Exercise 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)

Contents

- 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

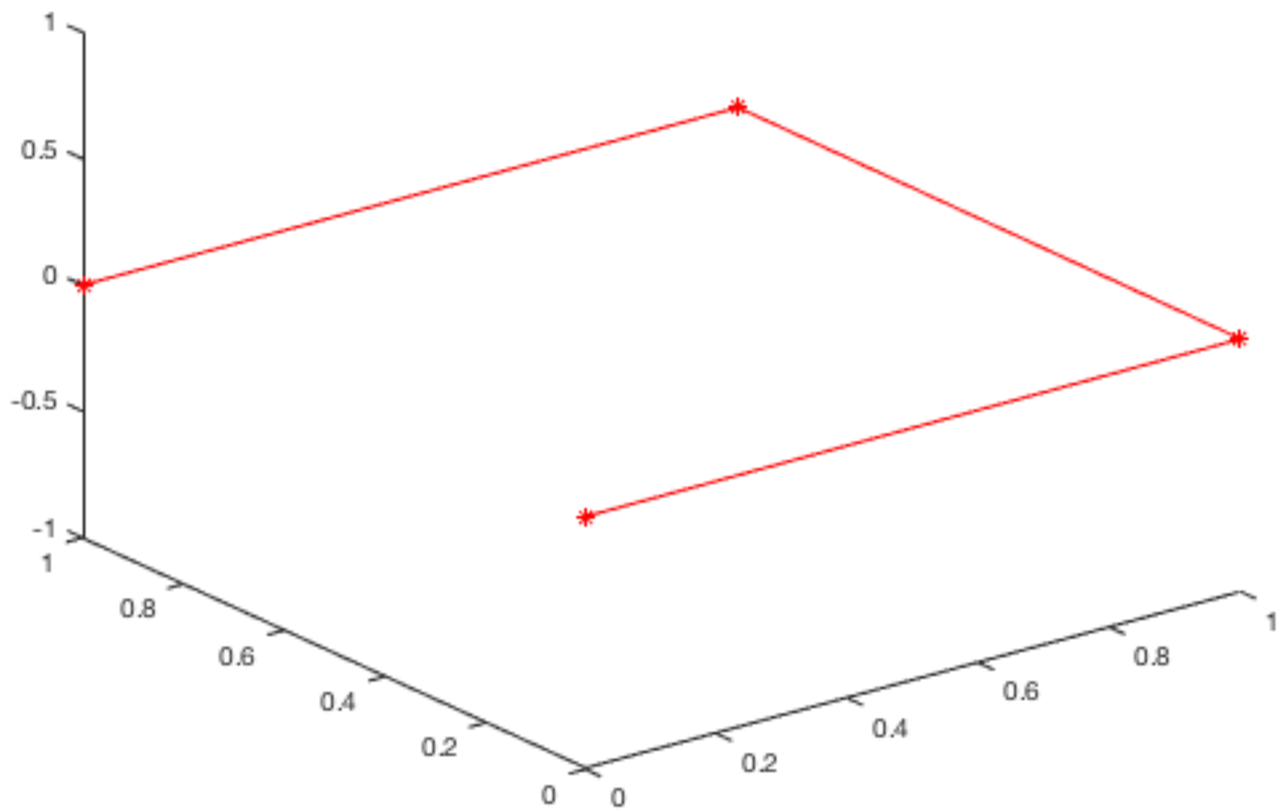
1. Creating and visualizing a simple base-contour by four 2D-points

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
```



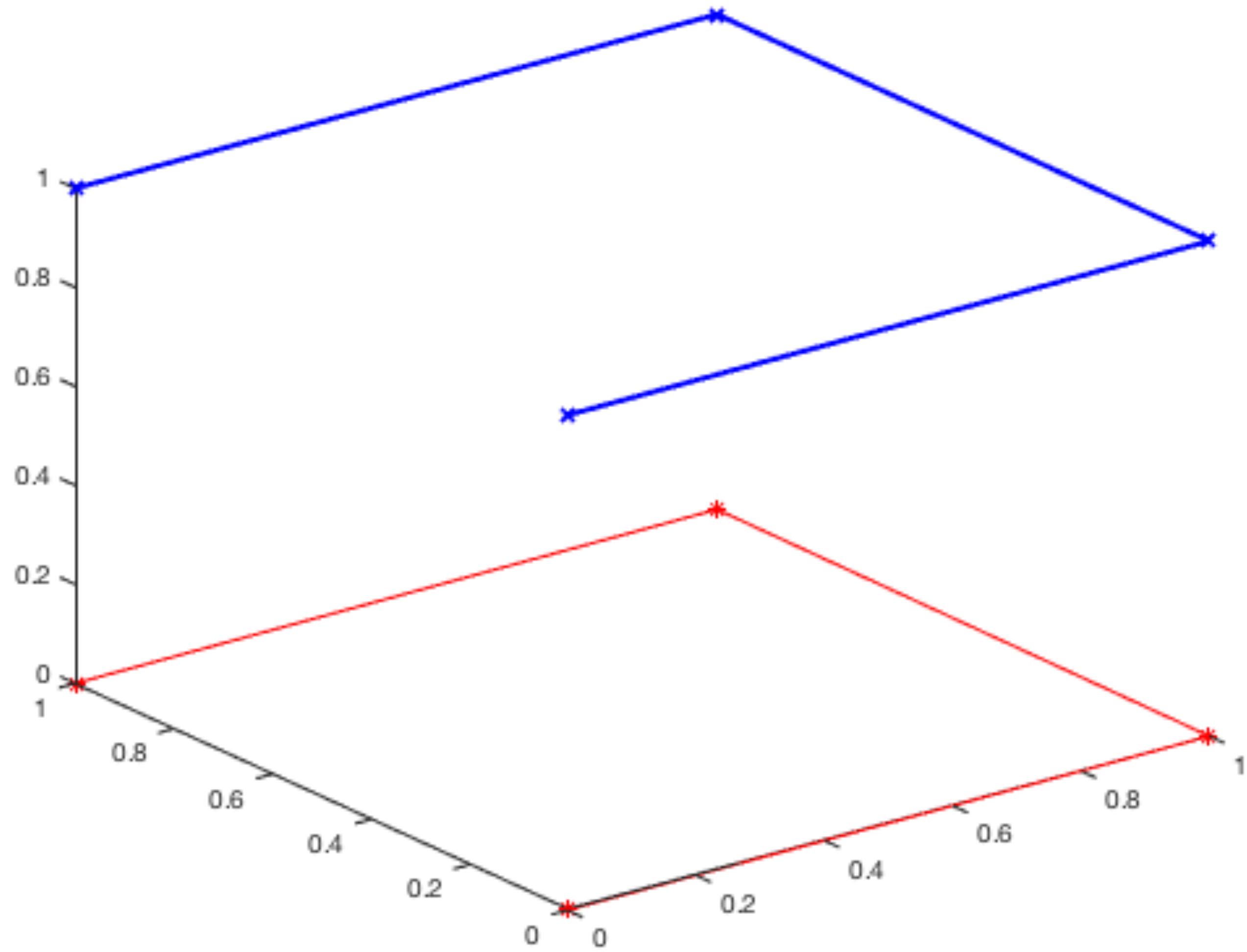
2. Append a 3rd coordinate column to get a vertex list

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
```

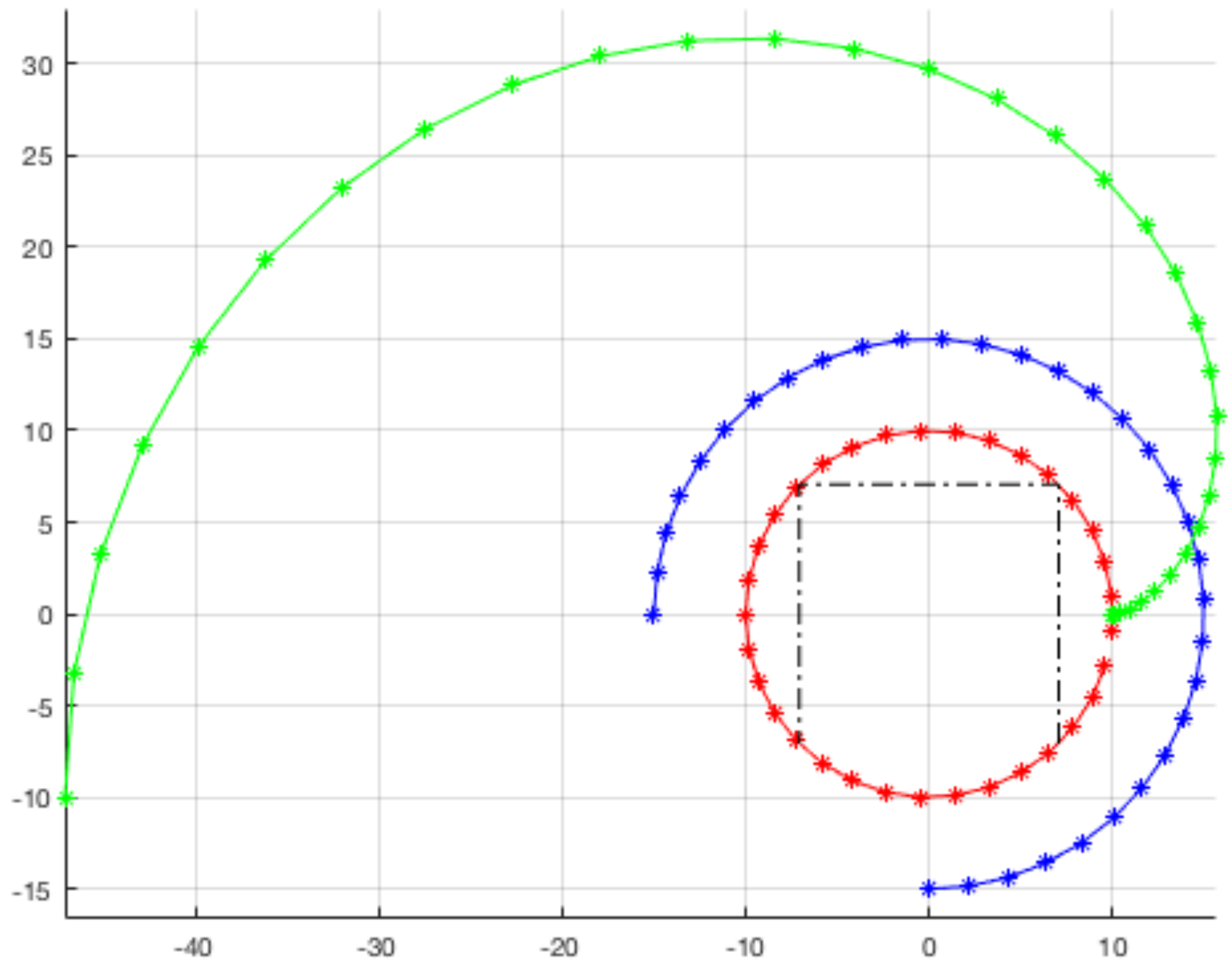


3. Predefined functions for the generation of often used planar polygons (PL)

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;
```

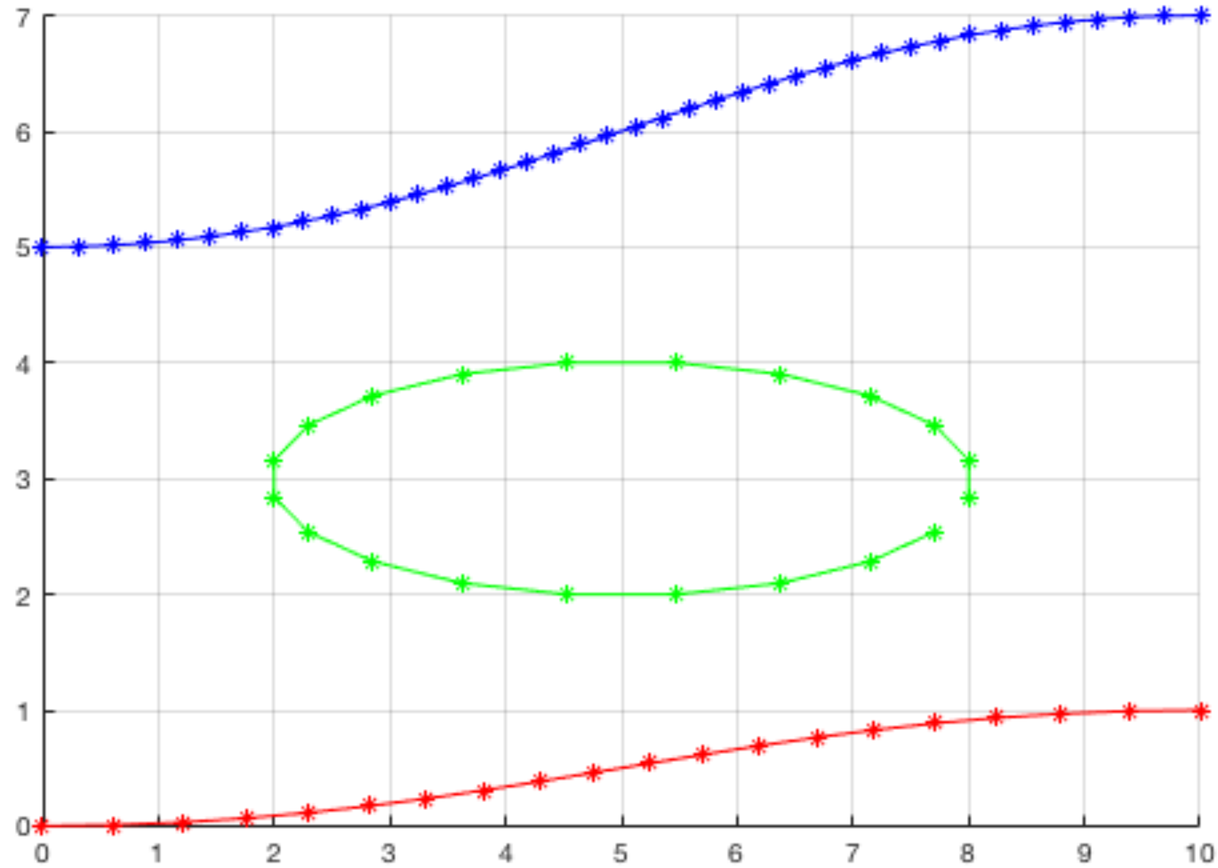


4. More predefined functions for planar polygons in 3D (VL)

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

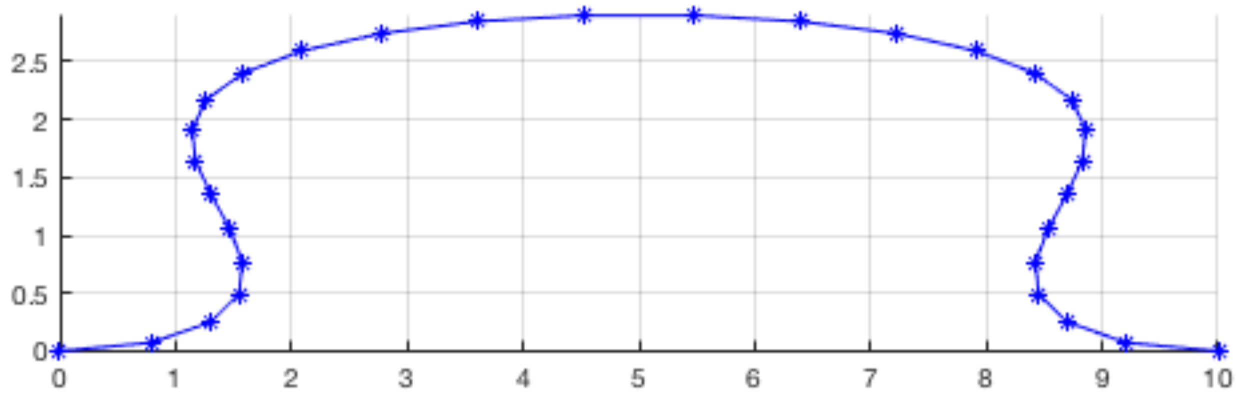
- **VLpolygon** to generate elliptic contours.
- **VLBezier4P** to generate a Bezier-curve using 4 points.
- **VLBezierC** to generate a Bezier-curve using as many points as possible.
- **VLremstraightCVL** to remove obsolete points on straight lines.

```
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.

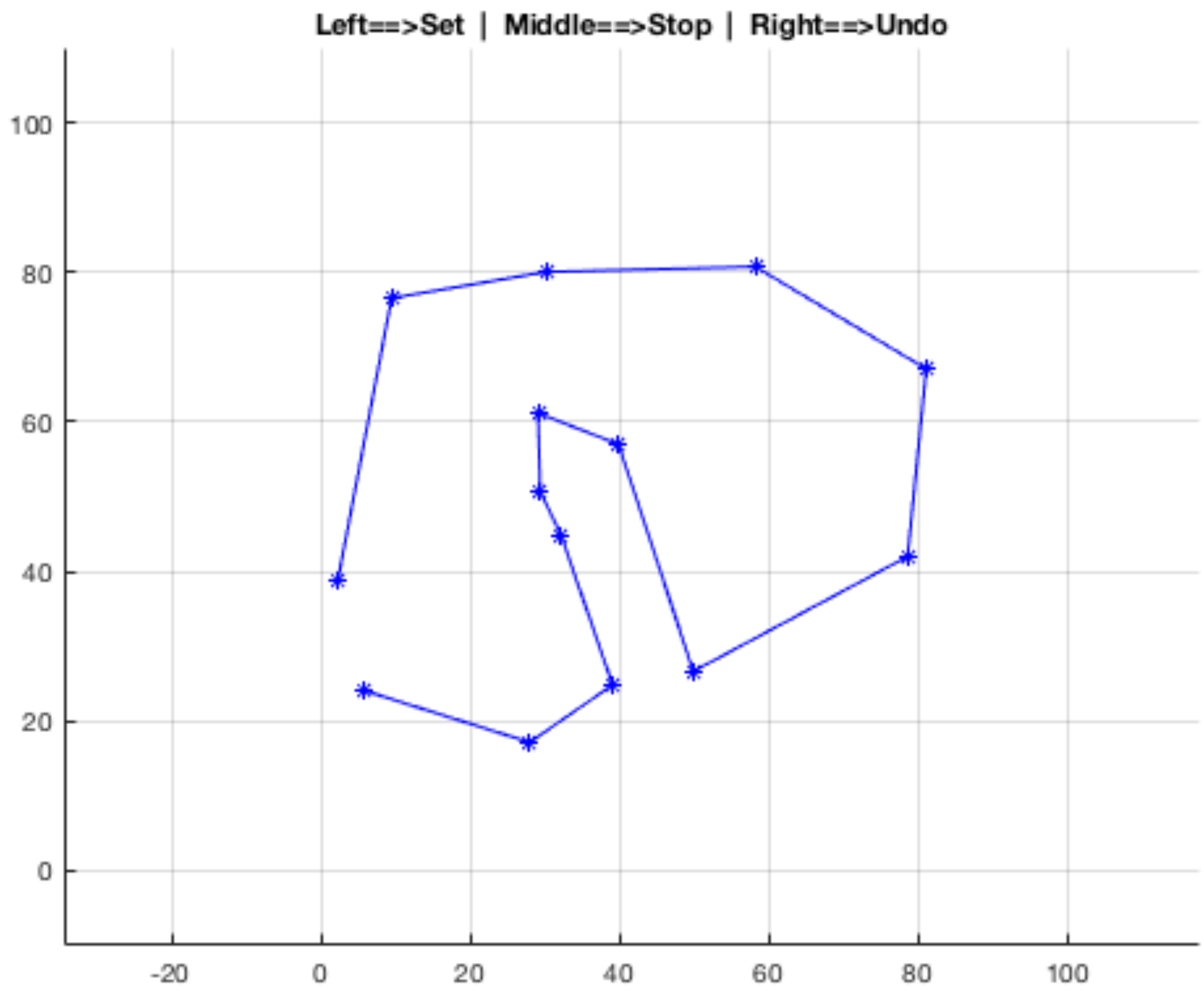
```
close all;
VL=VLBeziernoose(10,2,3,3,30);
VLplot (VL,'b*-'); show, axis equal, view (0,90); grid on
```



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

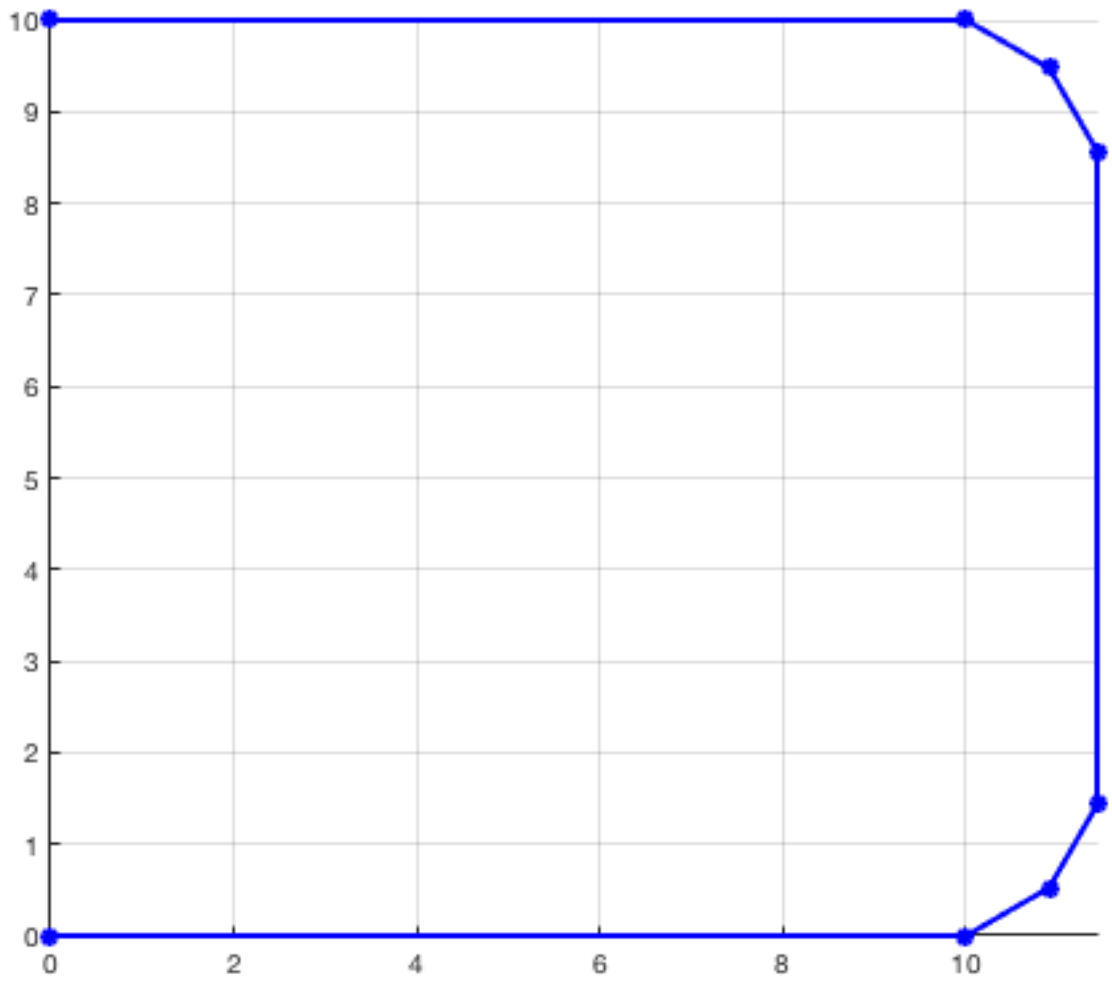
```
close all;
VLui=VLui
VLplot (VL,'b*-'); show, axis equal, view (0,90); grid on
```

```
VL =

     2.3000   38.8000         0
     9.5000   76.6000         0
    30.2000   80.1000         0
    58.2000   80.8000         0
    81.1000   67.1000         0
    78.6000   42.0000         0
    49.9000   26.6000         0
    39.9000   57.0000         0
    29.1000   61.2000         0
    29.4000   50.7000         0
    32.2000   44.8000         0
    39.1000   24.8000         0
    28.0000   17.1000         0
     5.6000   24.1000         0
```



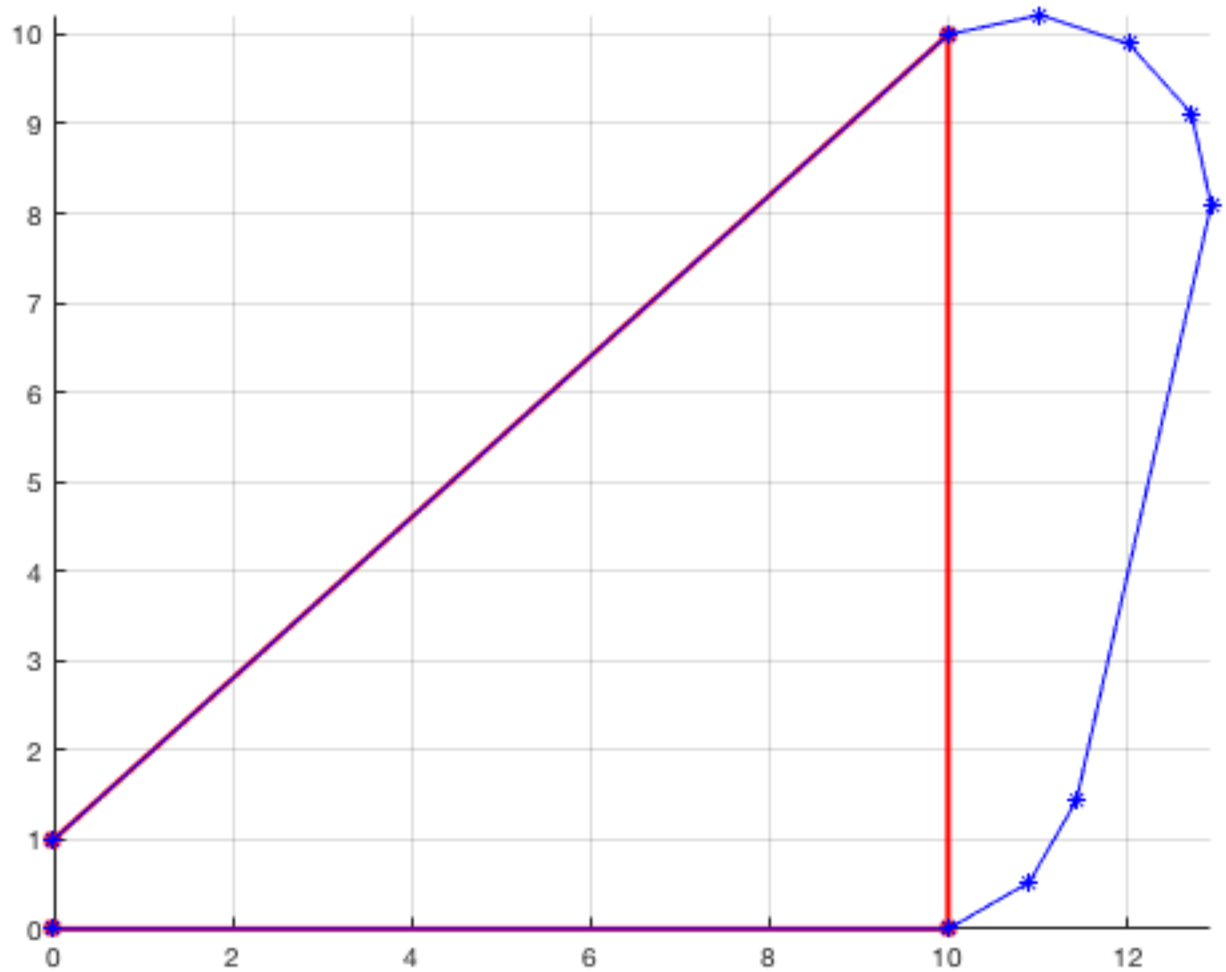
- **VLRadius4P** for inserting points to generate radial curves instead of corners.

```
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
```



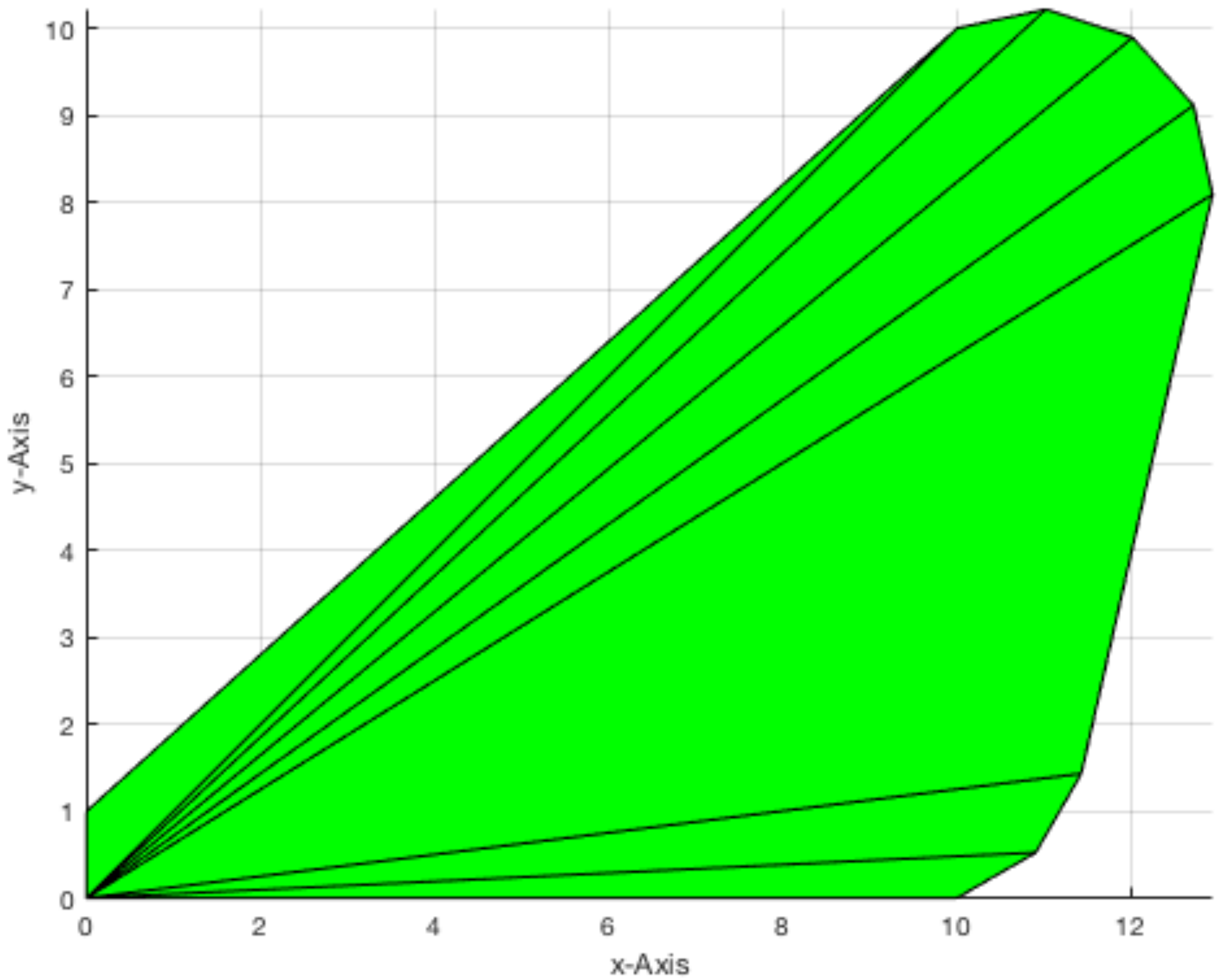
5. Calculation of the surface of a convex polygon

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);
```

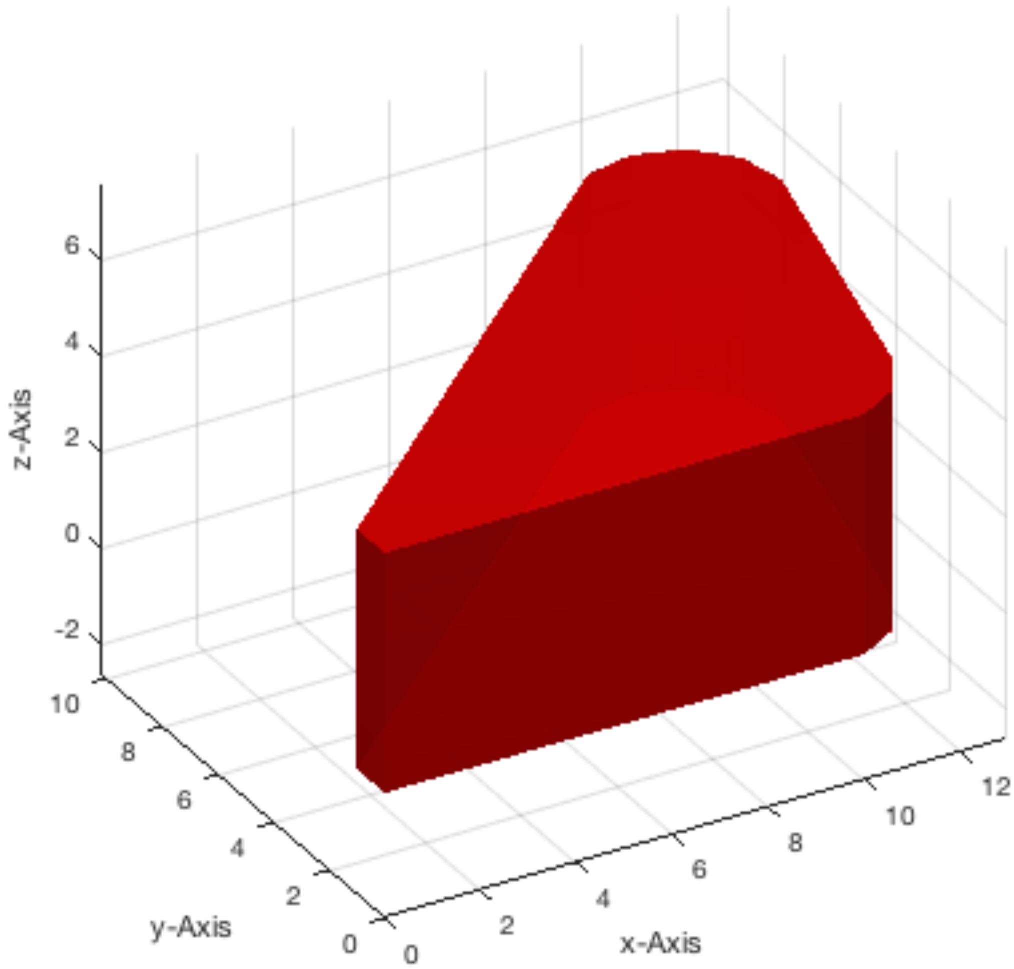
```
FL =

     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;
```



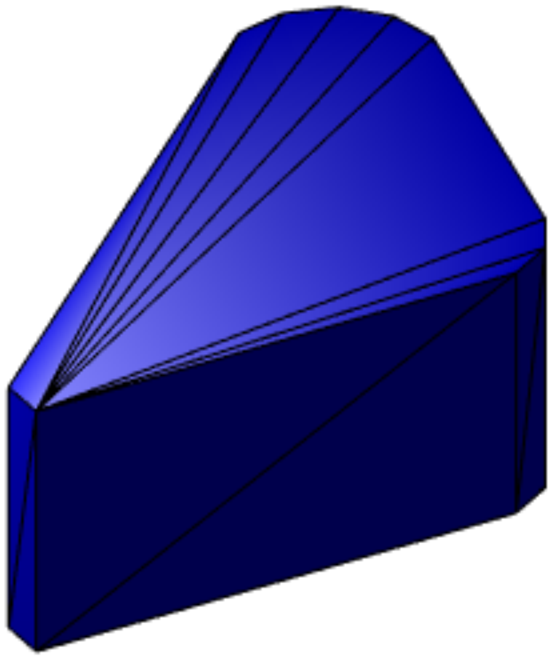
7. Graphical user interface for STL import, export, and viewing

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' : 29-Mar-2017 19:13:21



```
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:13:23 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: =============================
==========
map_toolbox
matlab
simulink
============================================================================================
==========
```

Published with MATLAB® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In Experiment 1 we've learned about

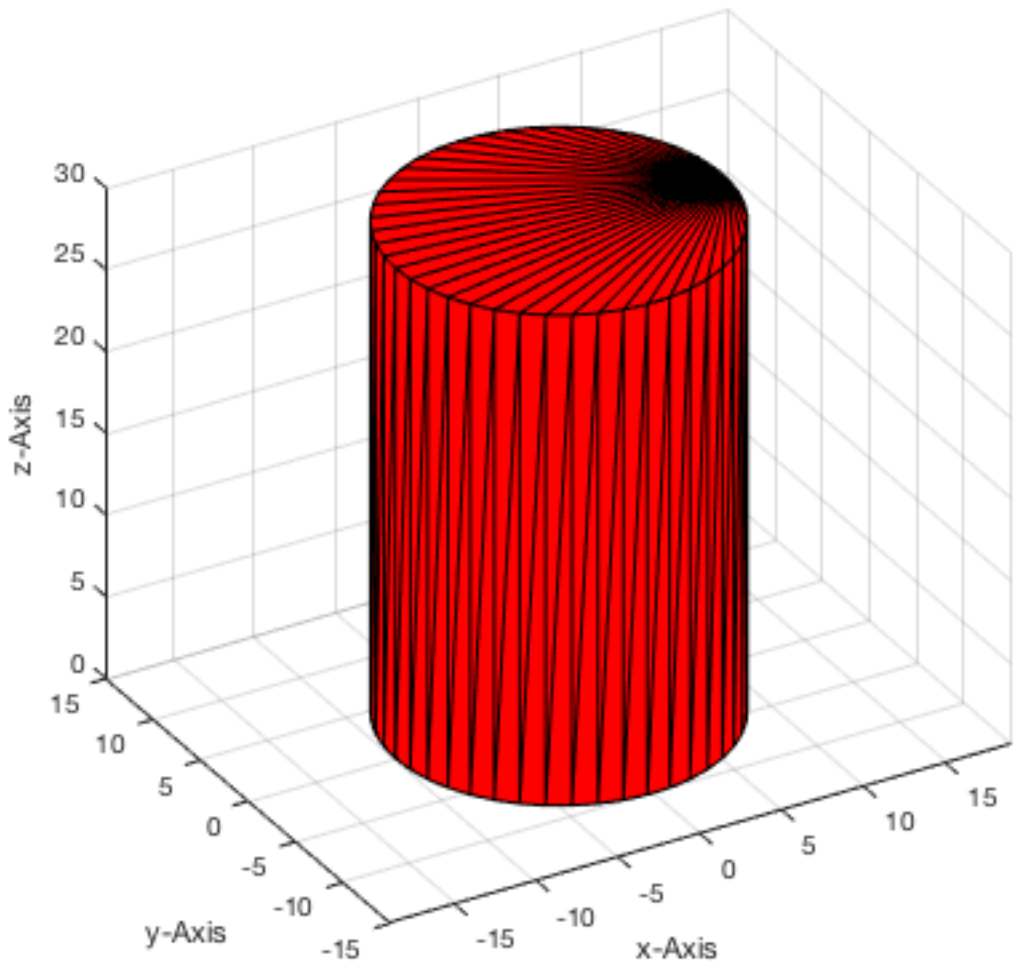
- Point lists (PL) as nx2 coordinate arrays that could be closed to contours.
- Vertex lists (VL) as nx3 coordiante arrays that could be closed to contours.
- Facet lists (FL) as nx3 index arrays that describe traingle facets of a surface.
- Functions for the interactive input of vertex lists.
- Functions for displaying PL and VL in 3D in different color and transparency.
- Functions for generating Bezier-curves or circular-curves.
- Functions for extruding solid volumes based on a closed convex contour.

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/lueth/Desktop/STL-ASCII.STL in ASCII MODE
completed.
WRITING STL (90 vertices, 176 facets) FILE /Users/lueth/Desktop/STL-BINAR.STL in BINARY MOD
E
completed.
```

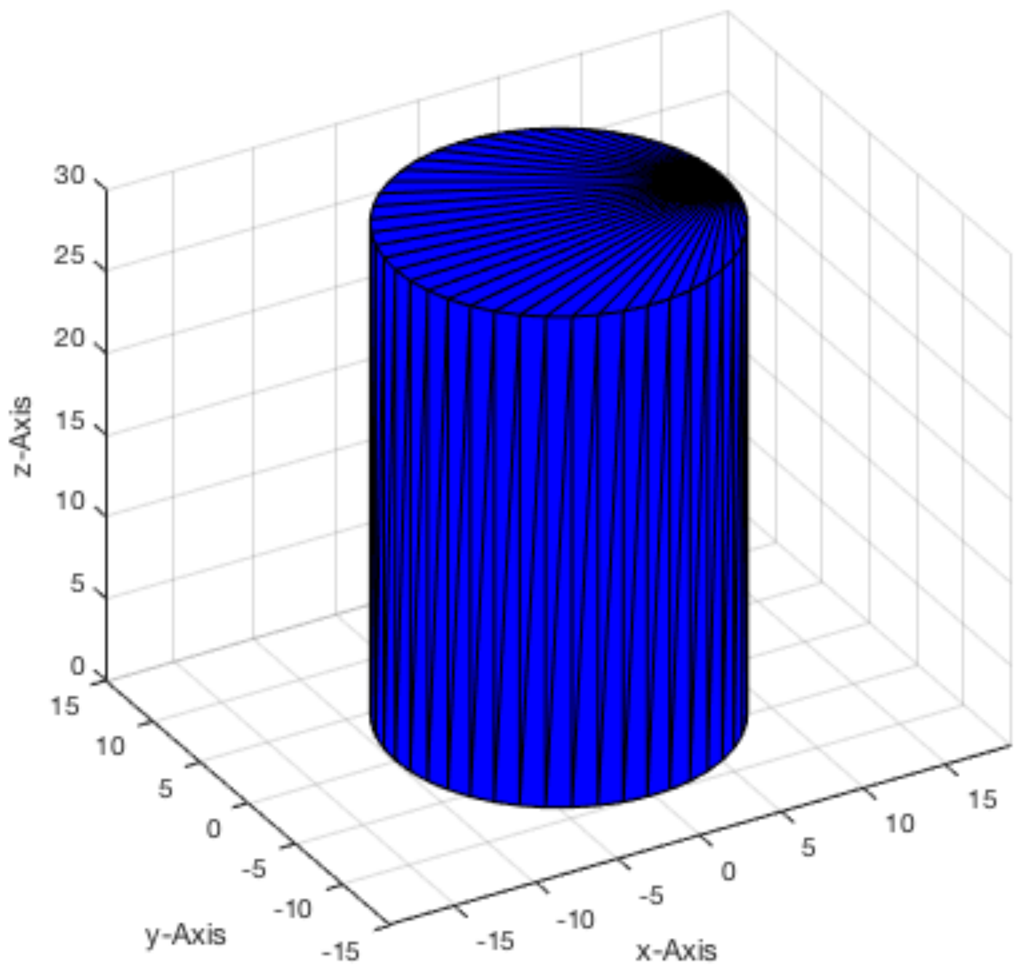
Similar it is possible to read the files in again

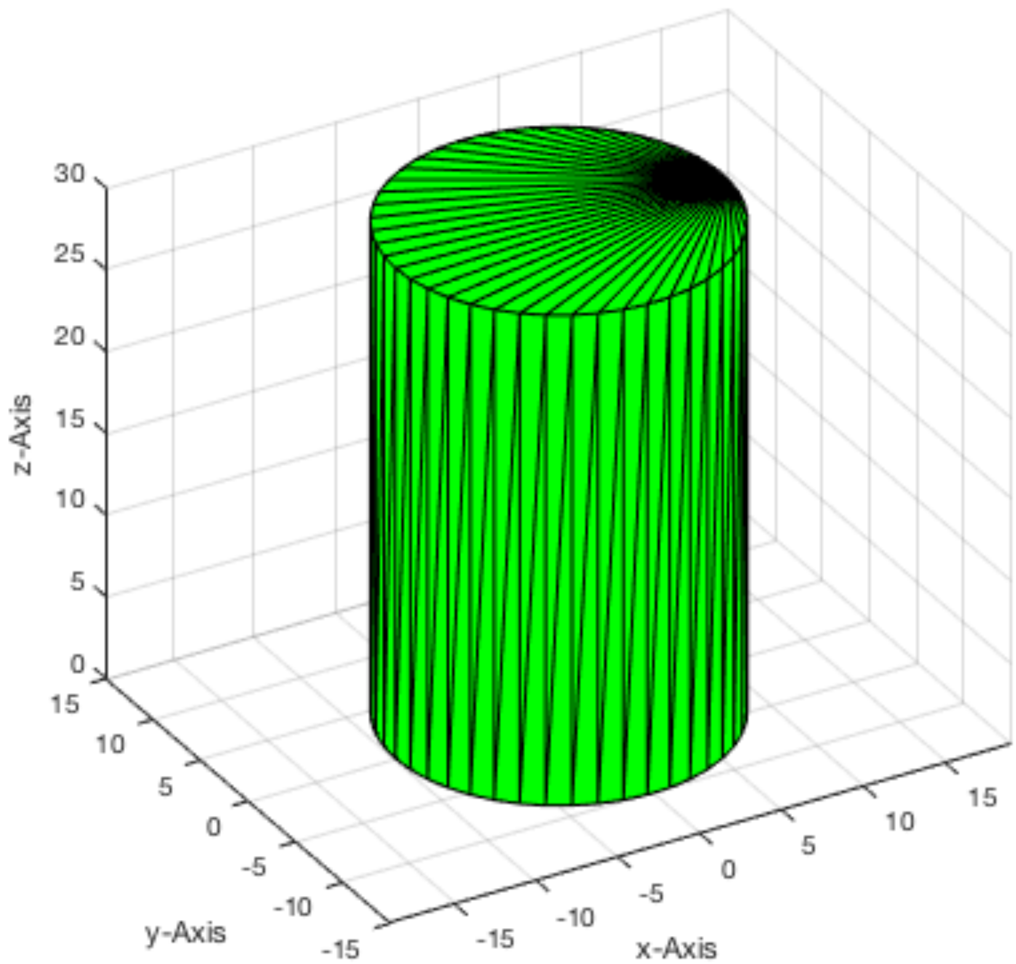
- **VLFLreadSTL** for importing STL-files in ascii file format.
- **VLFLreadSTLb** for importing STL-files in binary file format.

```
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/lueth/Desktop/STL-ASCII.STL
Processing 1234 lines:
Finishing solid AOI-LIB:"STL-ASCII by My Name" 29-Mar-2017 19:13:24 29-Mar-2017 19:13:24
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)
LOADING BINARY STL-File: /Users/lueth/Desktop/STL-BINAR.STL
Header: AOI-LIB:"STL-BINAR by My Name" 29-Mar-2017 19:13:24
Number of facets: 176
0..
Finishing and Checking
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)
```





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
```

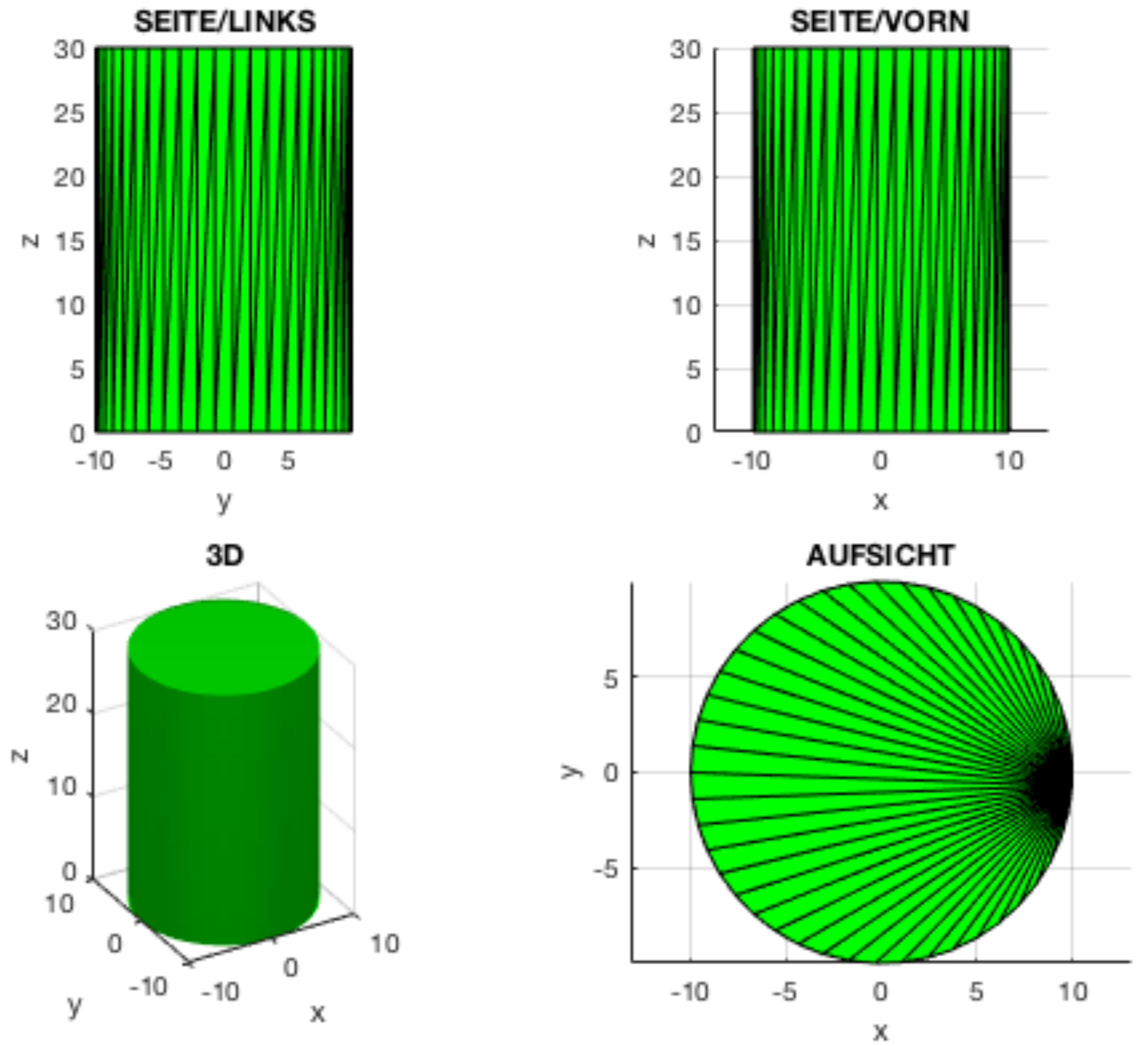
```
VLFLchecker: 90 vertices and 176 facets.
    0 FACET PROBLEMS DETECTED (ERRORS)
    0 VERTEX PROBLEMS DETECTED (OBSOLETE WARNING)
    0 EDGE PROBLEMS DETECTED (NON MANIFOLD WARNING)
    0 SOLID/EDGE PROBLEMS DETECTED (OPEN SOLID WARNING)
```

- **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

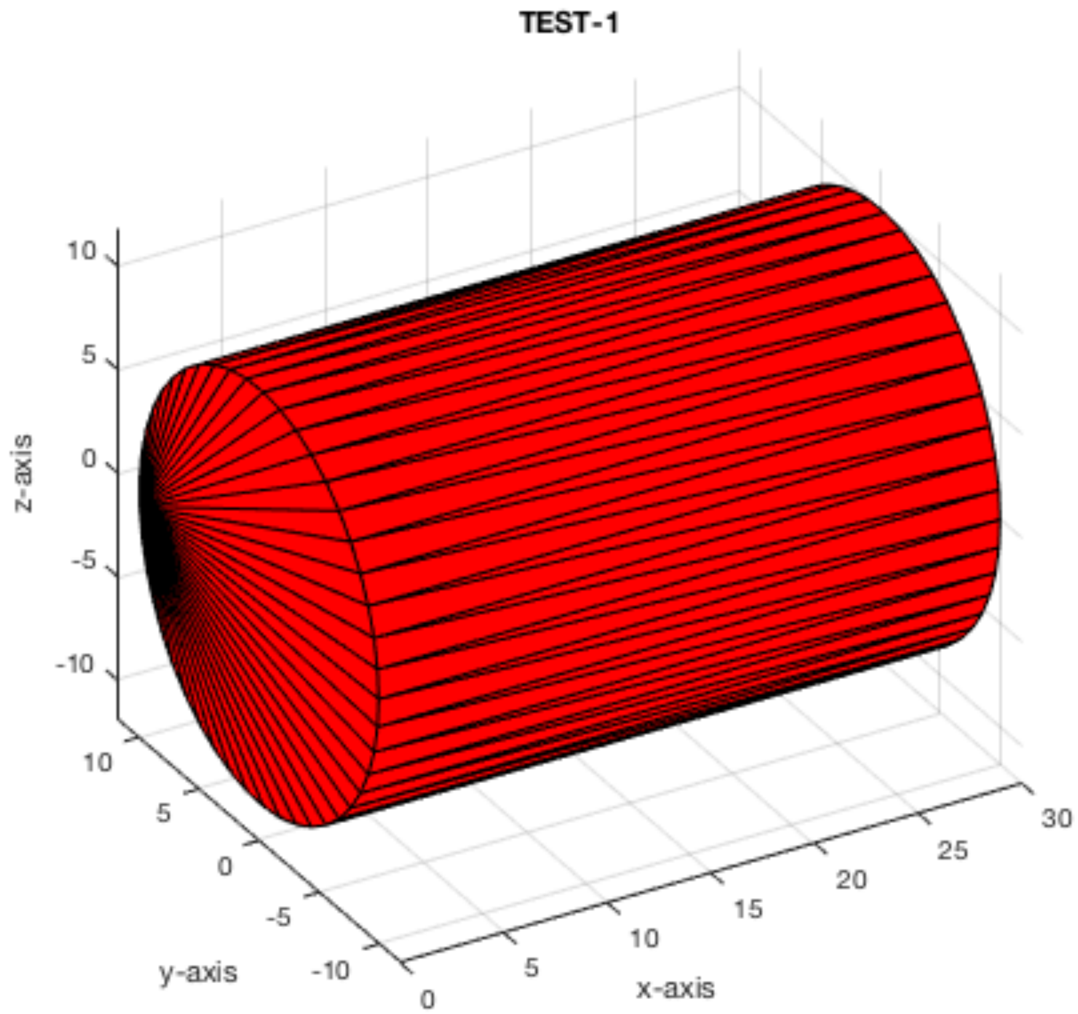
```
BBofVL(VL)
close all; VLFLplots4 (VL,FL,'g');
```

```
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
```



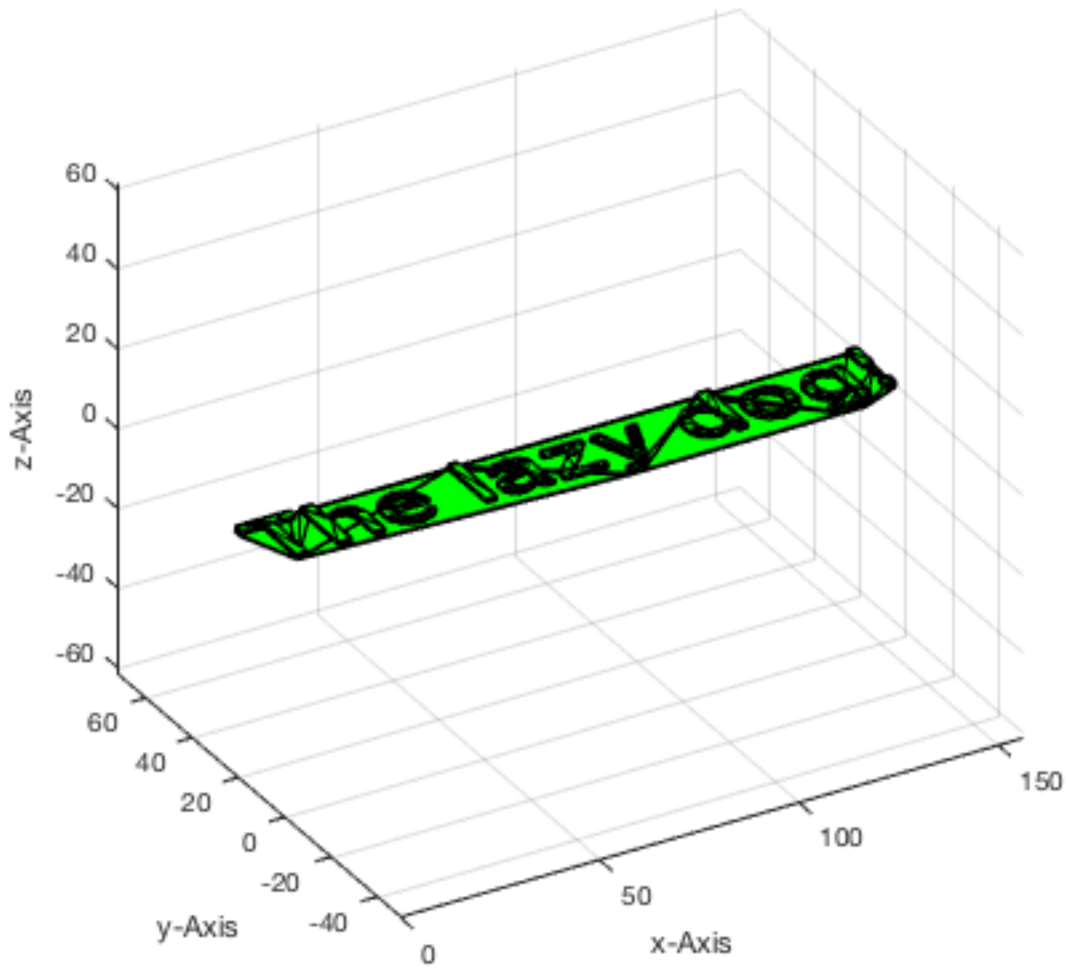
4. Generation of text, numbers, characters and formulas as solid volume

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');
```

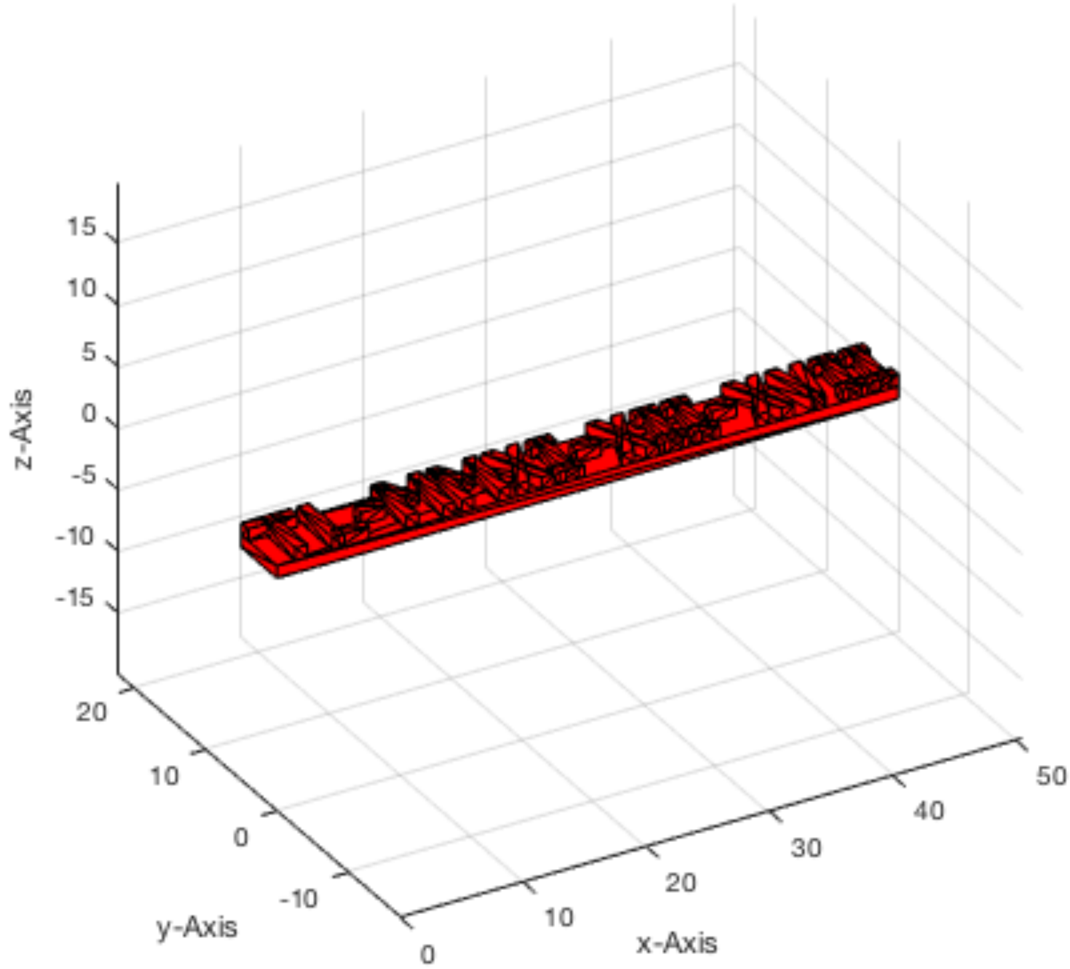
```
WRITING STL FILE /Users/lueth/Desktop/exp_2011_12_17.STL in ASCII MODE
completed.
```



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.

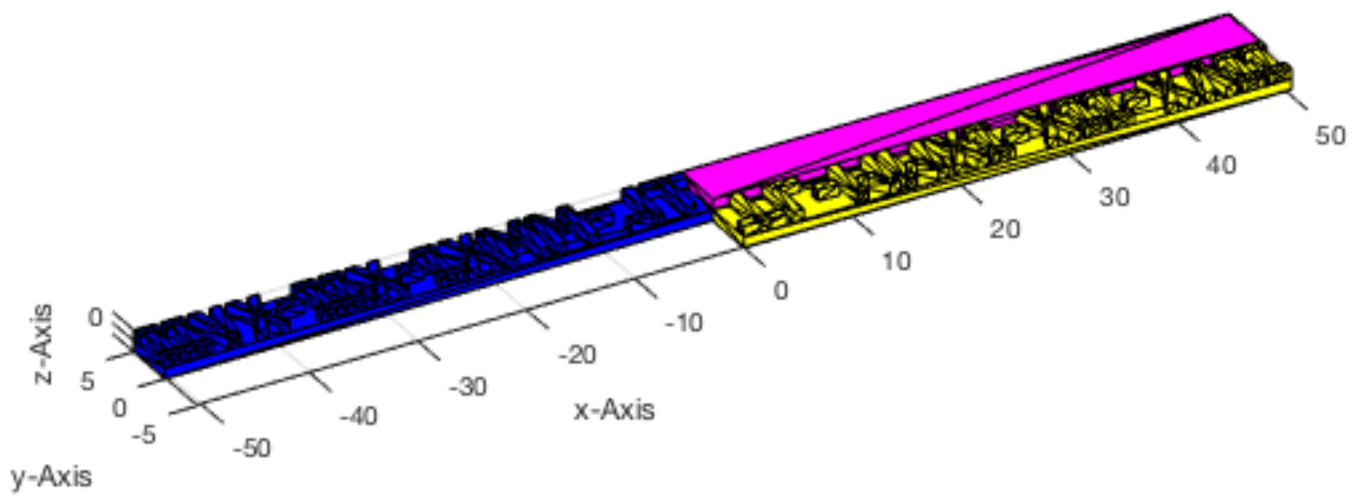
```
close all;
[VL,FL,d]=VLFLtext('TL-MMXI-XII-XVII'); VLFLplot(VL,FL,'r'); view(-30,30);
```



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).

