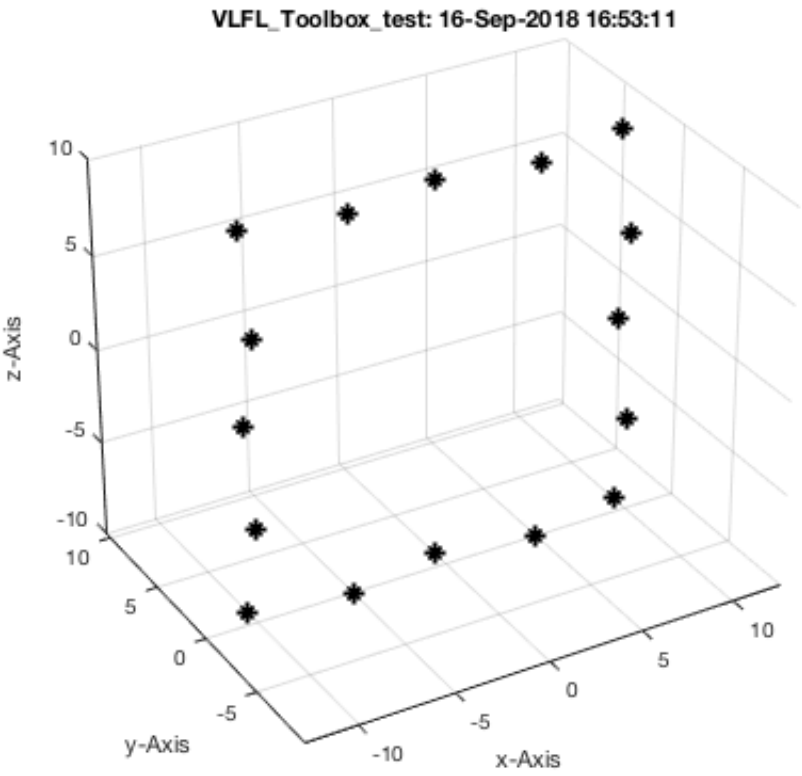
```
imageofVLprojection(VL,[100 100],[0 -100 0],[0 100 0],4);
set(gca,'Projection','perspective');    % this line is only required because of publishing function
```



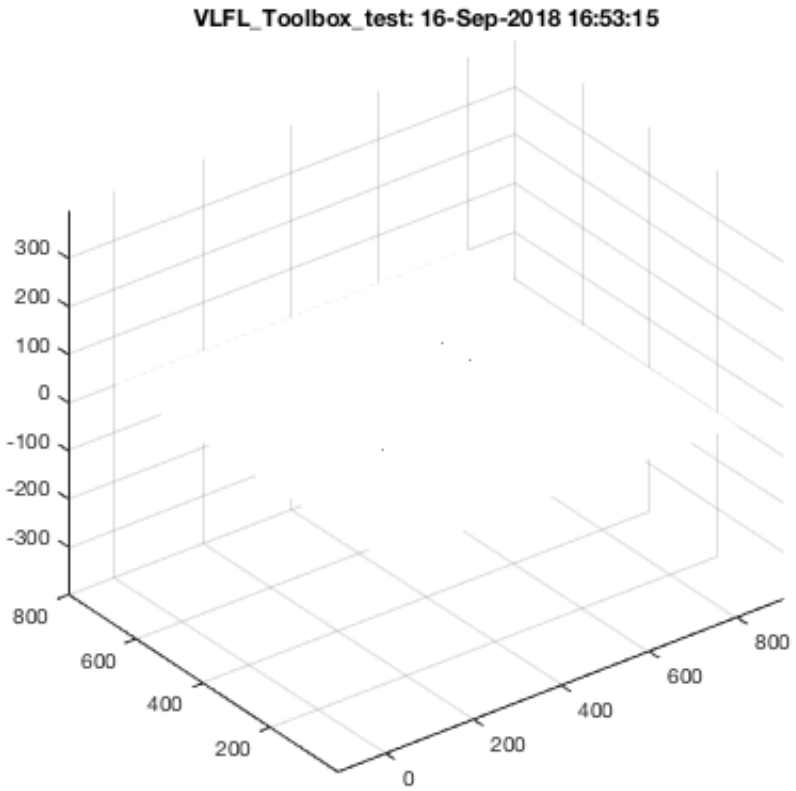
show the image

```
I=imageofVLprojection(VL,[100 100],[0 -100 0],[0 100 0],4);
set(gca,'Projection','perspective');    % this line is only required because of publishing
function
whos I                                  % this line is only required because of publishing
function
```

```
  Name        Size                Bytes  Class     Attributes

  I         800x800             5120000  double
```




```
SGfigure
imwarpT(I);
```

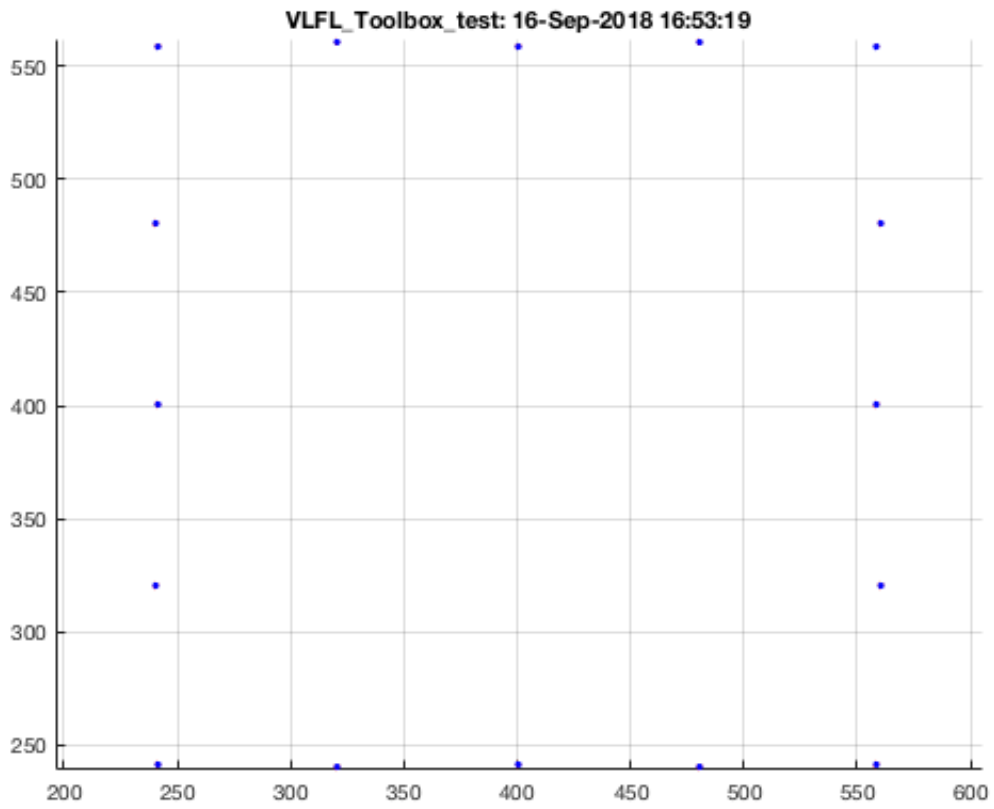


3. Find the marker points in the image

```
SGfigure;
CPL=CPLcontourc(I,1); % Contour segmentation on image base
CPLplot(CPL,'r-');
PL=centerCPL(CPL)
PLplot(PL,'b.',4);
size(I,2)                                % this line is only required because of publishing
function
```

```
PL =

  240.5000  480.5000
  240.5000  320.5000
  560.5000  480.5000
  560.5000  320.5000
  241.5000  558.5000
  241.5000  400.5000
  241.5000  241.5000
  558.5000  558.5000
  558.5000  400.5000
  558.5000  241.5000
  320.5000  560.5000
  320.5000  240.5000
  480.5000  560.5000
  480.5000  240.5000
  400.5000  558.5000
```

```
  400.5000  241.5000


ans =

   800
```

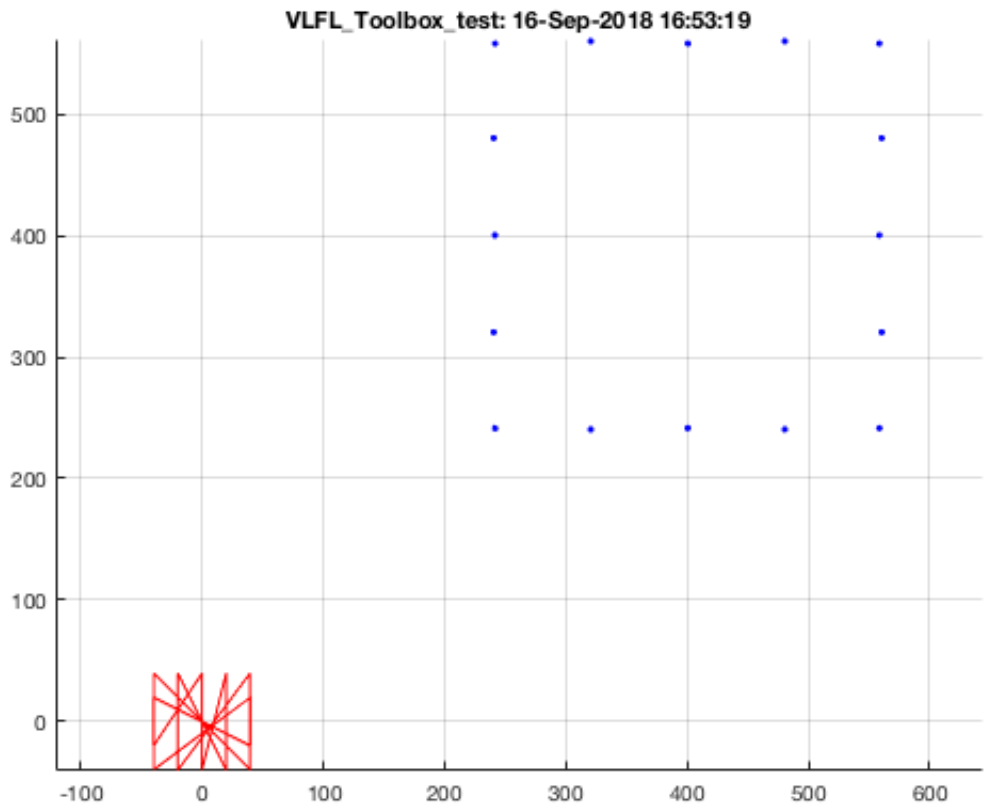


turn the coordinate

```
PL(:,2)=-PL(:,2)+size(I,2)    % flip up and down (y-axis)
PL=(PL-1)-size(I)/2           % Move coordinate into center
PL=PL/4                       % Scale using pixle size
CPLplot(PL,'r-');
```

```
PL =

  240.5000  319.5000
  240.5000  479.5000
  560.5000  319.5000
  560.5000  479.5000
  241.5000  241.5000
  241.5000  399.5000
  241.5000  558.5000
  558.5000  241.5000
  558.5000  399.5000
  558.5000  558.5000
```

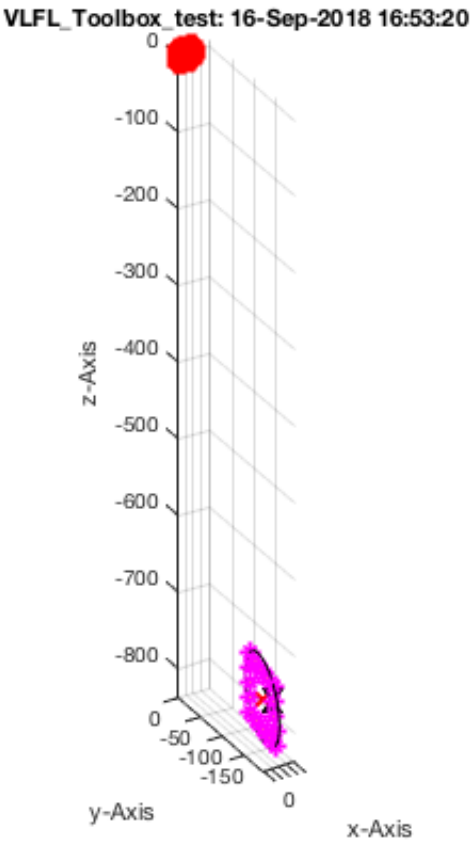
```
  320.5000  239.5000
  320.5000  559.5000
  480.5000  239.5000
  480.5000  559.5000
  400.5000  241.5000
  400.5000  558.5000


PL =

 -160.5000  -81.5000
 -160.5000   78.5000
  159.5000  -81.5000
  159.5000   78.5000
 -159.5000 -159.5000
 -159.5000   -1.5000
 -159.5000  157.5000
  157.5000 -159.5000
  157.5000   -1.5000
  157.5000  157.5000
  -80.5000 -161.5000
  -80.5000  158.5000
   79.5000 -161.5000
   79.5000  158.5000
   -0.5000 -159.5000
   -0.5000  157.5000


PL =

  -40.1250  -20.3750
  -40.1250   19.6250
   39.8750  -20.3750
   39.8750   19.6250
  -39.8750  -39.8750
  -39.8750   -0.3750
  -39.8750   39.3750
   39.3750  -39.8750
   39.3750   -0.3750
   39.3750   39.3750
  -20.1250  -40.3750
  -20.1250   39.6250
   19.8750  -40.3750
   19.8750   39.6250
   -0.1250  -39.8750
   -0.1250   39.3750
```



Some knowledge on corresponding axis

```
TofcamVLPL(sortrows(VL,[1 3]),sortrows(PL,[1 2]))
```

```
K =

   1.0e+03 *

    0.0008   -2.2940    0.4000
    0.0000    3.1786   -0.5534
    0.0000    0.0000    0.0018


s =

    0.5540


ans =

    0.0276   -0.1712   -0.9848   21.5558
    0.0003   -0.9852    0.1713 -135.1995
   -0.9996   -0.0050   -0.0271 -780.8350
         0         0         0    1.0000
```



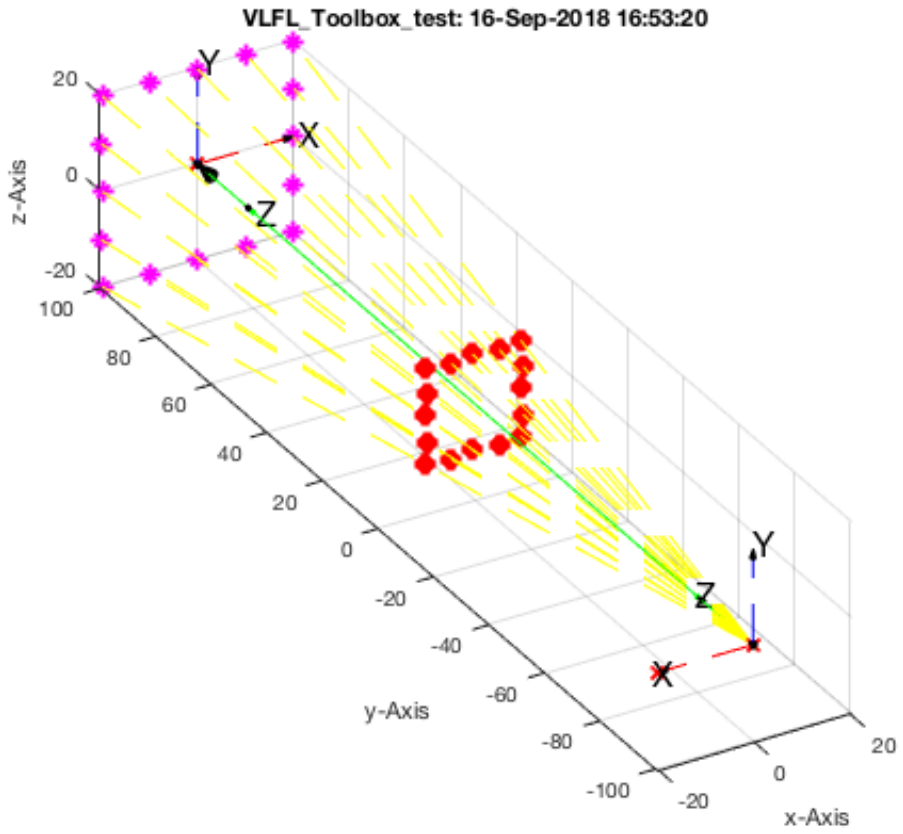
3. Calculate the Point Position of a X-Ray Camera

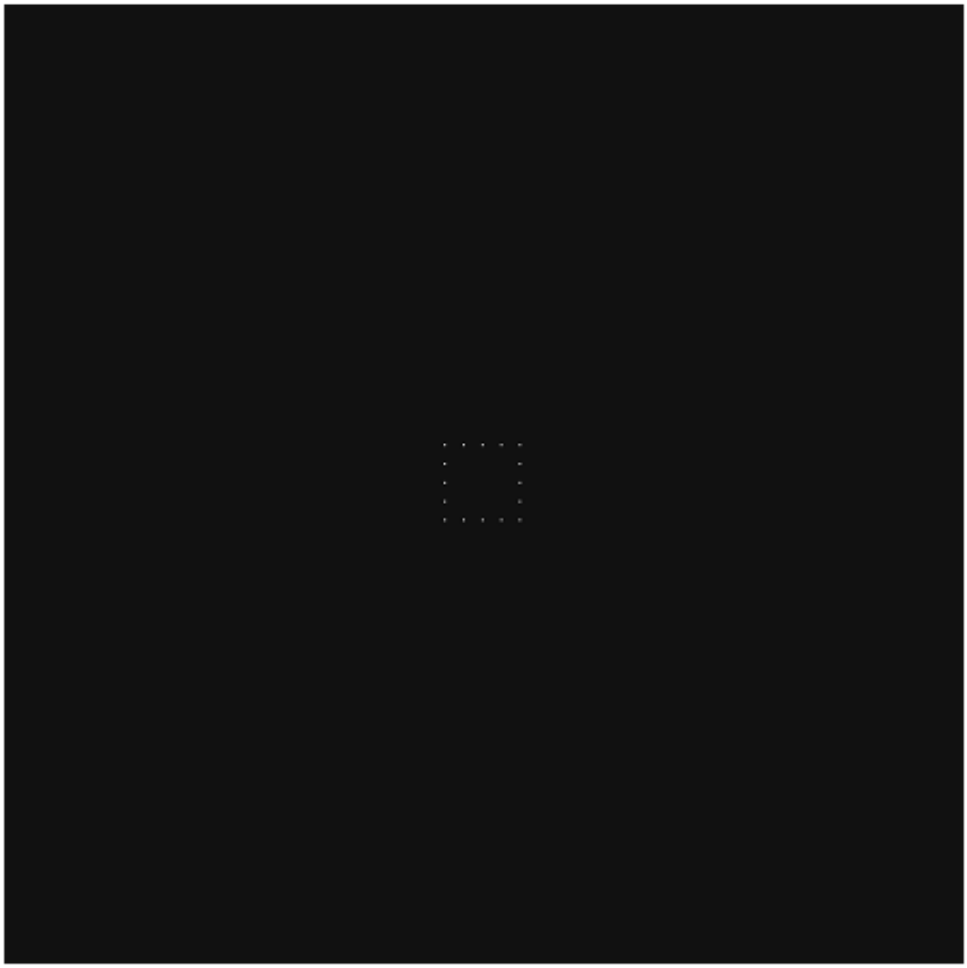
The x-ray source is at position [0 100 0]; The target is at [0 +100 0]

```
PLofVLprojection(VL,[0 -100 0],[0 100 0]);
PL=PLofVLprojection(VL,[0 -100 0],[0 100 0])
```

```
  Name        Size                Bytes  Class     Attributes

  I         512x512             2097152  double

  Name        Size                Bytes  Class     Attributes

  I         512x512             2097152  double


PL =

  -19.8020  -19.8020
  -10.0000  -20.0000
         0  -19.8020
   10.0000  -20.0000
   19.8020  -19.8020
   20.0000  -10.0000
   19.8020         0
   20.0000   10.0000
   19.8020   19.8020
```

```
   10.0000   20.0000
         0   19.8020
  -10.0000   20.0000
  -19.8020   19.8020
  -20.0000   10.0000
  -19.8020         0
  -20.0000  -10.0000
```



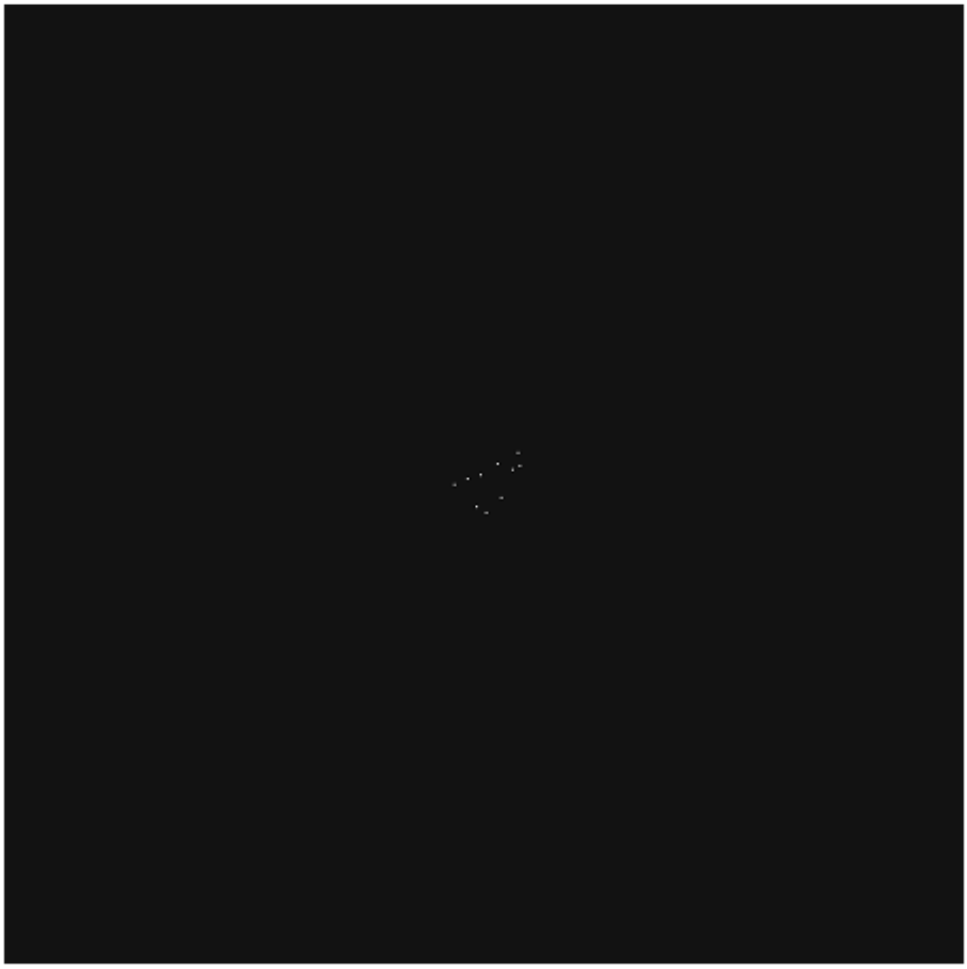


5. Comparision of point lists created by numerical projection or projection image reconstruction

```
VL=20*rand(10,3)-10; VL(:,2)=5*rand(10,1)';
I=imageofVLprojection(VL,[100 100],[0 -100 0],[0 100 0],4);
[sortrows(PLofimcontourc(I,true,1/4)) sortrows(PLofVLprojection(VL,[0 -100 0],[0 100 0]))]
```

```
  Name        Size                Bytes  Class     Attributes

  I         512x512             2097152  double


ans =

  -30.3750   -2.3750  -15.1131   -1.2480
  -16.8750    4.8750   -8.3027    2.4210
   -7.1250  -25.1250   -3.4168  -12.6239
   -2.8750    7.3750   -1.3460    3.7215
    2.8750  -31.6250    1.5936  -15.7831
   16.1250   19.3750    8.1775    9.7042
   20.6250  -15.8750   10.4555   -7.9246
   31.3750   13.8750   15.8333    6.9689
   37.1250   31.6250   18.6999   15.7792
```

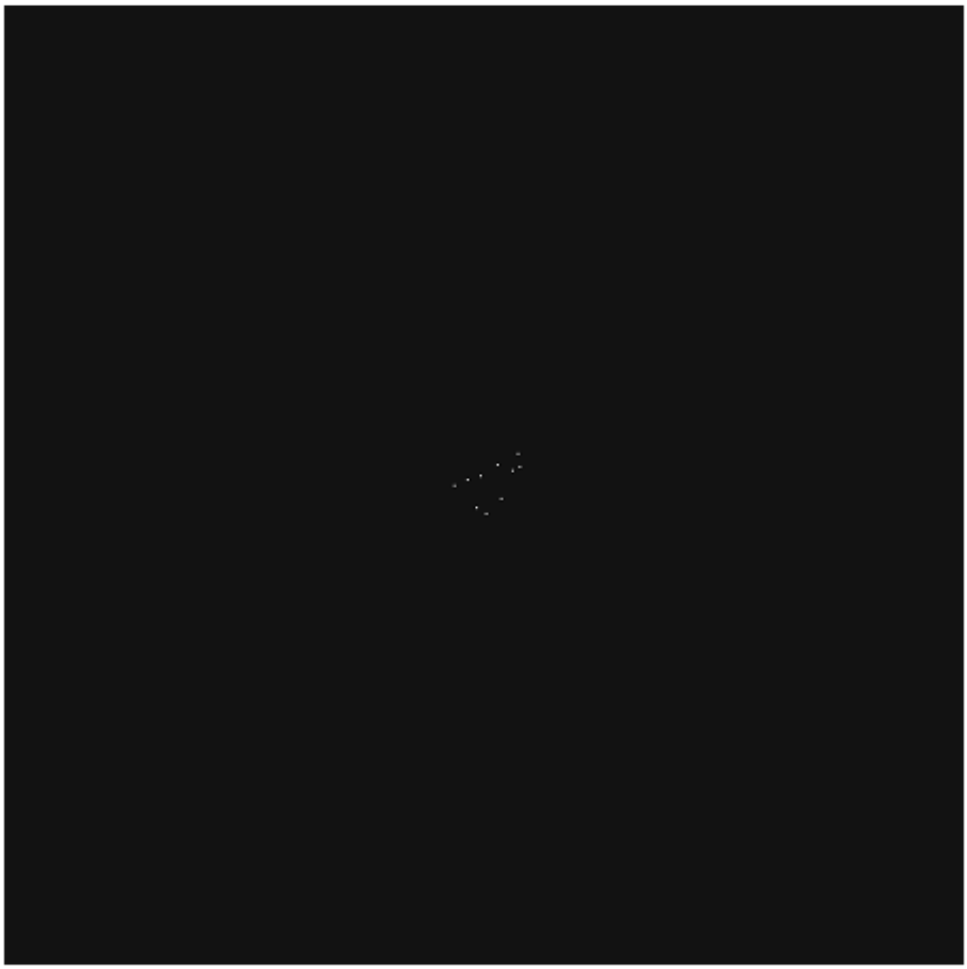
```
   39.6250   18.8750   19.9261    9.4254
```

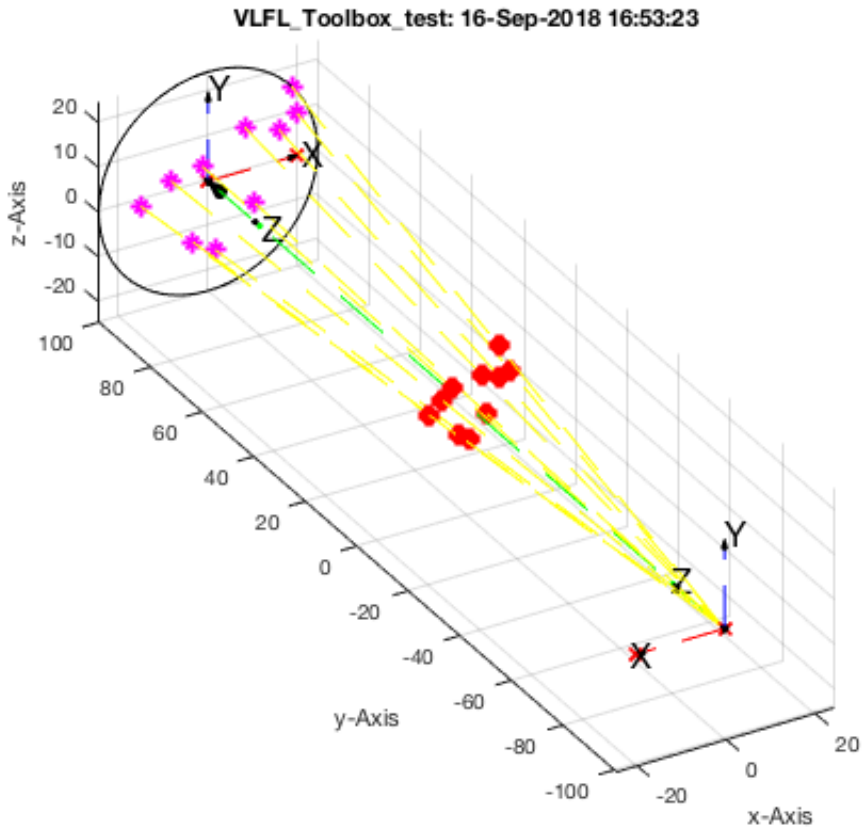


```
TofcamVLPL(sortrows(VL,[1 3]),sortrows(PLofVLprojection(VL,[0 -100 0],[0 100 0]),[1 2]))
```

```
  Name        Size                Bytes  Class     Attributes

  I         512x512             2097152  double


K =

    1.0000   -0.0000    0.0000
         0    1.0000    0.0000
   -0.0000   -0.0000    0.0050


s =

   200
```

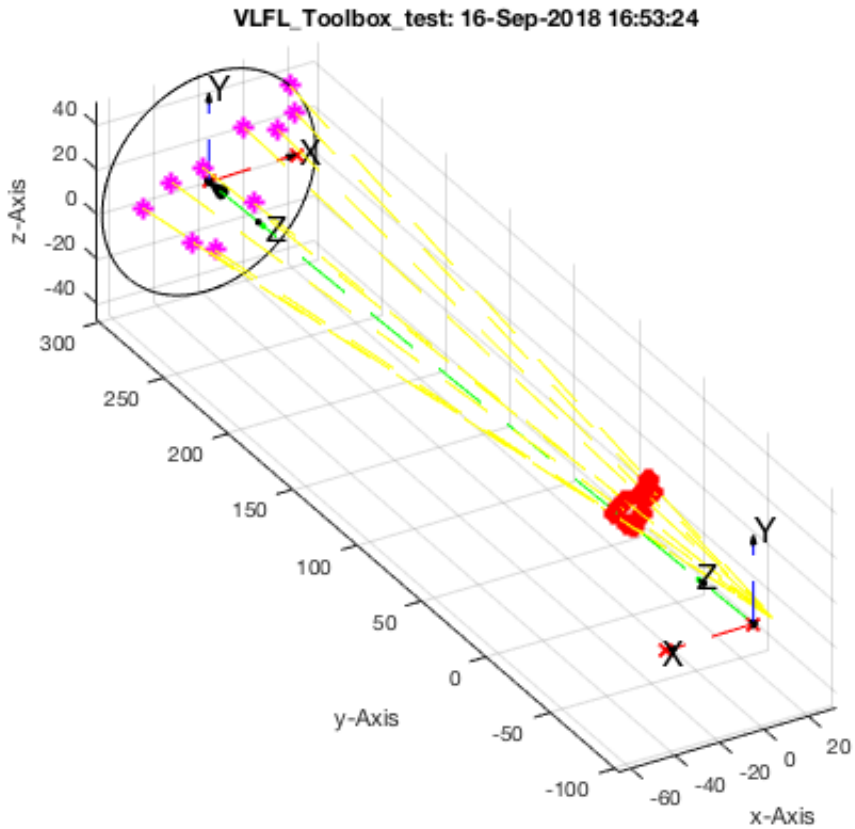
```
ans =

   -1.0000         0         0    0.0000
         0         0    1.0000 -100.0000
         0    1.0000         0    0.0000
         0         0         0    1.0000
```






```
TofcamVLPL(sortrows(VL,[1 3]),sortrows(PLofimcontourc(I,true,1/4),[1 2]))
```

```
K =

    0.9973   -0.0016   -0.0317
   -0.0000    1.0000    0.0021
    0.0000    0.0000    0.0025


s =

  400.0973


ans =

   -0.9995   -0.0020   -0.0330   -3.3114
   -0.0329   -0.0047    0.9994 -100.0093
   -0.0021    1.0000    0.0046   -0.1599
         0         0         0    1.0000
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:53:24!
Executed 16-Sep-2018 16:53:26 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 35: Creation of Kinematic Chains and Robot Structures

2017-07-04: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.0 required)
- 1. Loading STL Files or Surface Data
- 2. Attaching Frames to a Surface Model
- 3. Spatial Arrangment of Solids relative to Frames
- 4. Simple Sequential Kinematic Chains
- 5. Calibration of a Sequential Kinematic Chain
- 6. Creating Kinematic Trees
- 7. Calculating Boxes for Quick Collision Checks
- 8. Collision Check
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines

- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

Already in the tutorials 11 and 12 kinematic chains were presented. This tutorial is about creating tree-like structures for robotic systems. The example uses the structures of the robot JACO. function VLFL_EXP35

1. Loading STL Files or Surface Data

The Elements of the JACO were prepared by reading STL data in and save the variables using the save command. Now the surface data is available but also those surfaces have already defined frames "B" for base and "F" for follower. clear all

```
loadweb JACO_robot.mat
whos
```

```
Downloading "http://www.mimed.mw.tum.de/fileadmin/w00bhh/www/Matlab_Toolboxes/JACO_robot.ma
t" into: /Users/timlueth/Desktop/Toolbox_test/2018-09-16_TL_PCODE!
  Name          Size                    Bytes  Class                               Attribute
s

  A             1x1                      2292  struct

  A0            1x2                        16  double

  A1            1x2                        16  double

  B             1x1                     88500  struct

  B0            1x2                        16  double
```

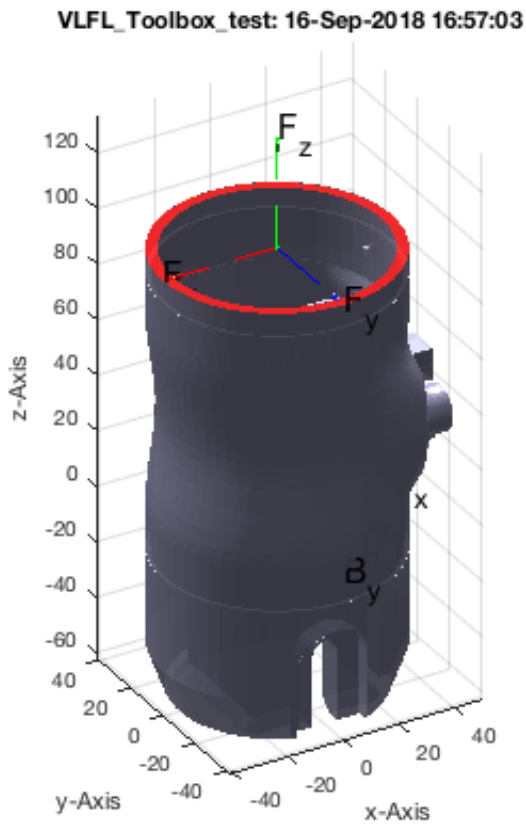
```
B1            1x2                        16  double
C             1x1                     94980  struct
C1            1x2                        16  double
C2            1x2                        16  double
C3            1x2                        16  double
CPA          21x2                       336  double
CPB          37x2                       592  double
CPL         159x2                      2544  double
CVL          93x3                      2232  double
D             1x1                     94980  struct
D1            1x2                        16  double
D2            1x2                        16  double
D3            1x2                        16  double
DVL           0x3                         0  double
FN            1x82                      164  char
FN2           1x73                      146  char
I           800x800                 5120000  double
I1          356x452x3                482736  uint8
ID            1x1                   1238680  struct
IE            1x1                   1931296  struct
IM            1x1                   2948152  struct
IT            1x1                   1931296  struct
JACO          1x8                   6371408  cell
JC0           1x1                   1100646  struct
JC00          1x1                   1465958  struct
JC01          1x1                    369662  struct
JC1           1x1                    843878  struct
JC2           1x1                    757118  struct
```

```
JC3          1x1                  695158  struct
JC4          1x1                  477846  struct
JC5          1x1                  477846  struct
JC6          1x1                 3731758  struct
JC61         1x1                 1431998  struct
JCF          1x1                  220710  struct
L            1x4                      32  double
L1           1x1                       8  double
L2           1x1                       8  double
L3           1x1                       8  double
L4           1x1                       8  double
LMax         1x1                       8  double
LMin         1x1                       8  double
PL          45x2                     720  double
RIL         45x1                     360  double
RL           1x9                      72  double
Ri           1x1                       8  double
Rnv          1x3                      24  double
Ro           1x1                       8  double
SG           1x1                    1312  struct
SG1          1x1                 1143640  struct
SG2          1x1                 1046272  struct
SG3          1x1                 4009792  struct
SG4          1x1                 2507896  struct
SG5          1x1                 2397568  struct
SG6          1x1                 3054472  struct
SGN          4x1                  410240  cell
SGX          1x1                 1143640  struct
T            4x4                     128  double
```

```
TC            4x4                        128  double

Ti            4x4                        128  double

V           512x512x126            66060288  uint16

VL           10x3                        240  double

VM          128x128x128            16777216  double

a           216x216x126            47029248  double

ans         176x3                      4224  double

as            1x3                        24  double

conn          1x1                         0  database.jdbc.connection

d             1x2                        16  double

h             1x1                         0  matlab.graphics.primitive.Patch

l             1x1                         8  double

l1            1x1                         8  double

l2            1x1                         8  double

l3            1x1                         8  double

ms            1x3                        24  double

p             3x1                        24  double

p1            3x1                        24  double

p2            3x1                        24  double

phi           1x1                         8  double

pn            1x3                        24  double

simOut        1x1                     23733  Simulink.SimulationOutput

slot          1x1                         8  double

smbsys        1x13                       26  char

v             3x1                        24  double

vname         1x82                      164  char

vs            1x3                        24  double

z             1x1                         8  double
```

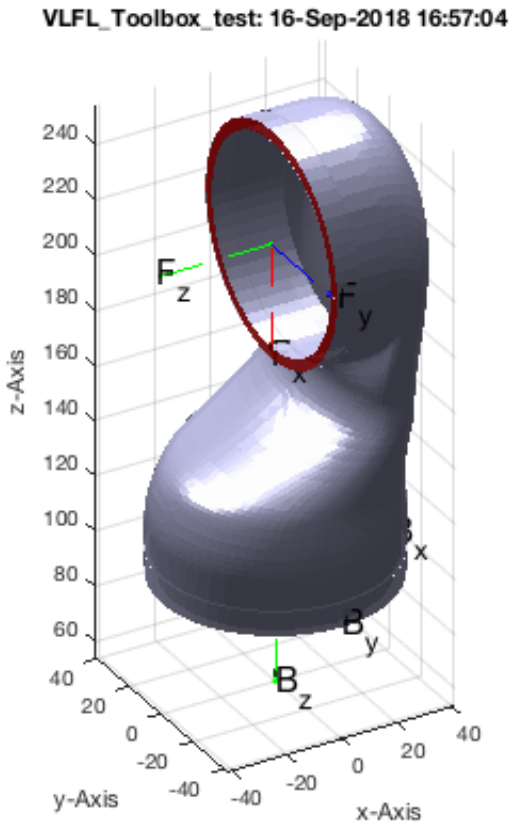
Plot the controller module/base of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC0);
```



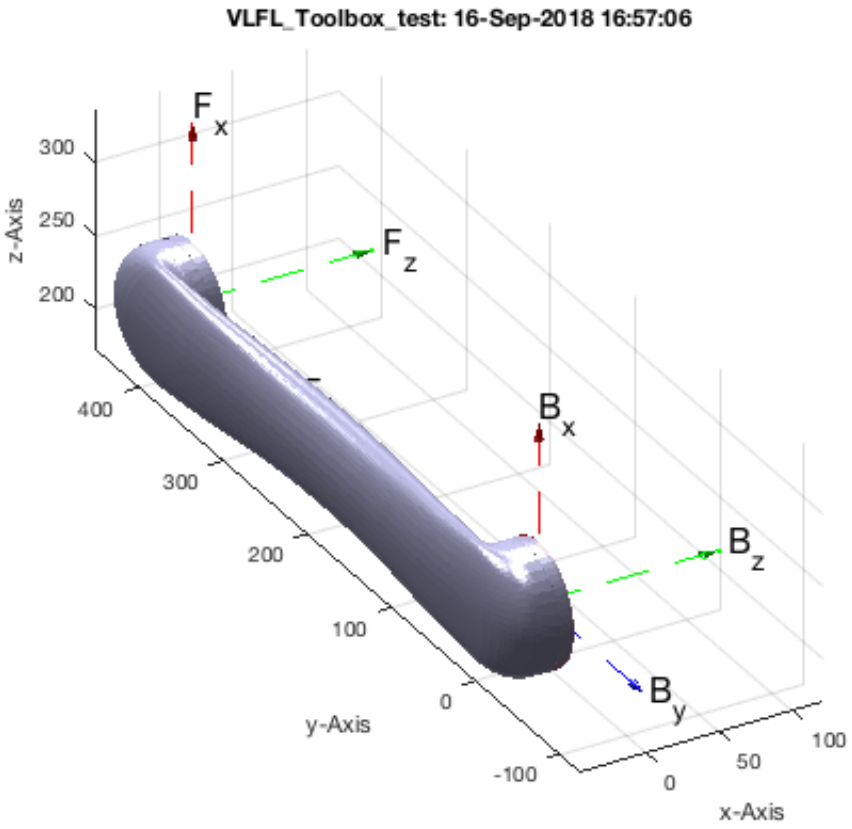
Plot the arm segment 1 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC1);
```



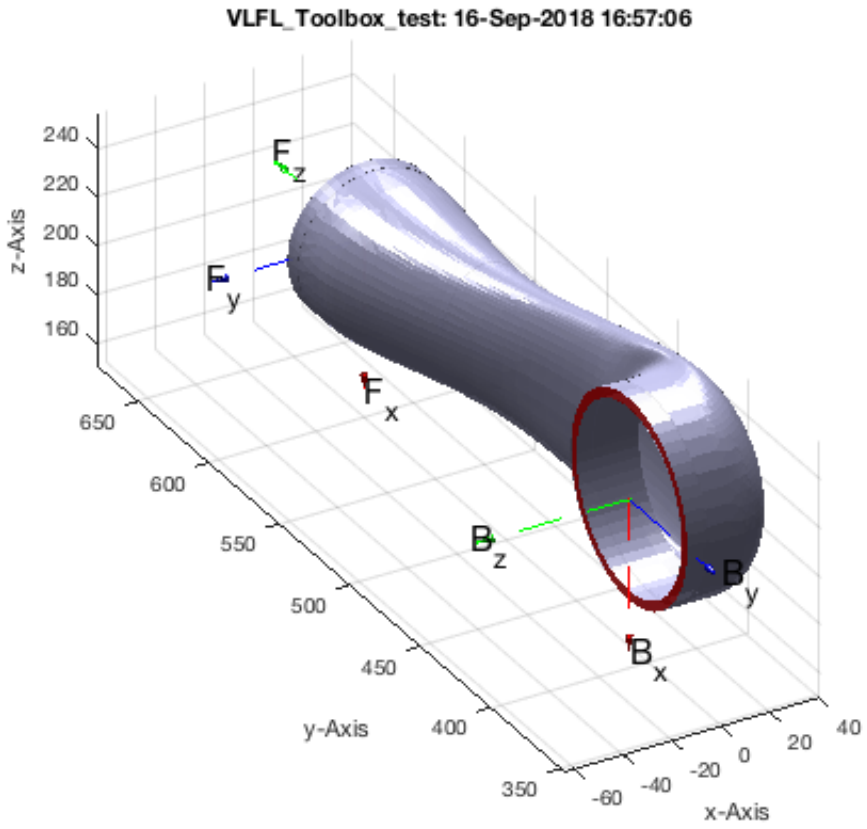
Plot the arm segment 2 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC2);
```



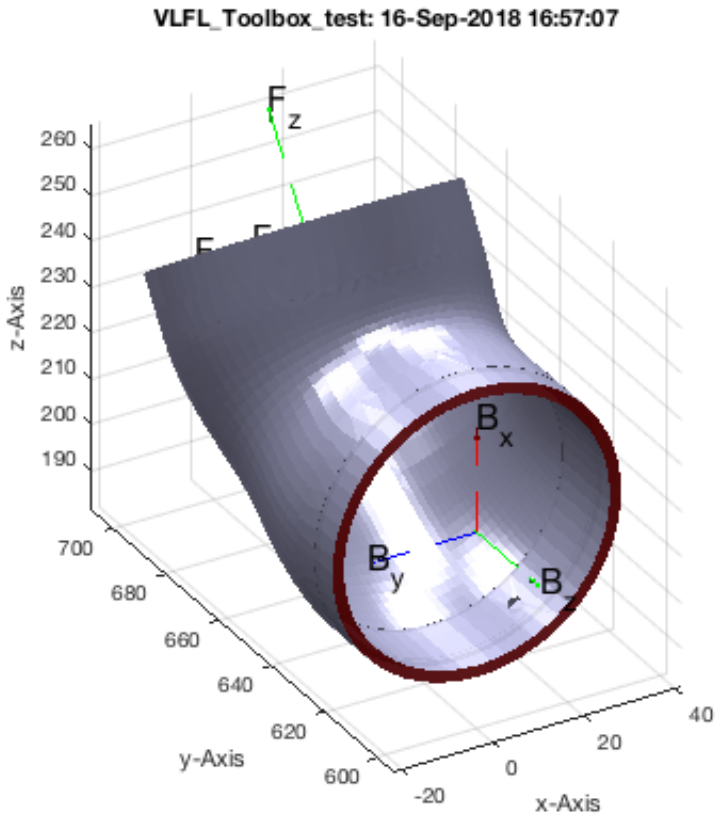
Plot the arm segment 3 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC3);
```



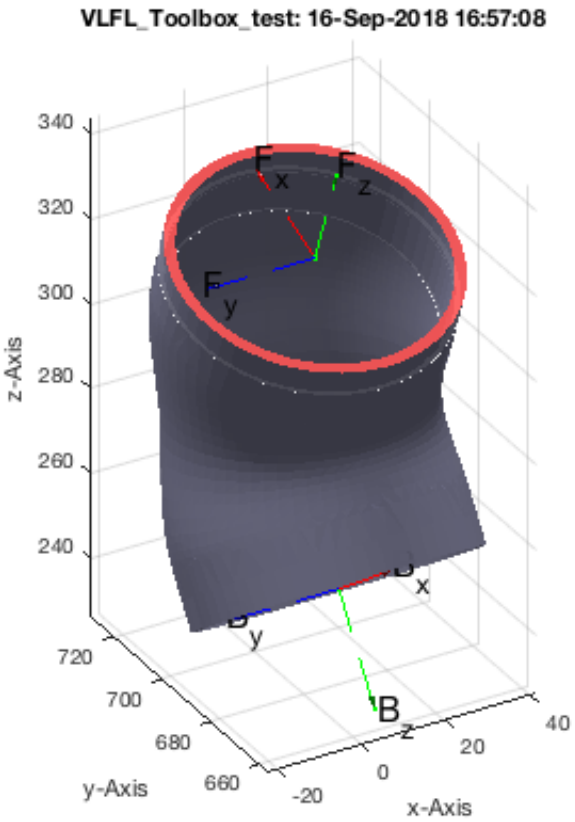
Plot the arm segment 4 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC4);
```



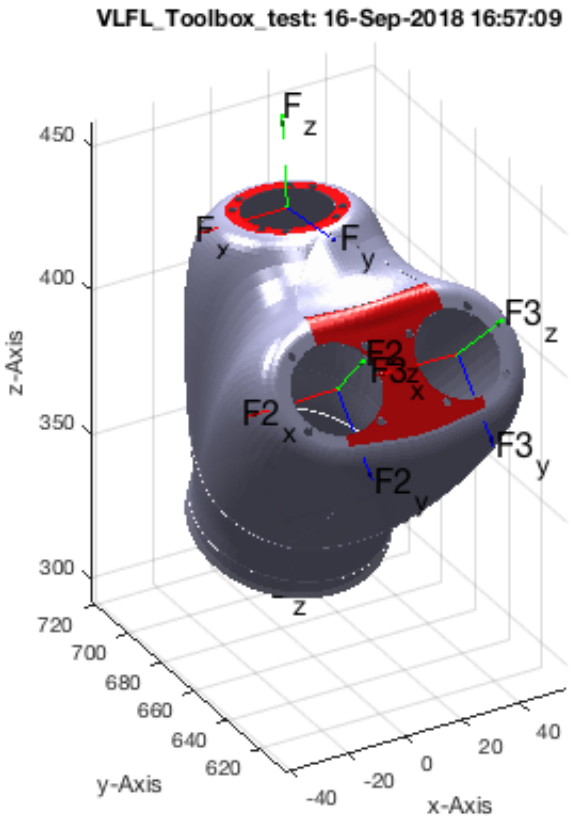
Plot the arm segment 5 of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC5);
```



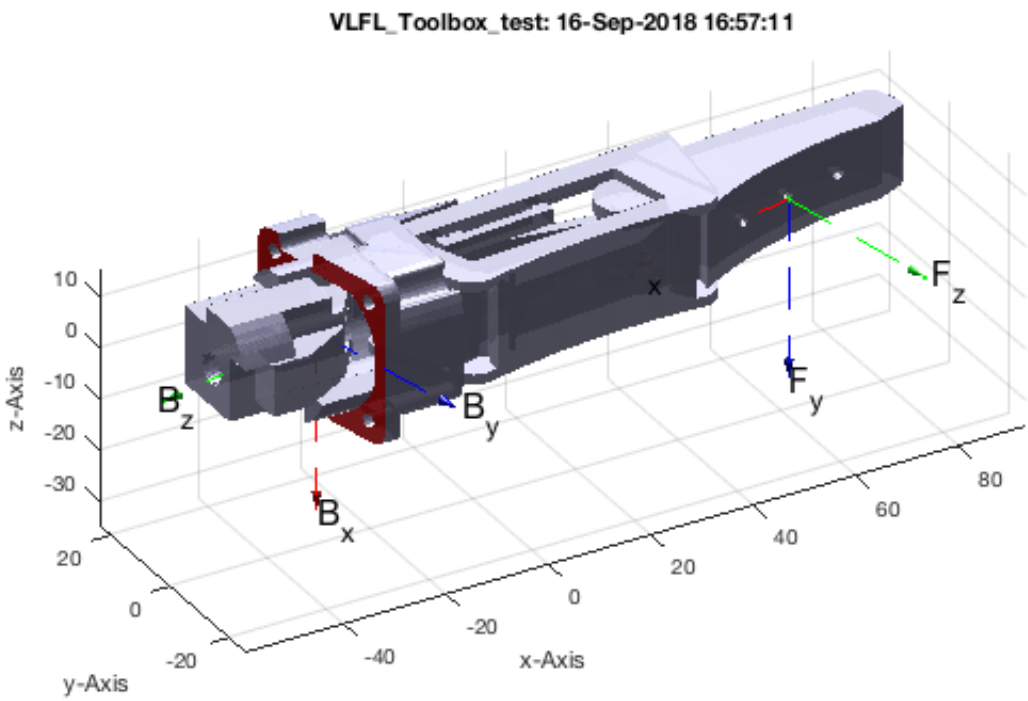
Plot the arm segment 6/the hand of the Jaco robot

```
SGfigure; view(-30,30); SGTplot(JC61);
```



Plot one finger segment 3 of the Jaco robot's hand

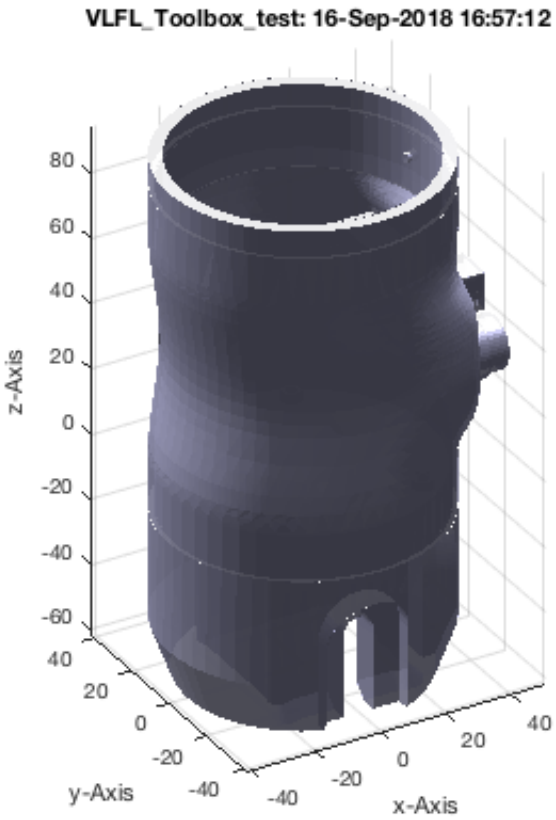
```
SGfigure; view(-30,30); SGTplot(JCF);
```



2. Attaching Frames to a Surface Model

To learn how to attach frames, we make a copy of only the surface of jaco's base.

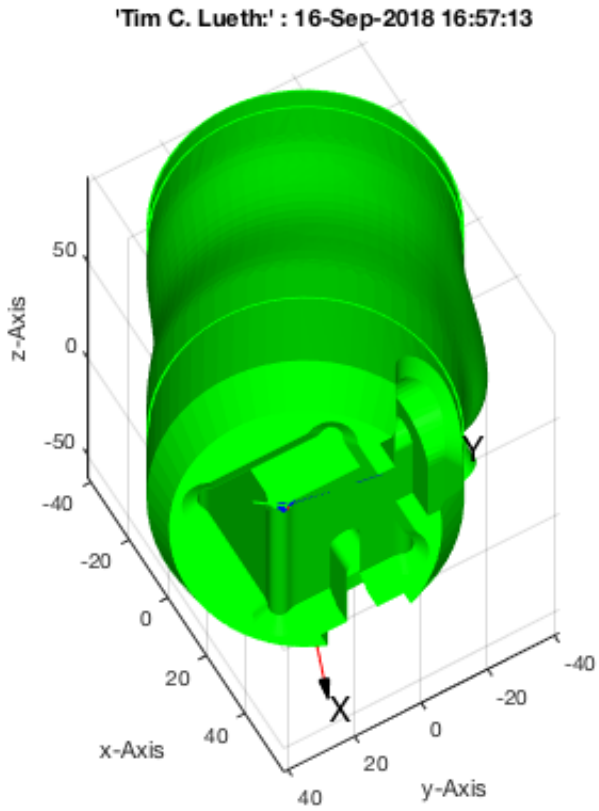
```
SG.VL=JC0.VL; SG.FL=JC0.FL; SG.col='w'; SG.alpha=0.9;

SGfigure; view(-30,30); SGplot(SG);
```



Now use SGTui to specify a planar or freeform surface by klicking on the surface. Turn the object before the klick into the desired orientation Now try to create a base frame by clicking on the lower surface. If you touch a freeform surface it may take while until the surfaces are autoamtically selected

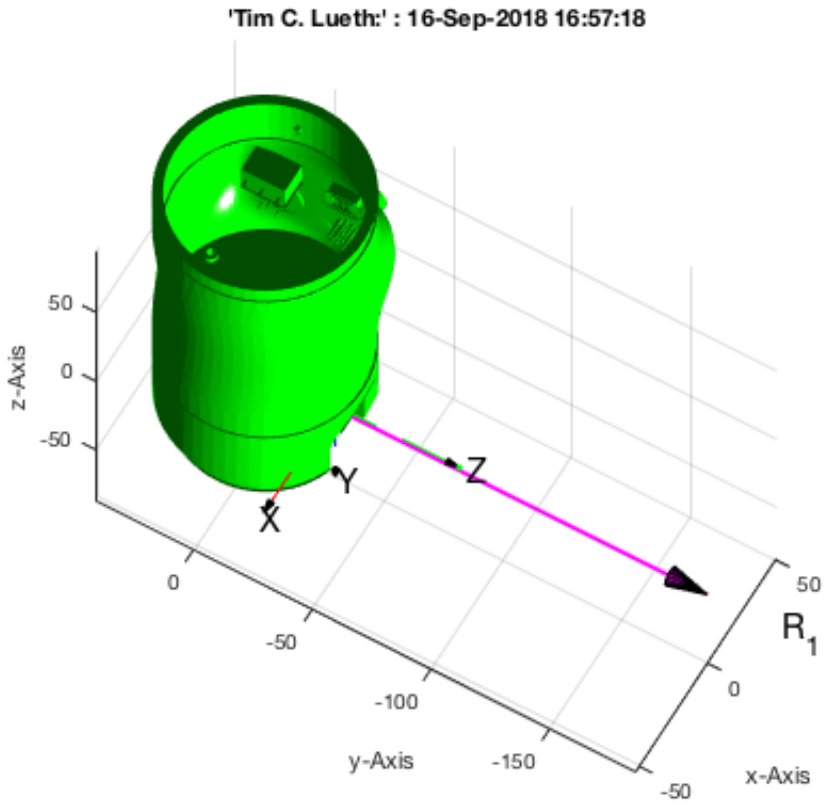
```
SGfigure; SG=SGTui(SG,'B'), view(-60,-60);
```

```
SG =

  struct with fields:

      VL: [15230×3 double]
      FL: [30472×3 double]
     col: 'w'
   alpha: 0.9000
   Tname: {'B'}
       T: {[4×4 double]}
    TFiL: {[]}
    TFoL: {[]}
```




```
SGfigure; SG=SGTui(SG,'F'), view(-60,+60);
```

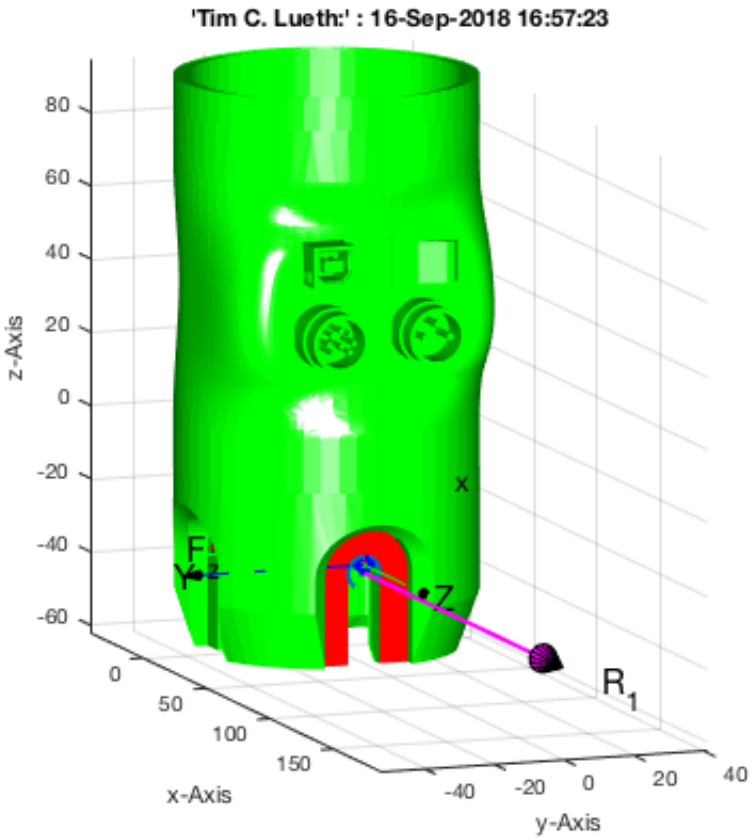
```
SG =

  struct with fields:

       VL: [15230×3 double]
       FL: [30472×3 double]
      col: 'w'
    alpha: 0.9000
    Tname: {'B'  'F'}
        T: {[4×4 double]  [4×4 double]}
     TFiL: {[]  [42×1 double]}
     TFoL: {[]}
```



You may have notices that not only a surface but also the center of circular contours were detected and those can also be used for selection

```
SGfigure; SG=SGTui(SG,'C'), view(70,+10);
```

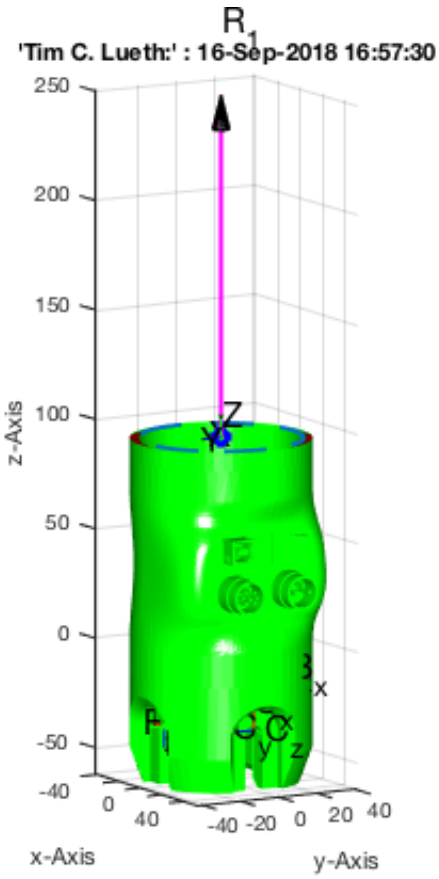
```
SG =

  struct with fields:

      VL: [15230×3 double]
      FL: [30472×3 double]
     col: 'w'
   alpha: 0.9000
   Tname: {'B'  'F'  'C'}
       T: {[4×4 double]  [4×4 double]  [4×4 double]}
    TFiL: {[]  [42×1 double]  [42×1 double]}
    TFoL: {[]}
```



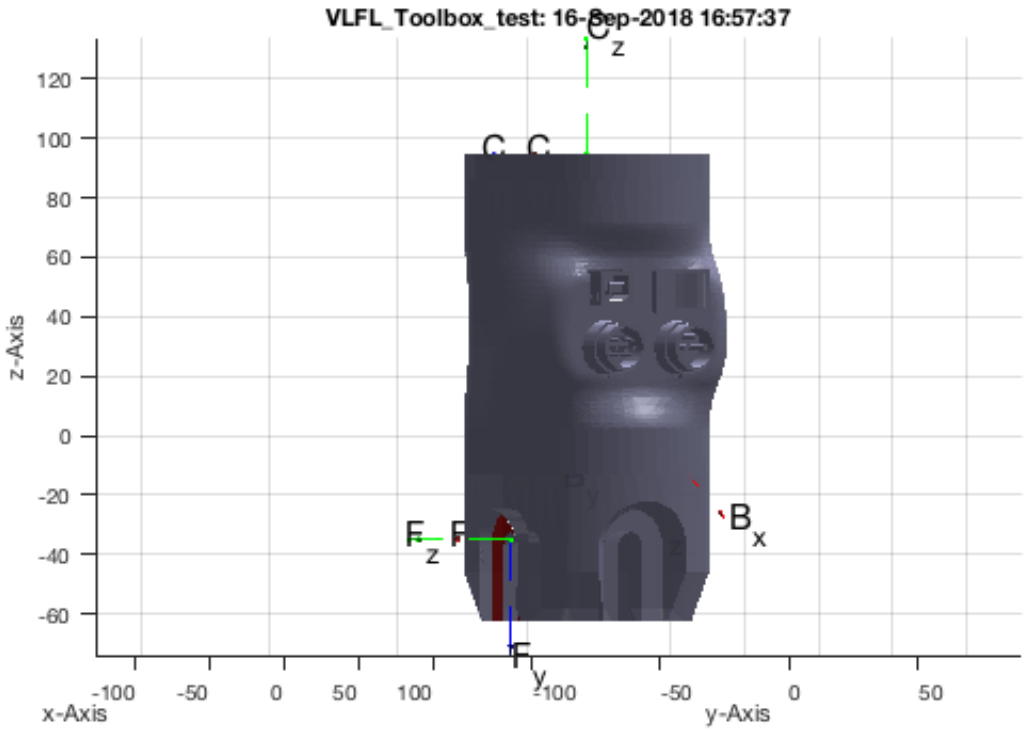
There is a slight difference between the center of the faces and the circle R1. By using 'R1' as parameter, the R1 coordinate system is used for the frame "C"

```
SGfigure; SG=SGTui(SG,'C','','R1'), view(60,+10);
```

```
SG =

  struct with fields:

      VL: [15230×3 double]
      FL: [30472×3 double]
     col: 'w'
   alpha: 0.9000
   Tname: {'B'  'F'  'C'}
       T: {[4×4 double]  [4×4 double]  [4×4 double]}
    TFiL: {[]  [42×1 double]  [117×1 double]}
    TFoL: {[]}
```

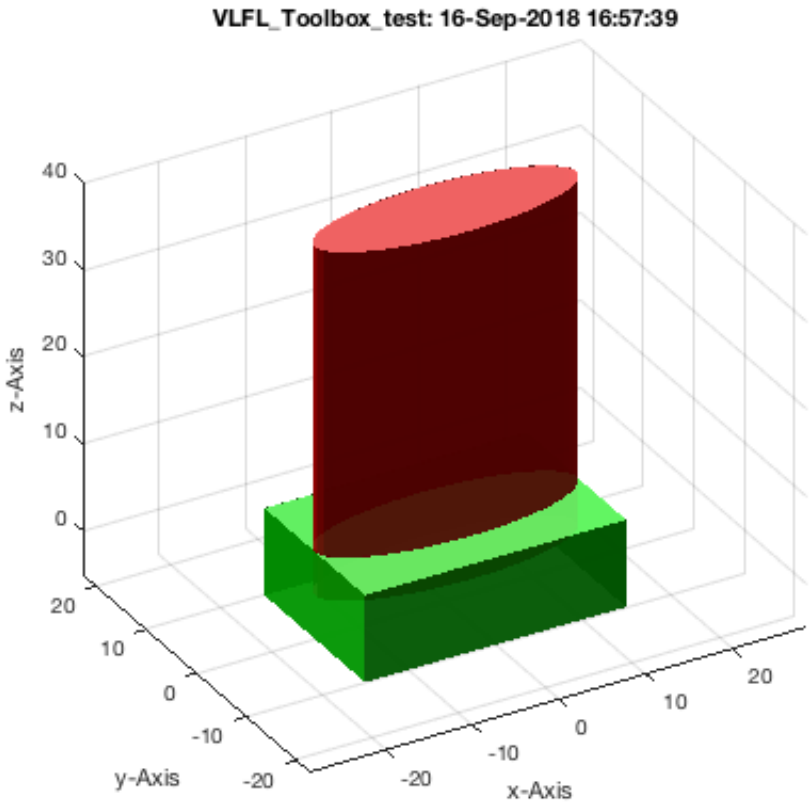



```
SGfigure;  SGTplot(SG,'C'); view(60,+0);
```



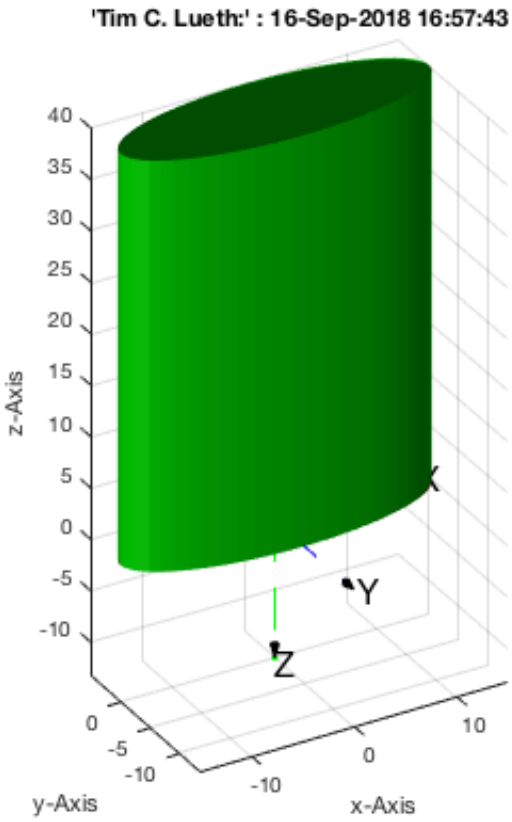
3. Spatial Arrangment of Solids relative to Frames

```
A=SGbox([30,20,10]); A.col='g'; A.alpha=0.9;
B=SGofCPLz(PLcircle(15,'','',5),40); B.col='r'; B.alpha=0.9;
SGfigure; SGplot({A,B}); view(-30,30);
```

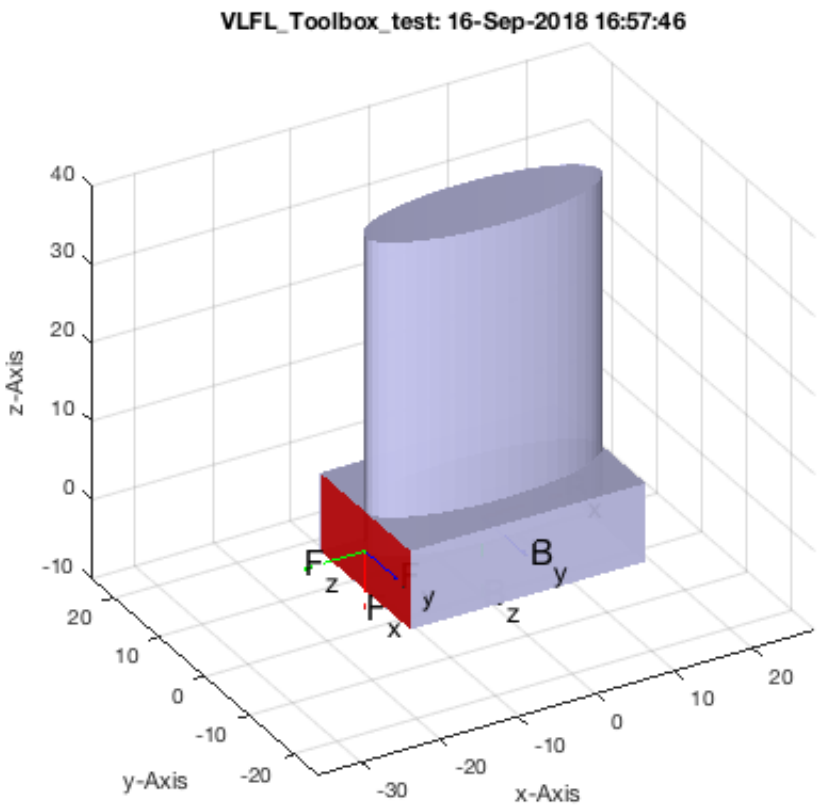


Now attach frame to both solids

```
A=SGTui(A,'F');  % Follower Frame
B=SGTui(B,'B');  % Base Frame
```



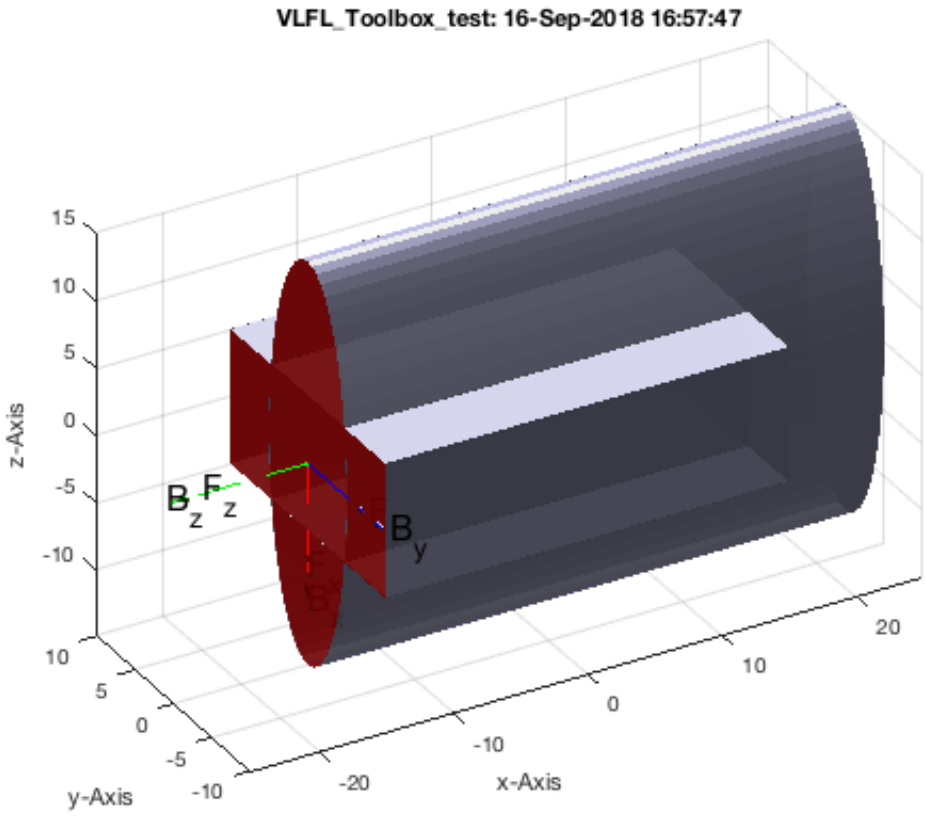

```
SGfigure; SGTplot(A); SGTplot(B); view(-30,30);
```


VLFL_Toolbox_test: 16-Sep-2018 16:57:46
z-Axis
40
30
20
10
0
-10
y-Axis
20
10
0
-10
-20
x-Axis
-30
-20
-10
0
10
20
F
z
y
x
B
y
z

Now position solid B that its Frame 'B' matches with Frame 'F' of Solid A Afterwards, both Frames overlap completely.

```
SGtransrelSG(B,A,'matchT',{'B','F'})
```

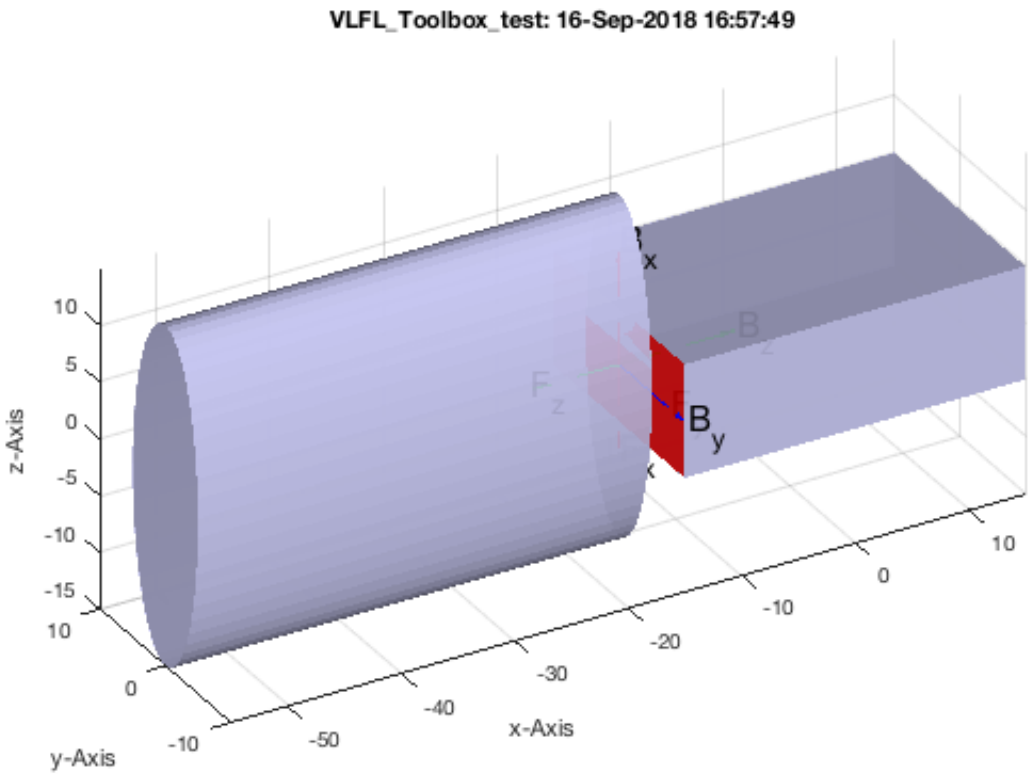
```
ans =

  struct with fields:

      CPL: [55×2 double]
       VL: [110×3 double]
       FL: [216×3 double]
       PL: [55×2 double]
       EL: [55×2 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations.

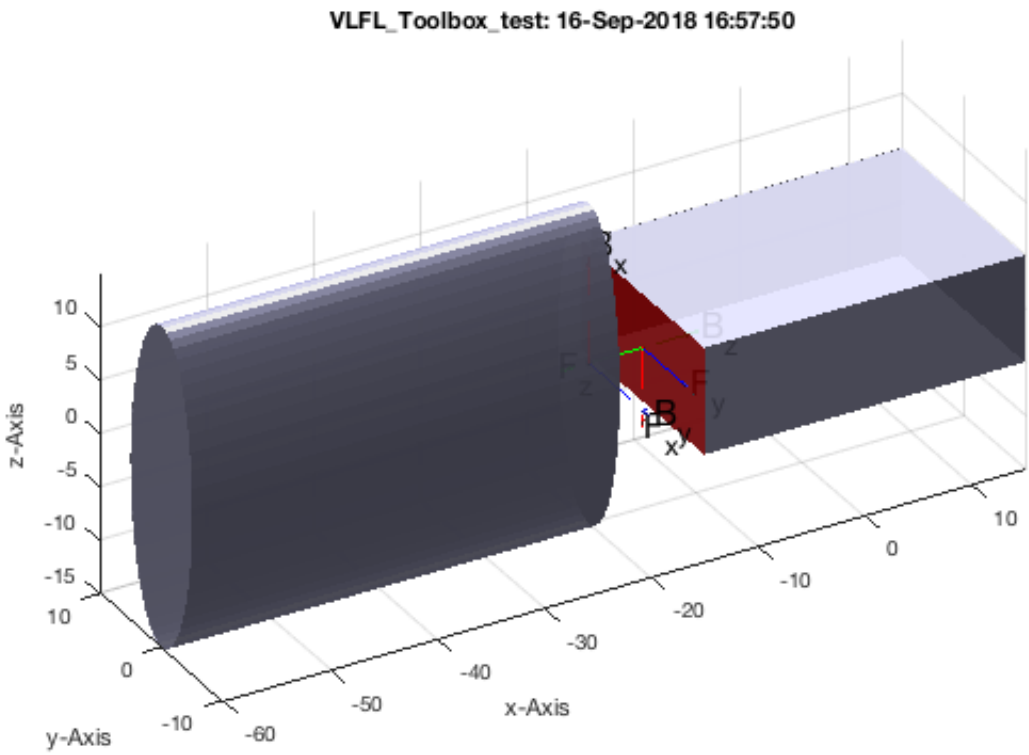
```
SGtransrelSG(B,A,'alignT',{'B','F'})
```

```
ans =

  struct with fields:

      CPL: [55×2 double]
       VL: [110×3 double]
       FL: [216×3 double]
       PL: [55×2 double]
       EL: [55×2 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations. IN ADDITION create a distance of 5 mm

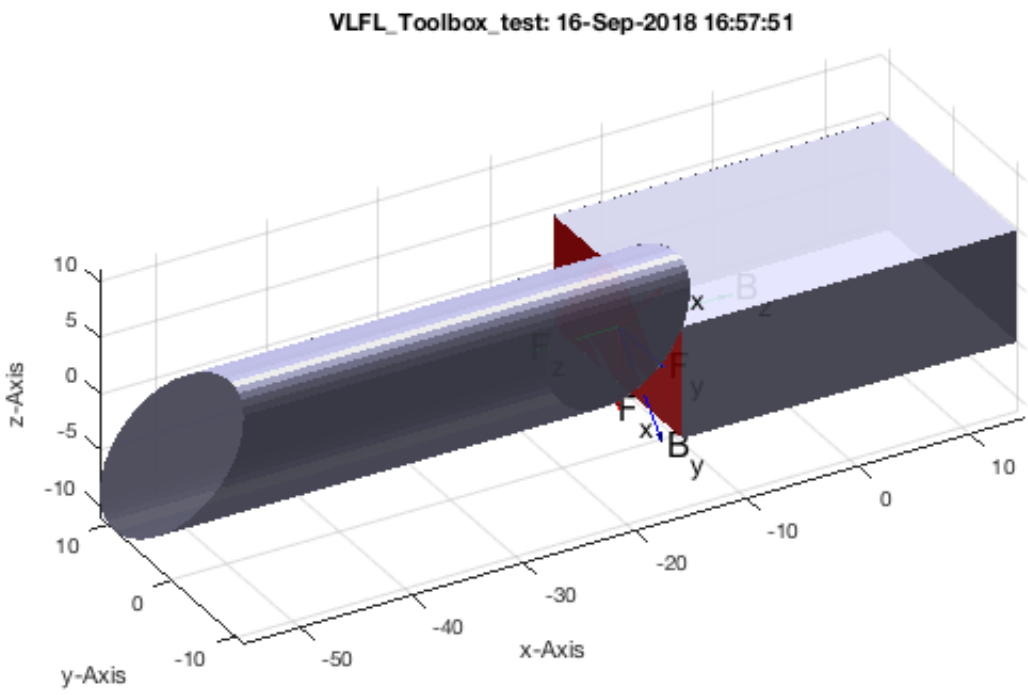
```
SGtransrelSG(B,A,'alignT',{'B','F',TofP([0 0 -5])});
```



Now position solid B that its Frame 'B' aligns with Frame 'F' of Solid A Afterwards, both only axis Y overlap completely. Z and X have opposite orienations. IN ADDITION TURN 45 degrees

```
SGtransrelSG(B,A,'alignT',{'B','F',TofR(rot(0,0,pi/4))})
```

```
ans =

  struct with fields:

      CPL: [55×2 double]
       VL: [110×3 double]
       FL: [216×3 double]
       PL: [55×2 double]
       EL: [55×2 double]
      col: 'r'
    alpha: 0.9000
    Tname: {'B'}
        T: {[4×4 double]}
     TFiL: {[53×1 double]}
     TFoL: {[]}
```

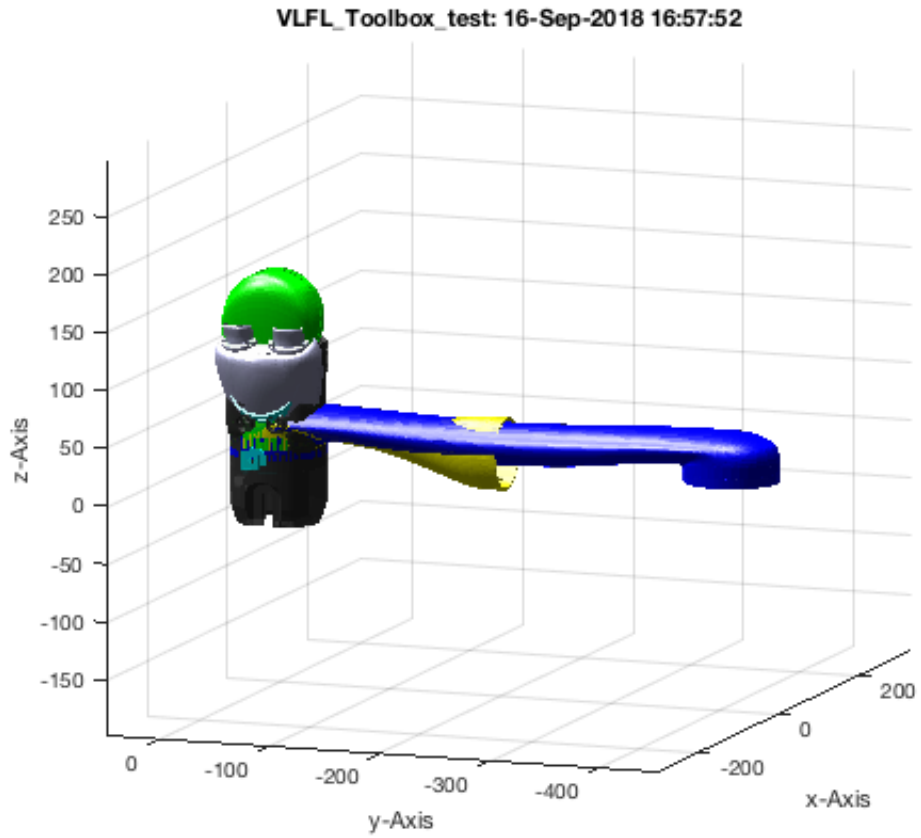


4. Simple Sequential Kinematic Chains

As soon as all solids have a base frame and a follower frame, it is possible to consider them als kinematic chain with some degrees of freedom between the frame. Such as rotation around the z-axis of the follower frame. The easist case is to define a cell list of all involved solids. To explain this feature, the origins of all solids are changed to their base frames. This is done just to avoid misunderstandings.

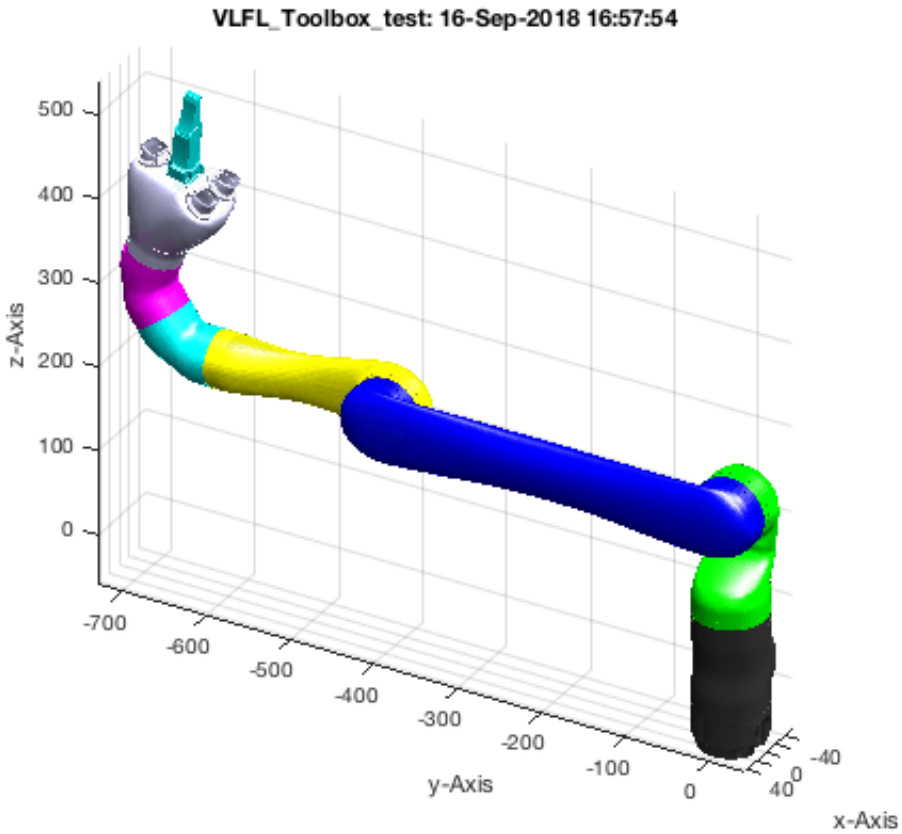
```
JC0=SGTsetorigin(JC0,'B'); % change the origin of Solid to Frame 'B'
JC1=SGTsetorigin(JC1,'B'); % change the origin of Solid to Frame 'B'
JC2=SGTsetorigin(JC2,'B'); % change the origin of Solid to Frame 'B'
JC3=SGTsetorigin(JC3,'B'); % change the origin of Solid to Frame 'B'
JC4=SGTsetorigin(JC4,'B'); % change the origin of Solid to Frame 'B'
JC5=SGTsetorigin(JC5,'B'); % change the origin of Solid to Frame 'B'
JC6=SGTsetorigin(JC6,'B'); % change the origin of Solid to Frame 'B'
JACO={JC0,JC1,JC2,JC3,JC4,JC5,JC6,JCF}
SGfigure; SGplot(JACO);   view(-70,10);
```

```
JACO =

  1×8 cell array

  Columns 1 through 4

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}

  Columns 5 through 8

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
```



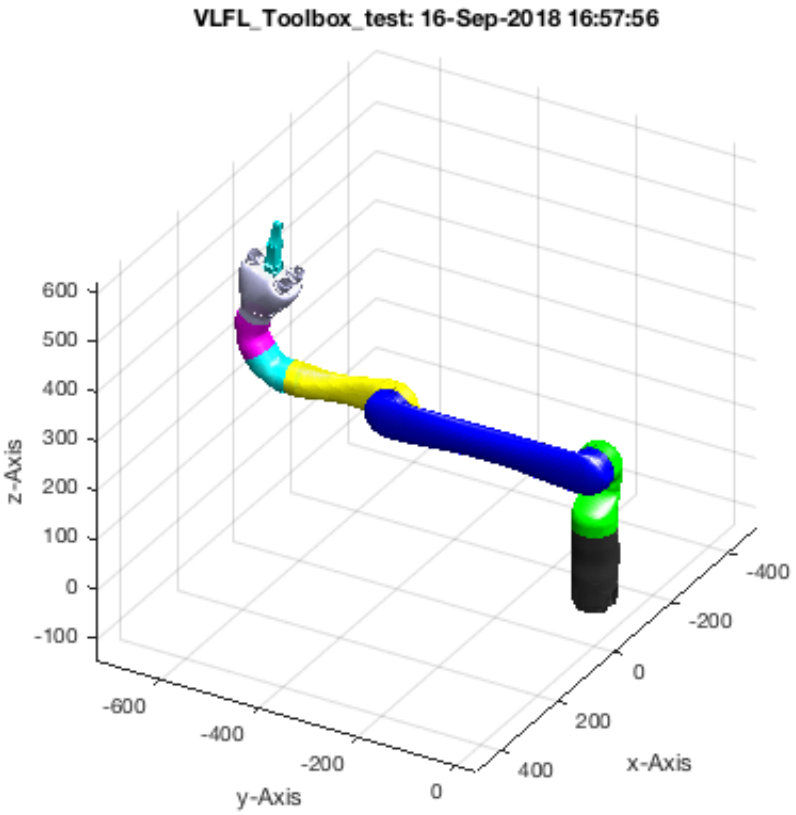
There is a function that aligns automatically base and follower frame AND modifies the vertex list for all of the solids but the first one.

```
SGfigure; SGTchain(JACO); view(120,30);
```



The function SGTchain changes the all vertex coordinates, therefor afterwards the parts seem to stay is space as the kinematic chain. In this example X is a pose of the robot if all frames are aligned. If X is plotted as a solid it looks like a robot in a specific pose.

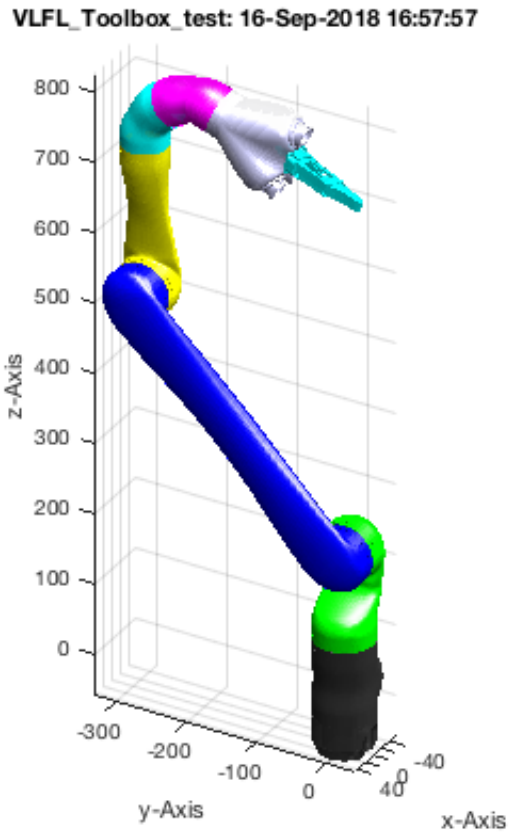
```
X=SGTchain(JACO);
SGfigure; SGplot(X); view(120,30);
```



SGTchain also allow to deliver additional rotatorial parameters. For each joint a rotating angle can be specified. NEvertehelss, currently the first value is ignored, since there is no base frame. The nth rotation is relative to the base frame

of the nth element.

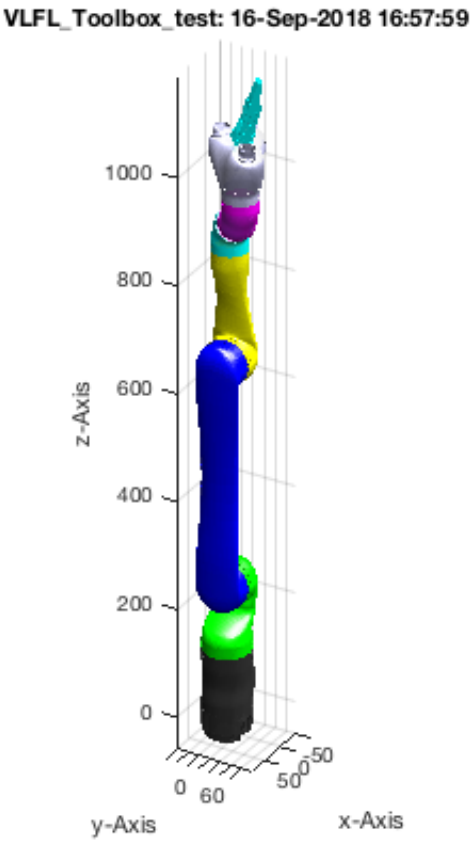
```
SGTchain(JACO,[nan 0 +pi/4 -pi/4]); view(120,30);

% again, the output value is the same surface cell list but describing
% exactly this position.
```



5. Calibration of a Sequential Kinematic Chain

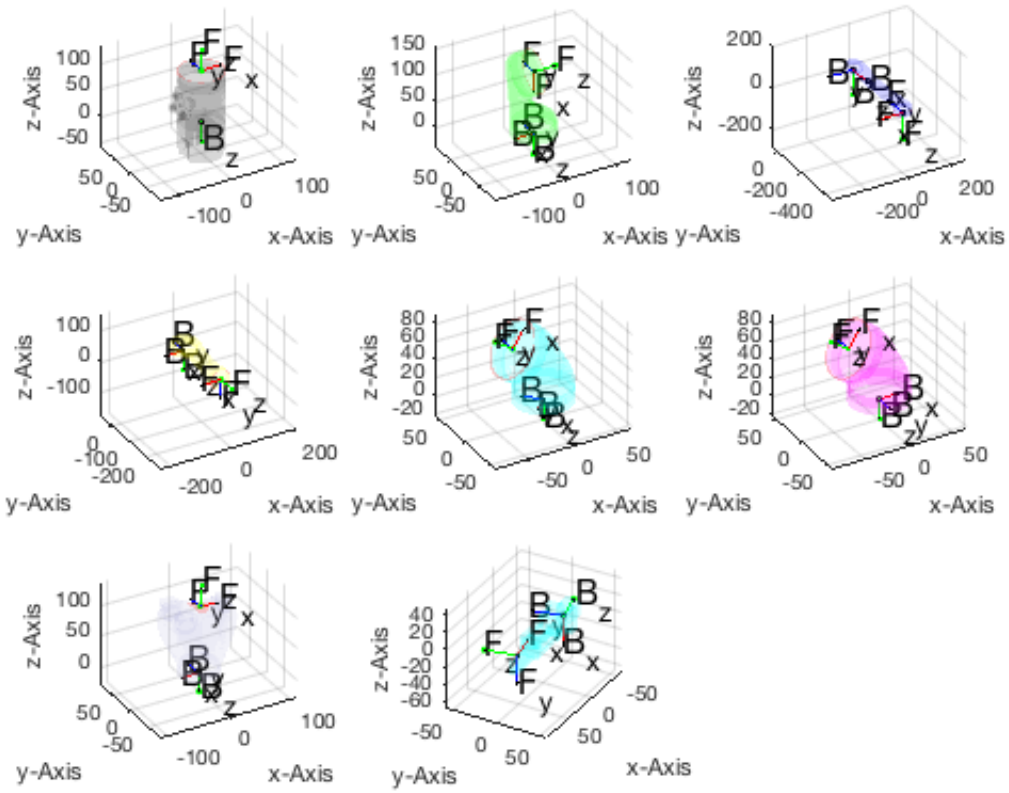
Often it is not possible to specify the frames using SGTui exactly as the real motor configuration is. Therefor it is necessary to calibrate the zero position. In case try to bring the robot by a set of rotating angles into the desired zero position or use an additional angle vector as offest. As soon as the offset is know call SGTcalibchain using the offset values. For example

```
SGTchain(JACO,[nan 0 pi/2 0 pi/4 -pi 0]); view(120,30);
```



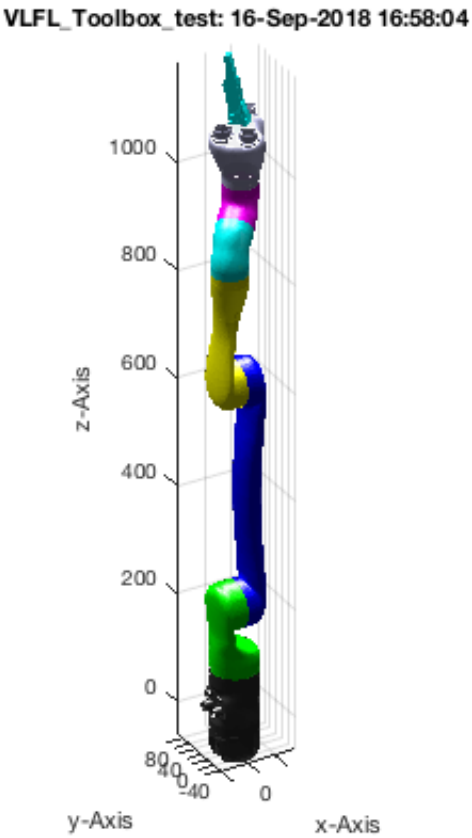
now change all frames of the chain to create a new zero position. In this case ALL elements need a value. Even the finger element.

```
SGTcalibchain(JACO,[nan 0 pi/2 0 pi/4 -pi 0 0]); view(120,30);
JACO_cal=ans;
```



Now the robot has a new zero position The position shown here has nothing to do with the real zero position of KINOVA's JACO robot.

```
SGTchain(JACO_cal);
```

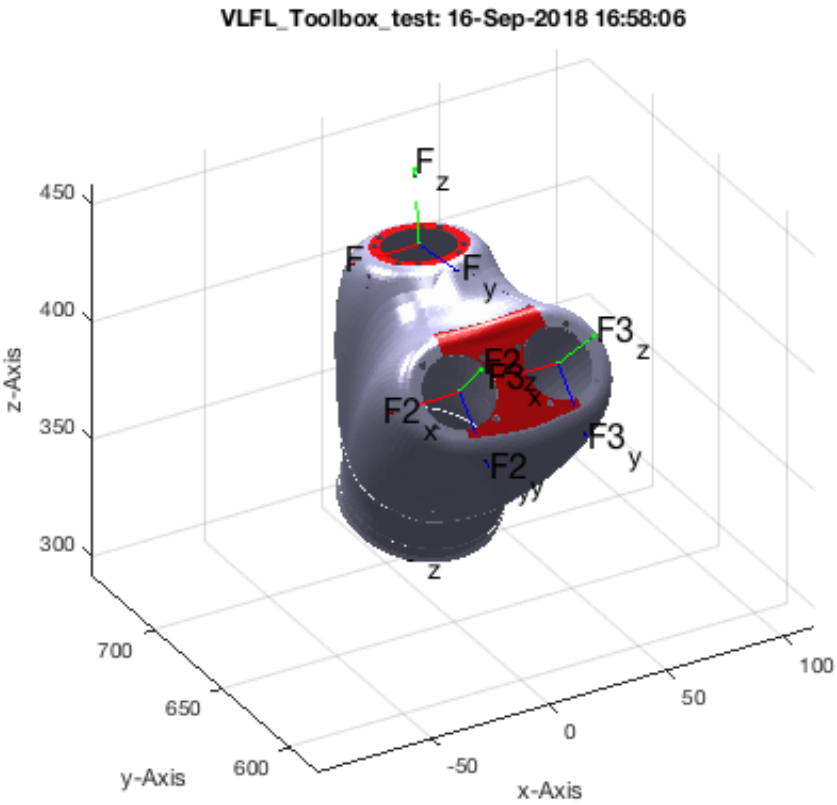


6. Creating Kinematic Trees

It is easy to see that the real JACO has three fingers and a simple chain is not enough. Therefor there is an additional format for SGTchain to explain the kinematic structure and the order of motors/angles. At first we need three follower frames. THis is part of solid JC61. Beside "F" tehre is also "F1" and "F2"

```
SGfigure; SGTplot(JC61); view(-30,30)
JACO={JC0,JC1,JC2,JC3,JC4,JC5,JC61,JCF}
```

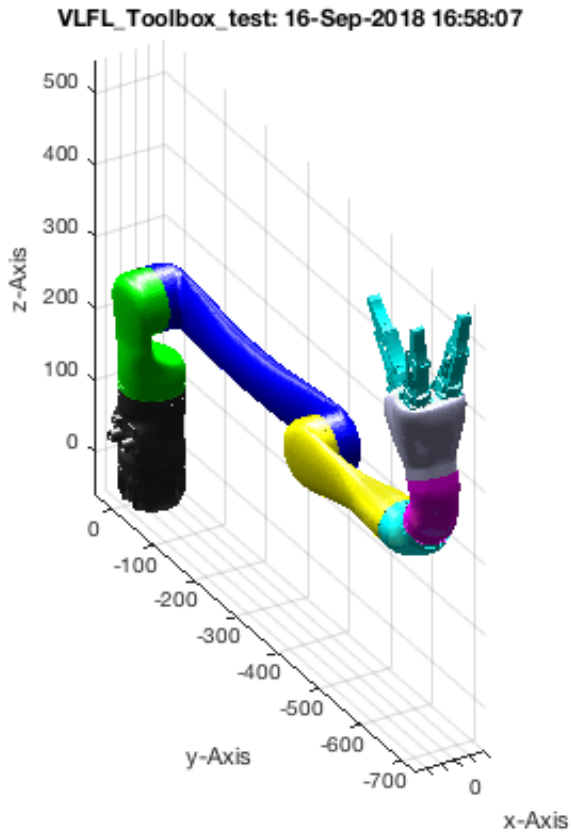
```
JACO =

  1×8 cell array

  Columns 1 through 4

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}

  Columns 5 through 8

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}
```



Next is to specify two additional degrees of freedom between Part 7 and Frame "F2" and Part 8 Frame "B" and Part 7 and Frame "F3" and Part 8 Frame "B". Automatically, there are two additional rotations or motors introduced. In case of the real JACO robot, the joints 7, 8, 9 are not rotational but linear for the fingers.