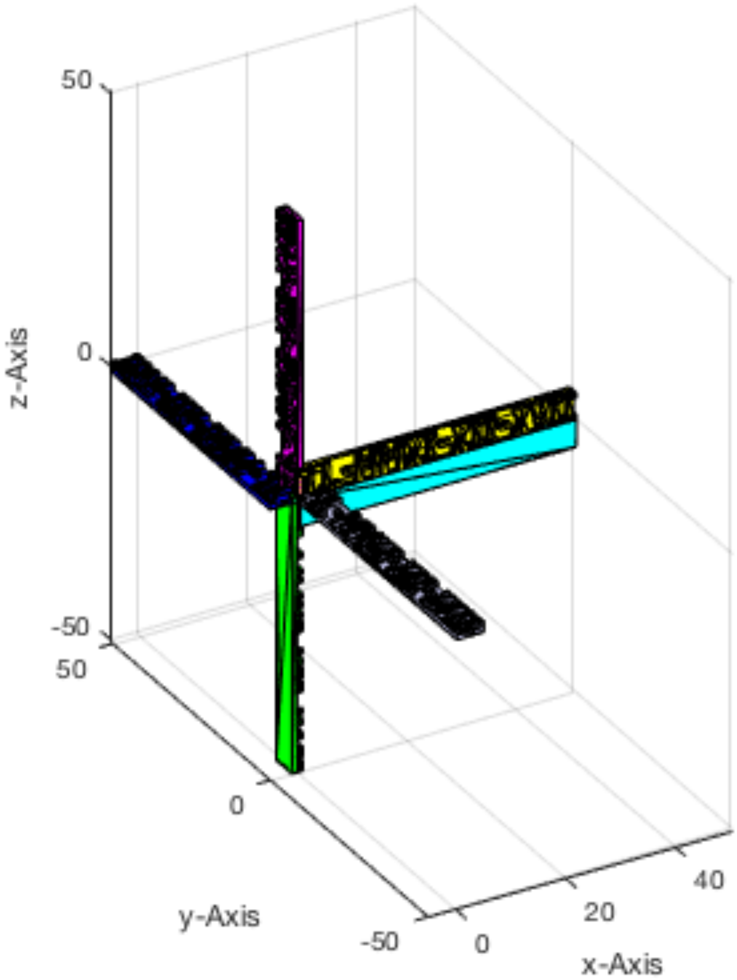
```
close all, view (-30,30); grid on;
VLFLplot(VLswapX(VL),FL,'b');        % mirror at x-axis
VLFLplot(VLswapY(VL),FL,'y');        % mirror at y-axis
VLFLplot(VLswapZ(VL),FL,'m');        % mirror at z-axis
```



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.

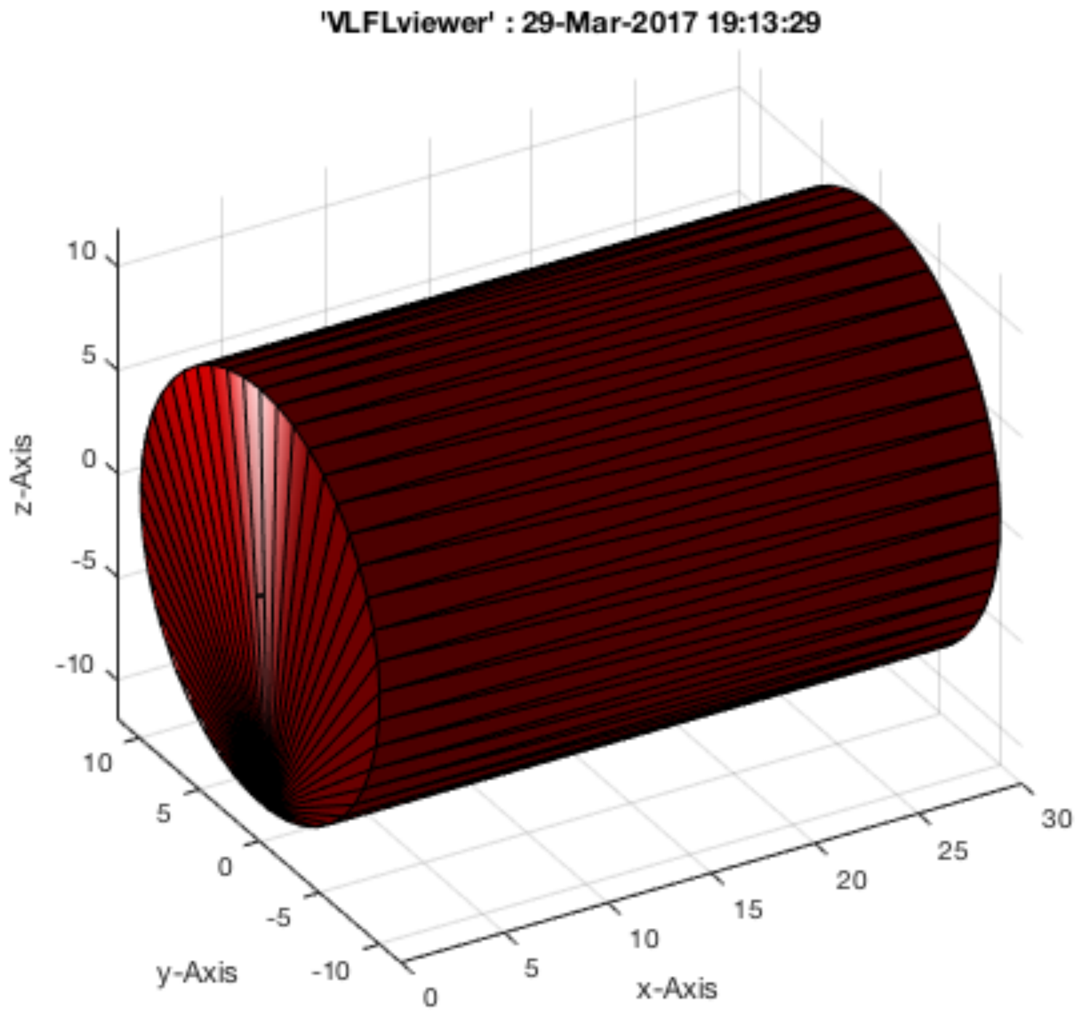
```
close all, view (-30,30); grid on
VLFLplot(VL,FL,'r');                % original solid
VLFLplot(VLswapXY(VL),FL,'b');      % turn the x-axis to the y-axis
VLFLplot(VLswapXZ(VL),FL,'m');      % turn the x-axis to the z-axis
VLFLplot(VLswapYZ(VL),FL,'y');      % turn the y-axis to the z-axis
VLFLplot(VLswapZY(VL),FL,'c');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

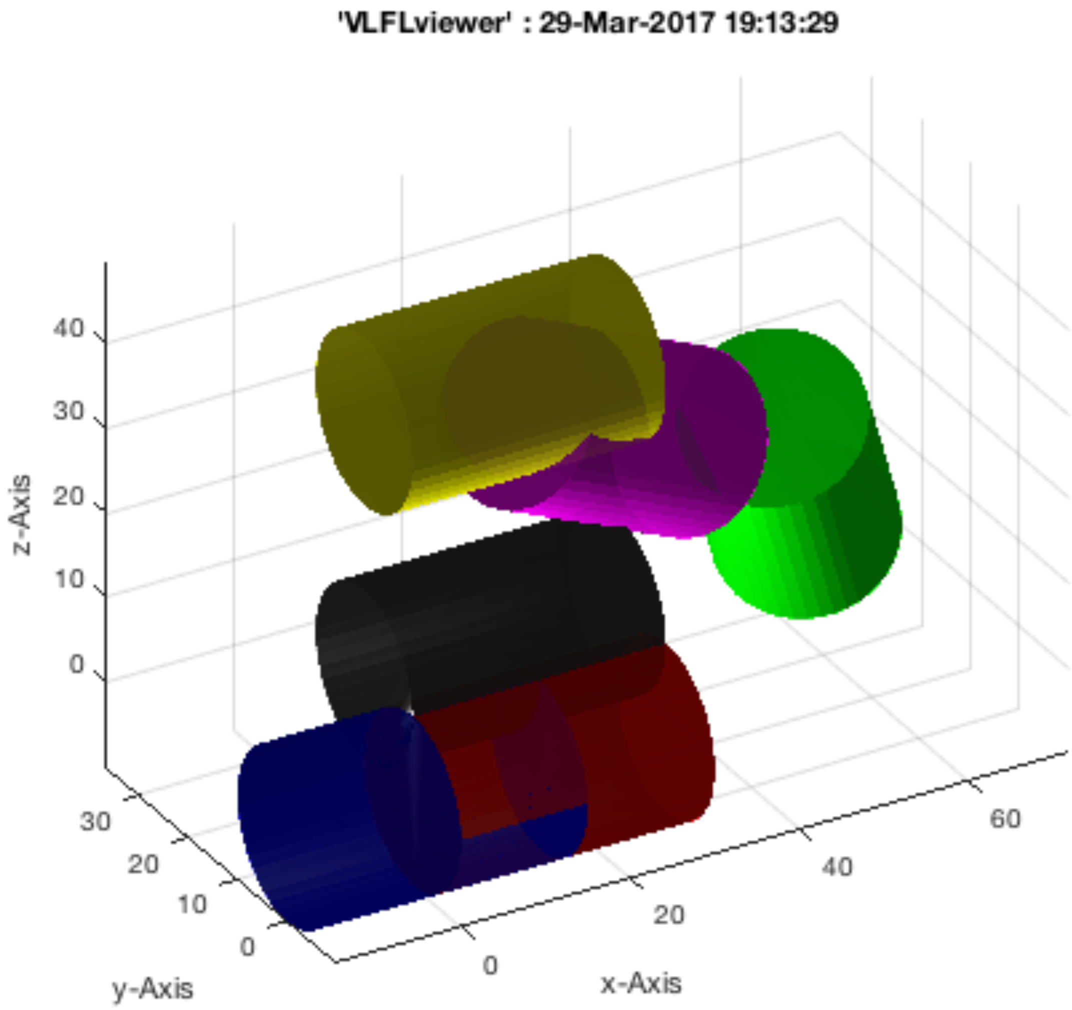
- **VLtrans0** for translating the solid in the coordiante system origin.
- **VLtrans1** for translating the solid into quadrant 1.
- **VLtransP** for translating the solid using a translation vector.
- **VLtransR** for rotation the solid using a rotation matrix.
- **VLtransT** for transformating using 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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:13:32 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In example 1 and 2 we've learned about

- Point lists (PL) and Vextlist (VL) as nx2/nx3 coordinate arrays
- Facet lists (FL) as nx3 index arrays that describe traingle facets of a surface
- Functions for generating/extruding surfaces and surface bounded solid volumes
- Functions for displaying points, surfaces, volumes different colors and transparency
- Functions for import, check, and export of STL-Data
- Functions for turning, mirroring and spatial transformation of objects
- Functions for generation of text string as solid objects

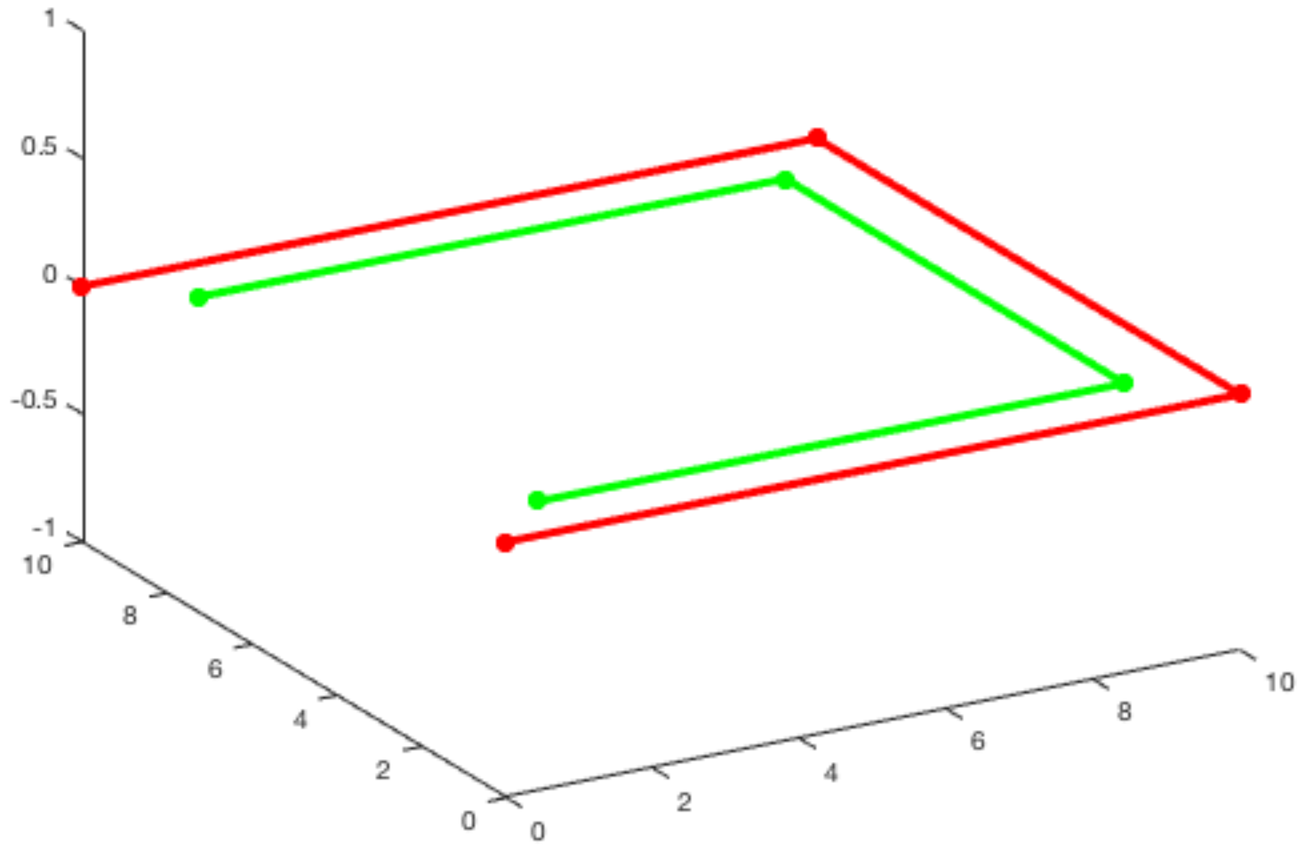
2. The contour polybool list (mapping toolbox)

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
```

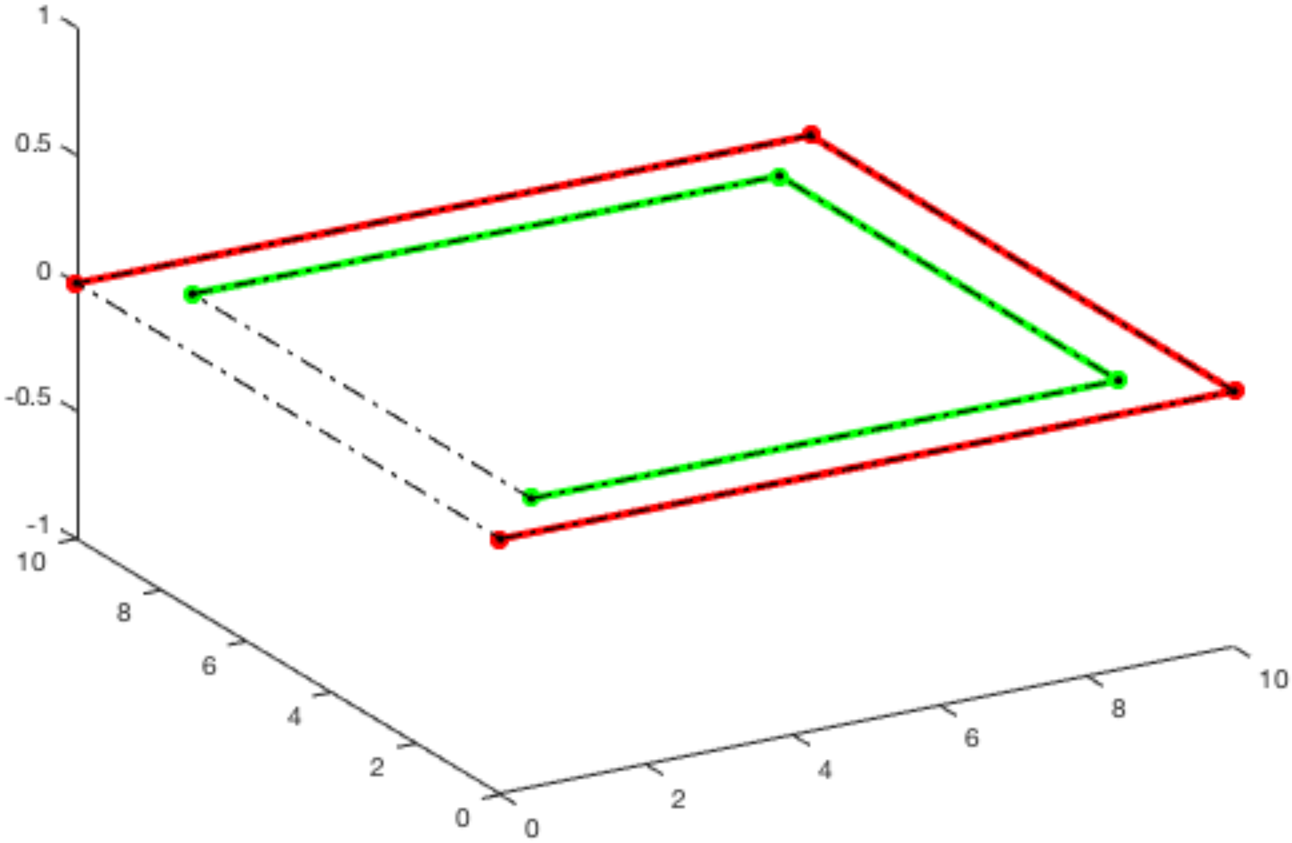


The concatenation generates then the closed polybool list (CPL). Two functions are helpful to draw a CPL and also to show the start point (black cube), the endpoint (black ring) and the direction of each edge of the CPL:

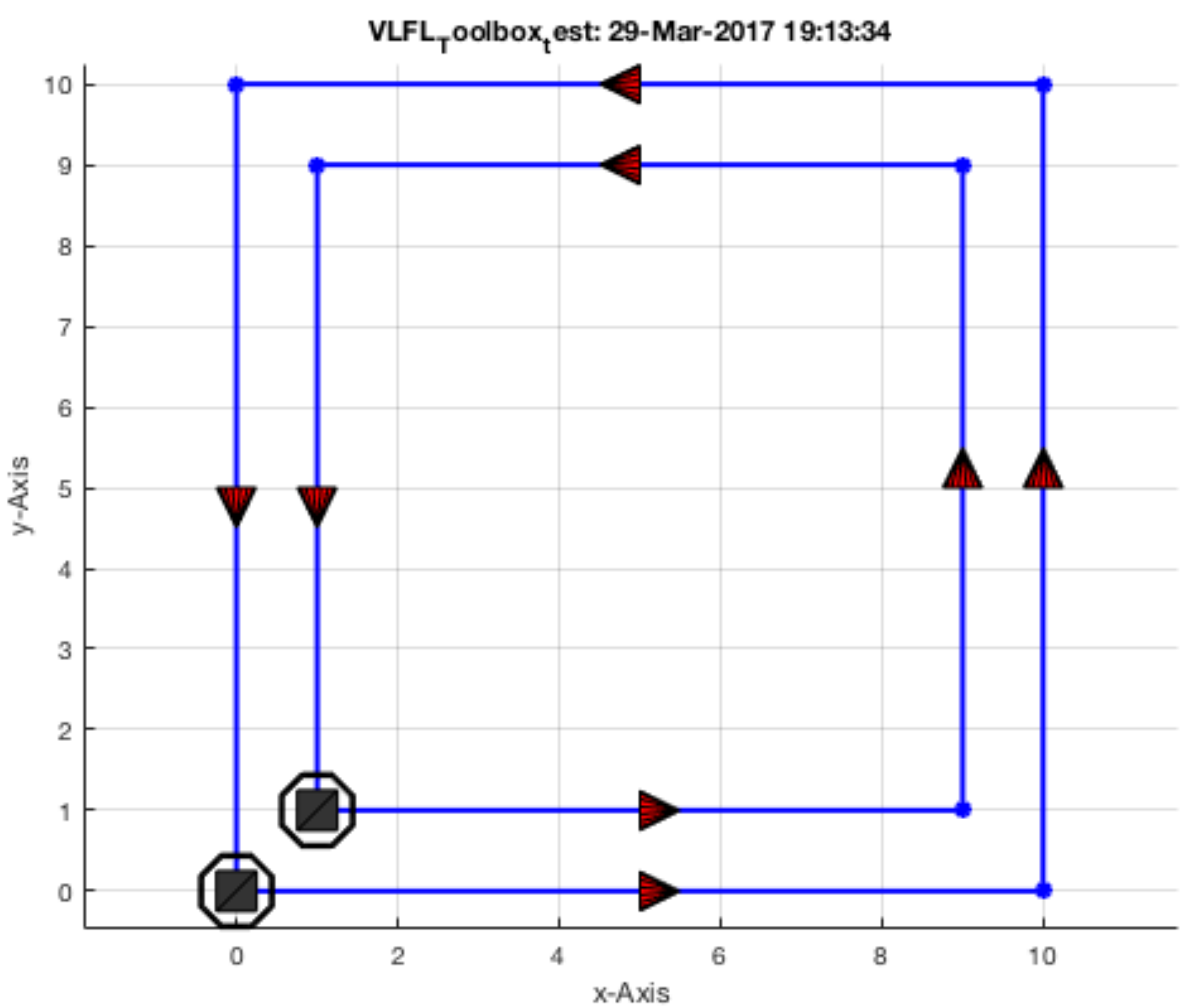
- **CPLplot** draws a closed contour polybool list (CPL).
- **PLELofCPL** draws a start point, end point and direction-arrows, when called without any output variable.

```
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.

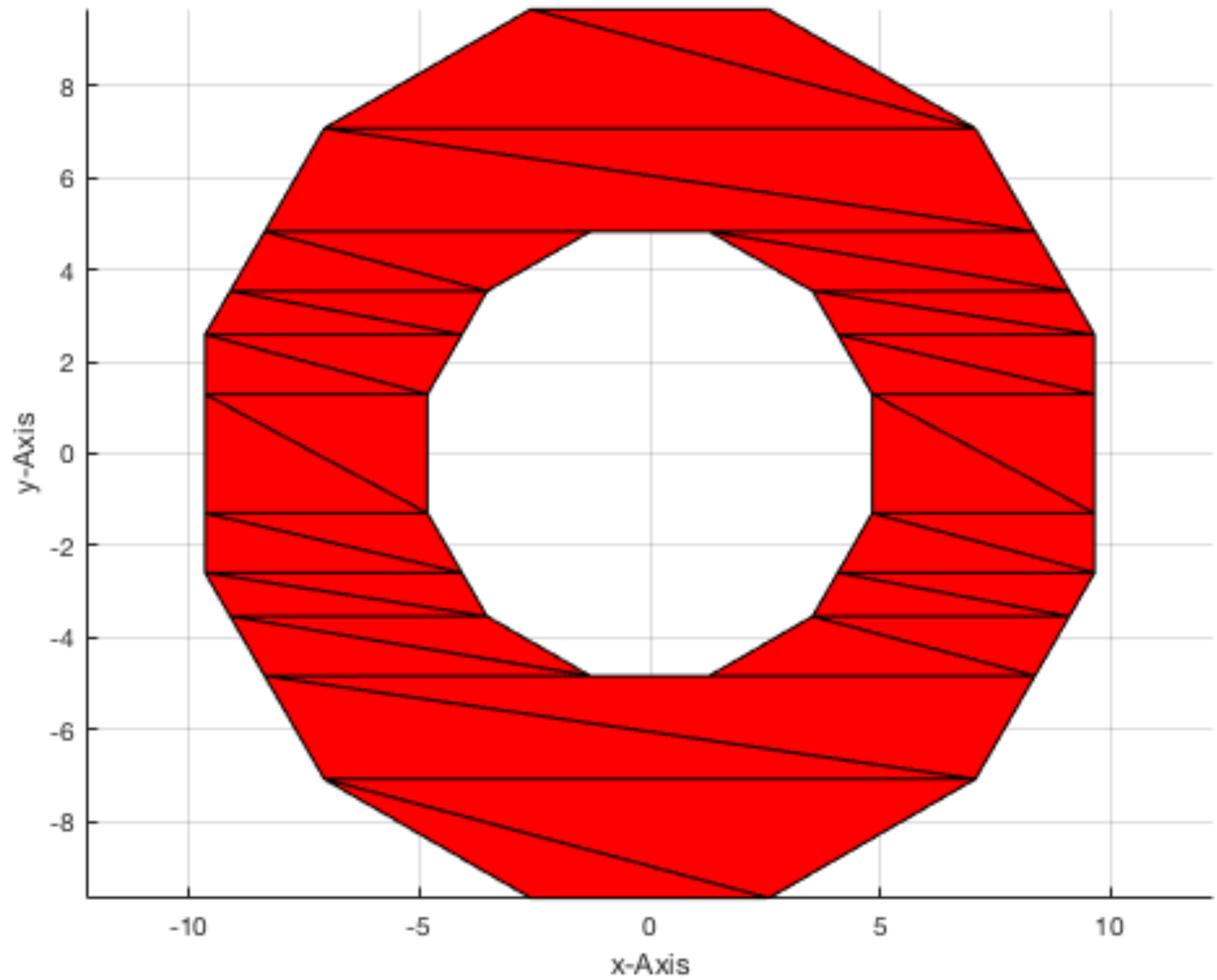
- **Row-Scanning-Triangulation** is able to handle all kinds of polygons (even enclosed), but the triangle facets are sometimes very small. Furthermore, the point contains redundant information.
- **Delaunay-Triangulation** is able to handle all kinds of polygons (even enclosed), but has problems with 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.

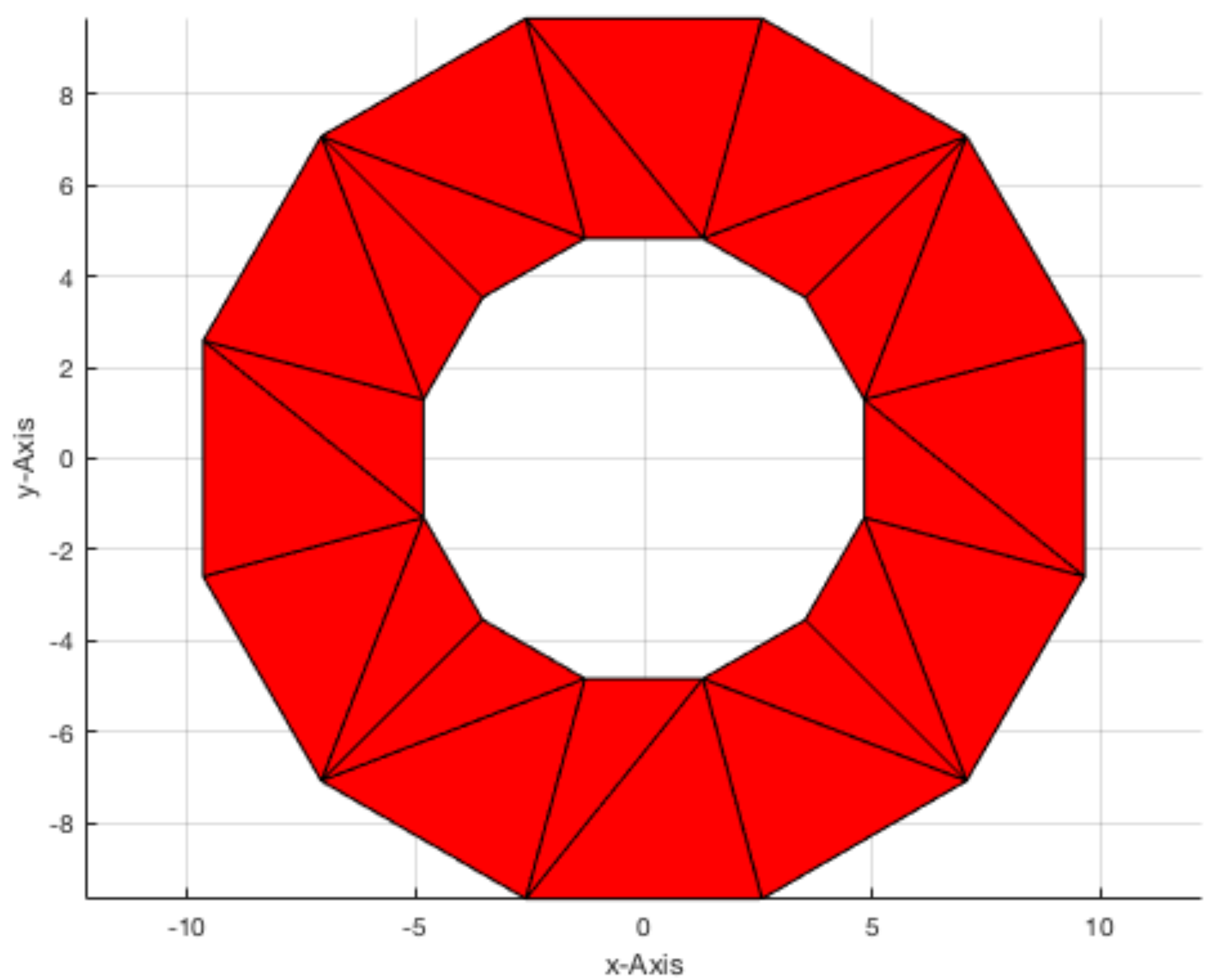
- **PLFLofCPLpoly** returns a facet tesselation by a simple y-coordinate row scanning (the points are ordered by increasing y, contour by contour, do not mix, but are redundant. Not as efficient as Delaunay, and not useful for 3D).
- **PLFLofCPLdelaunay** returns a facet tesselation by a Delaunay-triangulation (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
```



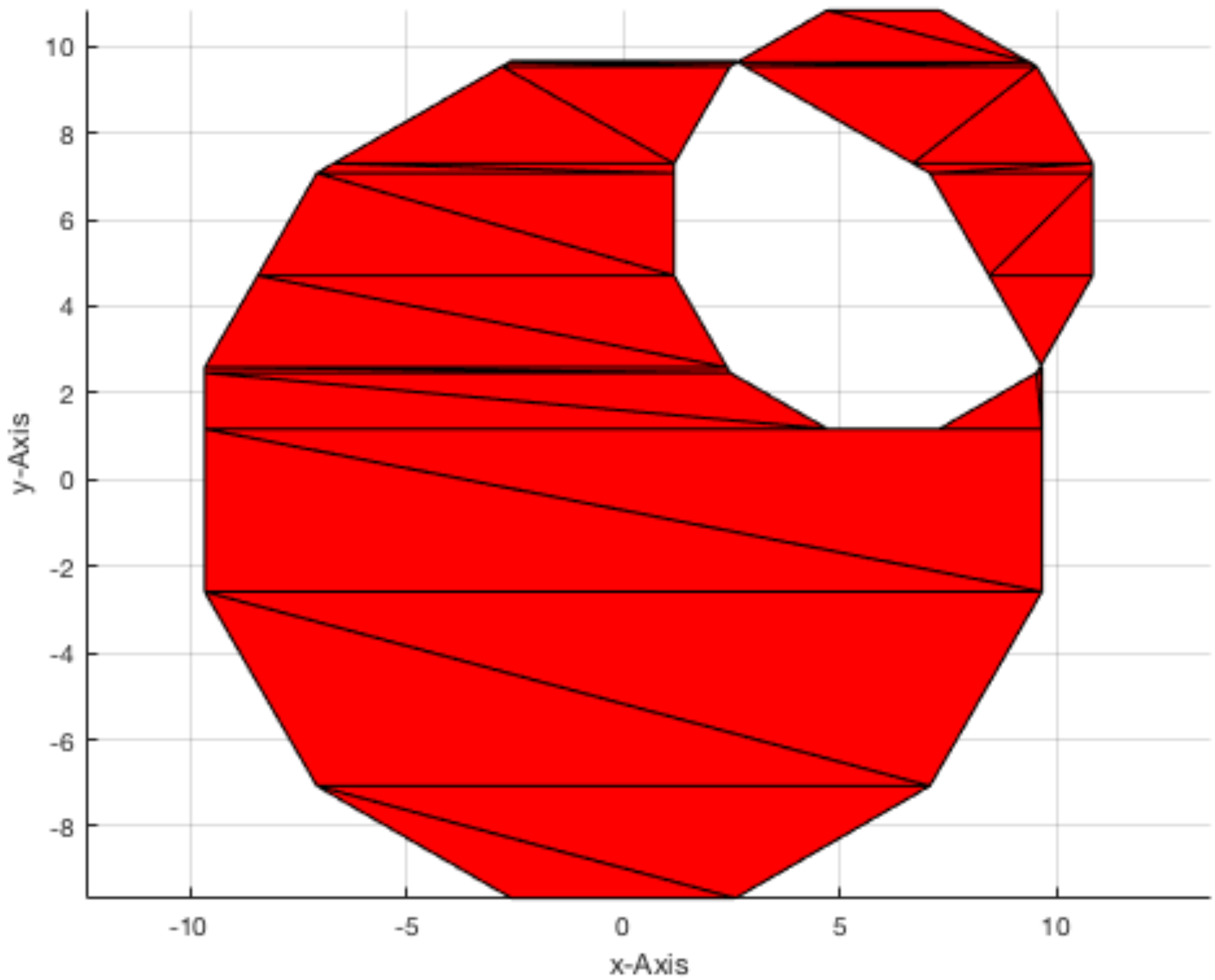

```
close all;
[PL,FL]=PLFLofCPLdelaunay(CPL); VLFLplot(PL,FL);
```

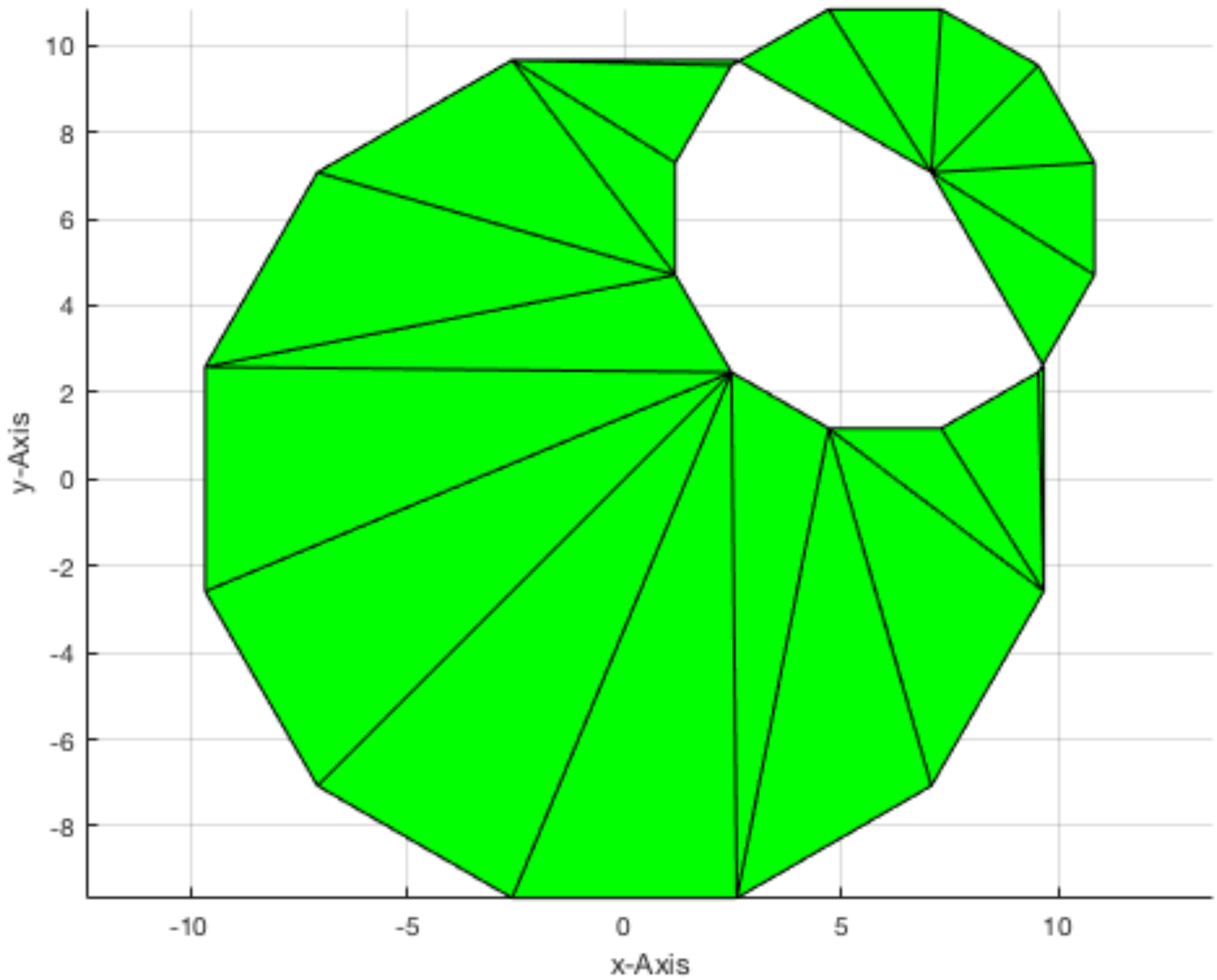


The next example shows the need for splitting contours:

```
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.
```





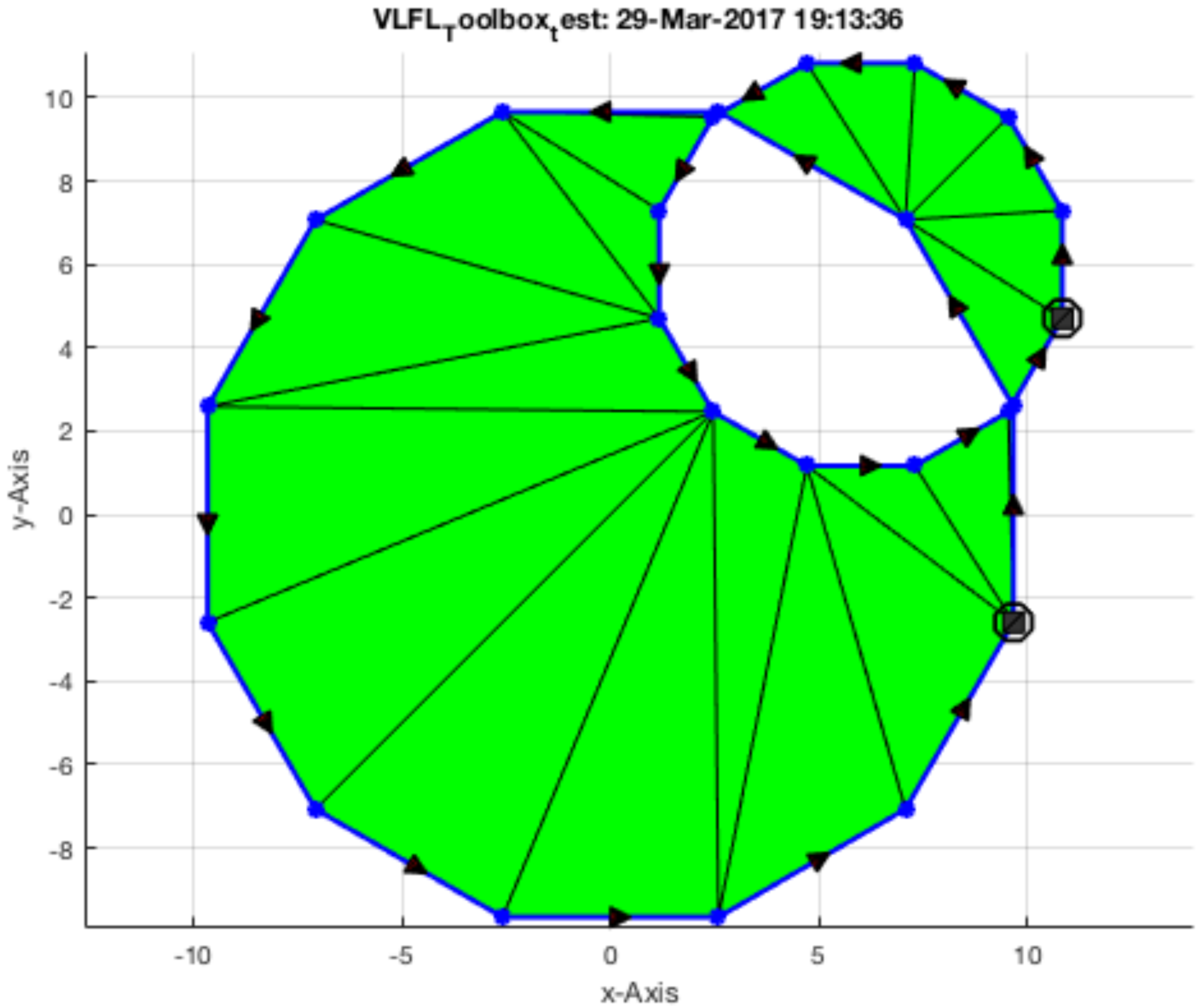
4. Orientation of outer and inner polygons of a CPL

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.

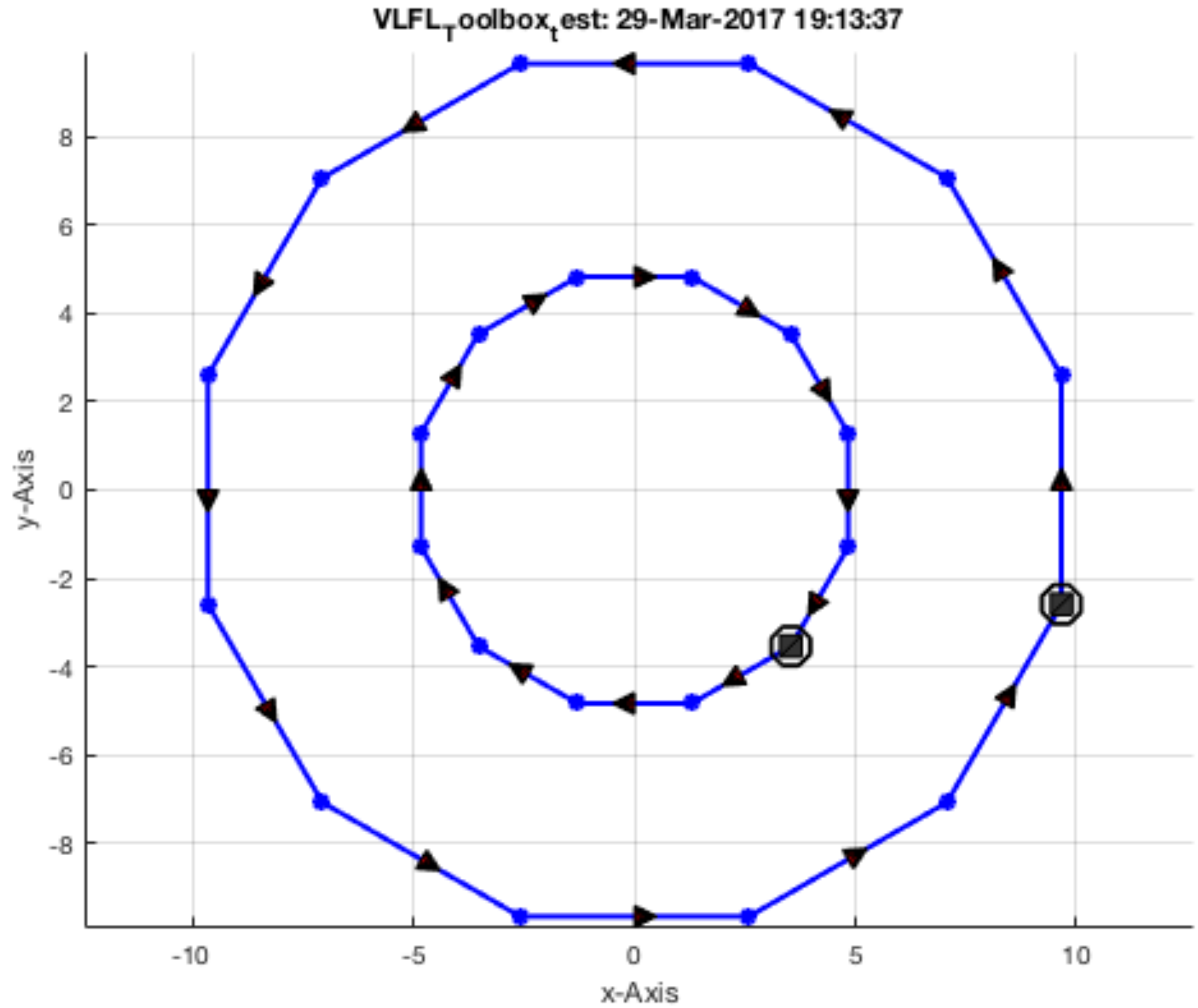
In the next figure, we see both polygons counter-clockwise:

```
PLELofCPL(CPL);
```



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

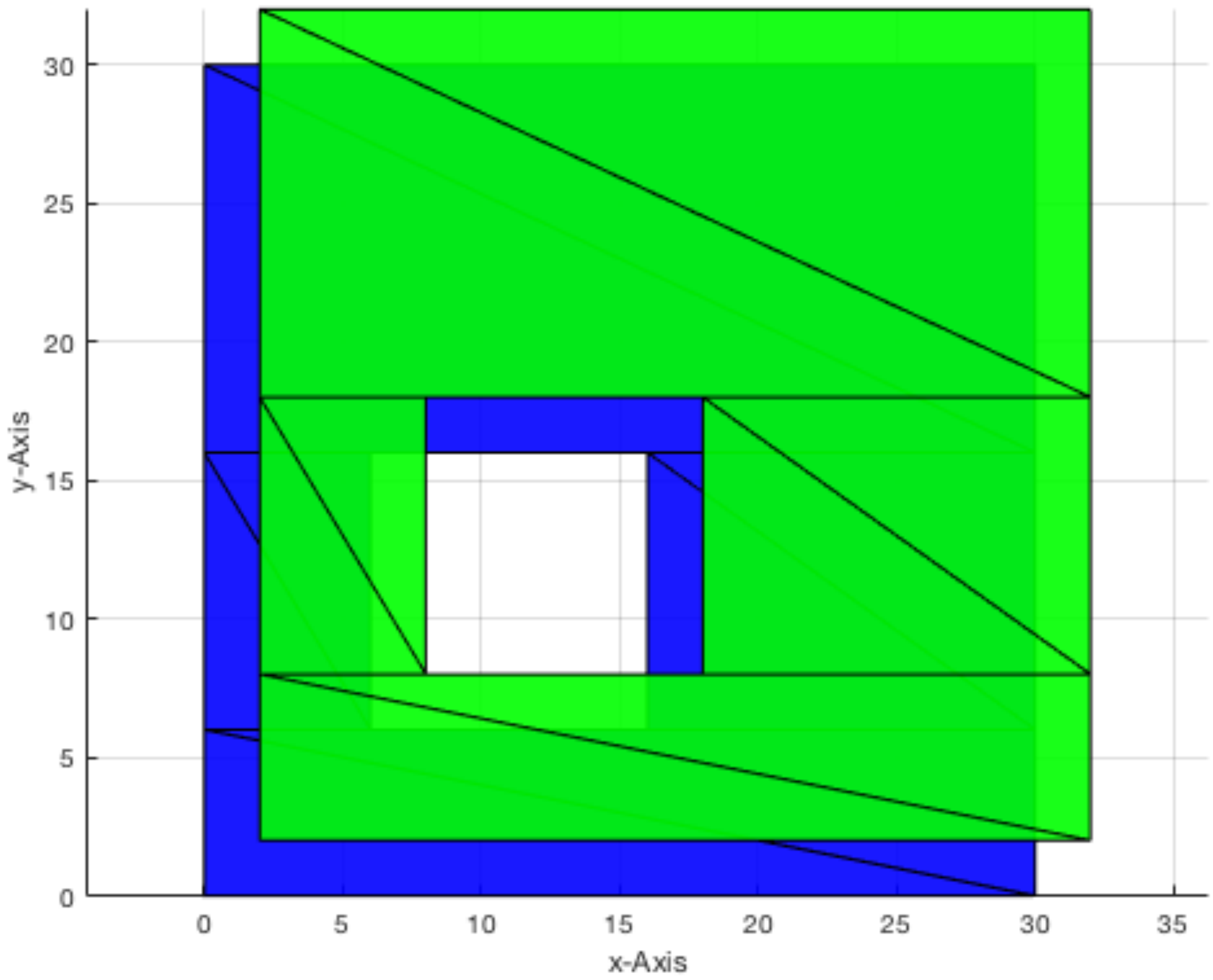
```
close all;
CPL=[PLcircle(10,12);NaN NaN;flip(PLcircle(5,12))];
PLELofCPL(CPL);
```



5. Boolean operations of contour polybool lists (CPL)

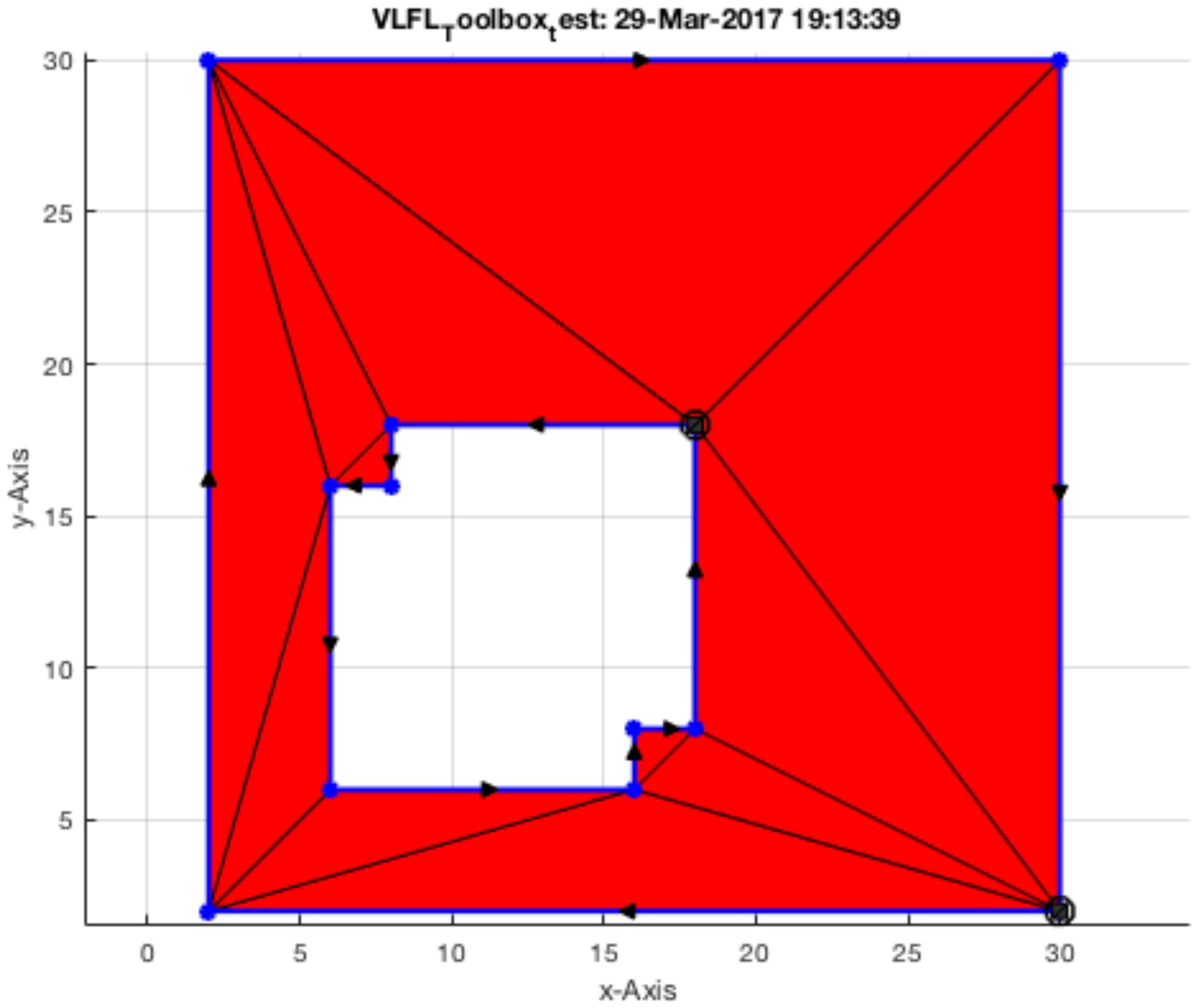
The main advantage of the CPL representation is currently the possibility to use the polybool functions of the mapping toolbox. Here, we show the use in embedded functions that help later to make the step forward to 3D modeling. We start with boolean operations of contour polybool lists (CPL).

```
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)
```



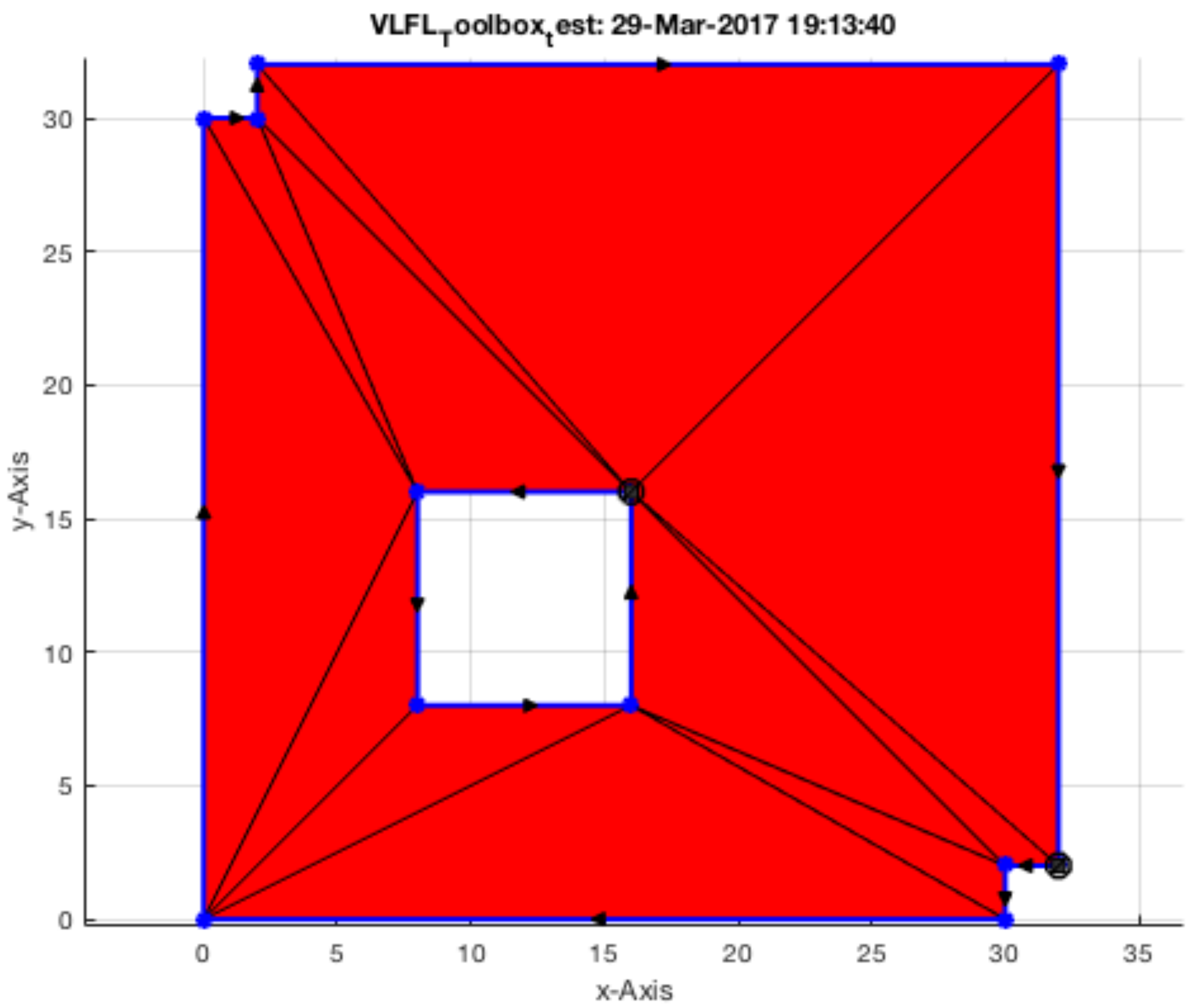
- **CPLpolybool('and',CPLA,CPLB)** delivers CPLA intersecting CPLB.

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



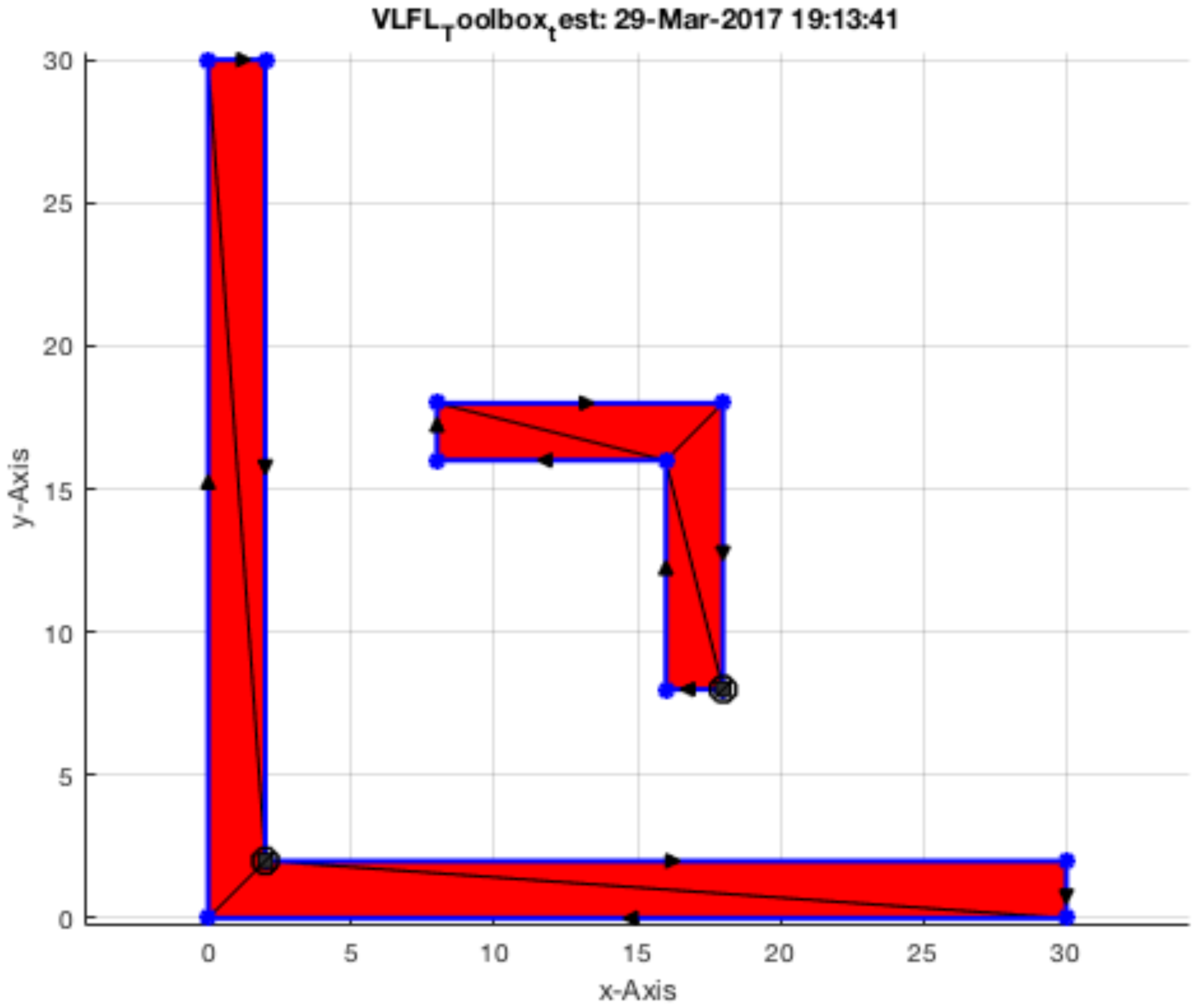
- **CPLpolybool('or',CPLA,CPLB)** delivers CPLA united with CPLB.

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



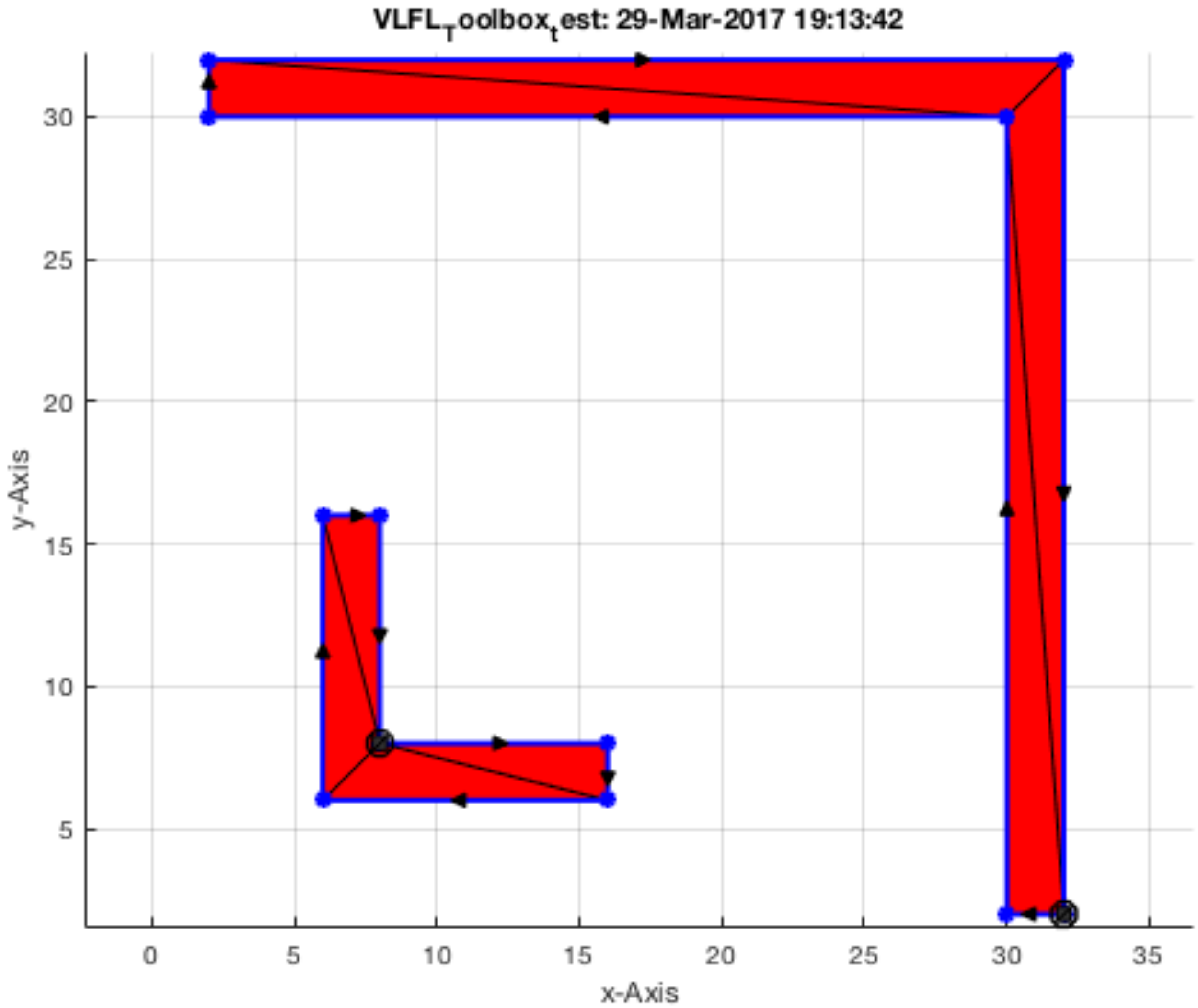
- **CPLpolybool('minus',CPLA,CPLB)** delivers CPLA minus CPLB.