```
SGTchain(JACO,'','',1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
```

```
ans =

  1×10 cell array

  Columns 1 through 4

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}

  Columns 5 through 8

    {1×1 struct}    {1×1 struct}    {1×1 struct}    {1×1 struct}

  Columns 9 through 10

    {1×1 struct}    {1×1 struct}
```

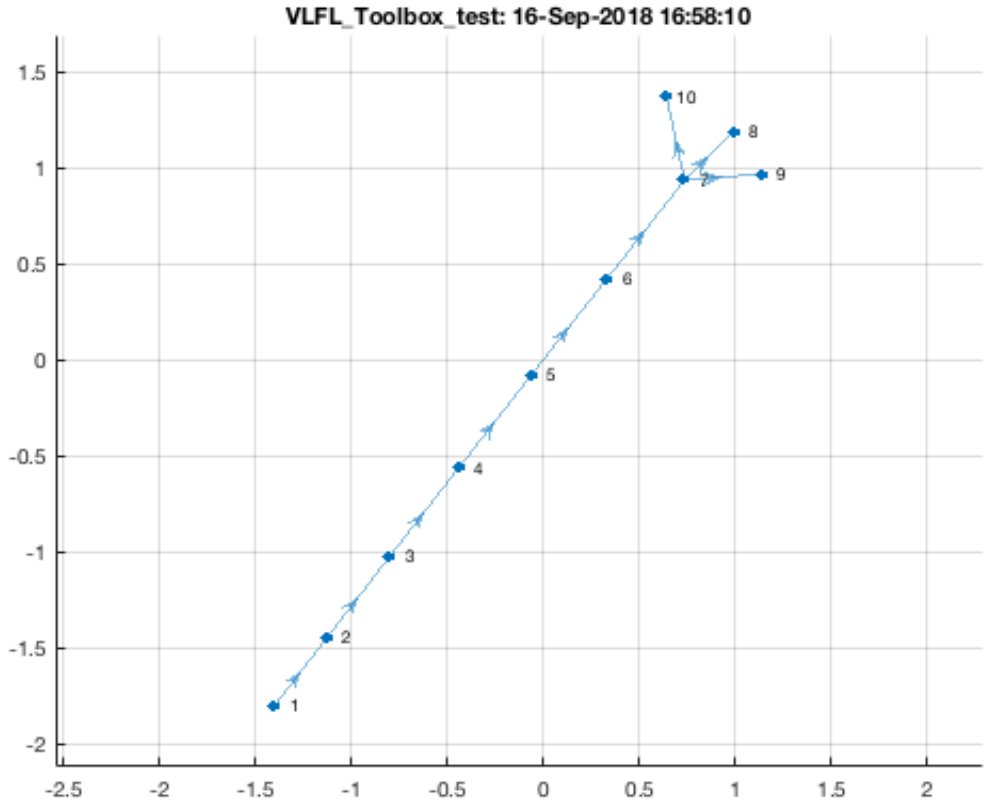


To understand better the kinematic chains, it is also possible to call a auxiliary function to create a kinematic chain table. This function returns the number/order of the DoF and which frames are connected and which solid was used for the

connection. In Future also the type of DoF will be added to this list

```
SGTframeChain(1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
```

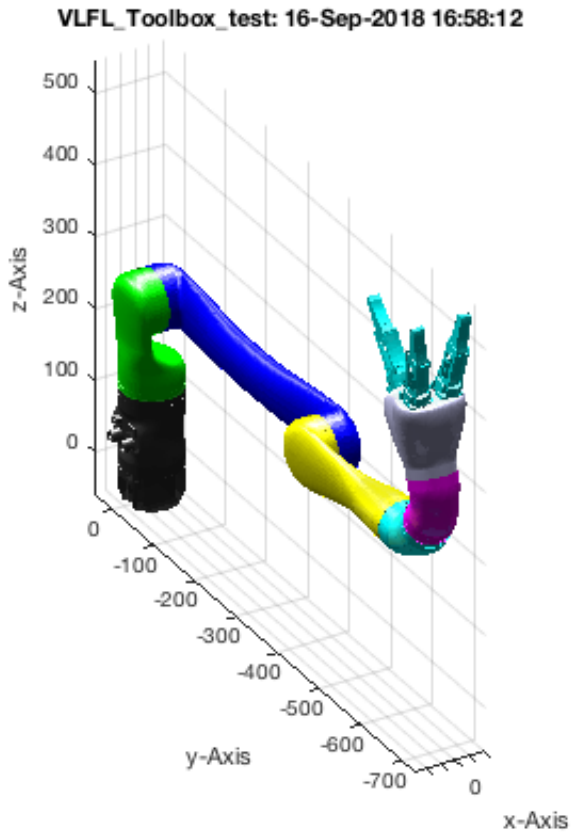
```
ans =

  10×5 cell array

    {[ 1]}    {'_' }    {'B'}    {[0]}    {[1]}
    {[ 2]}    {'F' }    {'B'}    {[1]}    {[2]}
    {[ 3]}    {'F' }    {'B'}    {[2]}    {[3]}
    {[ 4]}    {'F' }    {'B'}    {[3]}    {[4]}
    {[ 5]}    {'F' }    {'B'}    {[4]}    {[5]}
    {[ 6]}    {'F' }    {'B'}    {[5]}    {[6]}
    {[ 7]}    {'F' }    {'B'}    {[6]}    {[7]}
    {[ 8]}    {'F' }    {'B'}    {[7]}    {[8]}
    {[ 9]}    {'F2'}    {'B'}    {[7]}    {[8]}
    {[10]}    {'F3'}    {'B'}    {[7]}    {[8]}
```



It is also possible to call SGT directly using this table:

```
FC=SGTframeChain(1:8,[7 'F2' 8 'B', 7 'F3' 8 'B'])
SGTchain(JACO,'','',FC);
```

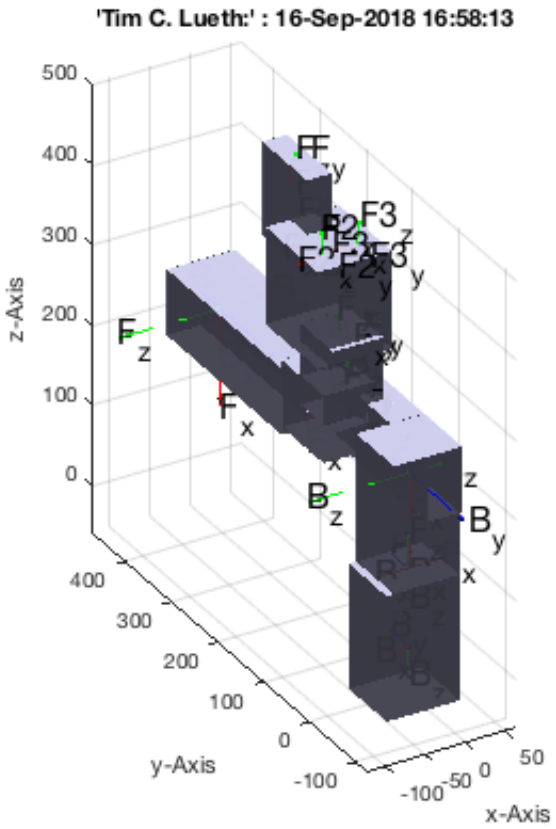
```
FC =

  10×5 cell array

    {[ 1]}    {'_' }    {'B'}    {[0]}    {[1]}
    {[ 2]}    {'F' }    {'B'}    {[1]}    {[2]}
    {[ 3]}    {'F' }    {'B'}    {[2]}    {[3]}
    {[ 4]}    {'F' }    {'B'}    {[3]}    {[4]}
    {[ 5]}    {'F' }    {'B'}    {[4]}    {[5]}
    {[ 6]}    {'F' }    {'B'}    {[5]}    {[6]}
    {[ 7]}    {'F' }    {'B'}    {[6]}    {[7]}
    {[ 8]}    {'F' }    {'B'}    {[7]}    {[8]}
    {[ 9]}    {'F2'}    {'B'}    {[7]}    {[8]}
    {[10]}    {'F3'}    {'B'}    {[7]}    {[8]}
```



7. Calculating Boxes for Quick Collision Checks

The algorithms for collision check are very time consuming since there is a need for testing all traingles for collision/penetration. This makes sensor for bollean operations but is not suitable for gast follision checks during a movement of a kinematic chain. Therefor there is a wish to perform these steps with a simplified kineamtic model, consisting of bounding boxes

```
J=SGTchain(JACO,[nan,0 pi pi]);

SGTBB(J); JB=ans; view(-30,30);
```



8. Collision Check

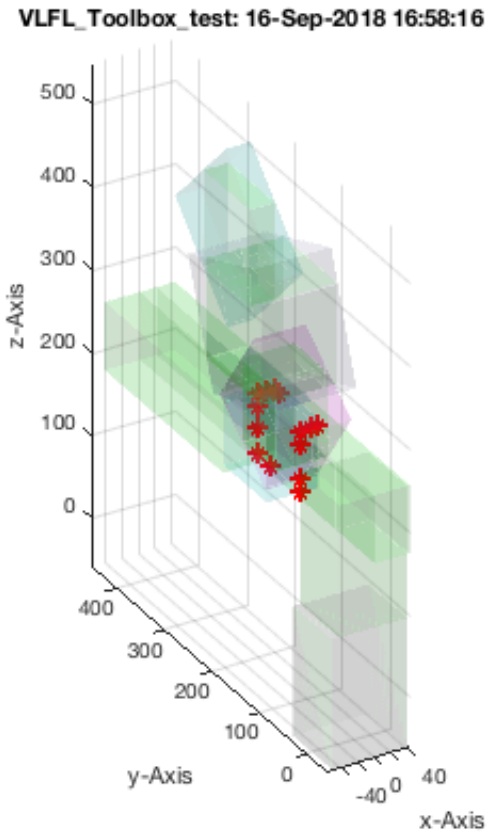
There are two functions:

- **iscollofVLBB** for testing of Vertices are inside of a bounding box
- **iscollofSG** for face testing of two solids or selftest of one solid Please read the documentation for both functions to see what is possible

```
% Self collision test in a safe configuration
iscollofSG(SGTchain(JB,[nan 0 pi pi]))
```

```
ans =

    1.4999  111.7080  257.2501
   14.7772  111.7080  257.2501
   13.5351  111.7080  257.2501
   13.5351  111.7080  257.2501
   21.7502  111.7080  257.2501
   21.7502  111.7080  257.2501
    1.4999  111.7080  241.2684
    1.4999  111.7080  241.2684
    1.4999  111.7080  257.2501
   14.7772  111.7080  257.2501
    1.4999  111.7080  184.4999
    1.4999  111.7080  184.4999
    1.4999  111.7080  200.3724
    1.4999  111.7080  200.3724
```

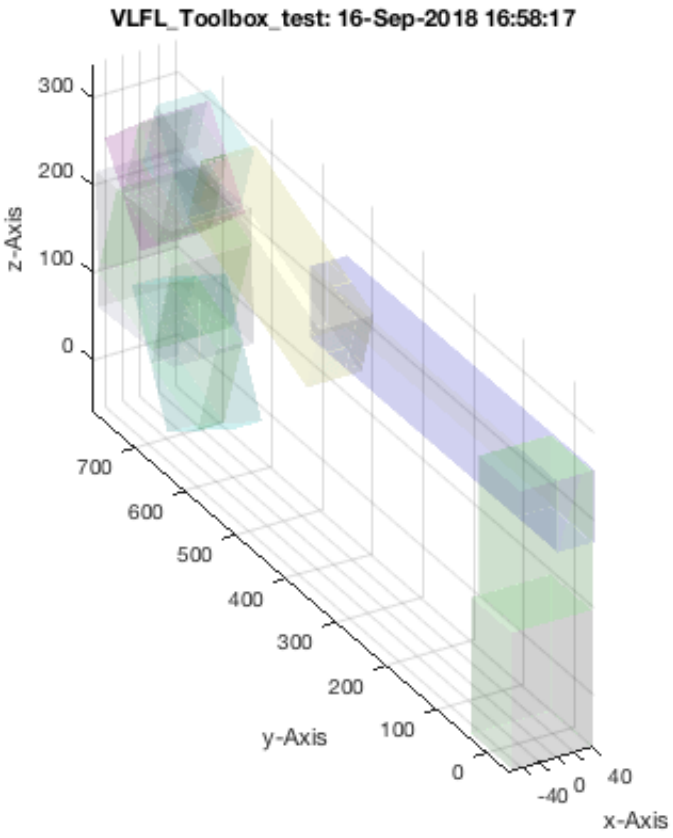
```
    1.4999  173.6560  184.4999
    1.4999  173.6560  184.4999
   21.7502  191.0959  257.2501
   21.7502  191.0959  257.2501
   21.7502  192.0389  257.2501
   21.7502  192.0389  257.2501
    1.4999  203.0001  241.2684
   13.5351  203.0001  257.2501
    1.4999  203.0001  241.2684
   13.5351  203.0001  257.2501
   21.7502  203.0001  257.2501
   21.7502  203.0001  257.2501
    1.4999  203.0001  215.6650
    1.4999  203.0001  257.2501
    1.4999  203.0001  184.4999
    1.4999  203.0001  184.4999
    1.4999  203.0001  257.2501
    1.4999  203.0001  215.6650
```



Self collision test in a problematic configuration

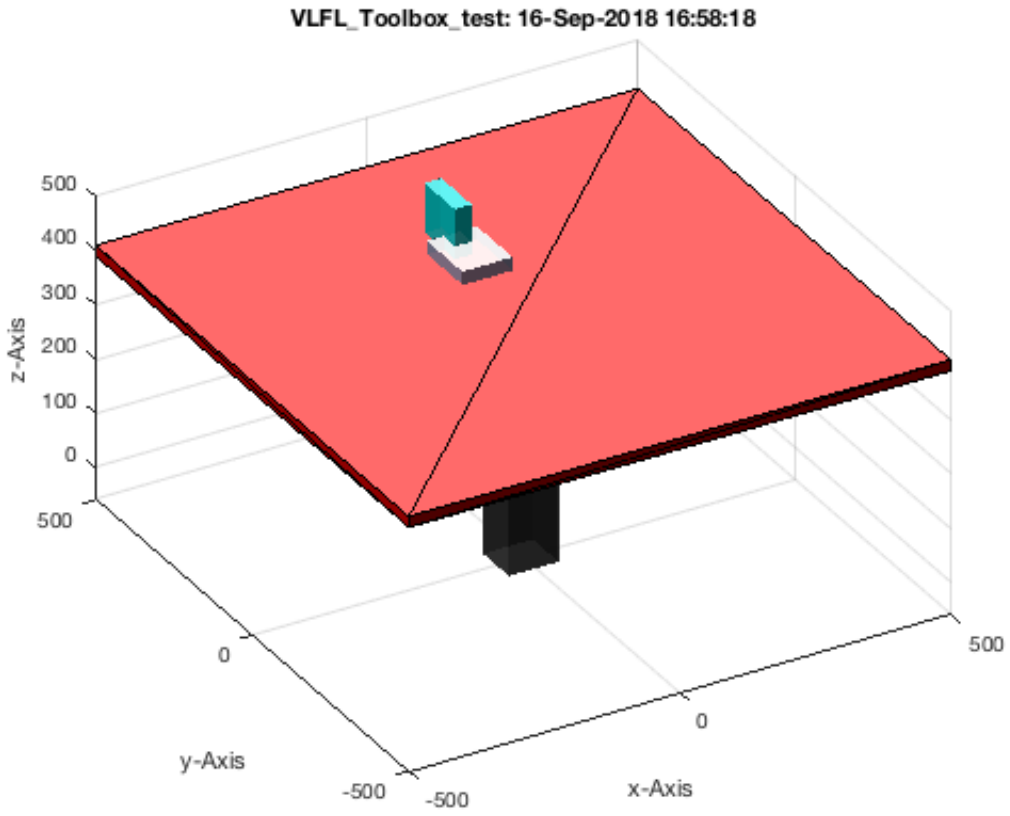
```
iscollofSG(SGTchain(JB,[nan 0 pi pi/10]))
```

```
ans =

  0×3 empty double matrix
```



Collsion collision test in a problematic configuration

```
A=SGbox([1000,1000,20]); A=SGtransP(A,[0 0 400]);
SGTchain(JB,[nan 0 pi pi]); SGplot(A);
```

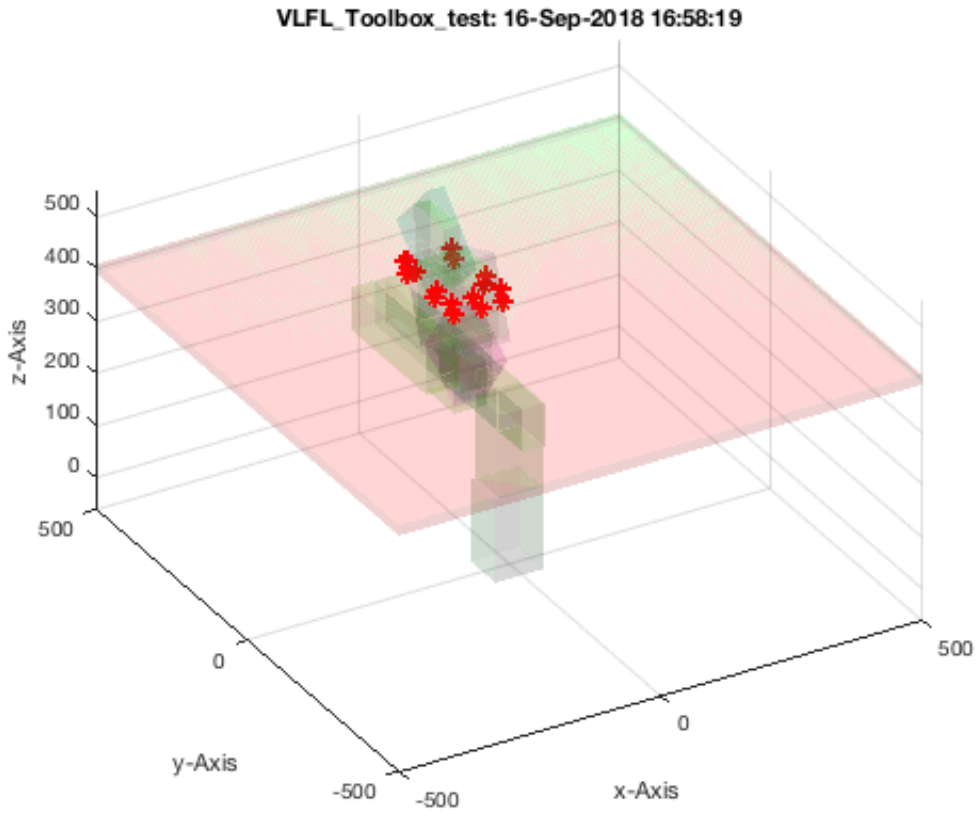


Now make a test for crossing robot and solid

```
iscollofSG(SGTchain(JB,[nan 0 pi pi]),A)
```

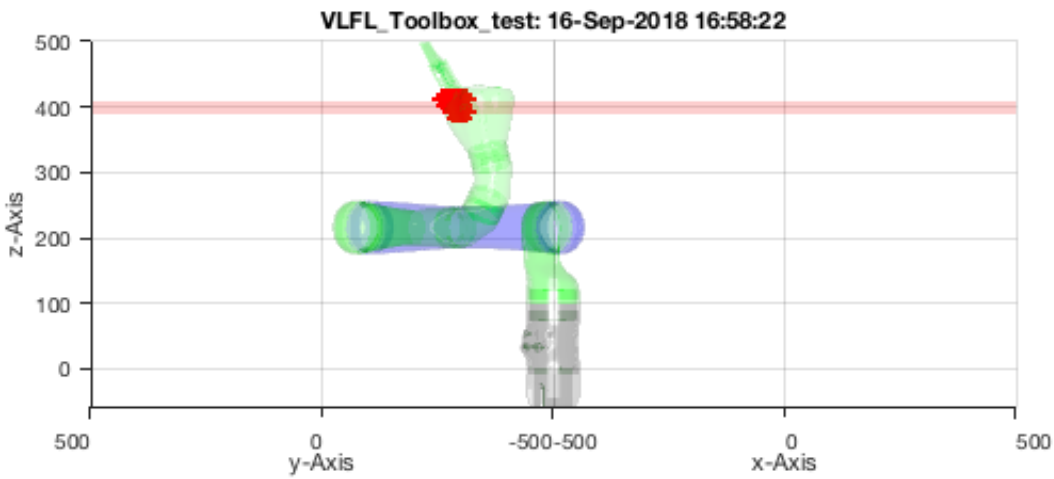
```
ans =

   36.9130   82.2649  389.9999
   36.9130   82.2649  389.9999
   -3.9110   82.2649  389.9999
   -3.9110   82.2649  389.9999
  -55.9248   82.2650  389.9999
  -55.9248   82.2650  389.9999
   36.9130   89.5443  410.0001
   36.9130   89.5443  410.0001
  -16.8148   89.5444  410.0001
  -16.8148   89.5444  410.0001
  -55.9248   89.5444  410.0001
  -55.9248   89.5444  410.0001
   36.9130  138.7881  410.0001
   36.9130  138.7881  410.0001
  -55.9248  138.7881  410.0001
  -55.9248  138.7881  410.0001
   36.9130  147.7560  389.9999
   36.9130  147.7560  389.9999
  -55.9248  147.7560  389.9999
  -55.9248  147.7560  389.9999
   36.9131  240.4606  389.9999
```

```
 36.9131  240.4606  389.9999
-36.7140  240.4606  389.9999
-36.7140  240.4606  389.9999
-55.9247  240.4607  389.9999
-55.9247  240.4607  389.9999
 36.9131  247.7400  410.0001
 36.9131  247.7400  410.0001
-49.6179  247.7401  410.0001
-49.6179  247.7401  410.0001
-55.9247  247.7401  410.0001
-55.9247  247.7401  410.0001
```



*The full test with the original geometry is much slower if the collision objects have more facets than those 12 of the simple box!

```
iscollofSG(SGTchain(JACO,[nan 0 pi pi]),A,true); view(-45,0)
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:58:24!
Executed 16-Sep-2018 16:58:26 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
pde_toolbox
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell

2017-07-07: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.0 required)
- 1. Loading Surface Data
- 2. Interactive Selection of the Area
- Cut the selected arm area
- Show only the selected
- Create the surface of vectors along x axis
- Cut the
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)

- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

I'd like to thank Gweni-Viani Alonso-Aruffo for her student internship in Spring 2017 in my lab at TU of Munich. She recorded the surface data and wrote Matlab fnctns for also creating SVG data for laser cutting of cloth to protect the skin from getting direct in contact to the 3D printed polymers.

```
% function VLFL_EXP36
```

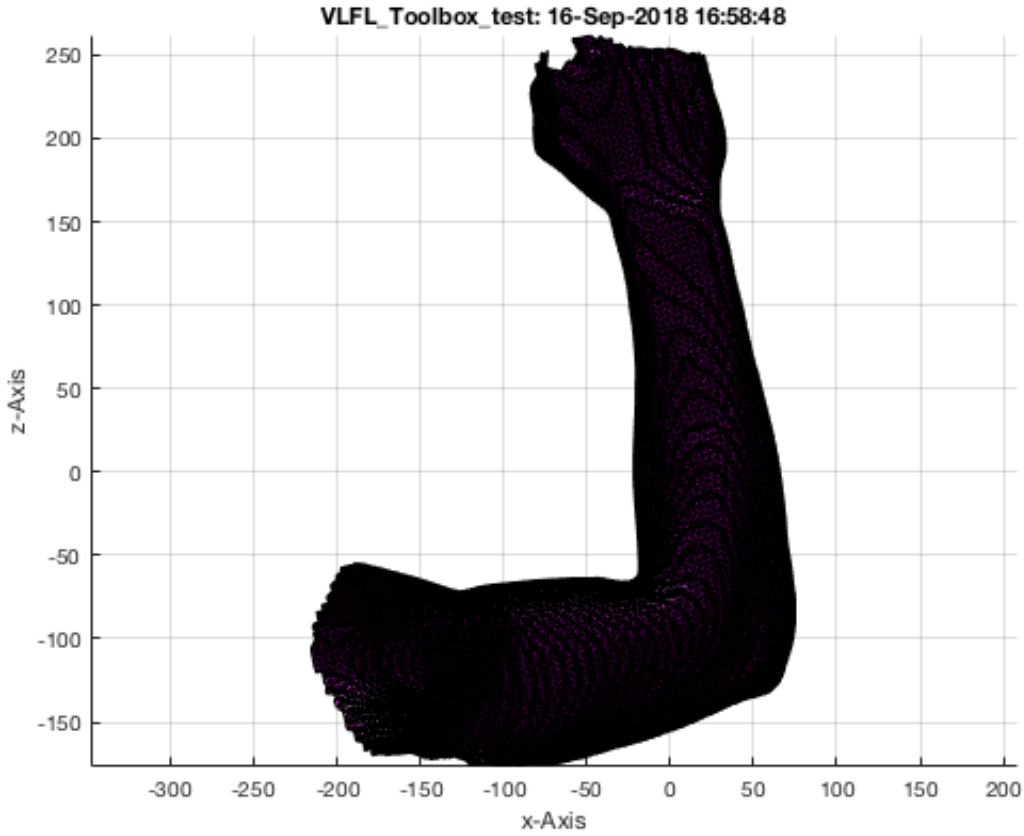
1. Loading Surface Data

2. Interactive Selection of the Area

```
load AAruffo_surf.mat  % use loadweb('AAruffo_surf.mat'); the first time

SGfigure(SG1); view(19,15); camlight;
beep; pause;
ginput(1); p1=select3d
ginput(1); p2=select3d
T=T2P(p1,p2-p1)
Ti=eye(4)/T
pn=VLtransT(p2',Ti); z=pn(3)

SGX=SGtransT(SG1,Ti); SGfigure(SGX); view(0,0)
```

```
p1 =

   -4.0180
  571.3952
```

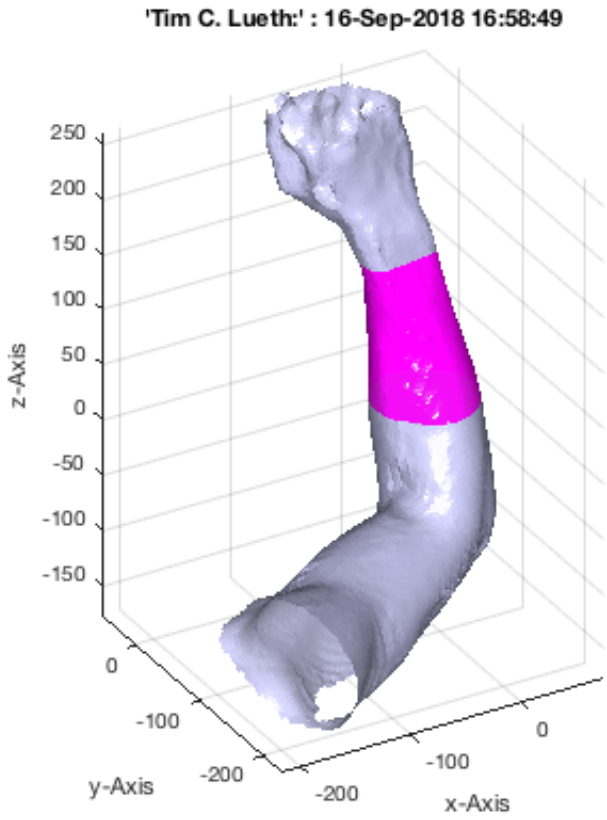
```
  -34.9557


p2 =

  -29.6720
  583.8052
   90.4696


T =

   -0.9799         0   -0.1995   -4.0180
   -0.0196   -0.9951    0.0965  571.3952
   -0.1985    0.0985    0.9751  -34.9557
         0         0         0    1.0000


Ti =

   -0.9799   -0.0196   -0.1985    0.3460
    0.0000   -0.9951    0.0985  572.0605
   -0.1995    0.0965    0.9751  -21.8452
         0         0         0    1.0000


z =

  128.6221
```



Cut the selected arm area

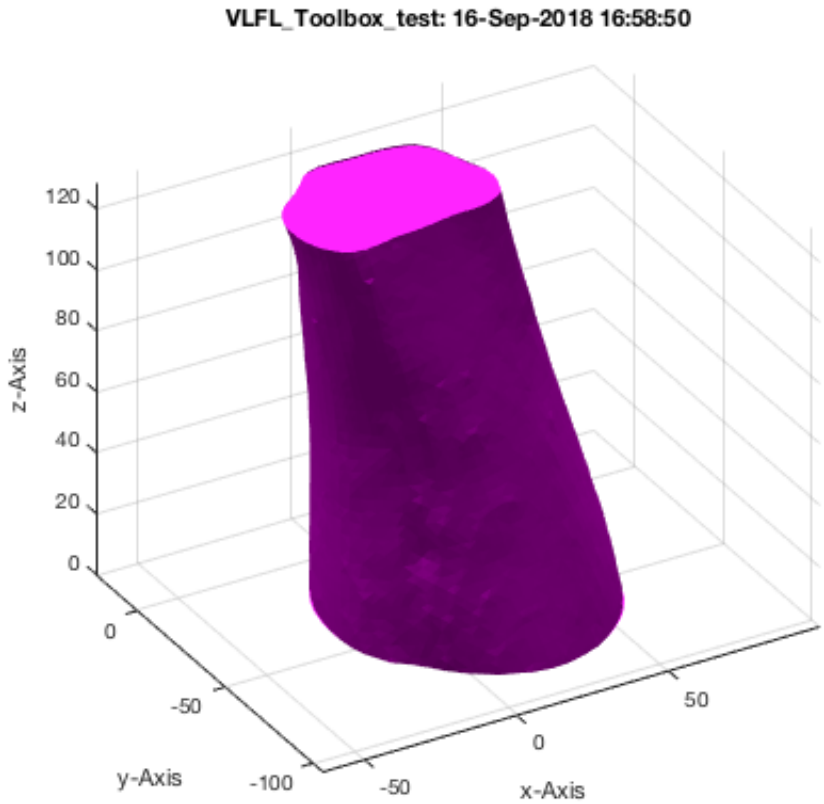
```
pn=VLtransT(p2',Ti); z=pn(3)
SGcut(SGX,[0 z]);
[SGout,SGin]=SGcut(SGX,[0 z]);
```

```
z =

  128.6221
```



Show only the selected

```
SGfigure(SGin); view(-30,30); VLFLplotlight(1,1);
```

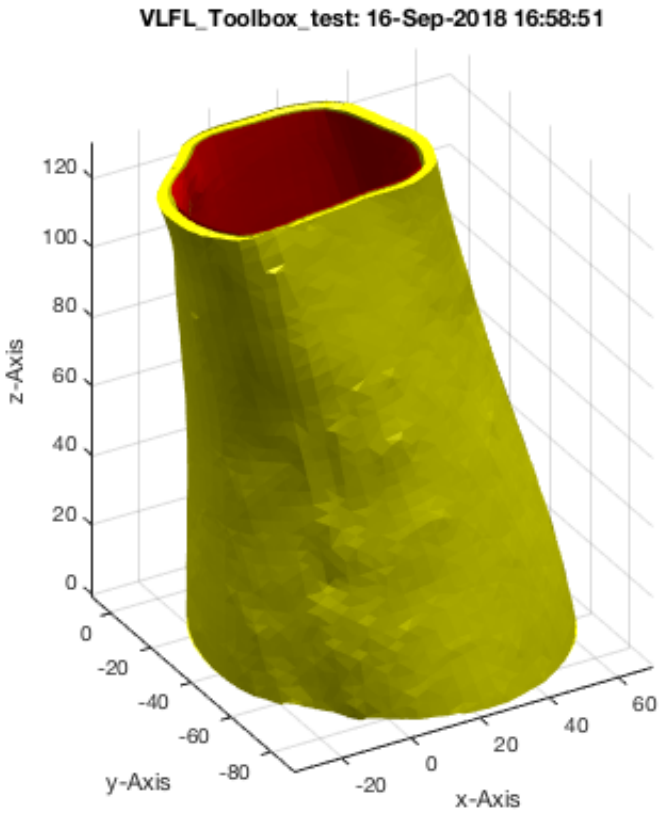


Create the surface of vectors along x axis

```
[FIL,k,FNL]=surfacesofSG(SGin);
[~,d]=VLnorm(FNL-[1 0 0]); dmin=min(d)
fi=find(d(d==dmin))
s=FIL(fi)
[S.VL,~,S.FL]=VLFLselect(SGin.VL,SGin.FL(FIL==s,:))
```

```
dmin =

    0.0210


fi =

     1


s =

     1


S =

  struct with fields:

    VL: [3014×3 double]
    FL: [5713×3 double]


S =

  struct with fields:

    VL: [3014×3 double]
    FL: [5713×3 double]
```

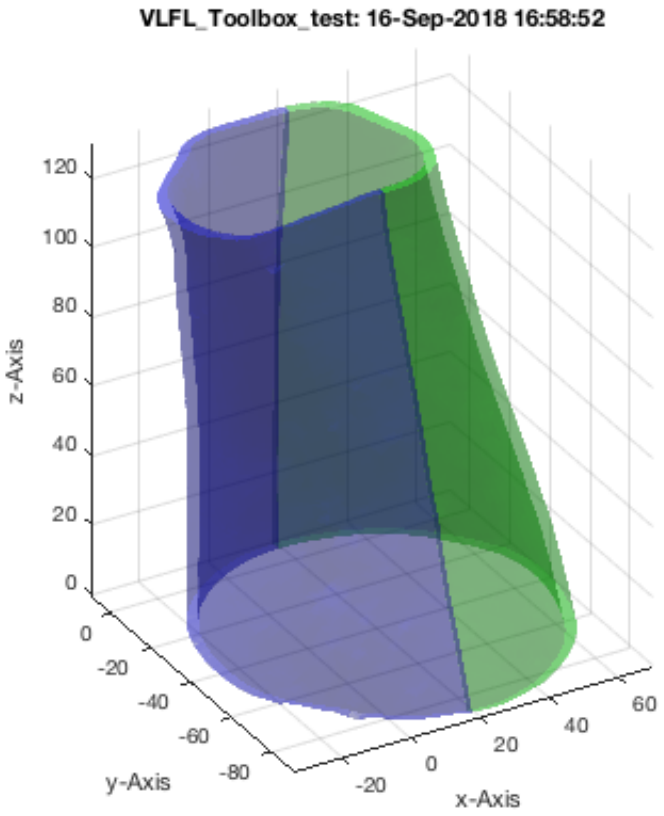
```
SGshell=SGofSurface(S.VL,S.FL,3,1);
SGofSurface(S.VL,S.FL,3,1); view(-30,30); VLFLplotlight(1,1);
```


VLFL_Toolbox_test: 16-Sep-2018 16:58:51
z-Axis
120
100
80
60
40
20
0
y-Axis
0
-20
-40
-60
-80
x-Axis
-20
0
20
40
60

Cut the

```
SGpuzzlecut3D(SGshell,[0.5 1 1]); VLFLplotlight(1,0.3);
SGshell=SGpuzzlecut3D(SGshell,[0.5 1 1]); VLFLplotlight(1,0.3);
```

```
50% 100%
50% 100%
```




```
SGwriteMultipleSTL(SGshell);
```

```
SGwritemultipleSTL: Writing 2 STL files in /Users/timlueth/Desktop/Toolbox_test/EXP-2018-09
-16/
```

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:58:53!
Executed 16-Sep-2018 16:58:55 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
pde_toolbox
```

```
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 37: Dimensioning of STL Files and Surface Data

2017-07-24: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.0 required)
- 1. Basic functions for dimensioning
- 2. Classifying 3D Contours
- 3. Finding Surfaces and Contours for Dimensioning
- 4. Interactive specifiying faces and coordinate systems
- 5. Dimensioning of border of surfaces: SGdimensioning
- 6. Creating of standard dimensioning using view angles: SGdimensioning
- 7. Creating of standard dimensioning using view angles and cross cuts
- 8. Using frames for dimensioning

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

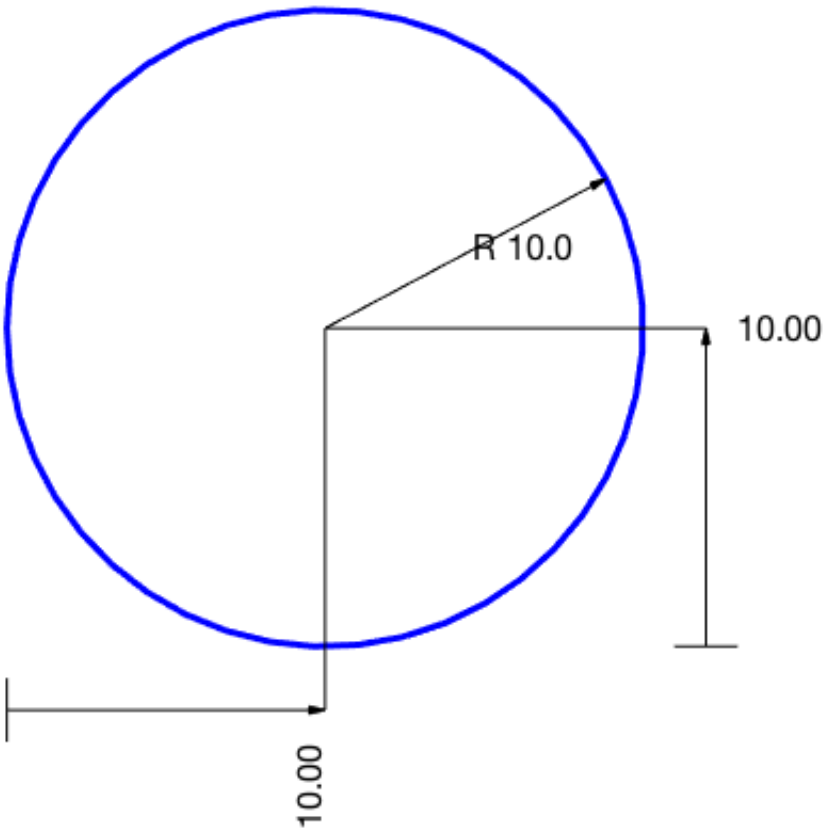
Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

If you use an STL file from the third page, then certain dimensions have been used which have an influence on a constructions. In this tutorial you will find the function to analyze STL files and to draw technical drawings for these surfaces.

1. Basic functions for dimensioning

PLdimensioning creates complete drawings for a single Point list. it should be used as a drawing function

```
PL=PLcircle(10);
PLdimensioning(PL);
```

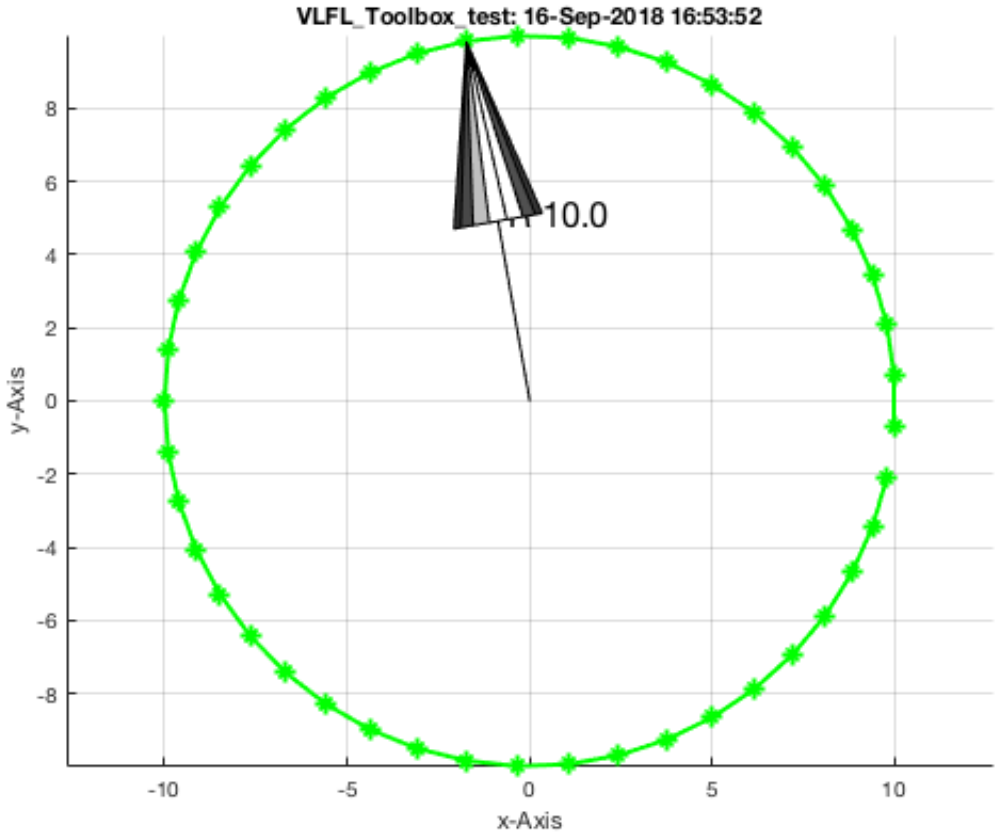


CVLdimclassifier is an auxiliary function that creates results for PLdimensioning It should be used for calculating features and supports CVL in 3D too. It returns points that are not part of a circe/ellipse and a list of circles and a list which vertices belong to circles and the normal vectors for the circles

```
CVLdimclassifier(PL)
[DVL,RL,RIL,Rnv]=CVLdimclassifier(PL)
```

```
ans =

  0×3 empty double matrix


DVL =

  0×3 empty double matrix


RL =

  Columns 1 through 7

         0         0         0   45.0000  360.0000   10.0000   -9.1355

  Columns 8 through 9

    4.0674         0
```

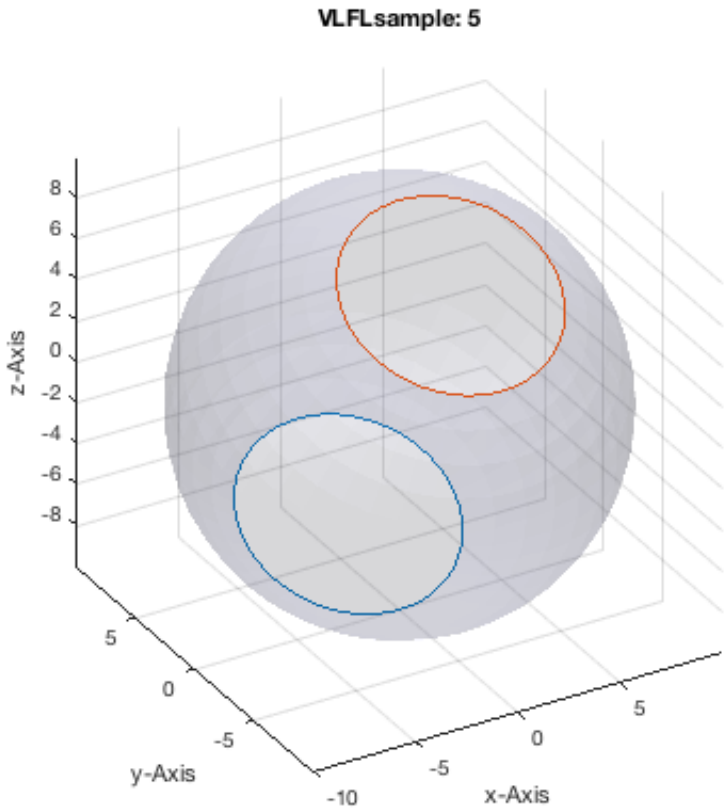
```
RIL =

     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1
     1


Rnv =

     0     0    -1
```



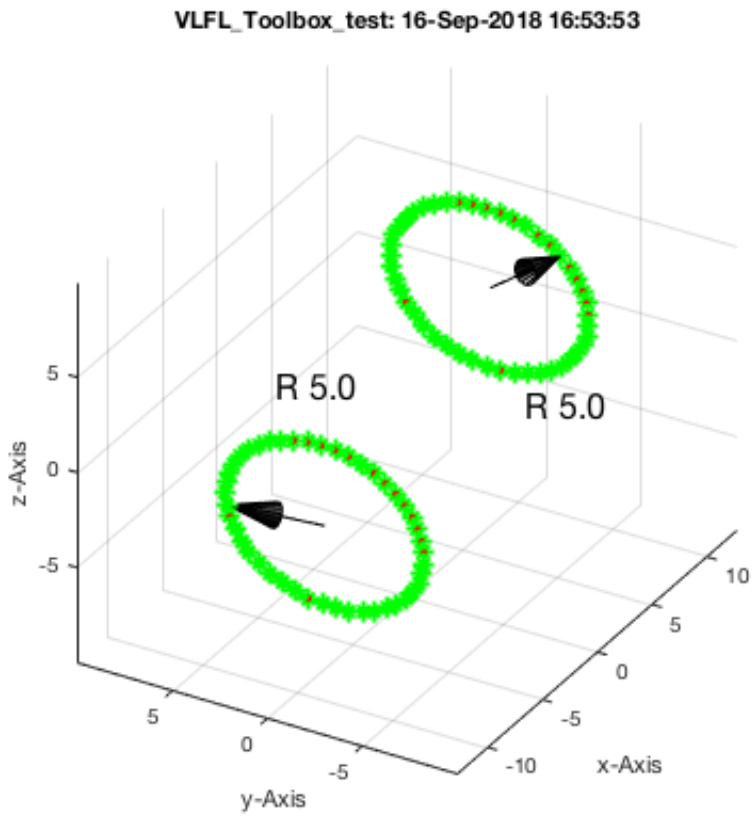
2. Classifying 3D Contours

```
% *CVLdimclassifier* is ablt to classify several indepenent contours in
3D space as CVL.
```

```
VLFLsample(5); VLFLplotlight (1,0.2);
[~,~,~,CVL]=VLFLsample(5);
CVLplot(CVL,'-');
```




```
CVLdimclassifier(CVL); view(-60,30);
```

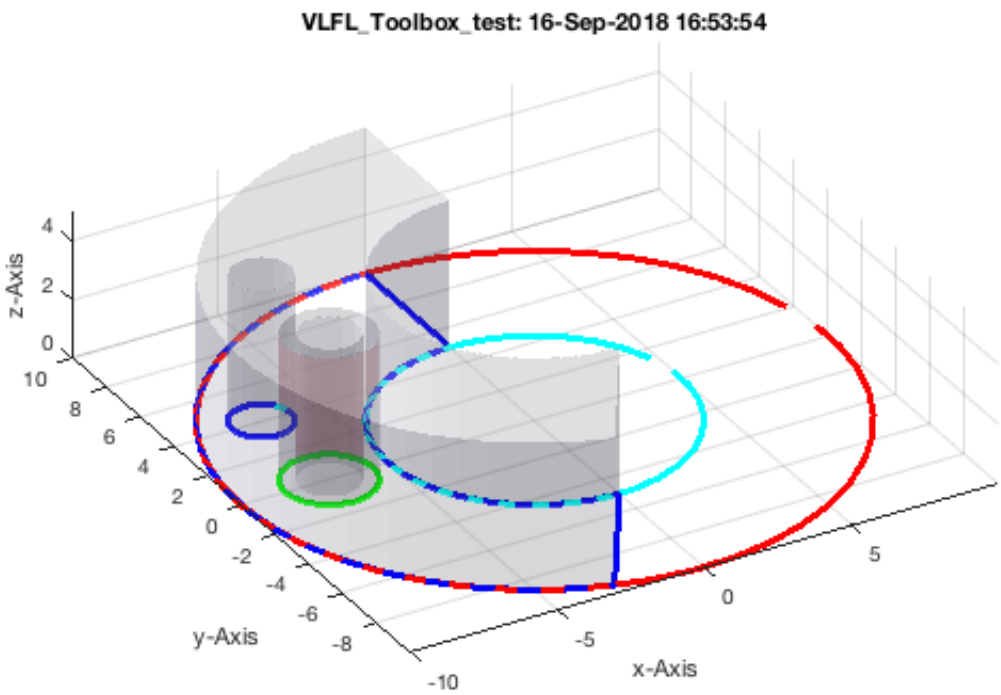


3. Finding Surfaces and Contours for Dimensioning

There are not several functions that help to define surfaces for dimensioning similar to featureedges we see feature surfaces to separate surfaces if a solid

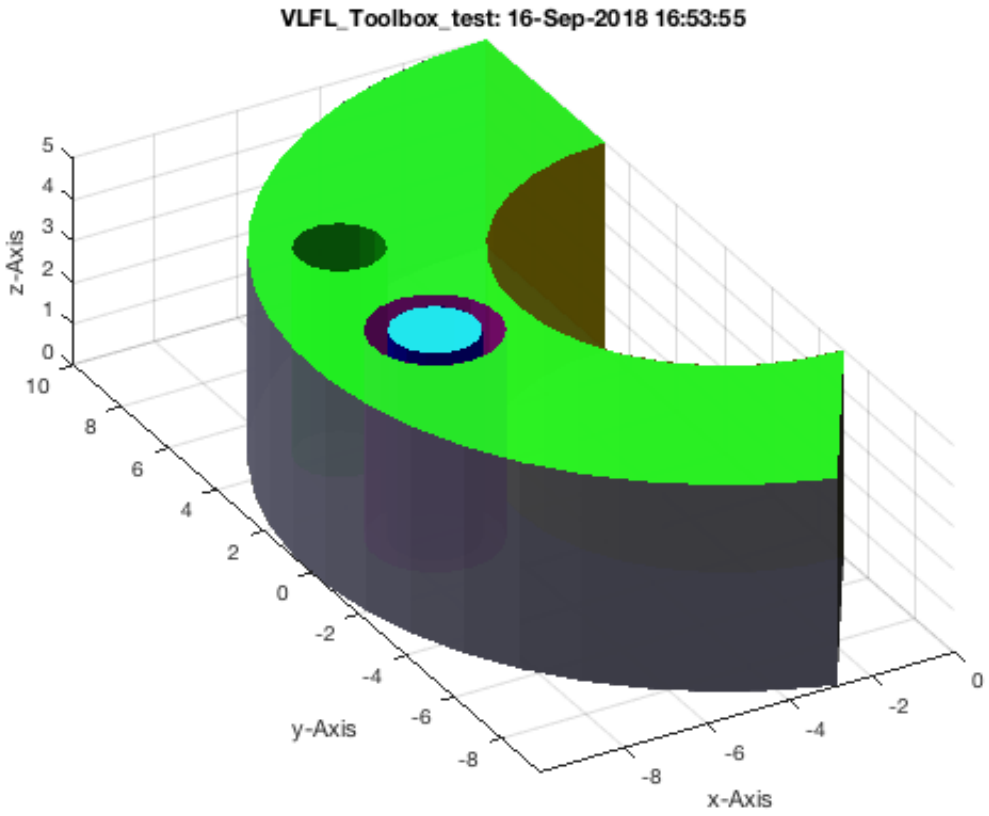
- **TR3mountingfaces** - finds connected surfaces starting from one or more faces
- **MLofSG** is a simliar function for a complete solid
- **surfacesofSG** - generates features surfaces of ONE closed solid
- **TR3neighborsAngle** and **neighborsAngleSurface** - find feature surfaces
- **FSofSG** supports also cells of solids

```
 TR3mountingfaces(SGsample(25),1); % facets, normals, neigbors, radial list, CVL of ONE sur
face
```

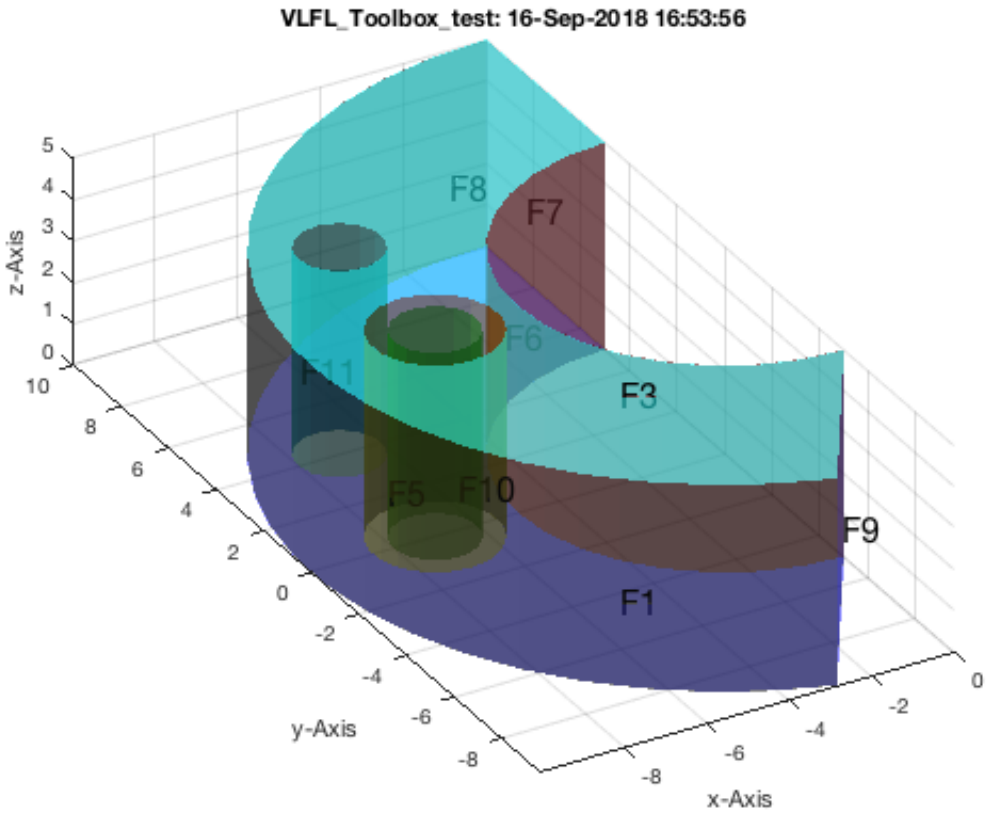



```
surfacesofSG(SGsample(25)); % facet index, normals, angles, neigbors, area
```

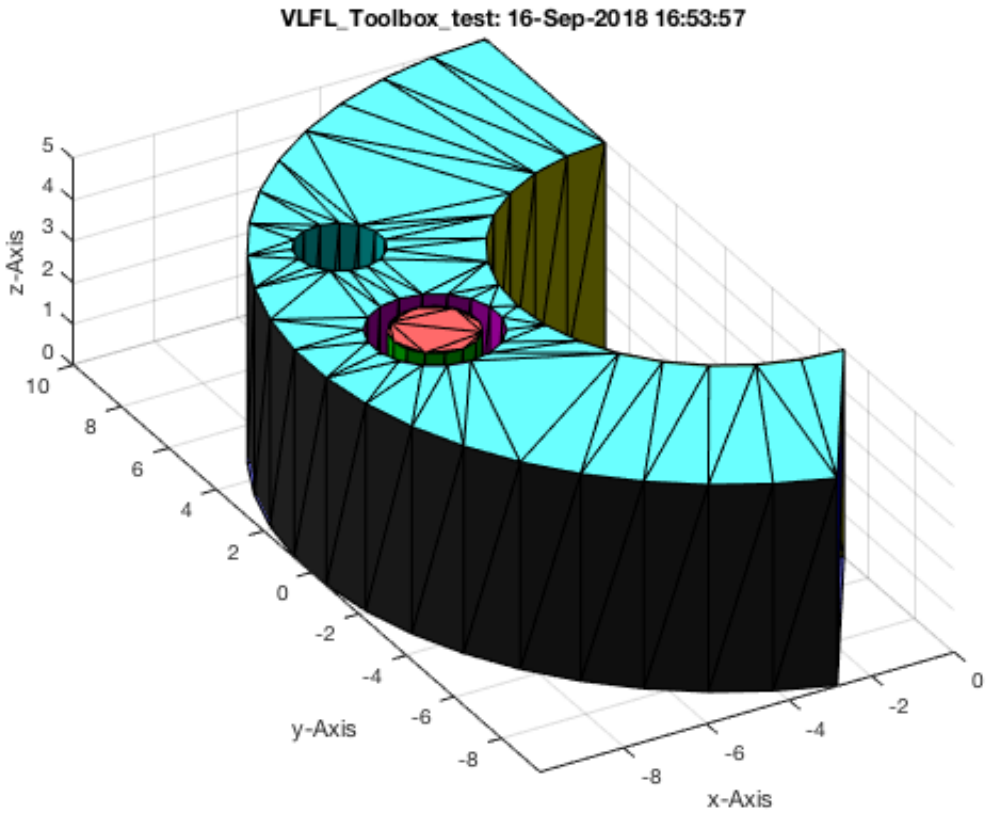
```
11 Feature Surfaces found! Only the largest 99% (0..143mm^2), i.e. 11 of 11 are shown.
```




```
FSofSG(SGsample(25)); % surface index list
```




```
MLofSG(SGsample(25));
```

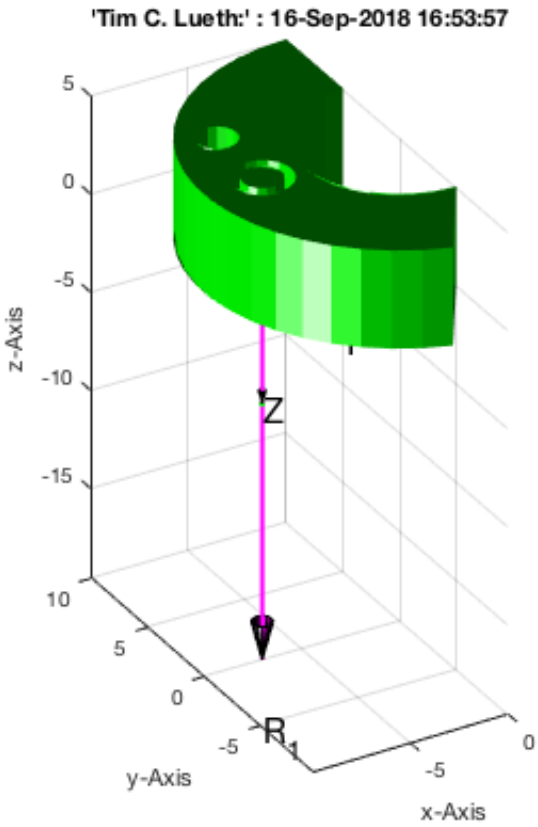


4. Interactive specifiying faces and coordinate systems

We already used ina nearlier tutorial(VLFL_EXP11) the function SGTui to specify coordinate system for planar surfaces or edges. By using the third parameter if SGTui, it is possible to make a feature surface search. A common value is 1 rad ~ 60 degree. The function is able to detect radial structures using CVLdimclassifier and allow to address other coordinate systems too.

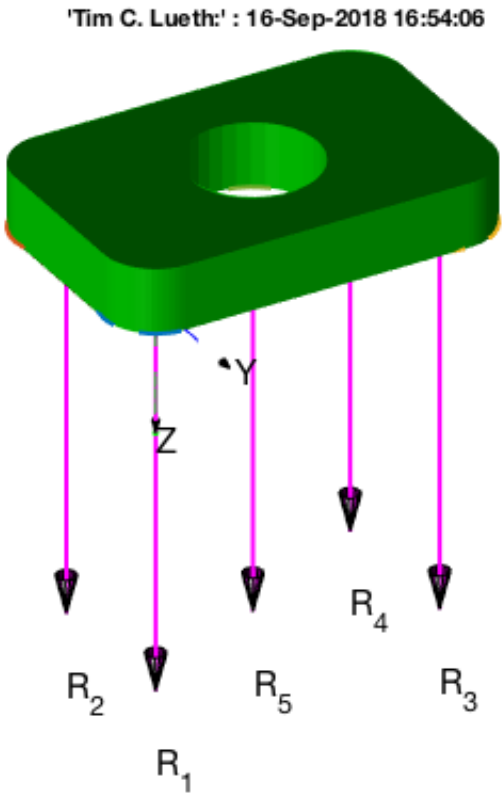
```
SGTui(SGsample(25),'Frame',1)
```

```
ans =

  struct with fields:

       VL: [174×3 double]
       FL: [348×3 double]
    Tname: {'Frame'}
        T: {[4×4 double]}
     TFiL: {[13×1 double]}
     TFoL: {[]}
```

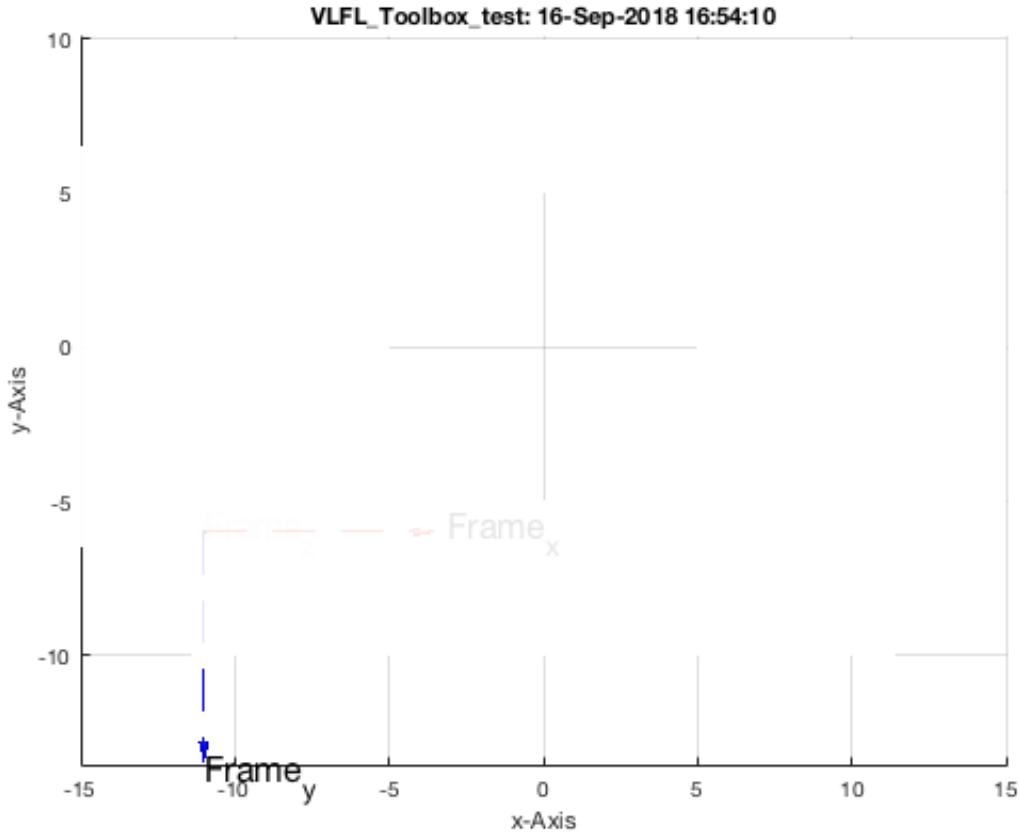



```
SG=SGTui(SGsample(27),'Frame',1,'R1')
```

```
SG =

  struct with fields:

      VL: [200×3 double]
      FL: [400×3 double]
   Tname: {'Frame'}
       T: {[4×4 double]}
    TFiL: {[100×1 double]}
    TFoL: {[]}
```

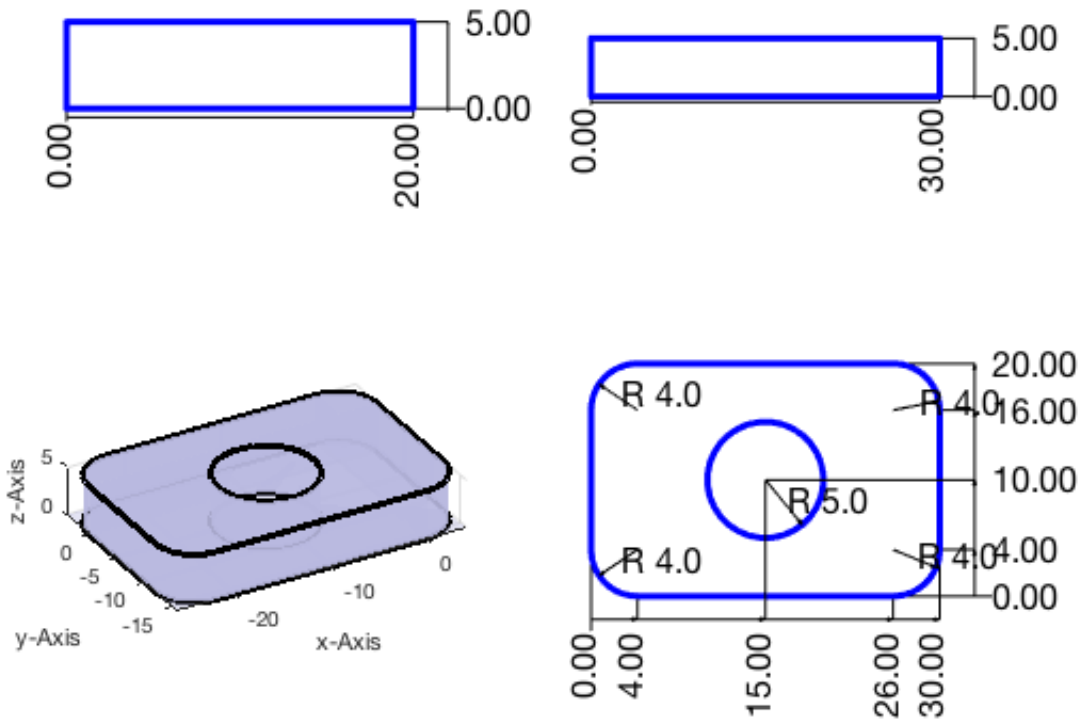



```
SGfigure; SGTplot(SG);
```



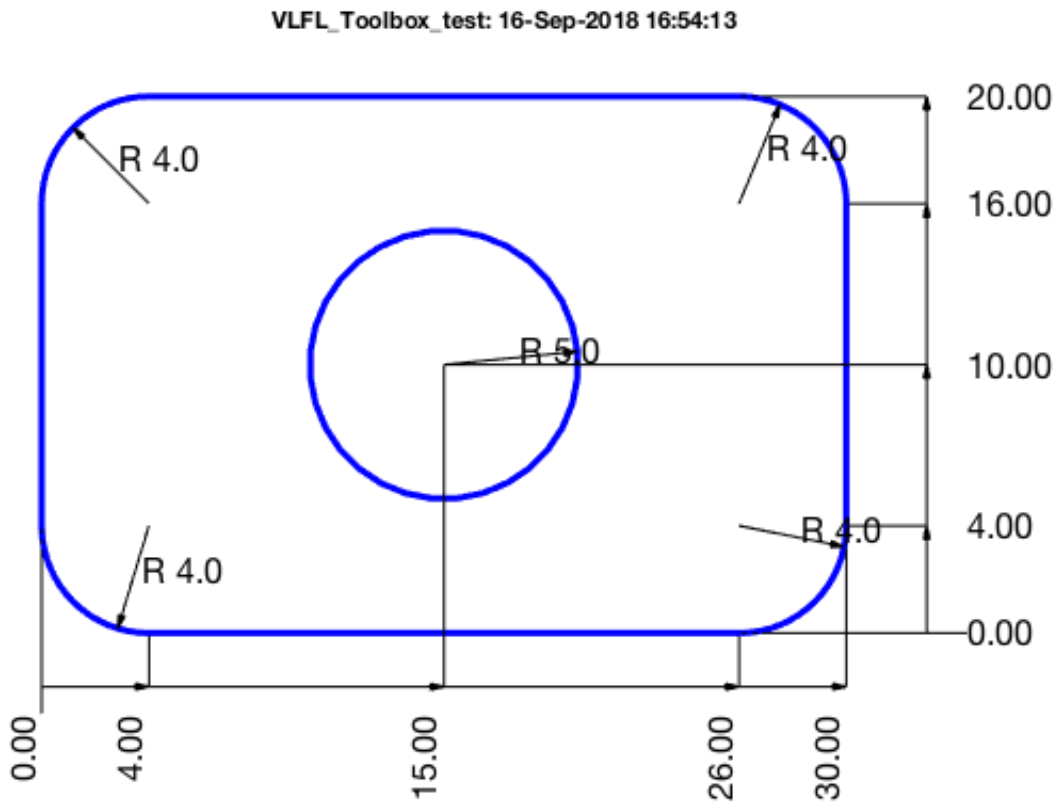
5. Dimensioning of border of surfaces: SGdimensioning

```
SGTdimensioning(SG,'Frame');
```

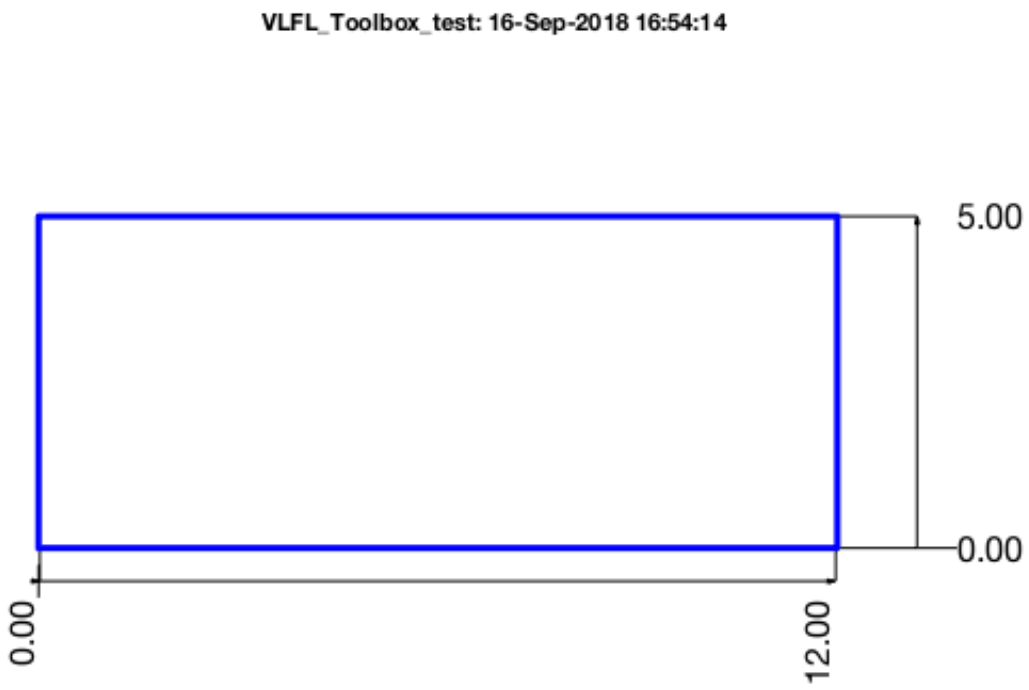


6. Creating of standard dimensioning using view angles: SGdimensioning

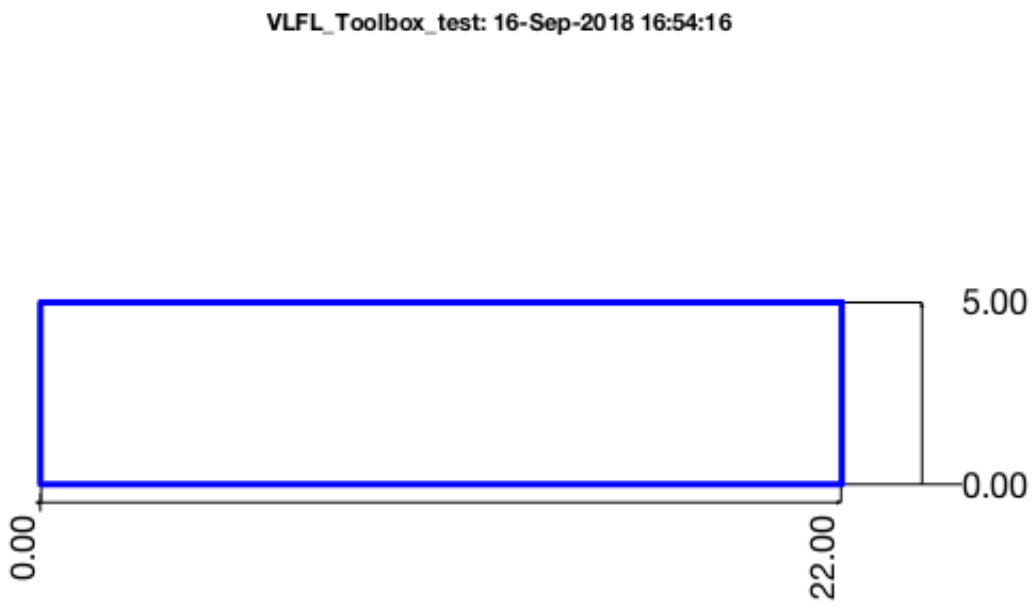
```
SGdimensioning(SG,0,90);
```




```
SGdimensioning(SG,90,0);
```

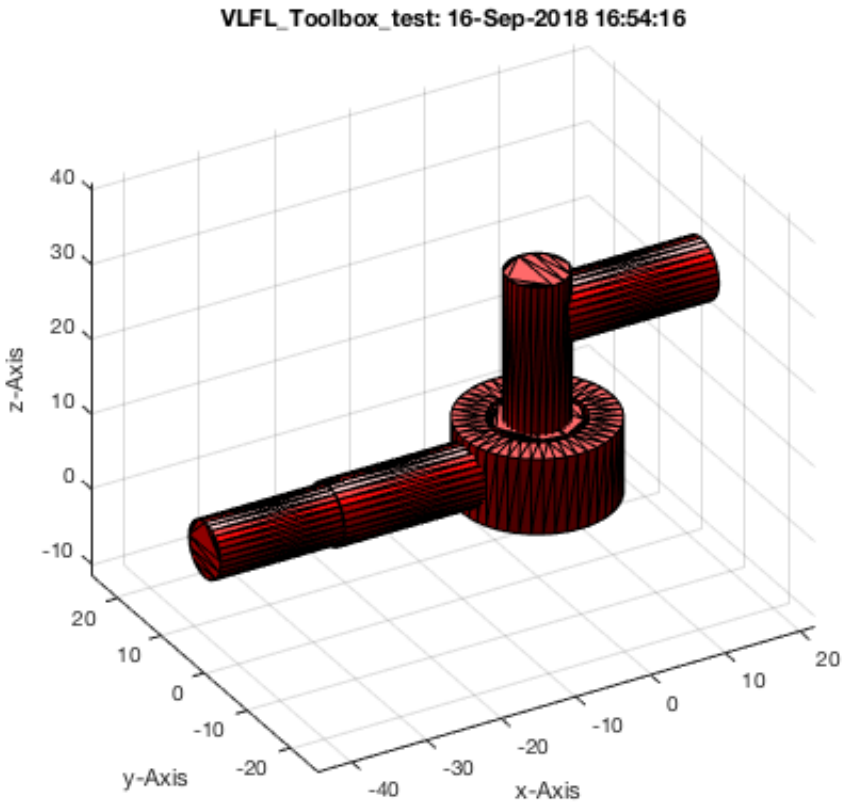



```
SGdimensioning(SG,0,0);
```

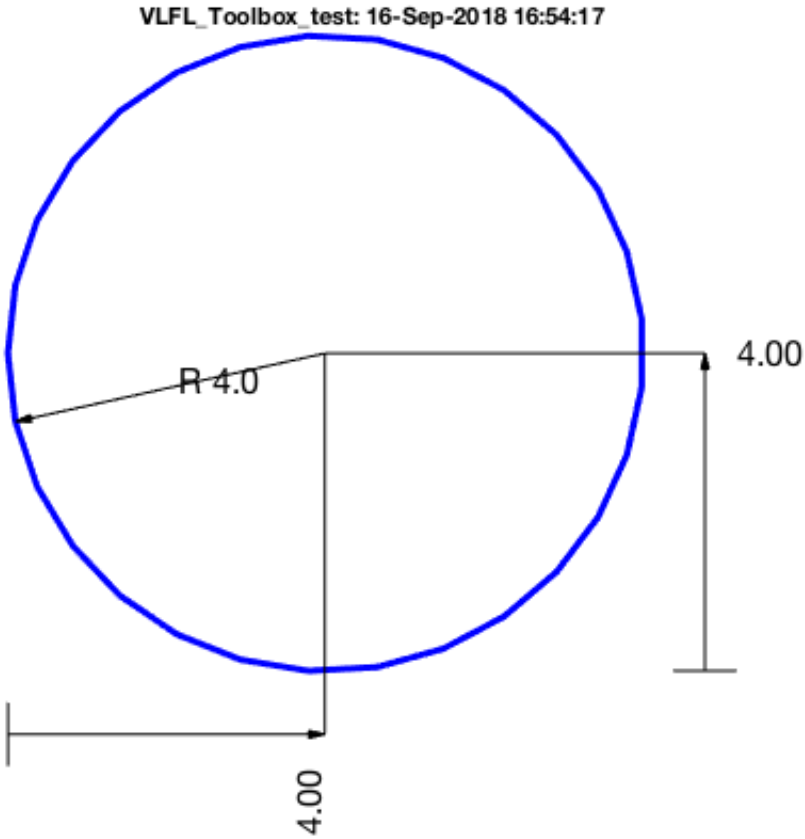


7. Creating of standard dimensioning using view angles and cross cuts

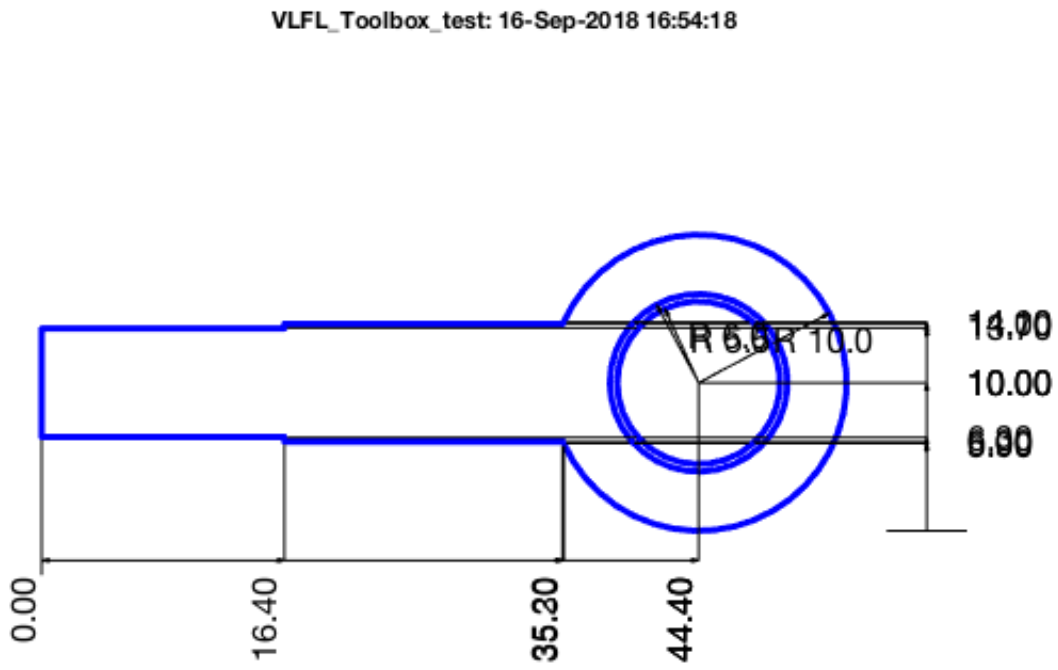
```
SG=SGsample(17); SGfigure; SGplot(SG); view(-30,30);
```



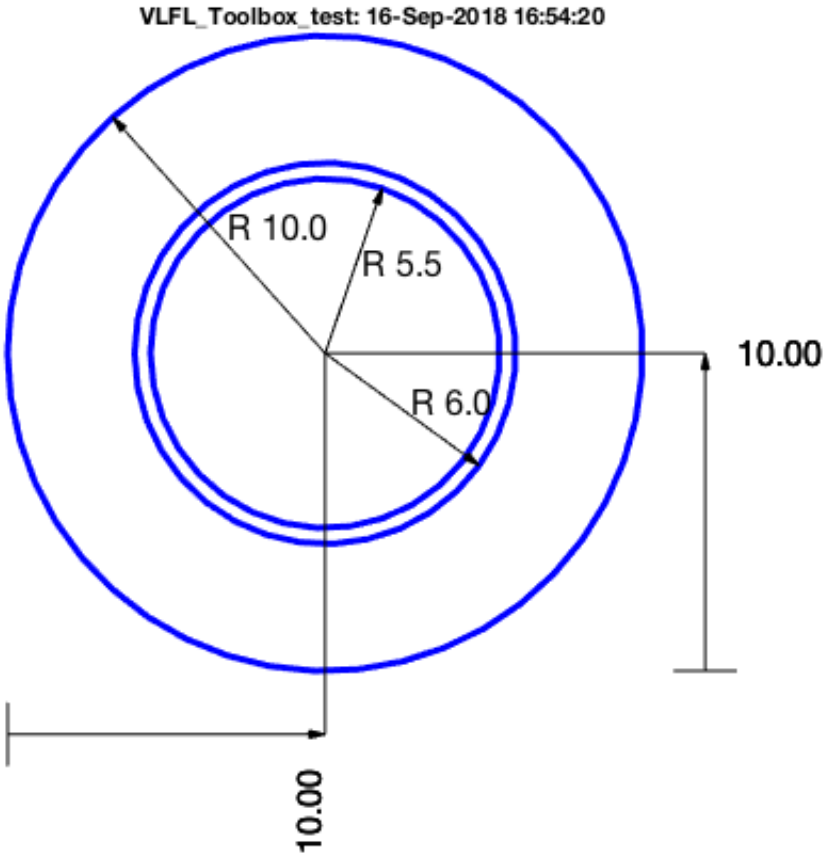

```
SGdimensioning(SG,0,90);
```



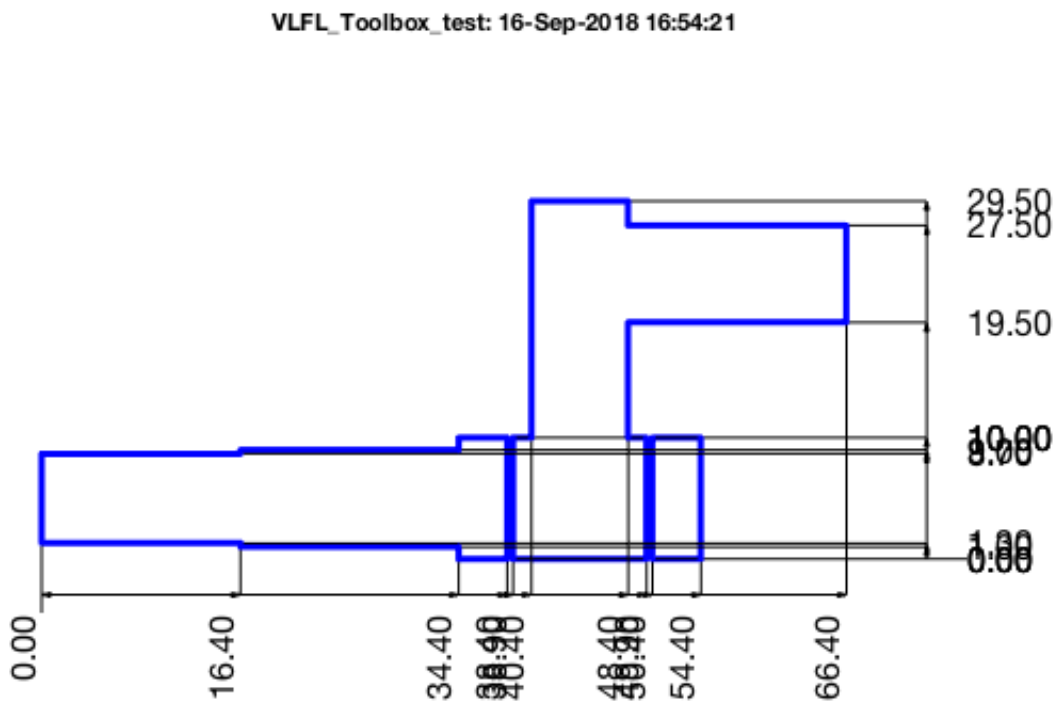

```
SGdimensioning(SG,0,90,[0 0 5]);
```




```
SGdimensioning(SG,0,90,[0 0 +10]);
```



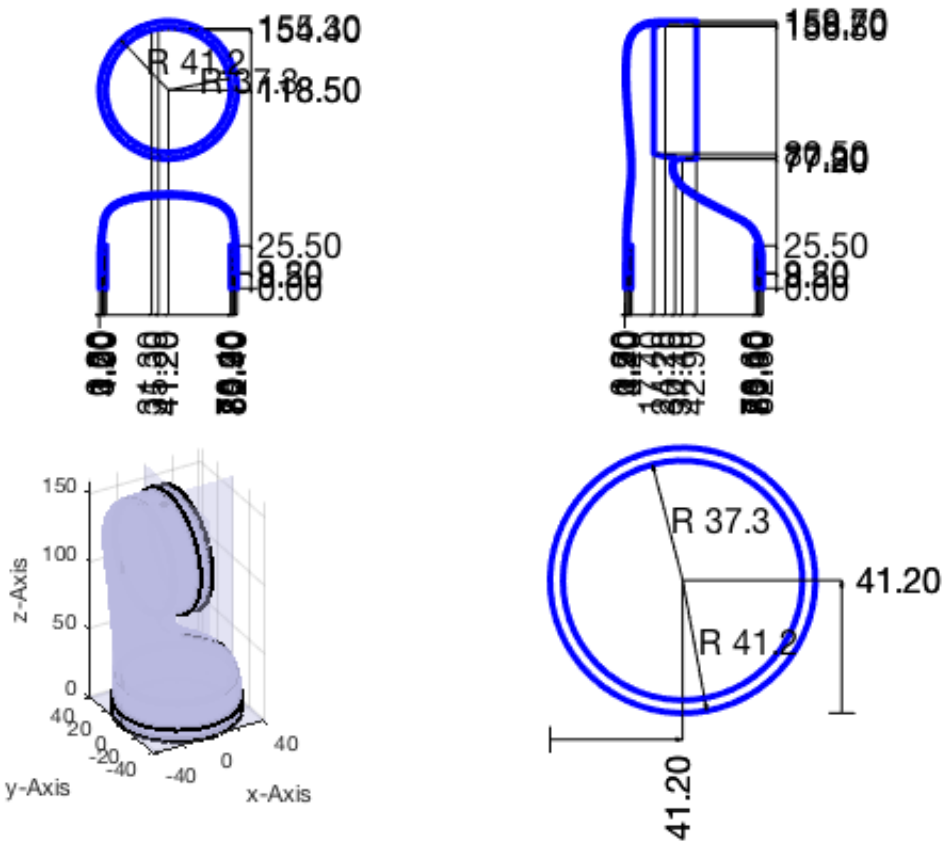

```
SGdimensioning(SG,0,0,[0 0 0]);
```



8. Using frames for dimensioning

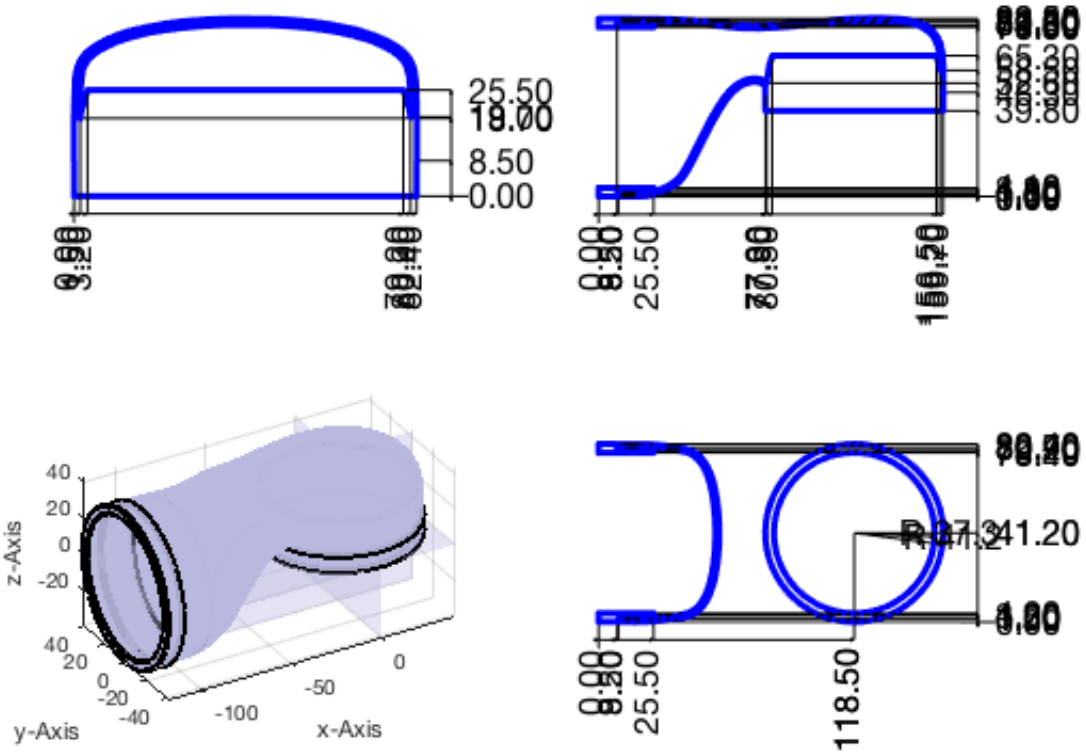
```
load JACO_robot.mat
SGTdimensioning(JC1,'B');
```

```
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
```




```
SGTdimensioning(JC1,'F');
```

```
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
Warning: Duplicate edge constraints have been detected and removed.
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
Warning: The triangulation is empty - the points may be collinear.
Warning: Matrix is singular, close to singular or badly scaled. Results may be
inaccurate. RCOND = NaN.
Warning: Matrix is singular, close to singular or badly scaled. Results may be
inaccurate. RCOND = NaN.
```



Published with MATLAB® R2018a

Tutorial 38: Some more solid geometry modelling function

2017-07-24: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.0 required)
- 1. Some elements of medical equipemnt in the operating room
- or select a c-arm device
- 2. Creating solids as links with spheres at the end
- 3. Creating Solids by connecting two CPLs with enclosed contours
- 4. Creating Solids by connecting two planar CPLS of different strucure
- 5. Creating branches between two contour
- 6. Chamfer the edges of a solid
- 7. Creating a drawing temmplate
- 8. Separating an solid into peaces
- 9. create a solid surface from an open surface
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

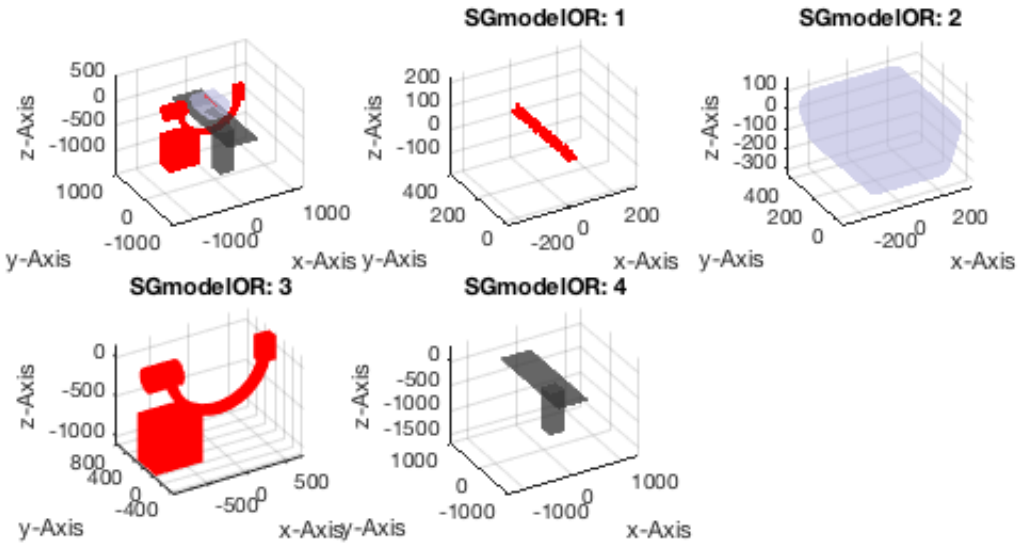
- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids

- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function

Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

1. Some elements of medical equipemnt in the operating room

```
SGmodelOR;
```

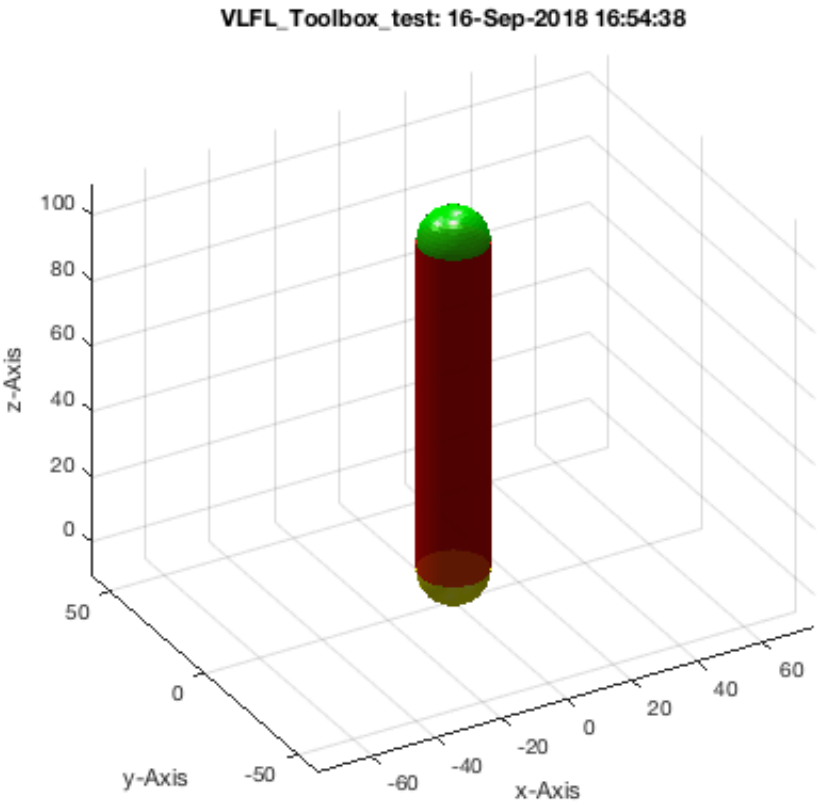


or select a c-arm device

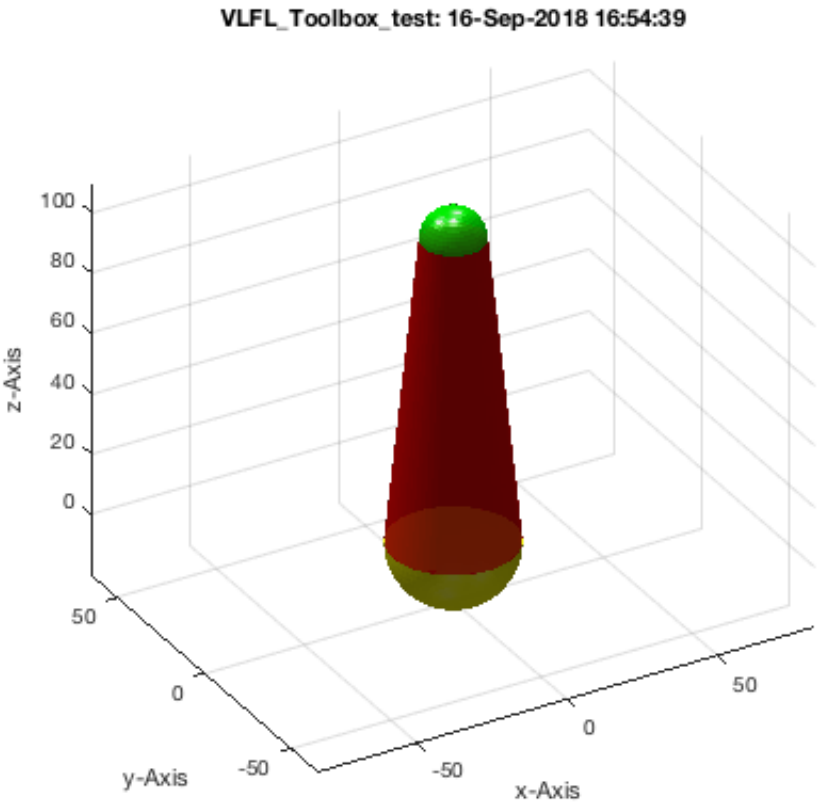
SGmodelOR(3)

2. Creating solids as links with spheres at the end

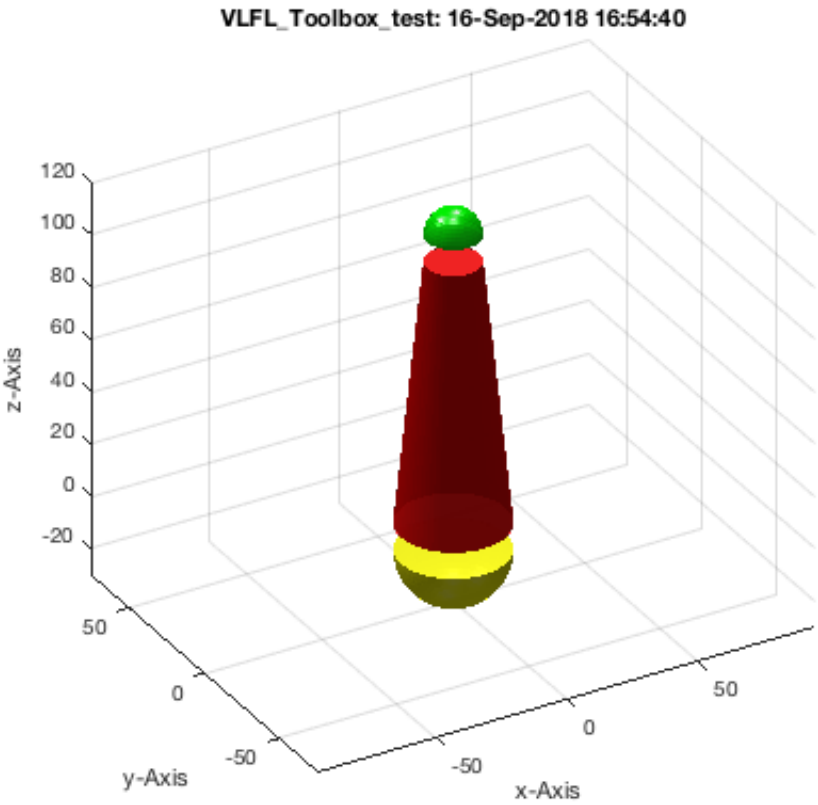
```
SGspherelink (100,10);
```



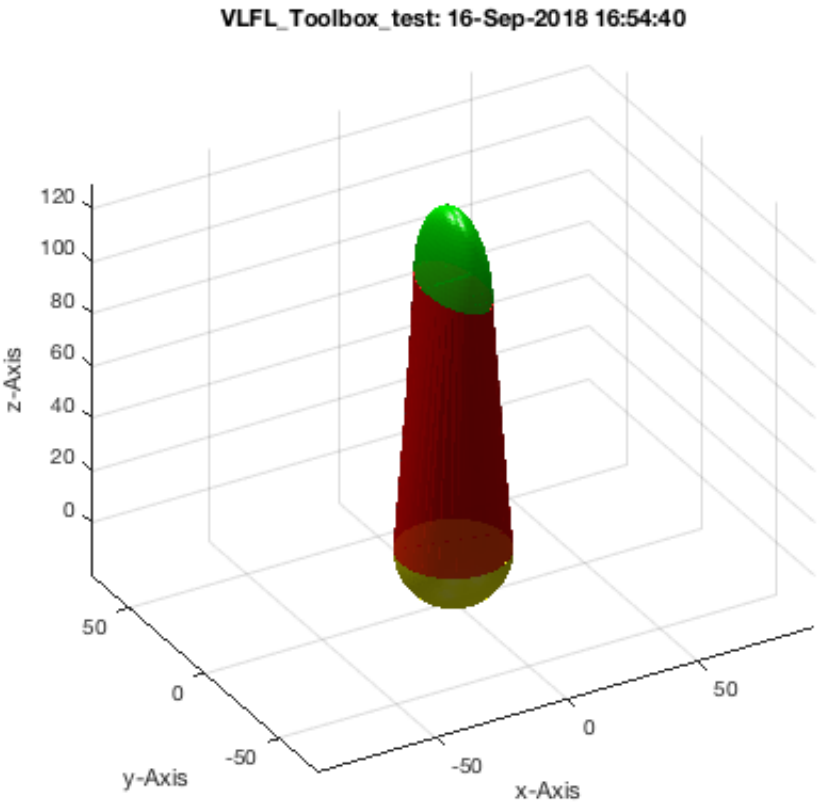

```
SGspherelink (100,10,20);
```



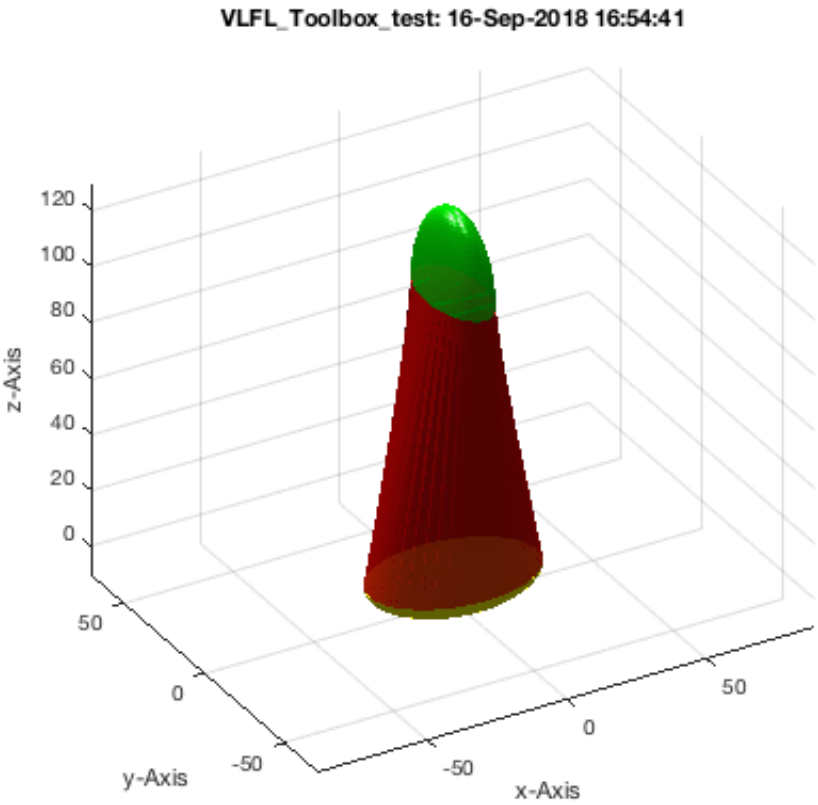

```
SGspherelink (100,10,20,-10);
```




```
SGspherelink (100,[10,20,30],20);
```

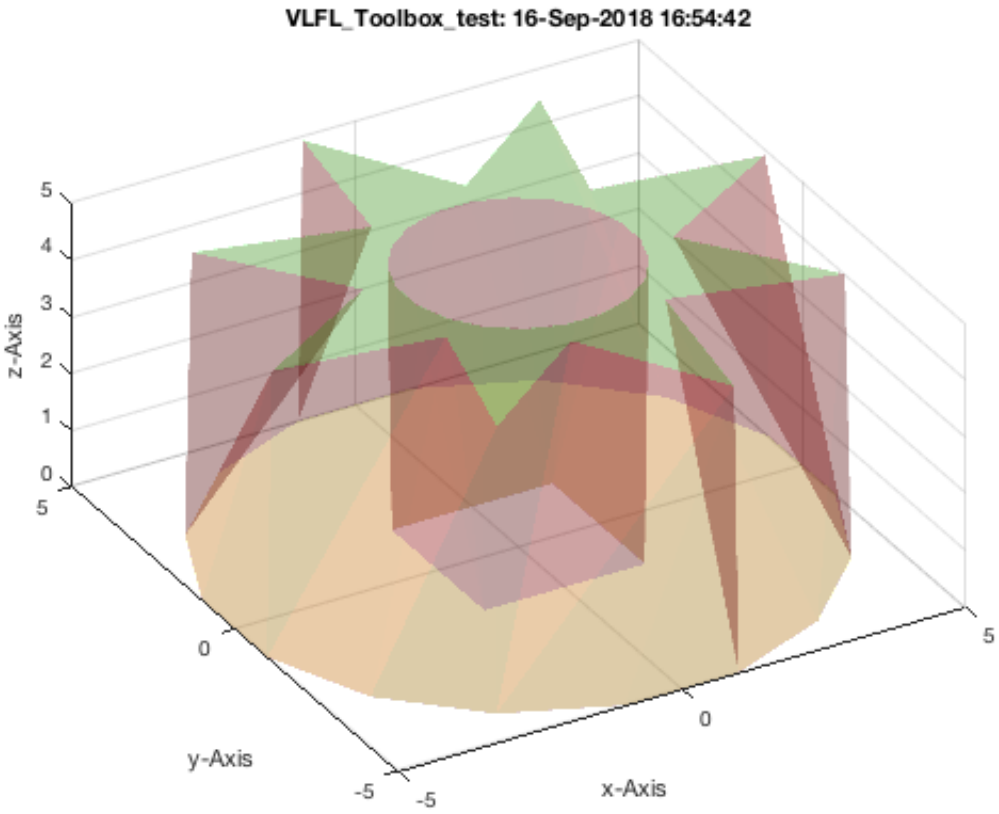



```
SGspherelink (100,[10,20,30],[30,20,10]);
```



3. Creating Solids by connecting two CPLs with enclosed contours

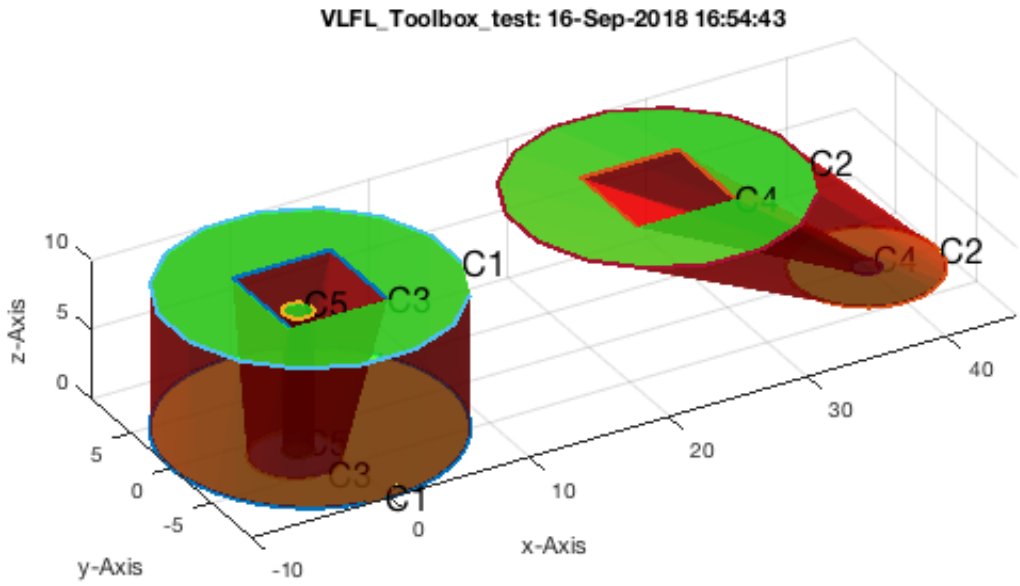
```
CPA=[PLcircle(5.1,16);NaN NaN; PLcircle(2,4)];
CPB=[PLstar(5,16,[],[],[],0.5);NaN NaN; PLcircle(2)];
SGof2CPLsz(CPA,CPB,5); VLFLplotlight(1,0.2);
```



4. Creating Solids by connecting two planar CPLS of different strucure

```
SGof2CPLzheurist(CPLsample(26),CPLsample(27),10)
```

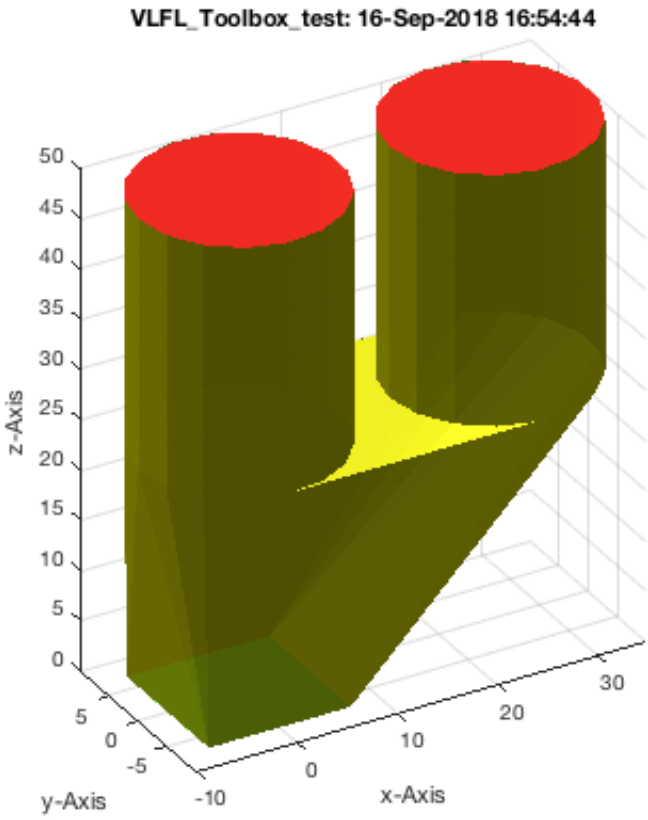
```
ans =

  struct with fields:

       VL: [187×3 double]
       FL: [370×3 double]
      col: 'w'
    alpha: 0.9000
```



5. Creating branches between two contour

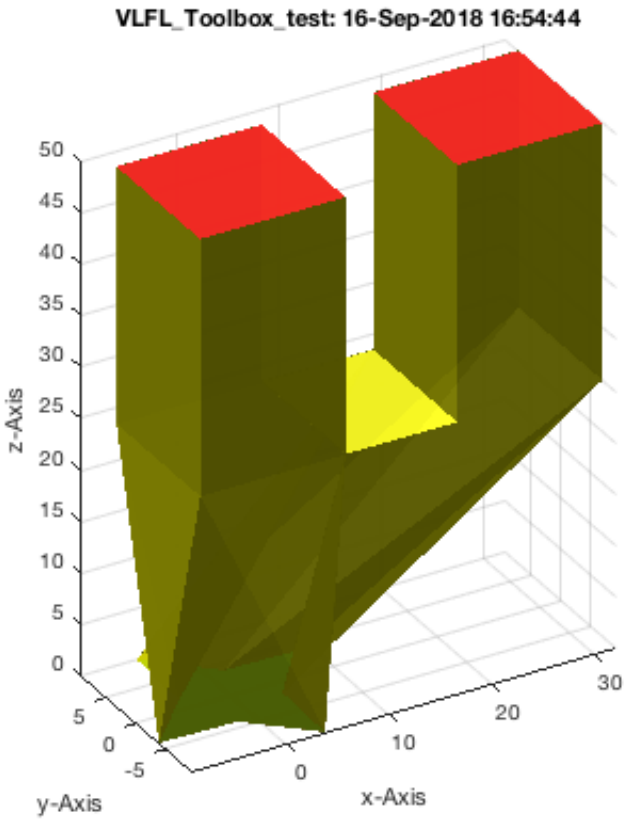
```
SGof2CPLzbranch(CPLsample(2), CPLsample(9),50)
```

```
ans =

  struct with fields:

    VL: [72×3 double]
    FL: [140×3 double]
```




```
SGof2CPLzbranch(CPLsample(6), CPLsample(10),50)
```

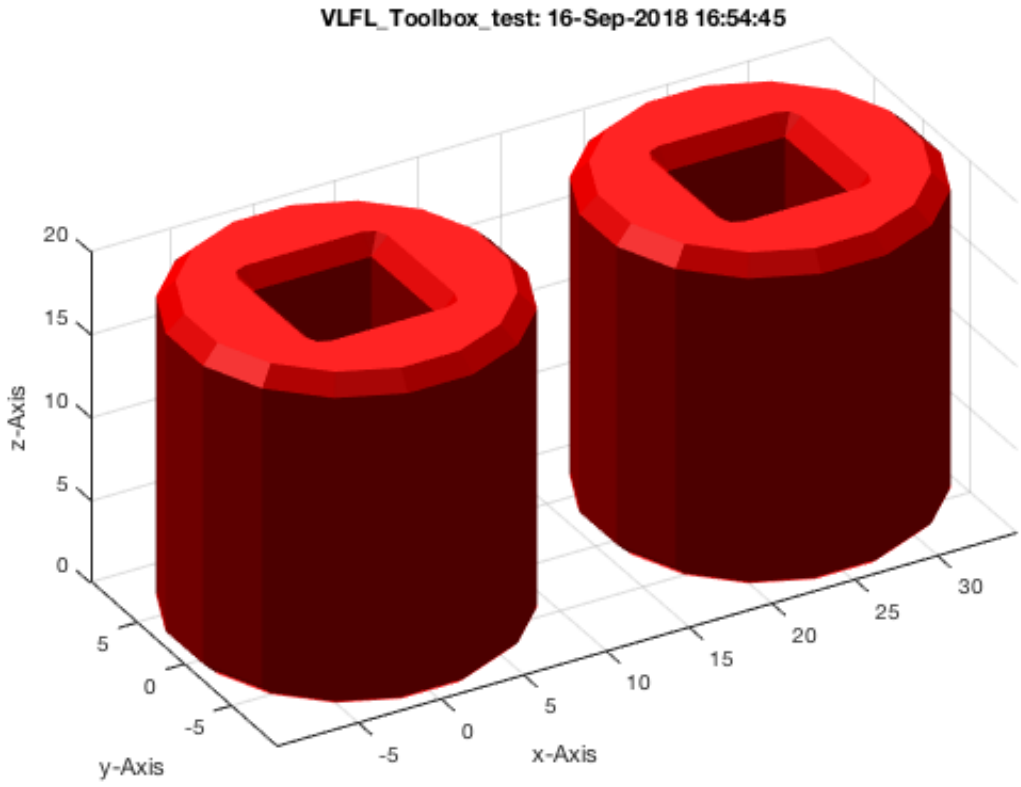
```
Warning: Edge constraints have been split by a coincident point.

ans =

  struct with fields:

    VL: [24×3 double]
    FL: [44×3 double]
```



6. Chamfer the edges of a solid

```
SGofCPLzchamfer(CPLsample(12),20,1)
```

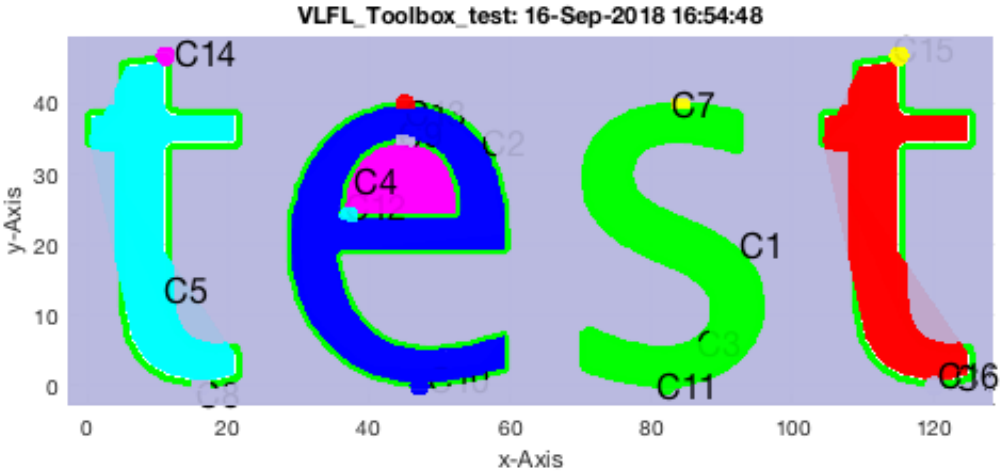
```
ans =

  struct with fields:

    VL: [192×3 double]
    FL: [384×3 double]
```



7. Creating a drawing temmplate

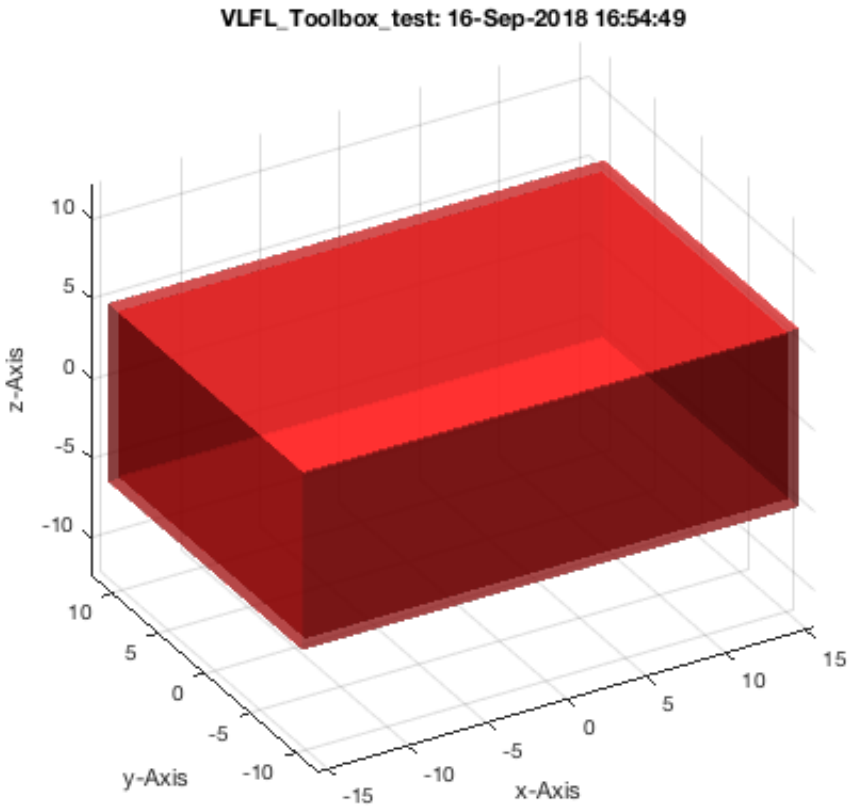
```
SGdrawingtemplateofCPL(CPLoftext('test'),'','','','','',true)
```

```
Drawing template is separated

ans =

  struct with fields:

    VL: [4172×3 double]
    FL: [8300×3 double]
```