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



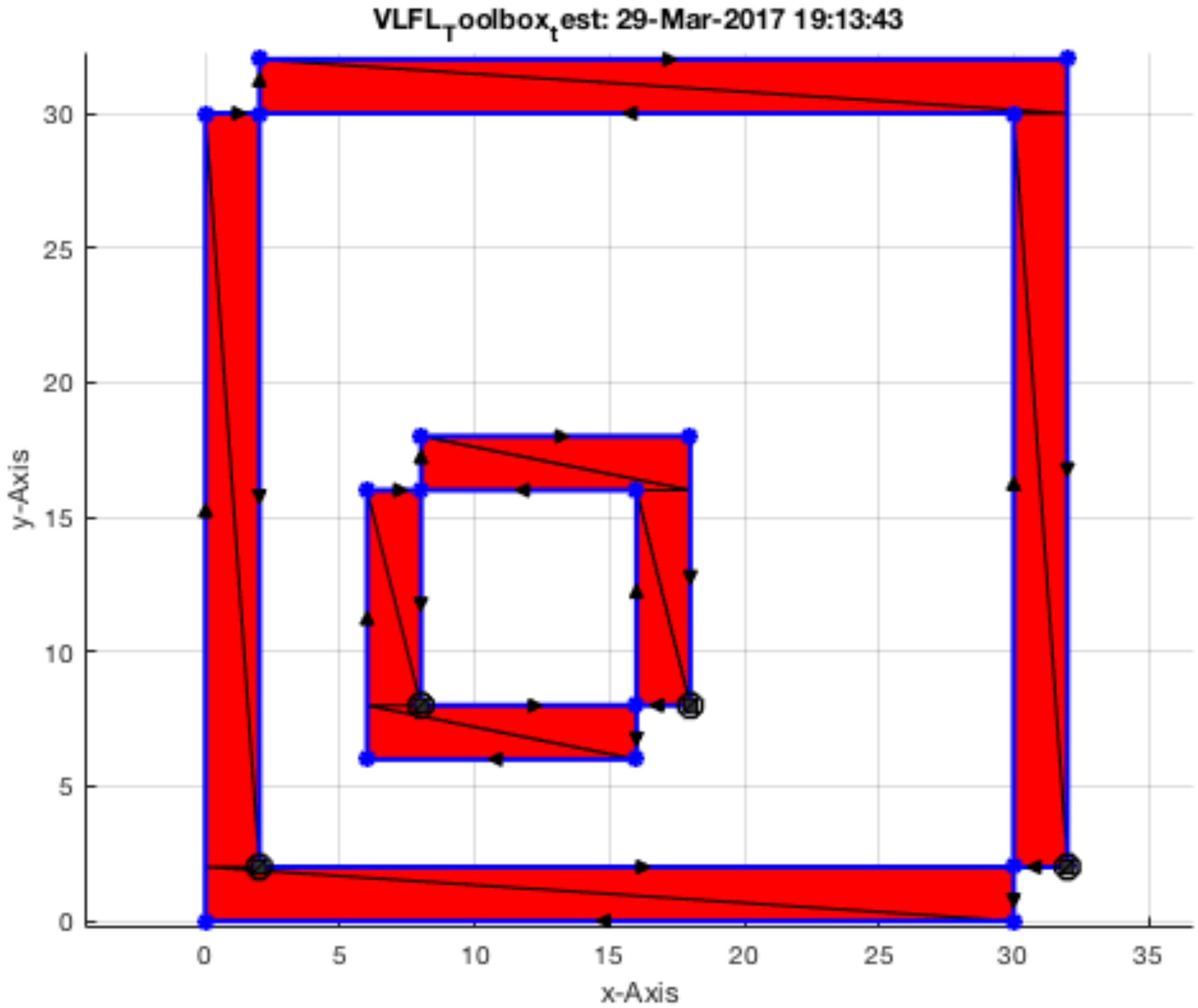
- **CPLpolybool('minus',CPLB,CPLA)** delivers CPLB minus CPLA.

```
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
```

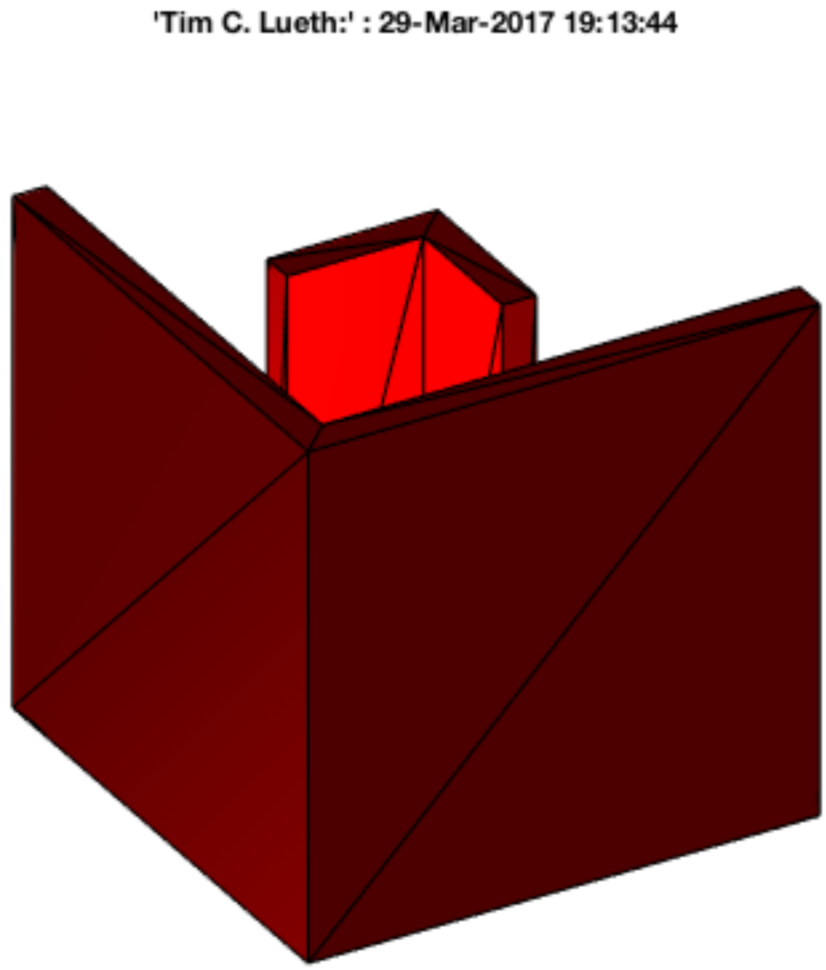
```
PL =

    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

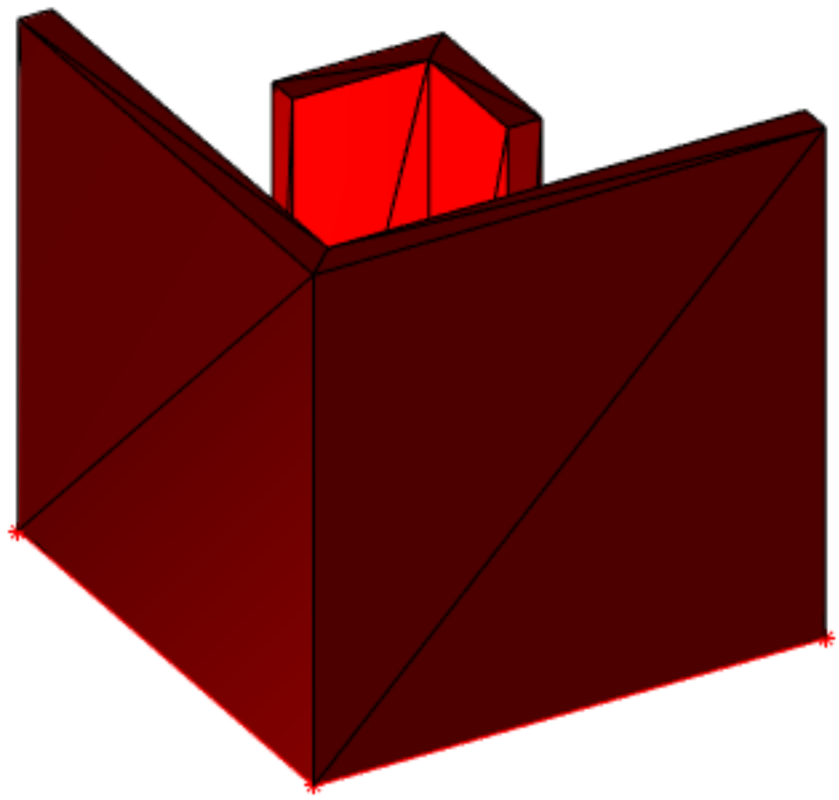
Finally, as it was possible to convert a CPL into and point list and an edge list, there is a function for the opposite.

- **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
```

'Tim C. Lueth:' : 29-Mar-2017 19:13:44



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:13:47 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In example 1-3 we've learned about:

- Point lists (PL), vertex lists (VL), facet lists (VL) and edge lists (EL)
- Functions for import, check, and export of STL-Data
- Closed polygon lists (CPL) for boolesche operations of embedded contours
- Orientation of inner and outer contours for 3D extrusion
- Different strategies for surface tesselation (Delaunay,row-scanning)
- Functions for generating/extruding surfaces and surface bounded solid volumes
- Functions for displaying points, surfaces, volumes different colors and transparency
- Functions for turning, mirroring and spatial transformation of objects
- Functions for generation of text string as solid objects

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

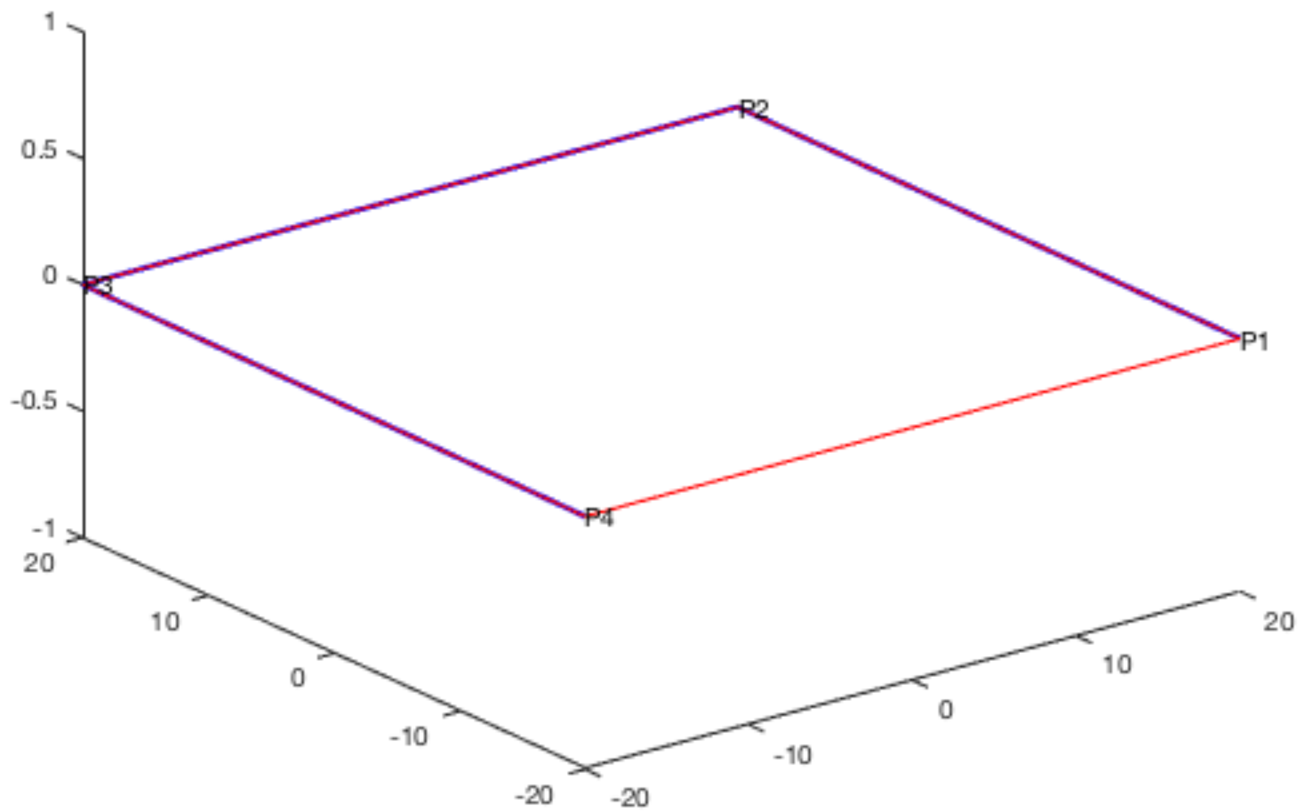
By using point lists (PL) and closed polybool lists (CPL) it is very convinient to design 2.5D objects. Here we see a first example to design a simple square by three function:

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
```

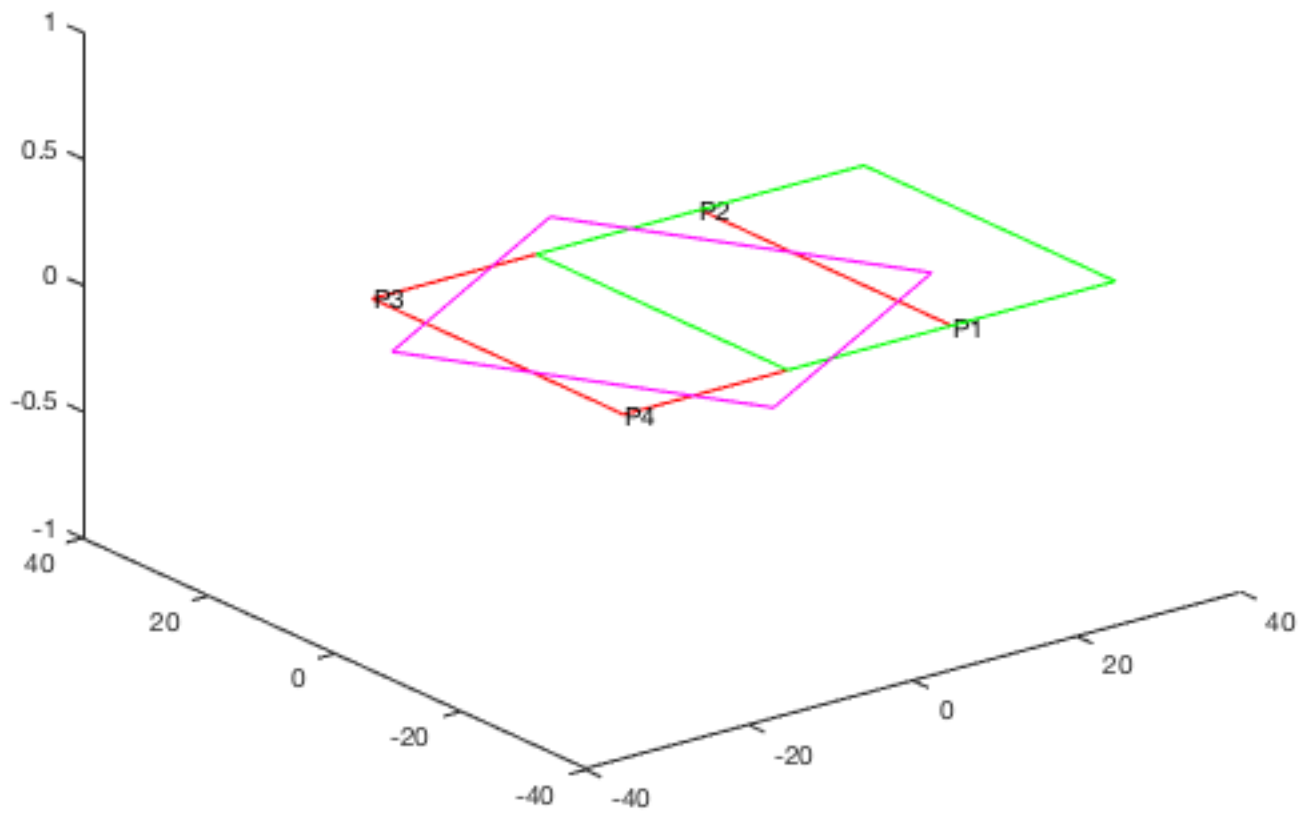
```
textVL (PLA);                              % Plots point descriptors
```



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
```

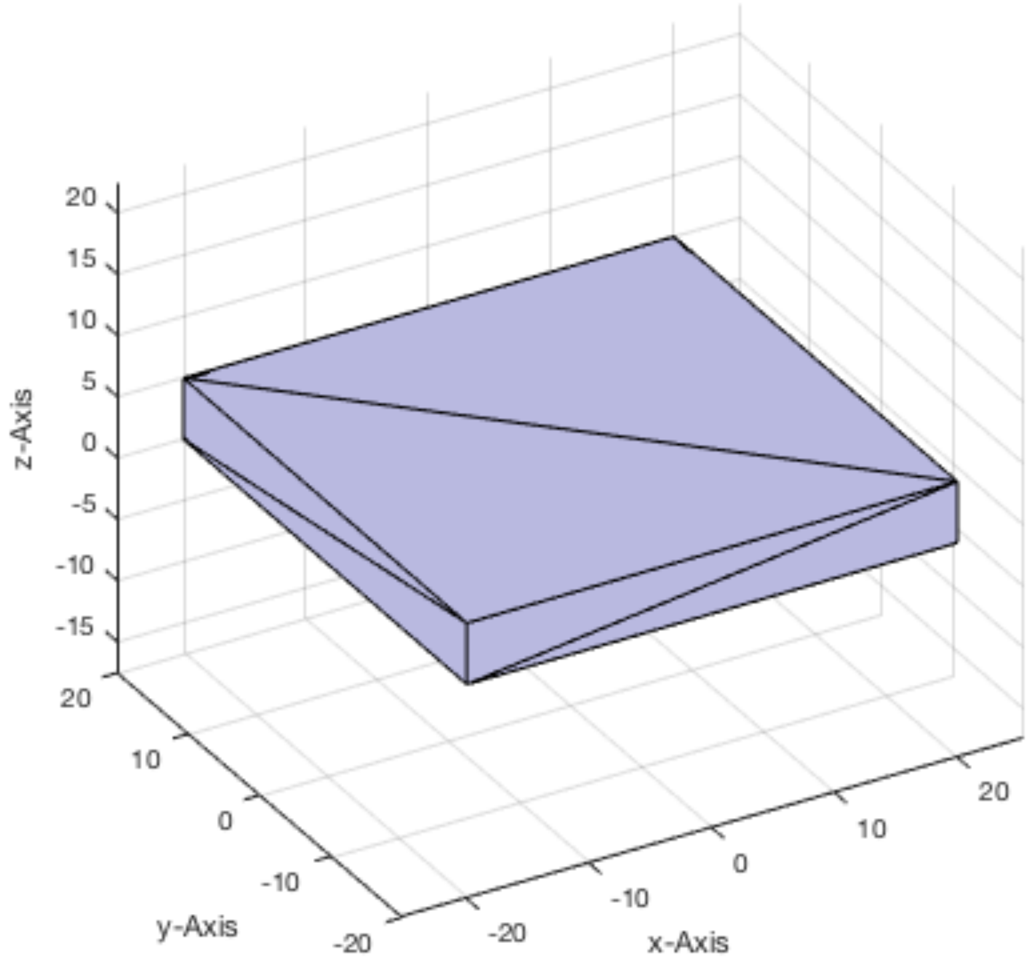


3. Simple extrusion of point lists (PL/CPL) to design 2½D solids

Next we extrude the square in 3D

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

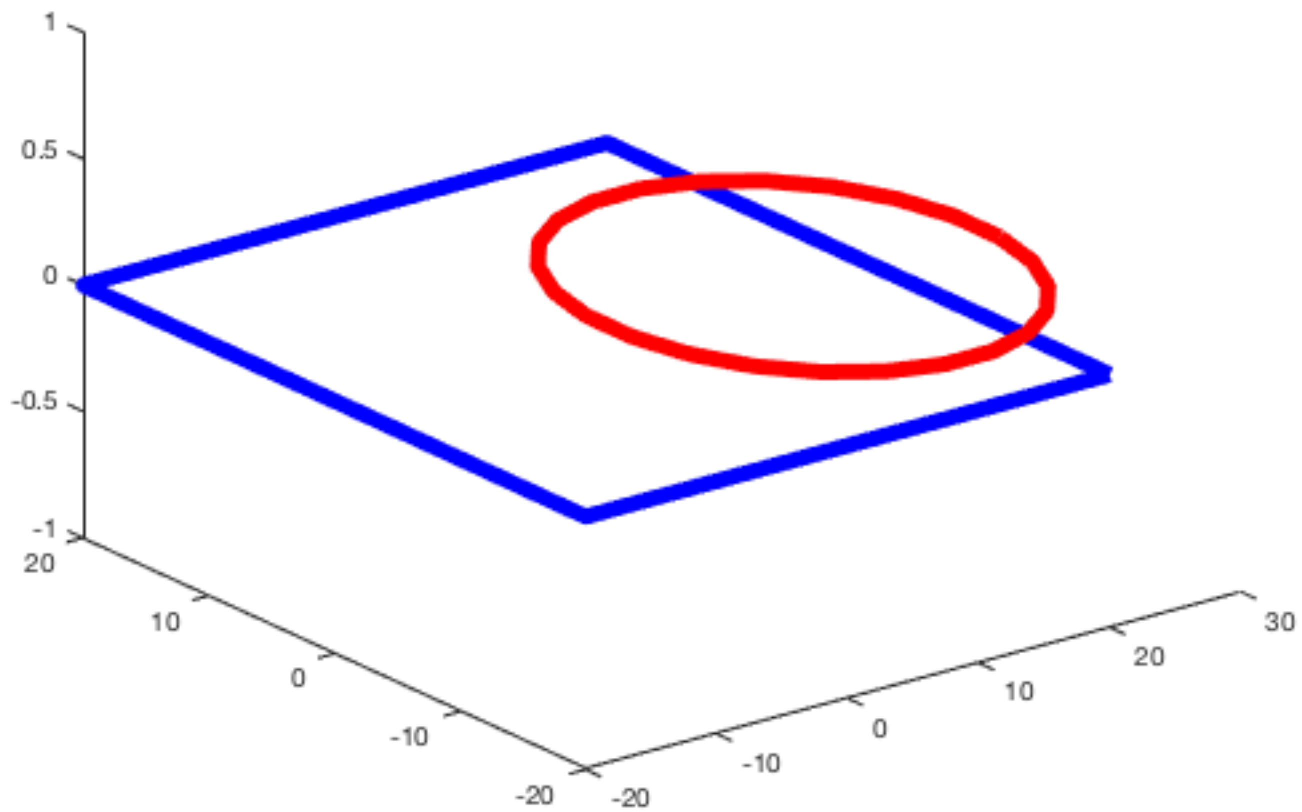
```
close all
[VL,FL]=VLFLofCPLz (PLA,5);
VLFLplot (VL,FL,'w'), axis equal, view(-30,30);
```



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

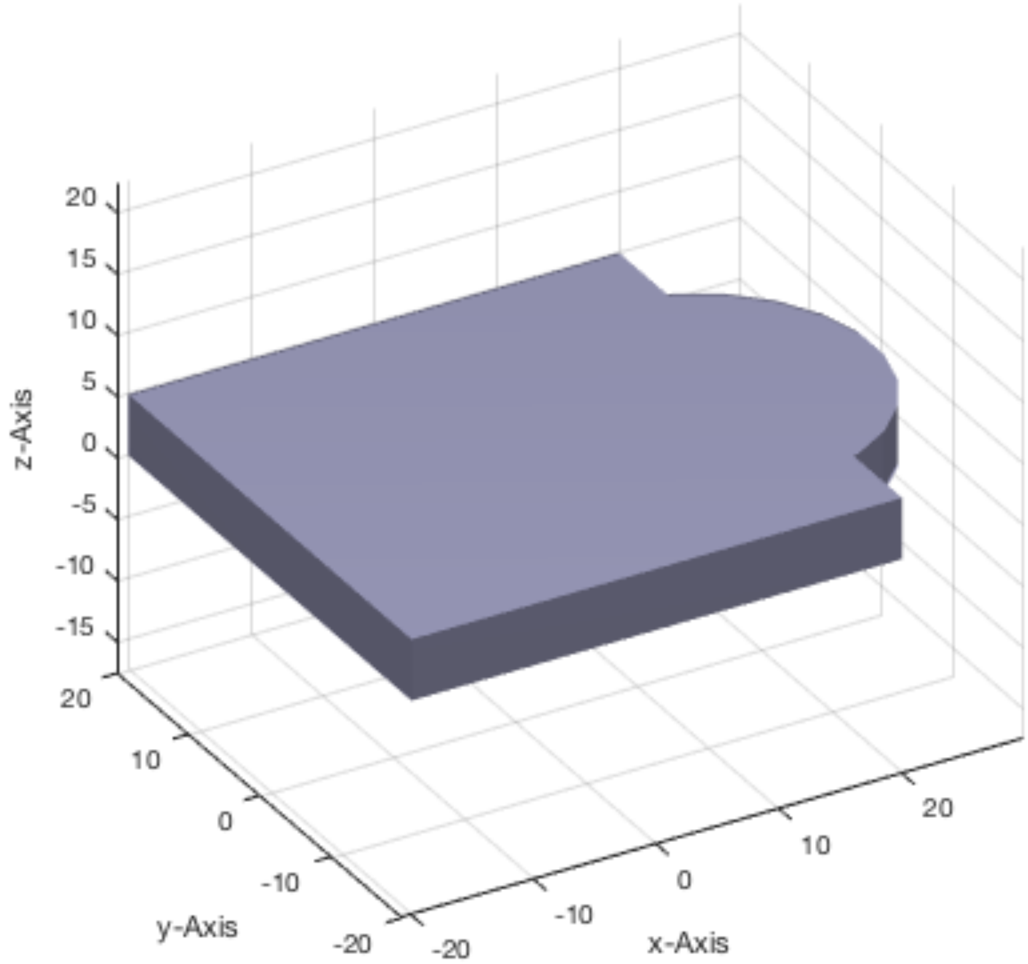
```
close all;
PLA=PLcircle(20*sqrt(2),4);
PLB=PLcircle(10*sqrt(2),24);  PLB=PLtransP(PLB,[15 0]);
CPLplot(PLA,'b',6); CPLplot(PLB,'r',6);
```



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')
```

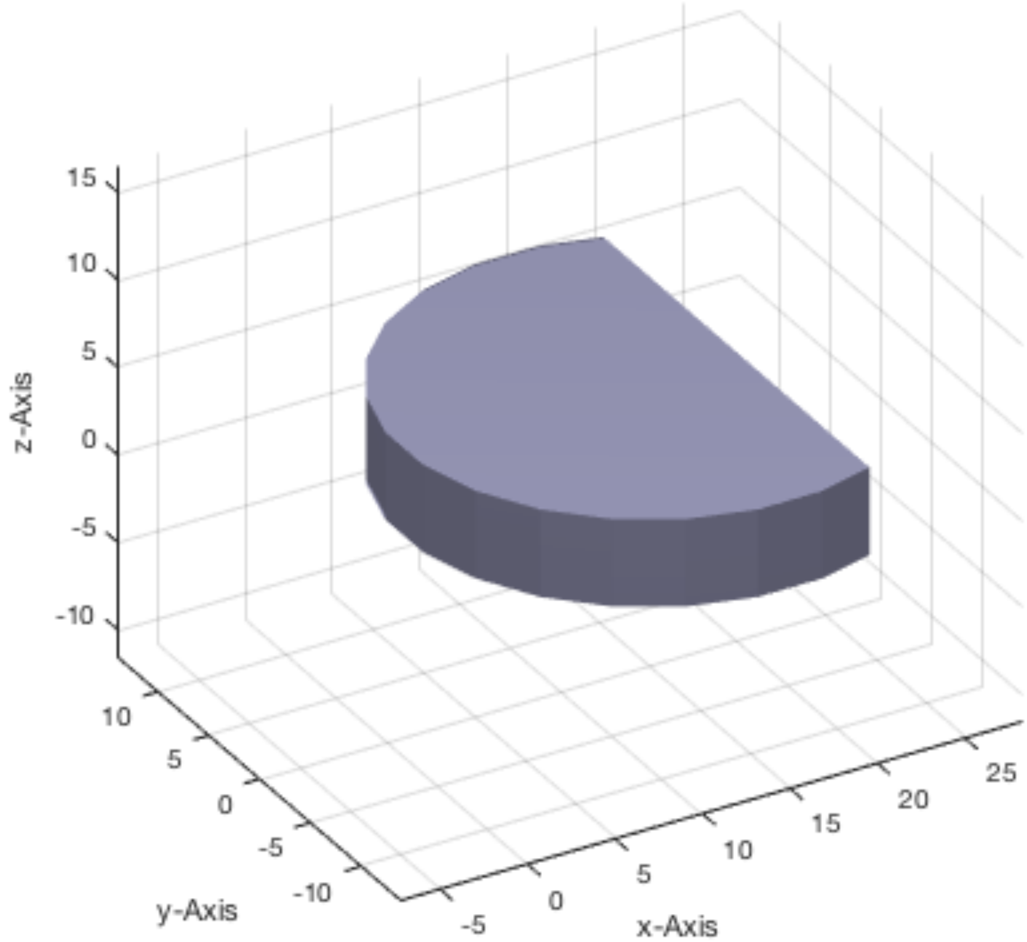
```
WRITING STL FILE /Users/lueth/Desktop/A.STL in ASCII MODE
completed.
```



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')
```

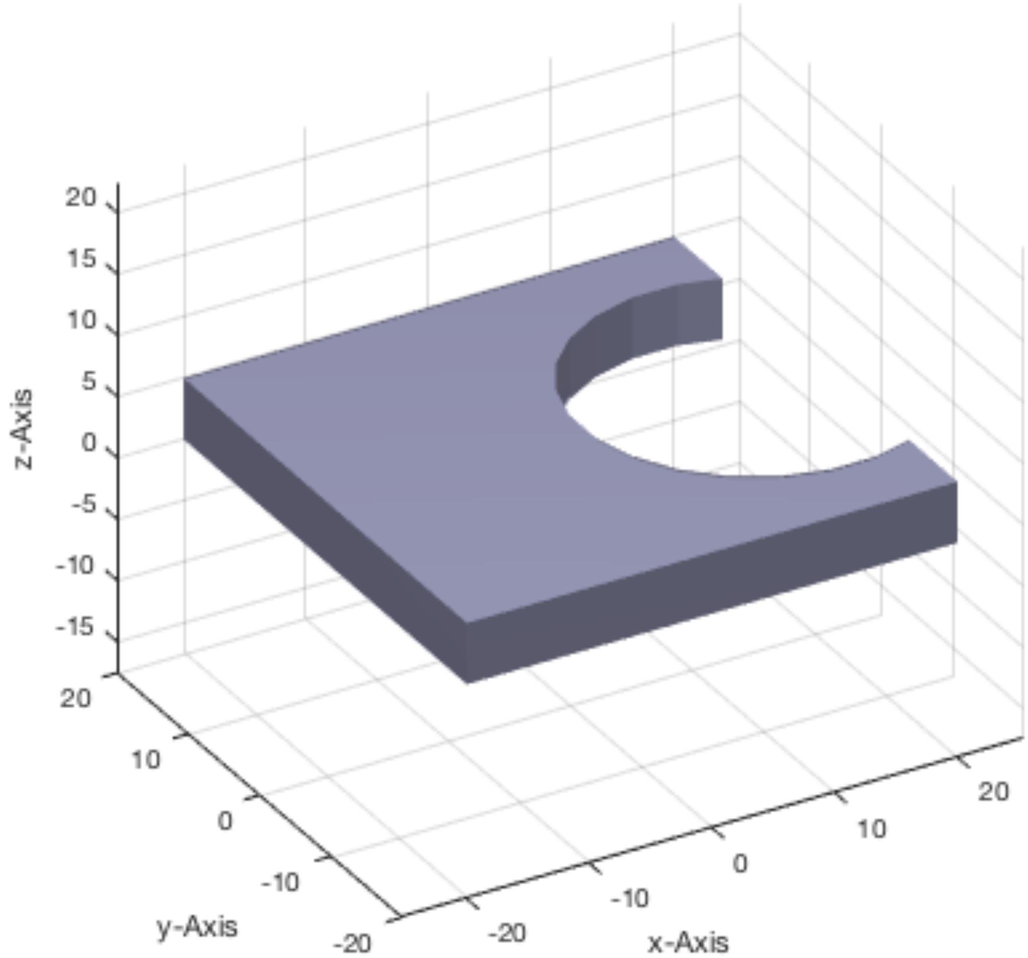
```
WRITING STL FILE /Users/lueth/Desktop/A.STL in ASCII MODE
completed.
```



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')
```

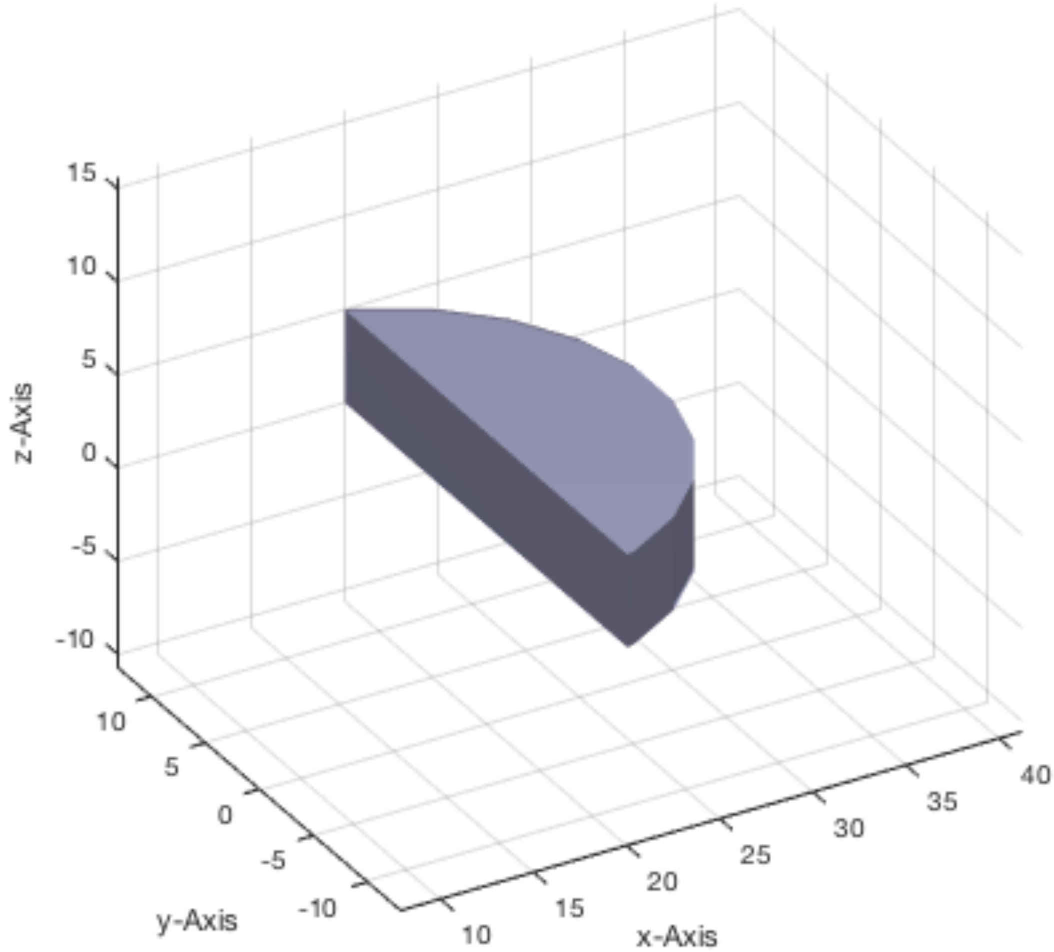
```
WRITING STL FILE /Users/lueth/Desktop/A.STL in ASCII MODE
completed.
```



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')
```

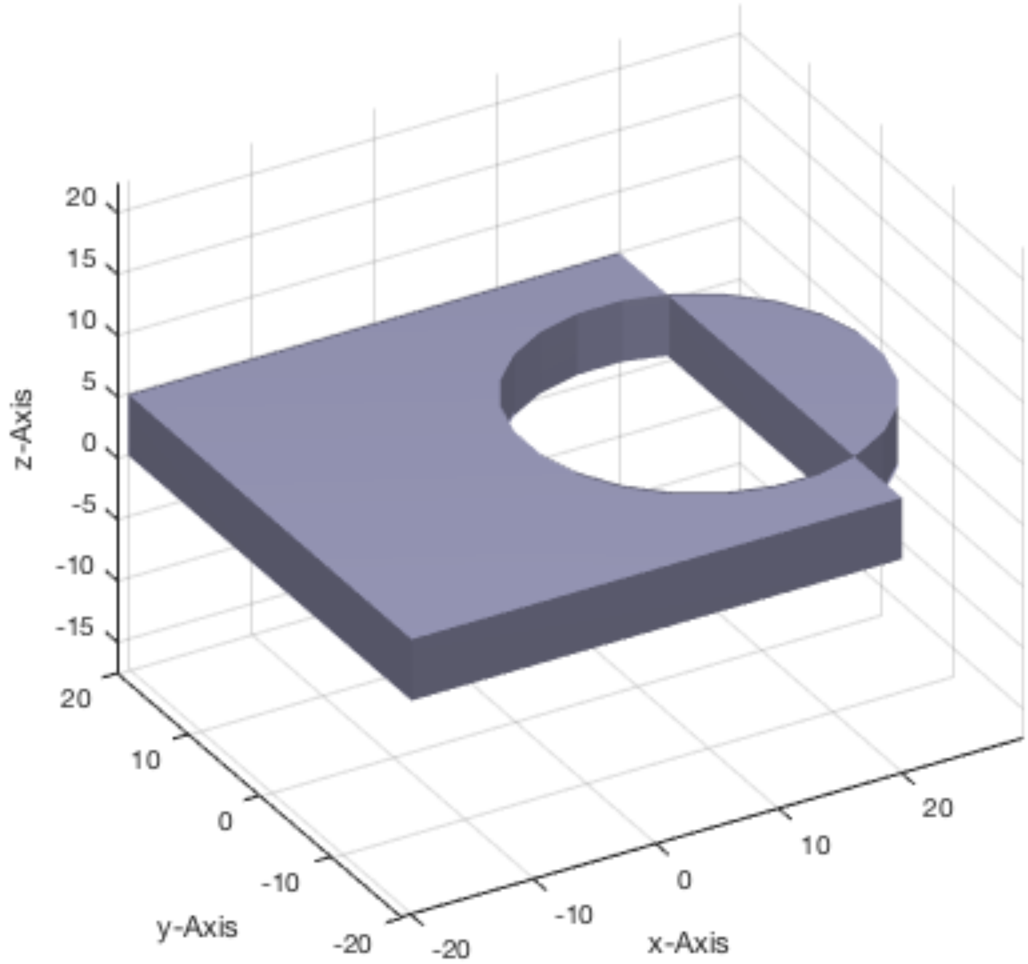
```
WRITING STL FILE /Users/lueth/Desktop/EXP04-AminusB.STL in ASCII MODE
completed.
```



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 property X.
WRITING STL FILE /Users/lueth/Desktop/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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:13:53 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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*

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-4 we've learned about:

- Point lists (PL), vertex lists (VL), facet lists (VL) and edge lists (EL)
- Functions to import, check, and export of STL-Data
- Functions for generating text strings as solid objects
- Closed polygon lists (CPL) for boolesche operations of even embedded contours
- 2½D design by extrusion of closed polyogon lists
- Spatial transformation of objects in 2D and 3D

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

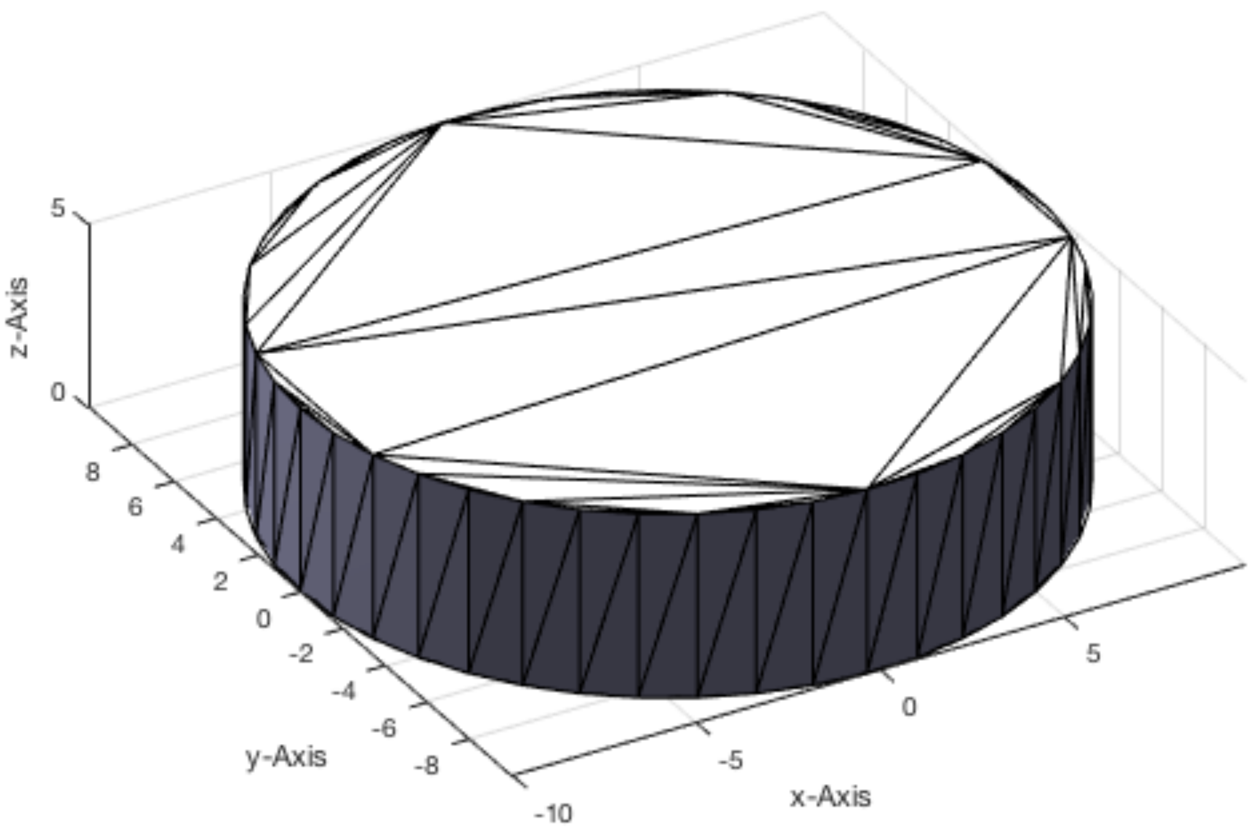
Even if it is useful to know that a vertex list (VL), and a facet list (FL) is required for 3D modeling, it is more convenient to use matlab structs for solid geometries (SG). Instead of writing VL or FL, we use SG.VL, SG.FL as variables. The advantage is, that each solid object is described by one struct 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');
```

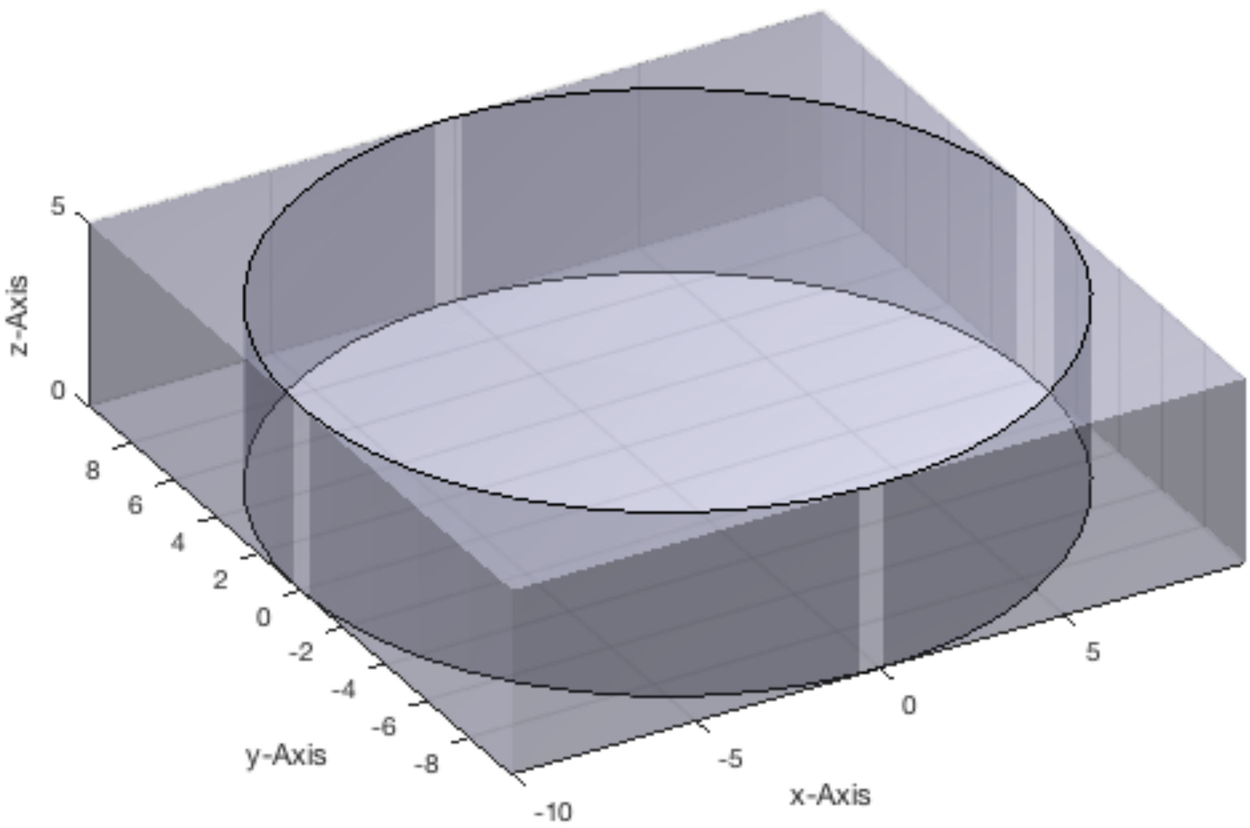
```
nSG =

  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);
```

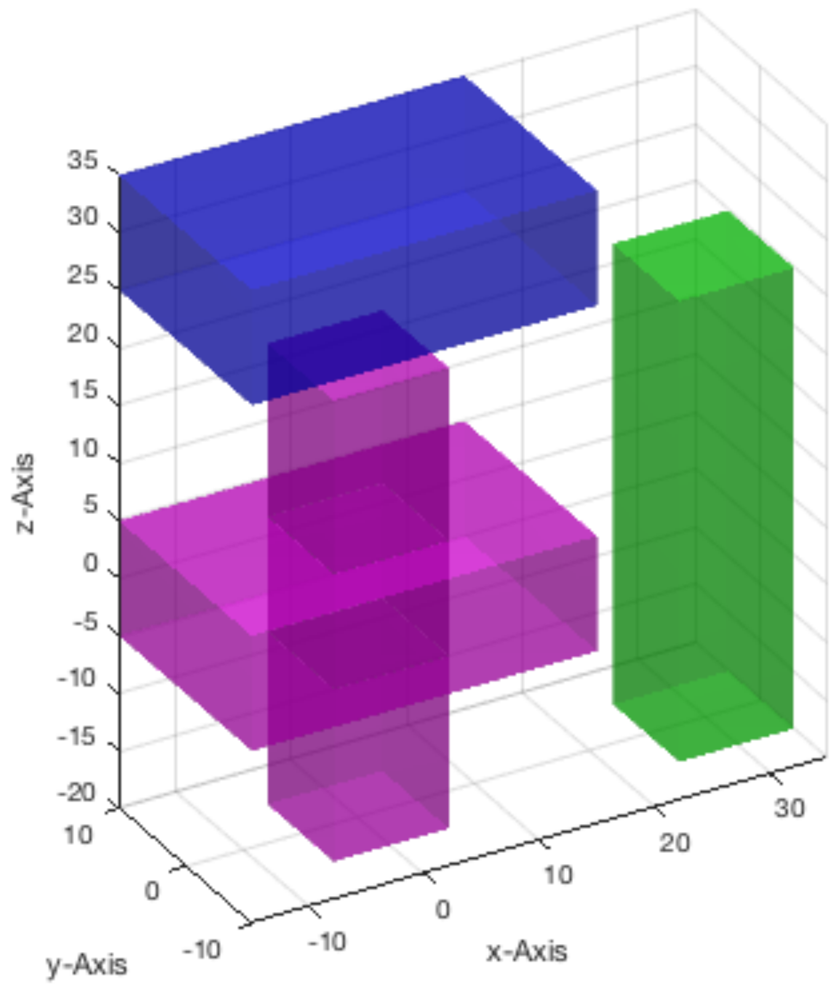


3. Spatial transformations of solid geometries and *sets of solid geometries*

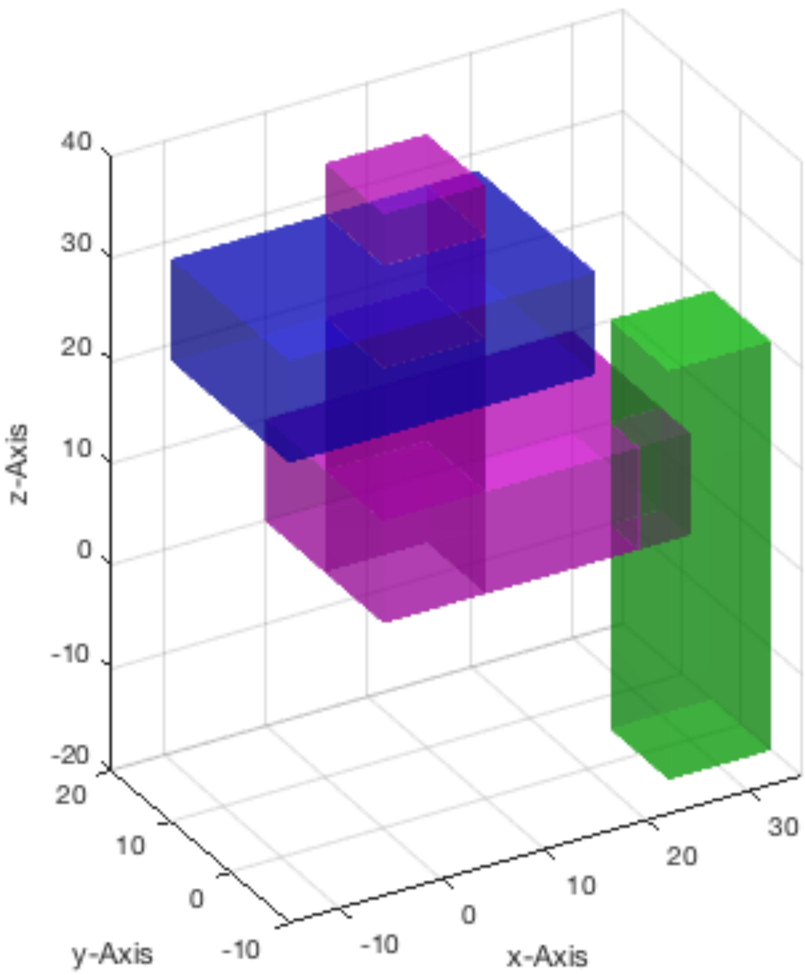
In contrast to manipulate an individual solid geometry as struct, it is often useful to manipulate or handle a set of solid geometries. For this purpose, we use the cell concept of Matlab. A=SGbox([30,20,10]); {A, A, A} is a set of three solid geometries that can be given as arguments of a function and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

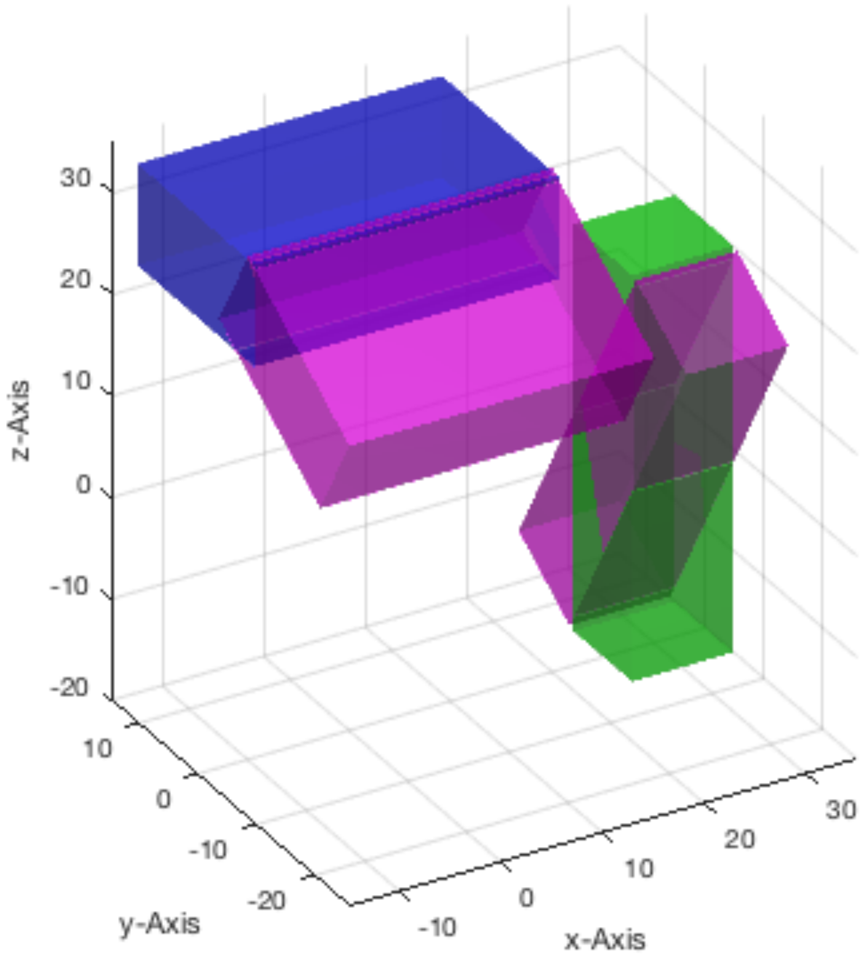
```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtrans0({A,B}),'m'); VLFLplotlight (1,0.5)
```



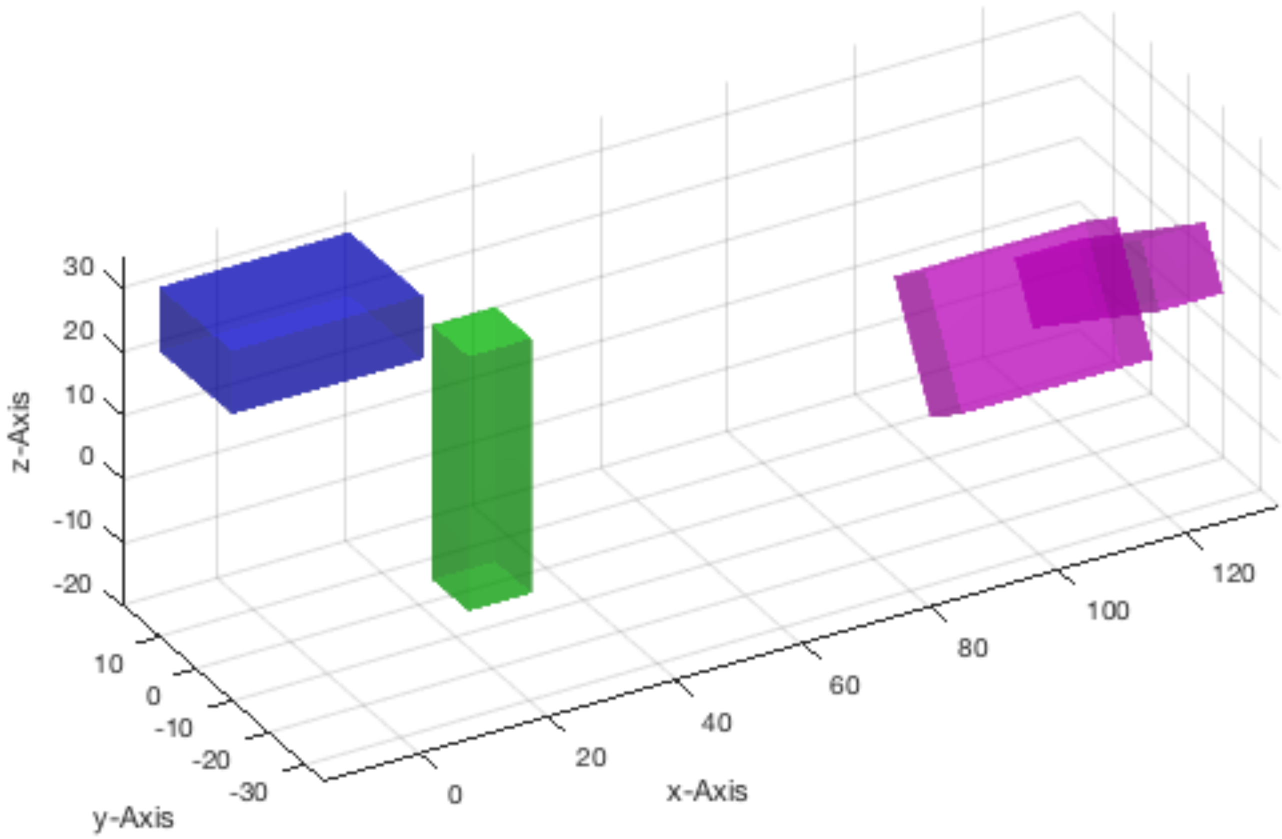

```
close all;
A=SGtransP(SGbox([30,20,10]),[0;0;30]); SGplot (A,'b'); show;
B=SGtransP(SGbox([10,10,40]),[30;0;0]); SGplot (B,'g'); show
view (-30,30);
SGplot(SGtrans1({A,B}),'m'); VLFLplotlight (1,0.5)
```




```
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)
```

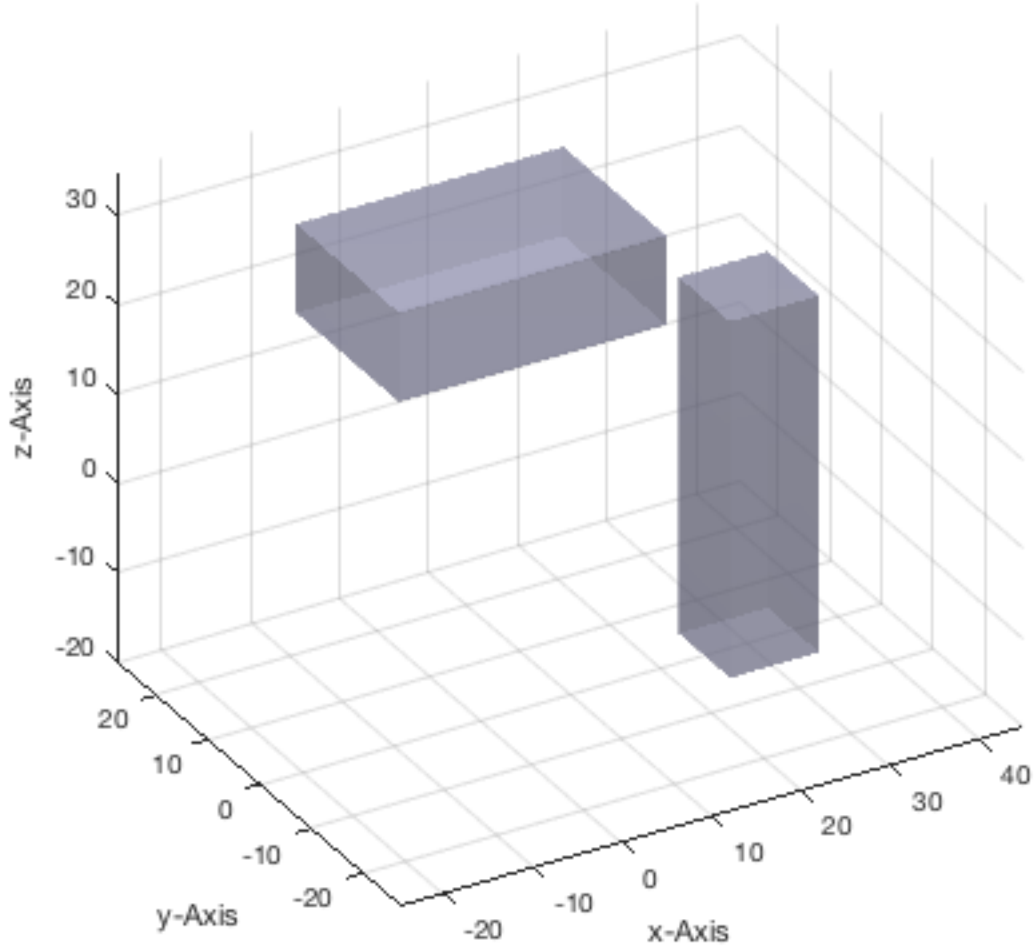


4. Merging of solid geometries (SG) and sets of solid geometries

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');
```

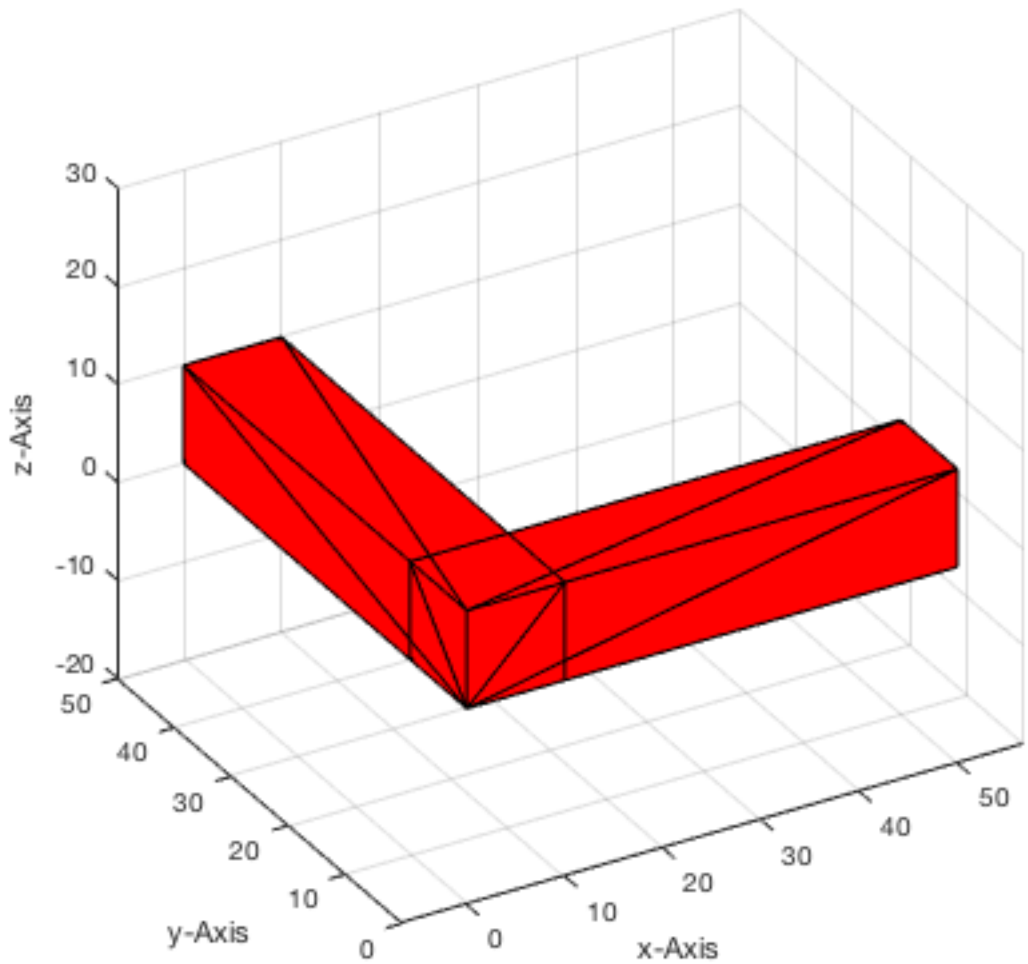


5. Non-manifold points, edges, and facets of solid geometries (SG)

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.