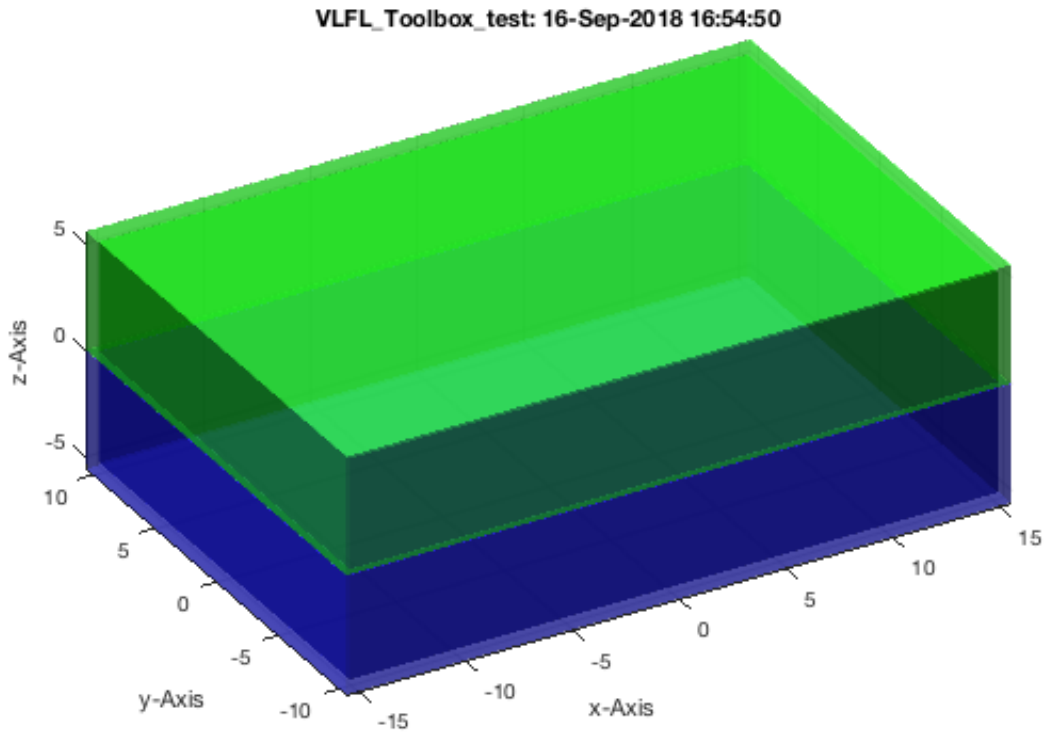
8. Separating an solid into peaces

```
SG=SGhollowsolid(SGbox([30,20,10]));
SGfigure; SGplot(SG); VLFLplotlight(1,0.5); view(-30,30);
```



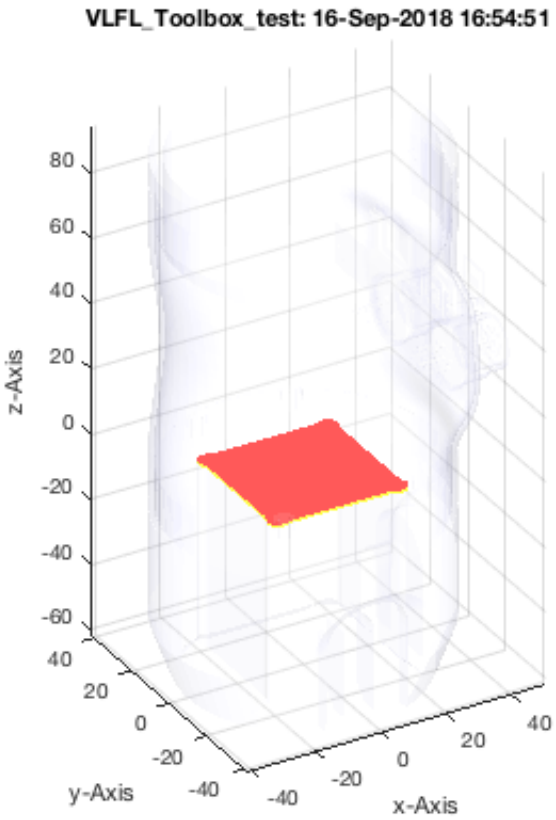

```
SGpuzzlecut3D(SG,[1 1 0.5]); VLFLplotlight(1,0.5); view(-30,30);
```

```
50% 100%
```



9. create a solid surface from an open surface

```
load JACO_robot.mat
VLFLsurfaceofSGT(JC0,'B'); h=SGplot(JC0); setplotlight(h,'w',0.1);
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:54:51!
Executed 16-Sep-2018 16:54:53 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 39: HEBO Modules robot design

2017-07-25: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-07-25

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.0 required)
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing

- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control

Motivation for this tutorial: (Originally SolidGeometry 4.0 required)

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 16:54:54!
Executed 16-Sep-2018 16:54:56 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries

2017-09-04: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2017-09-04

Contents

- FUNCTION NOT BUGF FEREE
- Create a simple bar type link
- Create a Folloer Frame at the x-Side of the solid
- Create a cutting frame in the middle
- Show a default cut at the cutting frame
- Show a 1mm cut at the cutting frame
- Show a z-cut 1mm by 40 mm at the cutting frame
- Analyze the cut and detec two separted solids
- Separate the solids into different solids
- Combined Function Simplified Peg in Hole using the same parameter as the cut
- Simplified Peg in Hole using a longer peg
- Now separate the parts
- now start to adjust the size to the required movements
- Final Remarks

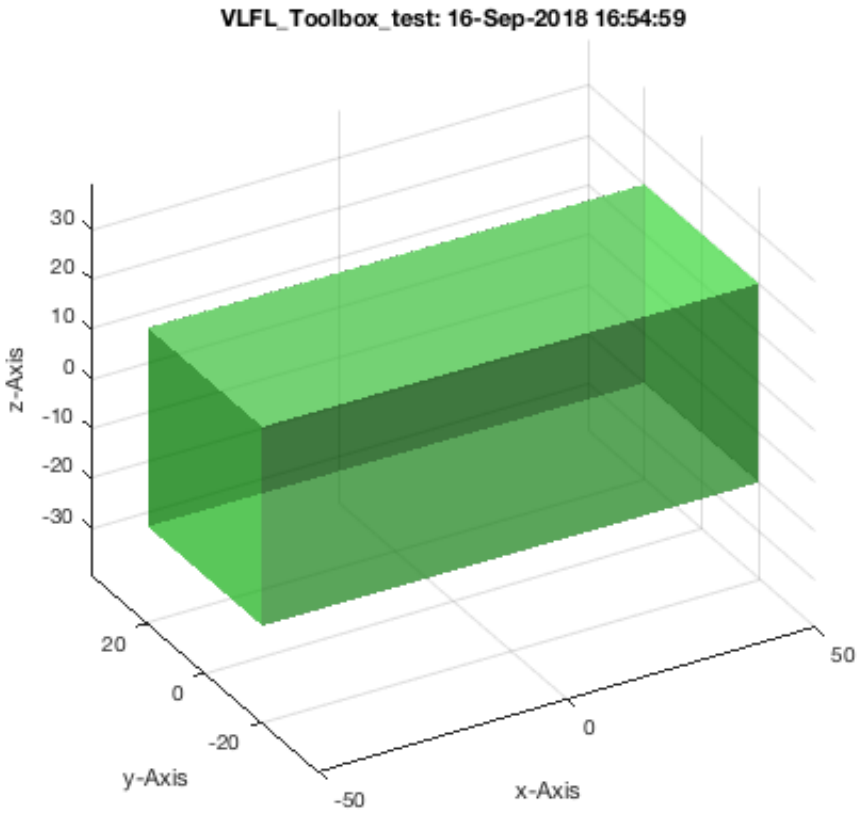
FUNCTION NOT BUGF FEREE

```
% function VLFL_EXP41
% clear all; close all;
```

Create a simple bar type link

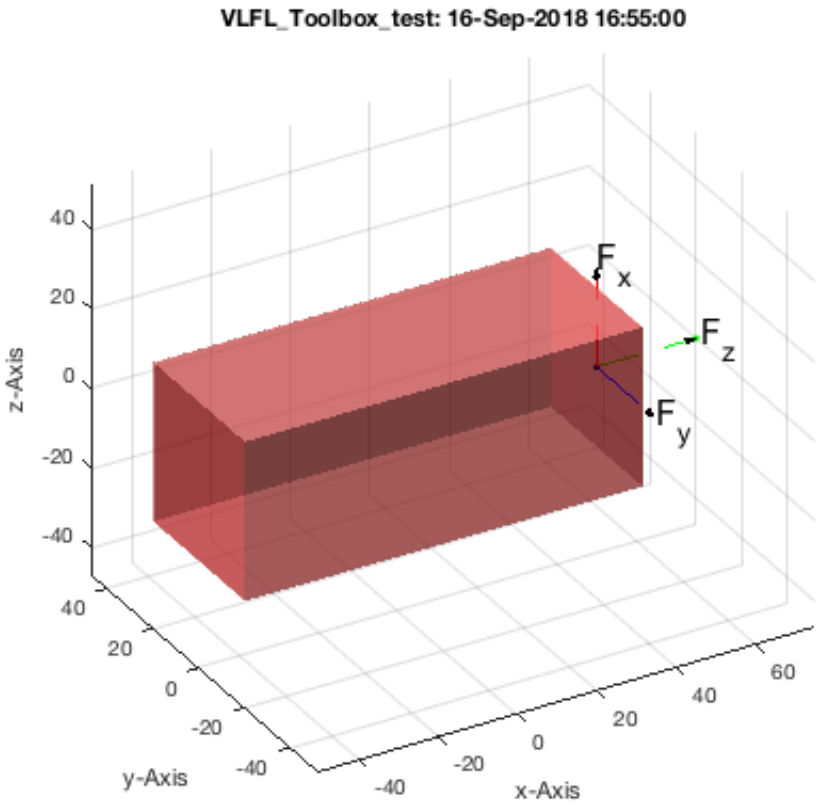
```
A=SGbox([100,40,40])
SGfigure; h=SGplot(A); view(-30,30); setplotlight(h,'g',0.5);
```

```
A =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]
```



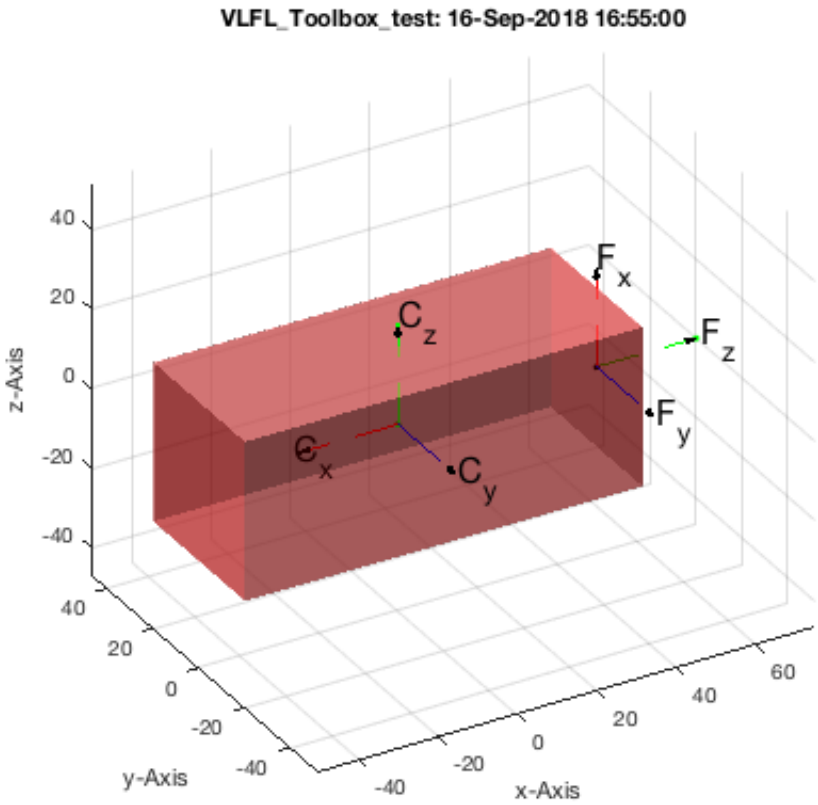
Create a Folloer Frame at the x-Side of the solid

```
A=SGTset(A,'F',TofFS(A,[1 0 0]));

SGfigure; h=SGplot(A); SGTframeplot(A);   view(-30,30); setplotlight(h,'r',0.5);
```



Create a cutting frame in the middle

```
A=SGTset(A,'C',TofT(SGTget(A,'F'),rot(0,+pi/2,0),[0 0 -50]));

SGfigure; h=SGplot(A); SGTframeplot(A);   view(-30,30); setplotlight(h,'r',0.5);
```



Show a default cut at the cutting frame

```
TC=SGTget(A,'C');
SGinsertCut(A,TC)
```

```
PL =

         0   21.0000
         0  -20.0000
         0  -21.0000
         0   -0.0000


a =

    3.1416
         0
         0
    3.1416
         0
       NaN
       NaN
         0
    3.1416
         0
         0
    3.1416
```

```
b =

    41
    41
     1
    21
    21
     0
    21
    21
     1
    41
    41
    41



Index in position 1 exceeds array bounds.

Error in CPLedgeNormal (line 40)
   dn= cp([2:end,1],:)-cp;             % direction vector

Error in CPLgrowEdge (line 70)
if nargout>1; [ENL,PNL,PDL]=CPLedgeNormal(RPL); end;

Error in PLgrow (line 37)
    [RPL,ENL,PNL,PDL]=CPLgrowEdge(PL,w);

Error in PLgrowline (line 32)
CPL=PLgrow(NPL,s/2,edge);

Error in SGinsertCut (line 54)
CPL=PLgrowline(PL,sl,true);

Error in VLFL_EXP41 (line 26)
SGinsertCut(A,TC)
```

Show a 1mm cut at the cutting frame

```
SGinsertCut(A,TC,1)
```

Show a z-cut 1mm by 40 mm at the cutting frame

```
SGinsertCut(A,TC,1,40)
```

Analyze the cut and detec two separted solids

```
B=SGinsertCut(A,TC,1,40);
SGseparatebyT(B,TC)
```

Separate the solids into different solids

```
[NX,NA,NB,NC]=SGseparatebyT(B,TC)
```

Combined Function Simplified Peg in Hole using the same parameter as the cut

```
SGinsertPeghole(B,TC,1,40)
```

Simplified Peg in Hole using a longer peg

```
SGinsertPeghole(B,TC,1,40,20)
```

Now separate the parts

```
C=SGinsertPeghole(B,TC,1,40,20)
SGseparatebyT(C,TC)
```

now start to adjust the size to the required movements

```
[X,Y]=SGseparatebyT(C,TC)

% SGboolTL(Y,'-',SGtransrelT(SGgrow(X,0.2),TC,TofR(rot(0,0,1*pi/10)))); Y=SGdelaunay(ans);
% SGboolTL(Y,'-',SGtransrelT(SGgrow(X,0.2),TC,TofR(rot(0,0,2*pi/10)))); Y=ans;
% SGboolTL(Y,'-',SGtransrelT(SGgrow(X,0.2),TC,TofR(rot(0,0,3*pi/10)))); Y=ans;
```

```
wlim=[0 +pi/4]
CVLofSGcutTrot(NB,TC,wlimm1);
[~,~,~,~,XA,XB]=CVLofSGcutTrot(NB,TC,wlim,1);
```

```
SGboolTL(Y,'-',XA)
```

```
SGboolTL(Y,'-',XB)
```

Final Remarks

```
close all
VLFLlicense
```

Published with MATLAB® R2018a

Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids

2018-02-27: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2018-03-08

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.2 required)
- List of function introduced in this tutorial
- 1. Conversion between triangle surface model and tetrahedon volumen model
- 1.1 Create a simple bar type link
- 1.2 Create a pde mesh model of the simple bar with voxel size 5mm
- 1.3 Show the tetrahedron volume structure of the mesh
- 1.4 Convert the tetrahedron volume into a surface model
- 1.5 Remove surface points of the surface model but protect the edge points
- 1.6 Remove unused edge points and surface points of the surface model
- 2. Selection of Feature Surfaces for load specification
- 2.1 Feature surface plot on surface model lebel
- 2.2 Feature surface plot on pde model lebel
- 3. Calculating surface load dependend displacement and von-Miss stress situation
- 3.1 Display a loading condition Fixed facet is 4, loaded surface is 3, load vector in z using Propertynames
- 3.2 Display a loading condition Fixed facet is 4, loaded surface is 3, load vector in z using varargin
- 3.3 Fixed facet is 4, loaded surface is 3, load vector in z using varargin
- 3.4 Fixed facet is 4, loaded surface is 3, load vector in z using Propertynames
- 3.5 Show von-mises-Stress for load condition
- 3.6 Show von-mises-Stress and load condition
- 3.7 Do the same for the matlab standard fem solid: BracketWithHole
- 4 Structural Optimization
- 4.1 CAO Optimization using load face 9
- 4.2 CAO Optimization using load face 6
- 4.3 CAO Optimization using load face 5
- 4.4 CAO Optimization using load face 1
- 4.5 CAO Optimization of a simple bar
- 4.6 Show the stress distribution in the CAO optimized shape
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control

- Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries
- Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids
- Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids

Motivation for this tutorial: (Originally SolidGeometry 4.2 required)

Yinlun Sun of TU Munich has supplemented the SG-Library with functions that allow a structural and topological optimization of geometric bodies with surface representation.

List of function introduced in this tutorial

- pdemodelofSG - creates a pde tetrahedron mesh-model from a solid surface geometry
- pdeplot3D - Plot 3-D solution or surface mesh
- SGofpdemodel - returns a solid geometry surface model of a pde model
- SGremsurfpoints - returns a surface model without surface points that are inside of a surface - boundary/edge points are unchanged
- SGremsurfedgepoints - returns a surface model without edge points and surface points that are inside of a surface
- pdegplot - Plot PDE tetrahedron mesh geometry
- FSplot - plots the featureEdges of TR, SG or VLFL
- pdeplotfaces - simply plots the surfaces to select; similar as FSplot
- SGplotsurfaceload - plots the surface load of a solid geometry
- 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 by YINLUN SUN
- SGshapeOptiCAO - returns the optimized shape of a given structure based on biological growth

```
function VLFL_EXP42
```

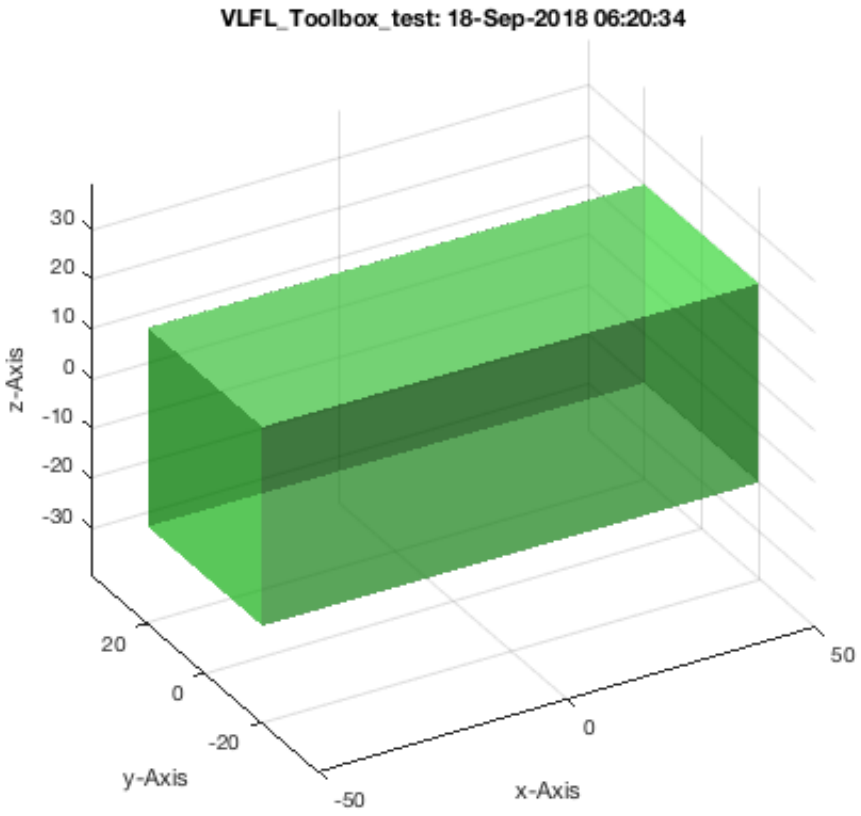
```
% clear all; close all;
```

1. Conversion between triangle surface model and tetrahedon volumen model

1.1 Create a simple bar type link

```
A=SGbox([100,40,40])
SGfigure; h=SGplot(A); view(-30,30); setplotlight(h,'g',0.5);
```

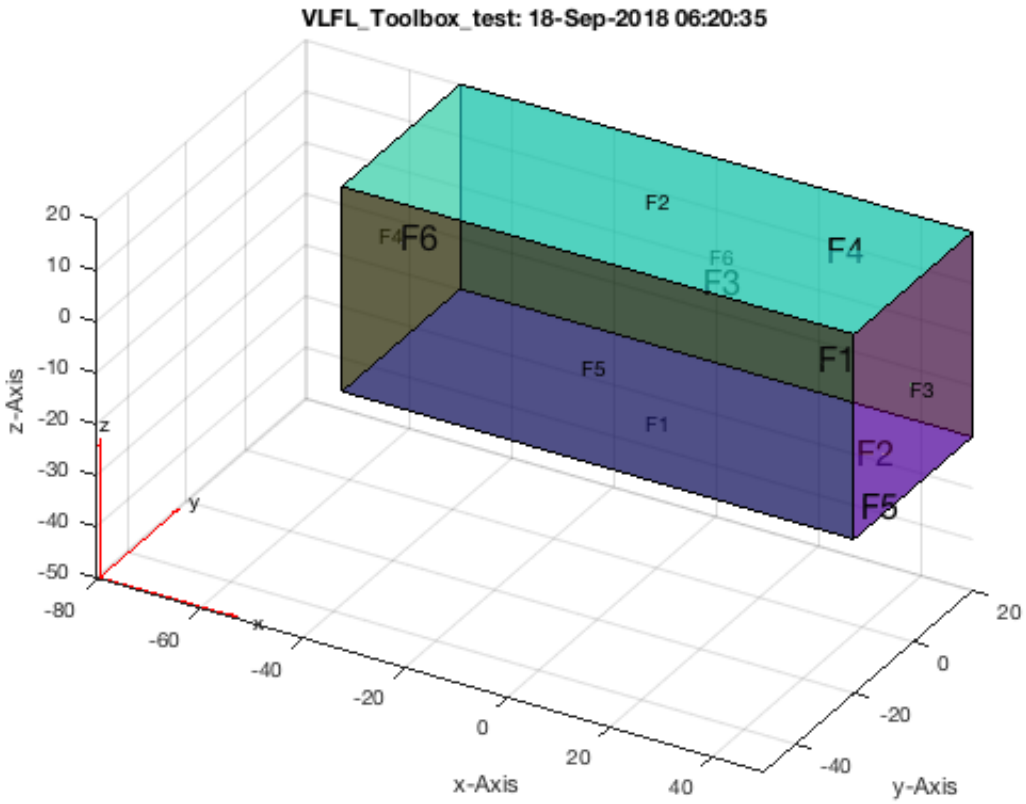
```
A =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]
```



1.2 Create a pde mesh model of the simple bar with voxel size 5mm

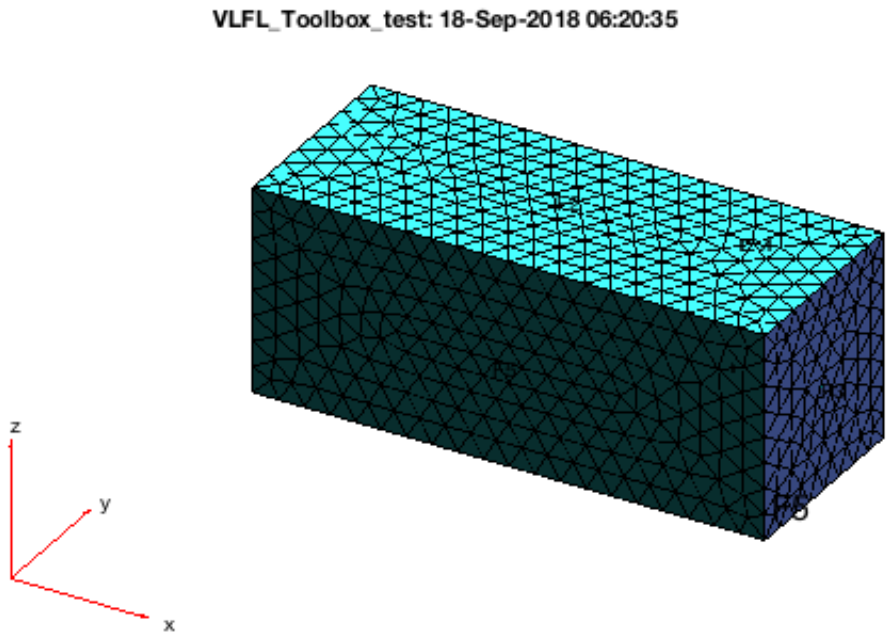
```
pdemodelofSG(A,5); model=ans
```

```
6 Feature Surfaces found! Only the largest 99.90% (4.000 .. 4000.0mm^2), i.e. 6 of 6 are sh
own.

model =

  PDEModel with properties:

          PDESystemSize: 3
        IsTimeDependent: 0
               Geometry: [1×1 DiscreteGeometry]
    EquationCoefficients: [1×1 CoefficientAssignmentRecords]
      BoundaryConditions: []
       InitialConditions: []
                    Mesh: [1×1 FEMesh]
          SolverOptions: [1×1 PDESolverOptions]
```



1.3 Show the tetrahedron volume structure of the mesh

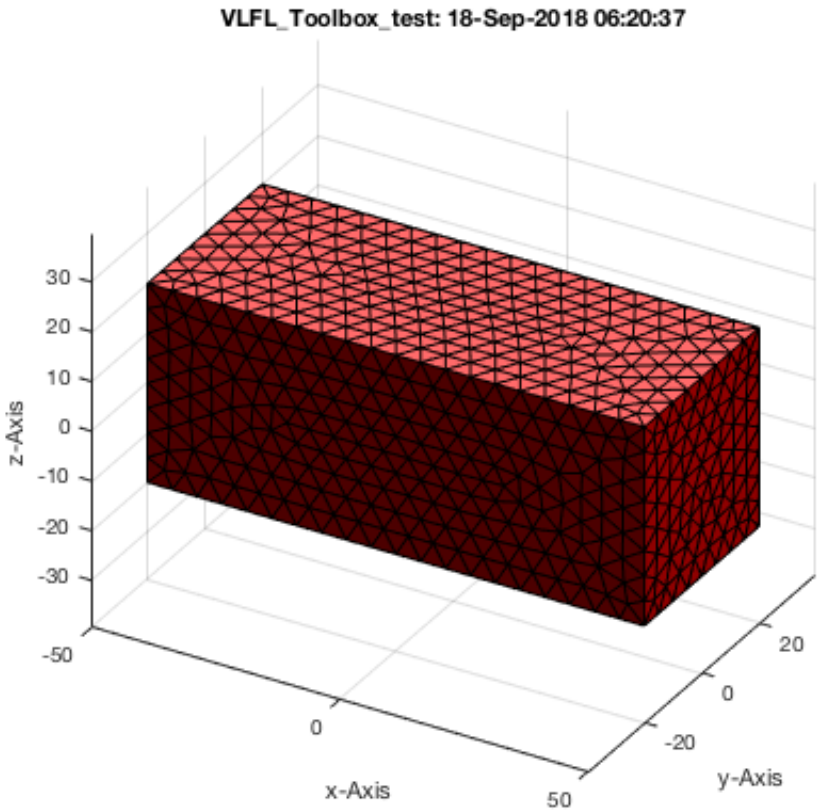
```
pdeplot3D(model);
```



1.4 Convert the tetrahedron volume into a surface model

```
SGofpdemodel(model); B=ans
```

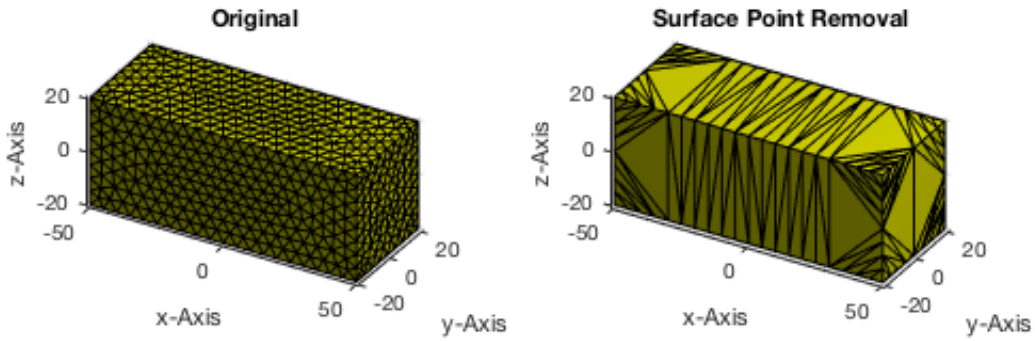
```
B =

  struct with fields:

    VL: [924×3 double]
    FL: [1844×3 double]
```



1.5 Remove surface points of the surface model but protect the edge points

```
SGremsurfpoints(B); C=ans
```

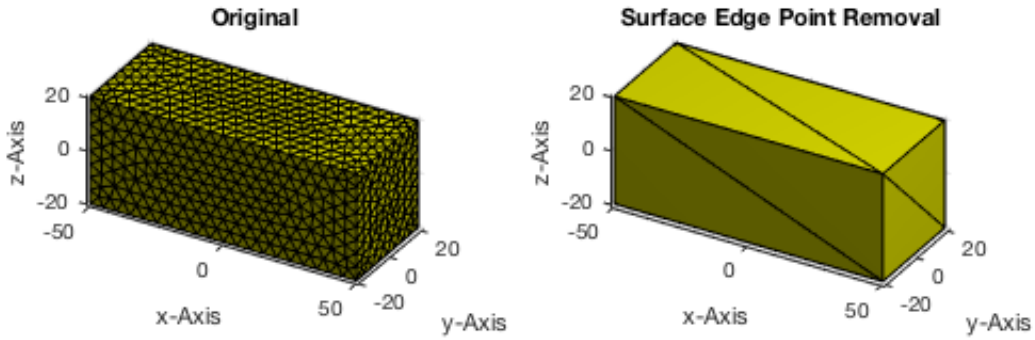
```
C =

  struct with fields:

    VL: [140×3 double]
    FL: [276×3 double]
```



1.6 Remove unused edge points and surface points of the surface model

```
SGremsurfedgepoints(B); C=ans
```

```
C =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]
```

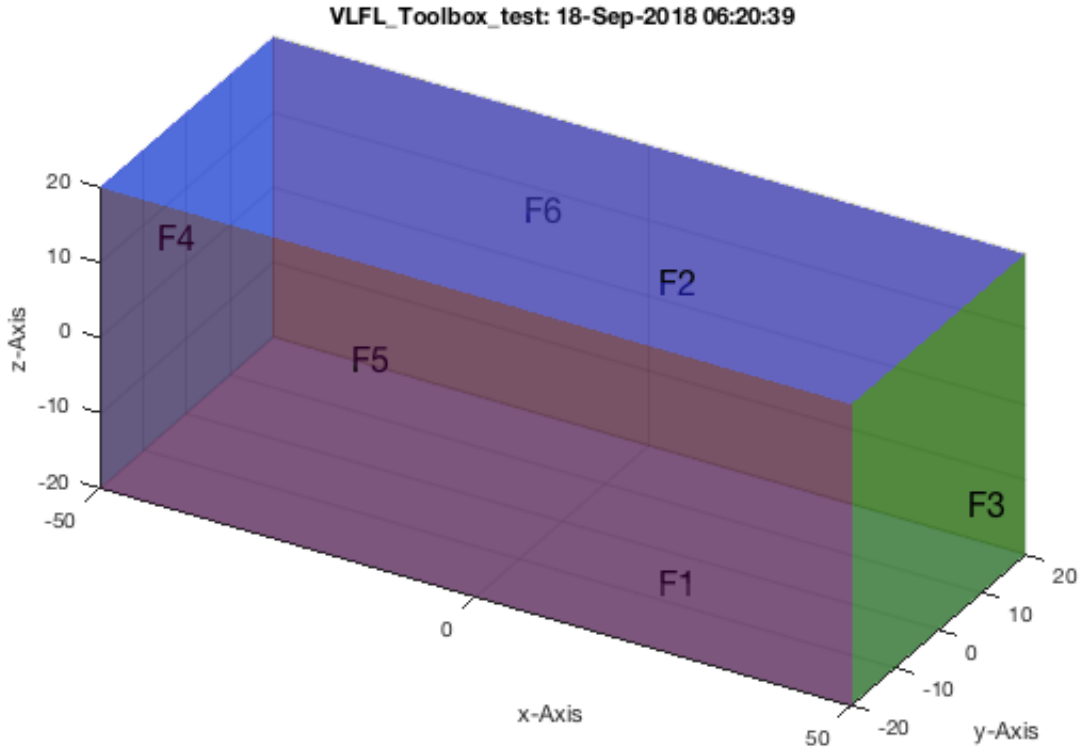


2. Selection of Feature Surfaces for load specification

2.1 Feature surface plot on surface model lebel

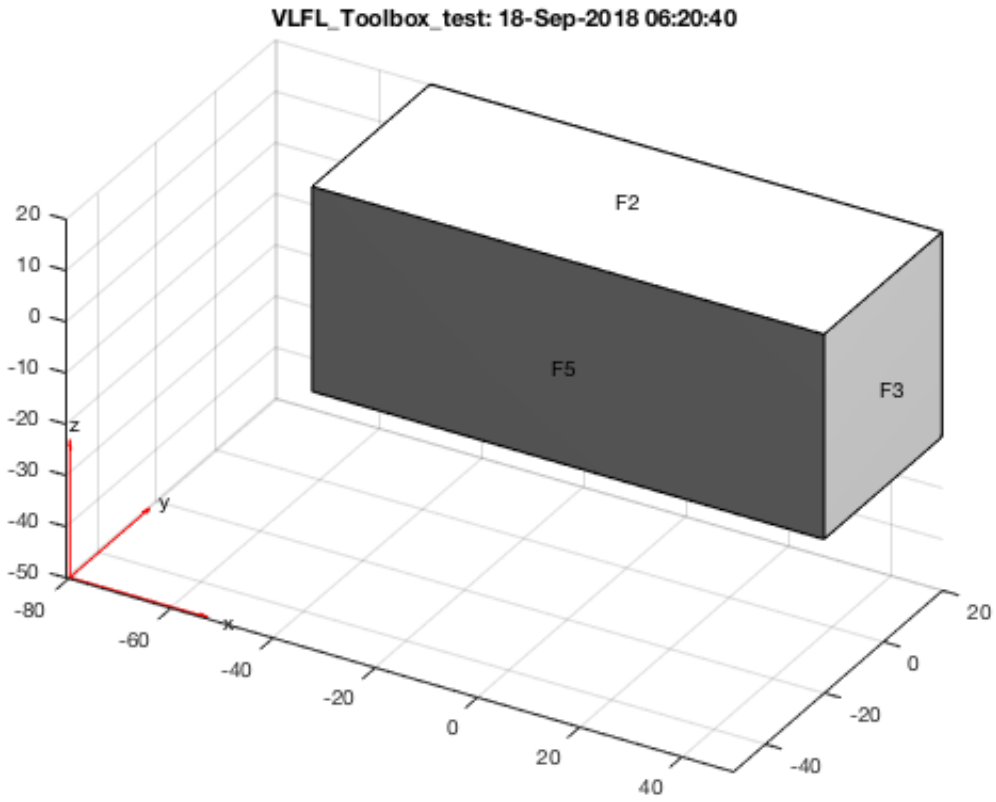
```
SGfigure; view(30,30);
FSplot(A);
```

```
6 Feature Surfaces found! Only the largest 99.90% (4.000 .. 4000.0mm^2), i.e. 6 of 6 are sh
own.
```



2.2 Feature surface plot on pde model lebel

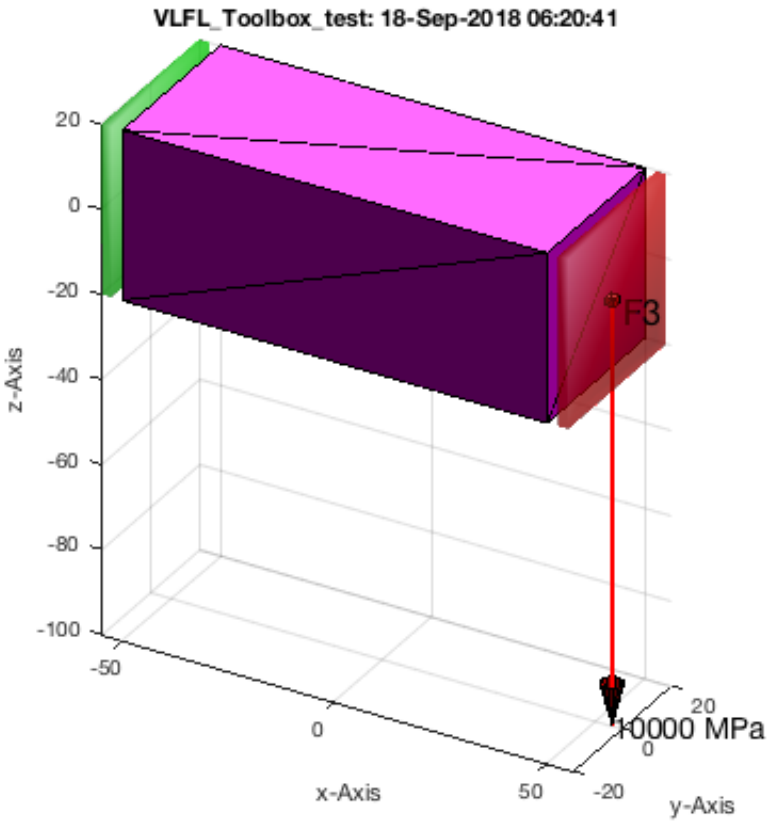
```
SGfigure; view(30,30)
pdeplotfaces(model);
```



3. Calculating surface load dependend displacement and von-Miss stress situation

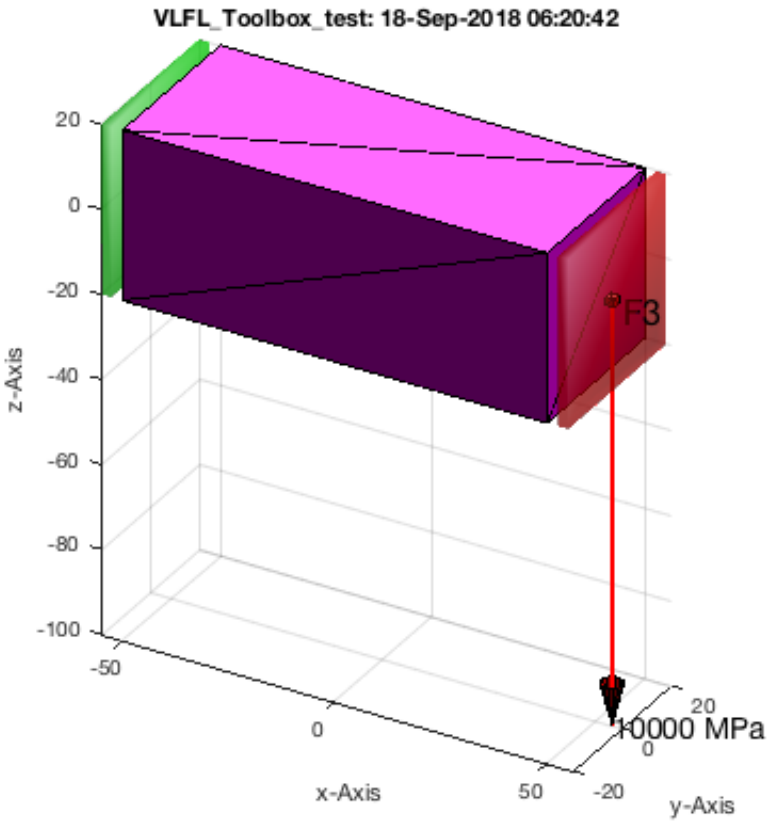
3.1 Display a loading condition Fixed facet is 4, loaded surface is 3, load vector in z using Propertynames

```
SGfigure; SGplot(A,'m'); view(30,30);
SGplotsurfaceload (A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e4]);
```



3.2 Display a loading condition Fixed facet is 4, loaded surface is 3, load vector in z using varargin

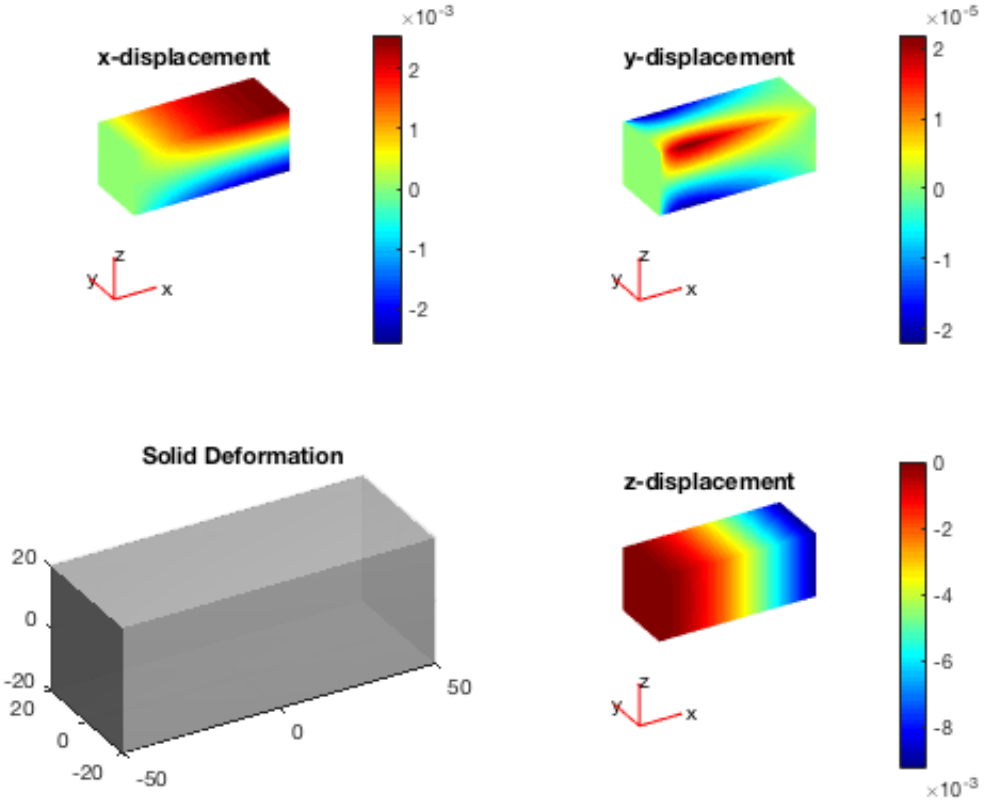
```
SGfigure; SGplot(A,'m'); view(30,30);
SGplotsurfaceload (A,4,3,[0 0 -1e4]);
```



3.3 Fixed facet is 4, loaded surface is 3, load vector in z using varargin

```
pdesolvesurfaceload(model,4,3,[0 0 -1e4]);
```

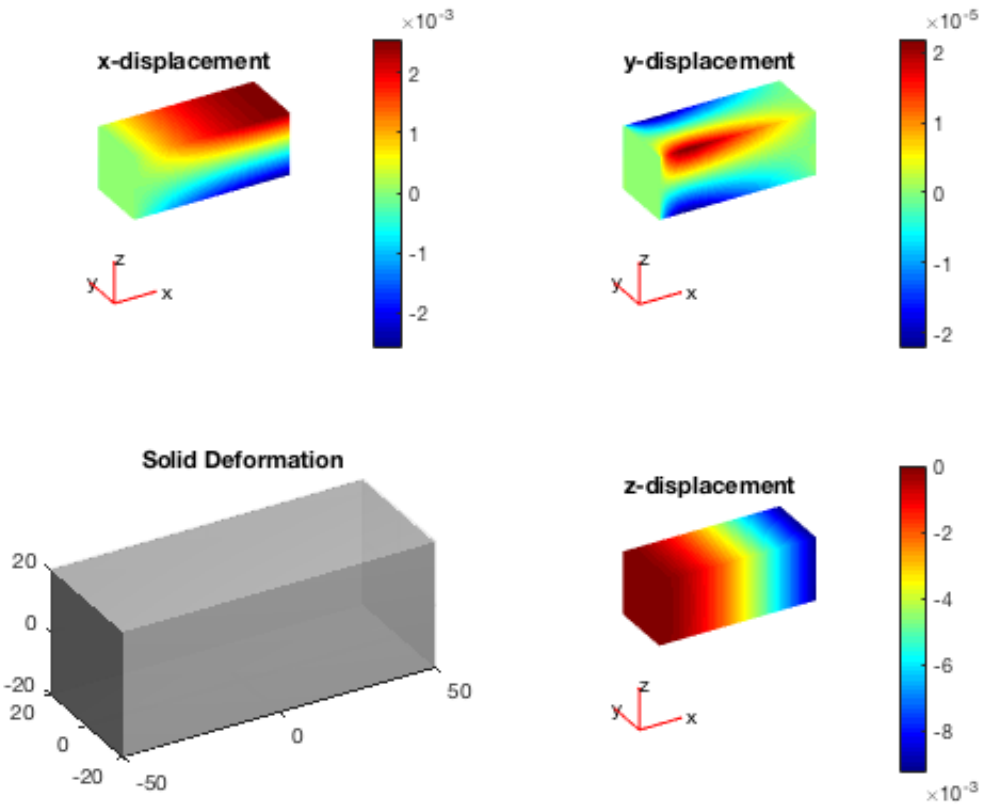
```
ATTENTION: The already existing pde BoundaryConditions are deleted first
```



3.4 Fixed facet is 4, loaded surface is 3, load vector in z using Propertynames

```
pdesolvesurfaceload(model,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e4]);
```

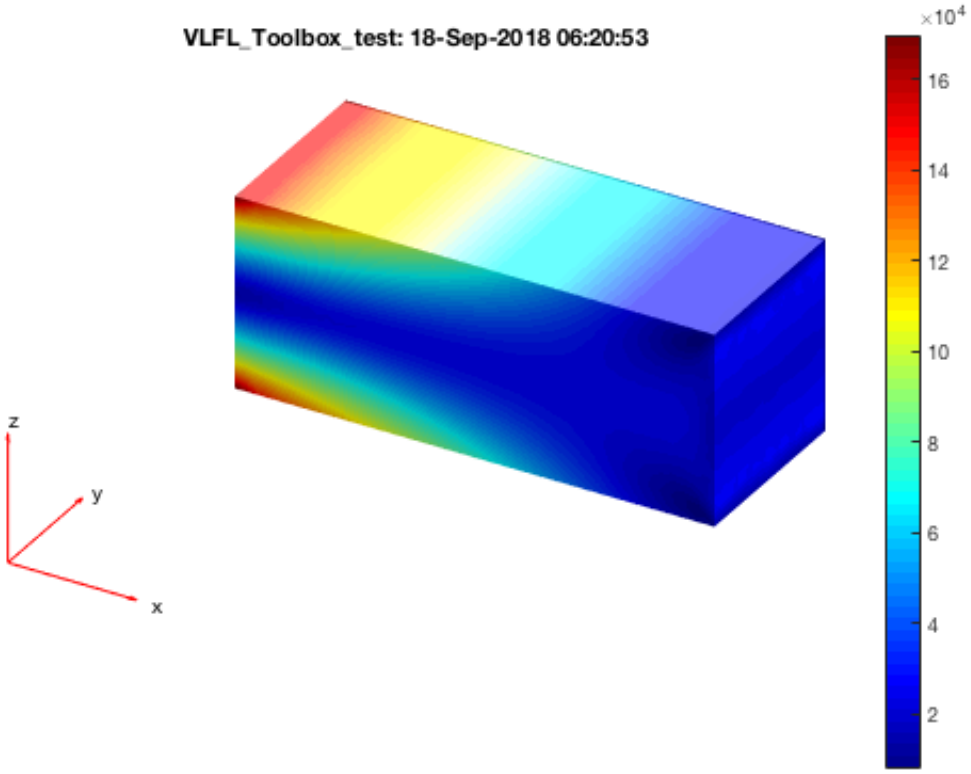
```
ATTENTION: The already existing pde BoundaryConditions are deleted first
```



3.5 Show von-mises-Stress for load condition

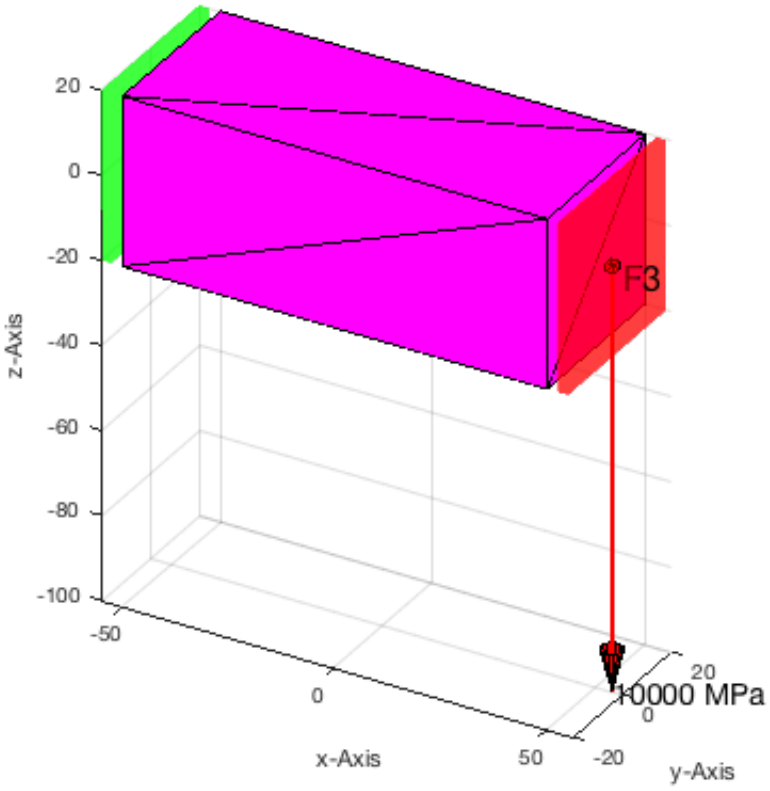
```
[result,model]=pdesolvesurfaceload(model,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0
 0 -1e4]);
pdestressstatic(model,result);
```

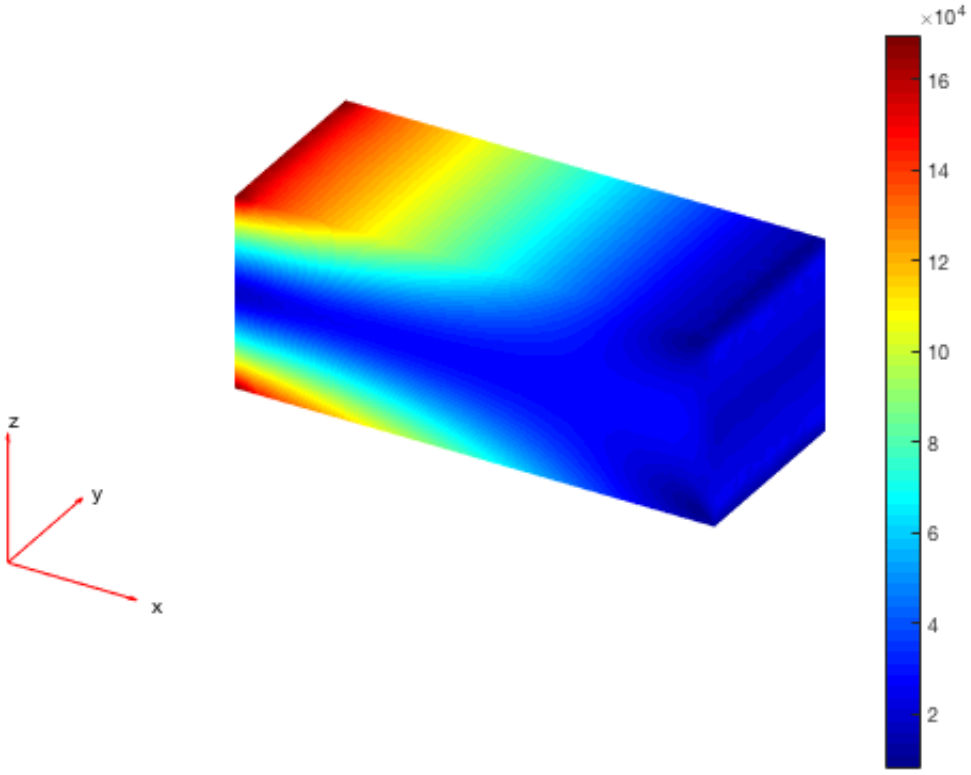
```
ATTENTION: The already existing pde BoundaryConditions are deleted first
```



3.6 Show von-mises-Stress and load condition

```
close all; figure(1);  view(30,30); SGplot(A,'m');
SGplotsurfaceload (A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e4]);
figure(2); view(30,30);
[~,stress]=pdestressstatic(model,result);
pdeplot3D(model,'colormapdata',stress);
```

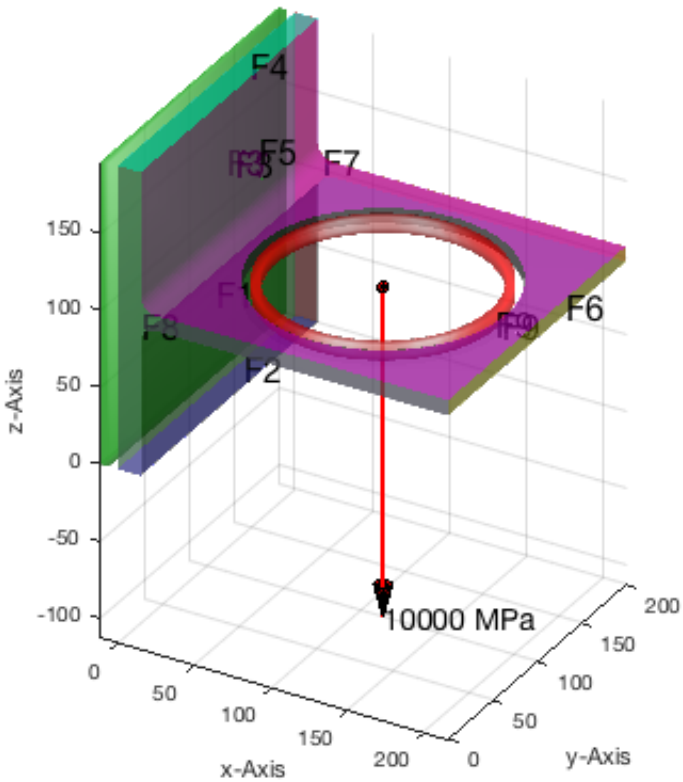


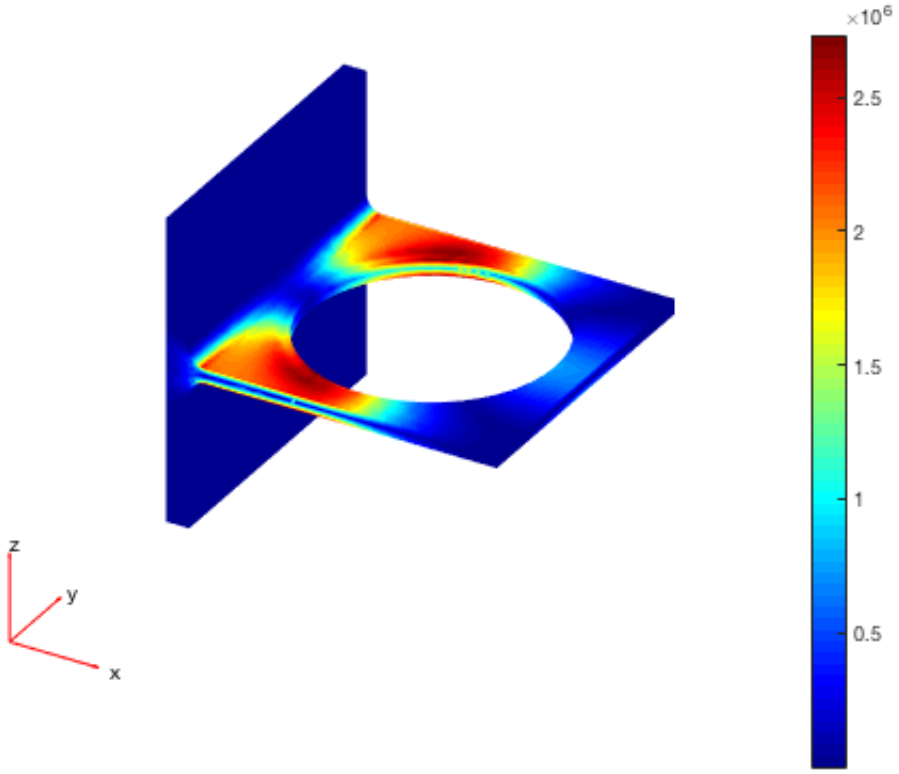


3.7 Do the same for the matlab standard fem solid: BracketWithHole

```
A=SGreadSTL(which('BracketWithHole.stl'),1000);
model=pdemodelofSG(A);
[result,model]=pdesolvesurfaceload(model,'FixedFaceIndices',3,'LoadFaceIndices',9,'Load',[0
 0 -1e4]);
close all; figure(1);  view(30,30); FSplot(A);
SGplotsurfaceload (A,'FixedFaceIndices',3,'LoadFaceIndices',9,'Load',[0 0 -1e4]);
figure(2); view(30,30);
[~,stress]=pdestressstatic(model,result);
pdeplot3D(model,'colormapdata',stress);
```

```
LOADING ASCII STL-File: /Applications/MATLAB_R2018a.app/toolbox/pde/pdedata/BracketWithHole
.stl scaling factor: 1000
Processing 2102 lines:
Finishing solid bracket_with_hole_metersATTENTION: The already existing pde BoundaryConditi
ons are deleted first
9 Feature Surfaces found! Only the largest 99.90% (39.433 .. 39433.2mm^2), i.e. 9 of 9 are
shown.
```



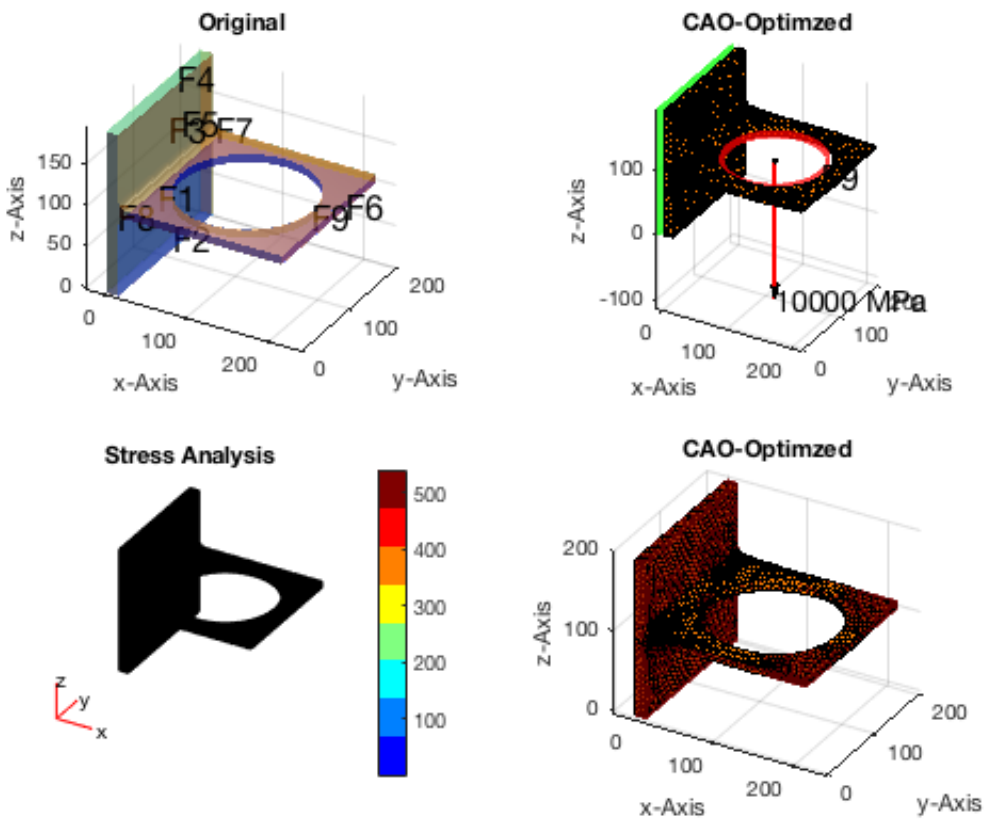


4 Structural Optimization

4.1 CAO Optimization using load face 9

```
SGshapeOptiCAO(A,'FixedFaceIndices',3,'LoadFaceIndices',9,'Load',[0 0 -1e4]);
```

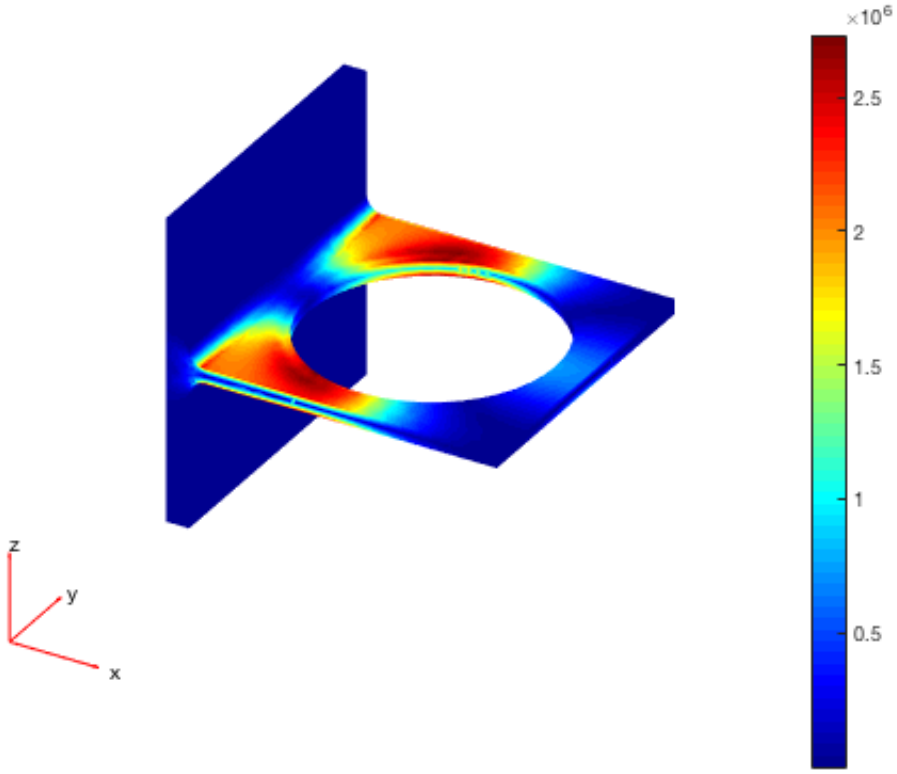
```
Iteration   0: Volume of SG is 801143.4667
Iteration   1: CAO end: CAO process stops because meshing size does not fit.

******************** CAO result *********************
Original volume: 801143.4667 mm^3
Optimized volume: 801143.4667 mm^3
Original maximal von Mises stress: 541.644 N/mm^2
Optimized maximal von Mises stress: 541.644 N/mm^2
9 Feature Surfaces found! Only the largest 99.90% (39.433 .. 39433.2mm^2), i.e. 9 of 9 are
shown.
```

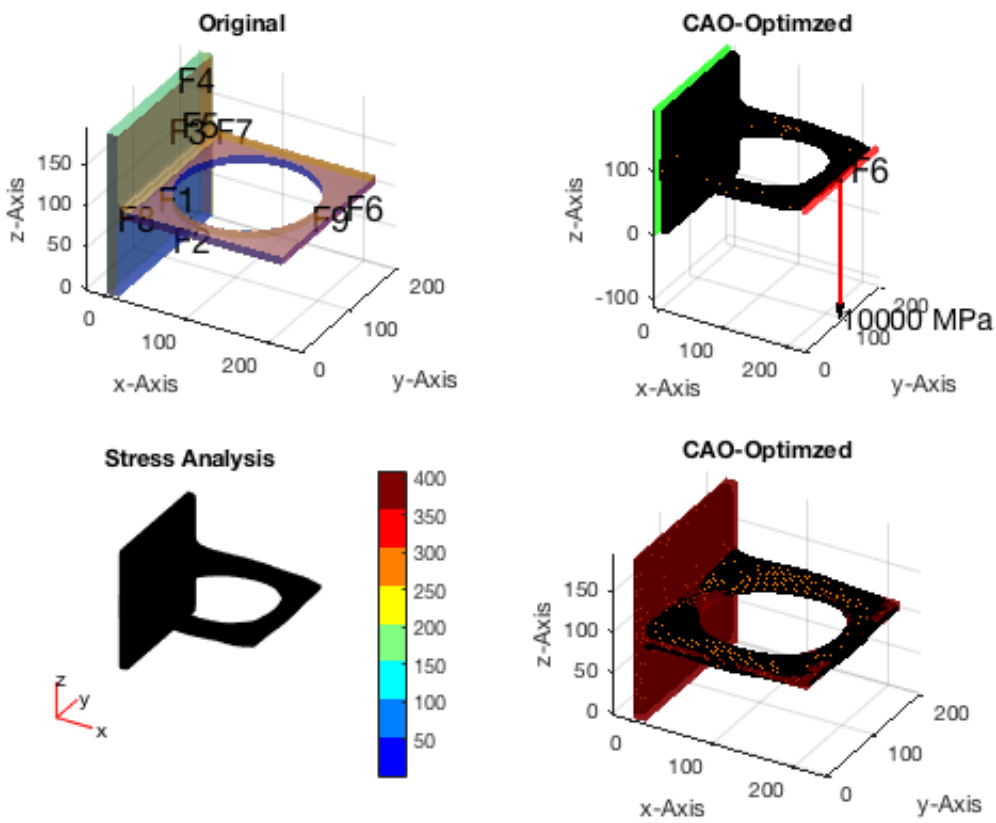


4.2 CAO Optimization using load face 6

```
SGshapeOptiCAO(A,'FixedFaceIndices',3,'LoadFaceIndices',6,'Load',[0 0 -1e4]);
```

```
Iteration   0: Volume of SG is 801084.6513
Iteration   1: Volume of SG is 789944.378
Iteration   2: Volume of SG is 779041.5887
Iteration   3: Volume of SG is 770902.8223
Iteration   4: Volume of SG is 764095.7351
Iteration   5: CAO end: CAO process stops because meshing size does not fit.

******************* CAO result *********************
Original volume: 801084.6513 mm^3
Optimized volume: 764095.7351 mm^3
Original maximal von Mises stress: 1679.0079 N/mm^2
Optimized maximal von Mises stress: 409.3249 N/mm^2
9 Feature Surfaces found! Only the largest 99.90% (39.433 .. 39433.2mm^2), i.e. 9 of 9 are
shown.
```





4.3 CAO Optimization using load face 5

SGshapeOptiCAO(A,'FixedFaceIndices',3,'LoadFaceIndices',5,'Load',[0 0 -1e4]);

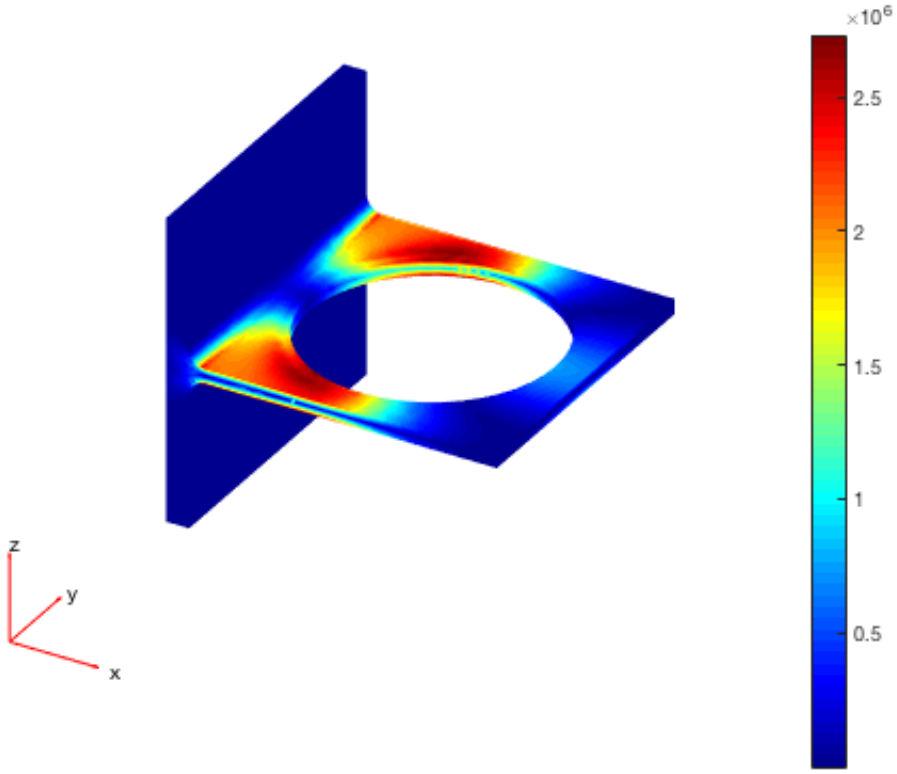
4.4 CAO Optimization using load face 1

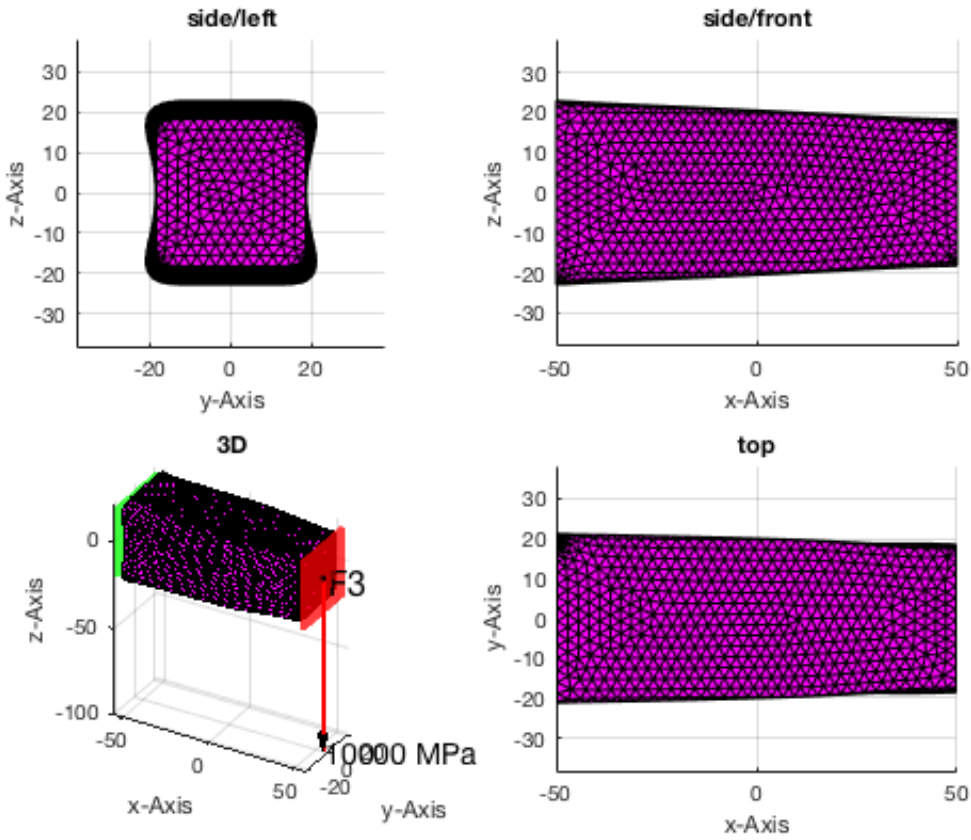
SGshapeOptiCAO(A,'FixedFaceIndices',3,'LoadFaceIndices',1,'Load',[0 0 -1e4]);

4.5 CAO Optimization of a simple bar

```
A=SGbox([100,40,40])
[B,result,model]=SGshapeOptiCAO(A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e4
]);
SGplot4(B,'m');
subplot(2,2,3); SGplotsurfaceload(A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1
e4]);
```

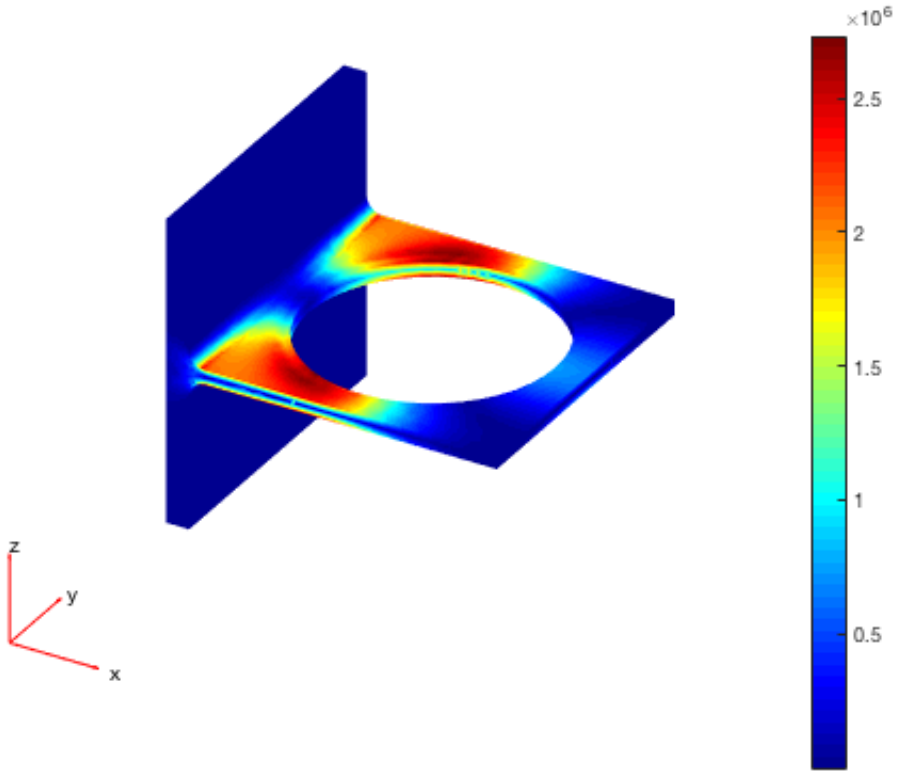
```
A =

  struct with fields:

    VL: [8×3 double]
    FL: [12×3 double]

Iteration   0: Volume of SG is 160000
Iteration   1: Volume of SG is 159744.8305
Iteration   2: Volume of SG is 159467.0401
Iteration   3: Volume of SG is 159107.6043
Iteration   4: Volume of SG is 158752.6601
Iteration   5: Volume of SG is 158360.4781
Iteration   6: Volume of SG is 158060.6989
Iteration   7: CAO end: CAO process stops because meshing size does not fit.

******************** CAO result *********************
Original volume: 160000 mm^3
Optimized volume: 158060.6989 mm^3
Original maximal von Mises stress: 110.75 N/mm^2
Optimized maximal von Mises stress: 76.6522 N/mm^2
```

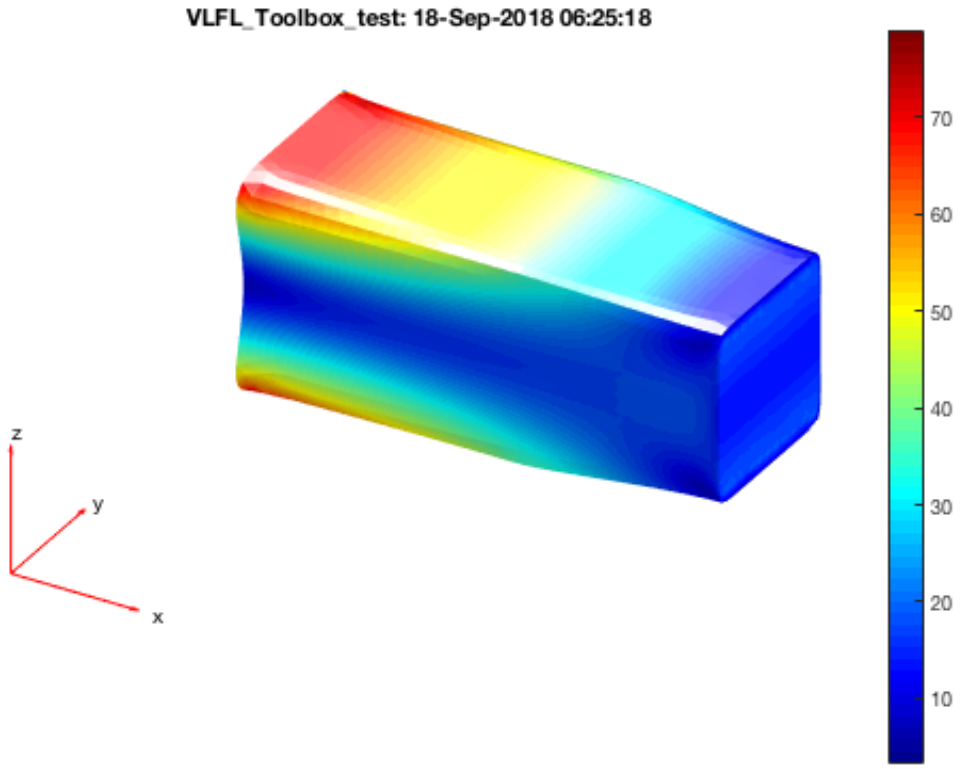




4.6 Show the stress distribution in the CAO optimized shape

```
SGfigure; pdestressstatic(model,result);
view(30,30);
```





Final Remarks

```
close all
VLFLlicense
```

```
Warning: Error creating or updating Patch
 Error in value of property  <a
 href="matlab:helpUtils.reference.showPropertyHelp('matlab.graphics.primitive.Patch','FaceV
ertexCData');")">FaceVertexCData</a>
  Number of colors must equal number of vertices or faces


This VLFL-Lib, Rel.  (2018-Sep-18), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 21-Jun-2073 06:25:19!
Executed 18-Sep-2018 06:25:21 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
matlab
pde_toolbox
==================================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids

2018-03-08: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2018-03-08

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.2 required)
- List of function introduced in this tutorial
- 1. Conversion between triangle surface model and tetrahedon volumen model
- 1.1 Create a simple bar type link
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model

- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control
- Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries
- Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids
- Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids

Motivation for this tutorial: (Originally SolidGeometry 4.2 required)

Yinlun Sun of TU Munich has supplemented the SG-Library with functions that allow a structural and topological optimization of geometric bodies with surface representation.

List of function introduced in this tutorial

```
**************
```

```
% function VLFL_EXP43
% clear all; close all;
```

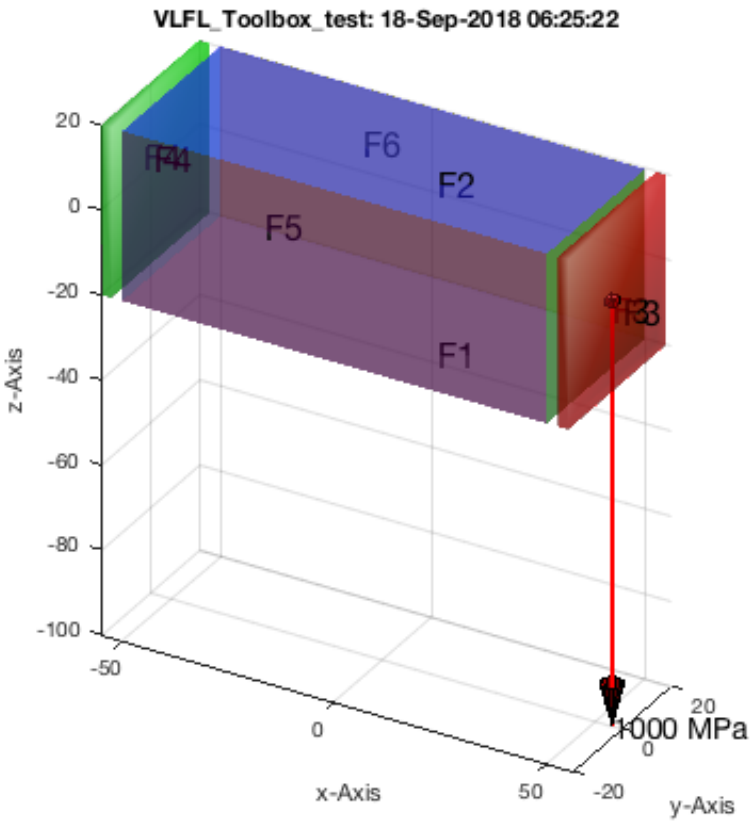
1. Conversion between triangle surface model and tetrahedon volumen model

1.1 Create a simple bar type link

```
A=SGbox([100,40,40])
SGfigure; FSplot(A); view(30,30);
SGplotsurfaceload (A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e3]);
```

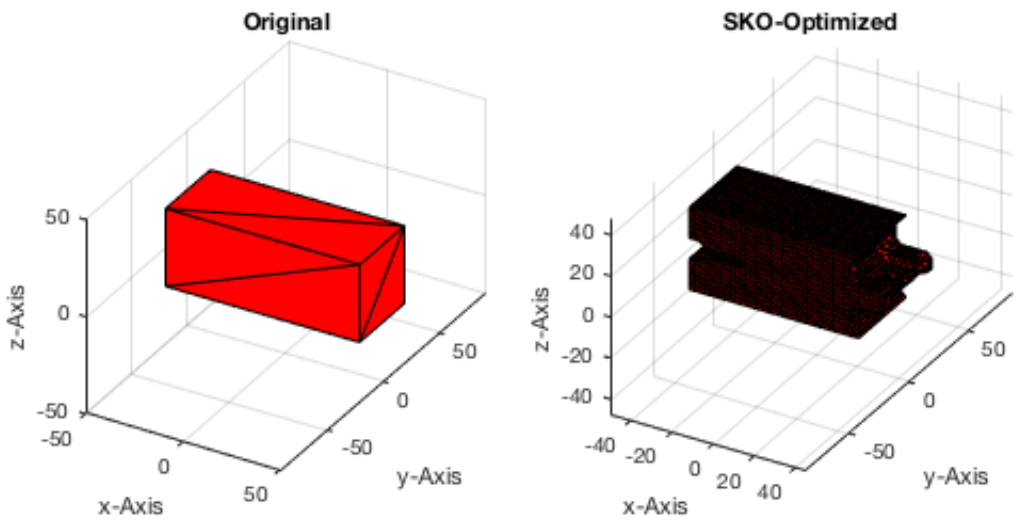
```
A =

  struct with fields:
```

```
    VL: [8×3 double]
    FL: [12×3 double]

6 Feature Surfaces found! Only the largest 99.90% (4.000 .. 4000.0mm^2), i.e. 6 of 6 are sh
own.
```




```
SGshapeOptiSKO(A,'FixedFaceIndices',4,'LoadFaceIndices',3,'Load',[0 0 -1e3]); B=ans
```

```
Iteration 0: Average stress: 2.95, Maximum stress: 10.61
Iteration 1: Average stress: 2.95, Maximum stress: 10.73
Iteration 2: Average stress: 2.95, Maximum stress: 10.85
Iteration 3: Average stress: 2.94, Maximum stress: 10.96
Iteration 4: Average stress: 2.94, Maximum stress: 11.08
Iteration 5: Average stress: 2.94, Maximum stress: 11.19
Iteration 6: Average stress: 2.94, Maximum stress: 11.30
Iteration 7: Average stress: 2.94, Maximum stress: 11.41
Iteration 8: Average stress: 2.94, Maximum stress: 11.52

B =

  struct with fields:

      VL: [3034×3 double]
      FL: [6054×3 double]
    pcon: 0.8000
```



Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-18), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 21-Jun-2073 06:26:10!
Executed 18-Sep-2018 06:26:12 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
matlab
pde_toolbox
==========================================================================================
==========
```

Published with MATLAB® R2018a

Tutorial 44: Creation of solids and kinematics from 3D curves and transformation matrices

2018-07-24: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de) - Last Change: 2018-07-24

Contents

- Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox
- Motivation for this tutorial: (Originally SolidGeometry 4.2 required)
- List of function introduced in this tutorial
- Using VLsample to create example funktions
- Creating edge normal function for an open spatial curve
- Creating normal function for a closed spatial curve
- Creating normal function for open spatial radial curve
- Creating normal function for closed spatial radial curve
- Creating Solid Geometries open
- Creating Solid Geometries open
- Creating Solid Geometries open
- 1. Conversion between triangle surface model and tetrahedon volumen model
- Final Remarks

Complete List of all Tutorials with Publishable MATLAB Files of this Solid-Geoemtries Toolbox

The following topics are covered an explained in the specific tutorials:

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons

- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data
- Tutorial 38: Some more solid geometry modelling function
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control
- Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries
- Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids
- Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids
- Tutorial 44: Creation of solids and kinematics from 3D curves and transformation matrices

Motivation for this tutorial: (Originally SolidGeometry 4.2 required)

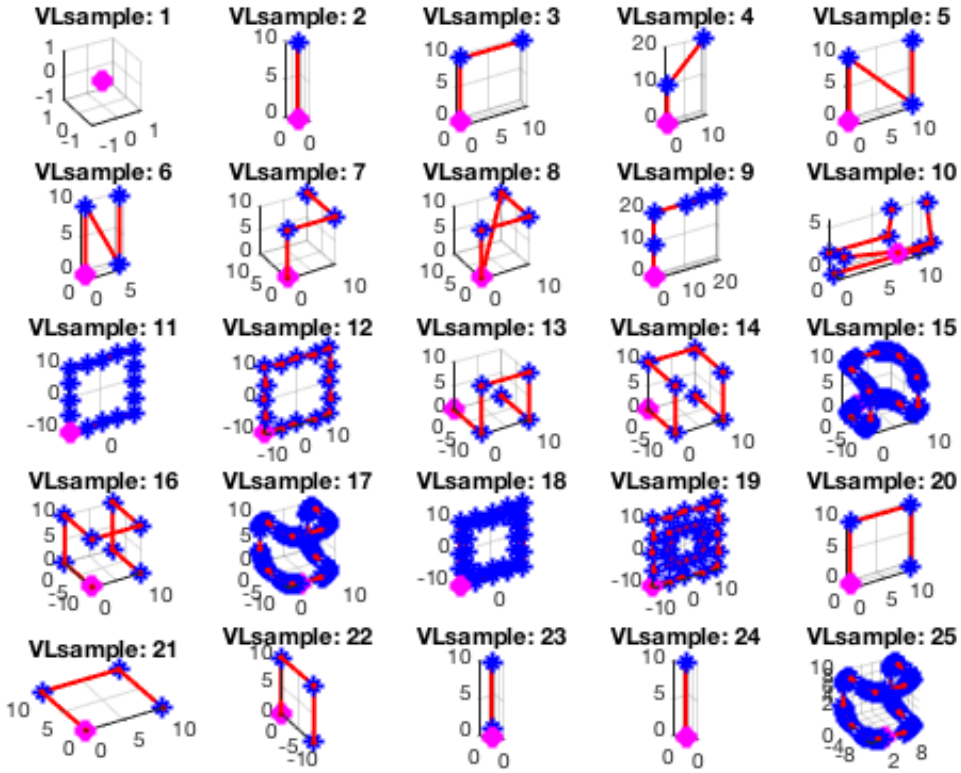
The creation of solids from space curves and a cross-section polygon is not trivial. The affected persons in the SGLib are VLsample - for generating example curves VLedgeNormal - Non trivial function for creating normal orthogonal vectors The creation of solids from space curves and a cross-section polygon is not trivial. The affected functions in the SGLib are VLsample - for generating example curves VLedgeNormal - Non-trivial function for generating normal orthogonal vectors SGcontourtube - The first function with rotating matrices (error in special cases) SGcontourtube2 - The new function with VLedgeNormal (previously error-free) SGofCPLCVVLR - Now based on SGcontourtube2 SGof2T - Now based on SGcontourtube2 SGTofDenavitHartenberg - Based on SGof2T SGTofDHset - Based on SGTofDenavitHartenberg

List of function introduced in this tutorial

* * * * * * * * * * * * * *

Using VLsample to create example funktions

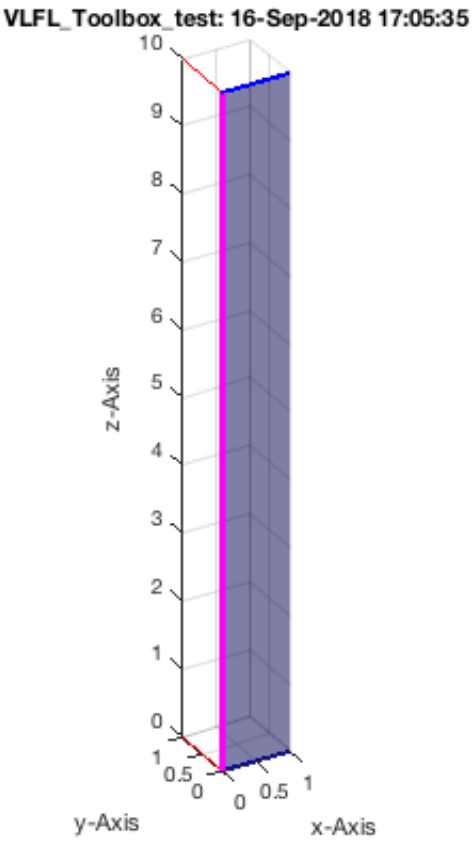
```
VLsample;
```



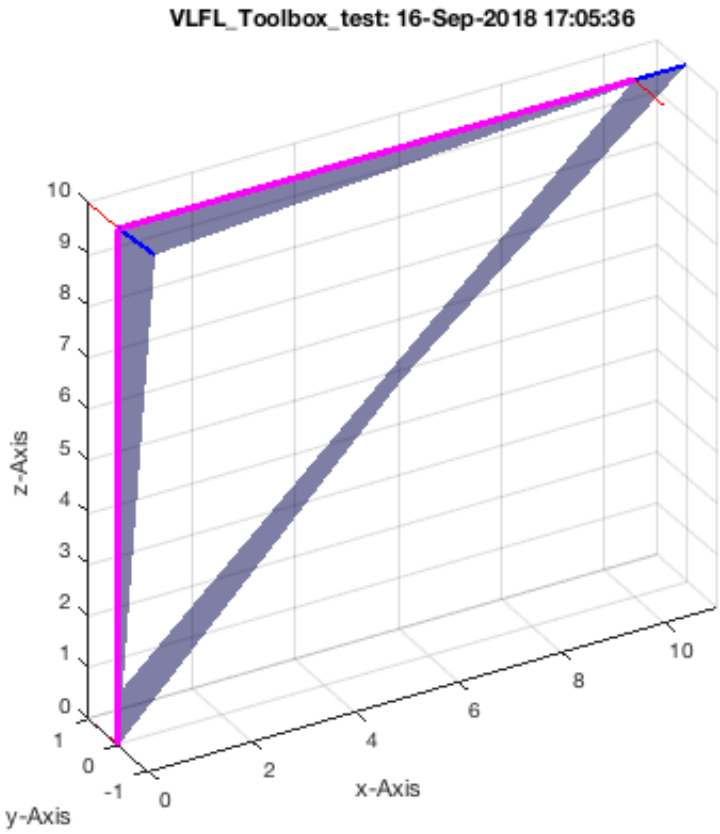
Creating edge normal function for an open spatial curve

If angles are larger than 90 degree (pi/2)

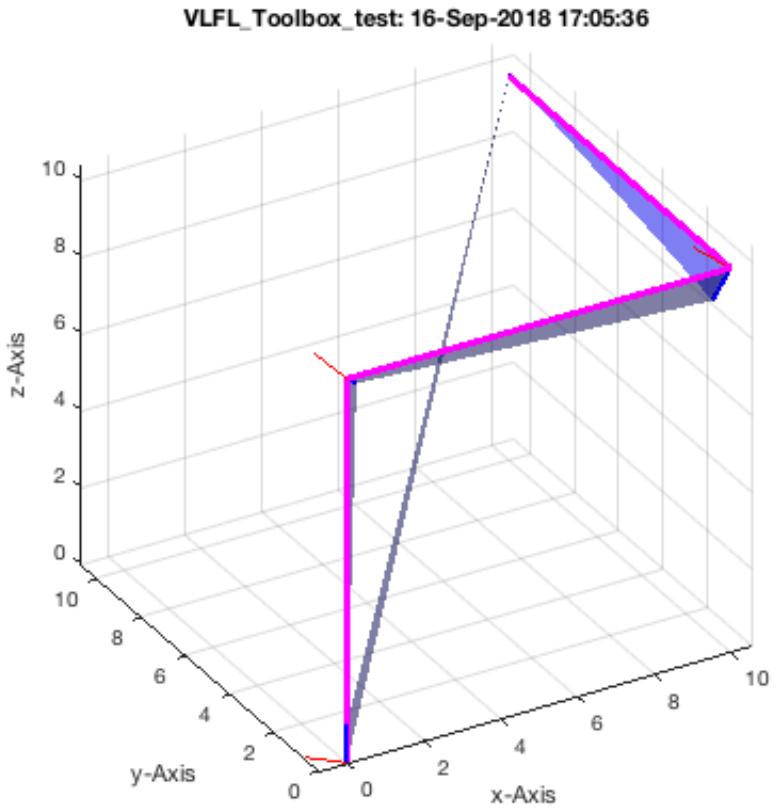
```
VLedgeNormal(VLsample(2));
```



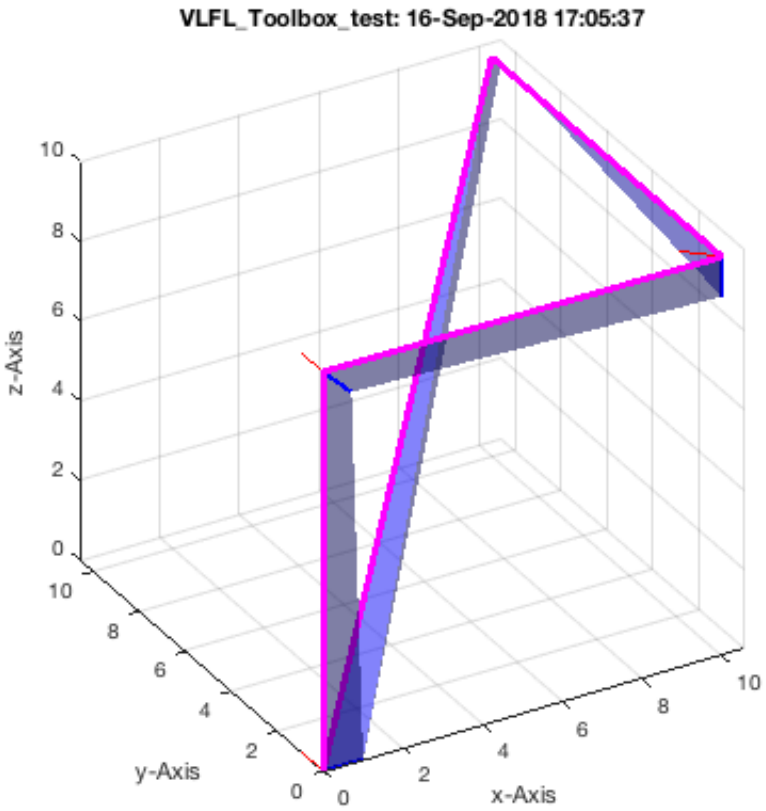

```
VLedgeNormal(VLsample(3));
```



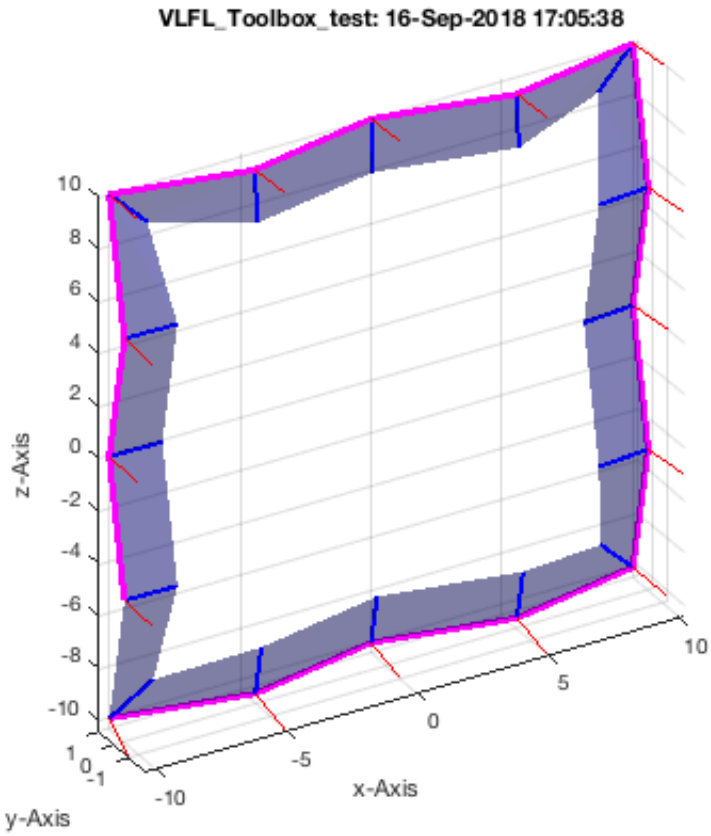

```
VLedgeNormal(VLsample(7));
```



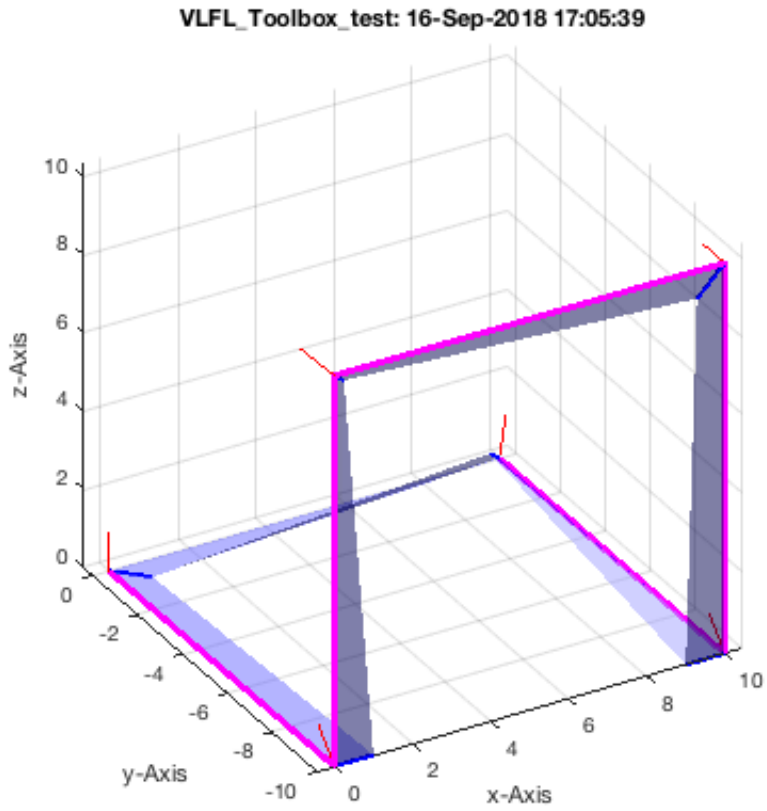

```
VLedgeNormal(VLsample(8));
```



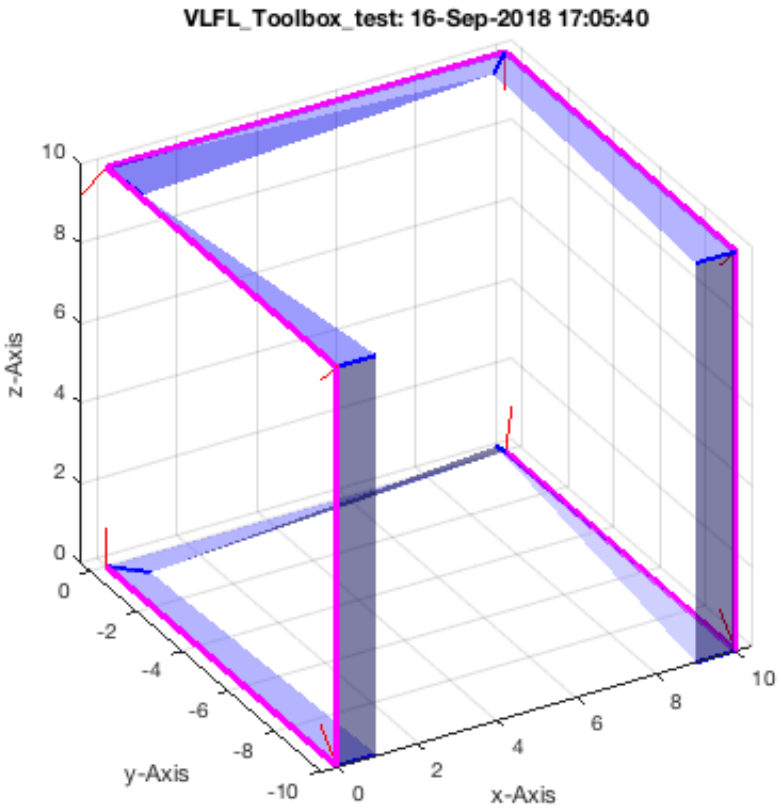

```
VLedgeNormal(VLsample(12));
```



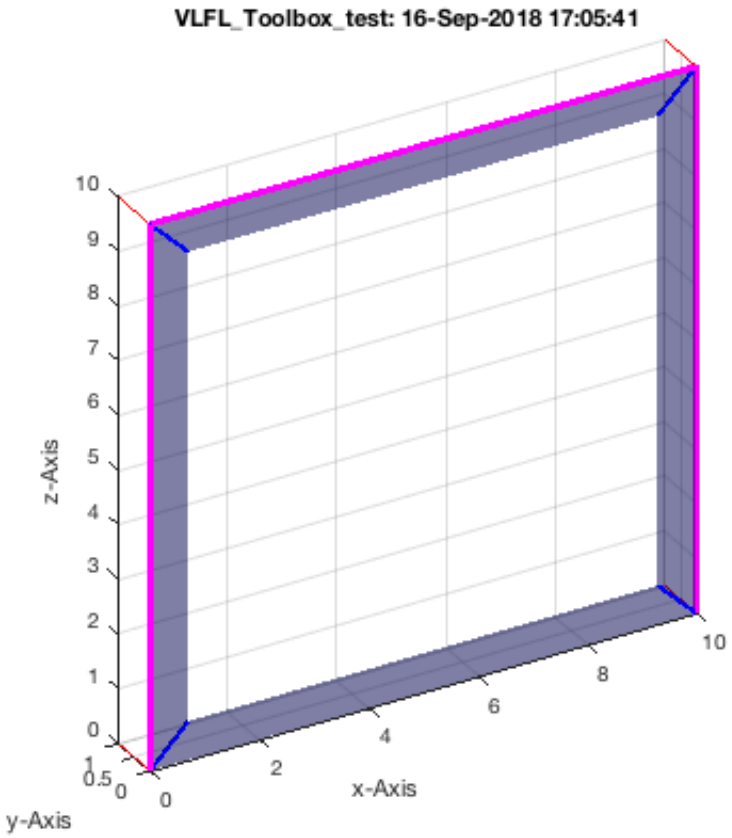

```
VLedgeNormal(VLsample(13));
```



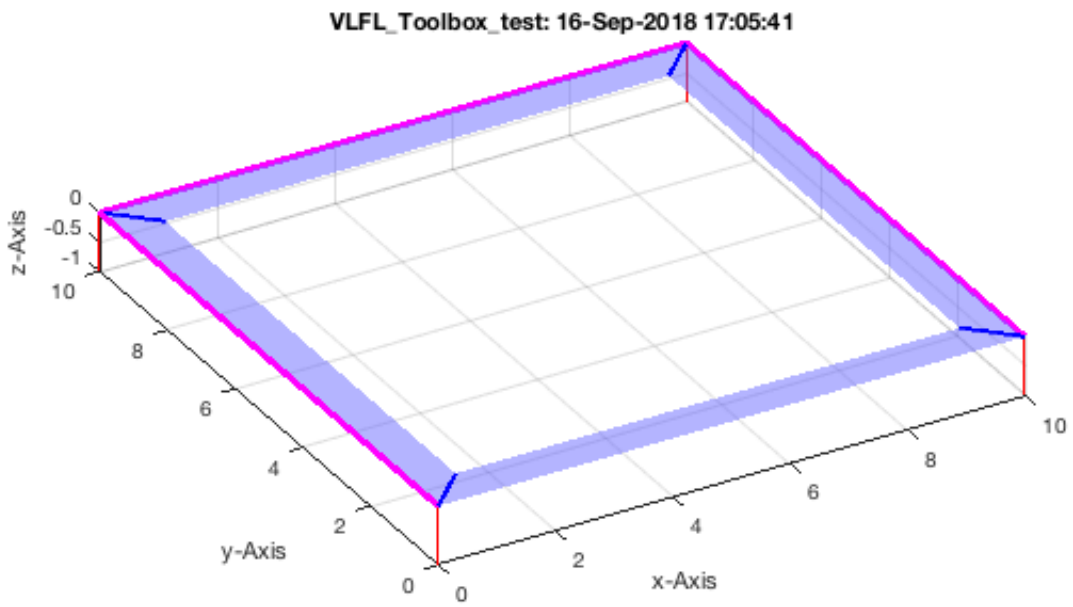

```
VLedgeNormal(VLsample(14));
```



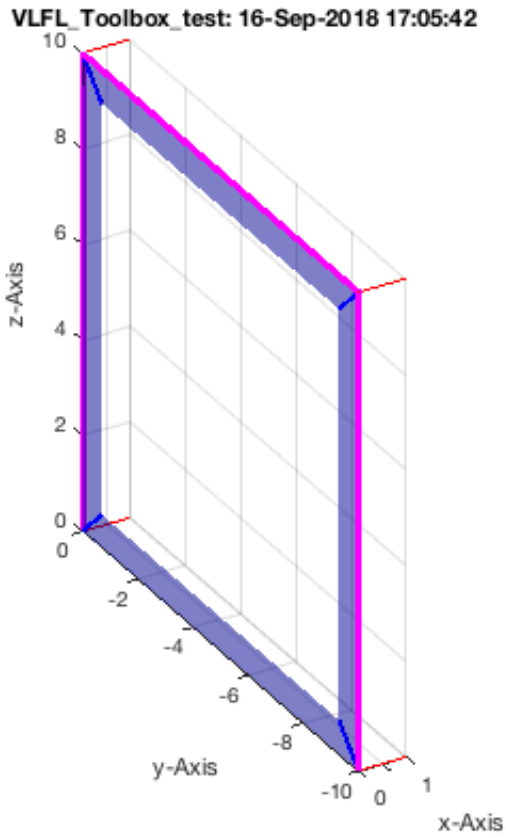

```
VLedgeNormal(VLsample(20));
```




```
VLedgeNormal(VLsample(21));
```



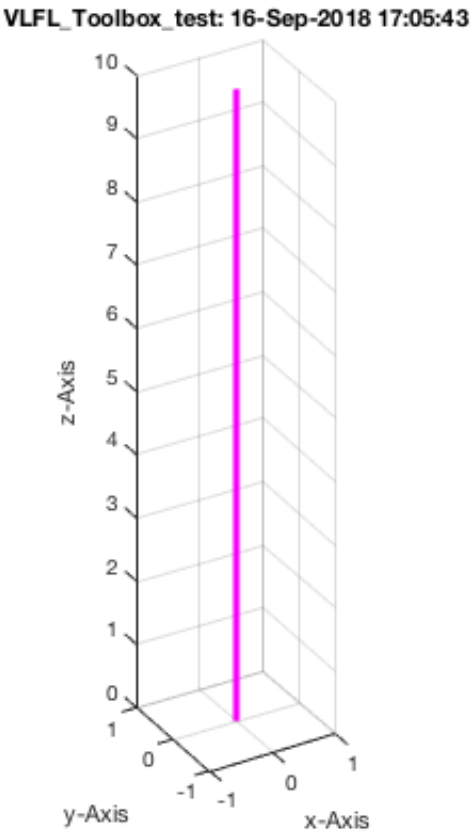

```
VLedgeNormal(VLsample(22));
```



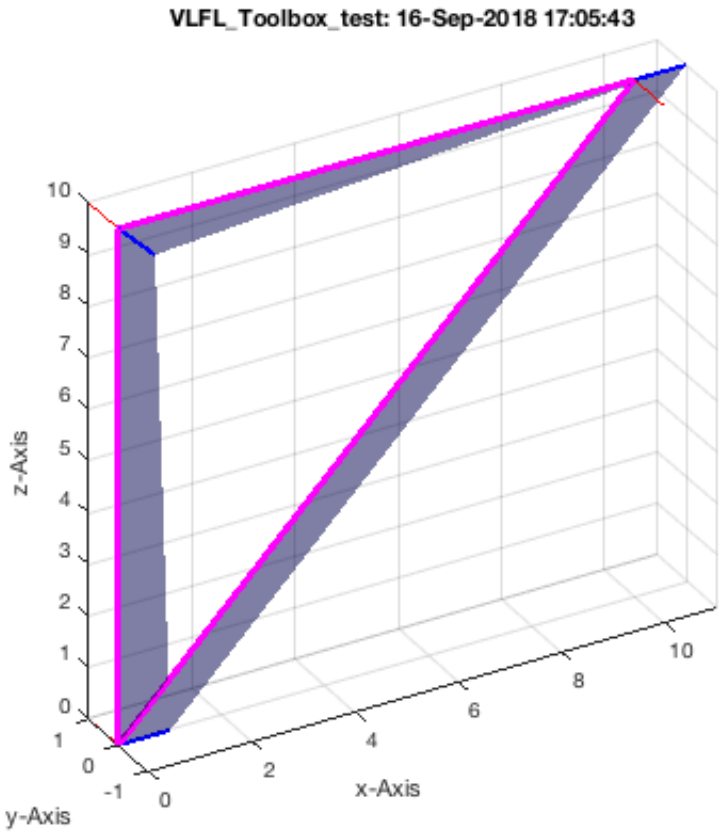
Creating normal function for a closed spatial curve

If angles are larger than 90 degree (pi/2)

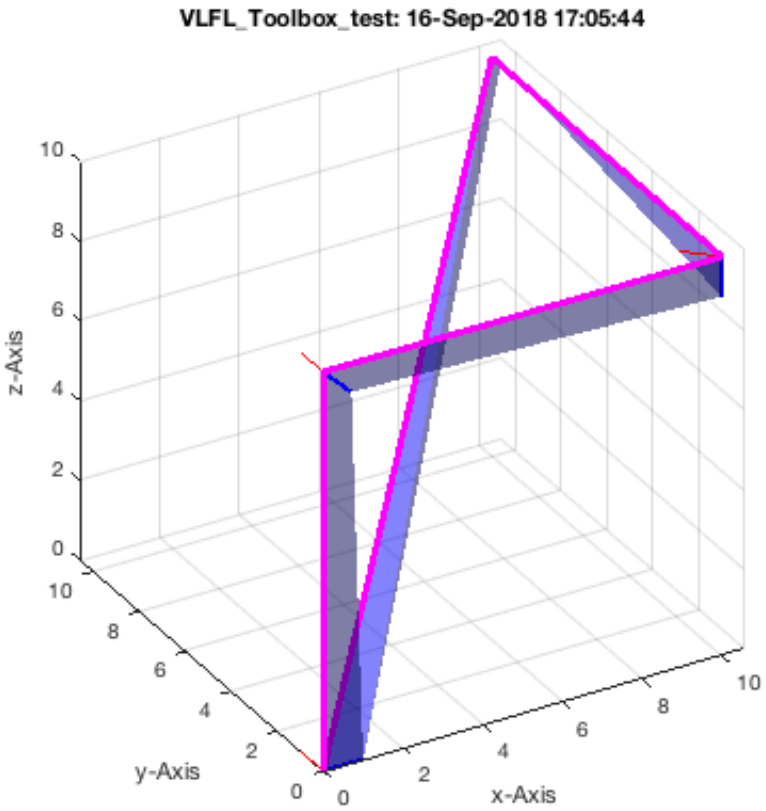
```
VLedgeNormal(CVLofVL(VLsample(2)));
```



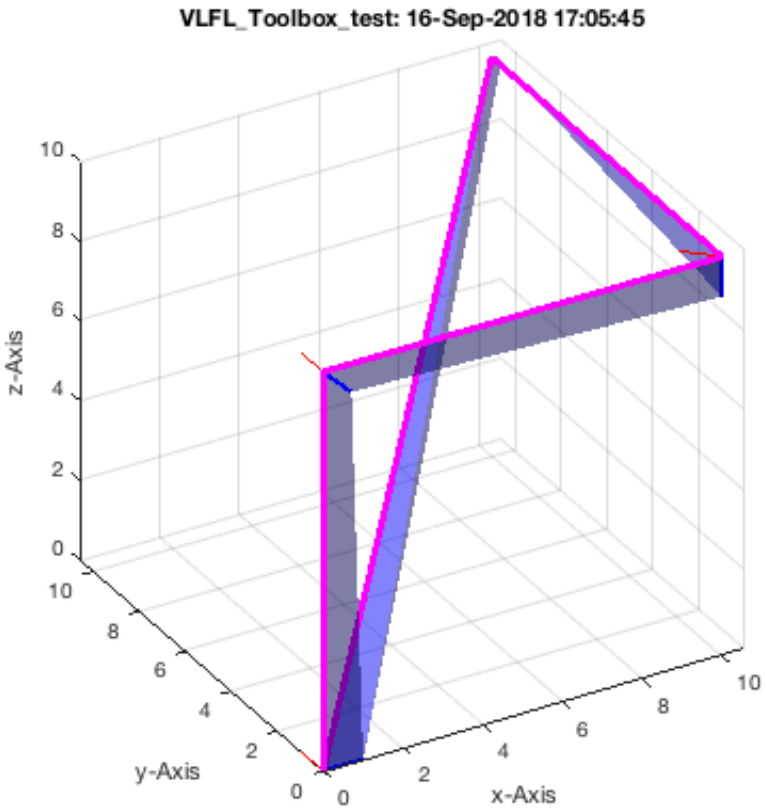