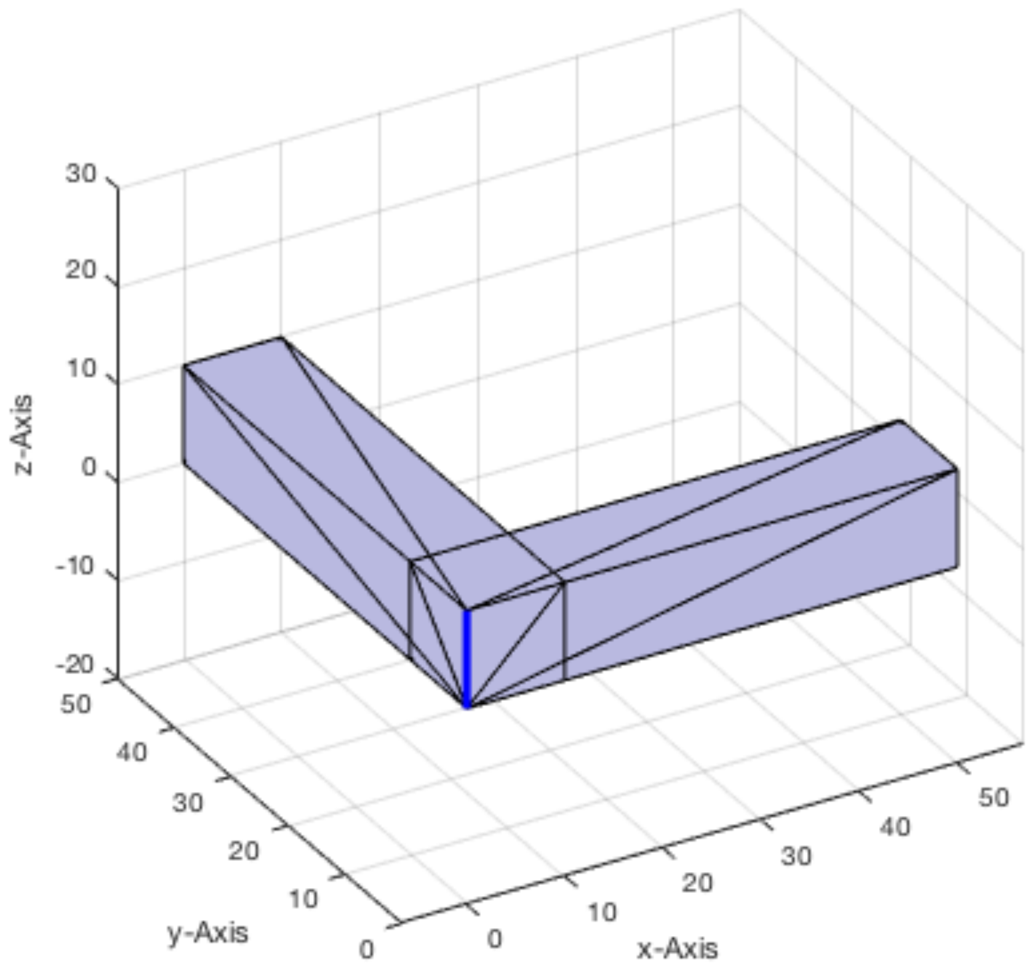
```
close all;
A=SGtrans1(SGbox([10,50,10])); B=SGtrans1(SGbox([50,10,10]));
nSG=SGcat({A,B});  SGplot (nSG); view (-30,30);
[VL,EL,PEL]=SGchecker (nSG);
if ~isempty(VL); VLELplots (VL(:,2:4),PEL,'m*-',4); VLFLplotlight (1,0.7); end
```



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
```

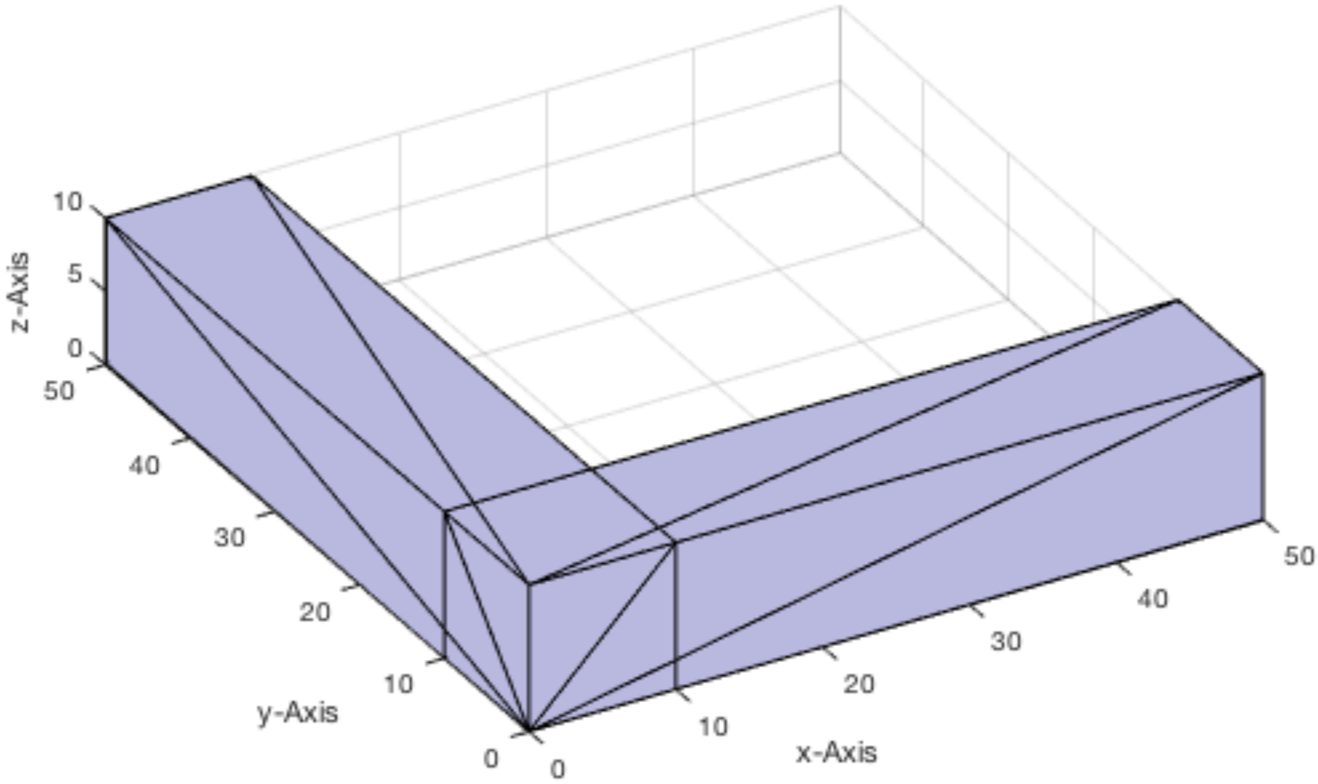


6. Additive Design: Separate or penetrate solid geometries (SG)

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');
```

```
==> SGchecker: 2 Surfaces to check.

Surface 1:
Surface 2:
```



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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:14:03 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-5 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polygon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets

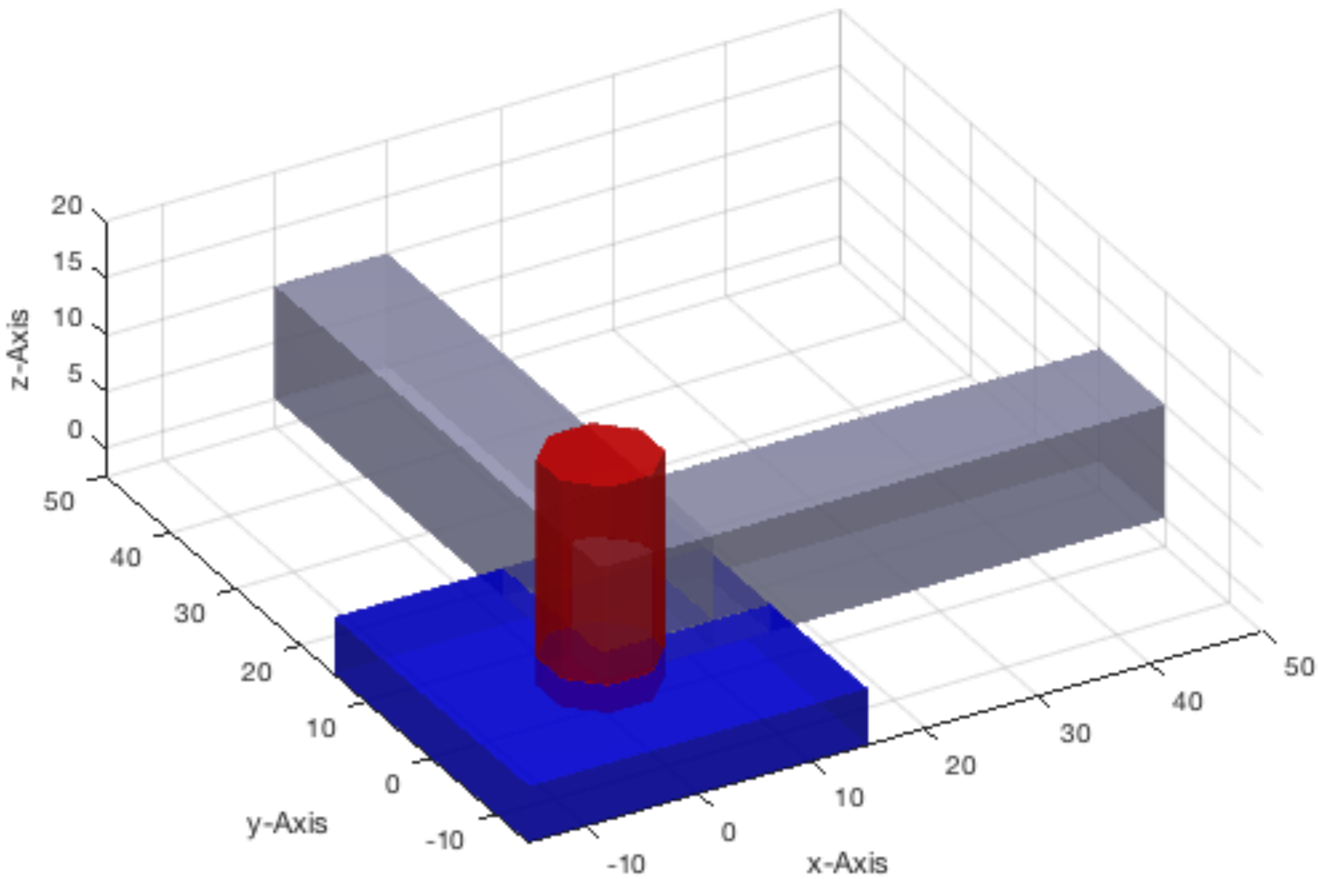
2. Relative spatial positioning of solid geometries (SG) using bounding boxes

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'

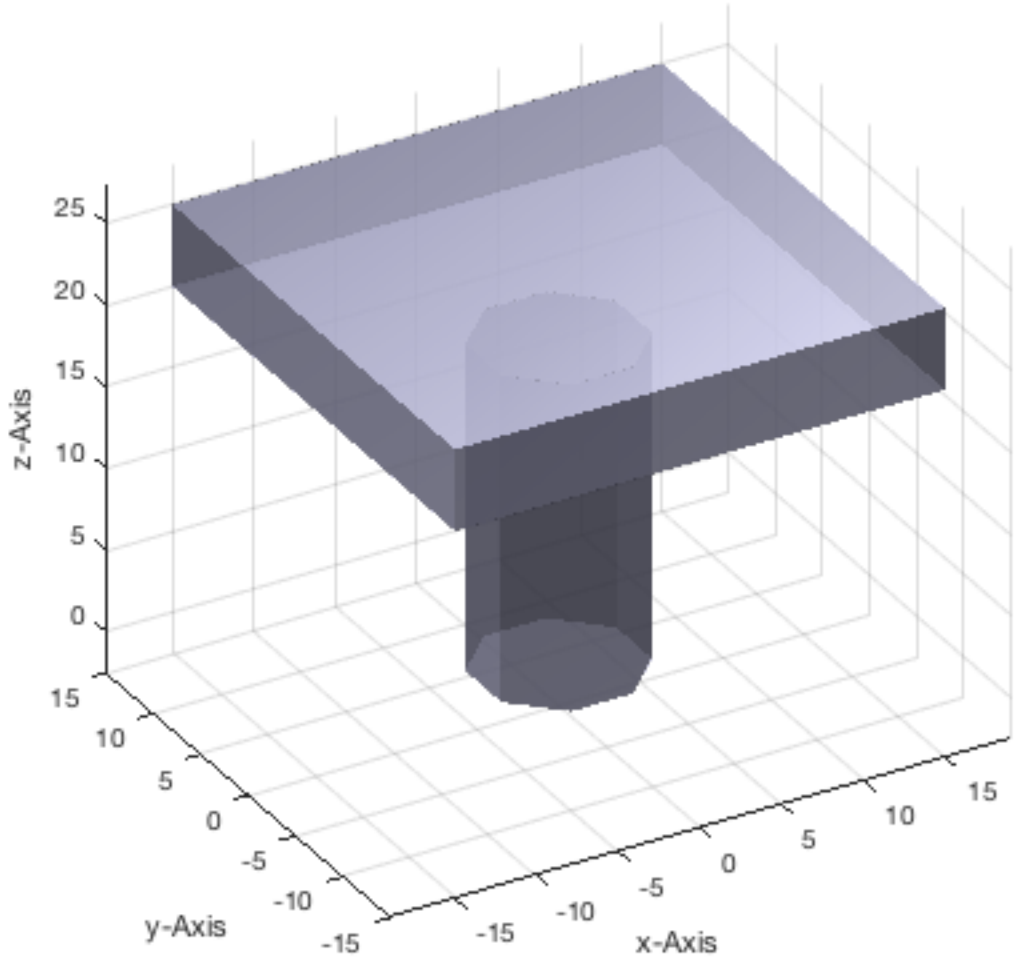
Define two solid geometrys 'A' and 'B'

```
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);
```



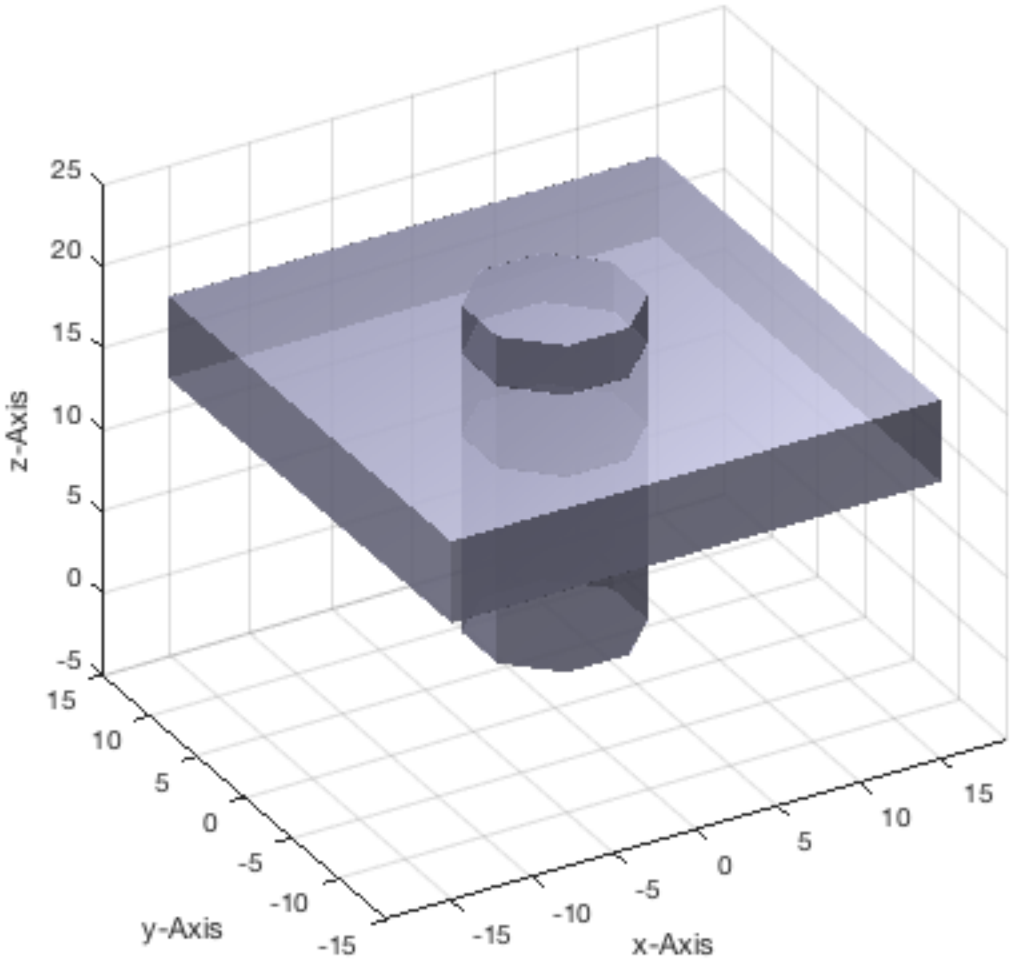
SGontop positions a solid geometry 'A' on top of solid geometry 'B'

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



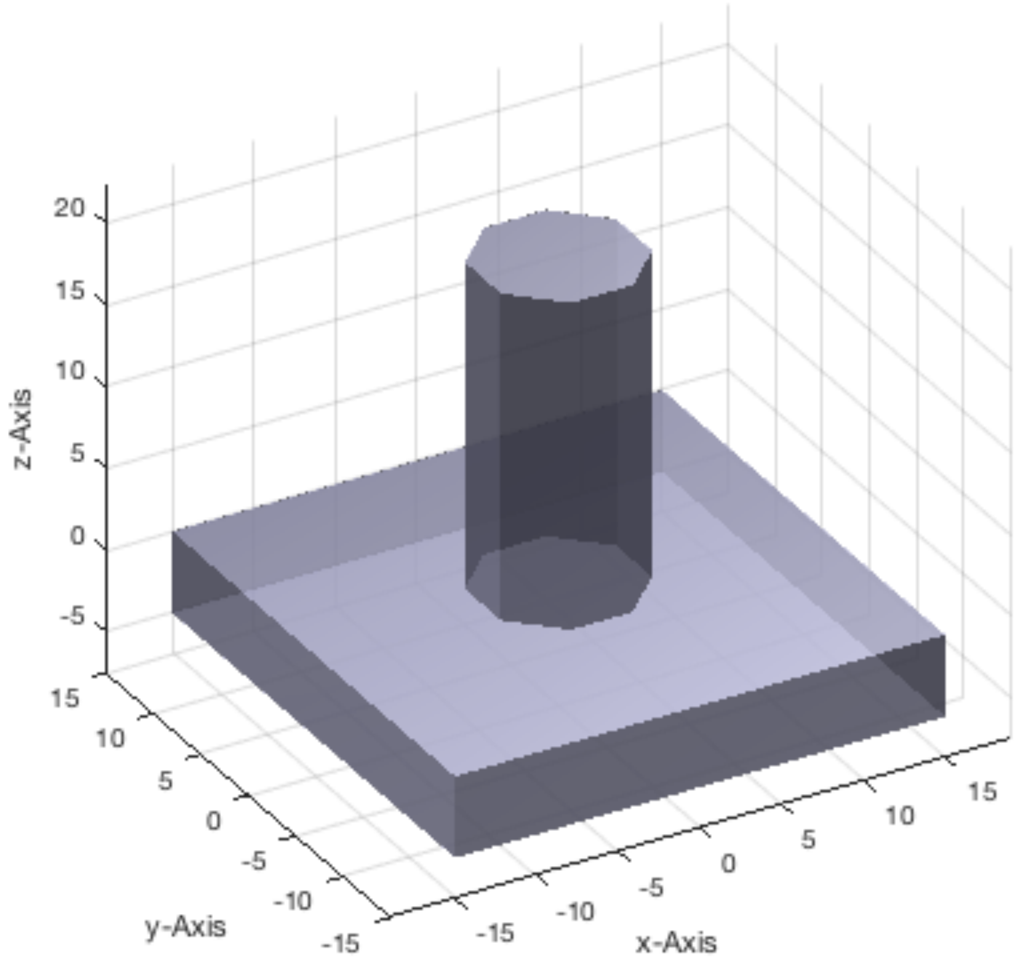
SGontop positions a solid geometry 'A' on top of solid geometry 'B' with a gap of -8

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



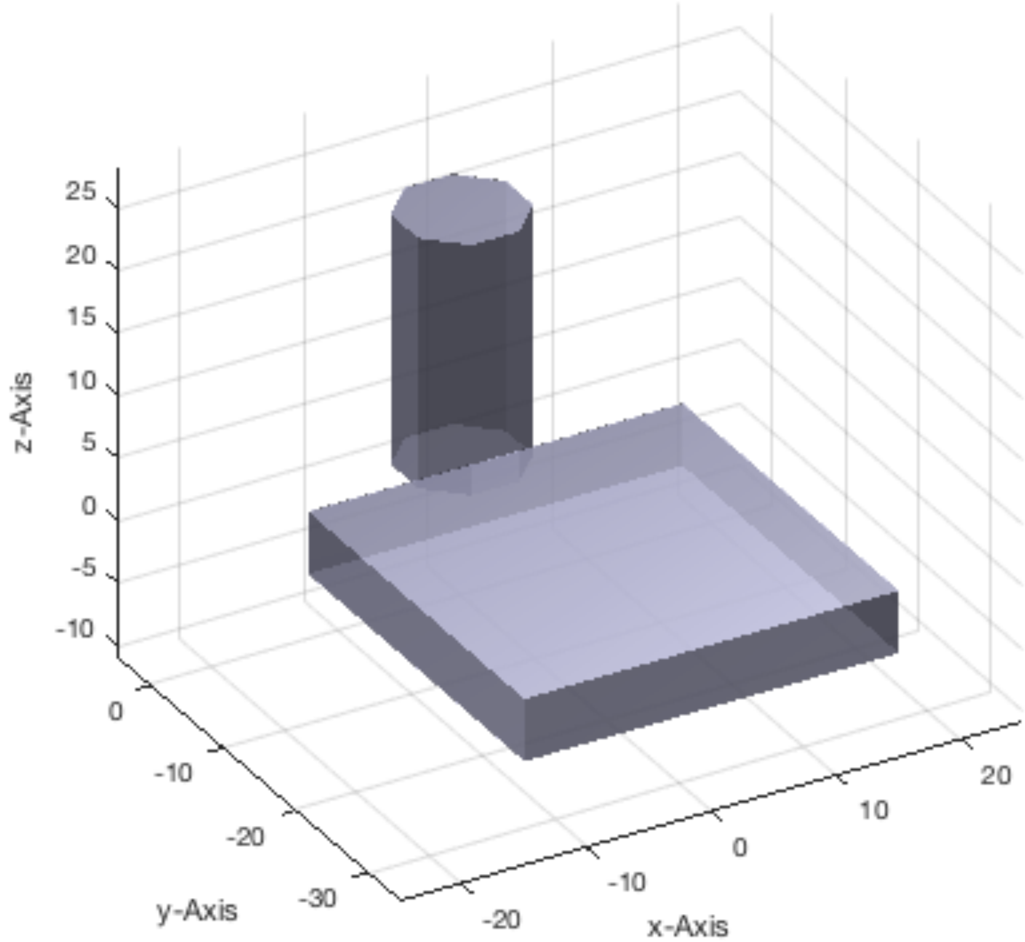
SGunder positions a solid geometry 'A' under of solid geometry 'B'

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



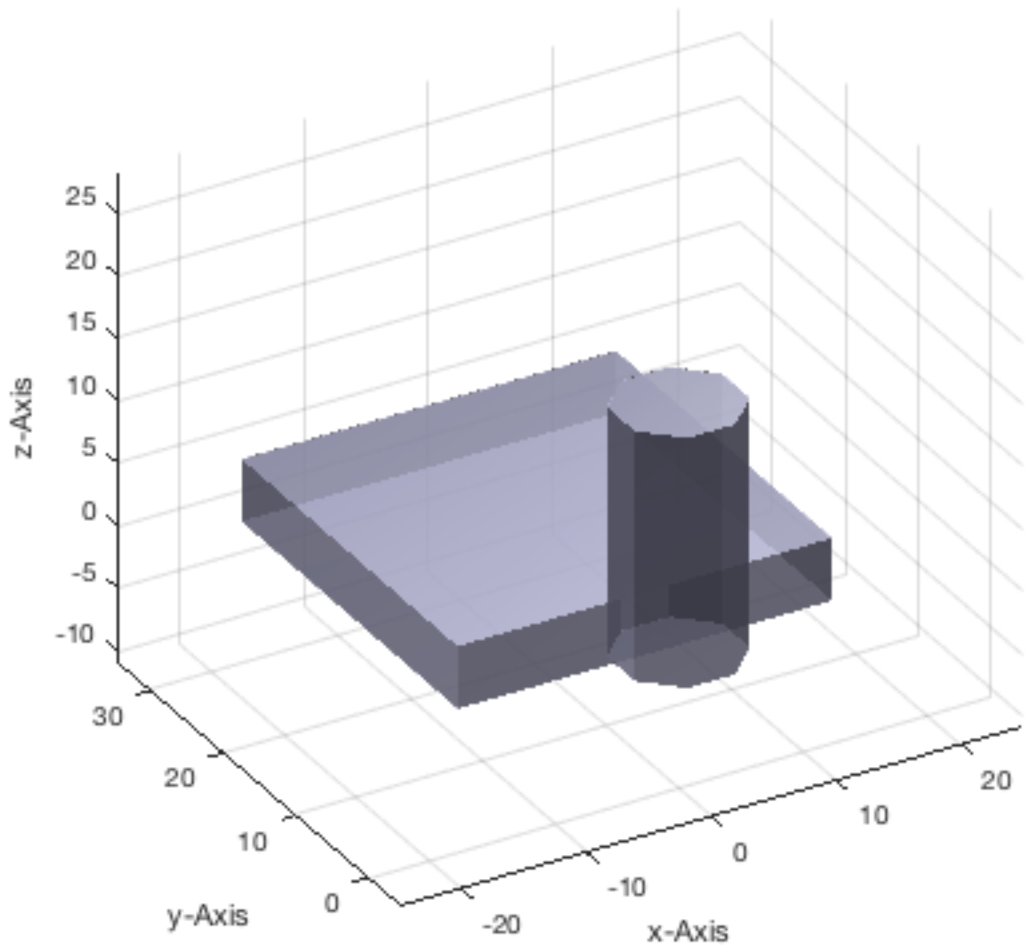
SGinfront positions a solid geometry 'A' in front of solid geometry 'B'

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



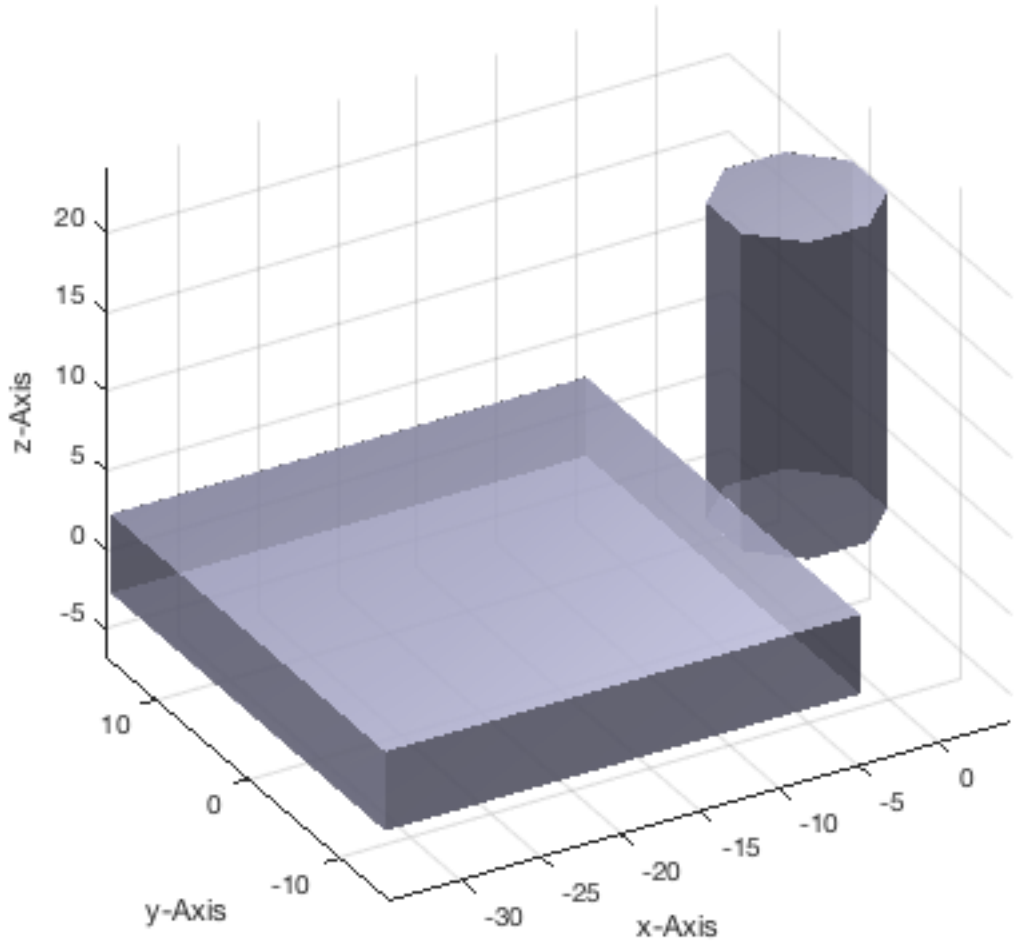
SGbehind positions a solid geometry 'A' behind of solid geometry 'B'

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



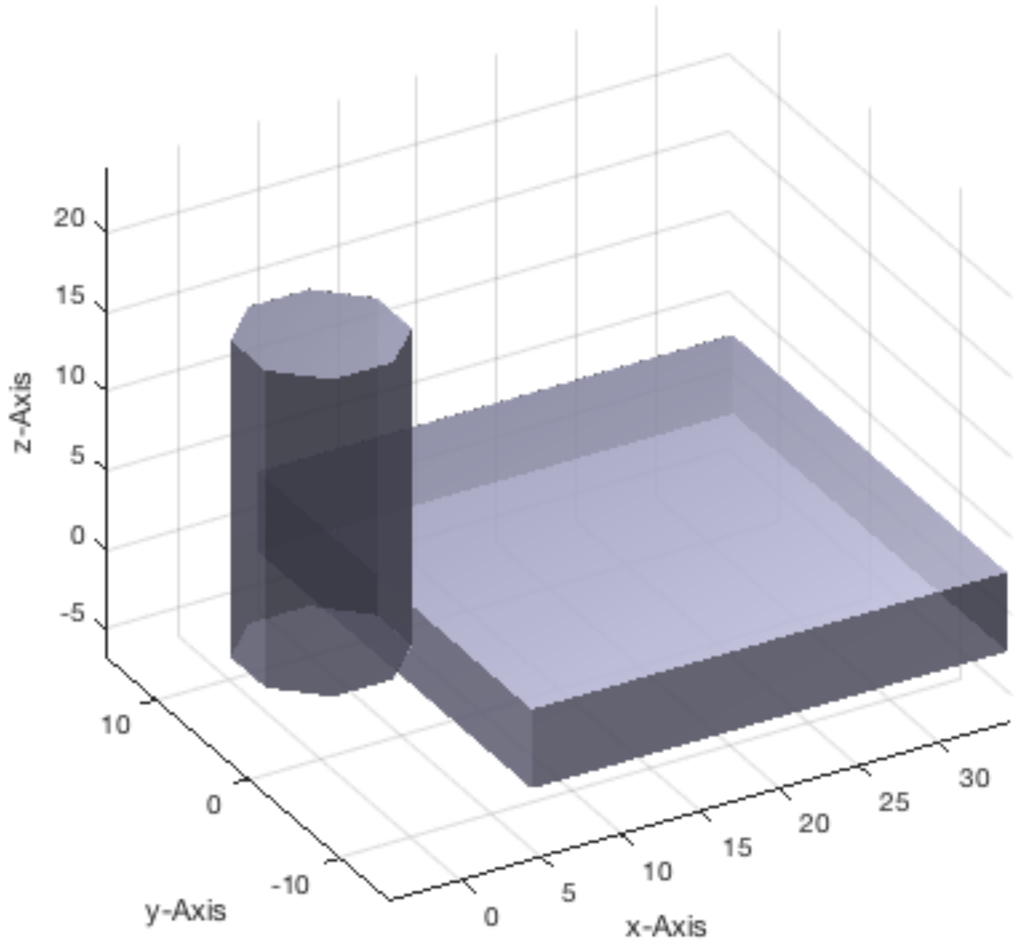
SGleft positions a solid geometry 'A' left of solid geometry 'B'

```
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);
```



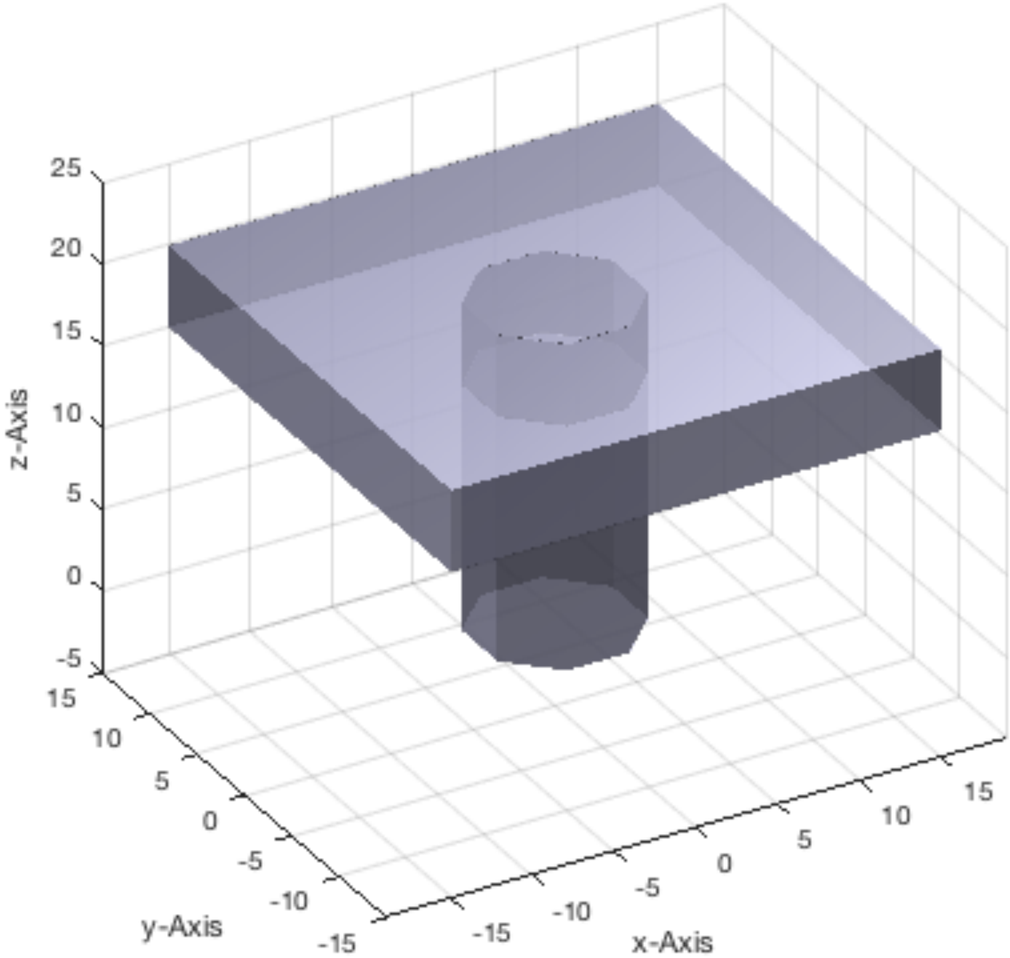
3. Relative spatial alignment of solid geometries (SG) using bounding boxes

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

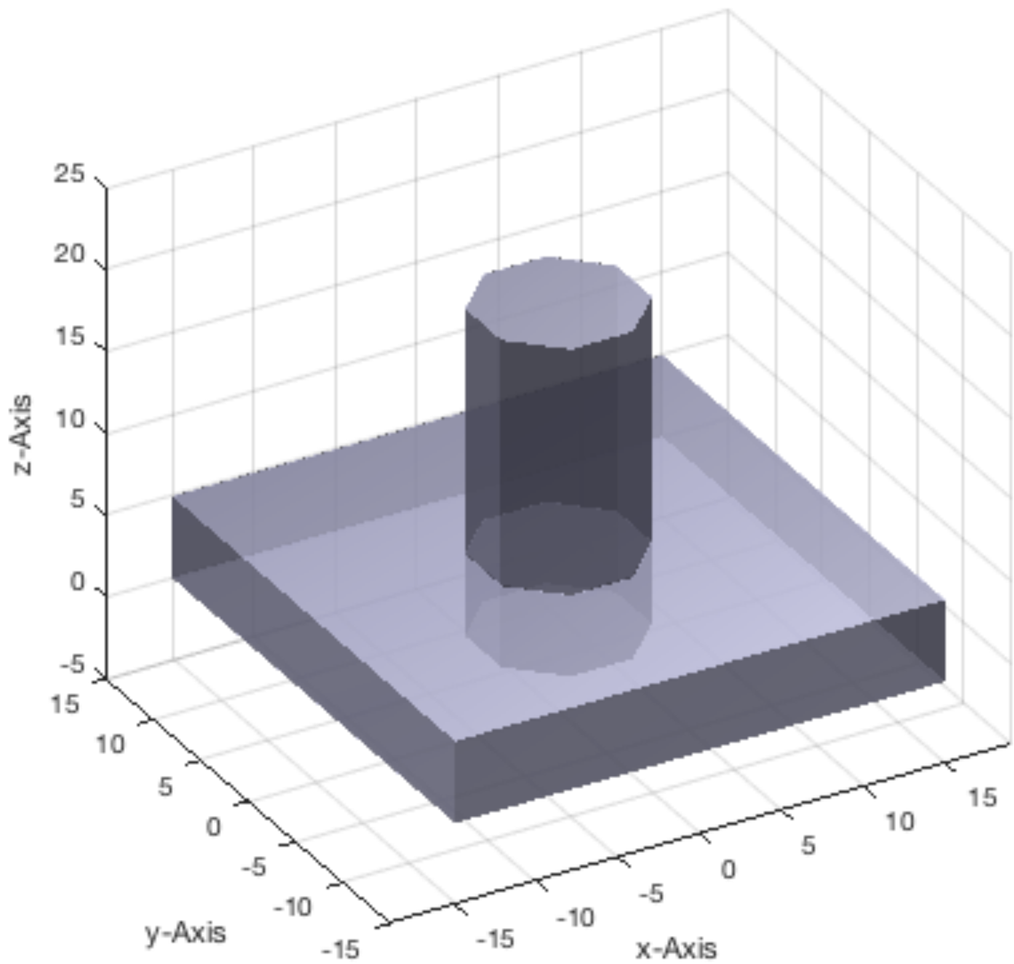
SGaligntop aligns the top of solid A with the top of solid B

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



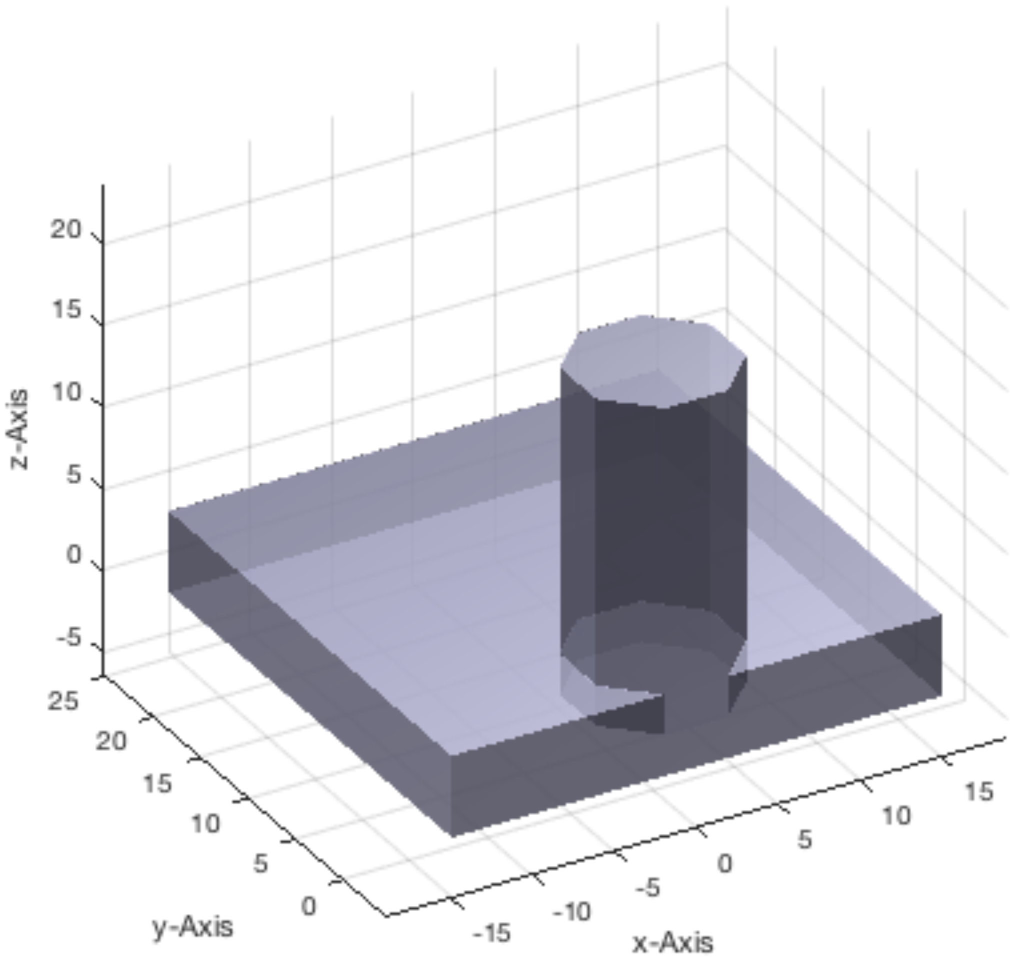
SGalignbottom aligns the bottom of solid A with the bottom of solid B

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



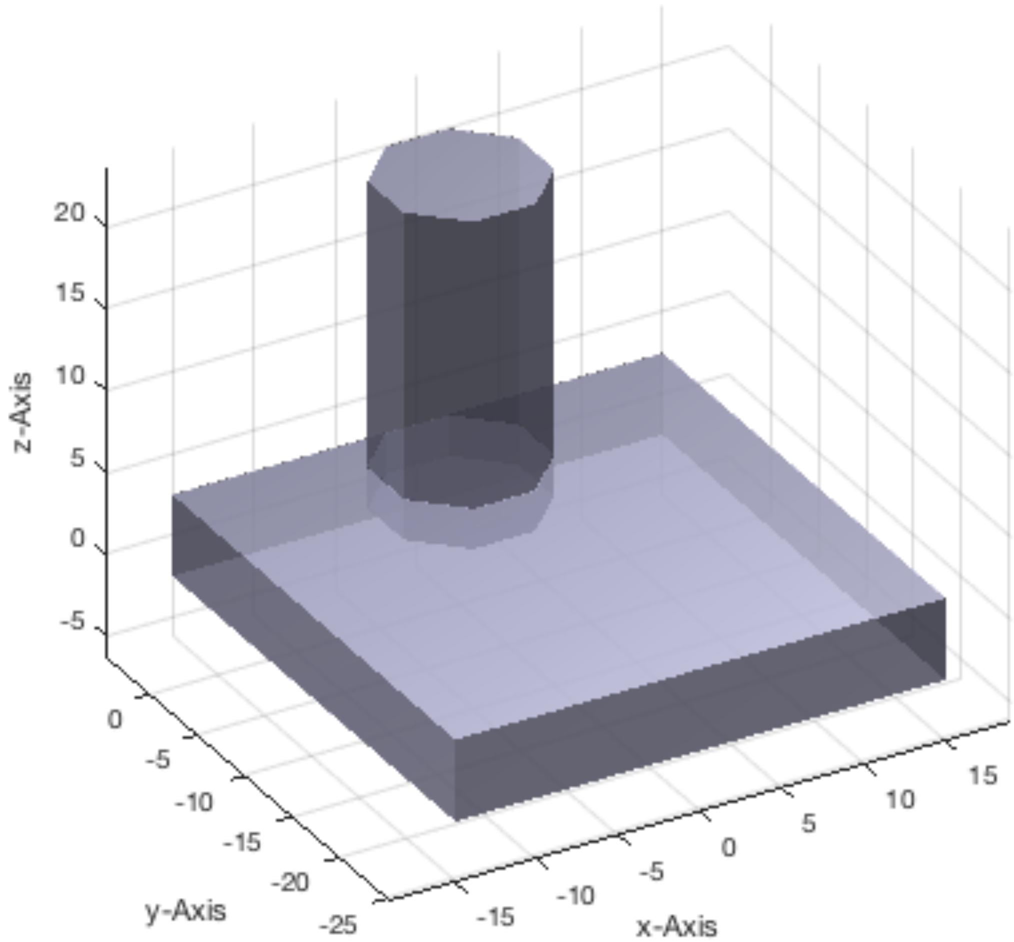
SGalignfront aligns the front of solid A with the front of solid B

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



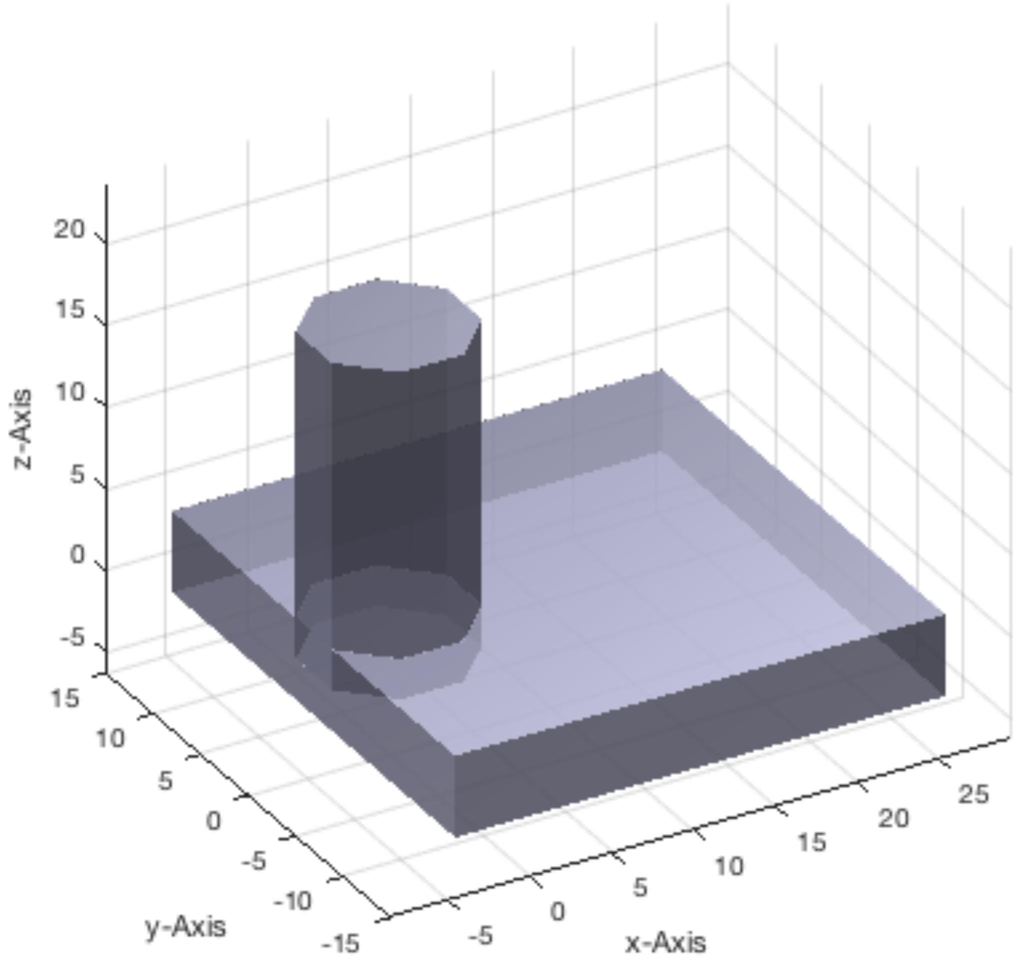
SGalignback aligns the back of solid A with the back of solid B

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



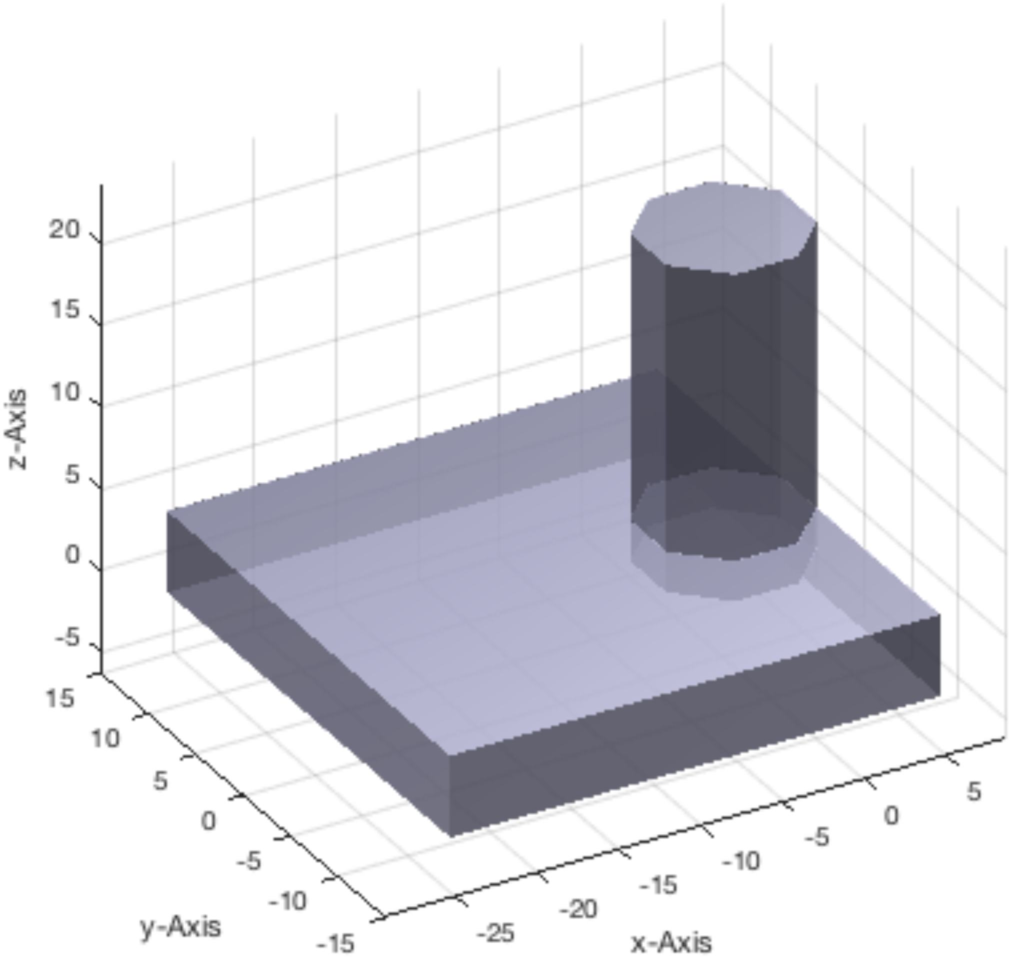
SGalignleft aligns the left side of solid A with the left side of solid B

```
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:14:13 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-6 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Import, and export of *STL-File* data, Conversion of text strings into solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation, positioning, alignment and merging of solid geometry sets

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

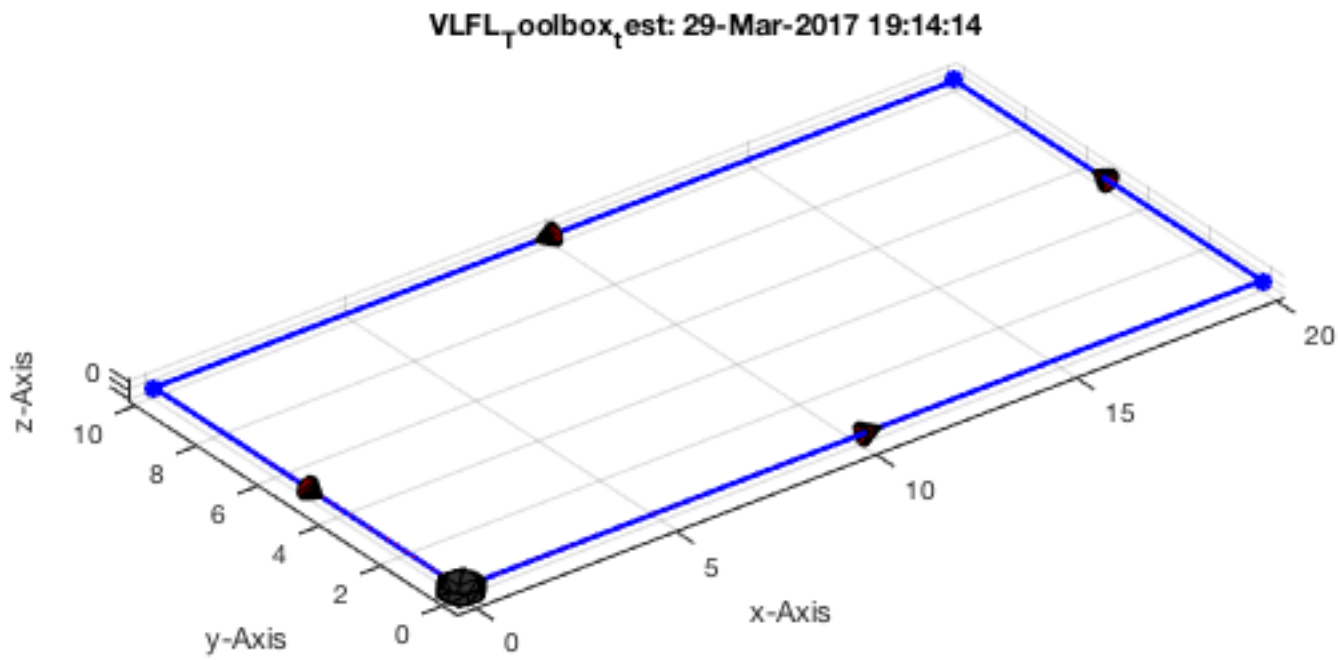
3. Rotation of closed polygon lists (CPL)

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

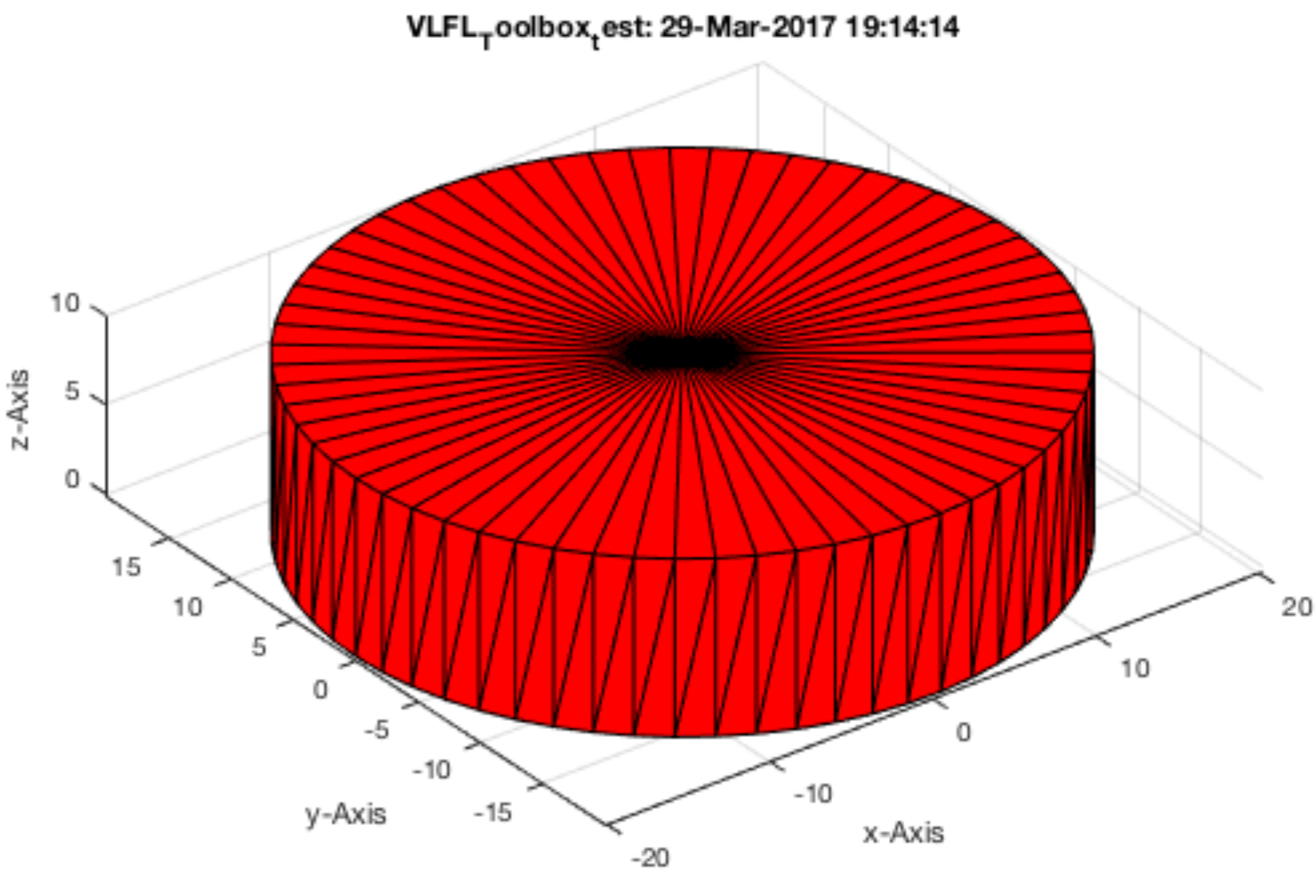
```
close all;
CPL=[0 0; 20 0; 20 10; 0 10];        % Create a simple rectangle (ccw)
CPLplot(CPL);                         % plot the rectangle
PLELofCPL(CPL);                       % show edges and directions
```



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

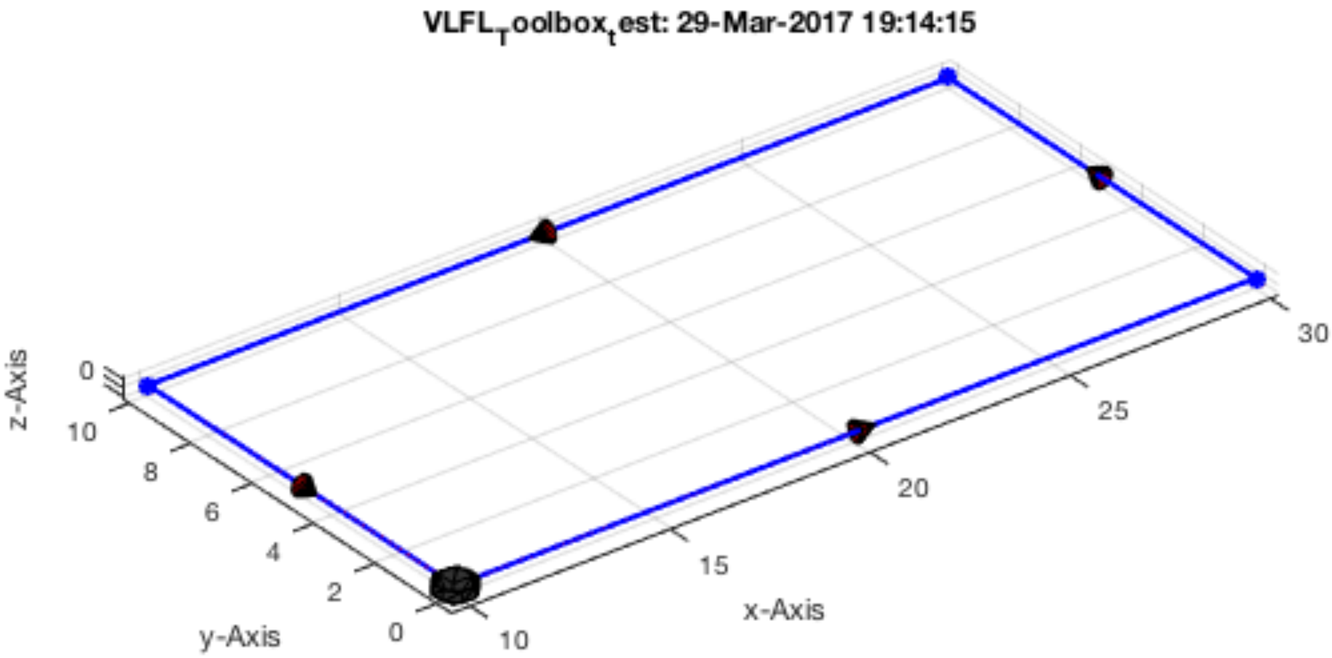
```
SG=SGofCPLrot(CPL);                   % Solid contour rotation
SGplot(SG);                           % show the solid
```