```
VLedgeNormal(CVLofVL(VLsample(3)));
```



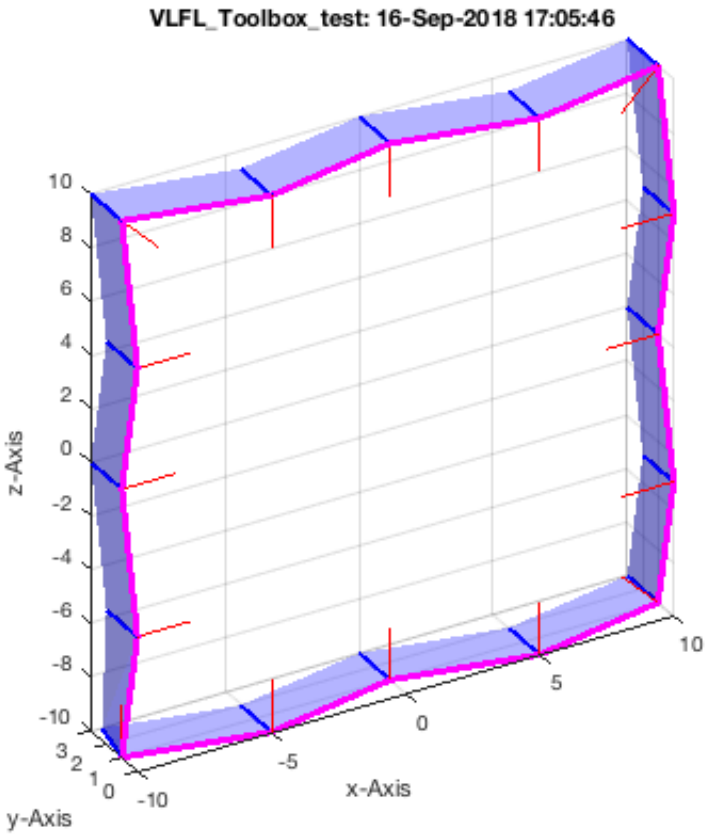

```
VLedgeNormal(CVLofVL(VLsample(7)));
```



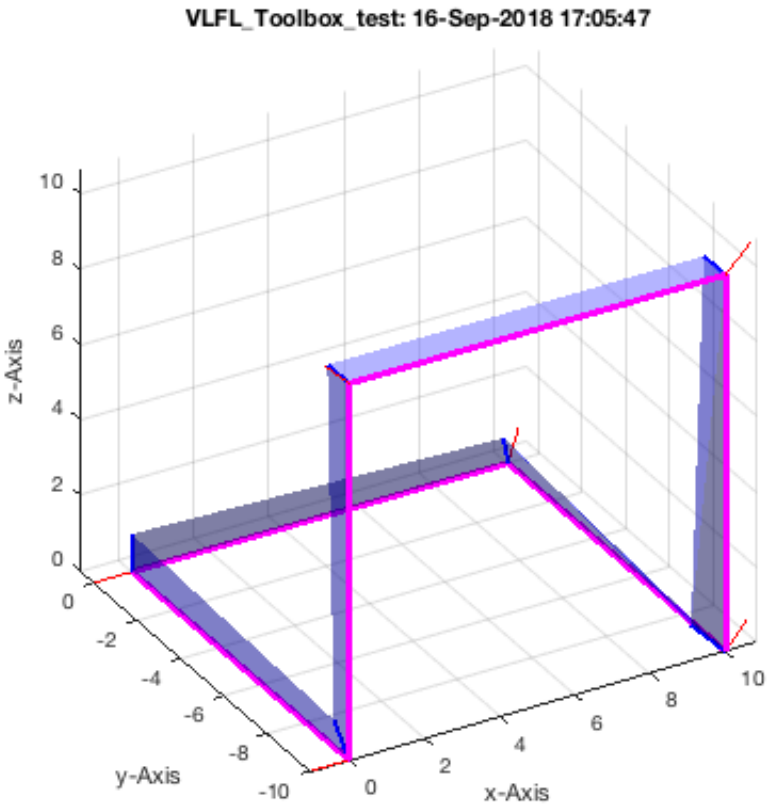

```
VLedgeNormal(CVLofVL(VLsample(8)));
```



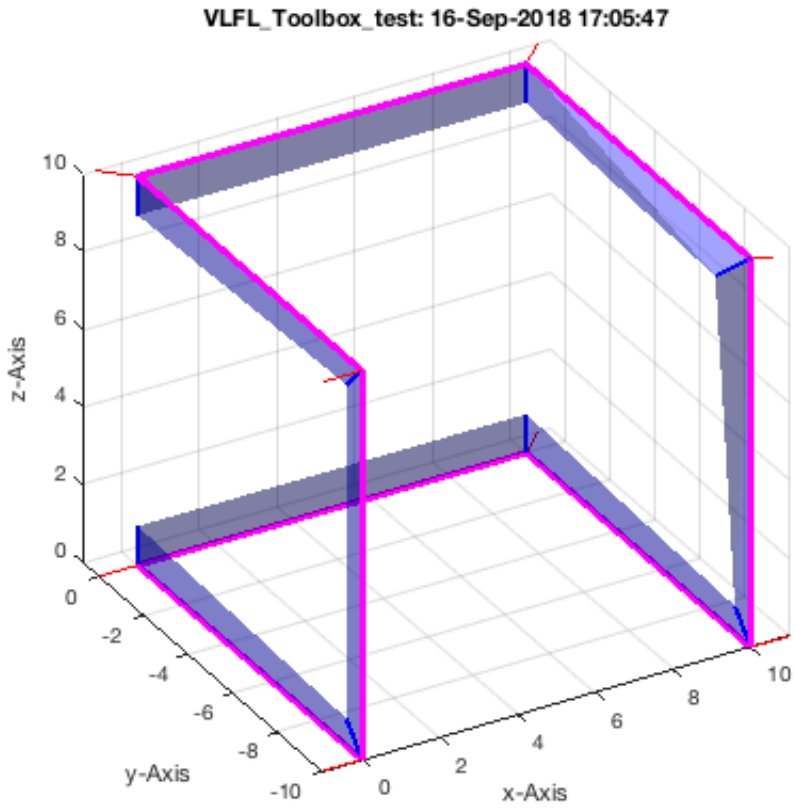

```
VLedgeNormal(CVLofVL(VLsample(12)));
```



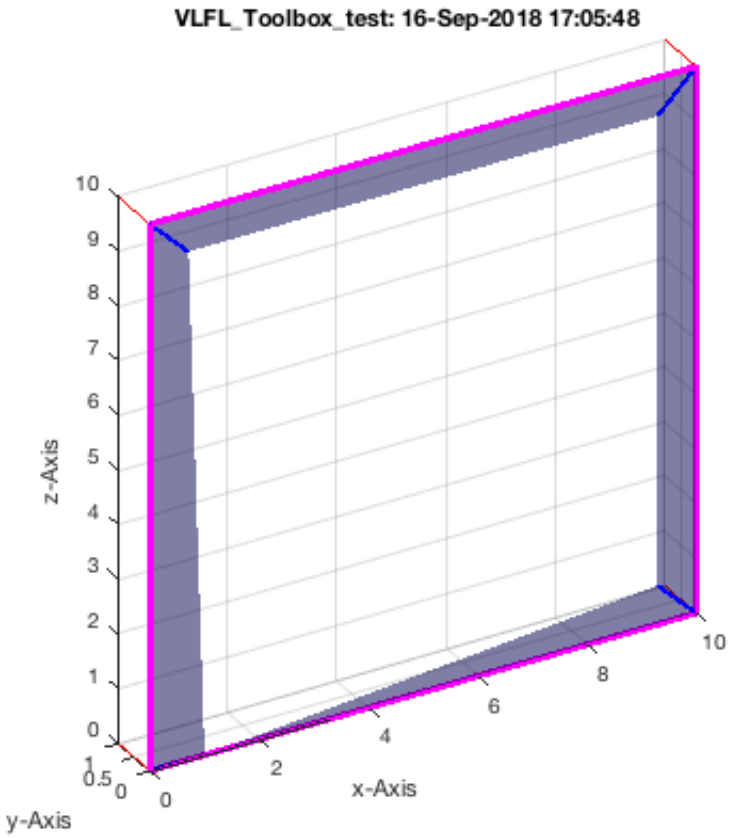

```
VLedgeNormal(CVLofVL(VLsample(13)));
```



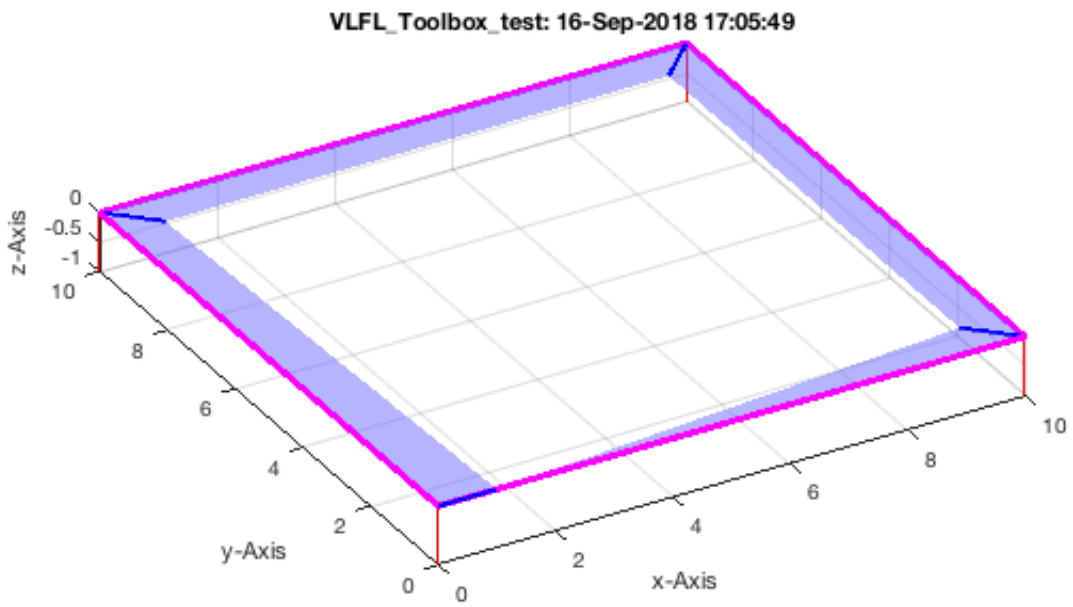

```
VLedgeNormal(CVLofVL(VLsample(14)));
```



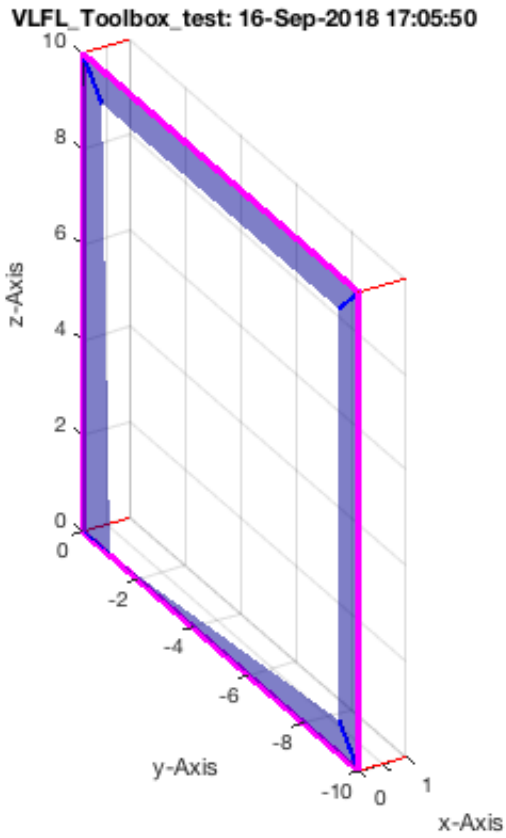

```
VLedgeNormal(CVLofVL(VLsample(20)));
```




```
VLedgeNormal(CVLofVL(VLsample(21)));
```



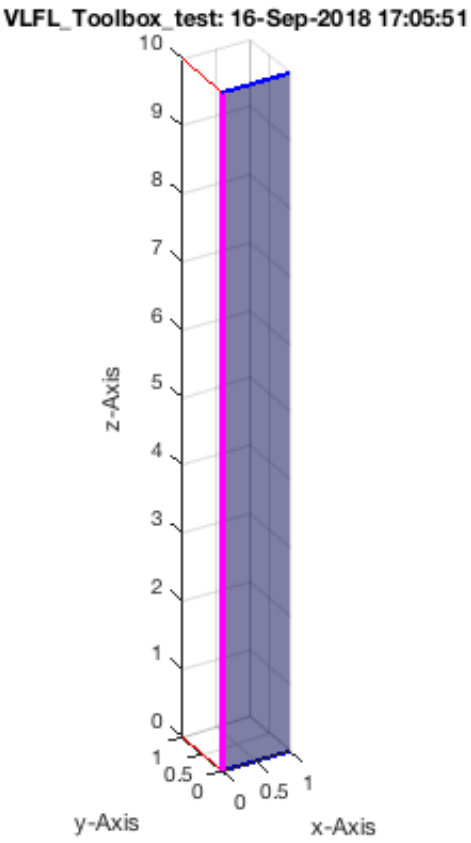

```
VLedgeNormal(CVLofVL(VLsample(22)));
```



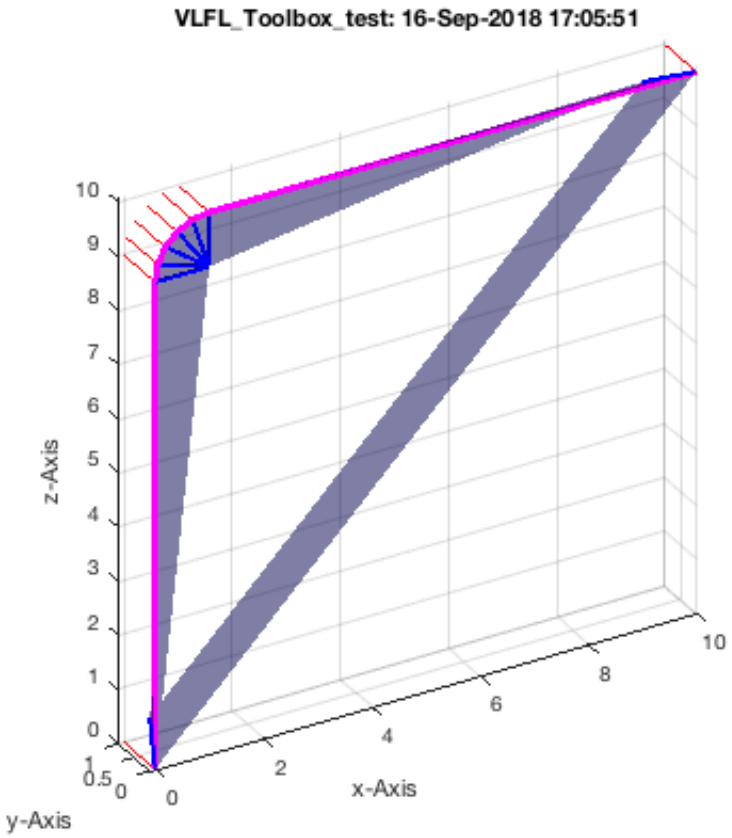
Creating normal function for open spatial radial curve

If angles are larger than 90 degree (pi/2)

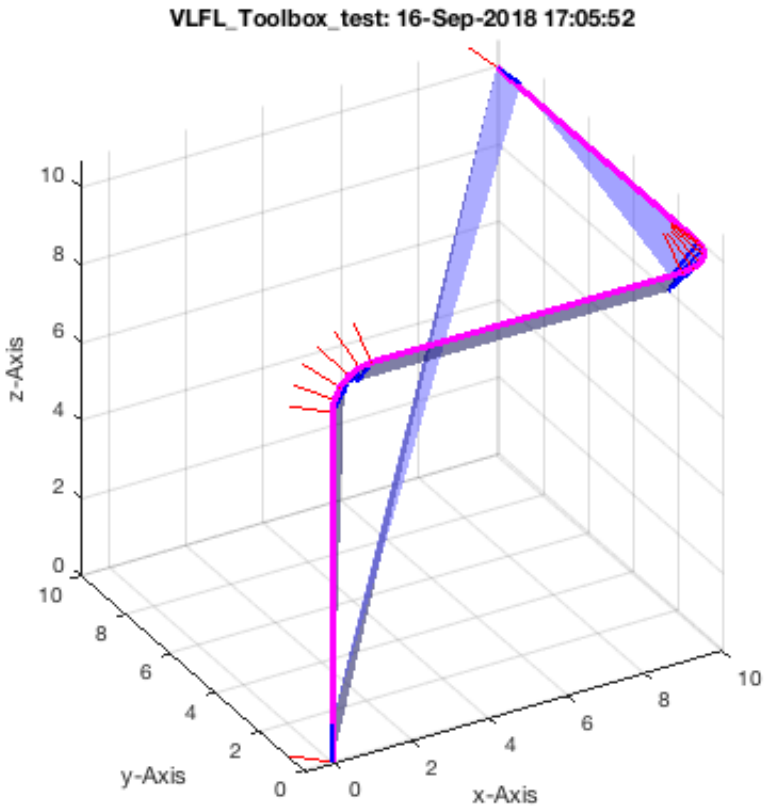
```
VLedgeNormal(VLradialEdges(VLsample(2)));
```



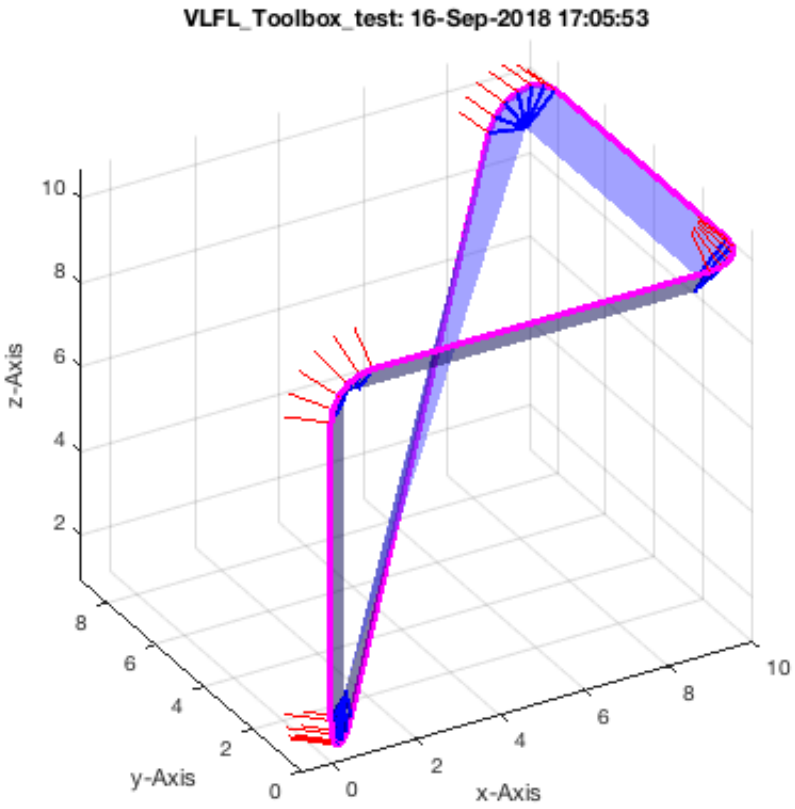

```
VLedgeNormal(VLradialEdges(VLsample(3)));
```



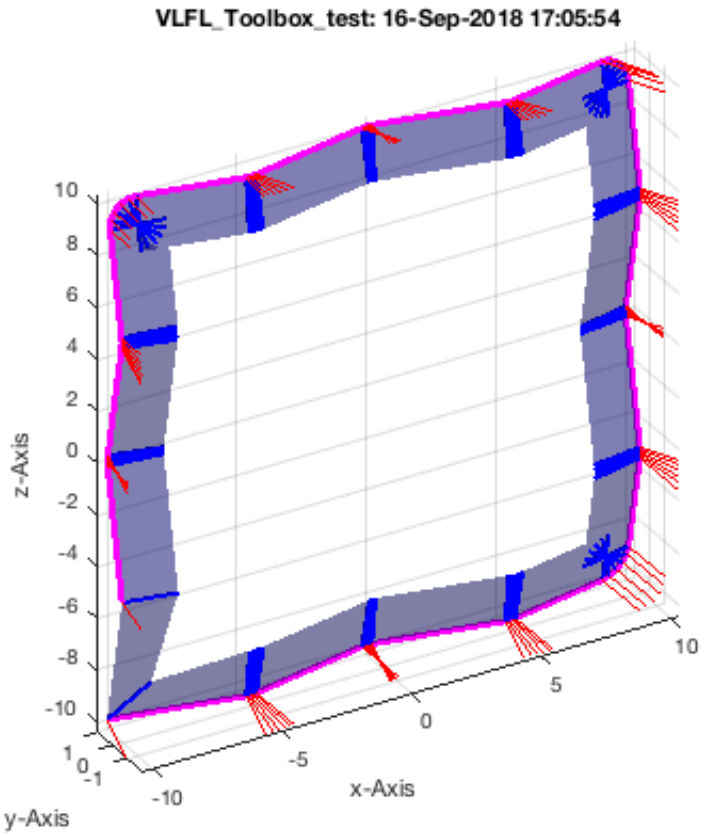

```
VLedgeNormal(VLradialEdges(VLsample(7)));
```



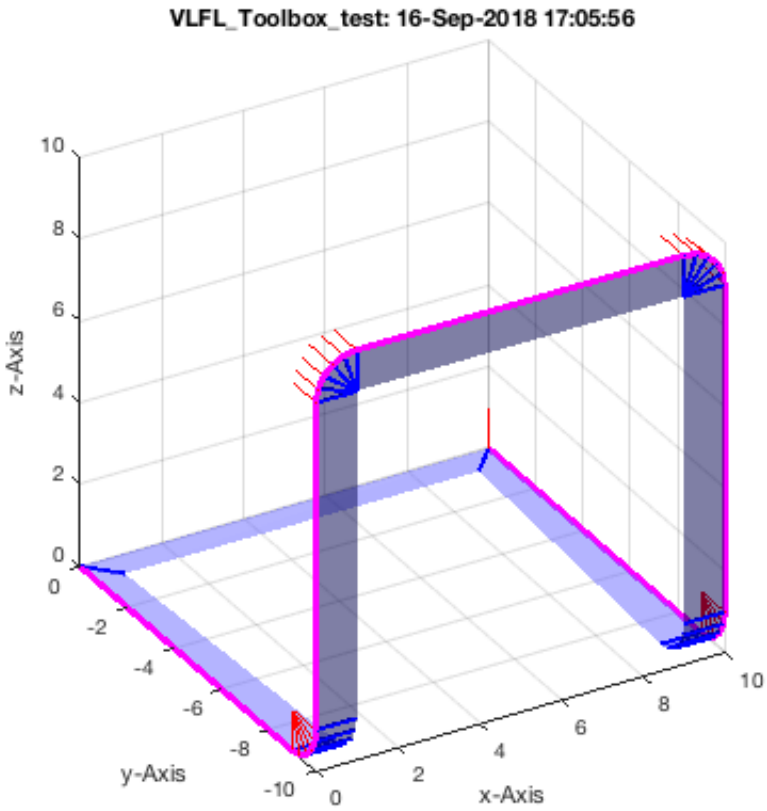

```
VLedgeNormal(VLradialEdges(VLsample(8)));
```



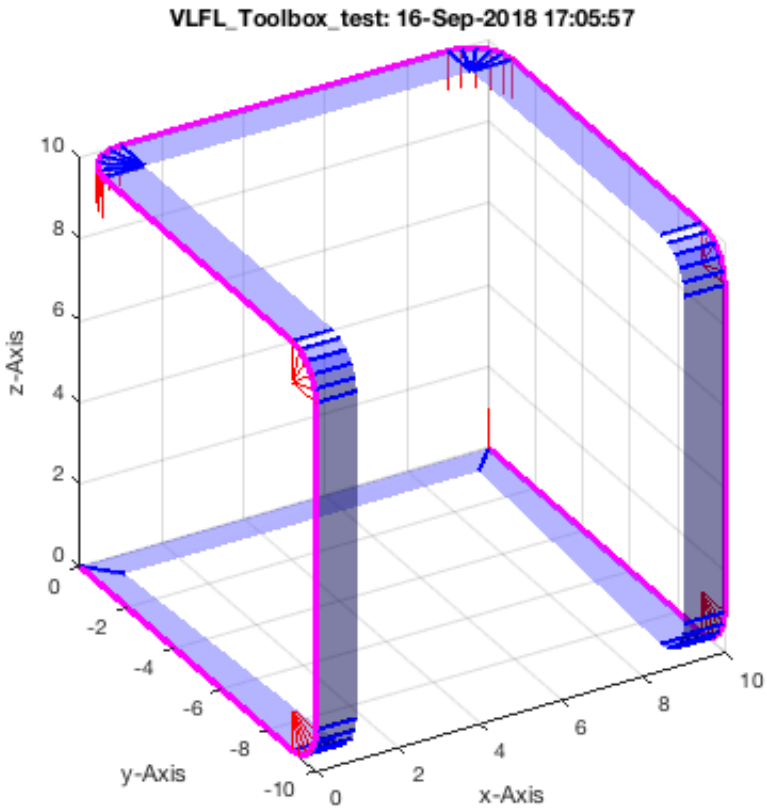

```
VLedgeNormal(VLradialEdges(VLsample(12)));
```



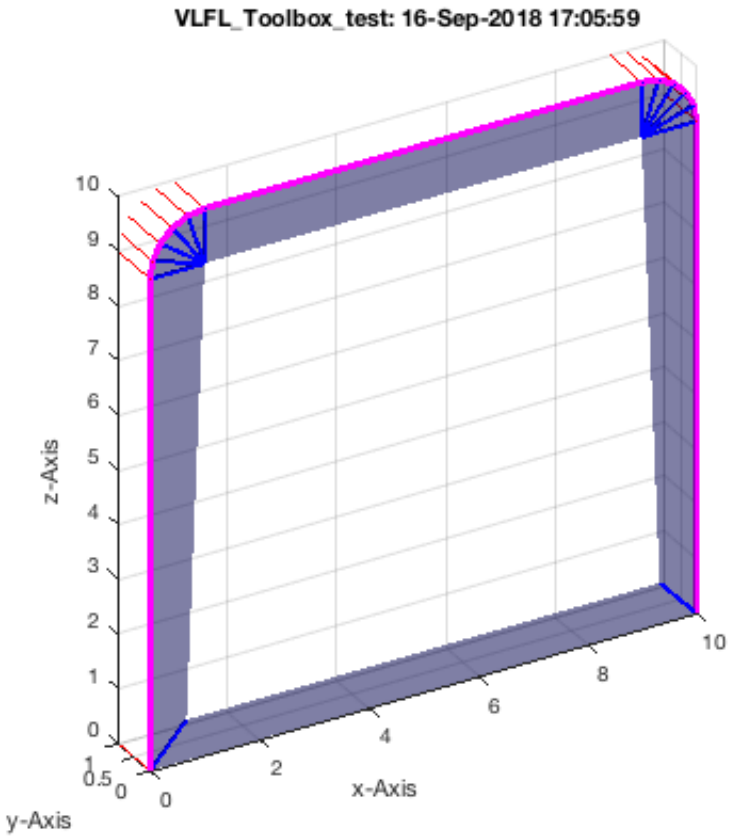

```
VLedgeNormal(VLradialEdges(VLsample(13)));
```



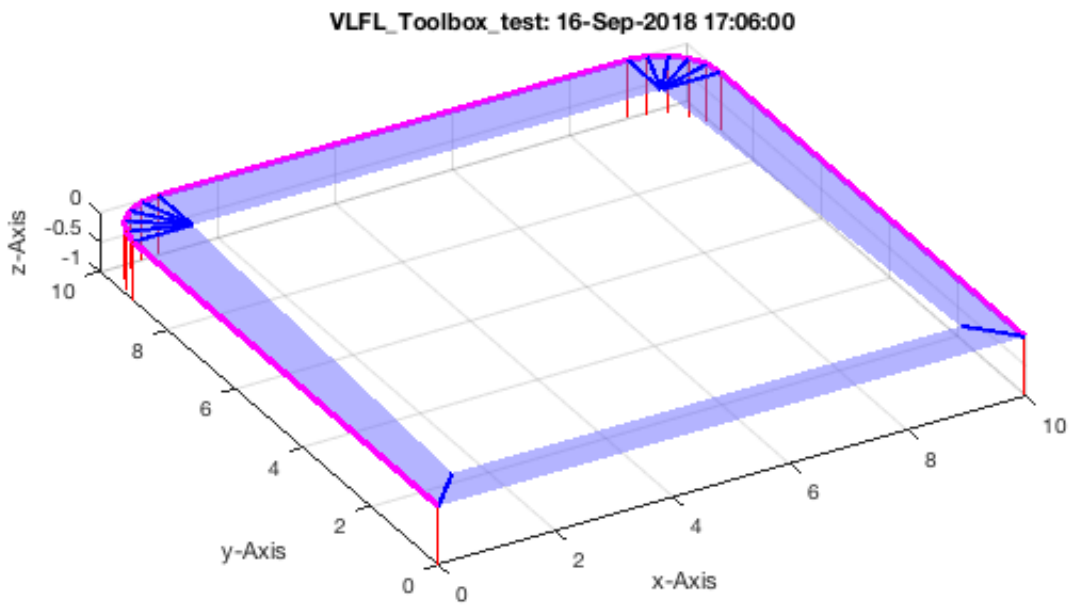

```
VLedgeNormal(VLradialEdges(VLsample(14)));
```



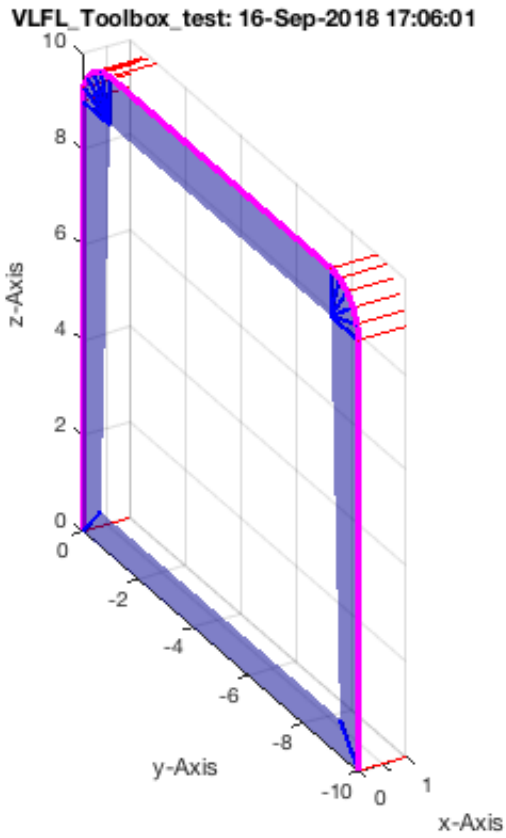

```
VLedgeNormal(VLradialEdges(VLsample(20)));
```




```
VLedgeNormal(VLradialEdges(VLsample(21)));
```



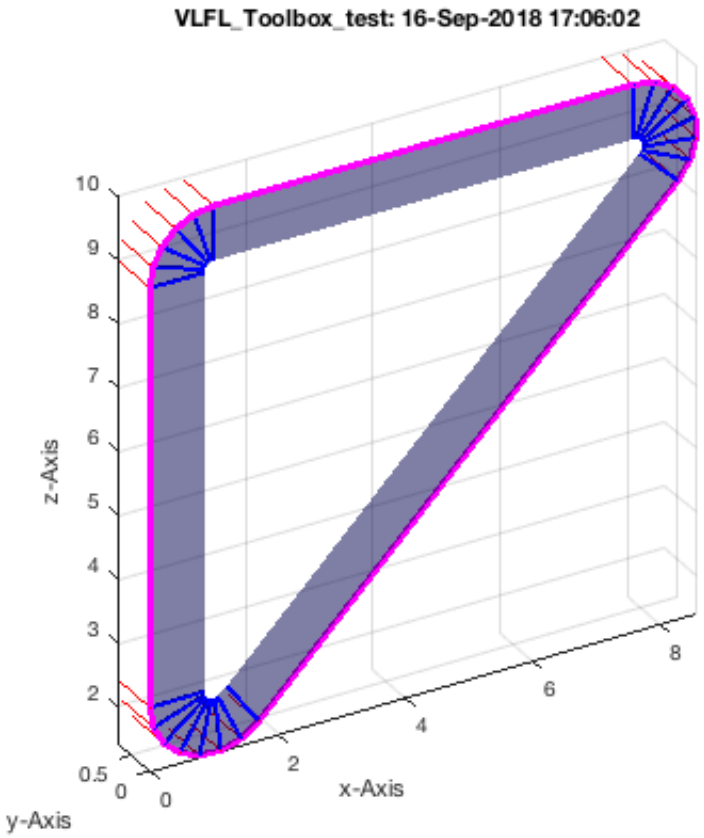

```
VLedgeNormal(VLradialEdges(VLsample(22)));
```



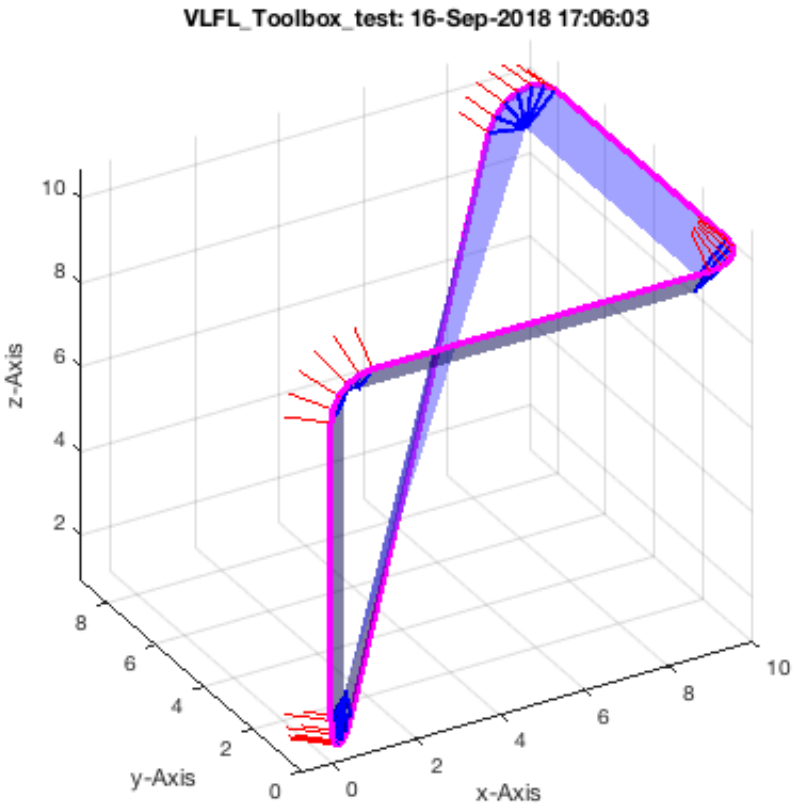
Creating normal function for closed spatial radial curve

If angles are larger than 90 degree (pi/2)

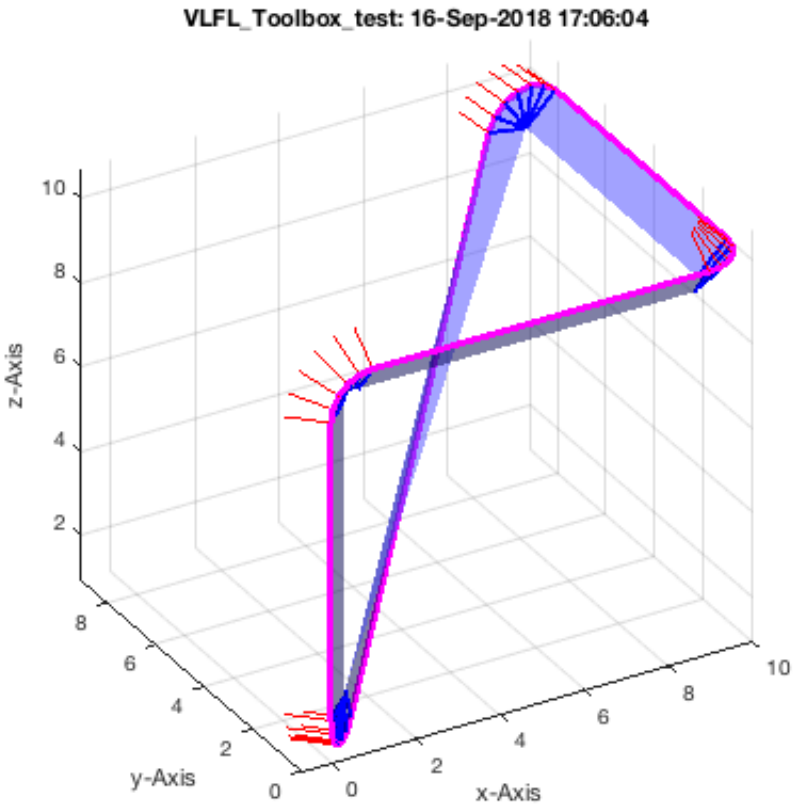
```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(3))));
```



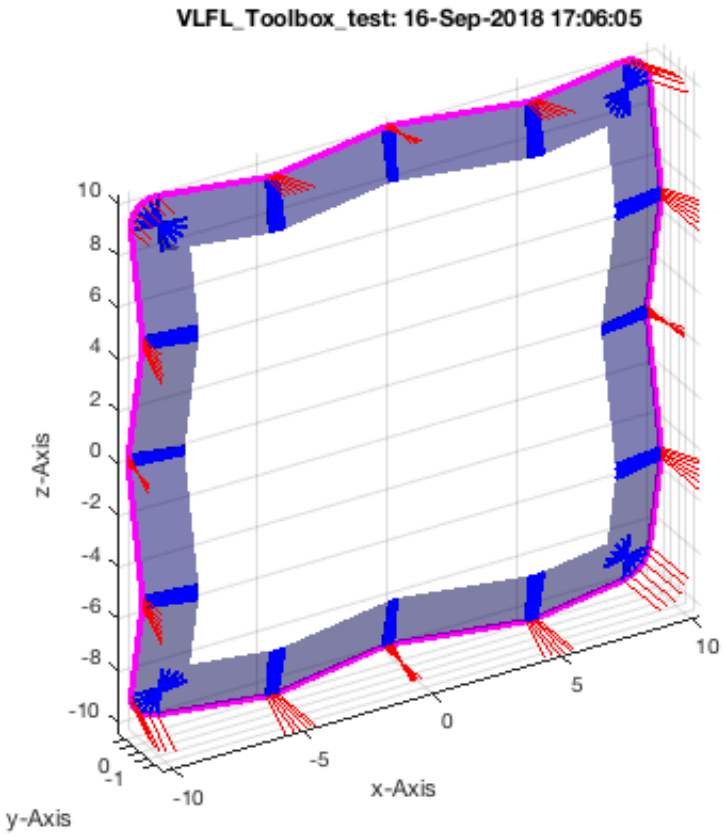

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(7))));
```



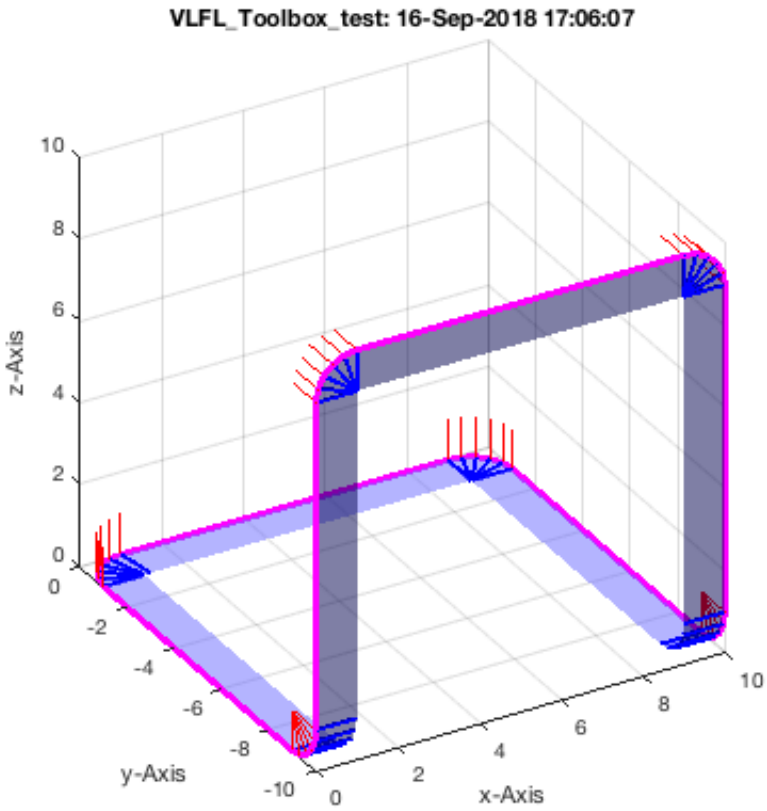

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(8))));
```



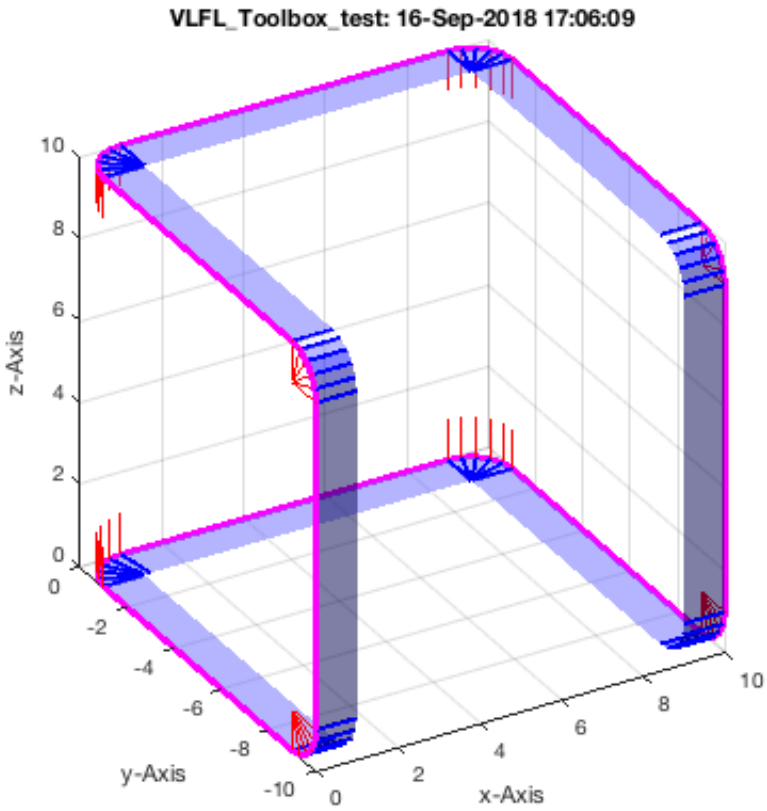

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(12))));
```



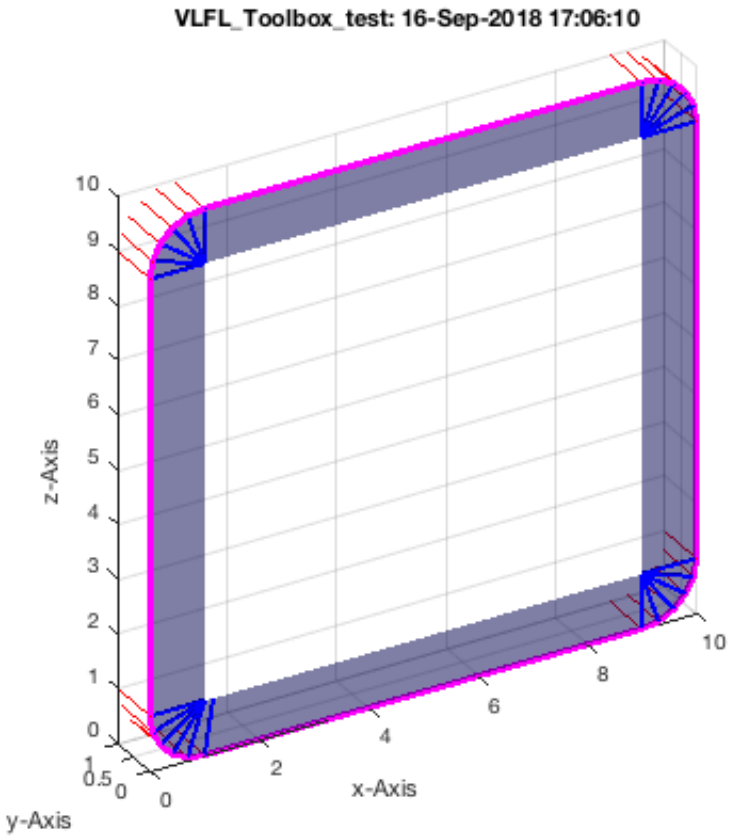

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(13))));
```



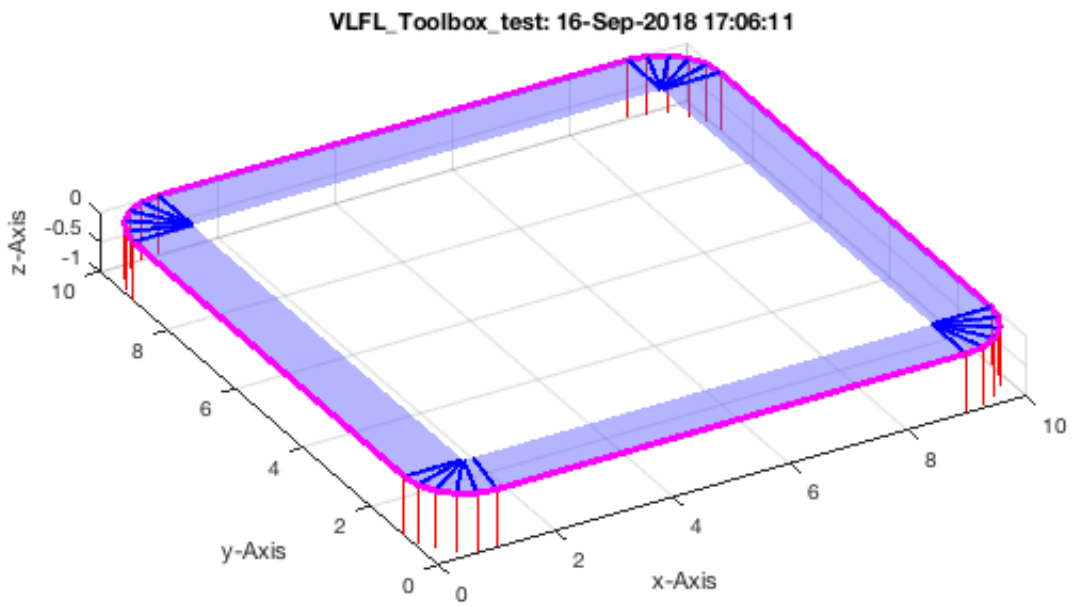

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(14))));
```



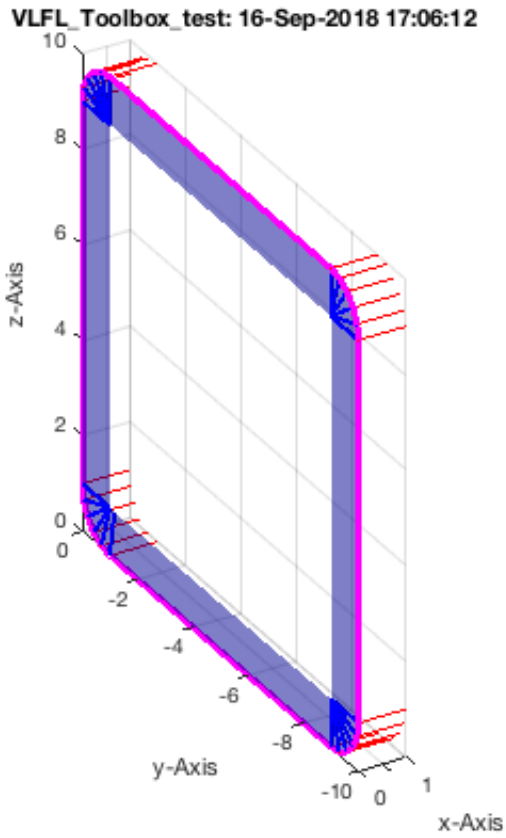

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(20))));
```




```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(21))));
```



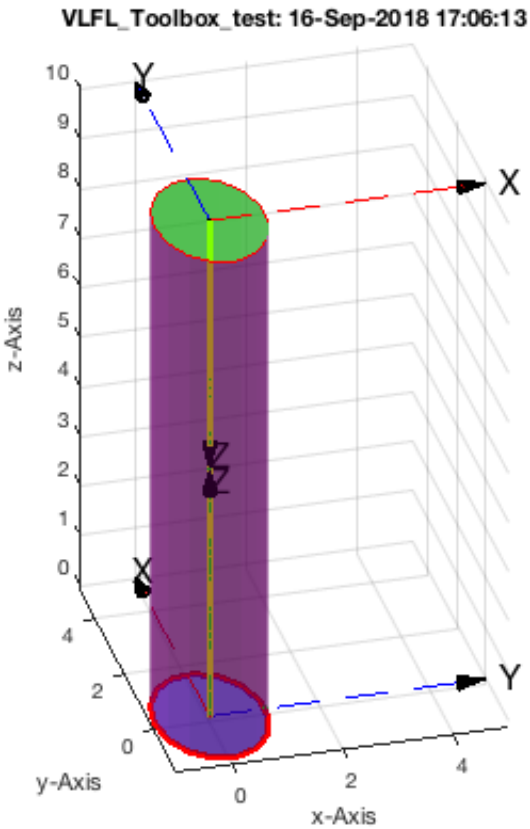