```
Removed 252(504) facets
```



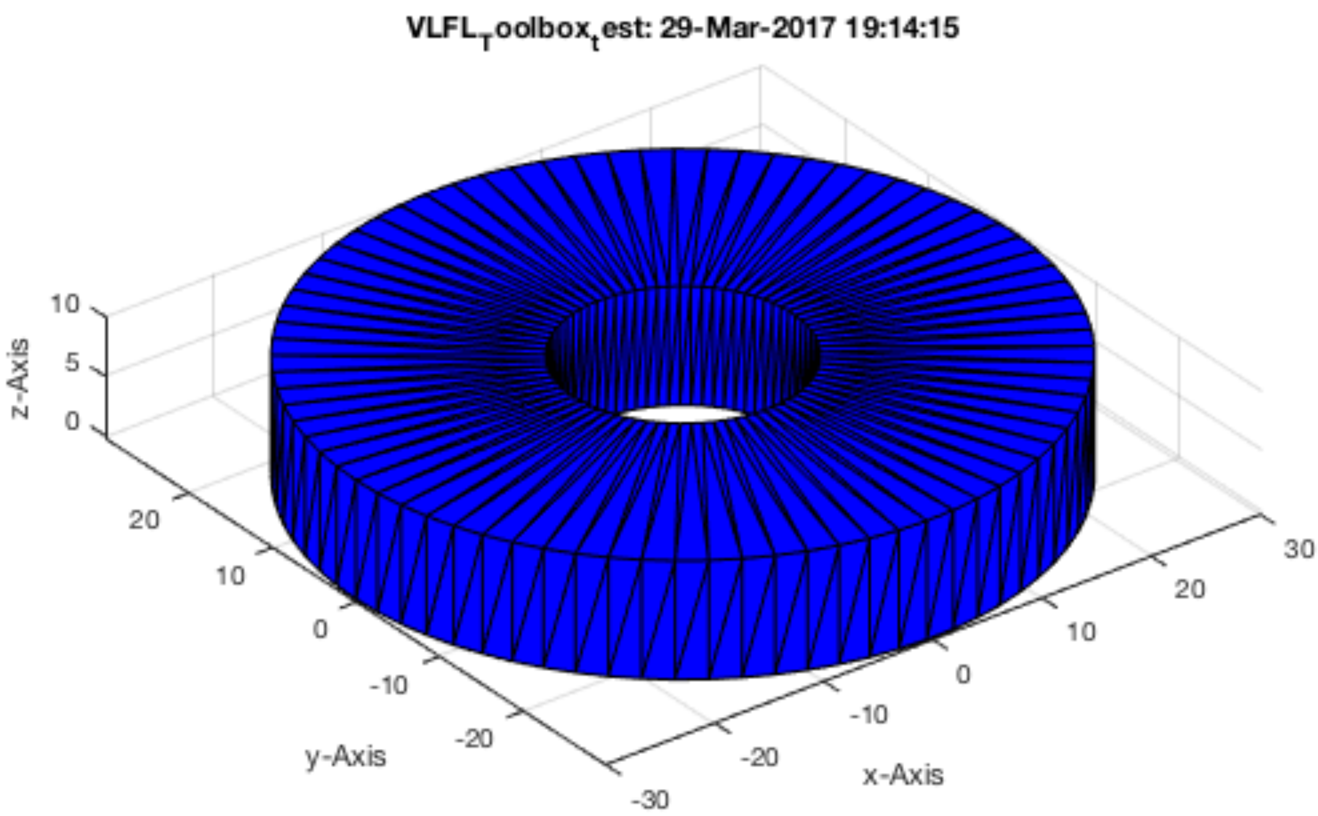
Exercise: Create a simple point list with distance to the y-axis

```
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

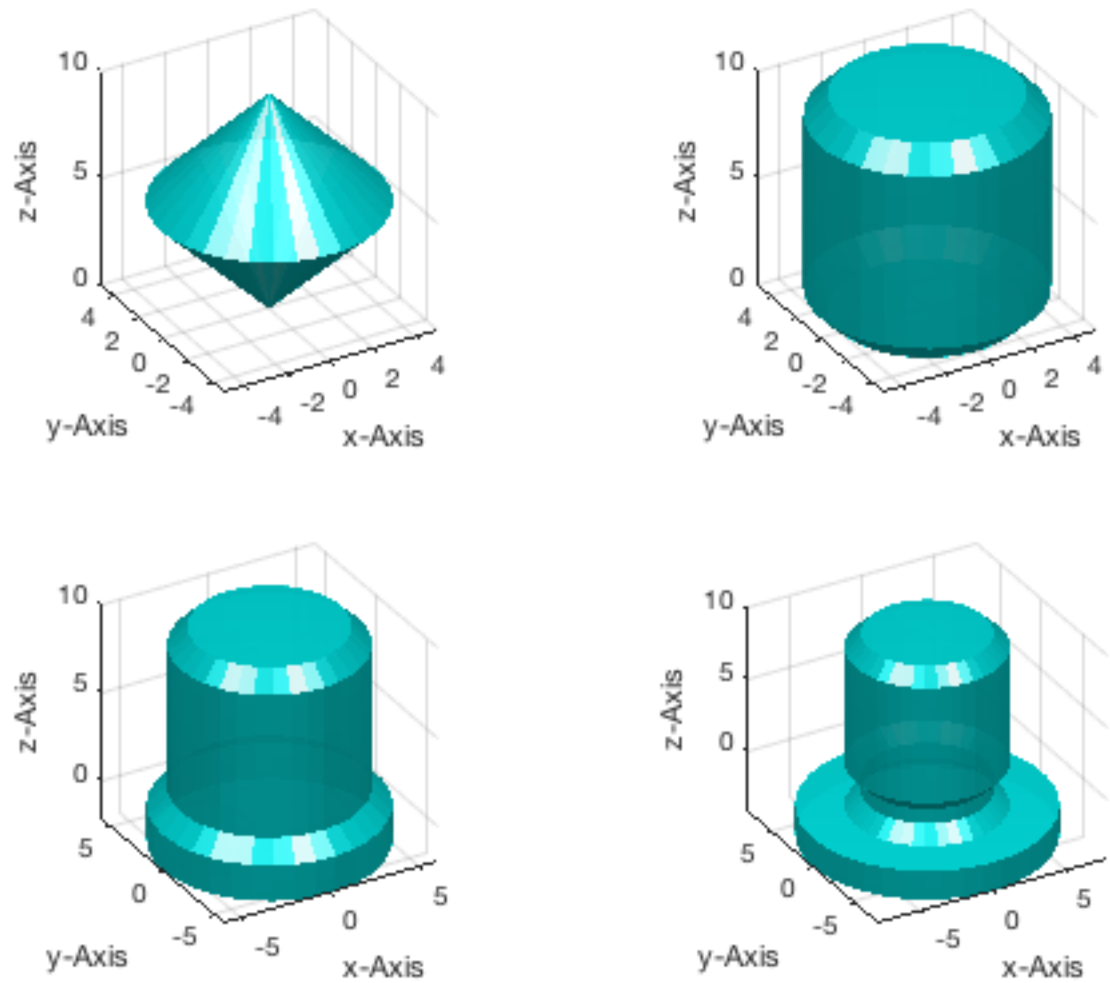
```
SG=SGofCPLrot(CPL);                % Solid contour rotation
SGplot(SG,'b');
```



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);
```

```
Removed 192(256) facets
Removed 256(512) facets
Removed 350(700) facets
Removed 560(1200) facets
```

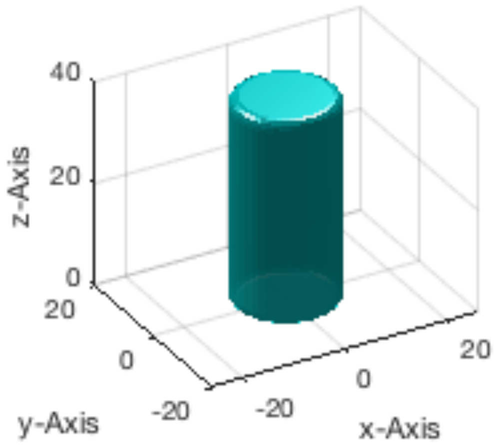


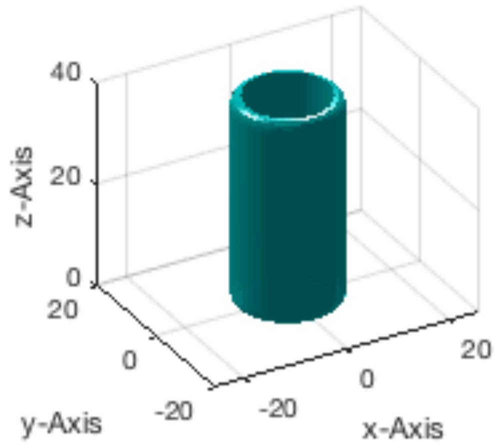
The warnings 'Removed n(m) facets' can be ignored. These warning appear if a part of the contour touches or crosses the x=0 line (y-axis).

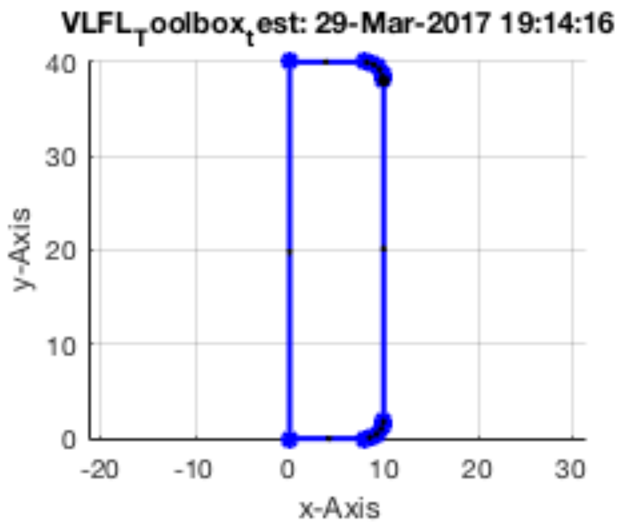
Exercise: Creating a bold and a sleeve

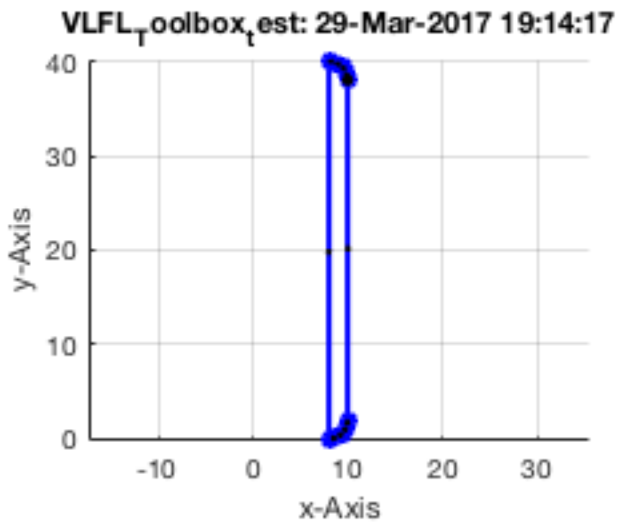
```
closeall;
r=2; H=40; R=10;

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);
```

```
Removed 900(1800) facets
```







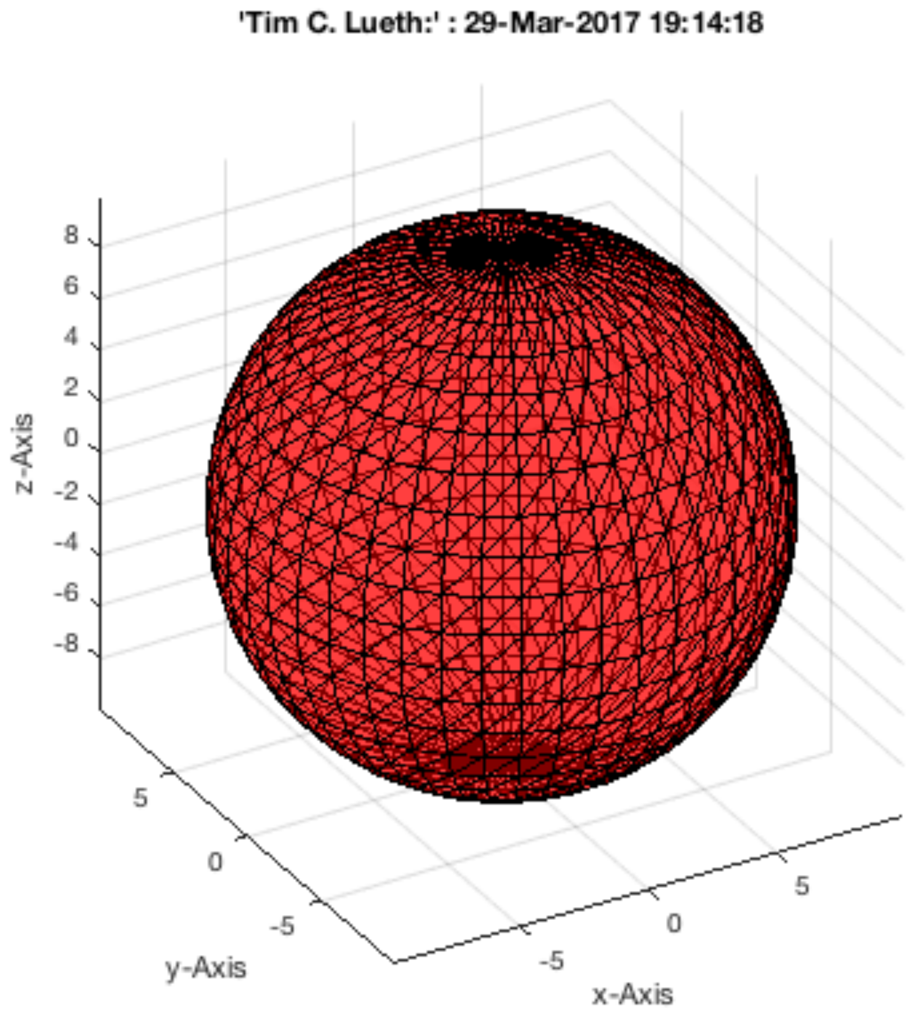


4. Creating spheres by rotating half-circles

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);
```

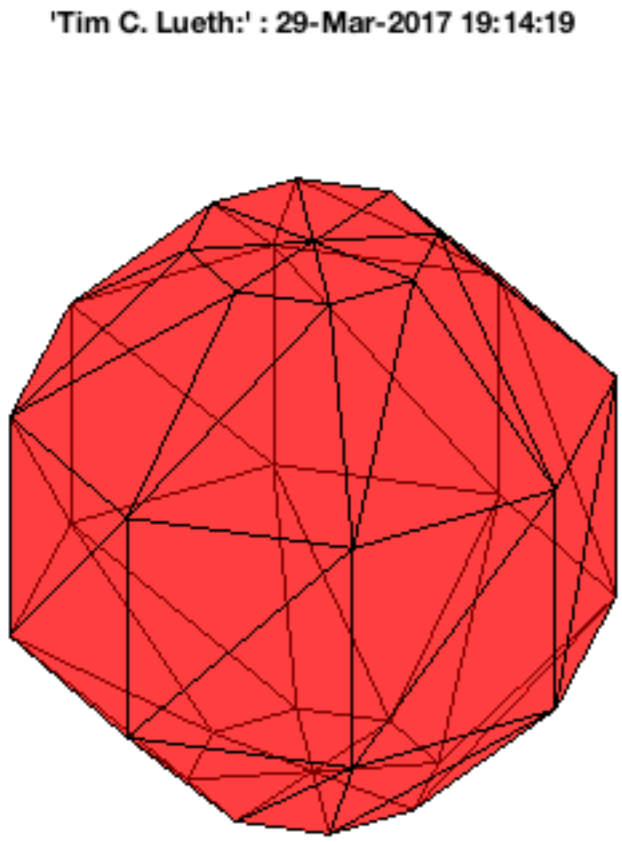
```
Removed 2160(4140) facets
```



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);
```

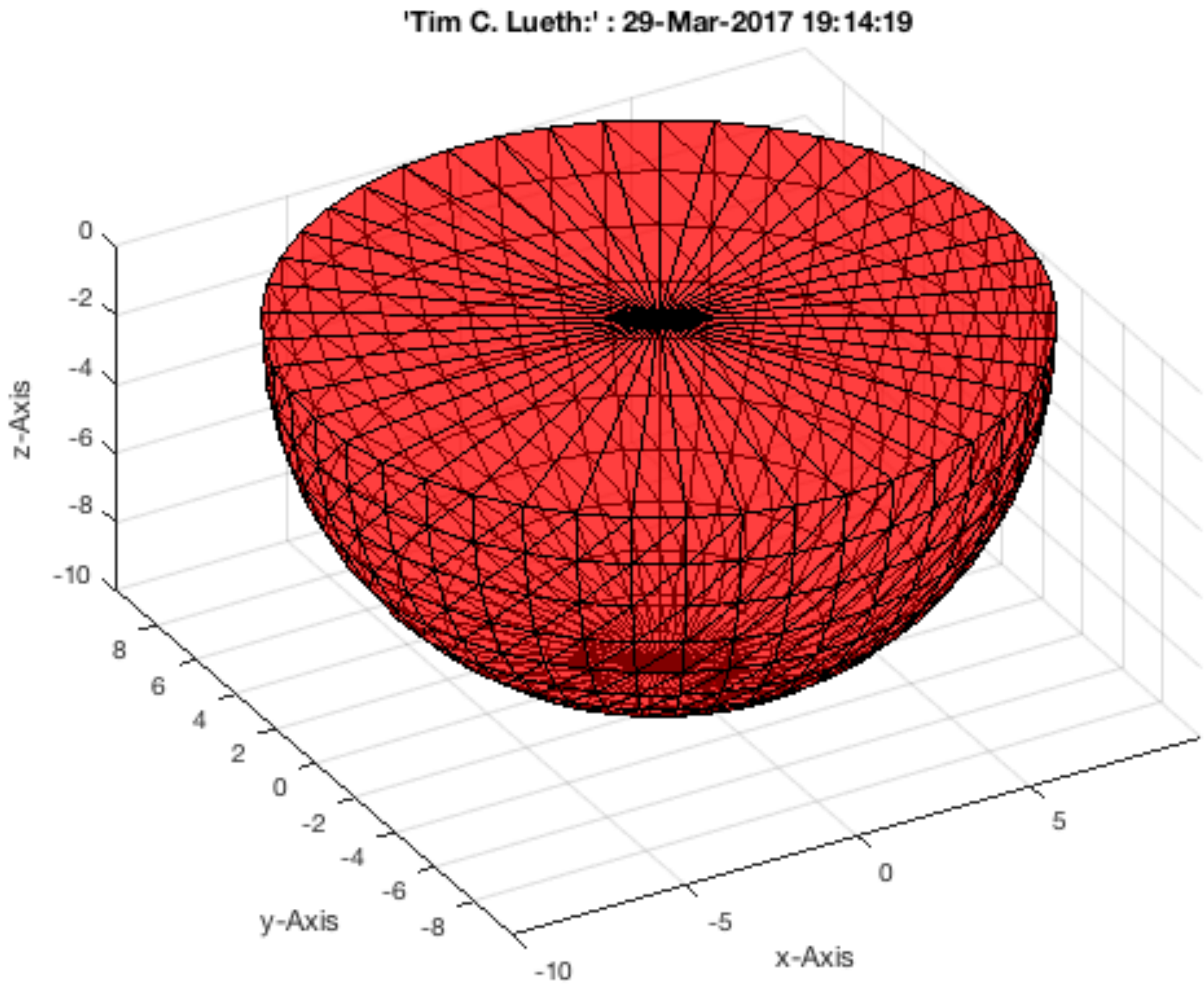
```
Removed 64(128) facets
```



Exercise: Creating a half sphere

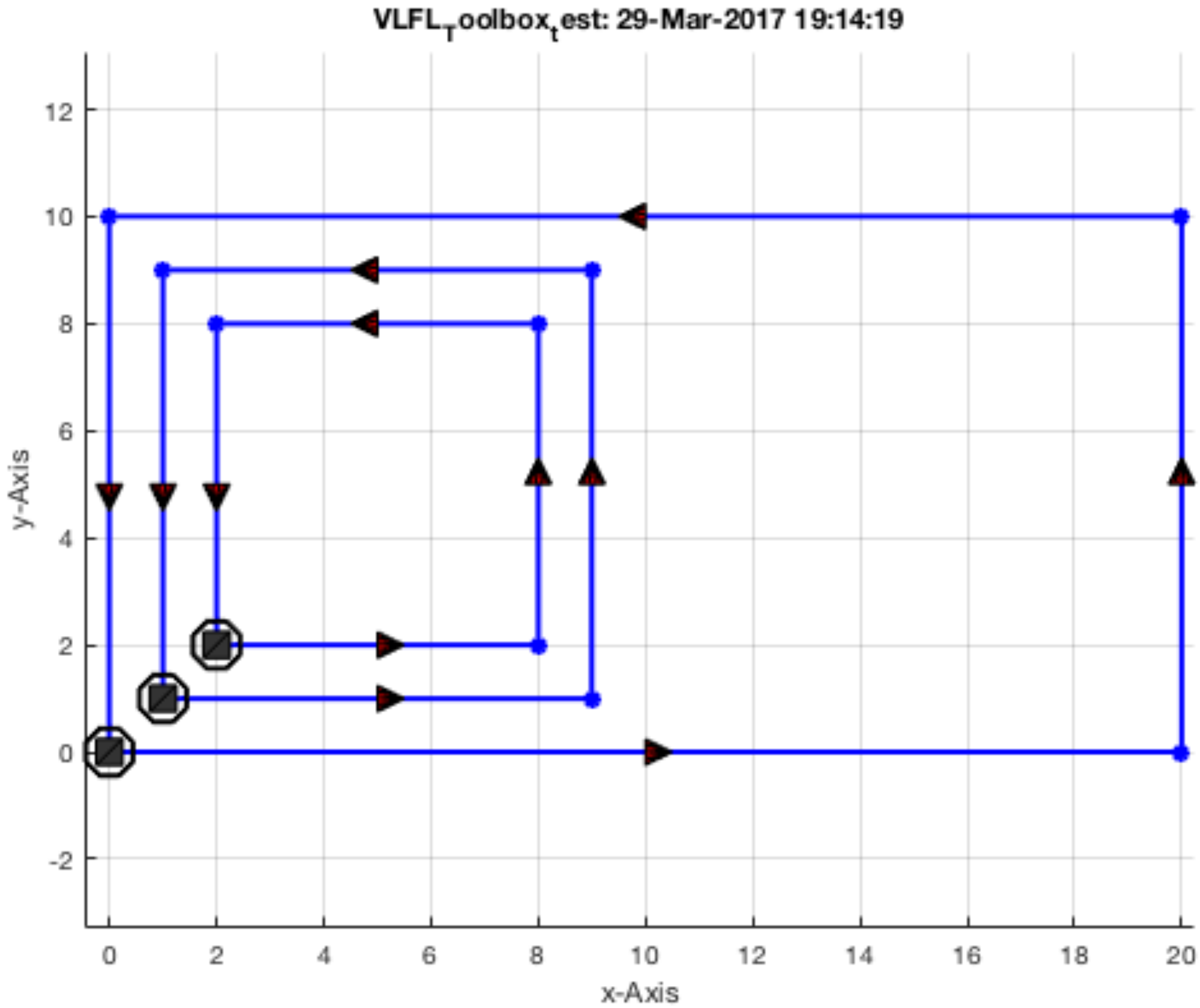
```
close all;
PL=[PLcircseg(10,[],-pi/2,0); 0 0];

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

```
Removed 1080(2070) facets
```

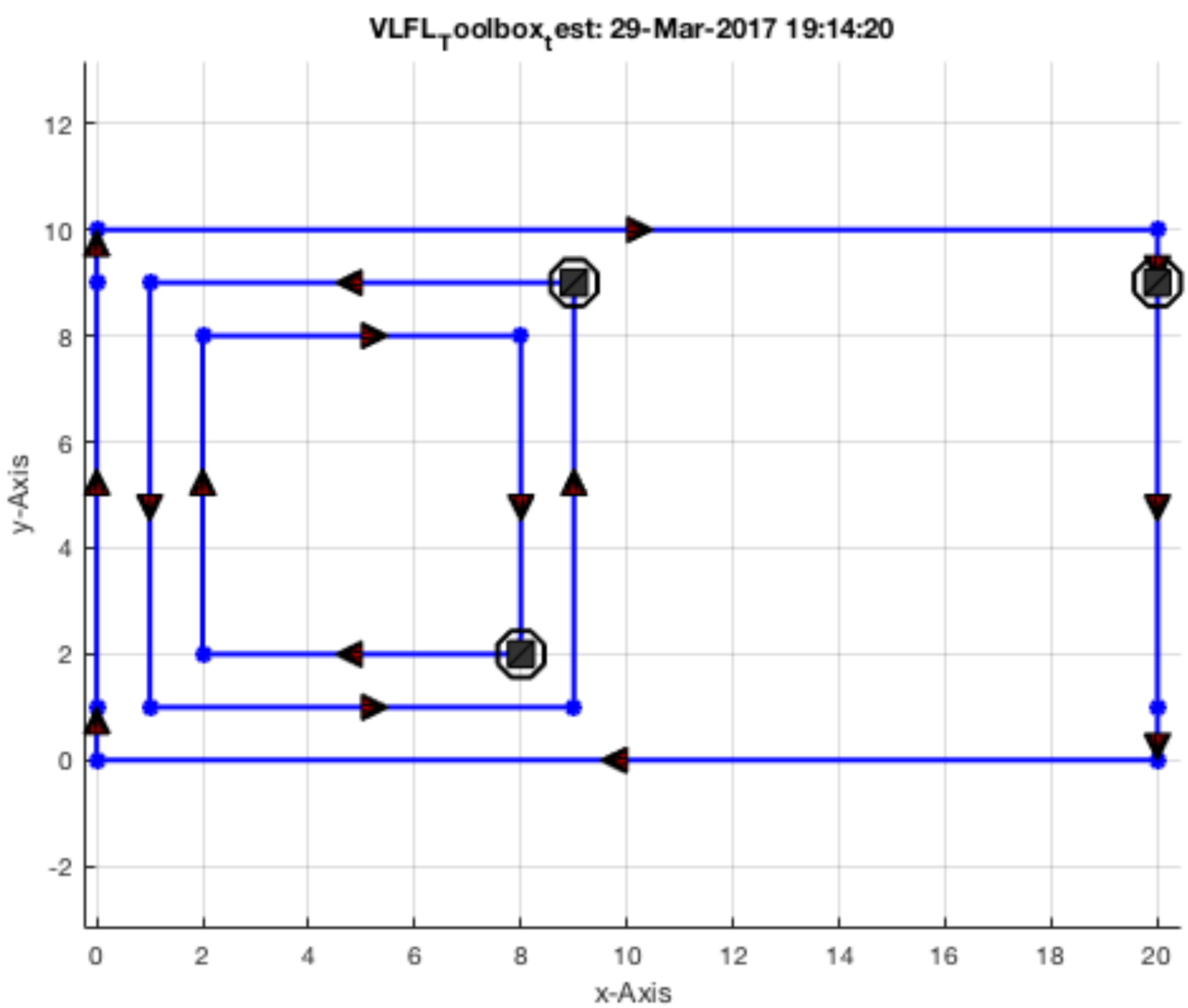


5. Creating embedded contours

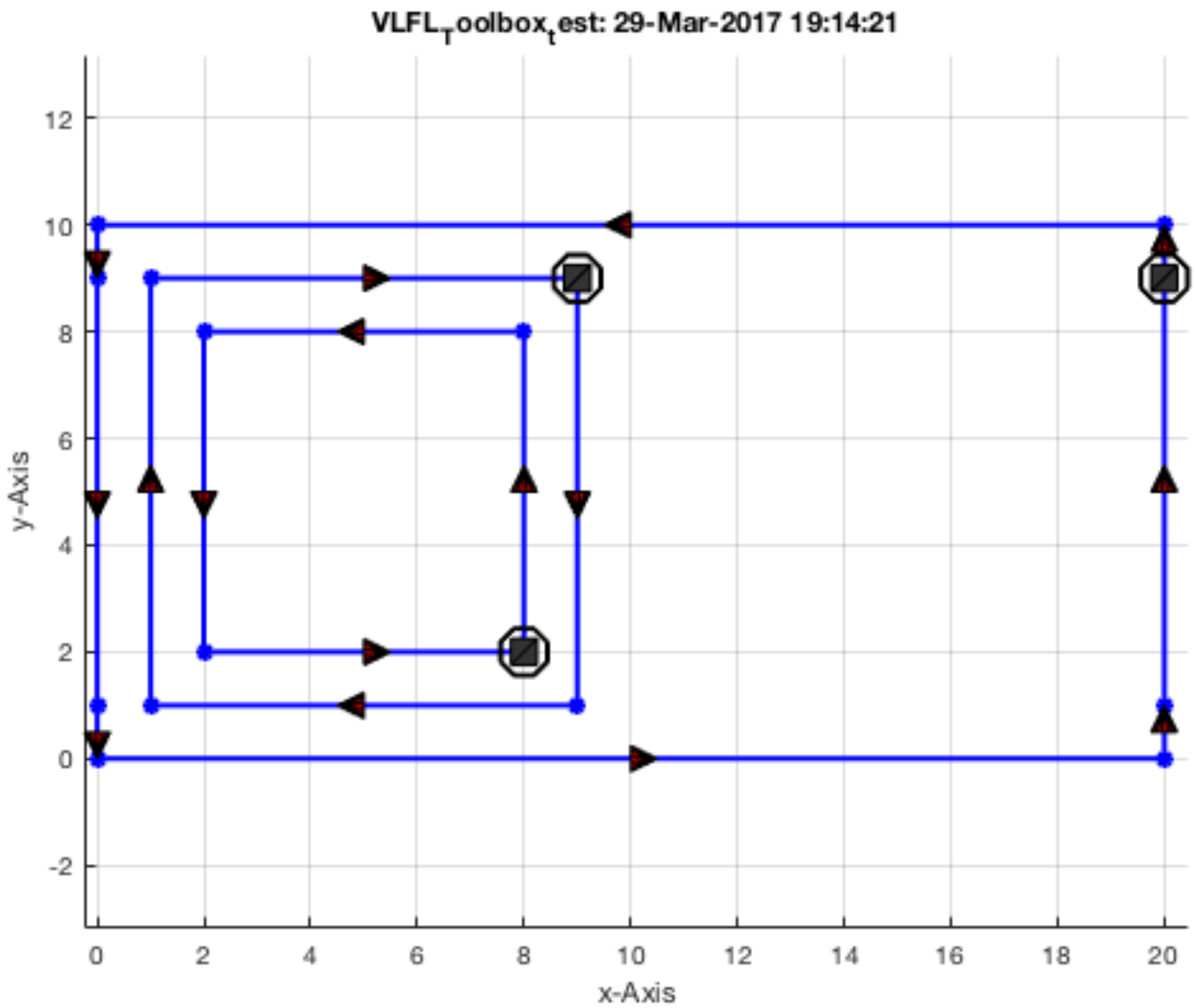
```
CPL=[0 0; 20 0; 20 10; 0 10; NaN NaN; 1 1; 9 1; 9 9; 1 9; NaN NaN; 2 2; 8 2; 8 8; 2 8 ];

close all;PLELofCPL(CPL);
```




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



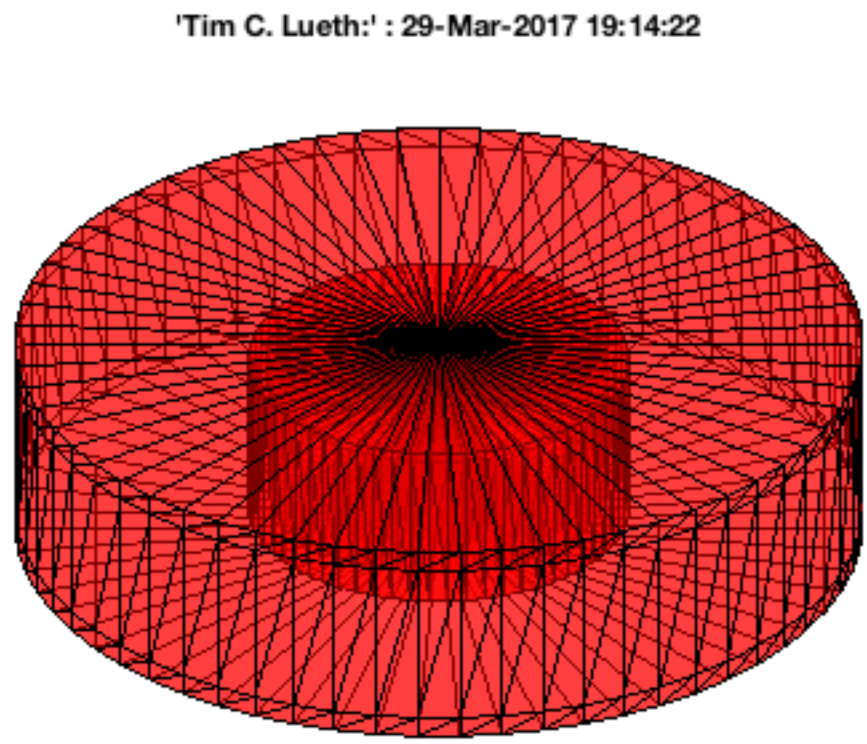
```
CPL=flip(CPL);
close all; PLELofCPL(CPL);
```



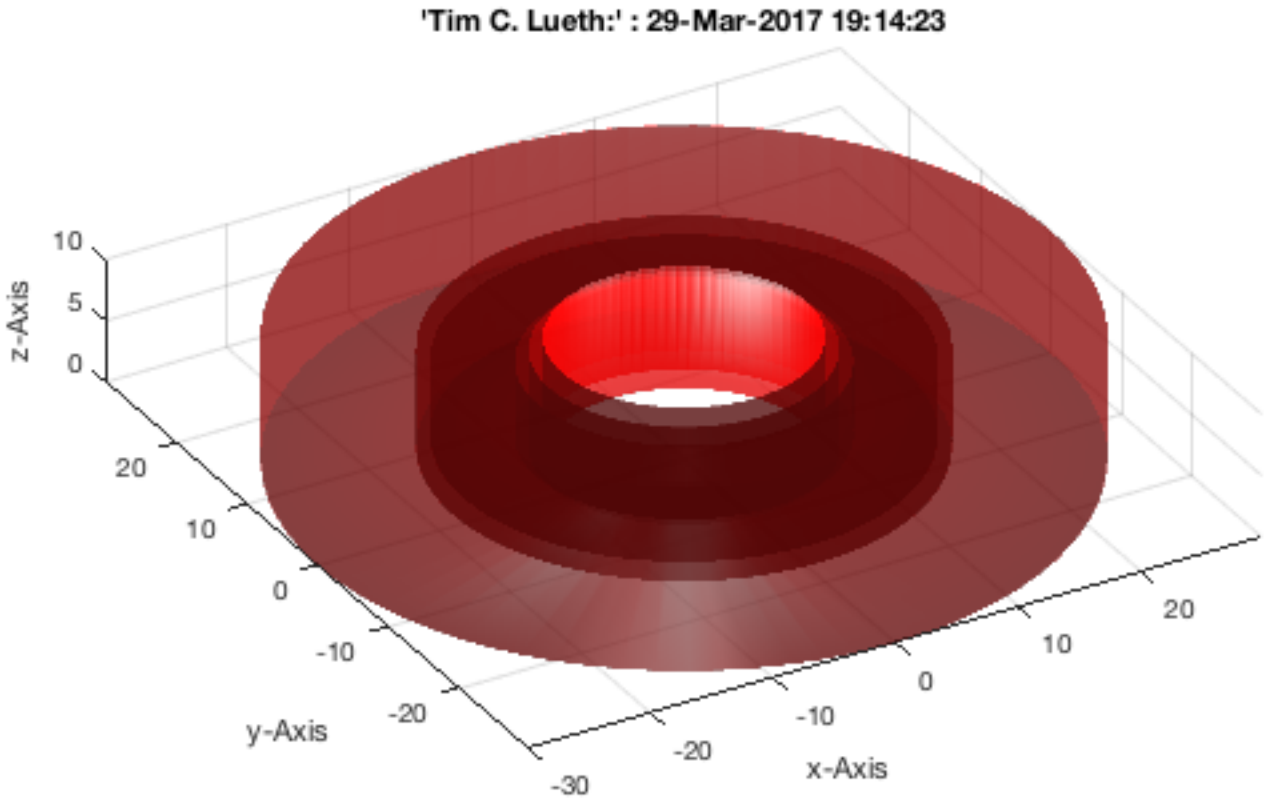
6. Rotate Contours around the z-axis

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

```
Removed 504(1008) facets
```




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



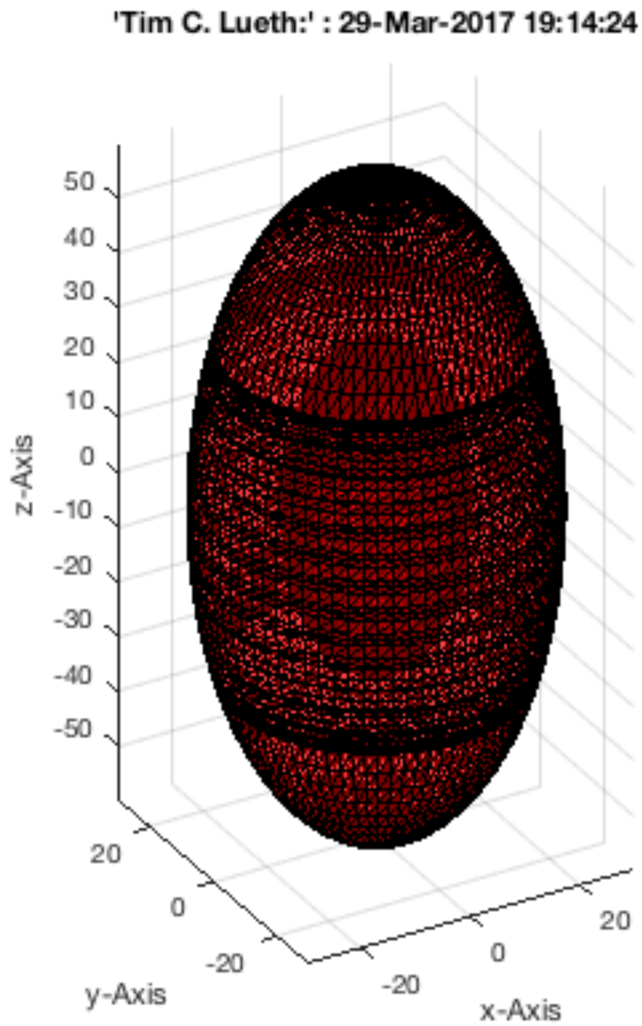

```
VLFLfigure; view(-30,30); grid on;
CPL=PLcircle(30); CPL(:,2)=CPL(:,2)*2;
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (1,0.5);
```

```
Removed 6160(12012) facets
```

'Tim C. Lueth:' : 29-Mar-2017 19:14:23

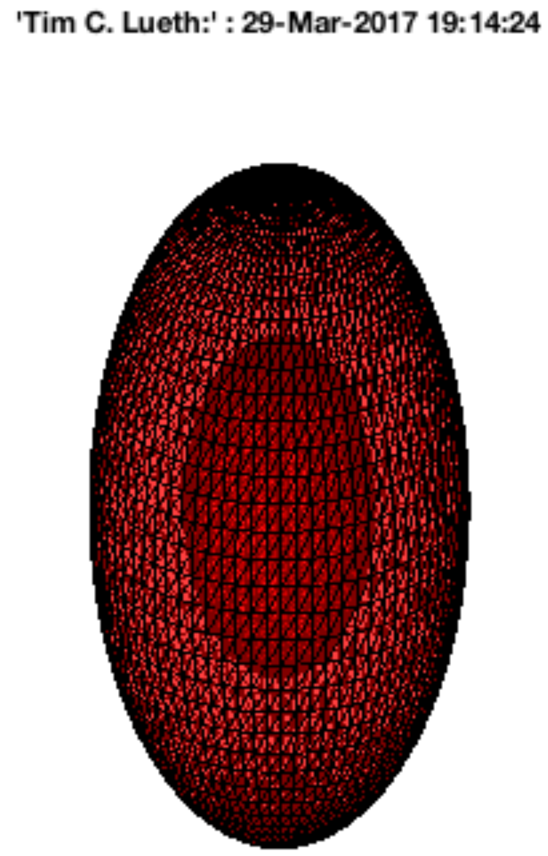
```
VLFLfigure; view(-30,30); grid on;
CPL=PLcircle(30); CPL(:,2)=CPL(:,2)*2;
CPL=[CPL;NaN NaN;CPL*0.5];
SG=SGofCPLrot(CPL);
SGplot(SG); VLFLplotlight (0,0.5);
```

```
Removed 4620(24332) facets
```




```
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);
```

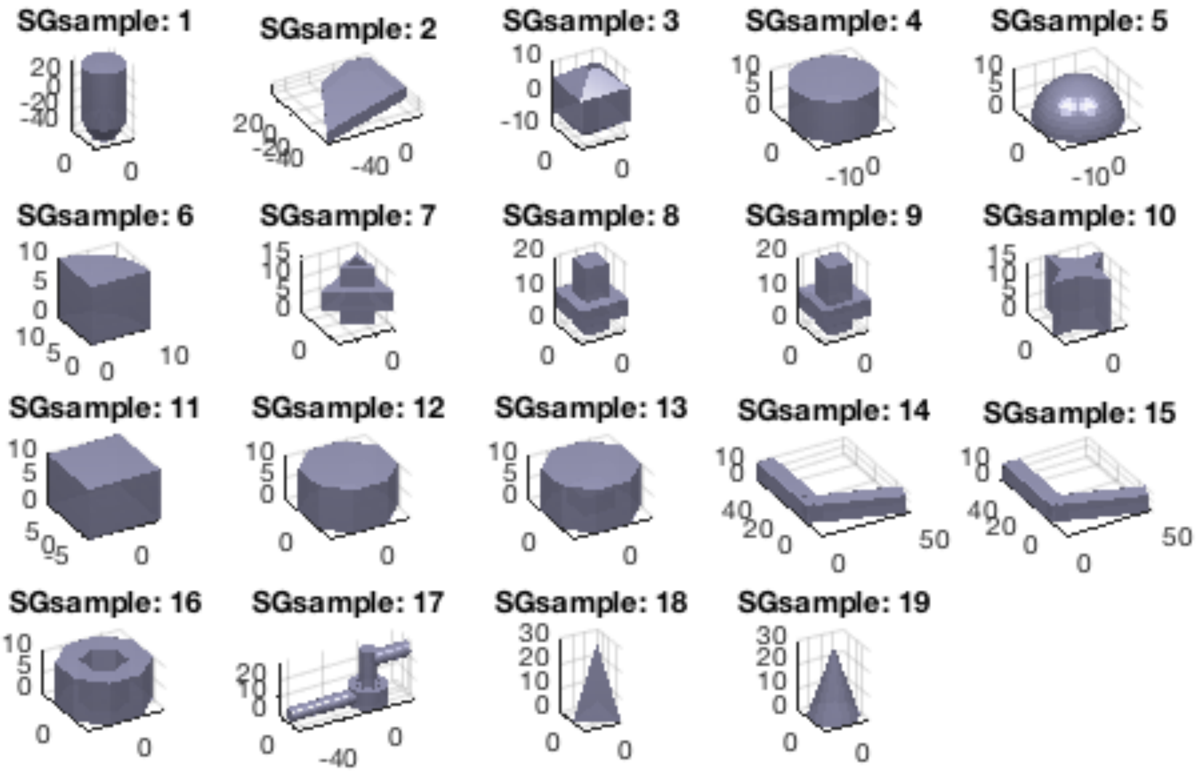
```
Removed 6160(12012) facets
Removed 4400(8580) facets
```



7. Samples of 3D Design

```
close all
SGsample;
```

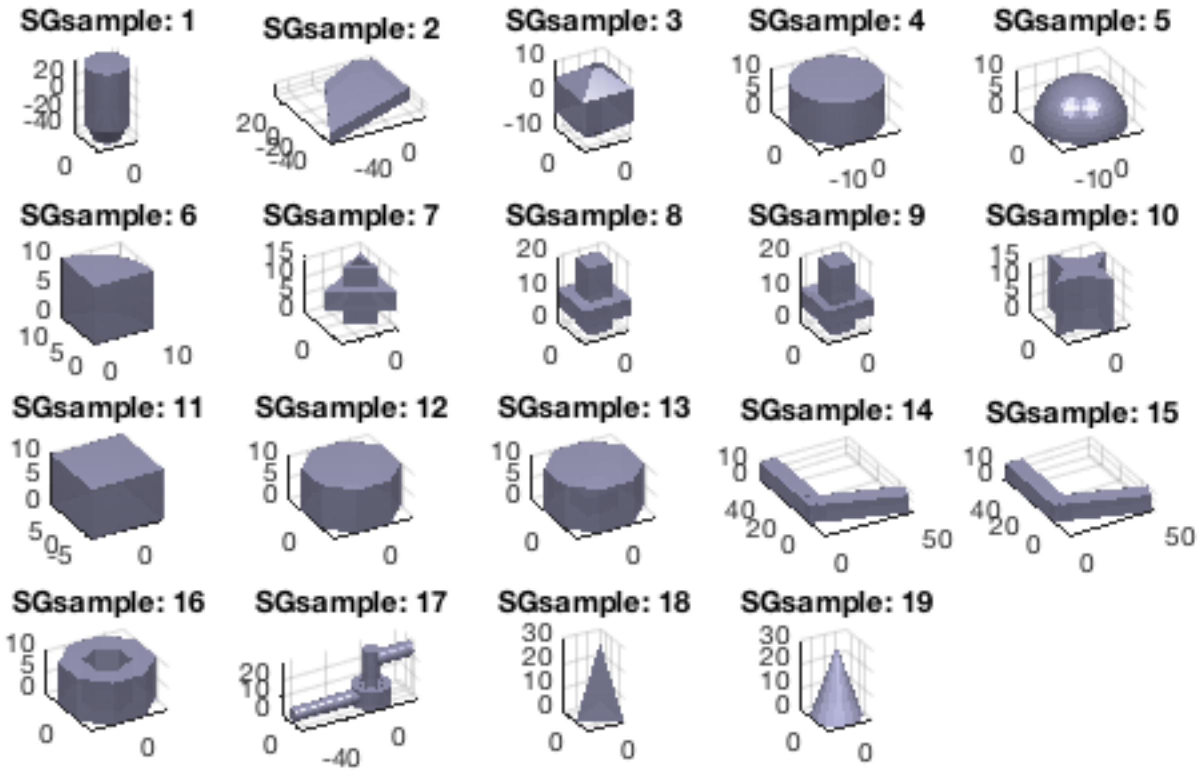
```
Removed 414(1104) facets
Removed 1080(2070) facets
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:14:30 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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*

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

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-7 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Import, and export of *STL-File* data, Conversion of text strings into solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- 2.5D rotation of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- Spatial transformation, positioning, alignment and merging of solid geometry sets

2. Create a sample solid for this exercise

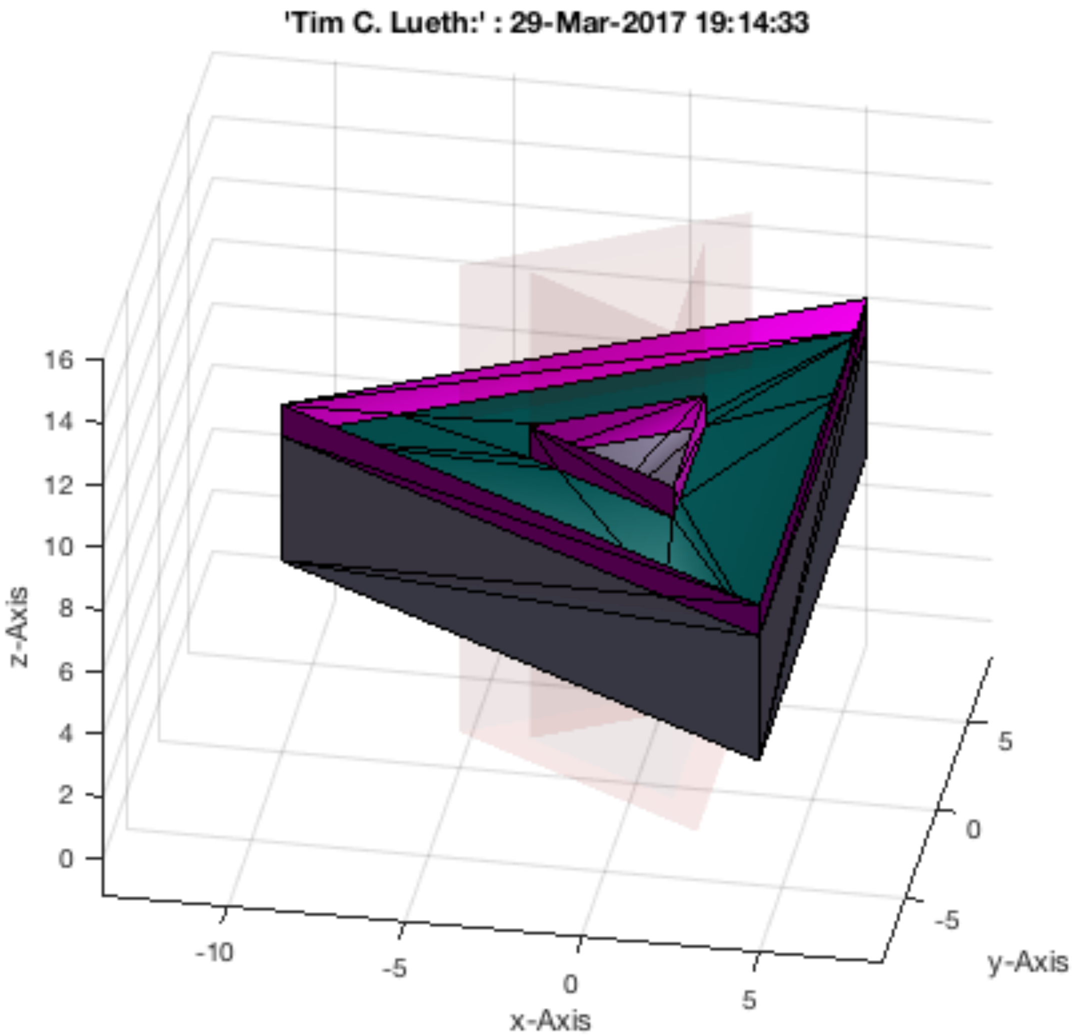
Using the function SGsample it is possible to create samples for an experiment, to see all of them or to select one.

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

3. Analyze a slice plane through a solid geoemtry

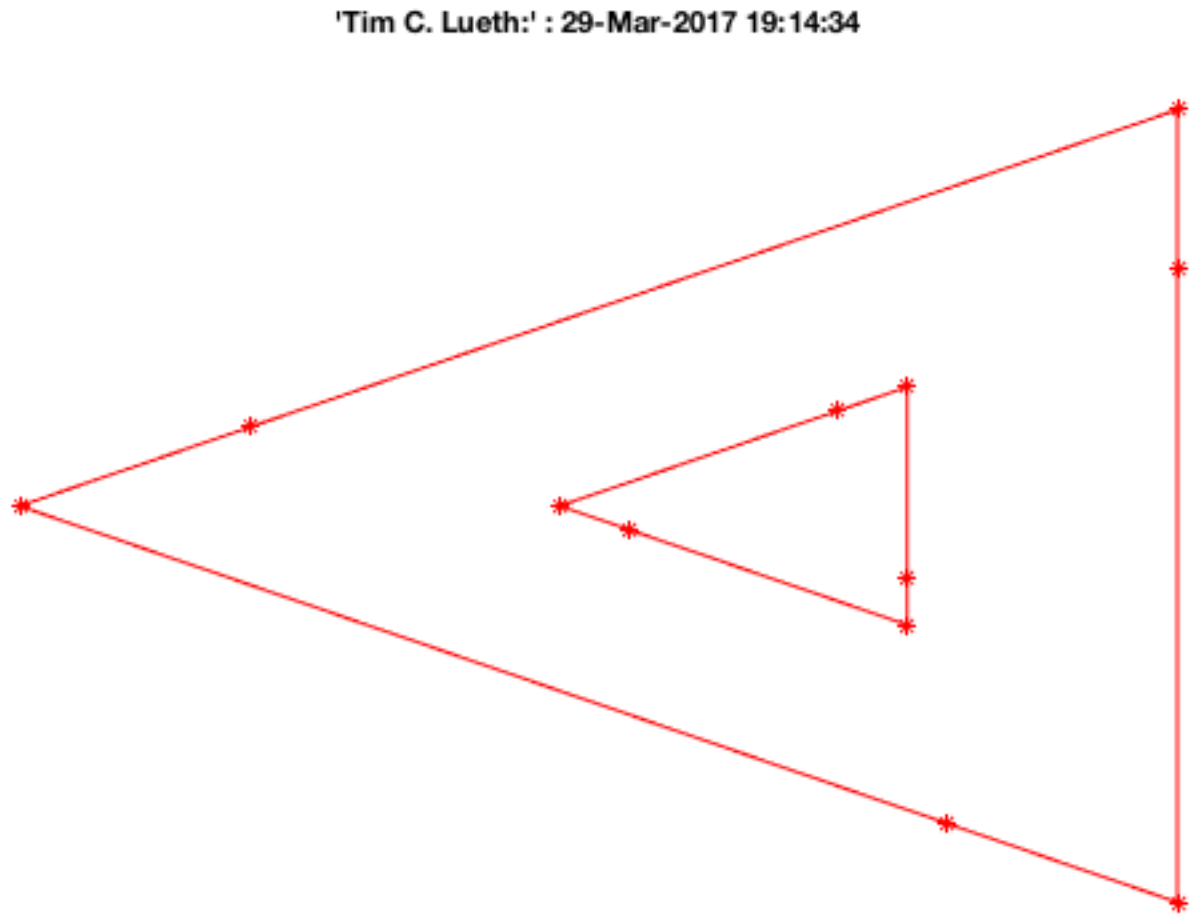
Slicing at a specified z-coordinates is a more complex procedure than expected if several solids are processed that can penetrate each other. By slicing a single solid, the crossed facets/triangles are separated into 2 upper and lower parts that will lead to 2 lower and 1 upper facets or 1 lower and 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);
```



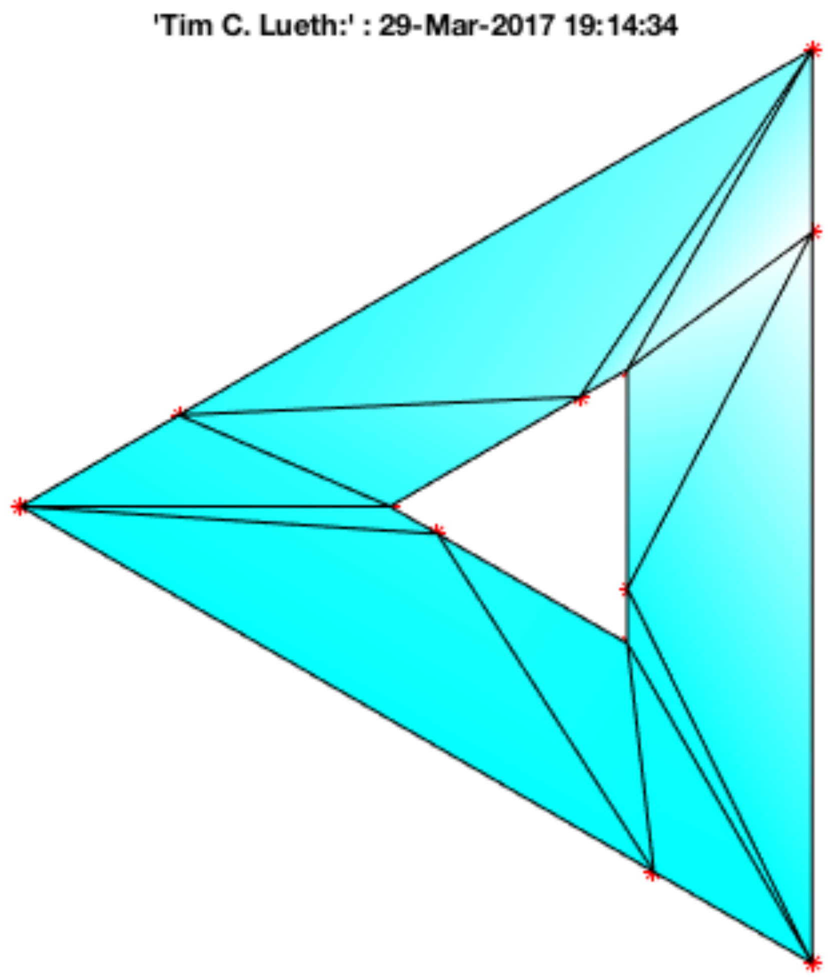
It is also possible just to show the cutting edges of the cutting contour

```
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.

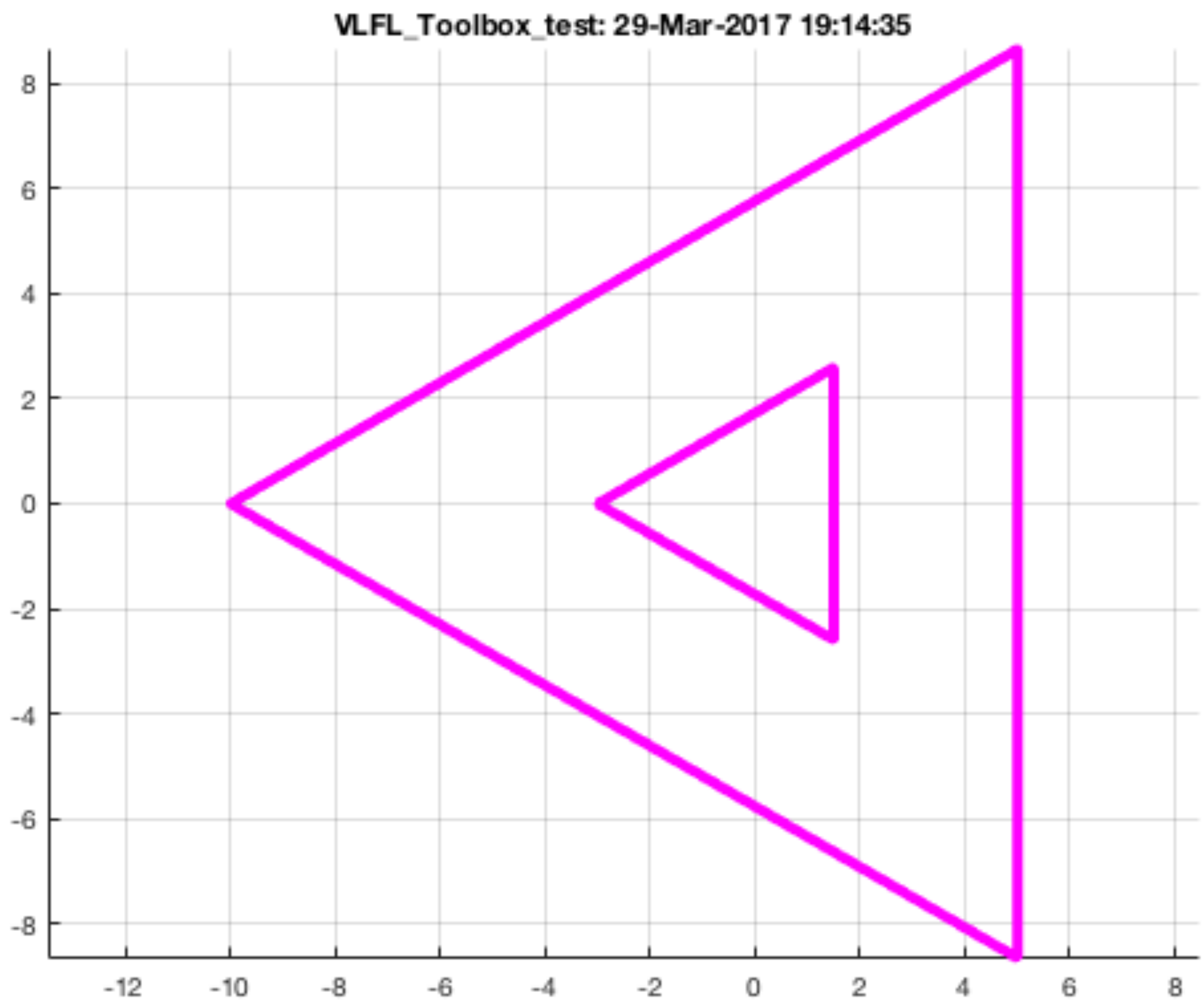
```
in=isInterior(TR2);
VLFLplots(TR2.Points, TR2.ConnectivityList(in,:),'c');
```



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

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

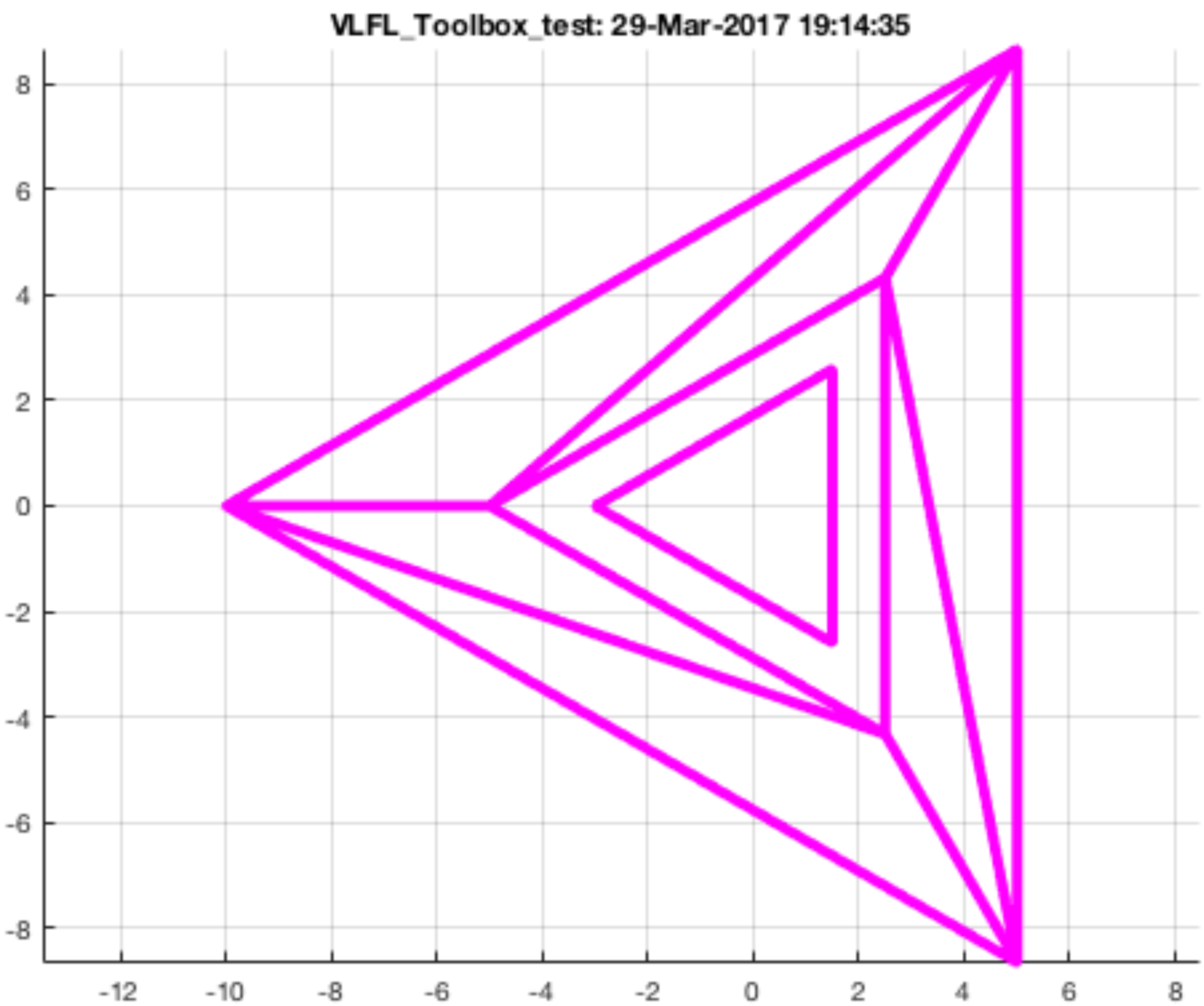
```
warn =

  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

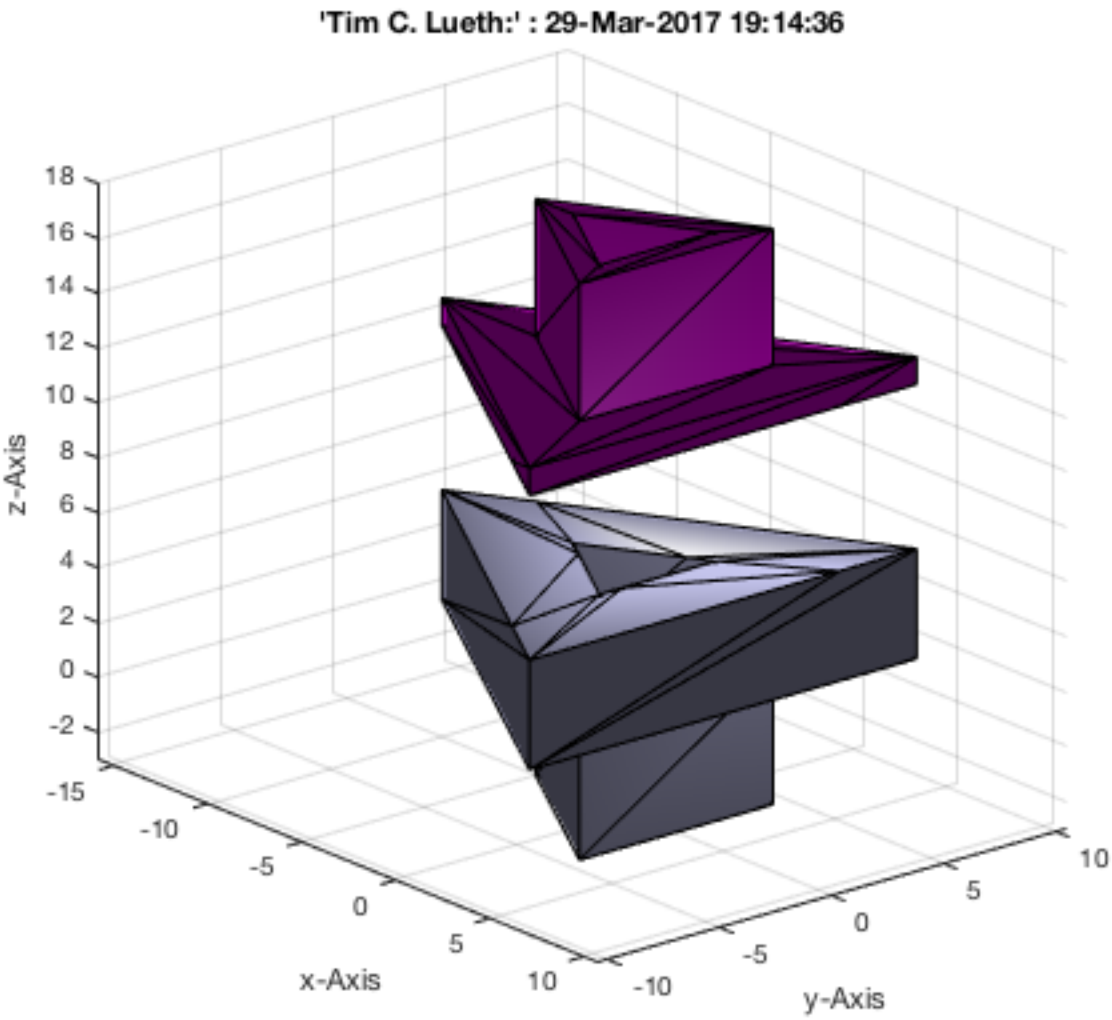
By using the output of SGslicer it is possible to create an upper and lower part of an object or even by two cutting plane to cut a part out of a larger obect. This is done by the function **SGcut**.

```
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]
```

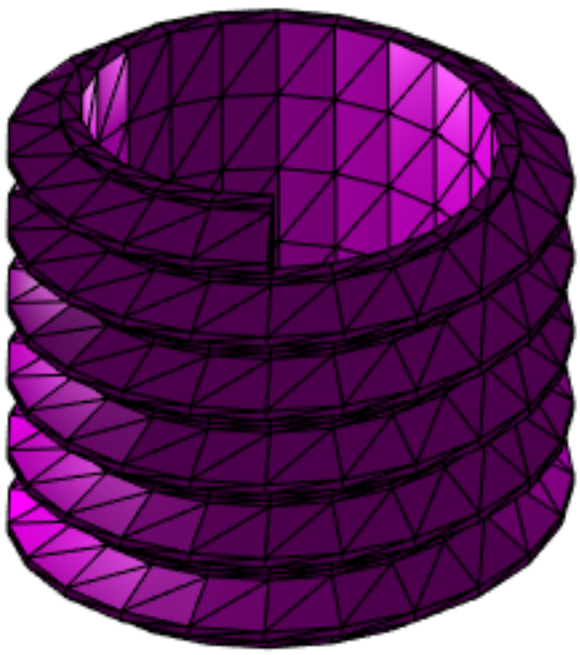


5. Cutting as useful tool for the ending of complex shaped geoemtries

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:

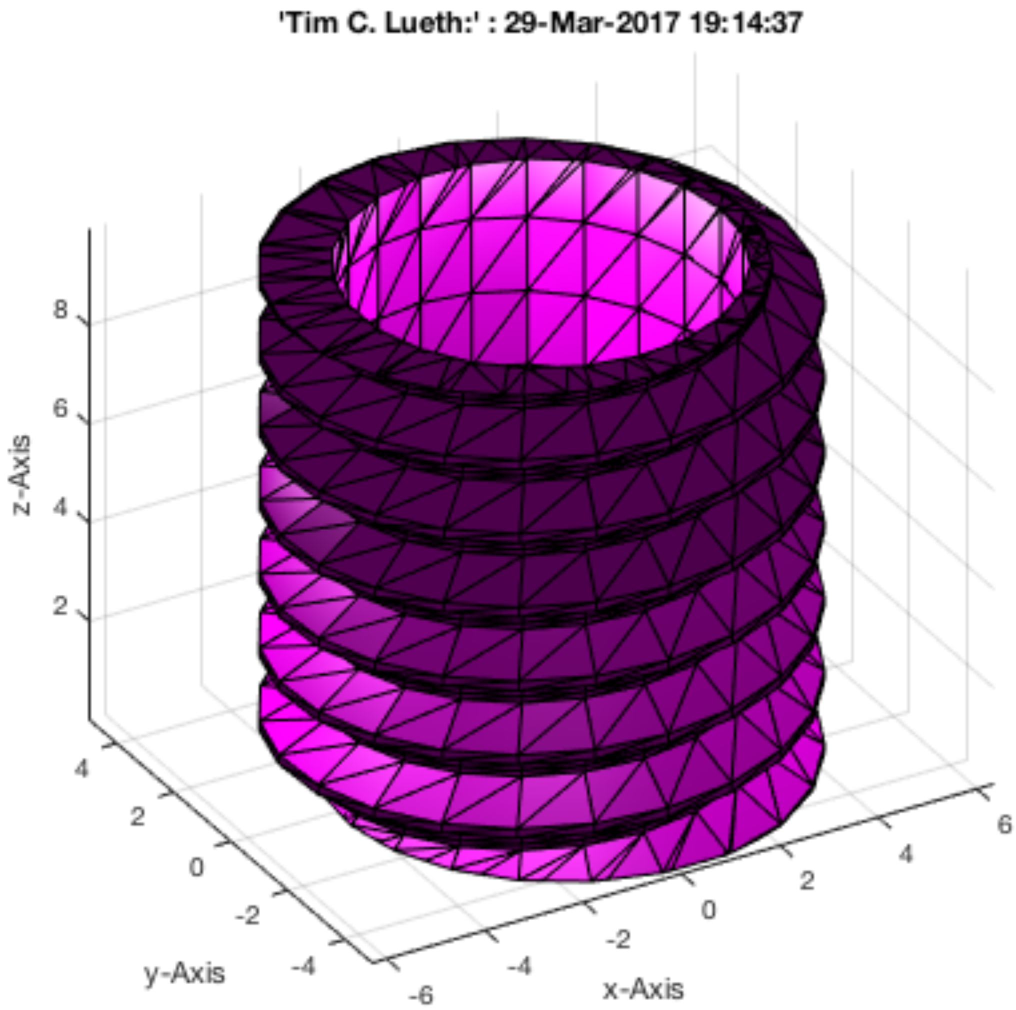
```
VLFLfigure;
SGthread (10,10,[],[],'C'); view (-30,30);
% [A,b,c]=SGthread (10,10);
```

'Tim C. Lueth:' : 29-Mar-2017 19:14:37



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

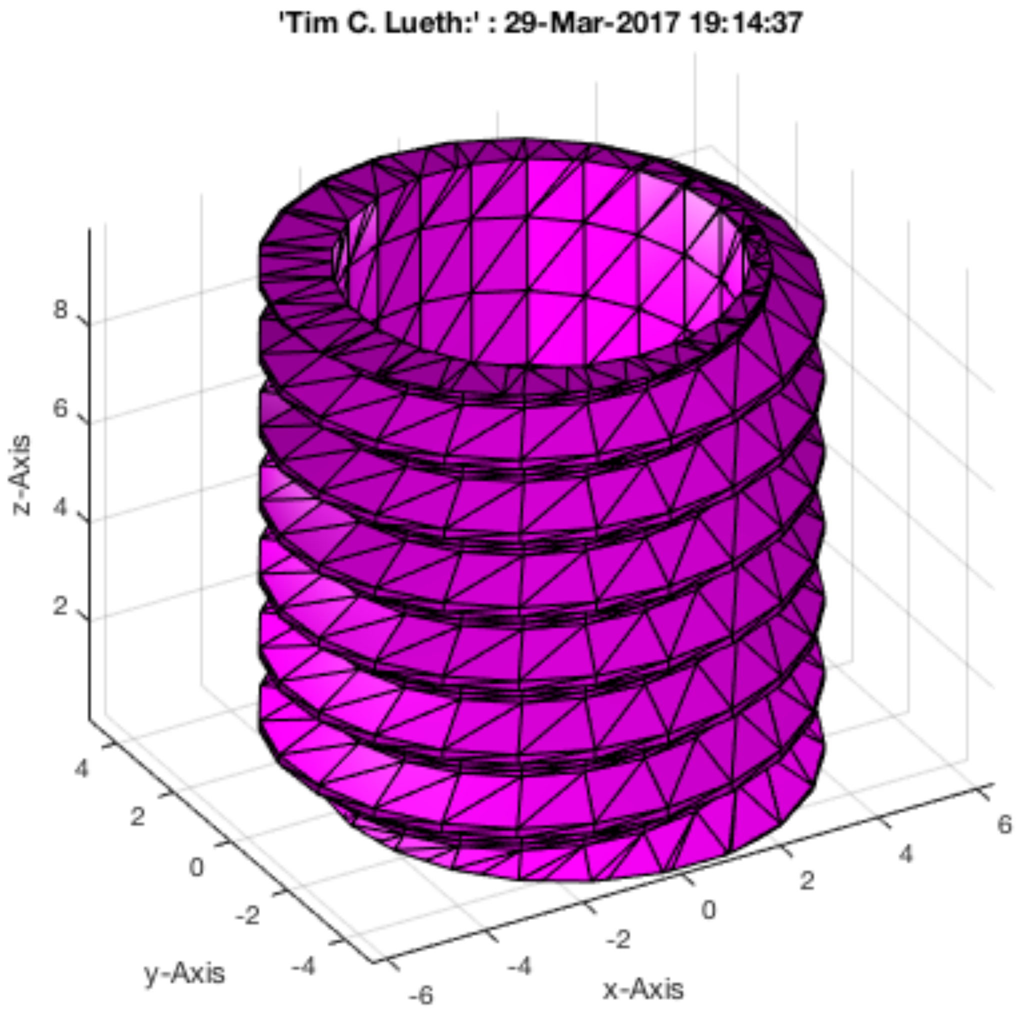
```
VLFLfigure;
A=SGthread (10,10+5+5,[],[],'C');
[~,B]=SGcut(A,[5.05 14.95]); B=SGtransP (B,[0 0 -5]);
SGplot(B,'m'); view (-30,30);
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:14:40 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 09: Boolean Operations with Solid Geometries

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 2. Remarks on boolean operations for solid geometries based on surfaces
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-6 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Import, and export of *STL-File* data, Conversion of text strings into solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- 2.5D rotation of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- Spatial transformation, positioning, alignment and merging of solid geometry sets

2. Remarks on boolean operations for solid geometries based on surfaces

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. #

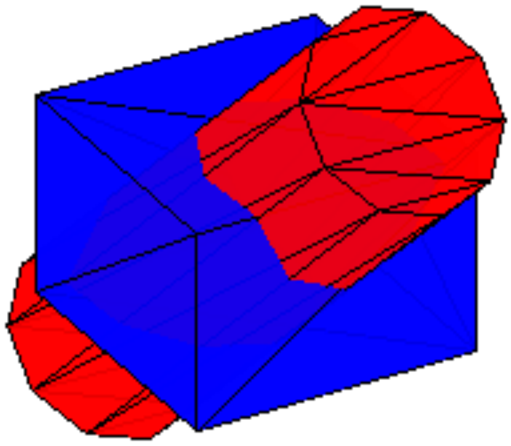
3. Creating two solids for showing the boolean operations

```
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)
```

```
A =

  struct with fields:

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


B =

  struct with fields:

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

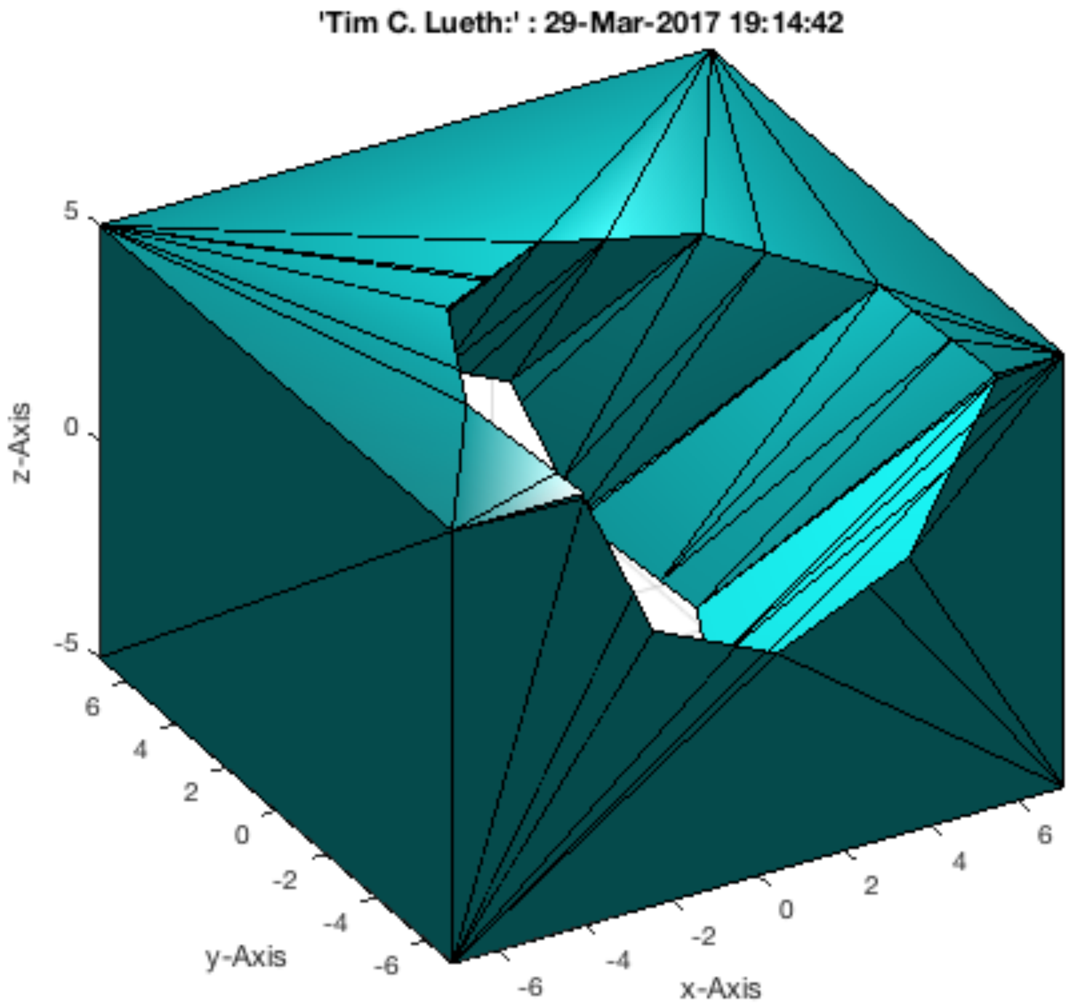
'Tim C. Lueth:' : 29-Mar-2017 19:14:41



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

```
close all;

C=SGbool ('-',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

```
C =

  struct with fields:

    VL: [60×3 double]
    FL: [120×3 double]
```

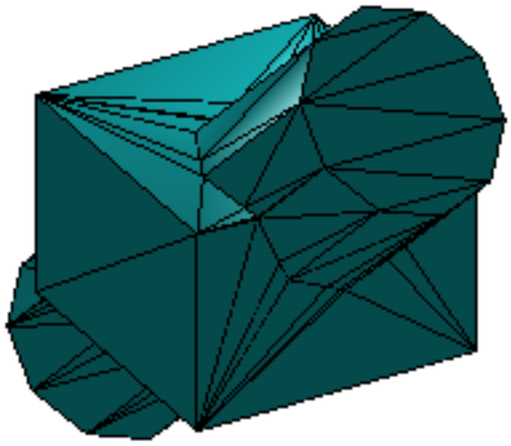


5. Boolean operator: Substraction A+B

```
close all;

C=SGbool ('+',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
return
```

```
C =

  struct with fields:

    VL: [80×3 double]
    FL: [156×3 double]
```

'Tim C. Lueth:' : 29-Mar-2017 19:14:43



6. Boolean operator: Substraction B\A

```
close all;

C=SGbool ('B',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

7. Boolean operator: A xor B

```
close all;

C=SGbool ('x',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

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
```

- *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*

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

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-9 we've learned about:

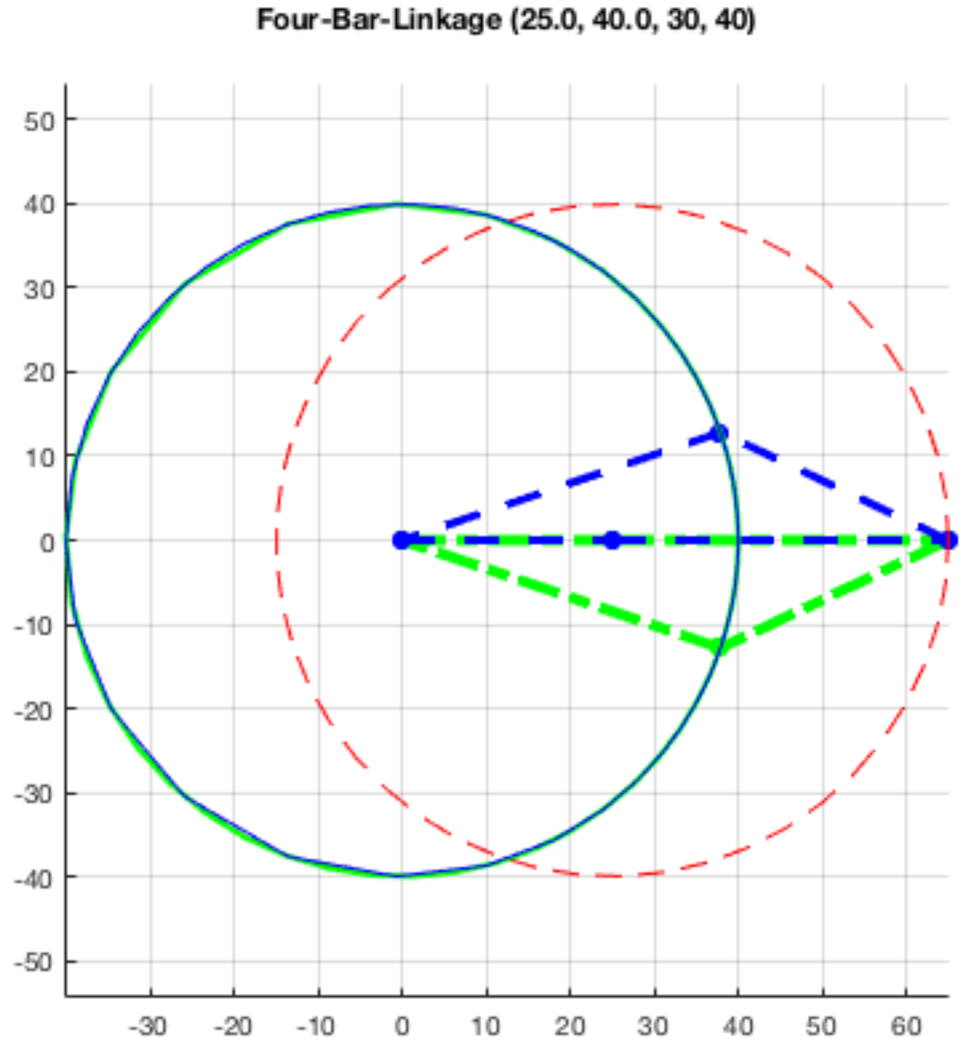
- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets
- Relative positioning of solids geometries and Boolean operations on Solids

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

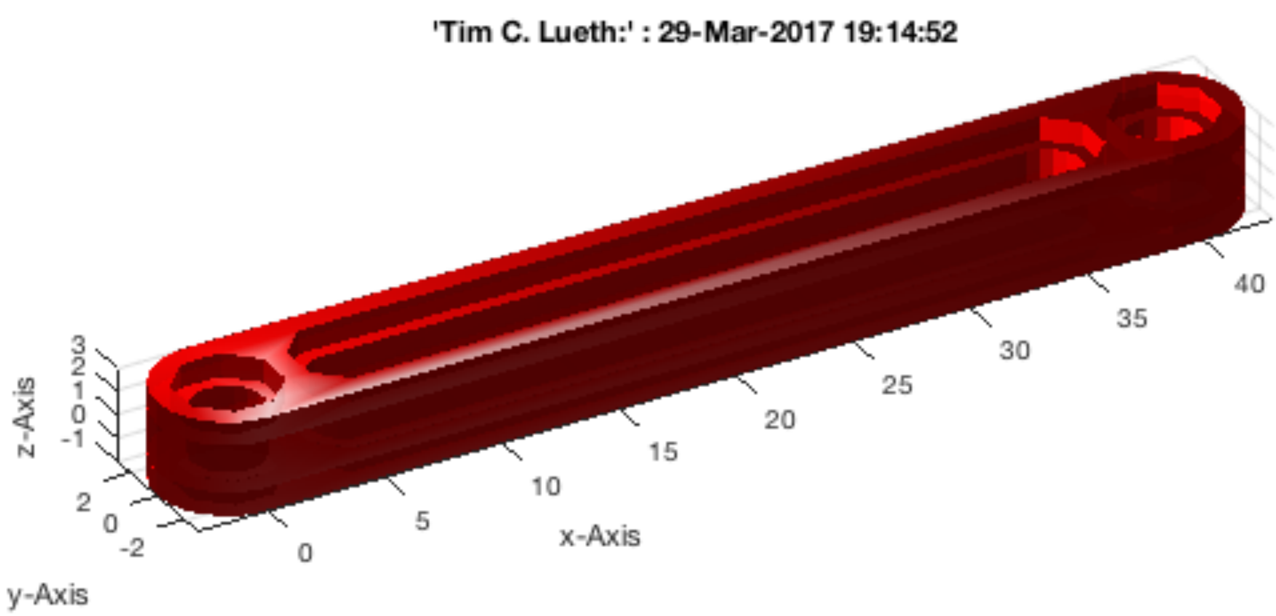
A very interesting mechanism in mechanics is the four-bar-linkage. It consists of four bars that are linked together by 4 rotatorial joints. Such a mechanism can be built by 4 different elememts

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

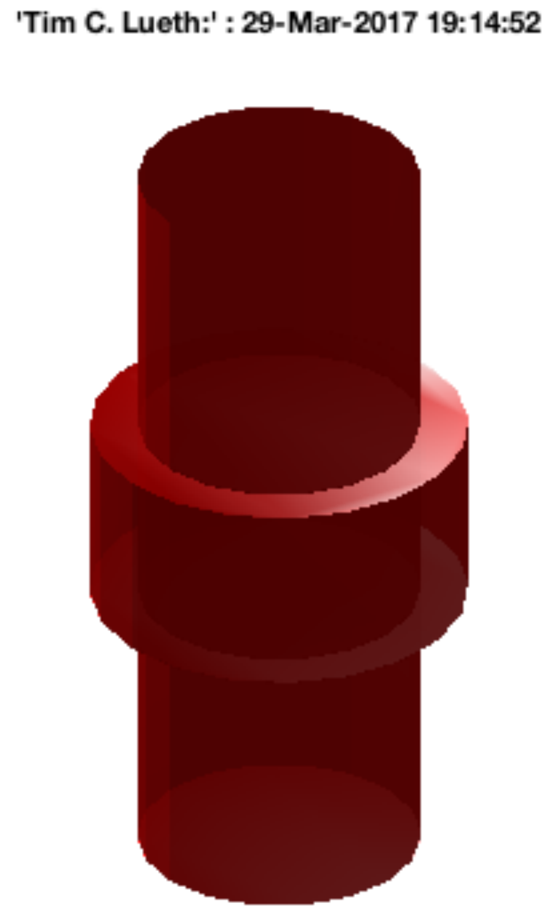
```
close all;
fourBarLinkage (25,40,30,40);
```



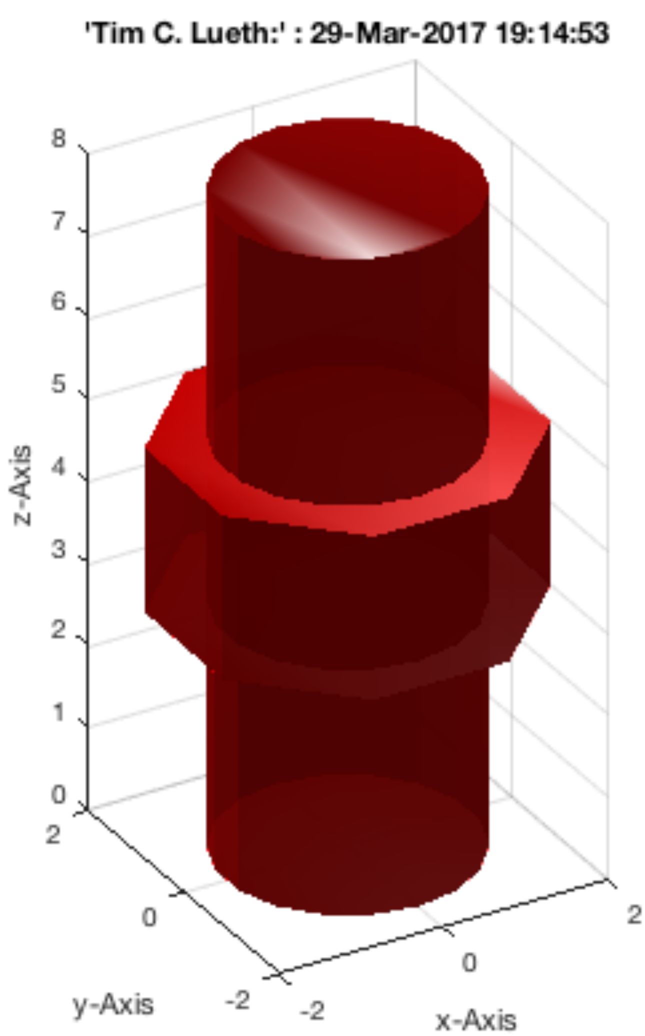

```
fourBarLinkageKit ('Bar',40);
```




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




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




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

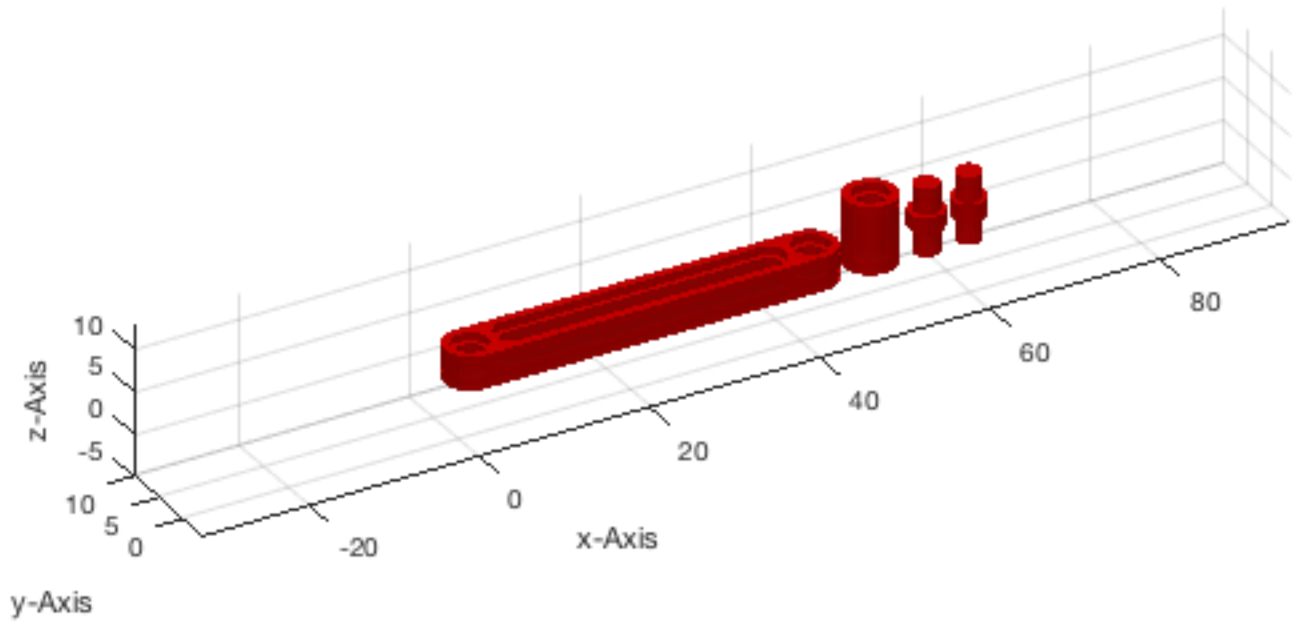
3. Packaging a set of solid geometries in a volume

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;
```

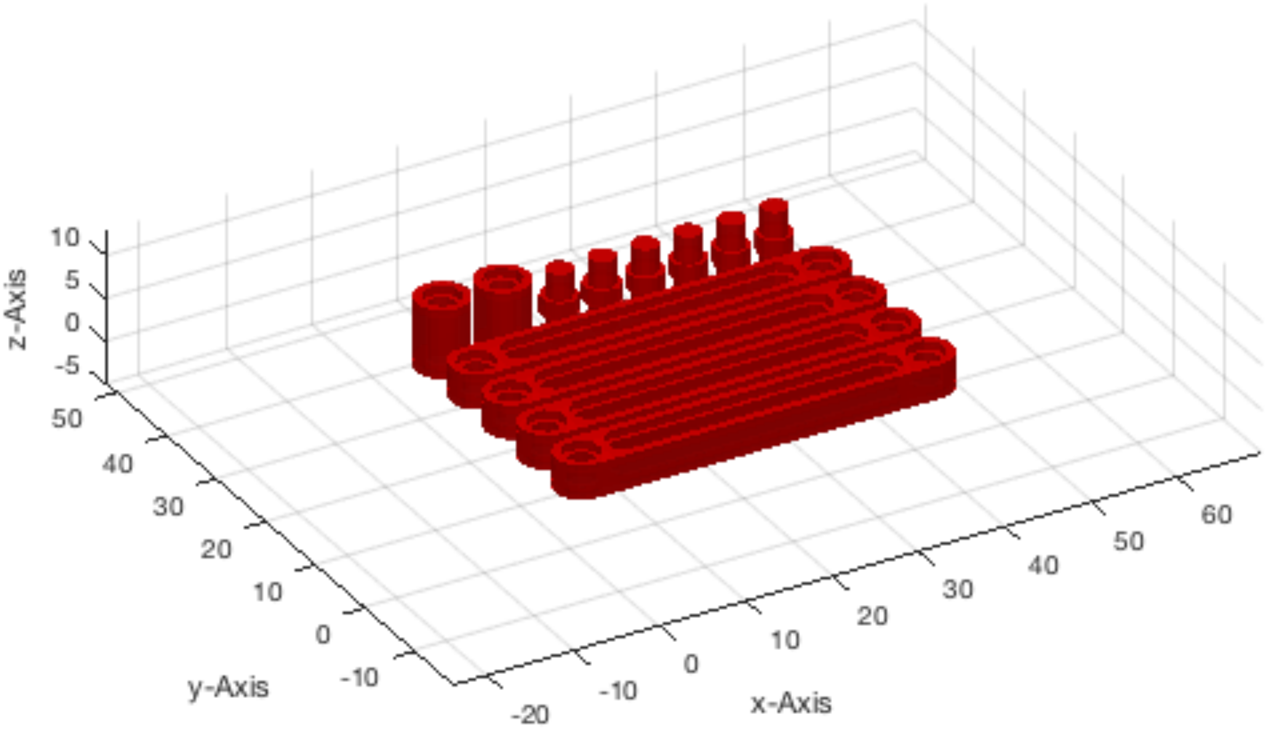
```
Packing 4 objects (h=23):
```



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=43):
```



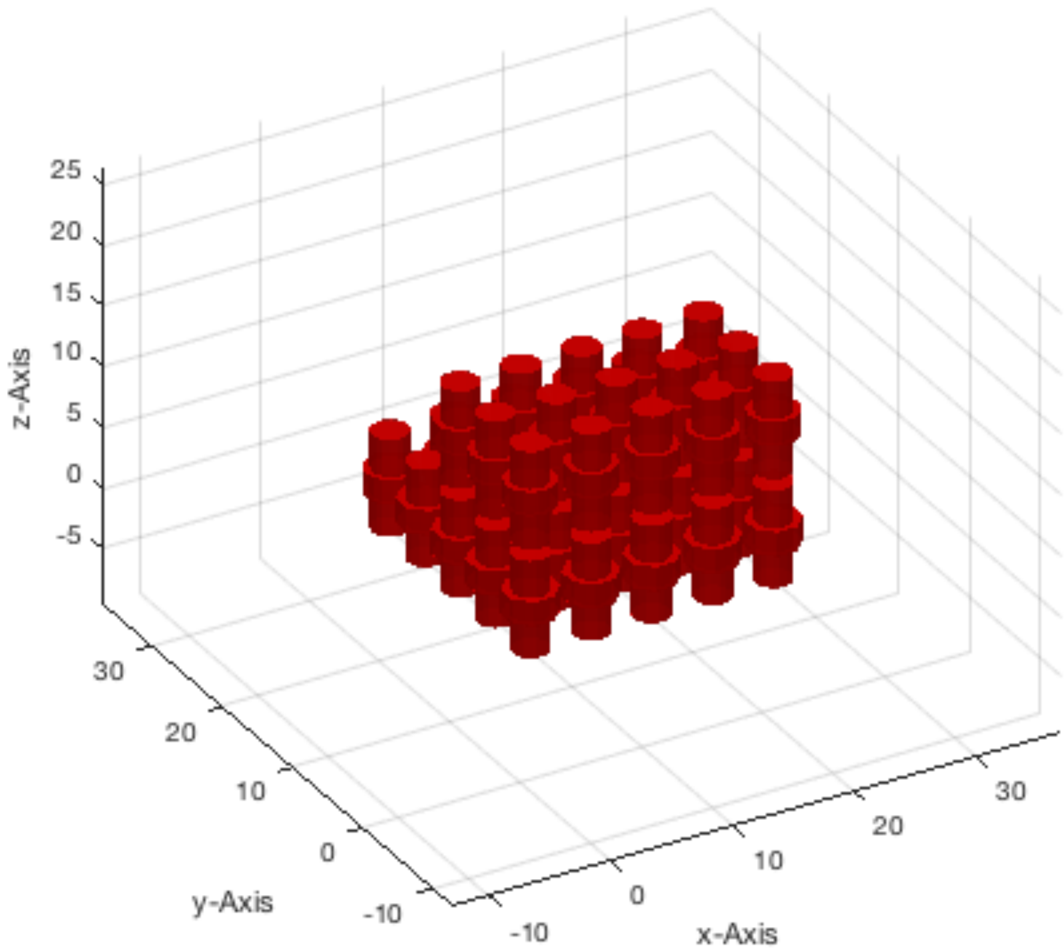
4. Using container/collections insted of itemizing the solids

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=38): ....................
```

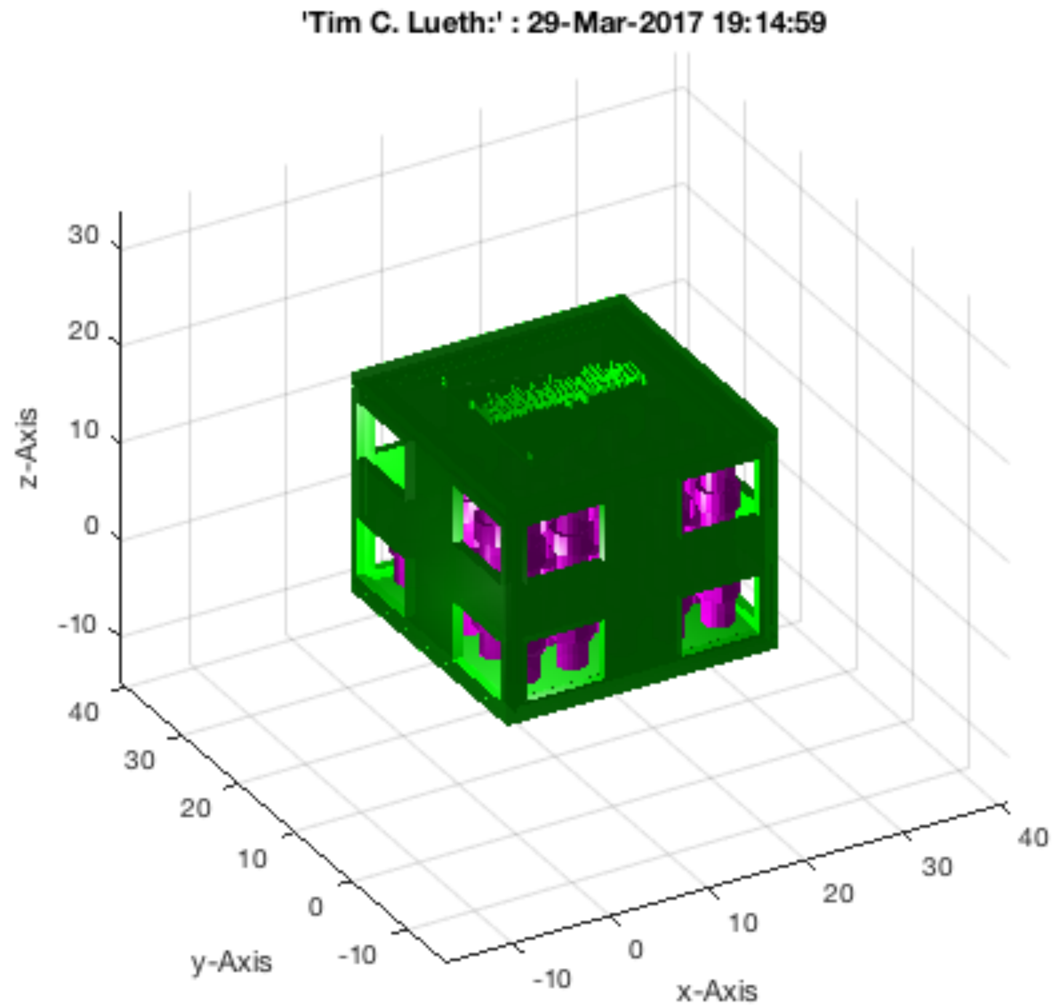


5. Create boxes around the packed solids for the final 3D print job

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;
```

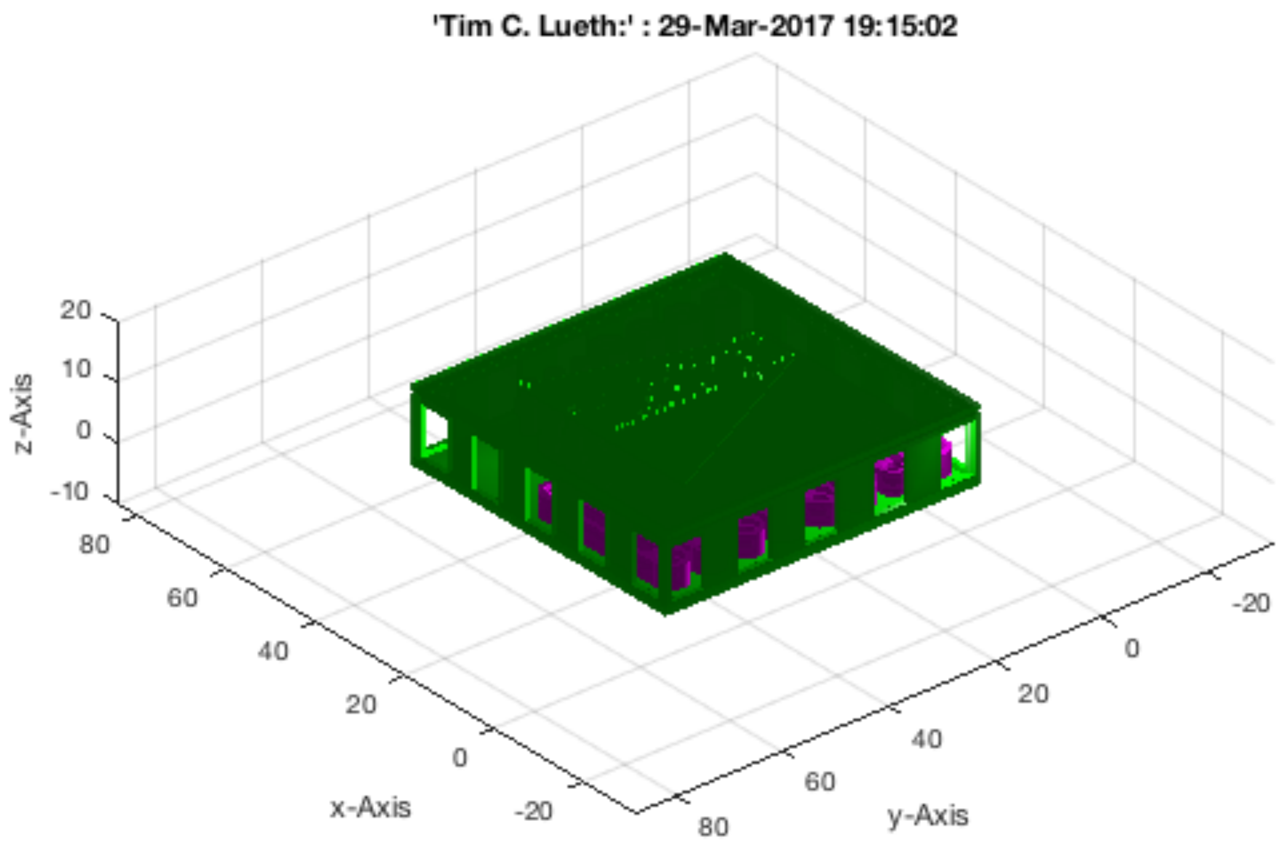
```
==>TEXT GENERATION..Analyze union areas for 60 facets:
Major union vectors: 6 found with maximum size of 1757.
Finally 2 union areas found with size > 100
Text attached to union Nr: 2
..finished!
```



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=57): ....................
==>TEXT GENERATION..Analyze union areas for 60 facets:
4 Dimension warnings
Major union vectors: 6 found with maximum size of 7157.
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..
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:15:05 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 11: Attaching Coordinates Frames to Create Kinematik Models

2015-06-08: Tim C. Lueth, MIMED - Technische Universität München, Germany 2015-08-18: Tim C. Lueth, changed for better kinematic model structure (URL: http://www.mimed.de)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

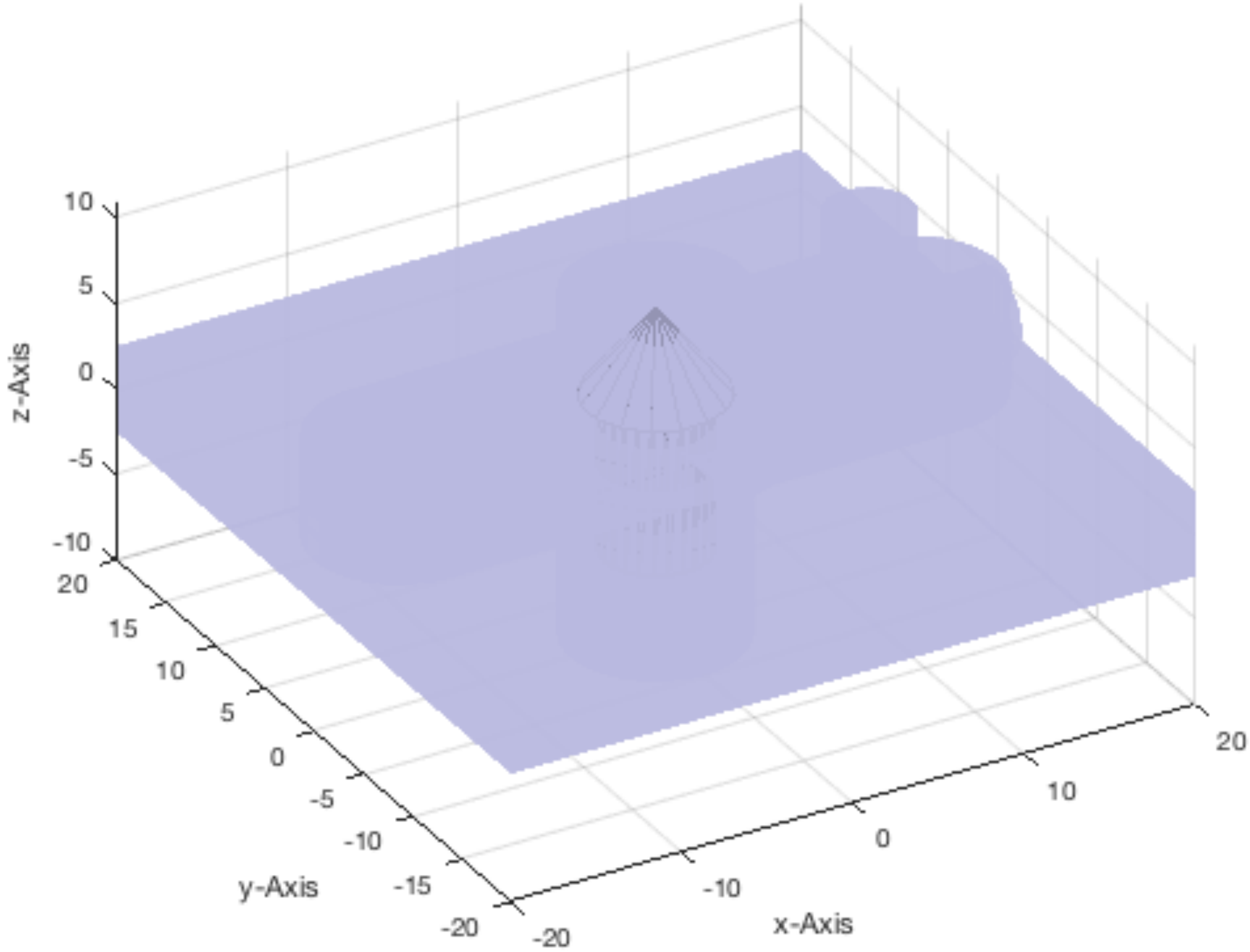
In examples 1-10 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polygon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets
- Relative positioning of solids geometries and boolean operations on solids
- Packaging and boxing of several parts into a complete assemble kit

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

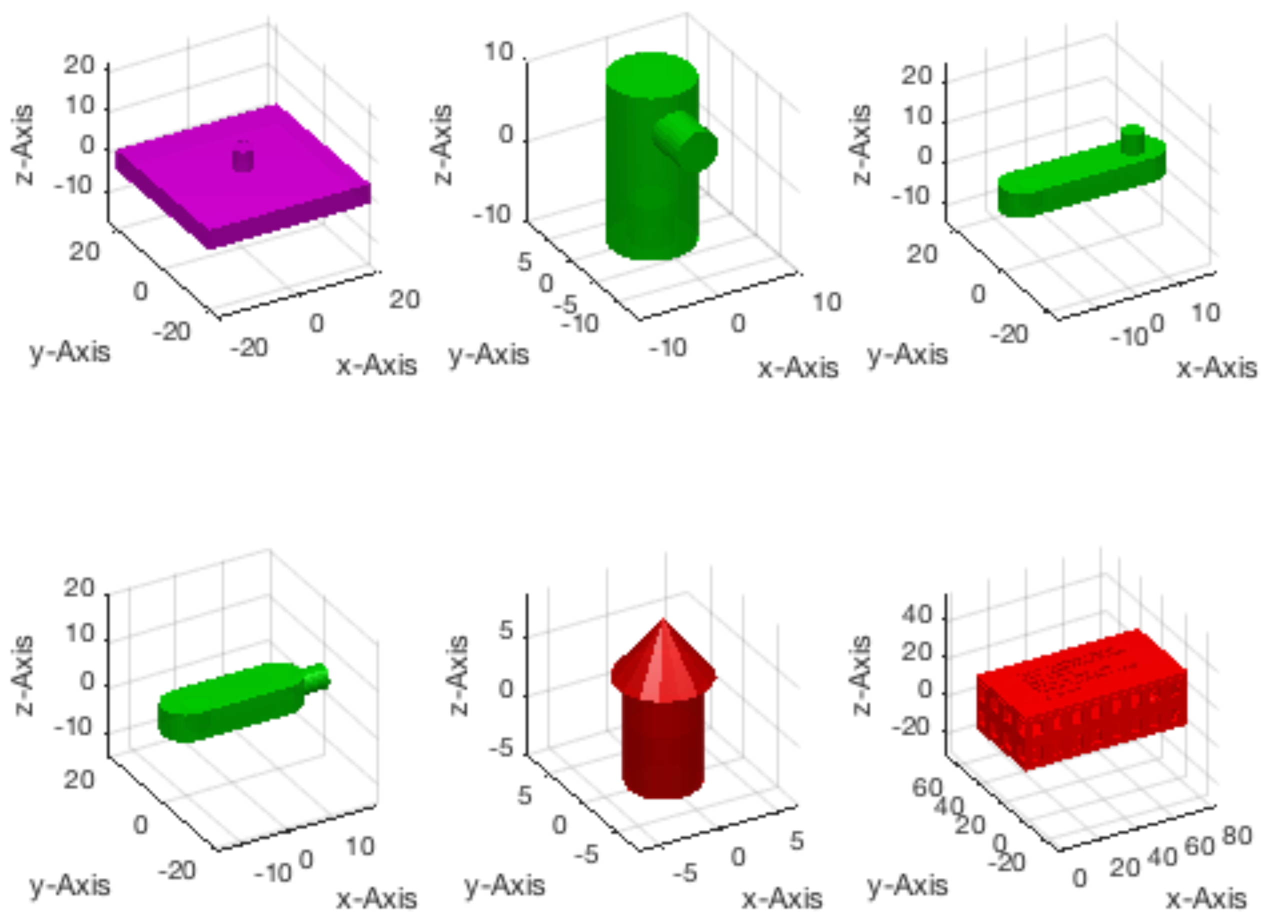
```
load AIM_SGrobot
SG0=SGfixerrors(SG0,1e-3); SGchecker(SG0);
SG1=SGfixerrors(SG1,1e-3); SGchecker(SG1);
SG2=SGfixerrors(SG2,1e-3); SGchecker(SG2);
SG3=SGfixerrors(SG3,1e-3); SGchecker(SG3);
SG4=SGfixerrors(SG4,1e-3); SGchecker(SG4);
save ('AIM_SGrobot','SG0','SG1','SG2','SG3','SG4','SGrobot');
```



- 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]}
```

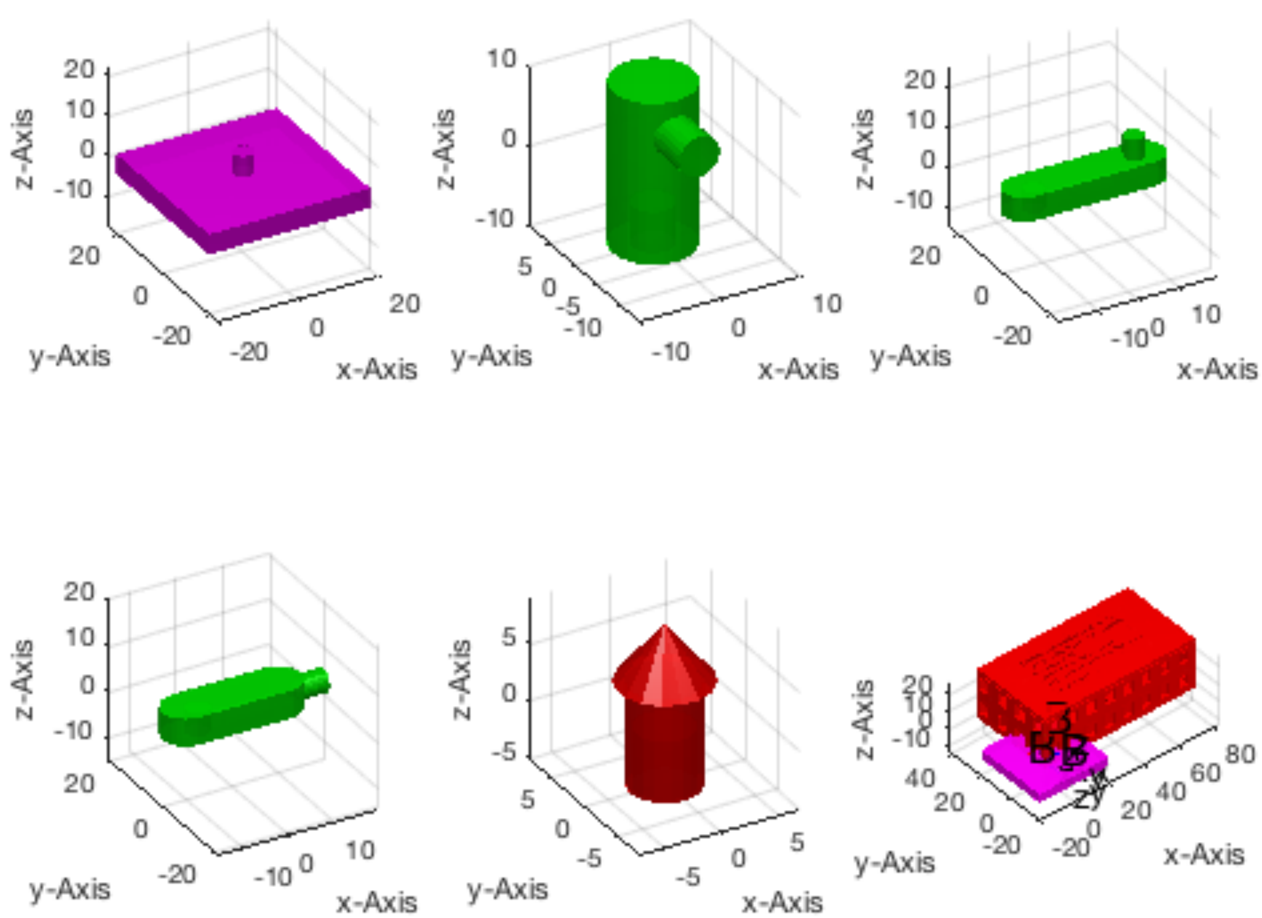
- *Tname* is a cell list contain the ascii-string of the names of the coordinate frames
- *T* is the 4x4 homogenous transformation matrix related to the indexed name
- *TFiL* is an optional index of the facets that belong to the surface that defines the coordinate system To the homogenous transformation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

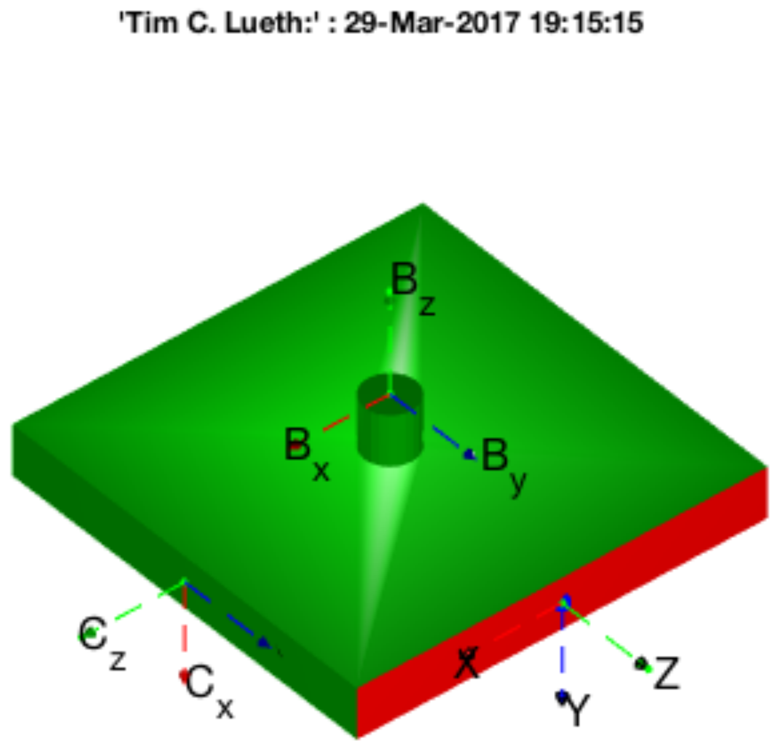
Since there is no requirement to use the facets TFiL, T matrices and their name can easily added by a programm during the design phase. Nevertheless, there is also a need to add frames interactively. For that purpose there are two other functions to add or to remove frames.

- **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]}
```

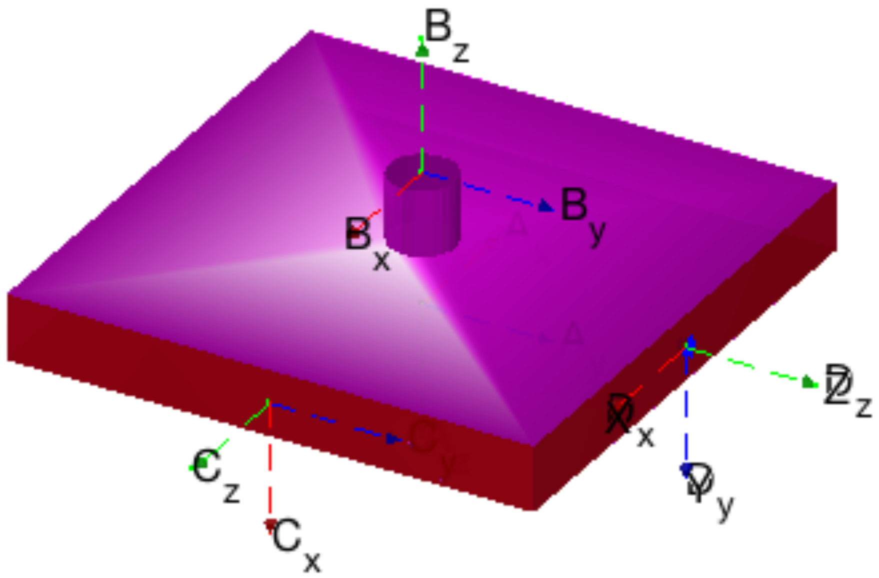


No we remove the first two frames 'A' and 'B'

```
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]}
```