```
VLedgeNormal(VLradialEdges(CVLofVL(VLsample(22))));
```



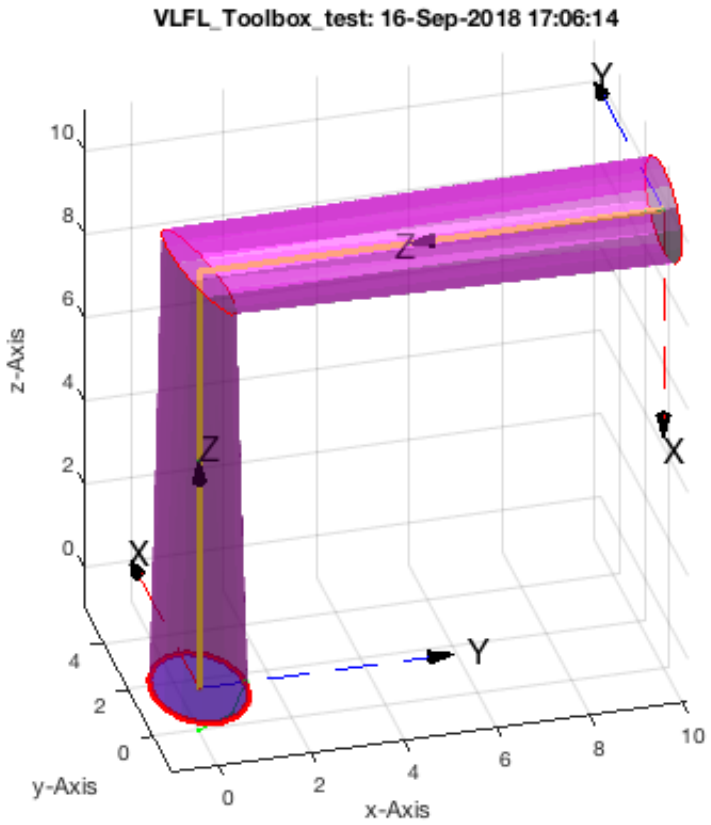
Creating Solid Geometries open

If angles are larger than 90 degree (pi/2)

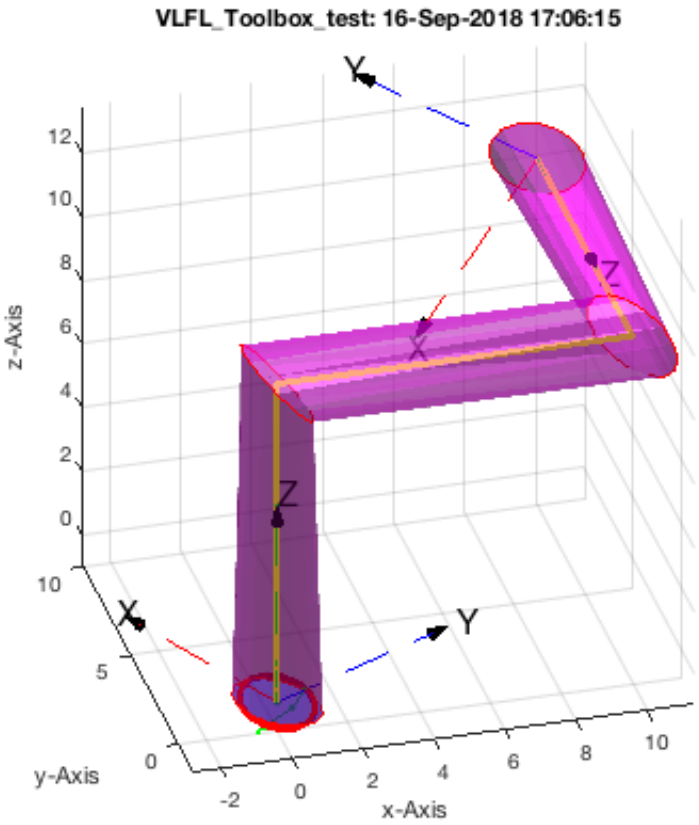
```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(2));
```



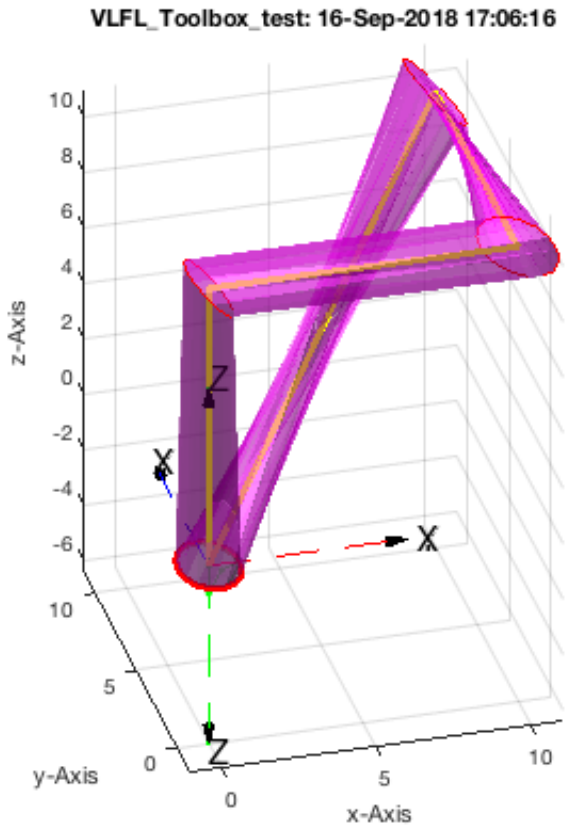

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(3));
```



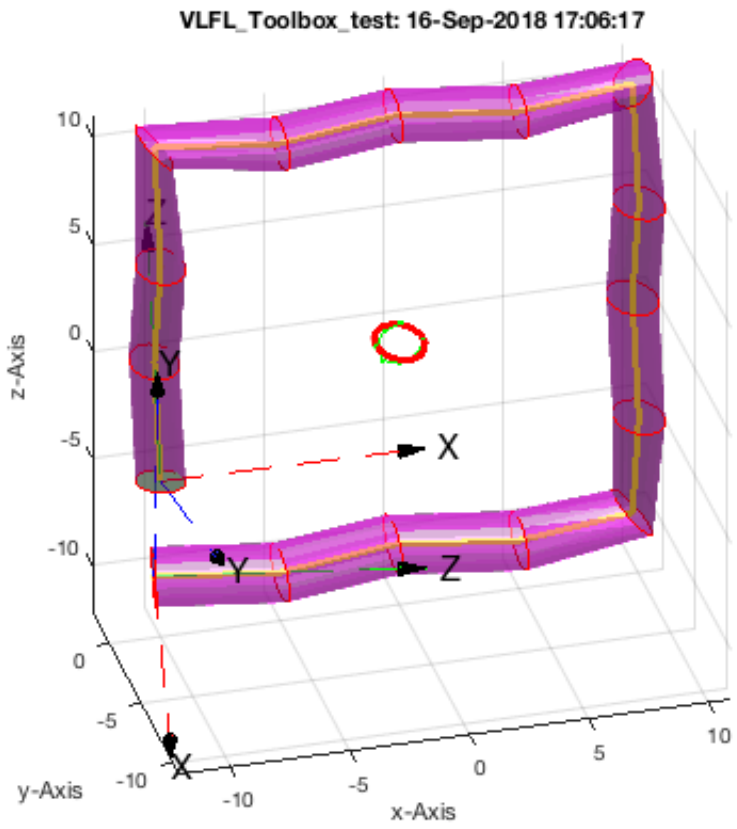

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(7));
```



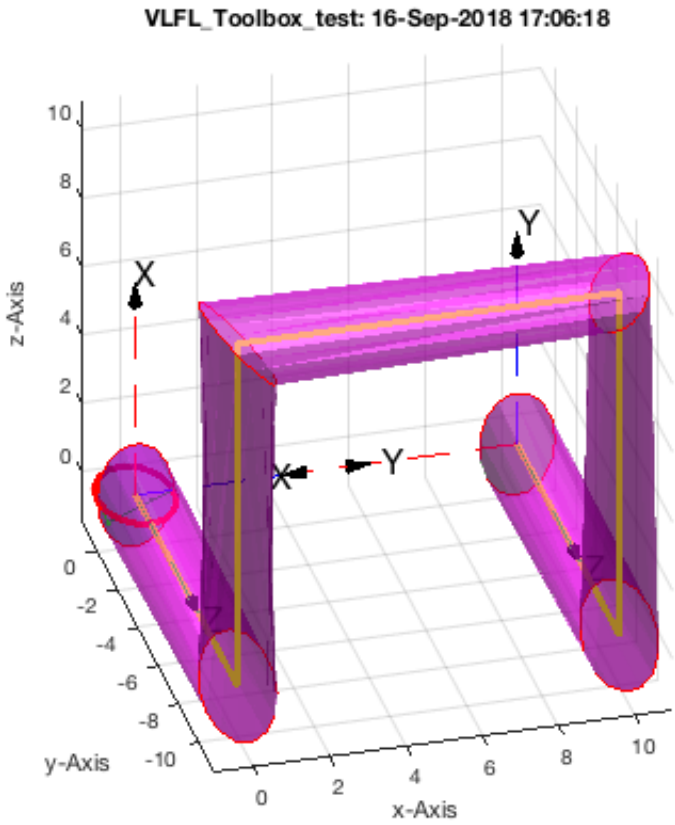

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(8));
```



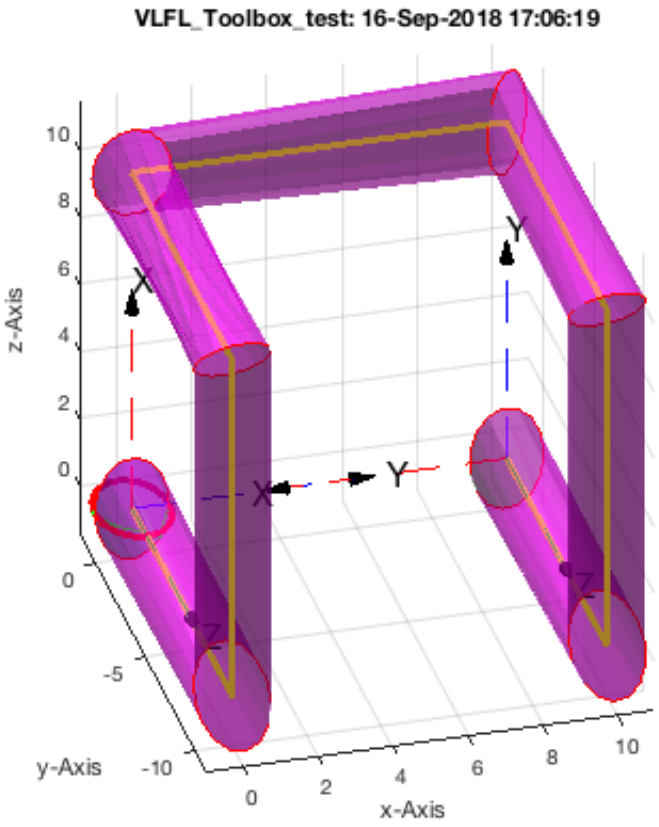

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(12));
```



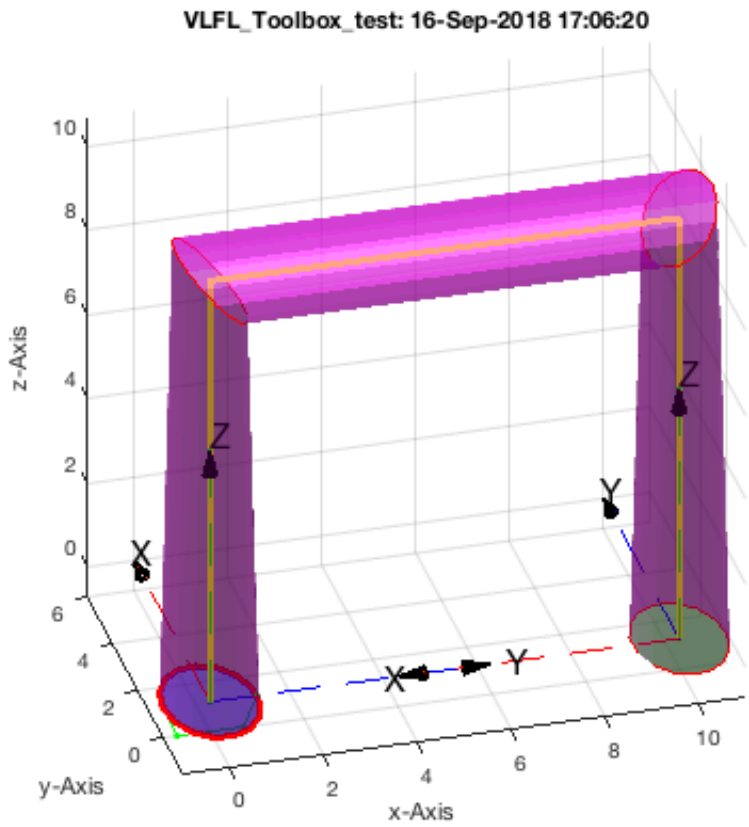

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(13));
```



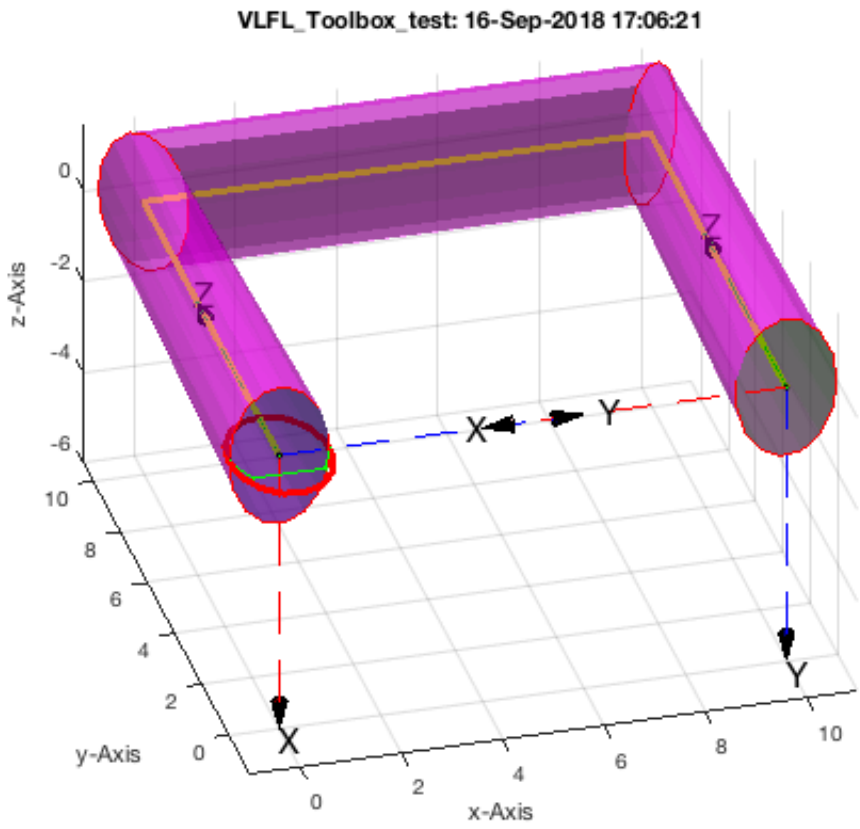

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(14));
```



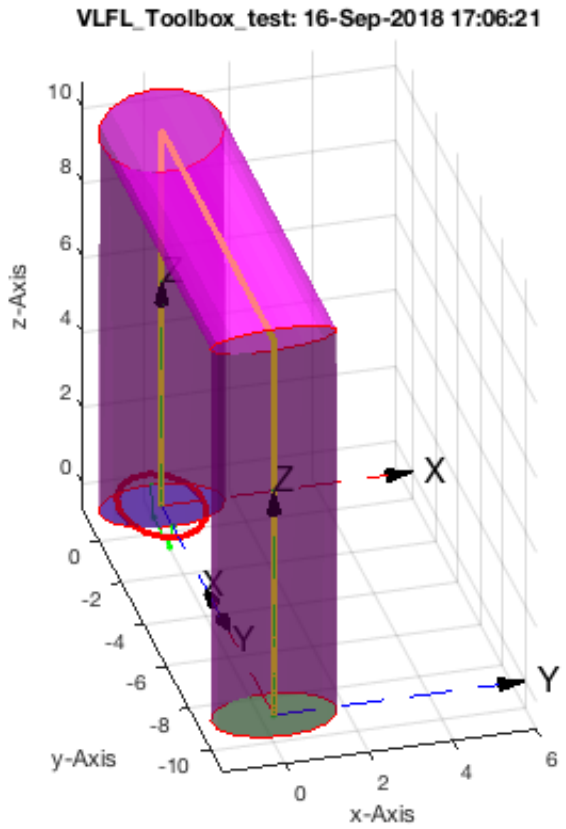

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(20));
```




```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(21));
```



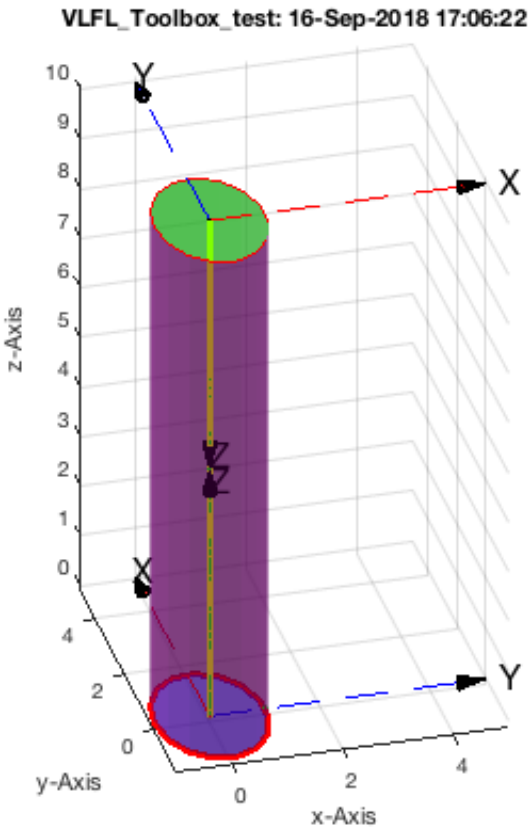

```
SGcontourtube2(PLcircle(1,'','',1.5),VLsample(22));
```



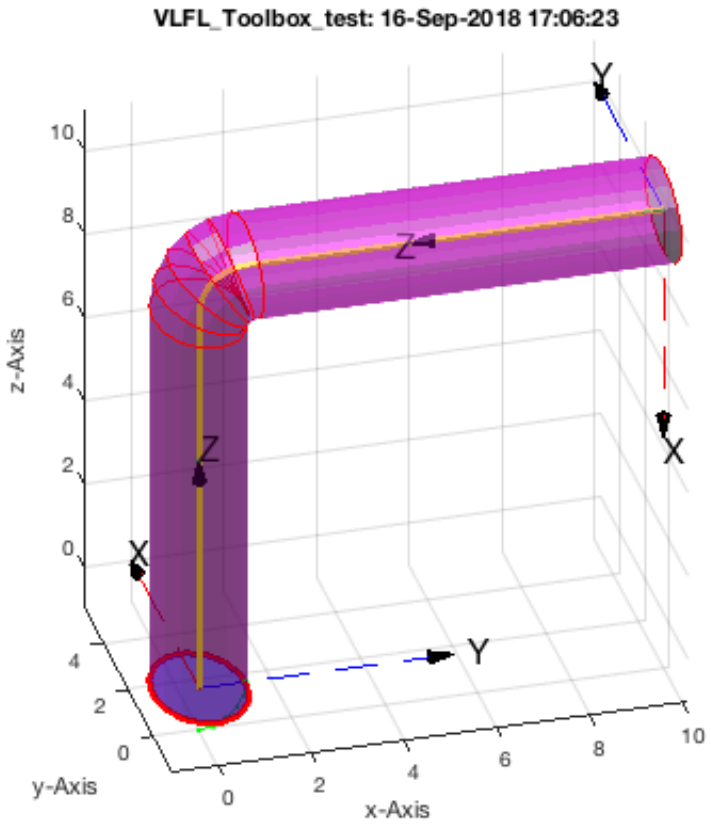
Creating Solid Geometries open

If angles are larger than 90 degree (pi/2)

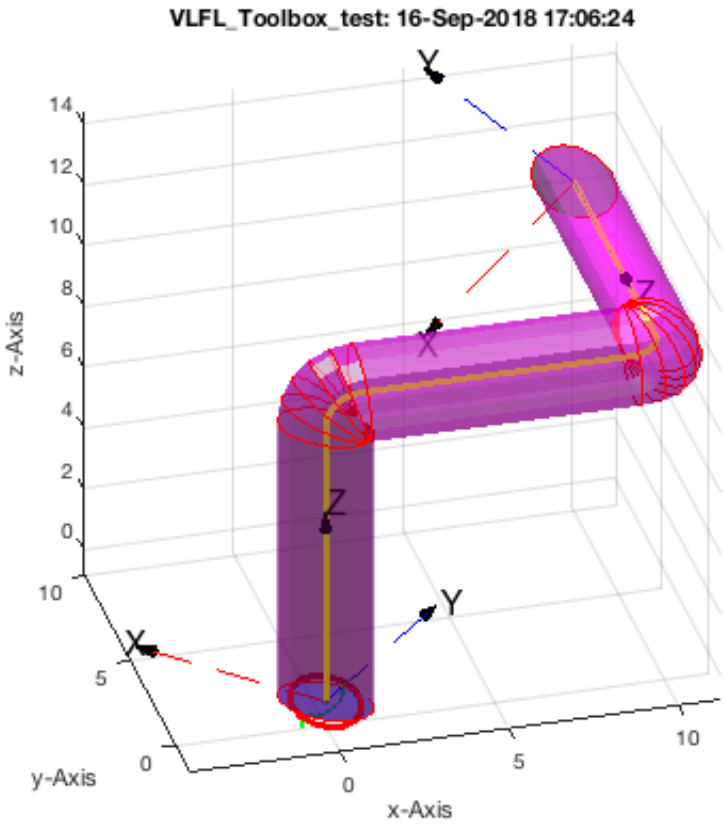
```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(2)));
```



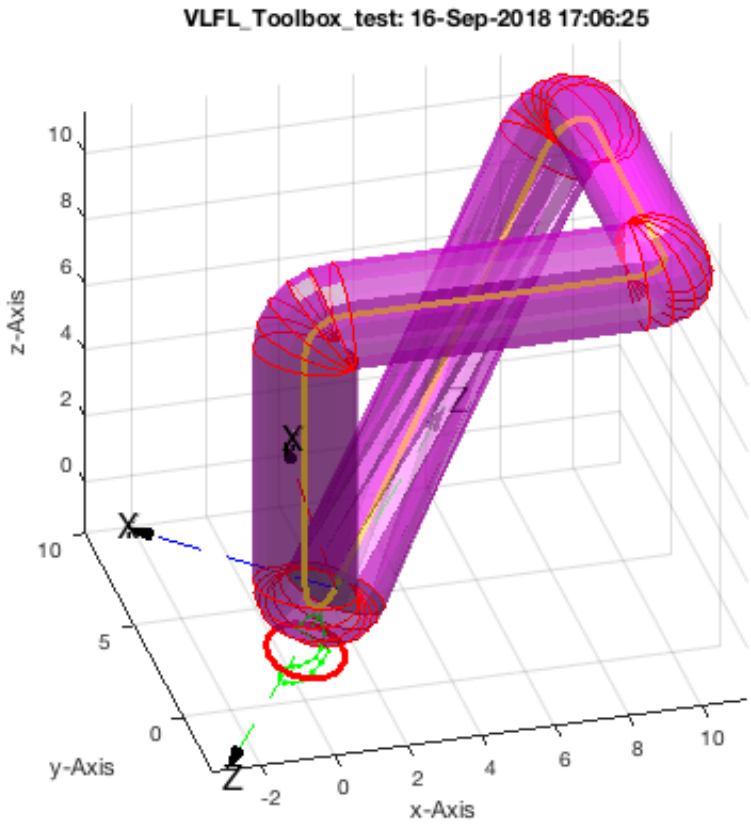

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(3)));
```



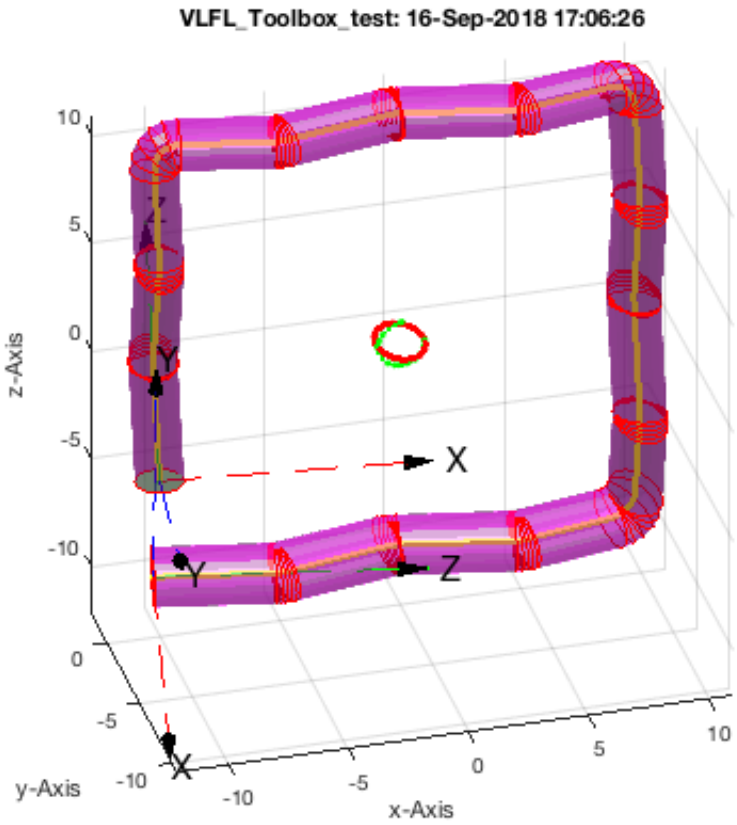

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(7)));
```



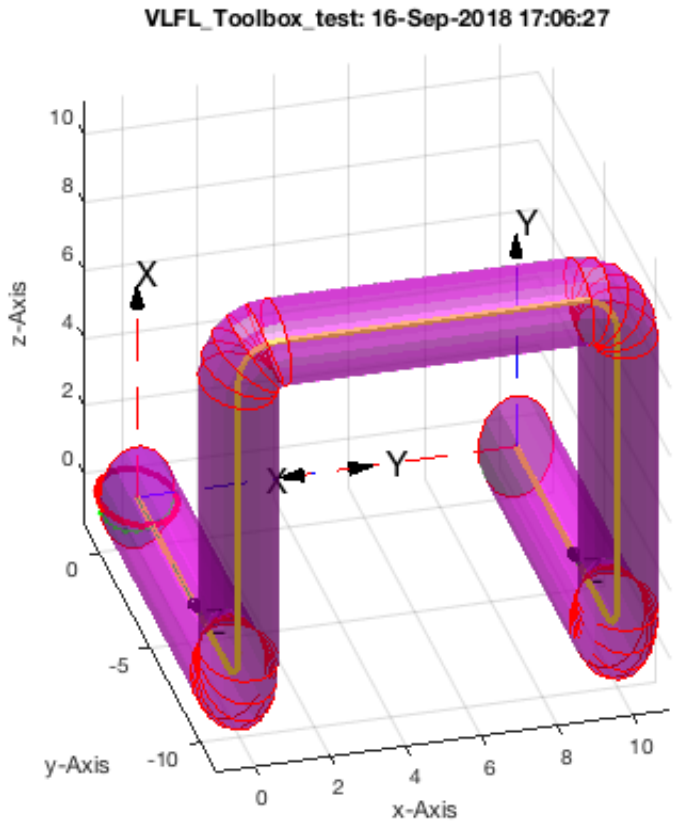

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(8)));
```



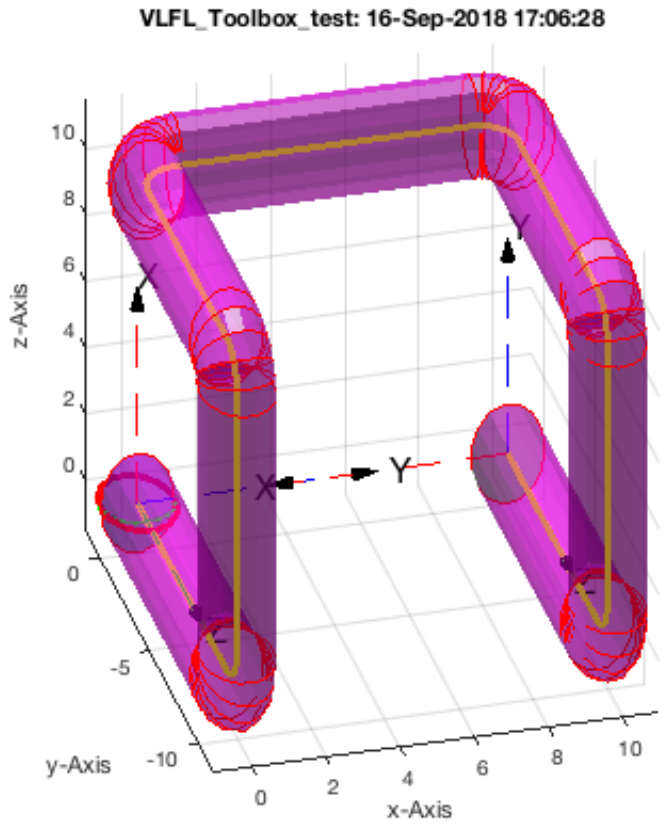

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(12)));
```



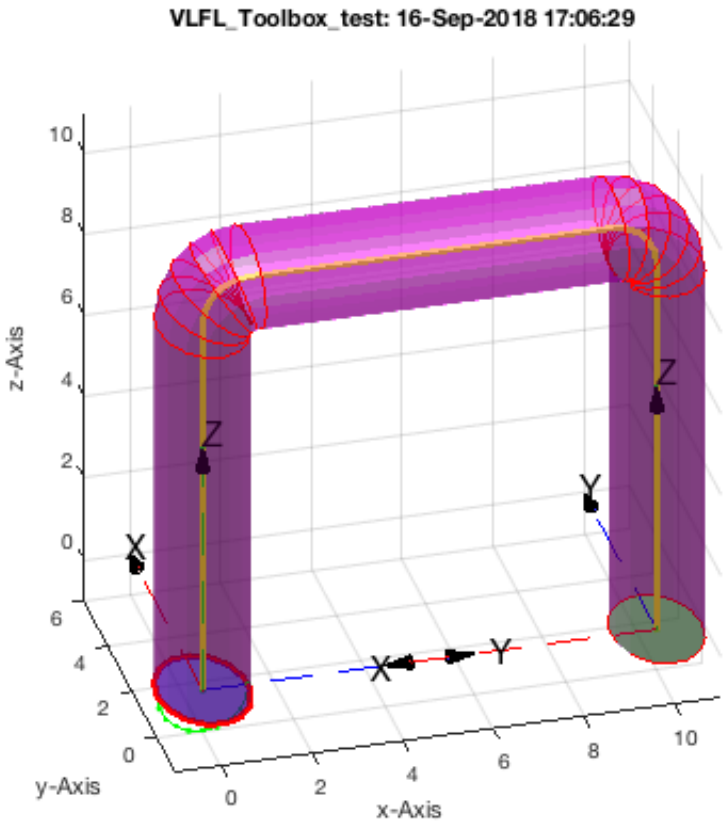

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(13)));
```



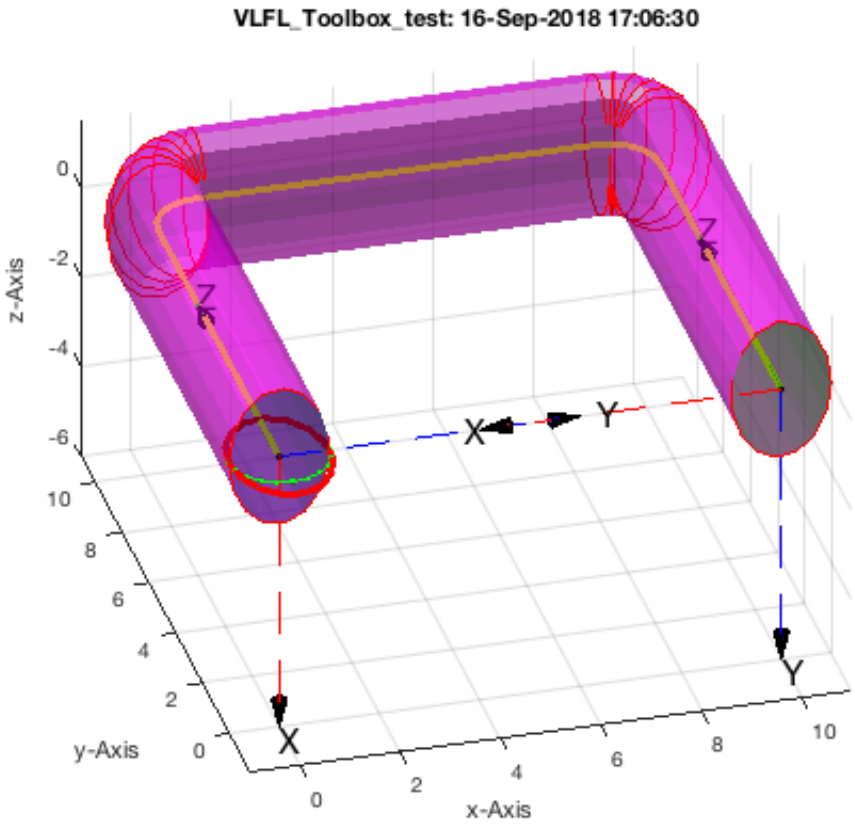

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(14)));
```



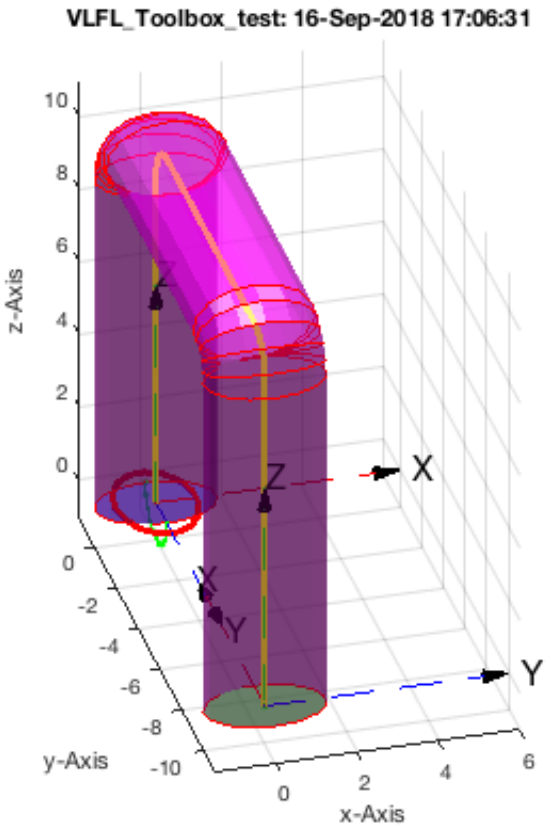

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(20)));
```




```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(21)));
```



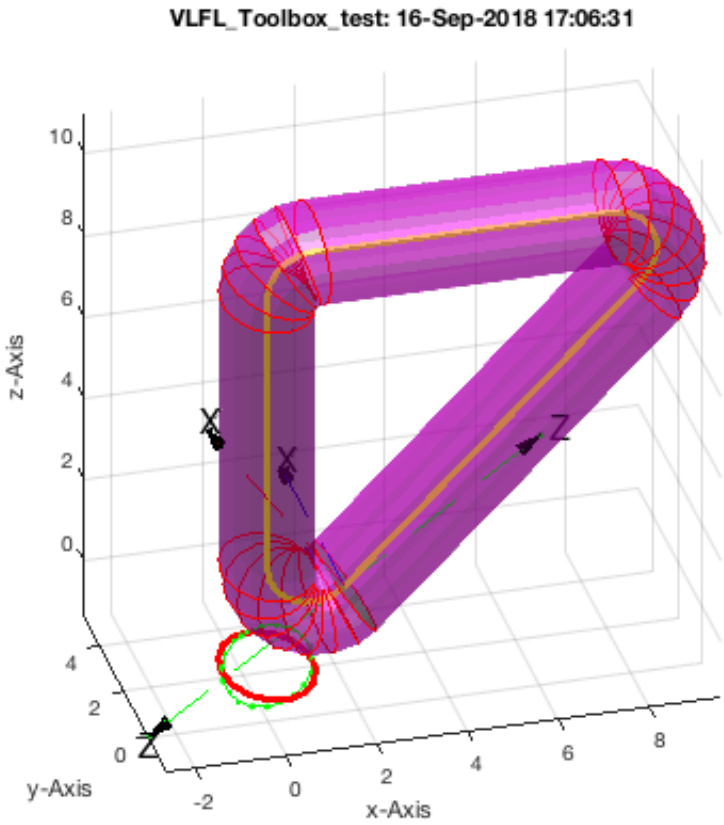

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(VLsample(22)));
```



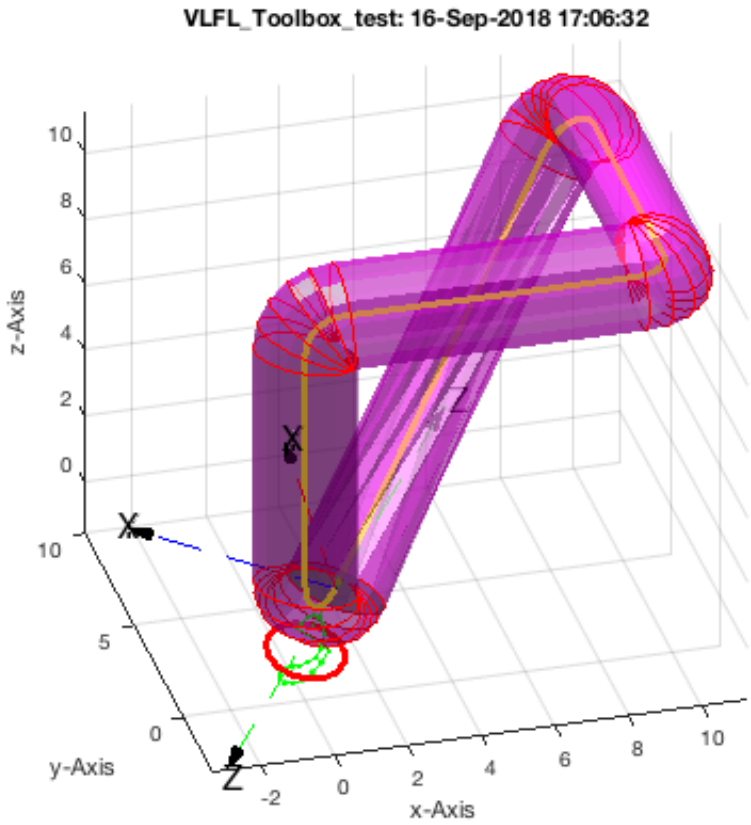
Creating Solid Geometries open

If angles are larger than 90 degree (pi/2)

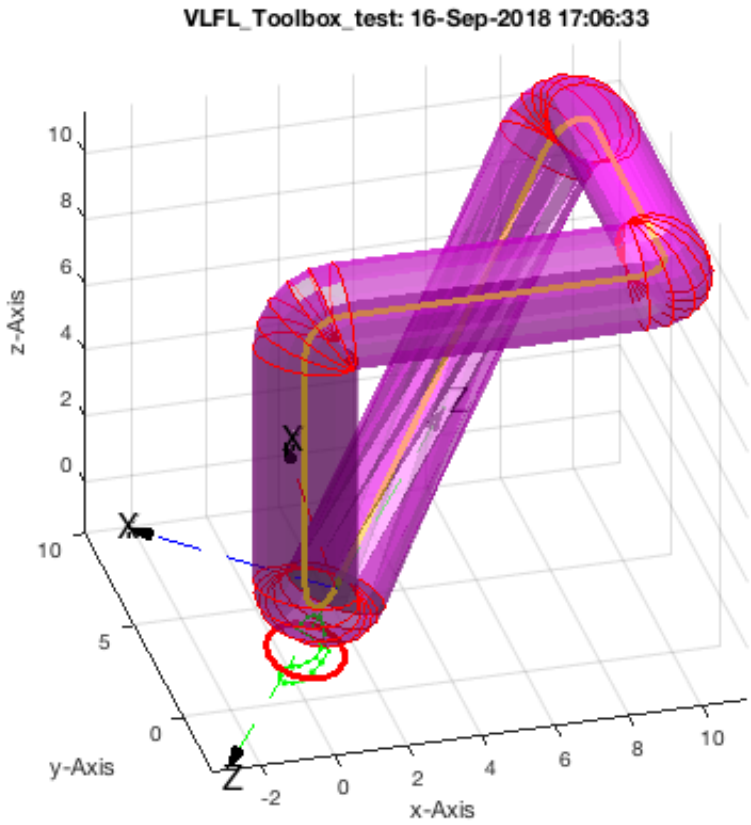
```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(3))));
```



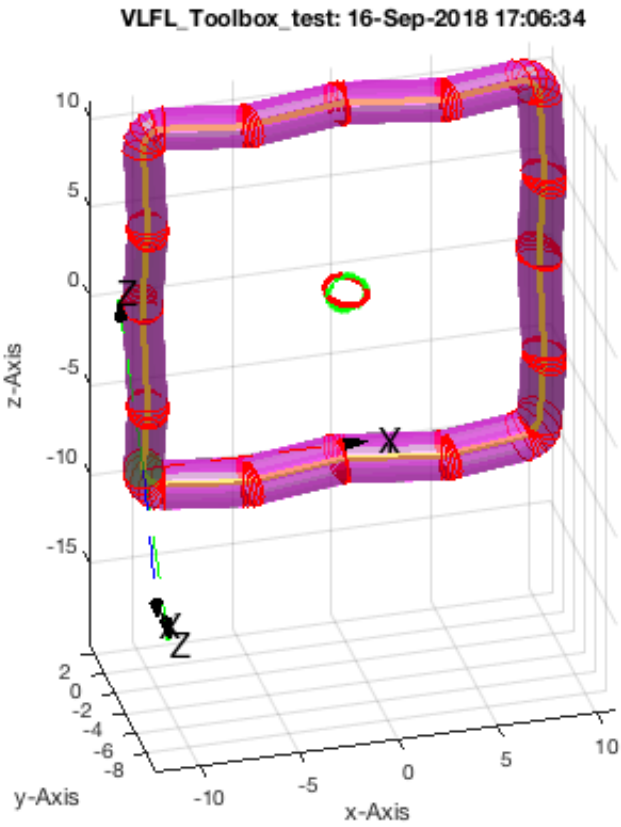

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(7))));
```



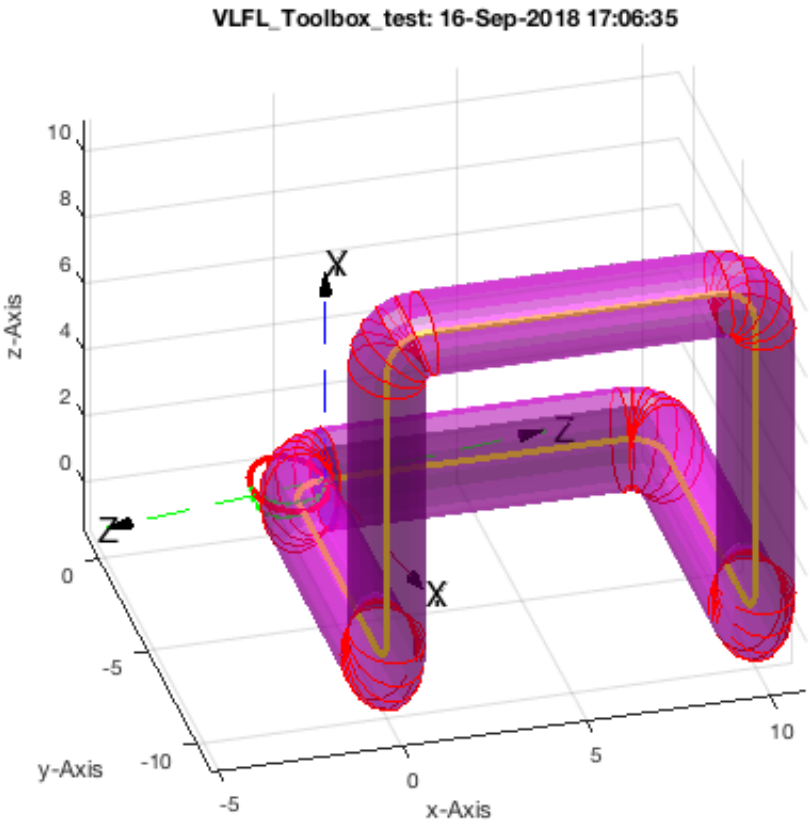

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(8))));
```



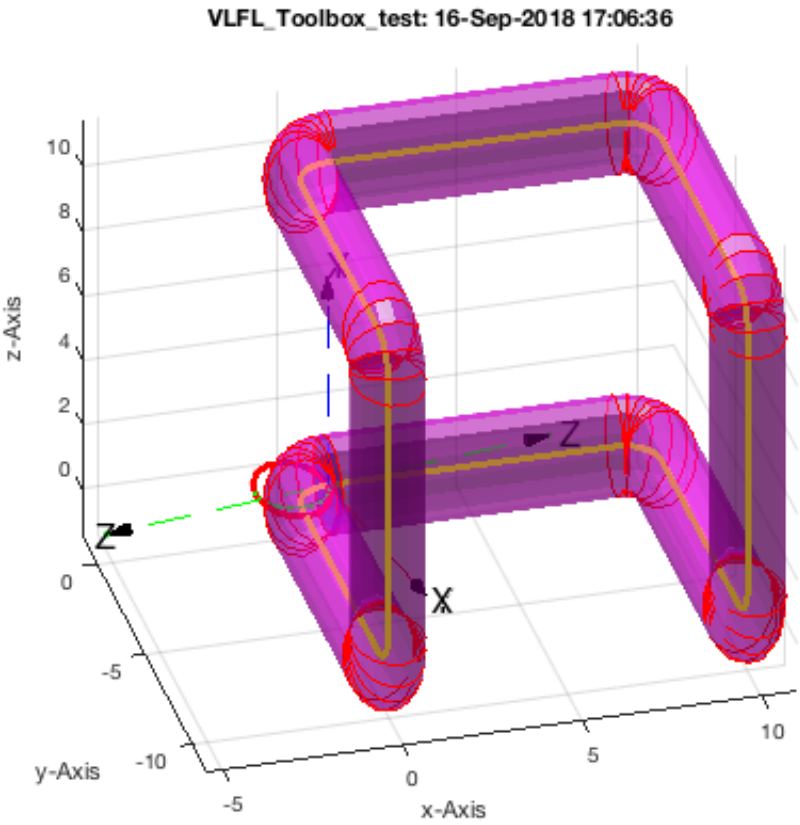

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(12))));
```



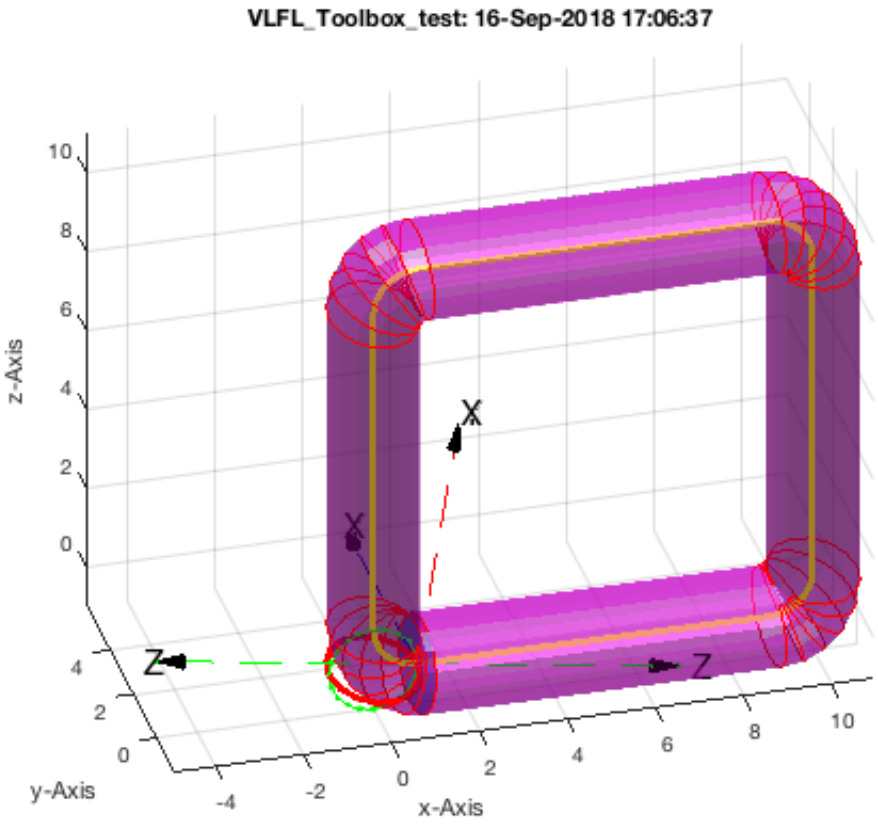

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(13))));
```



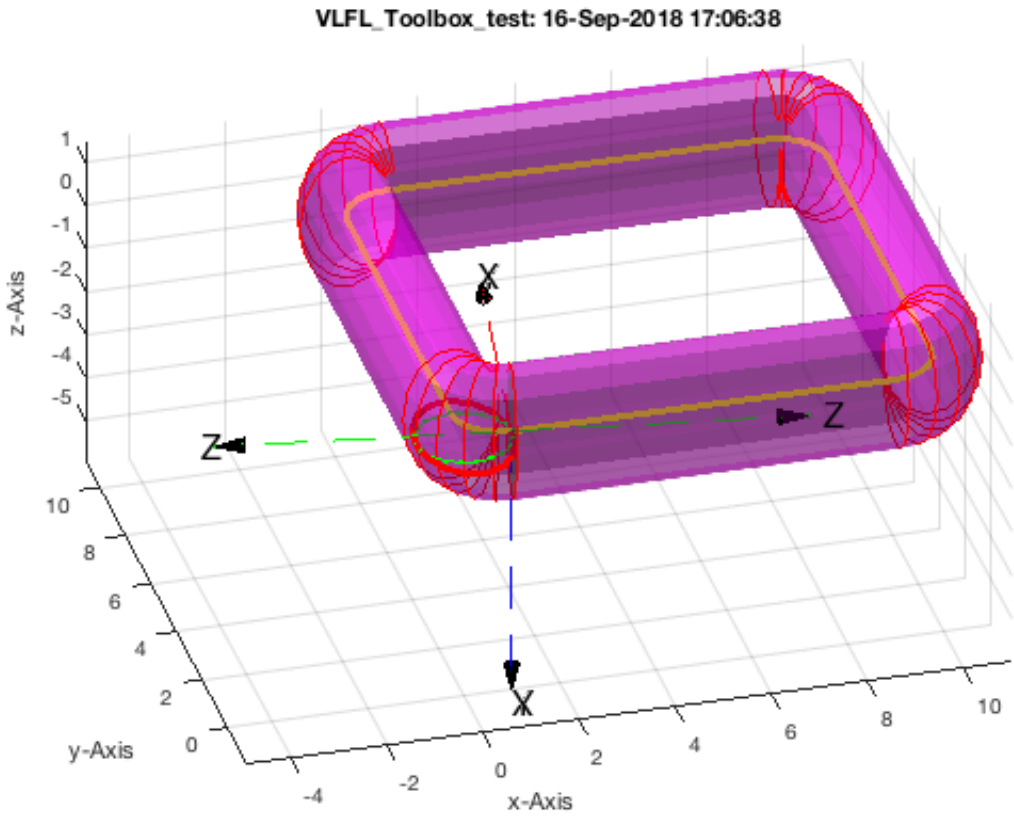

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(14))));
```



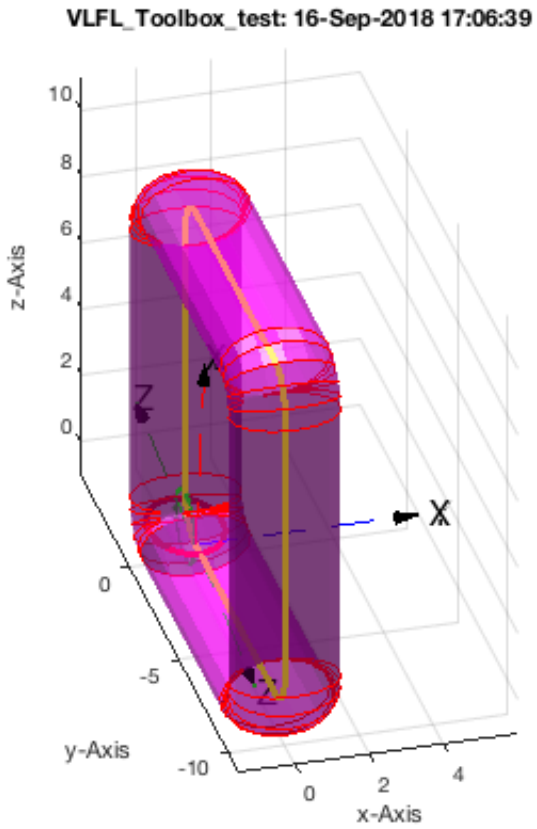

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(20))));
```



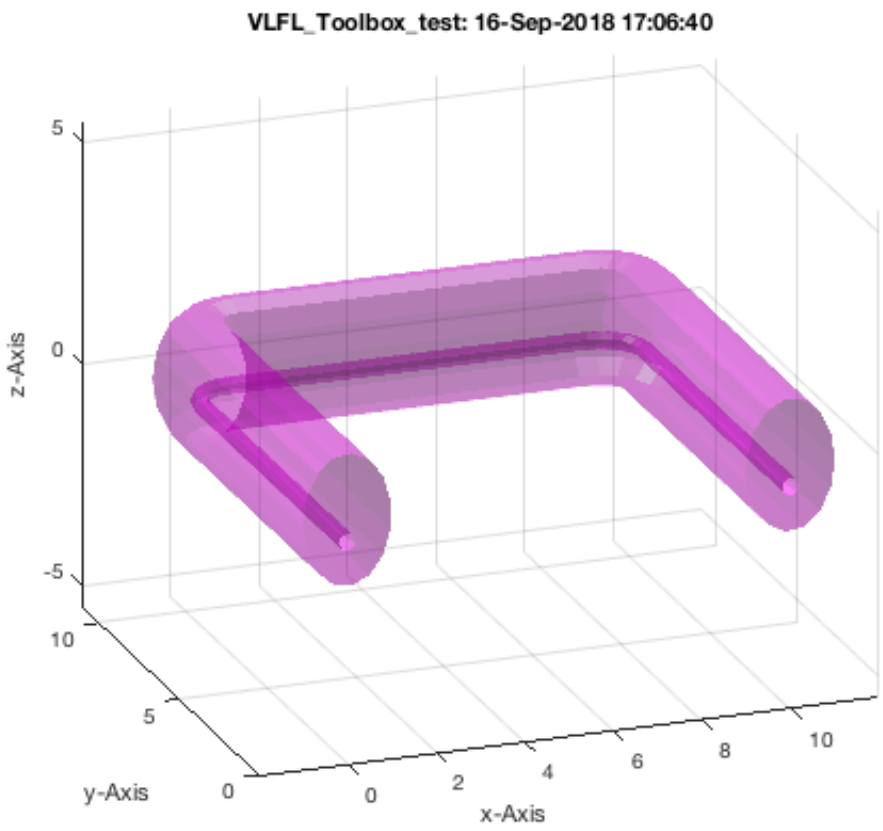

```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(21))));
```




```
SGcontourtube2(PLcircle(1,'','',1.5),VLradialEdges(CVLofVL(VLsample(22))));
```




```
SGcontourtube2([PLcircle(1,'','',1.5);NaN NaN;PLcircle(0.2)+[0 0.5]],VLradialEdges(VLsample
(21))); SG=ans;
SGfigure(SG);VLFLplotlight(1,0.3); view(-20,20);
```



1. Conversion between triangle surface model and tetrahedon volumen model

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2018-Sep-16), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 19-Jun-2073 17:06:41!
Executed 16-Sep-2018 17:06:43 by 'timlueth' using Matlab 9.4.0.929293 (R2018a) Update 5 on
a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
map_toolbox
matlab
pde_toolbox
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2018a