'Tim C. Lueth:' : 29-Mar-2017 19:15:15



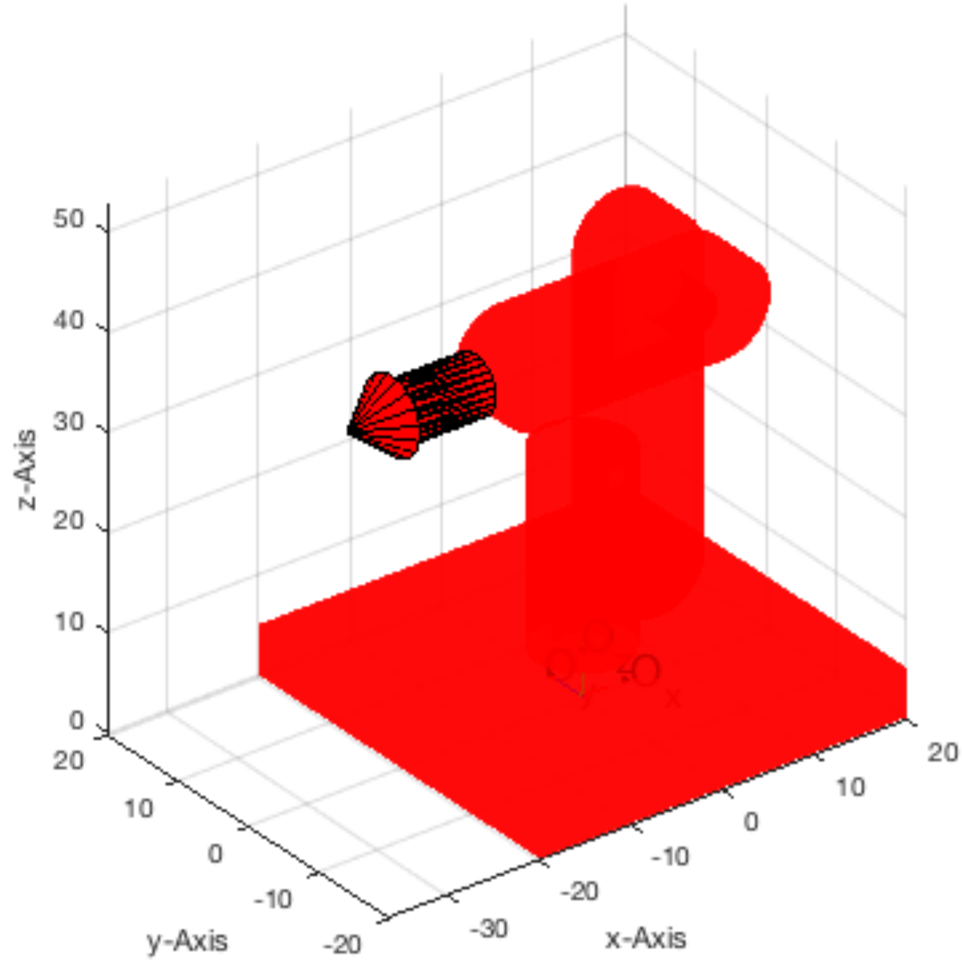
5. Creating kinematic models consisting of named solids

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'}
```

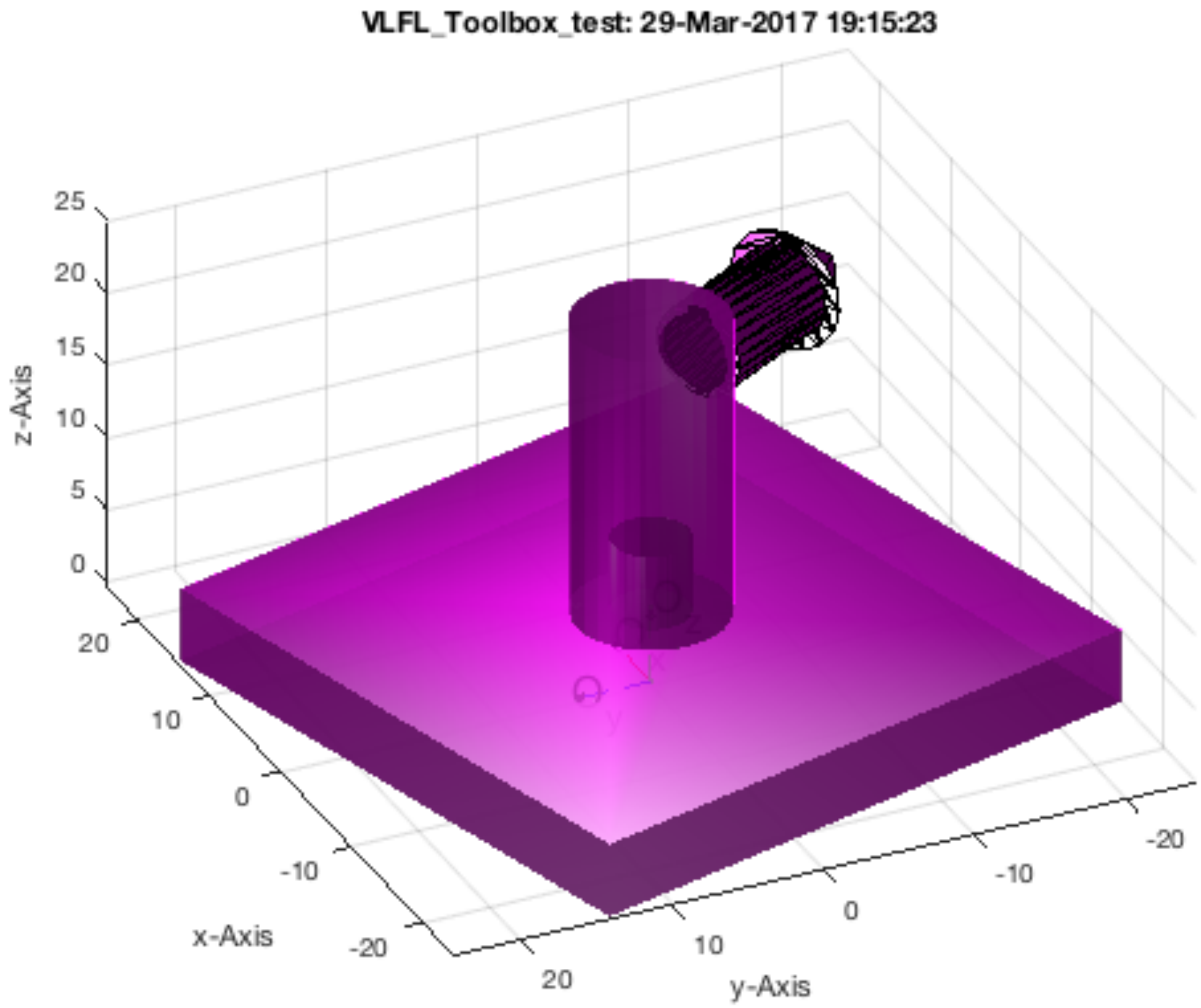
```
        BT: {1×5 cell}


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    0.0000    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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:15:26 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 12: Define Robot Kinematics and Detect Collisions

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-11 we've learned about:

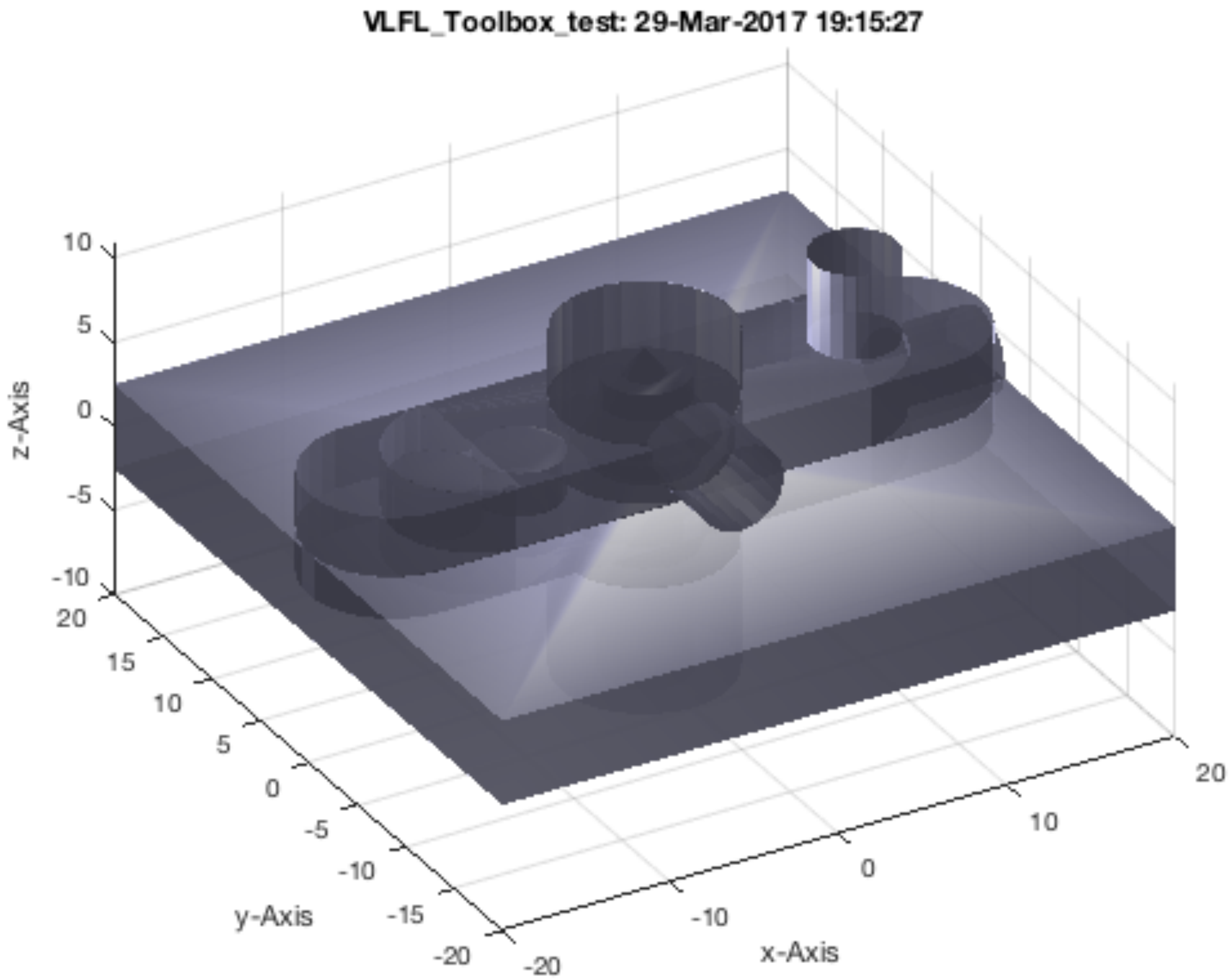
- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets
- Relative positioning of solids geometries and boolean operations on solids
- Packaging and boxing of several parts into a complete assemble kit
- Attaching coordinates frames for creation of Kinematik Models

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
```

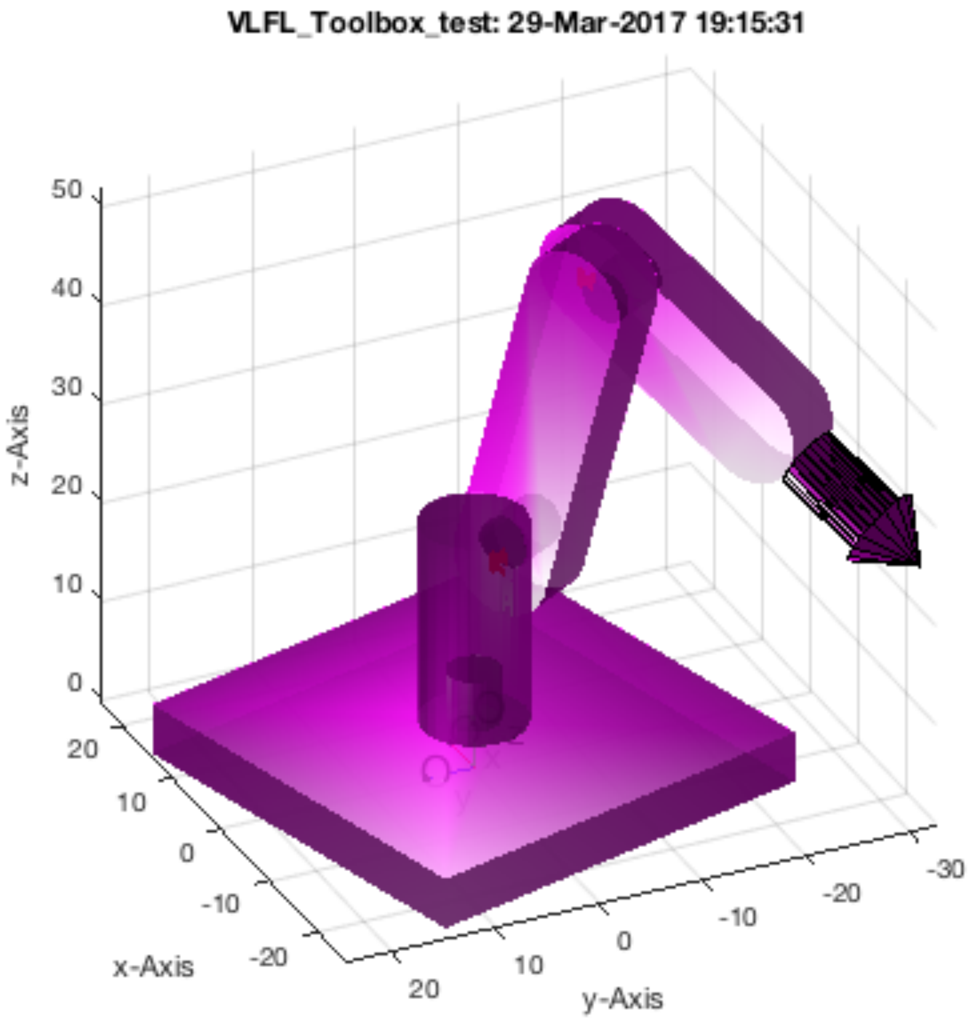
```
close all; SGfigure;
load ('AIM_SGrobot')
SG0=SGfixerrors(SG0,1e-3); SGchecker(SG0);
SG1=SGfixerrors(SG1,1e-3); SGchecker(SG1);
SG2=SGfixerrors(SG2,1e-3); SGchecker(SG2);
SG3=SGfixerrors(SG3,1e-3); SGchecker(SG3);
SG4=SGfixerrors(SG4,1e-3); SGchecker(SG4);
% save ('AIM_SGrobot','SG0','SG1','SG2','SG3','SG4','SGrobot');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



3. Automatic creation of a the robot

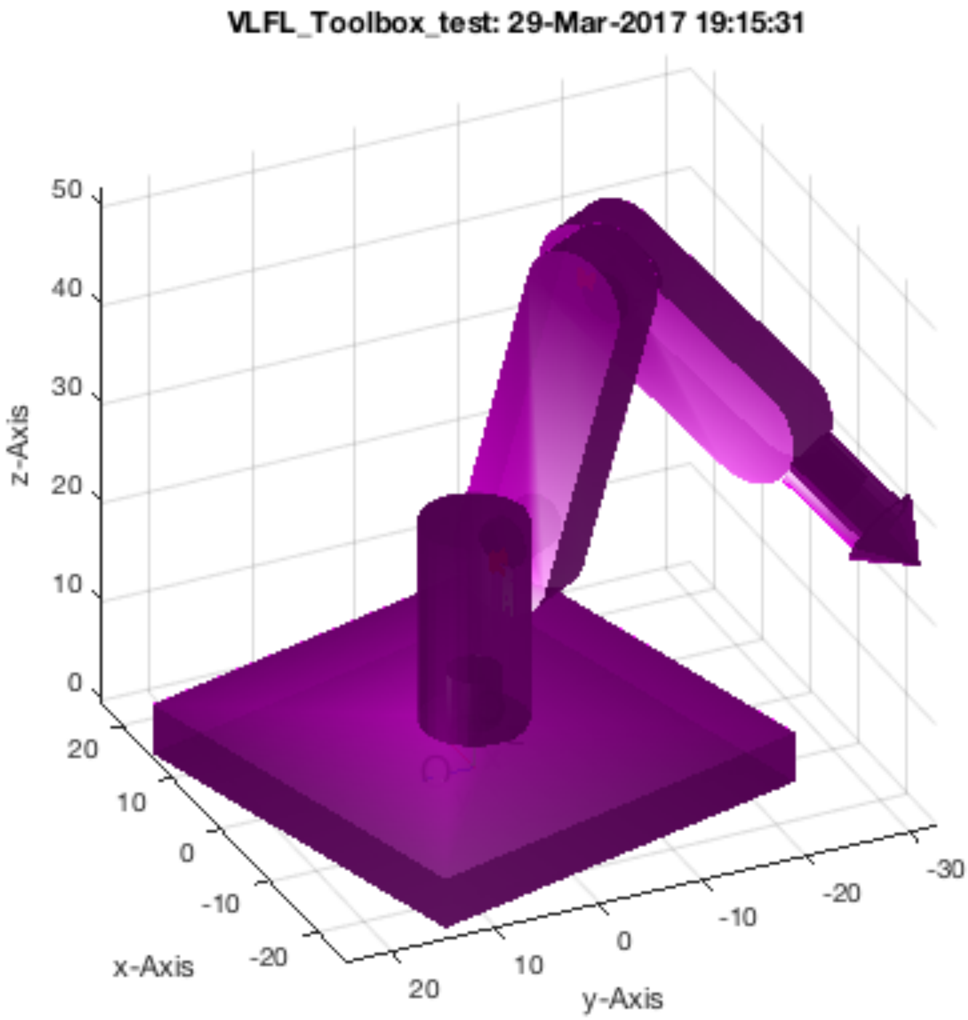
```
KMofSGs({SG0,SG1,SG2,SG3,SG4});
```

```
Warning in KMofSGs: 20 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);
```

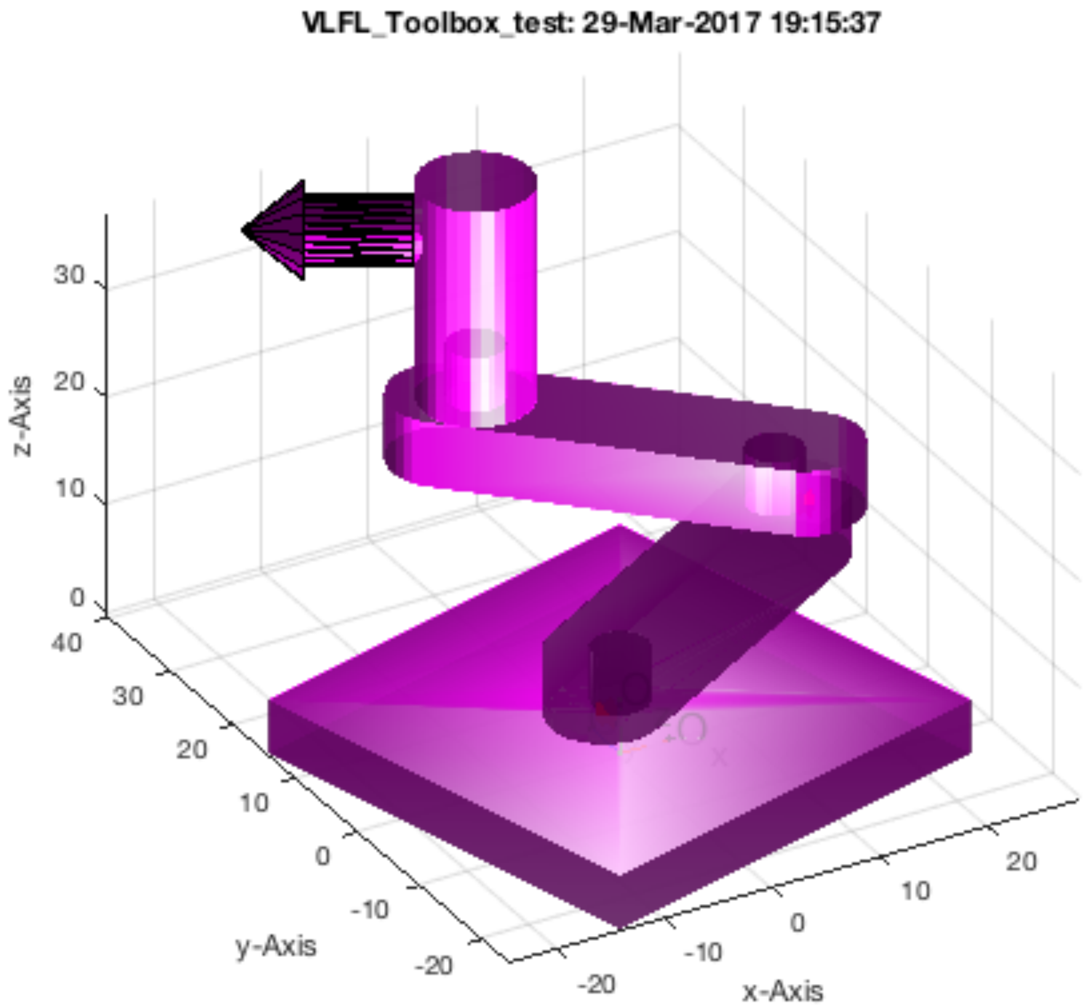



```
if ~isempty(XVL); zoompatch(XVL); VLplot(XVL,'k*',10);  end;
```

4. Showing a different robot

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

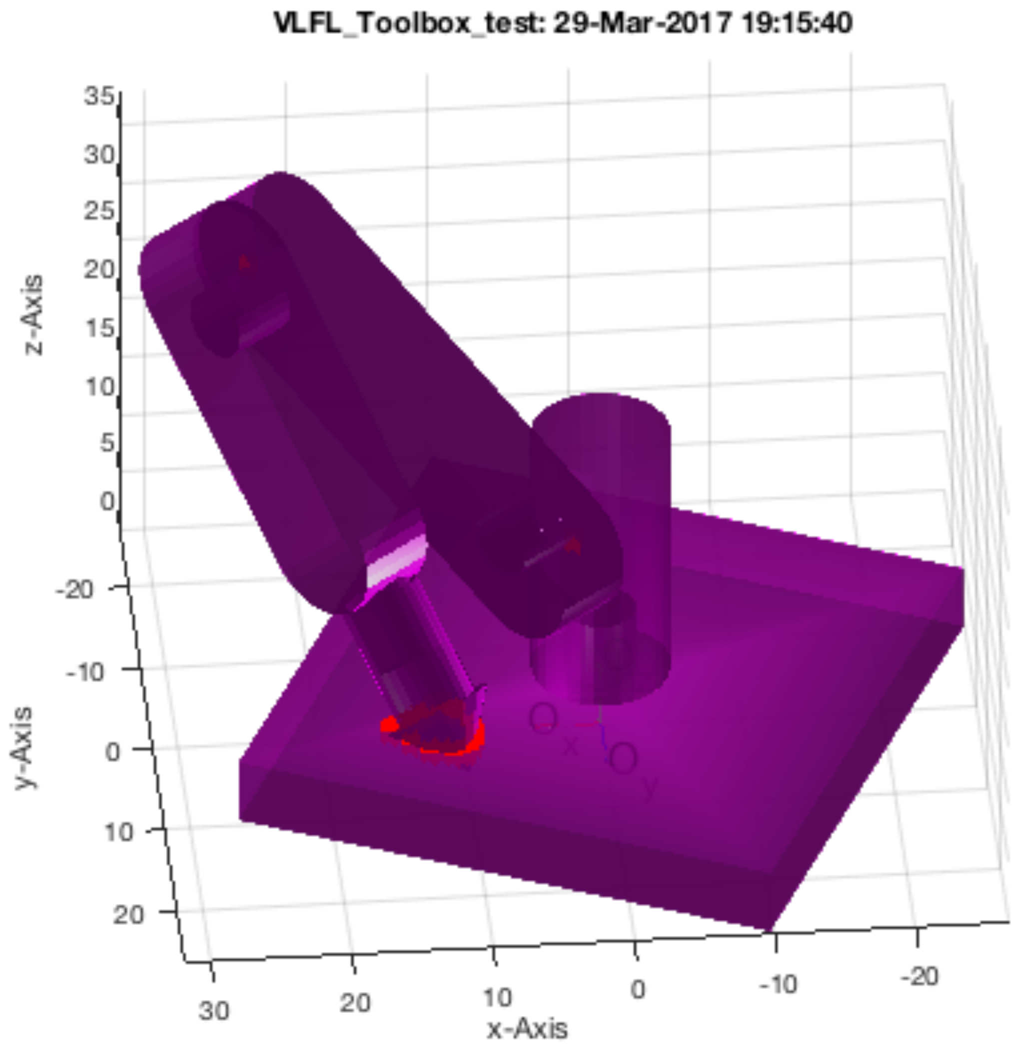
```
Warning in KMofSGs: 20 collisions found for tolerance: 0.10
```



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: 125 collisions found for tolerance: 0.10
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:15:43 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-12 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets
- Relative positioning of solids geometries and boolean operations on solids
- Packaging and boxing of several parts into a complete assemble kit
- Attaching coordinates frames and the creation of kinematik models

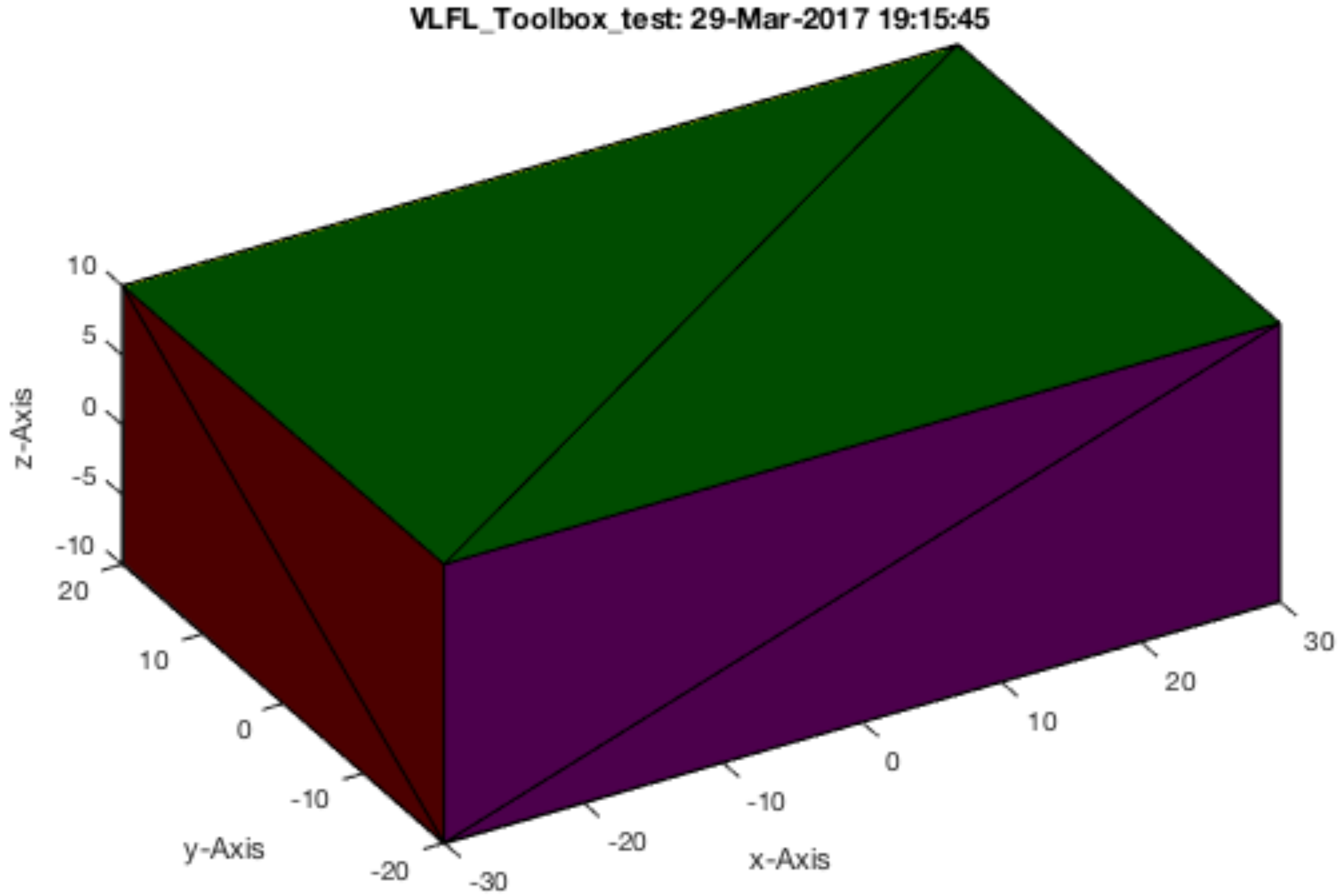
2. Analyzing mouting faces of flat surfaces

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

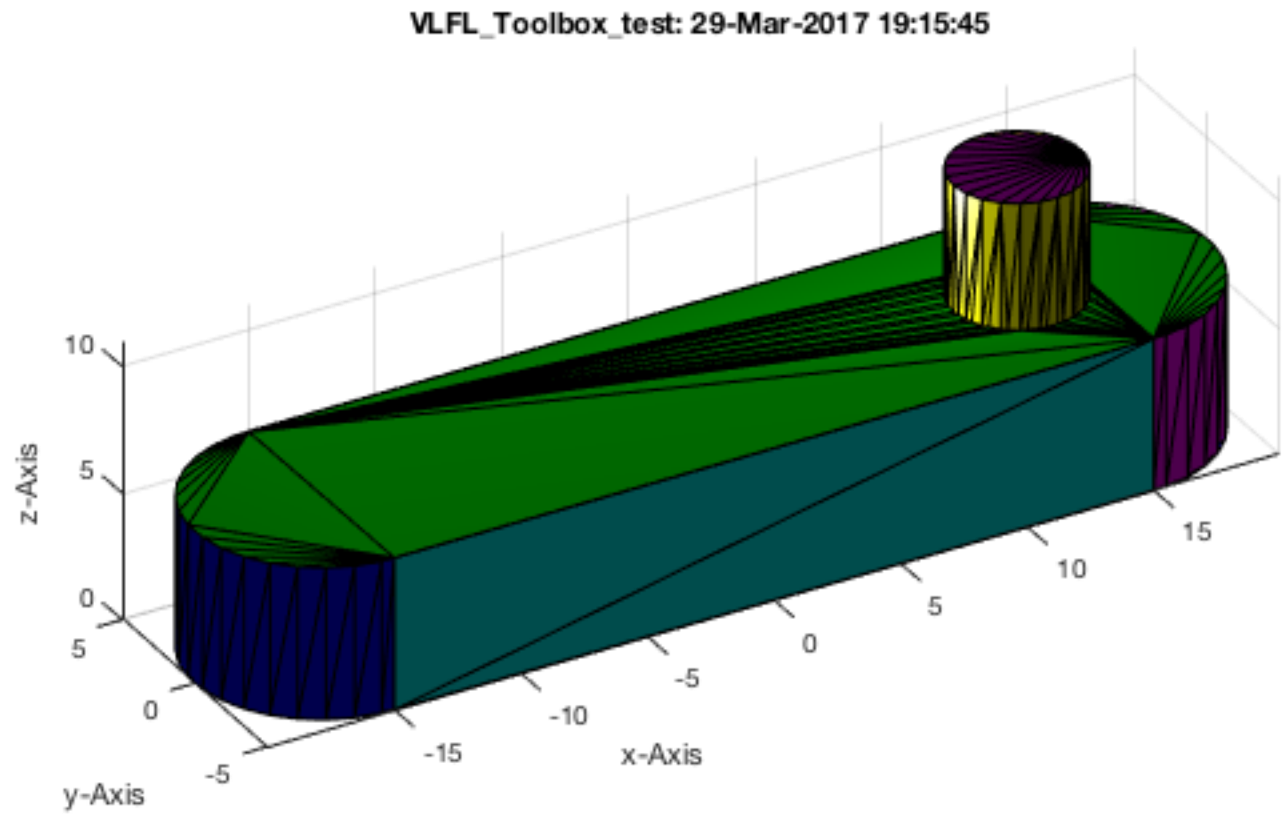
```
SG
```

```
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);
```

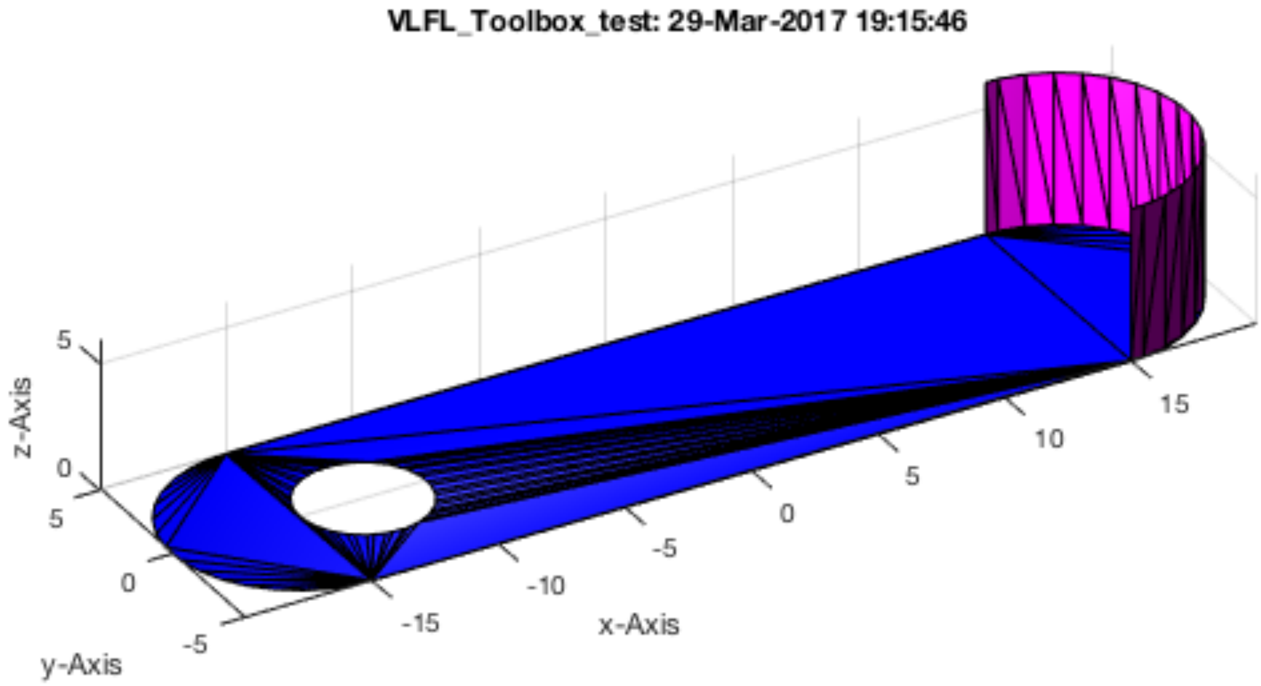


Now we plot only the two half-spherical mounting faces 5 and 1 of the robot link

```
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   -0.0000         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    0.0000         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
```

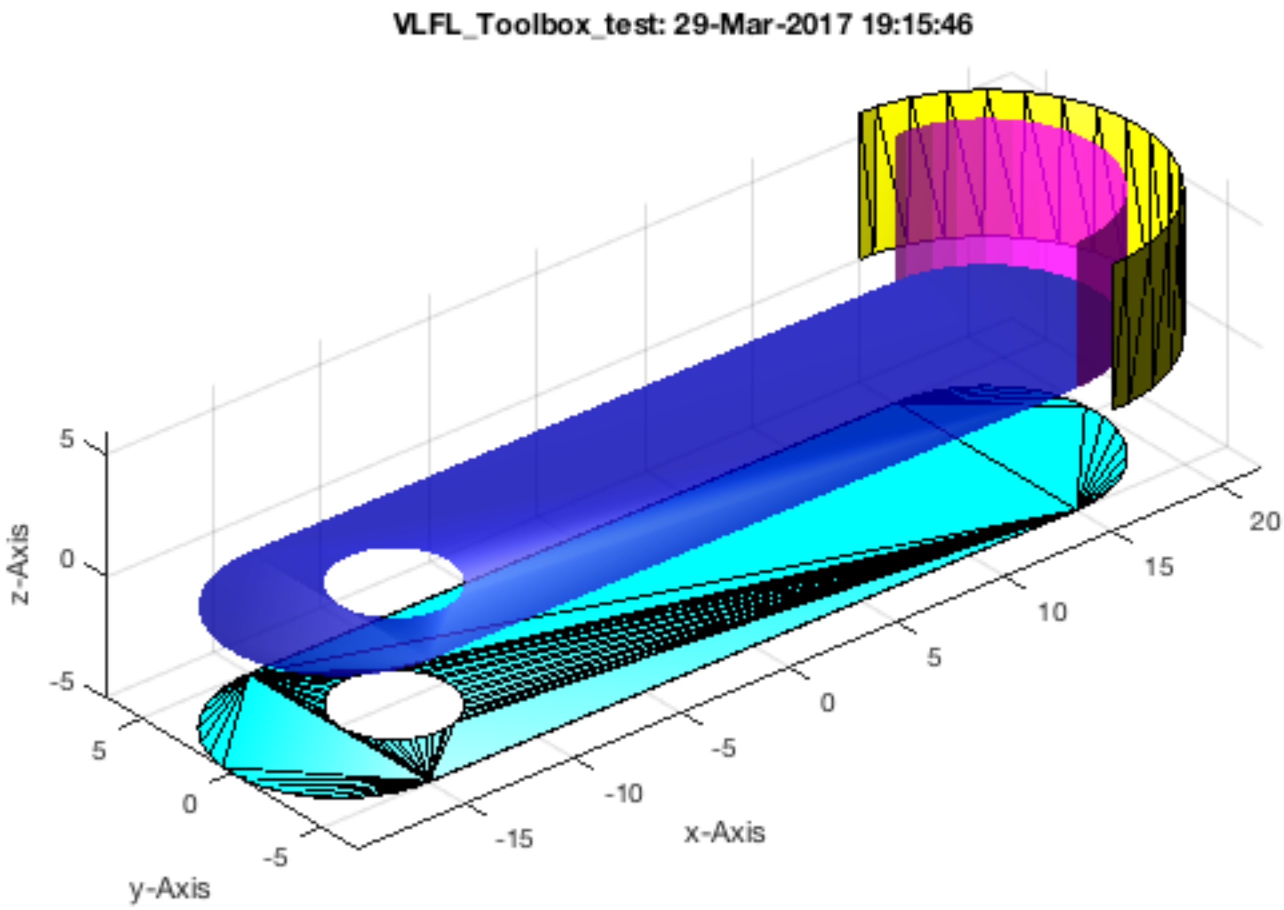


4. Create corresponding surfaces parallel to mounting faces

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');
```

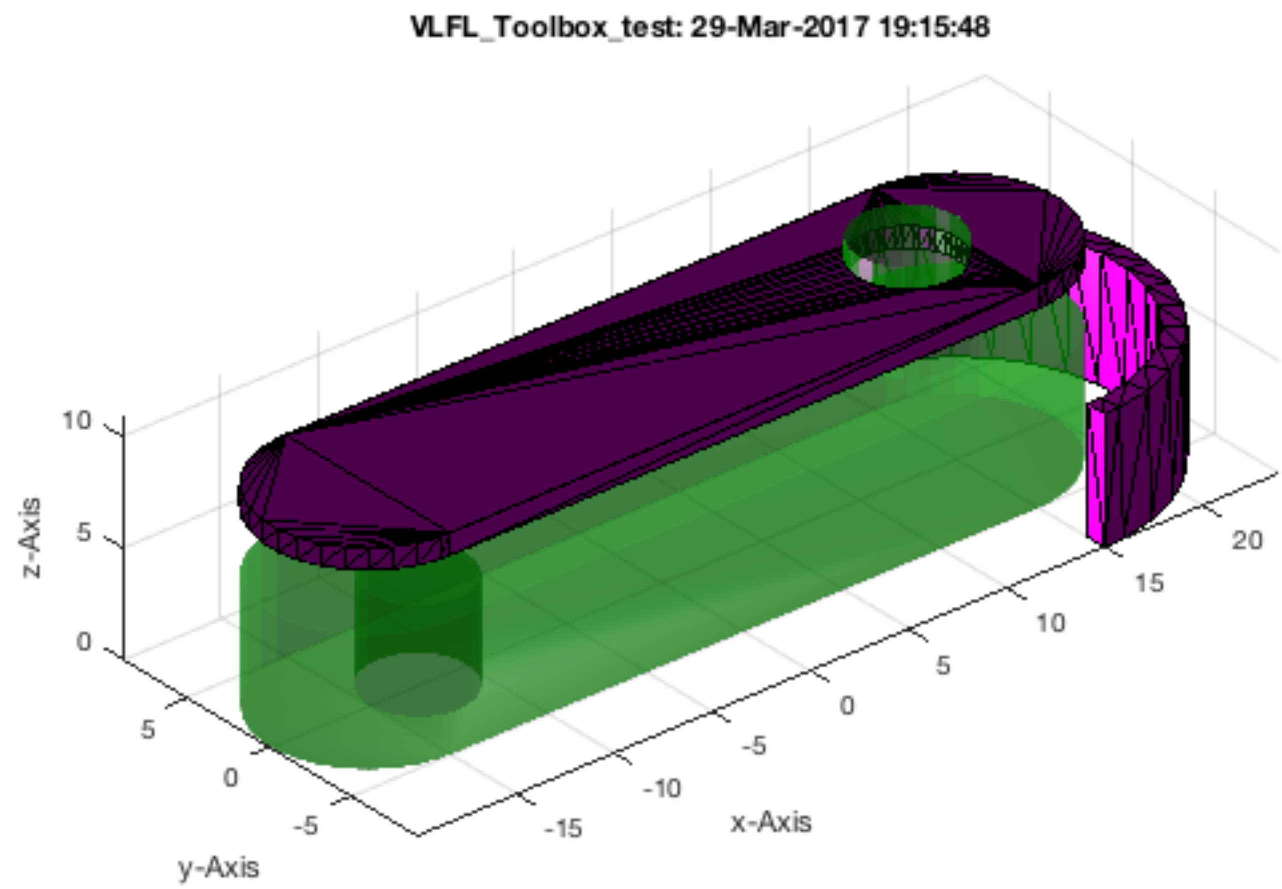


5. Create solids using the parallel surfaces and a plate thickness

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');
```

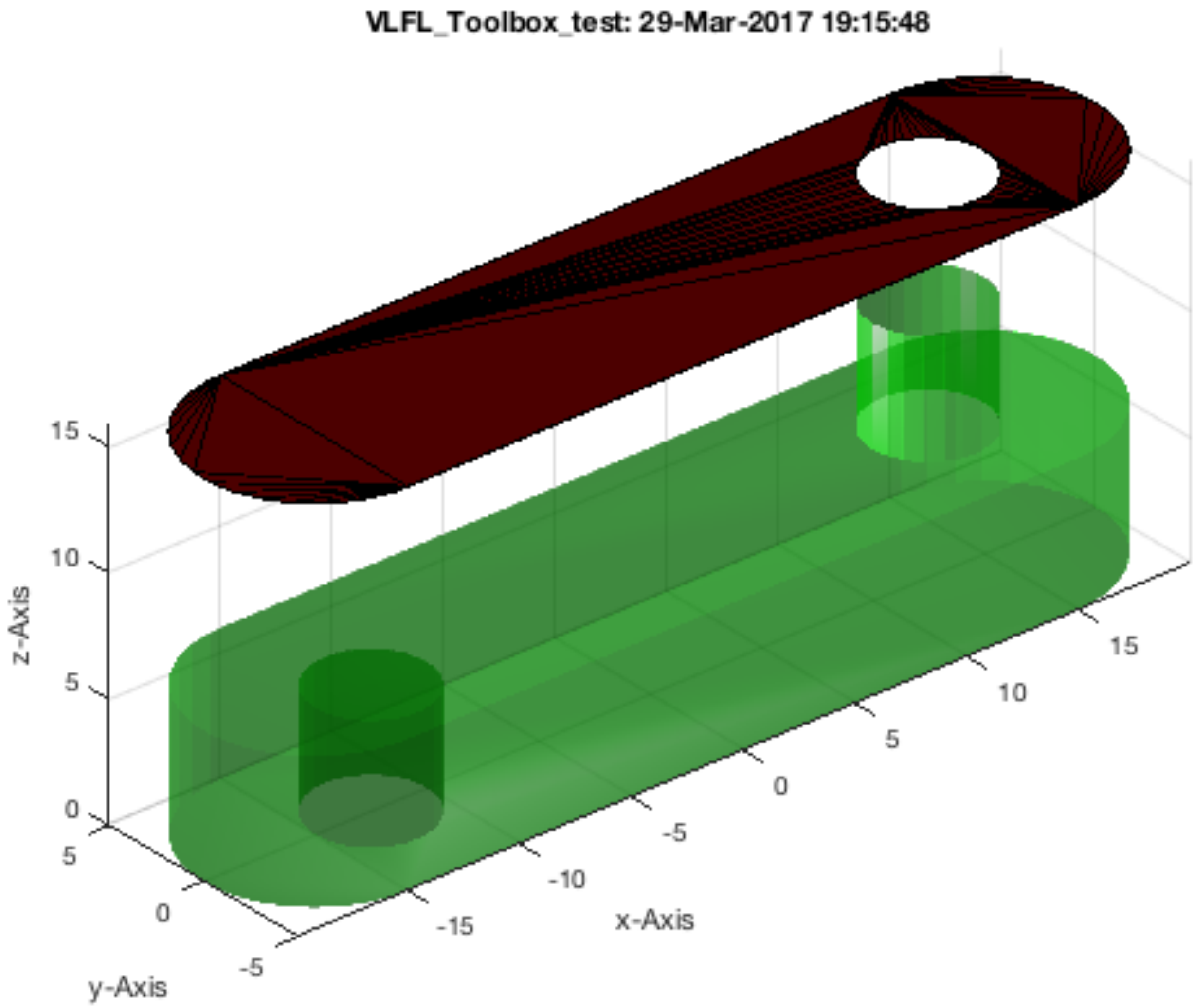


6. Finding the 2D CPL of a planar 3D Surface

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.

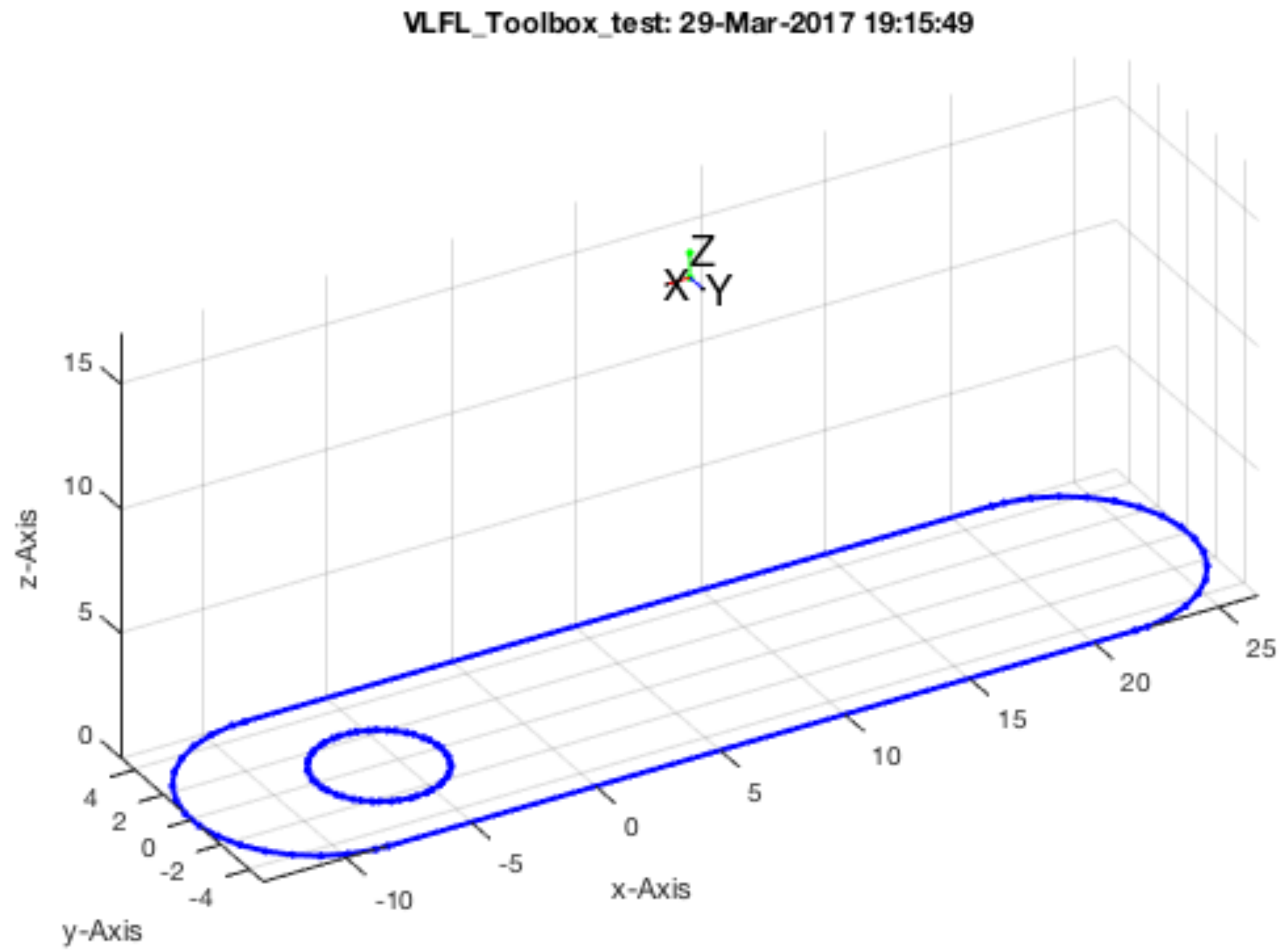
- [PL,T]=PLofVLFL(VL,FL) - returns a PL and a transformation matrix
- [CPL,T]=CPLofVLFL(VL,FL) - returns a CPL and a transformation matrix

```
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);
```

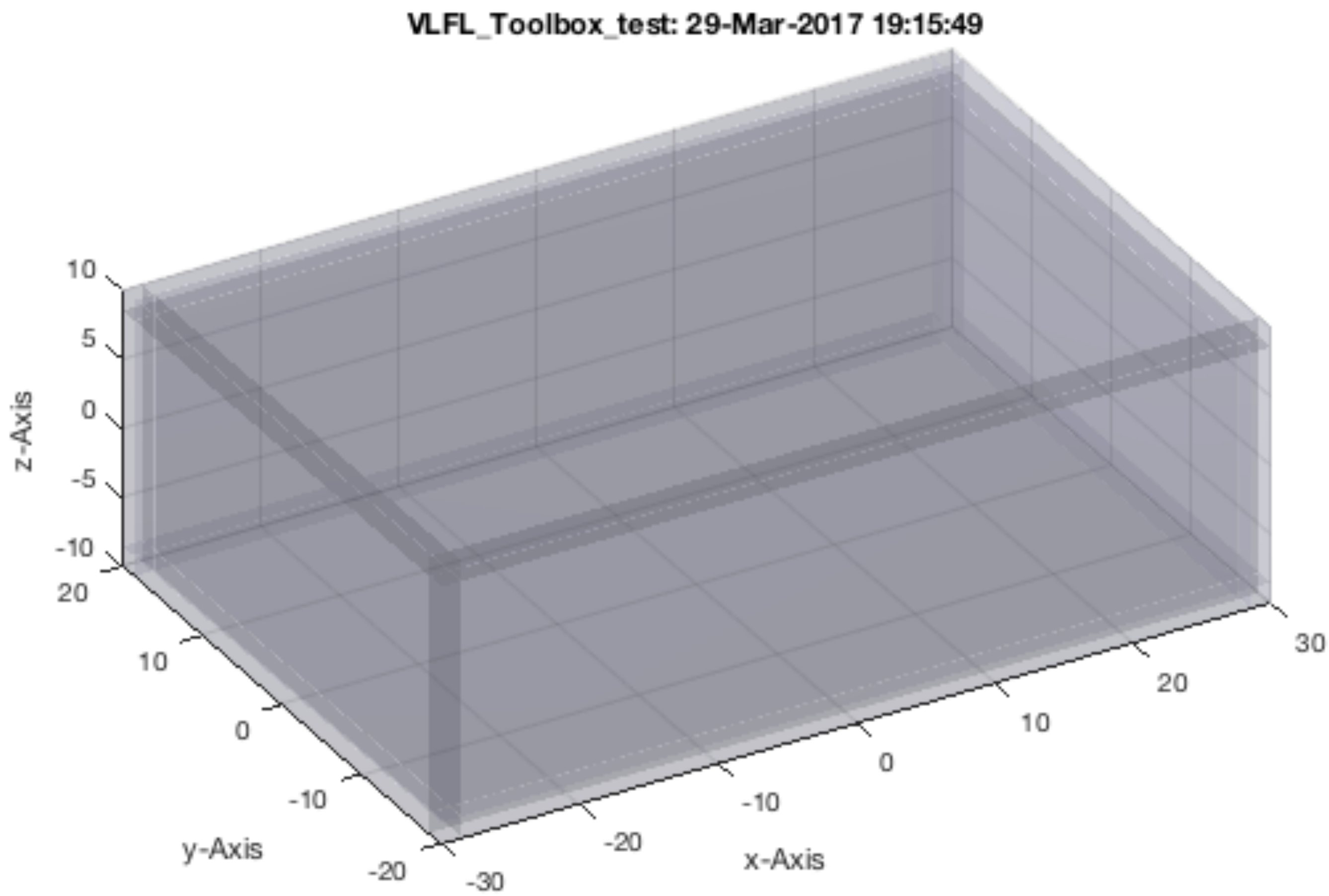


7. Replace a solid block by covering plates

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

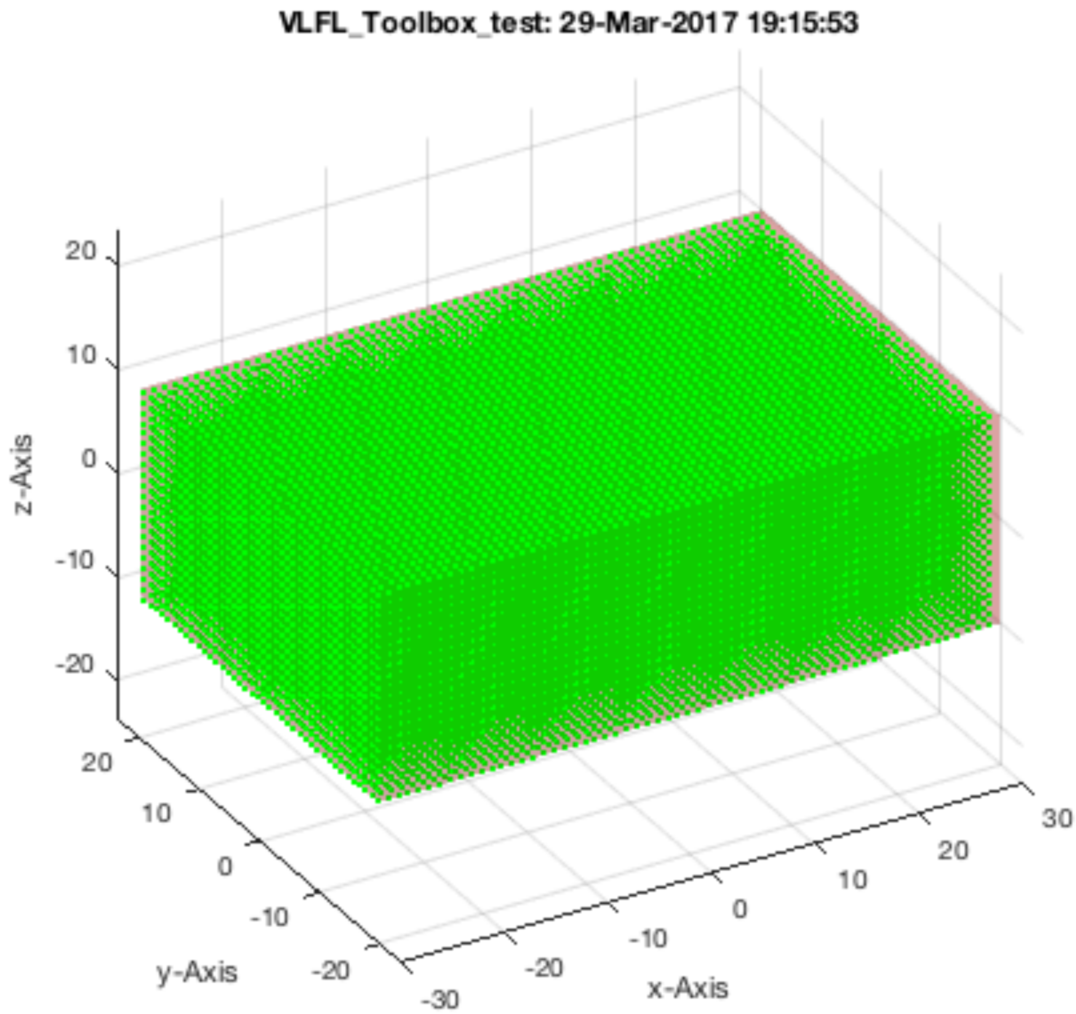
- SGplatesofSGML(SG,thickness) - convert a solid into a plate structure
- SGweight (SG,sepecific weight,resolution) - slowly calculates the weight

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

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



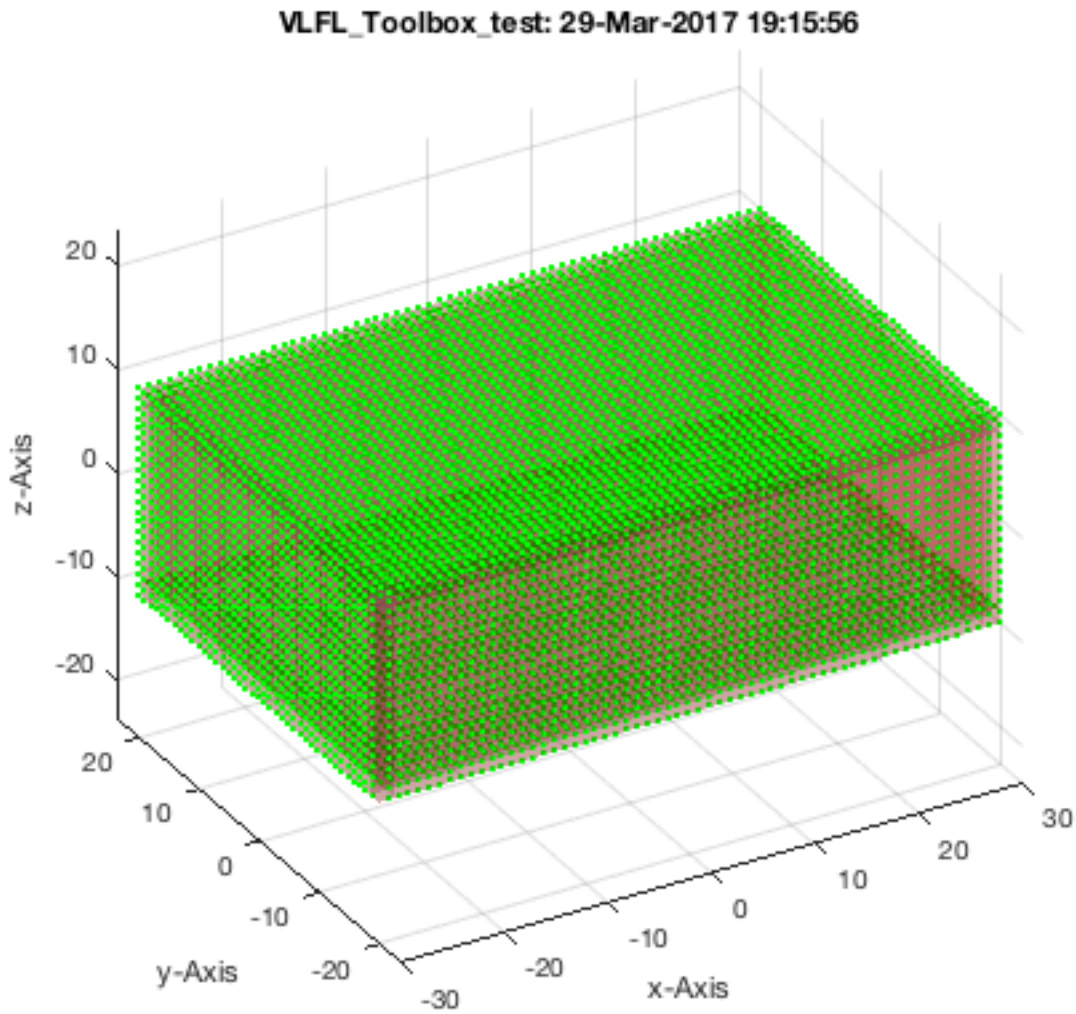

```
SGweight(SGbox([60,40,20]),[],1);
```

```
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 3.004384 seconds.
```




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

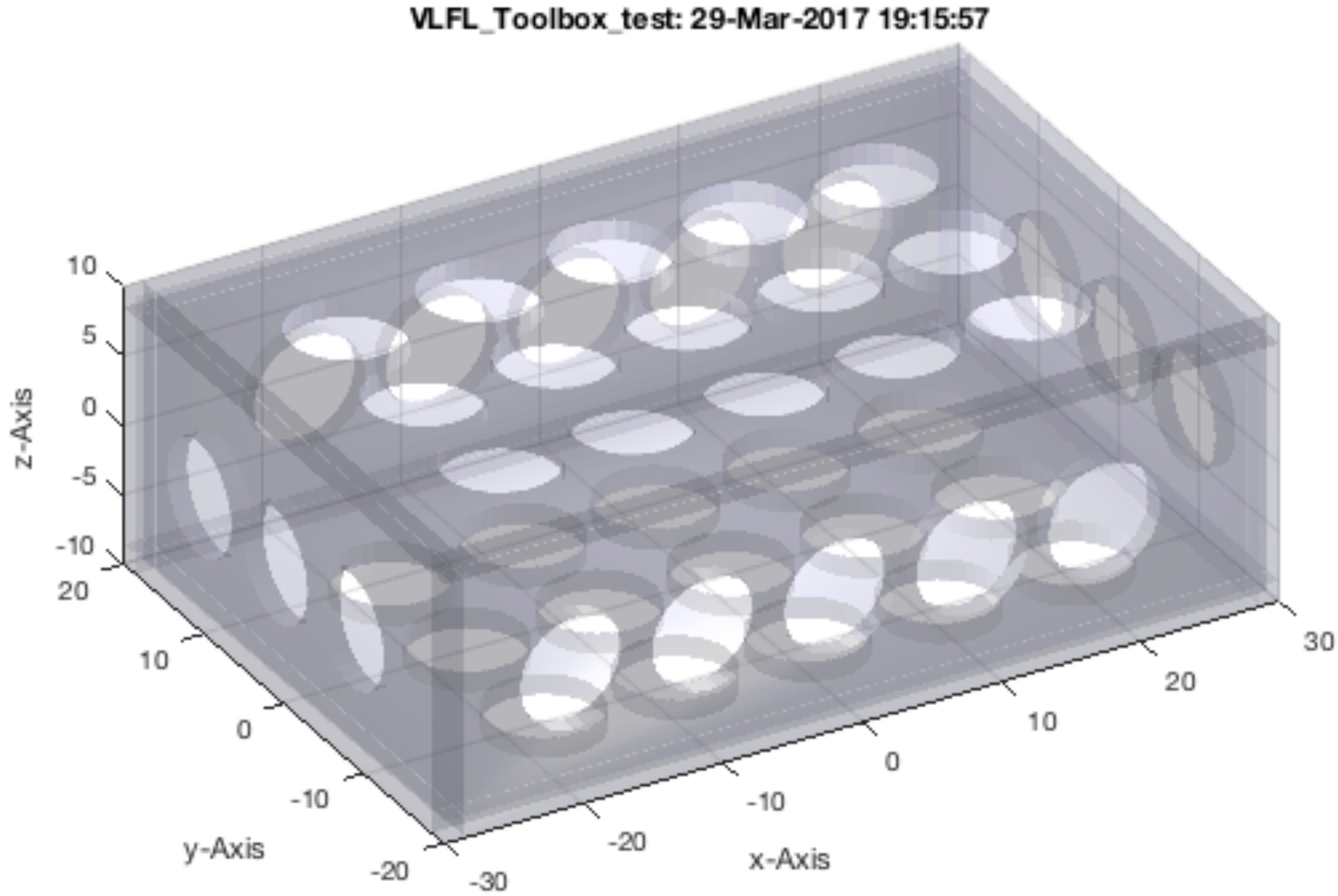
```
Using a resolution of 1.0 mm^3 (n=62307) and a specific weight of 1.15 milligramm per mm^3,
 the overall weight is ca. 11 gramm.
Elapsed time is 3.339398 seconds.
```



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

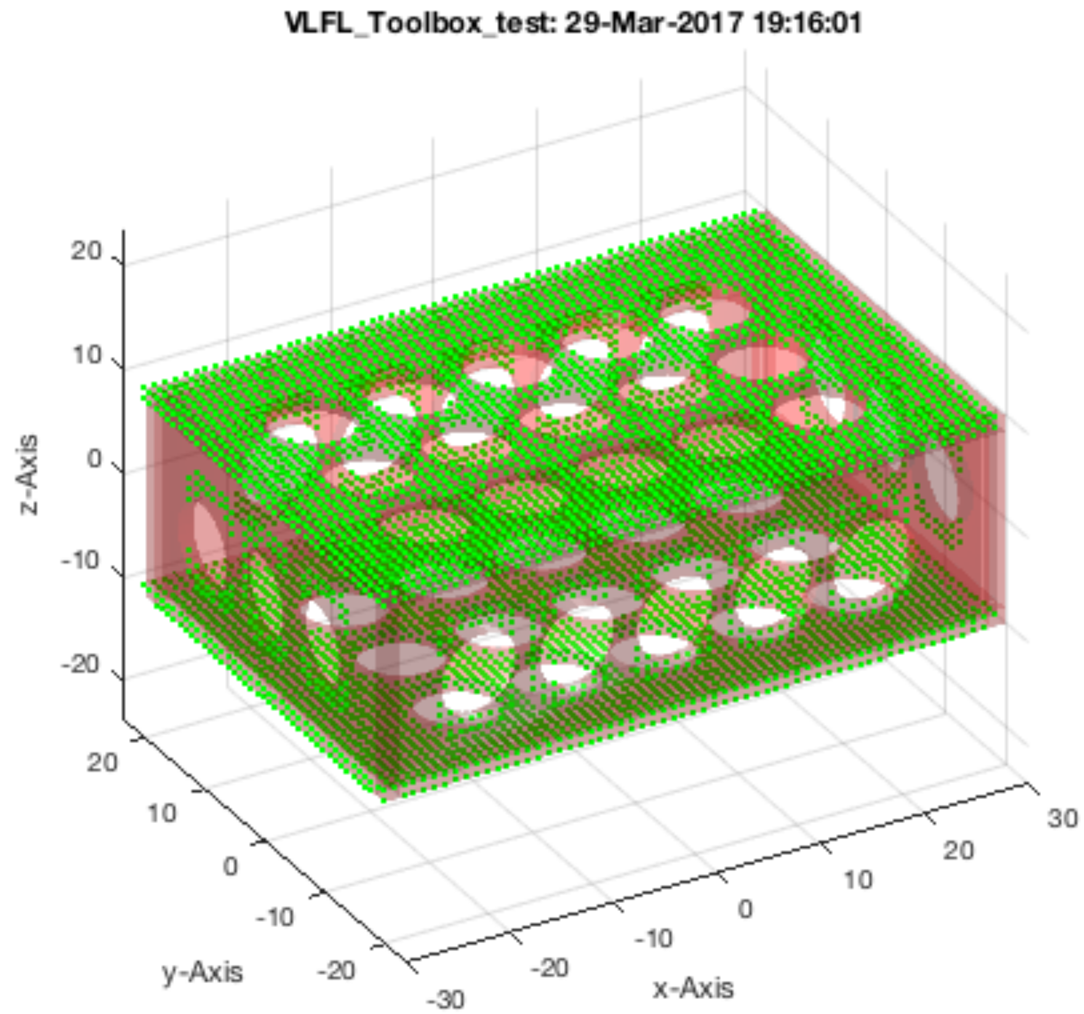
- SGplatesofSGML(SG,thickness,CPL) - convert a solid into a punched plate structure

```
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. 9 gramm.
Elapsed time is 3.312505 seconds.
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:16:03 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-13 we've learned about:

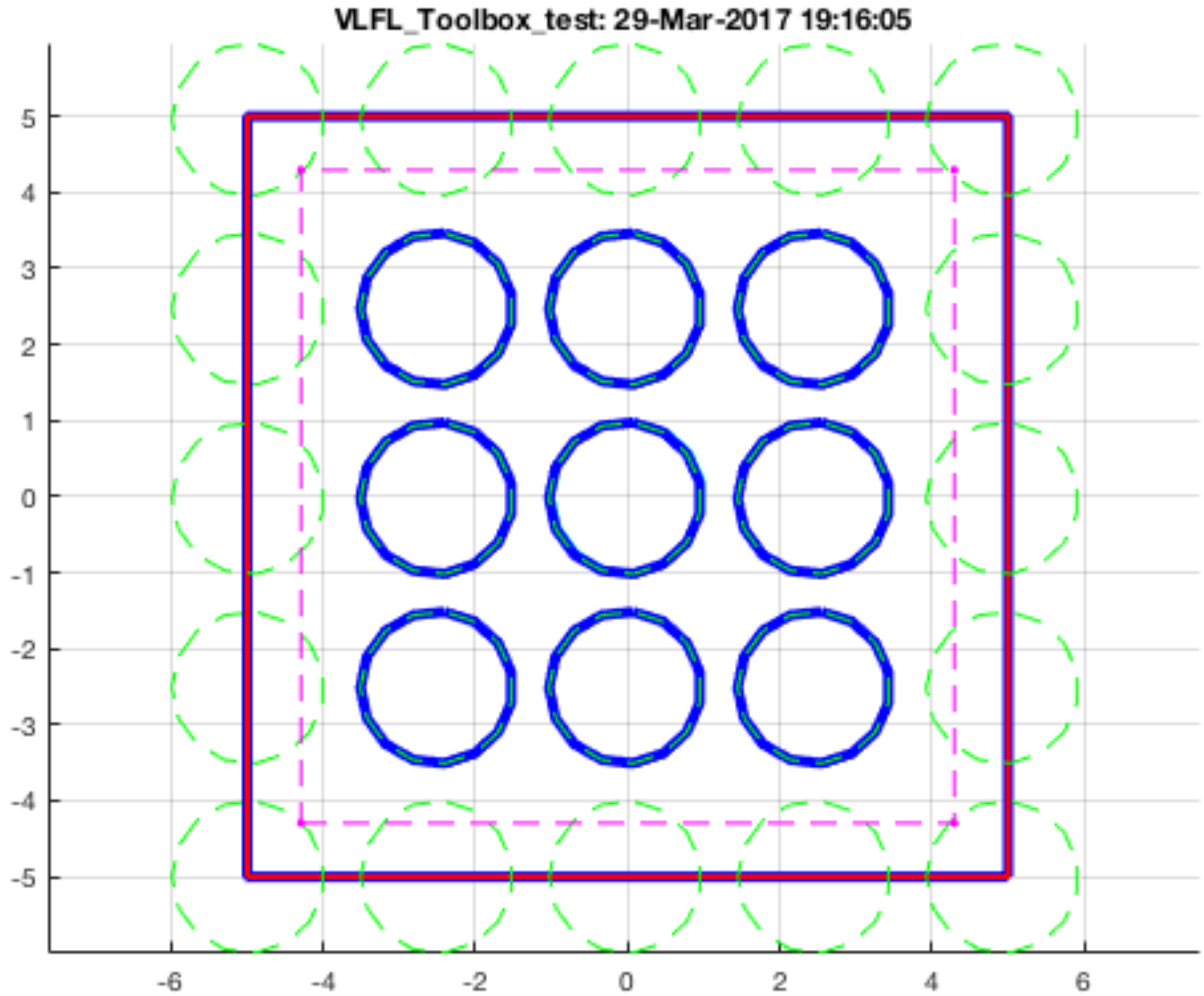
- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Functions to import, check, and export of *STL-File* data
- Functions for generating text strings as solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to *Solid Geometries* (SG)
- Spatial transformation and merging of solid geometry sets
- Relative positioning of solids geometries and boolean operations on solids
- Packaging and boxing of several parts into a complete assemble kit
- Attaching coordinates frames and the creation of kinematik models
- Function for mounting faces of solids and creation of lightweight-solids

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:

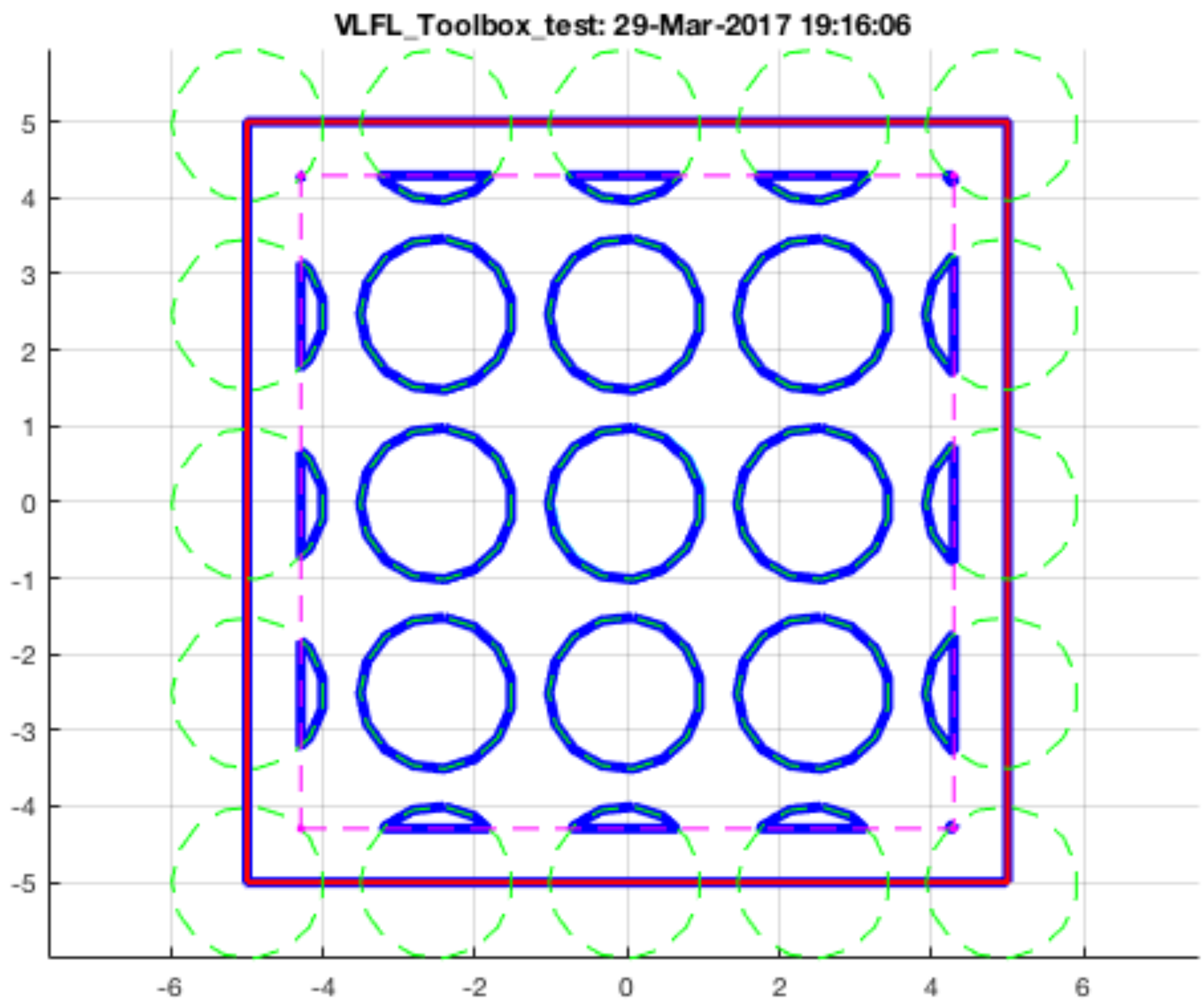
- **CPLfillPattern(CPLA, CPLB,w,d)** - fills a contour CPLA with copies of the pattern CPLB with a distance to the outer contour w and distance between the patterns of d

```
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);
```



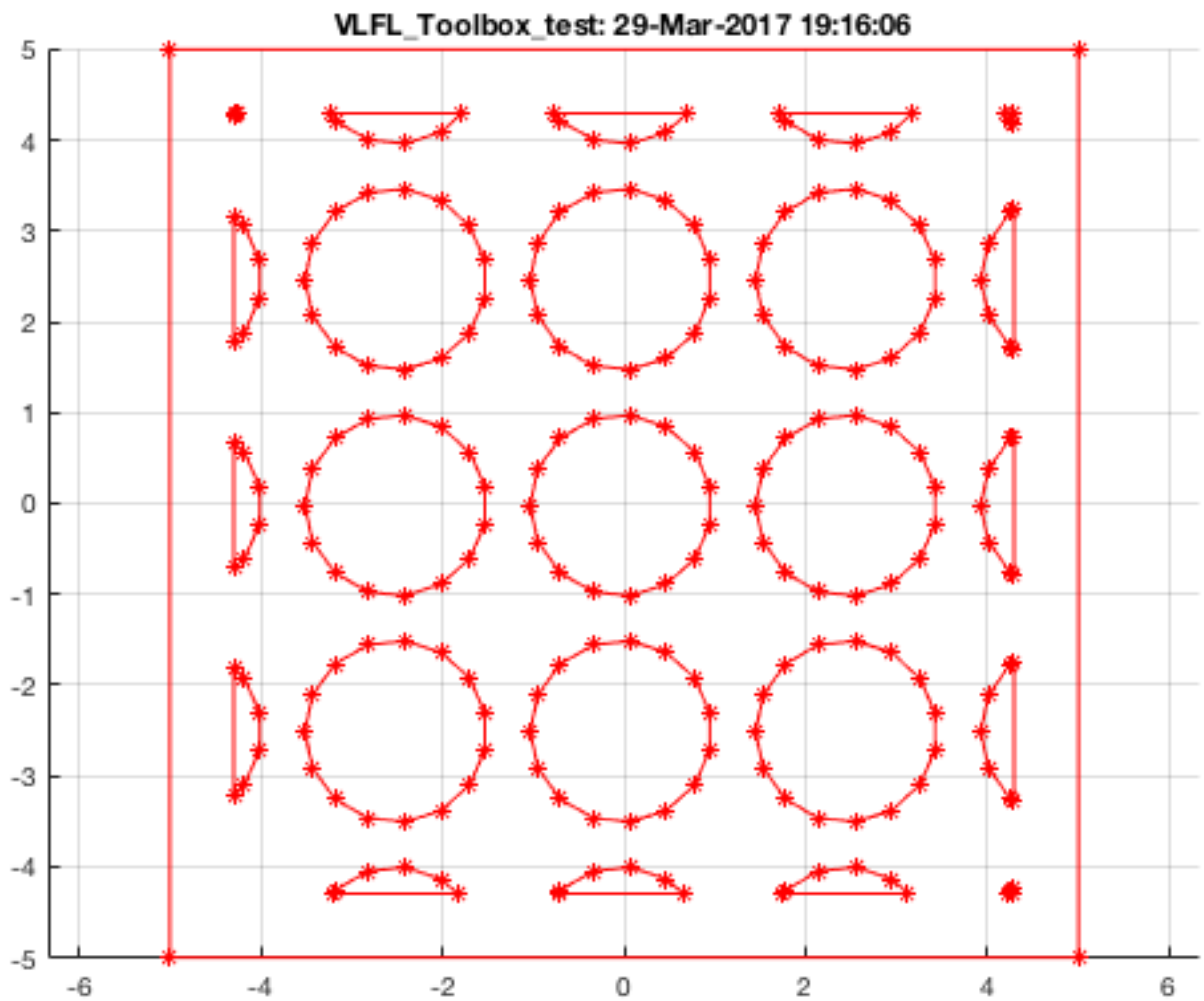
3. Writing a contour as SVG-File for laser-cutting

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

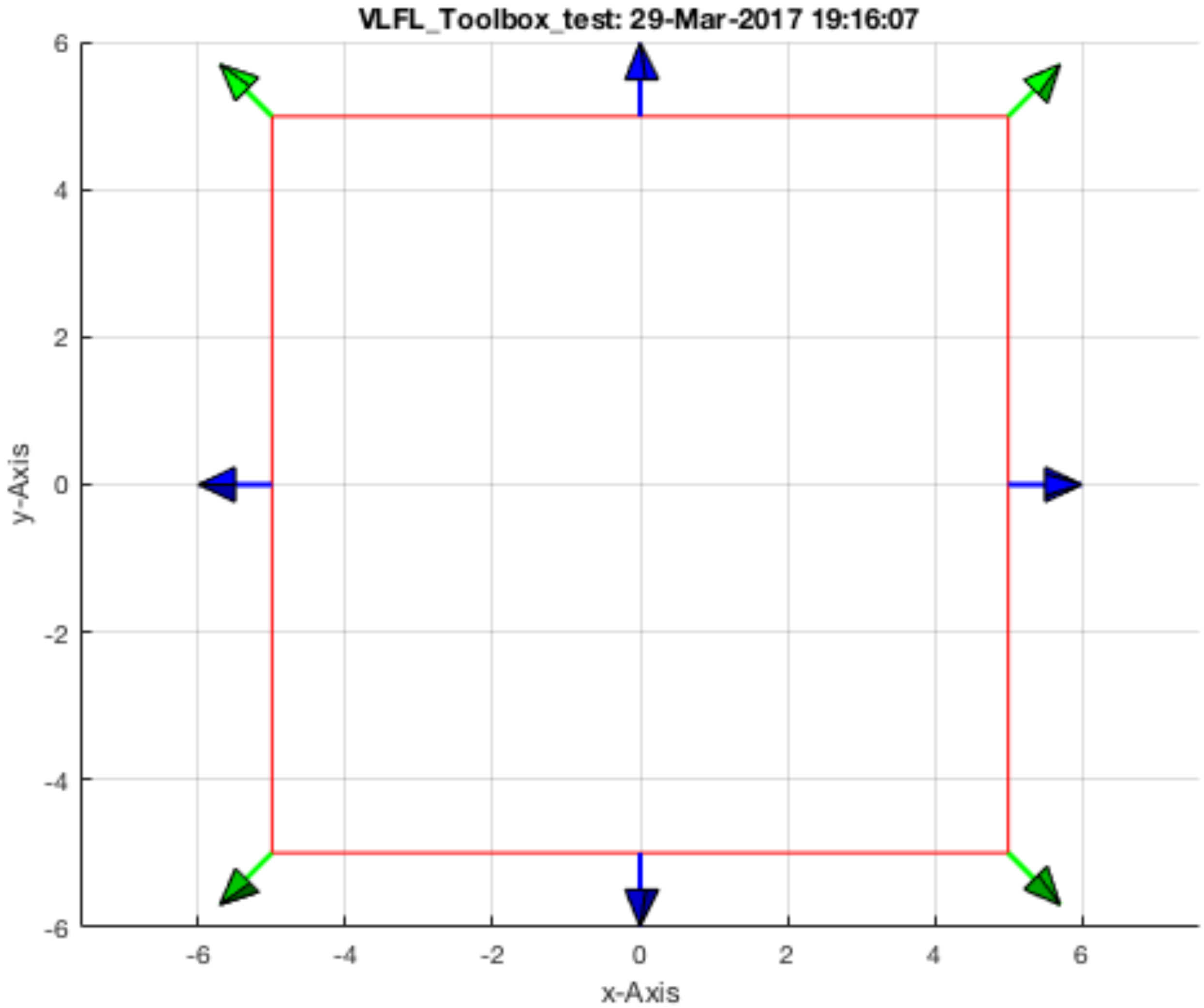
```
SGfigure; view(0,90); axis on;
A=CPLfillPattern(PLsquare(10,10),PLcircle(1),1,[],true);
CPLplot(A);
CPLwriteSVG(A,'VLFL_EXP14');
```

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



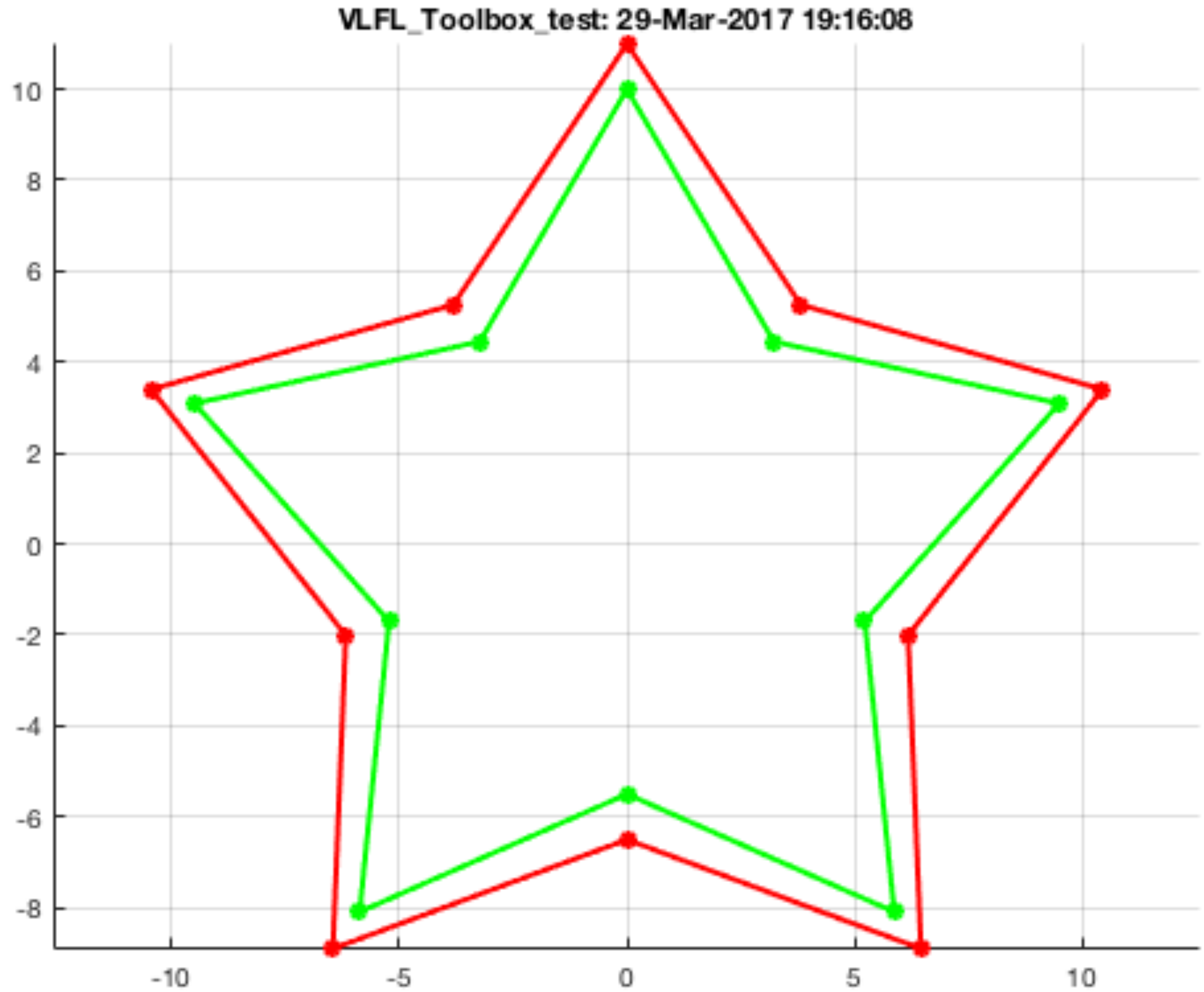
4. Calculating the normal vectors of edges and points

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



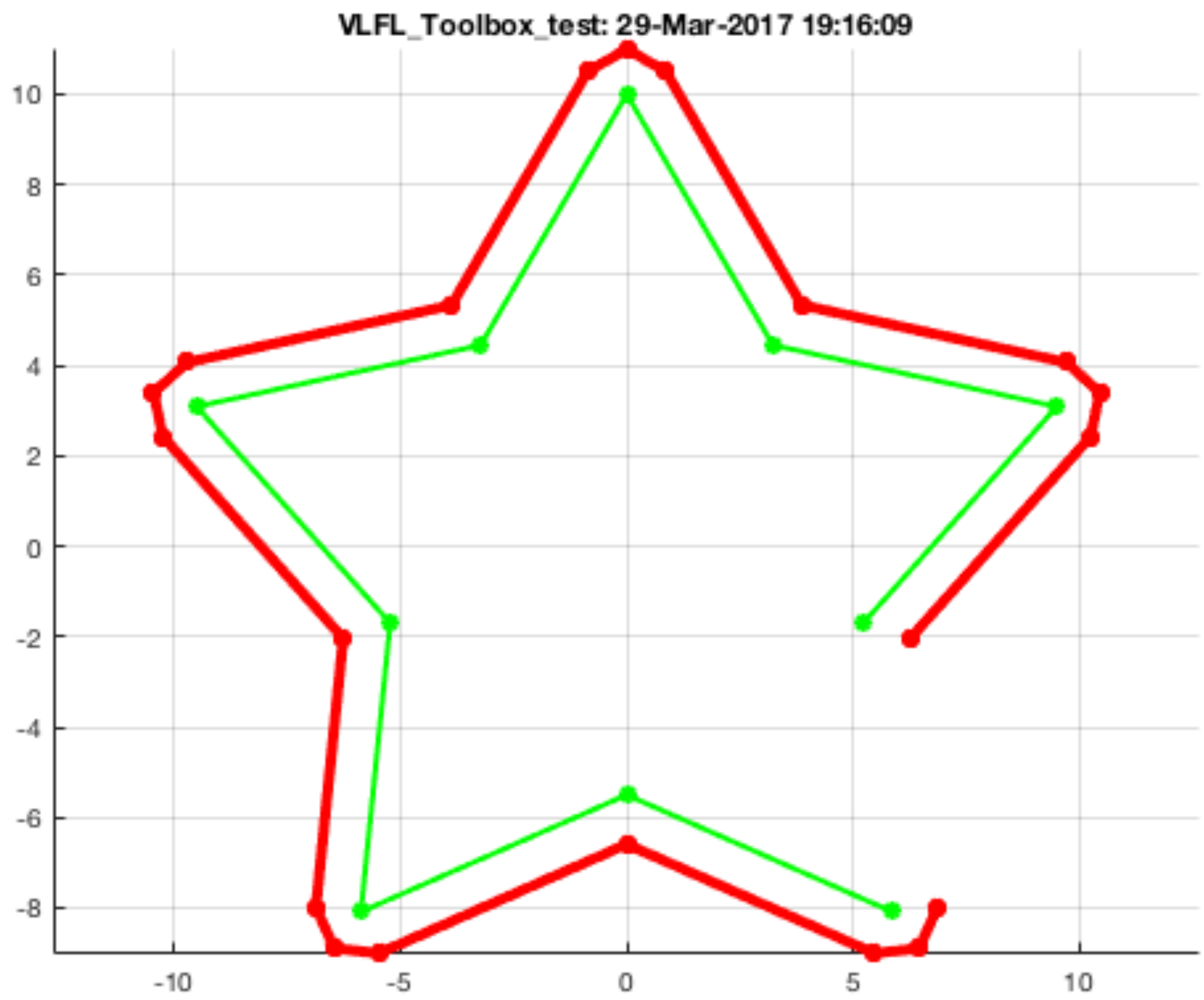
5. Growing with same number of points

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



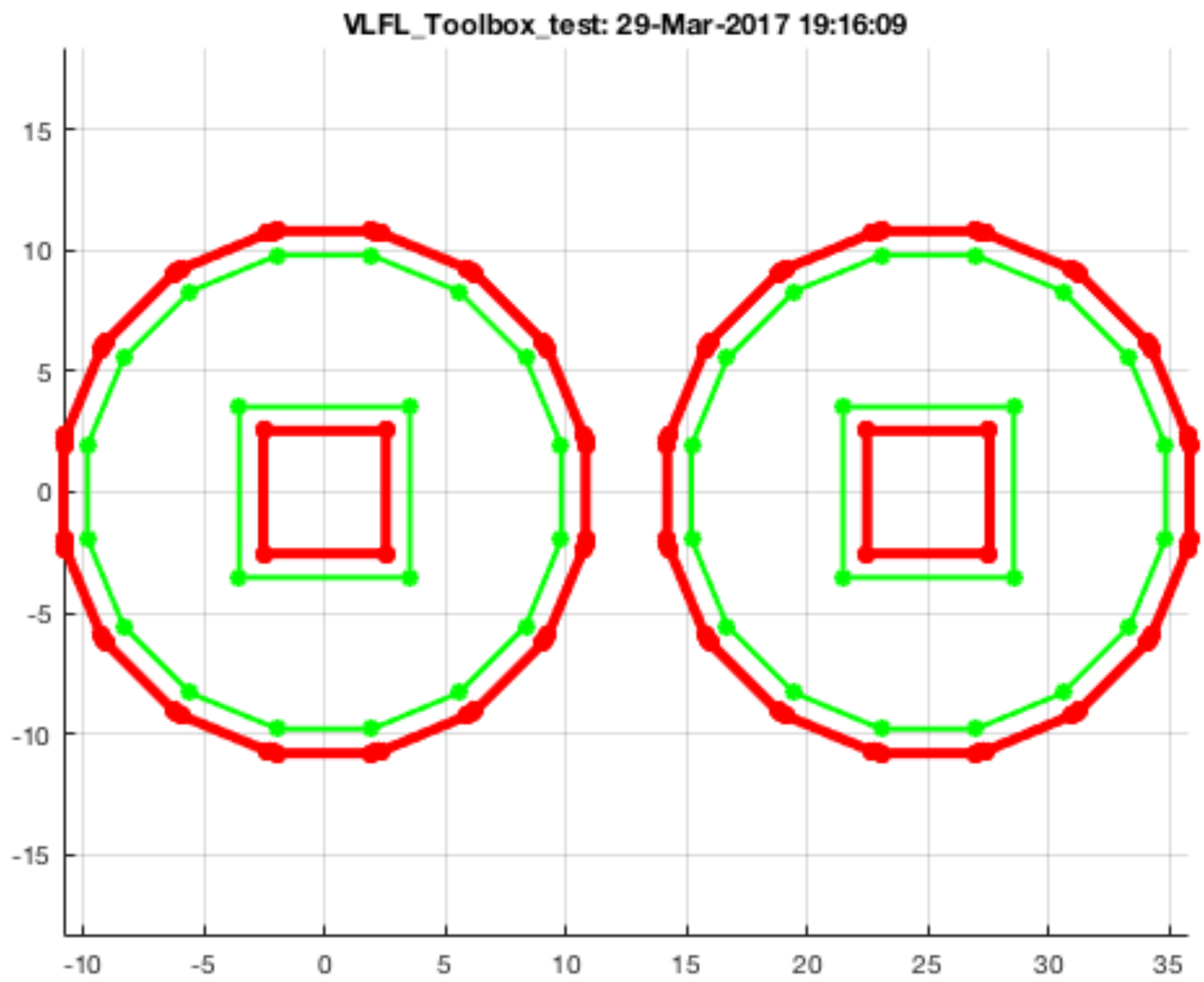
6. Growing with correct distance to edges

```
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;
```

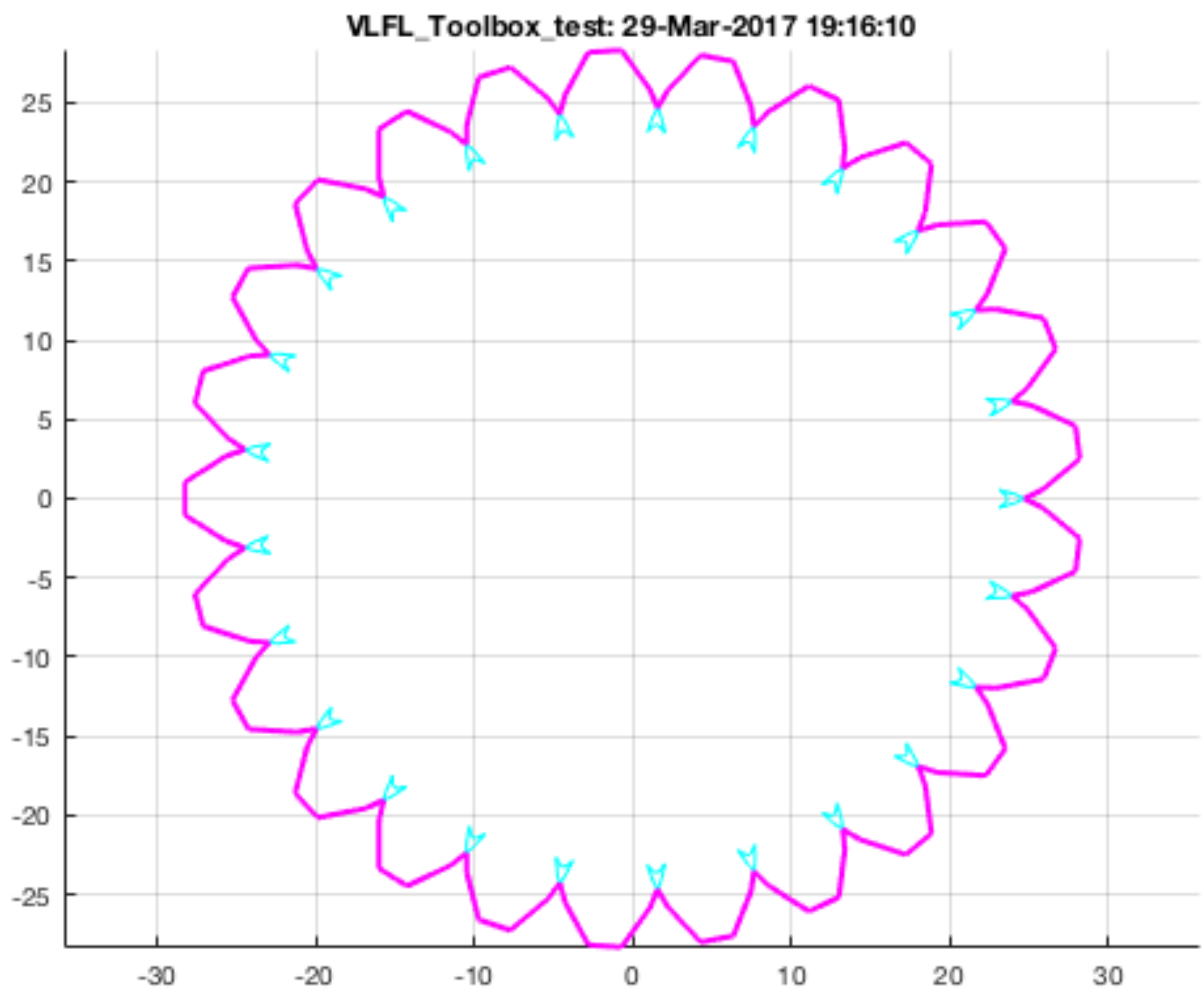


Growing may have no problems

```
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
```

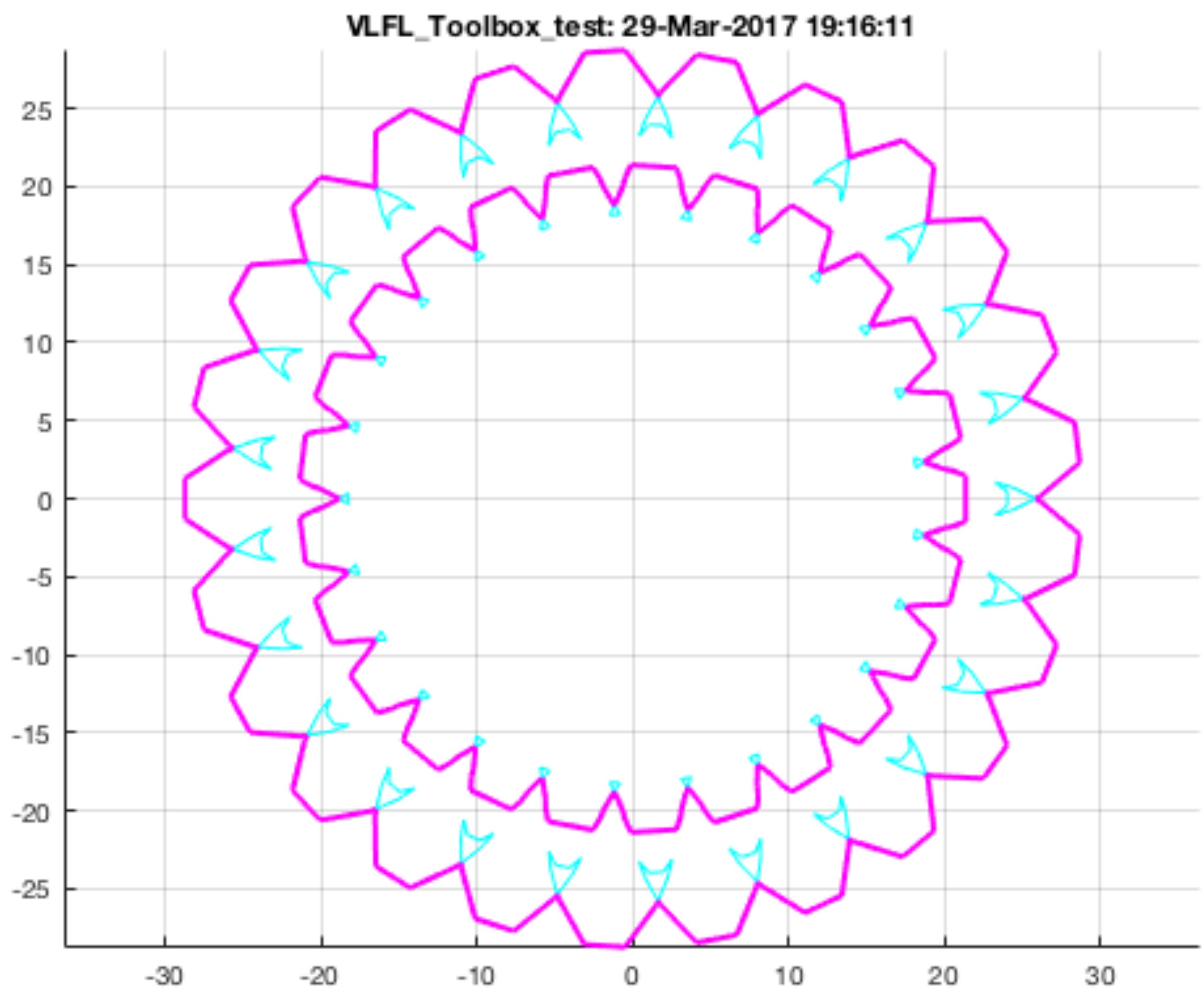


Another example using CPLoutercontour

```
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
```

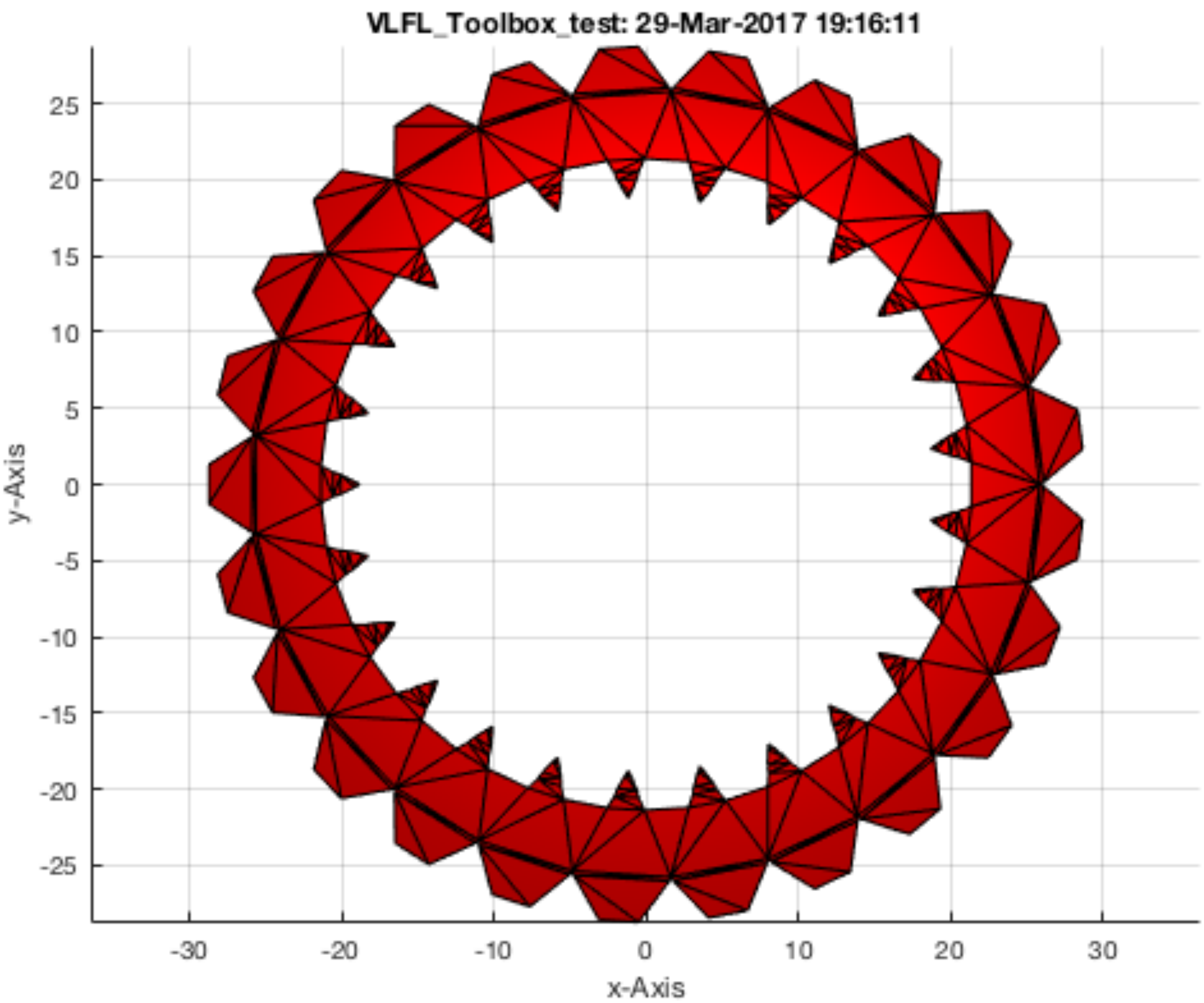
```
Tooth head width: 0.56
Difference tooth gap and tooth head width: 1.14
Modul: m=1.50
Teeth Nr: z=25
Rolling circle: d=37.50
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=41.25
Head diameter: dh=34.50
Tooth foot width: 3.63
Involute angle: 34.09
Tooth head width: 0.25
Difference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

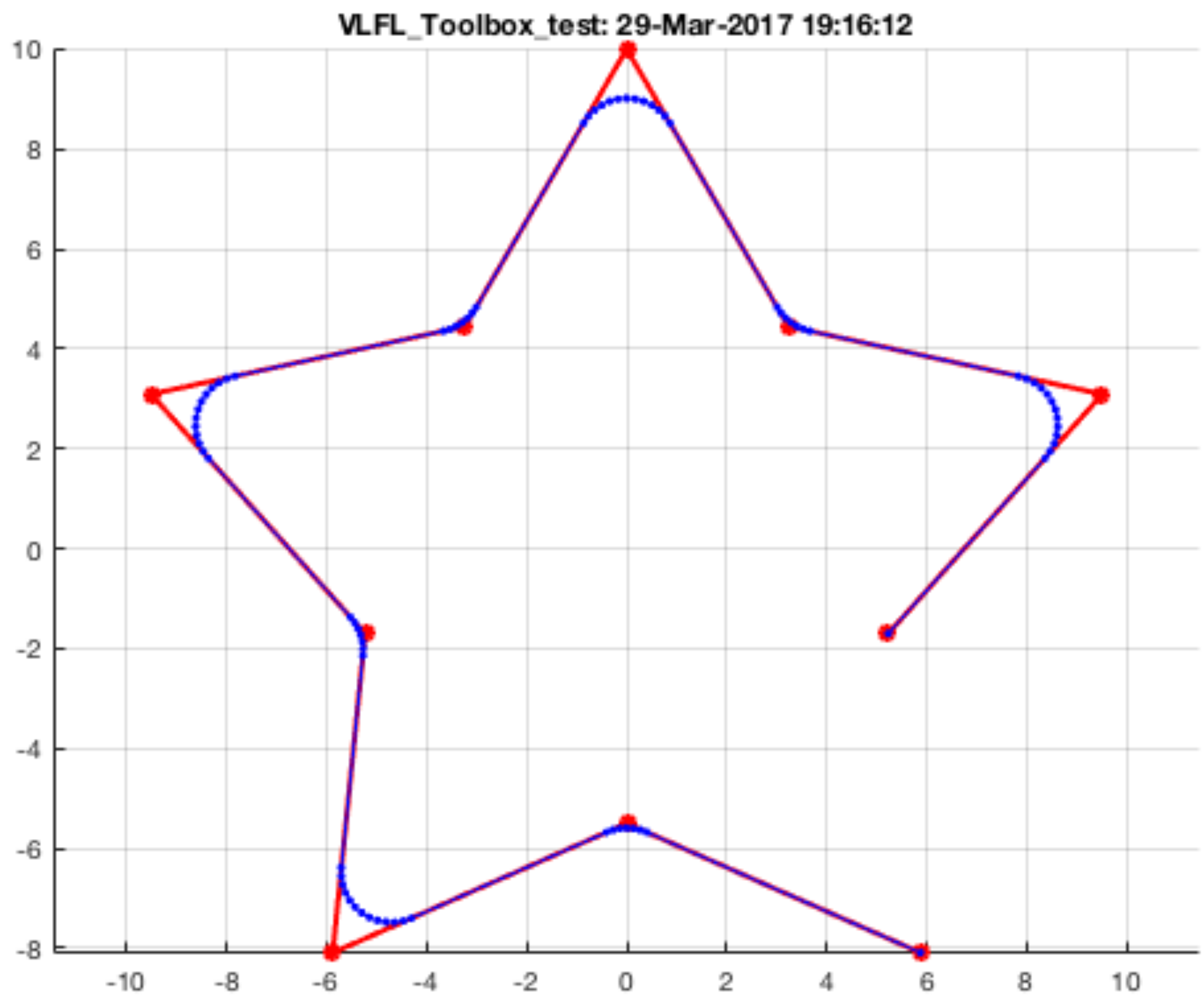
```
Foot ratio: rsp=0.70
Tooth size factor: x=1.00
Foot diameter: df=41.25
Head diameter: dh=34.50
Tooth foot width: 3.63
Involute angle: 34.09
Tooth head width: 0.25
Difference tooth gap and tooth head width: 1.30
```



7. Rounded edges inside a contour

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

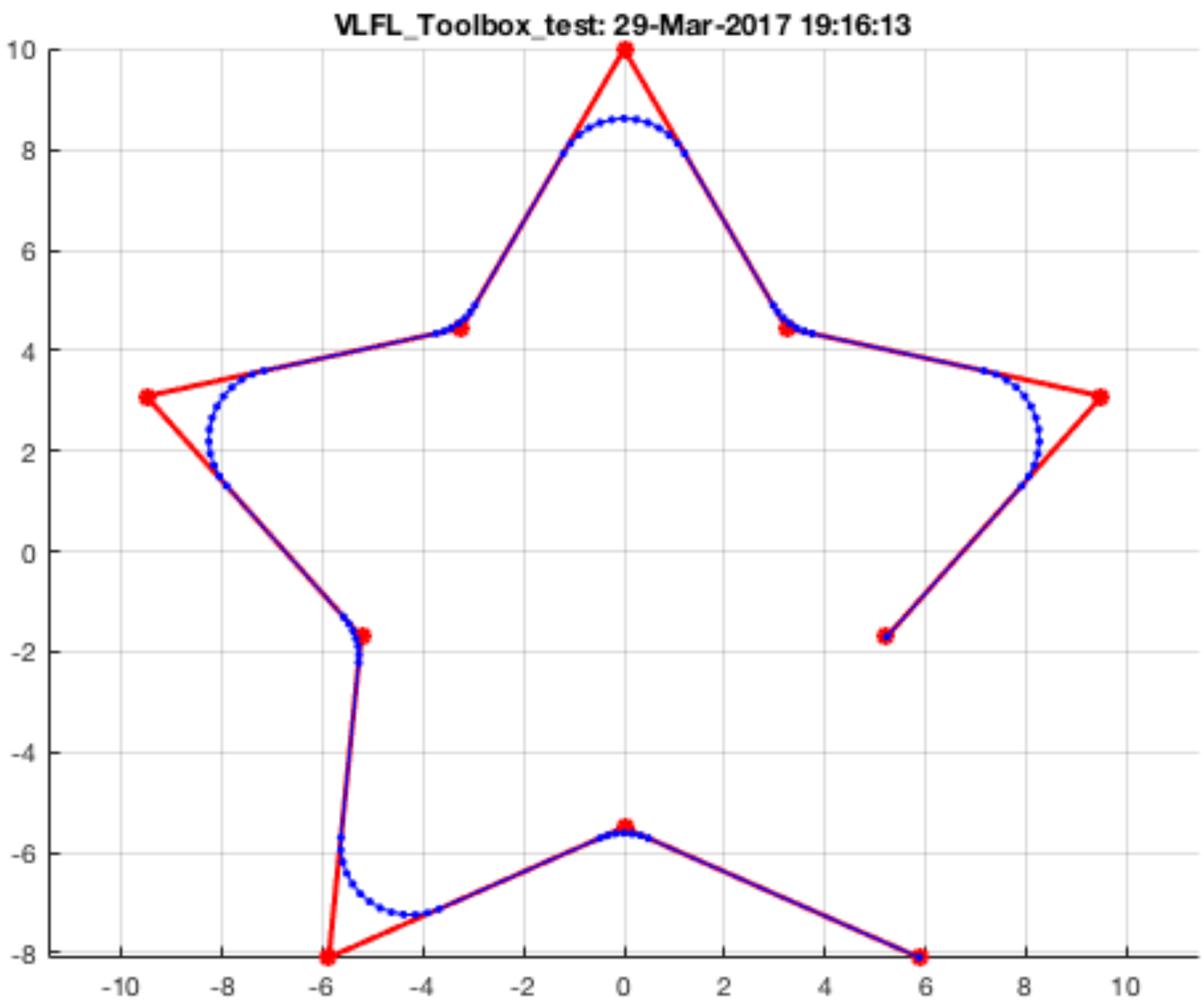
```
SGfigure; view(0,90);
PLradialEdges(PLstar(10,10));axis on;
```



Another example using radius=2

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

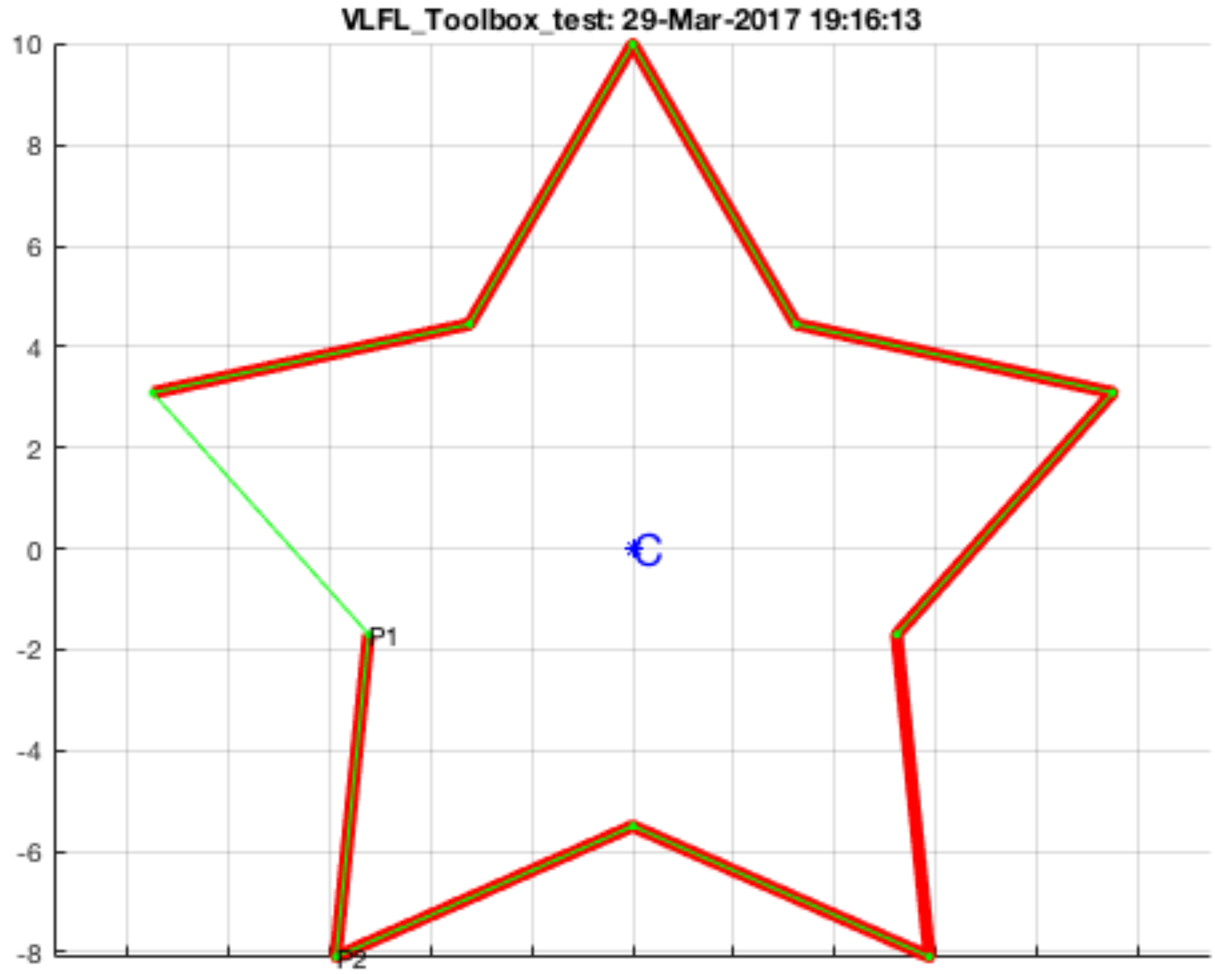
```
Radial egde radius reduced R=2 to 1.4 at [9.5 3.1] w=119.6
Radial egde radius reduced R=2 to 1.2 at [3.2 4.4] w=-47.6
Radial egde radius reduced R=2 to 1.4 at [0.0 10.0] w=119.6
Radial egde radius reduced R=2 to 1.2 at [-3.2 4.4] w=-47.6
Radial egde radius reduced R=2 to 1.4 at [-9.5 3.1] w=119.6
Radial egde radius reduced R=2 to 1.2 at [-5.2 -1.7] w=-47.6
Radial egde radius reduced R=2 to 1.4 at [-5.9 -8.1] w=119.6
Radial egde radius reduced R=2 to 1.2 at [-0.0 -5.5] w=-47.6
```



8. Sort CPLs around its center

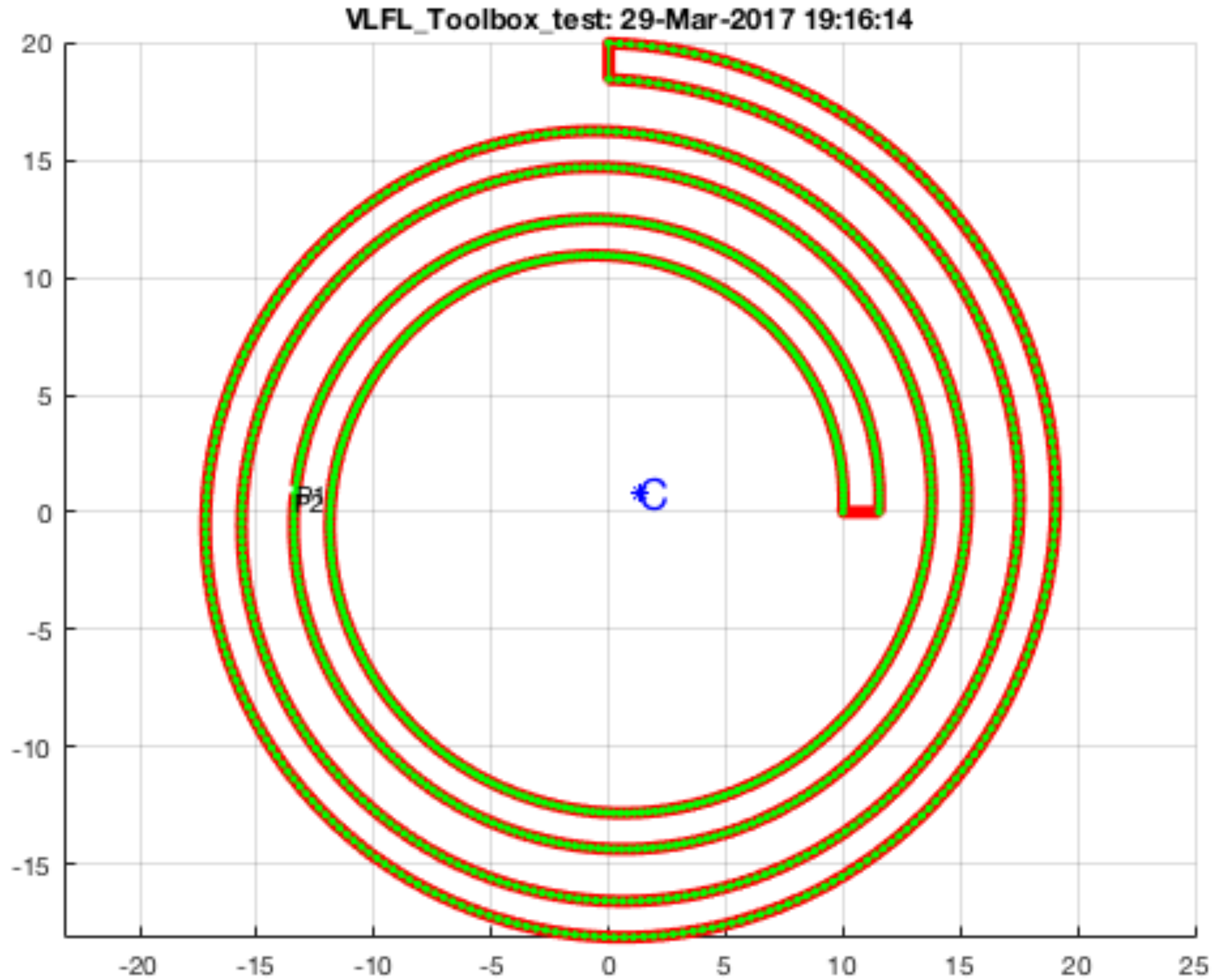
Find the minmal angle value of a star

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



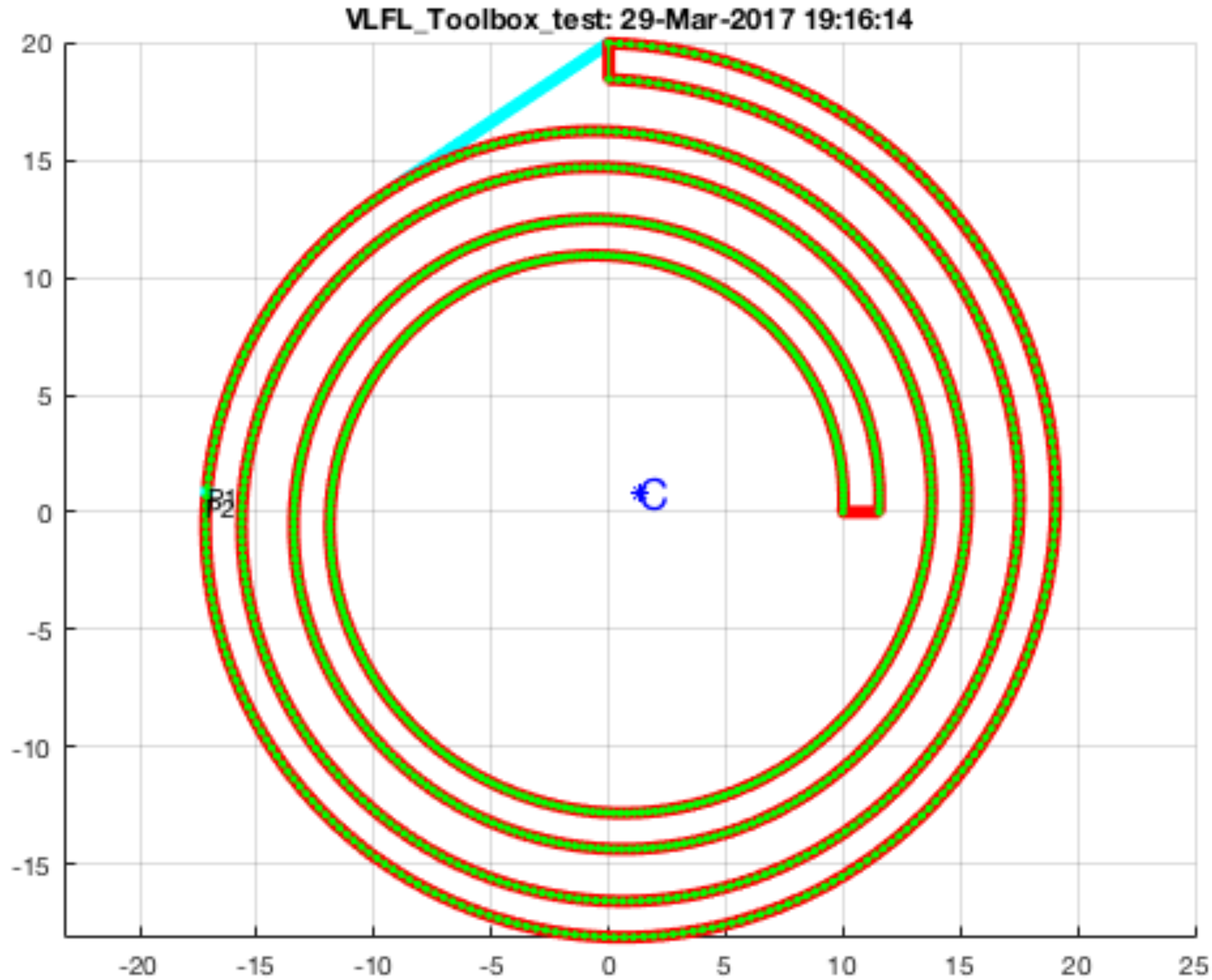
Find the minmal angle value of a spiral

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



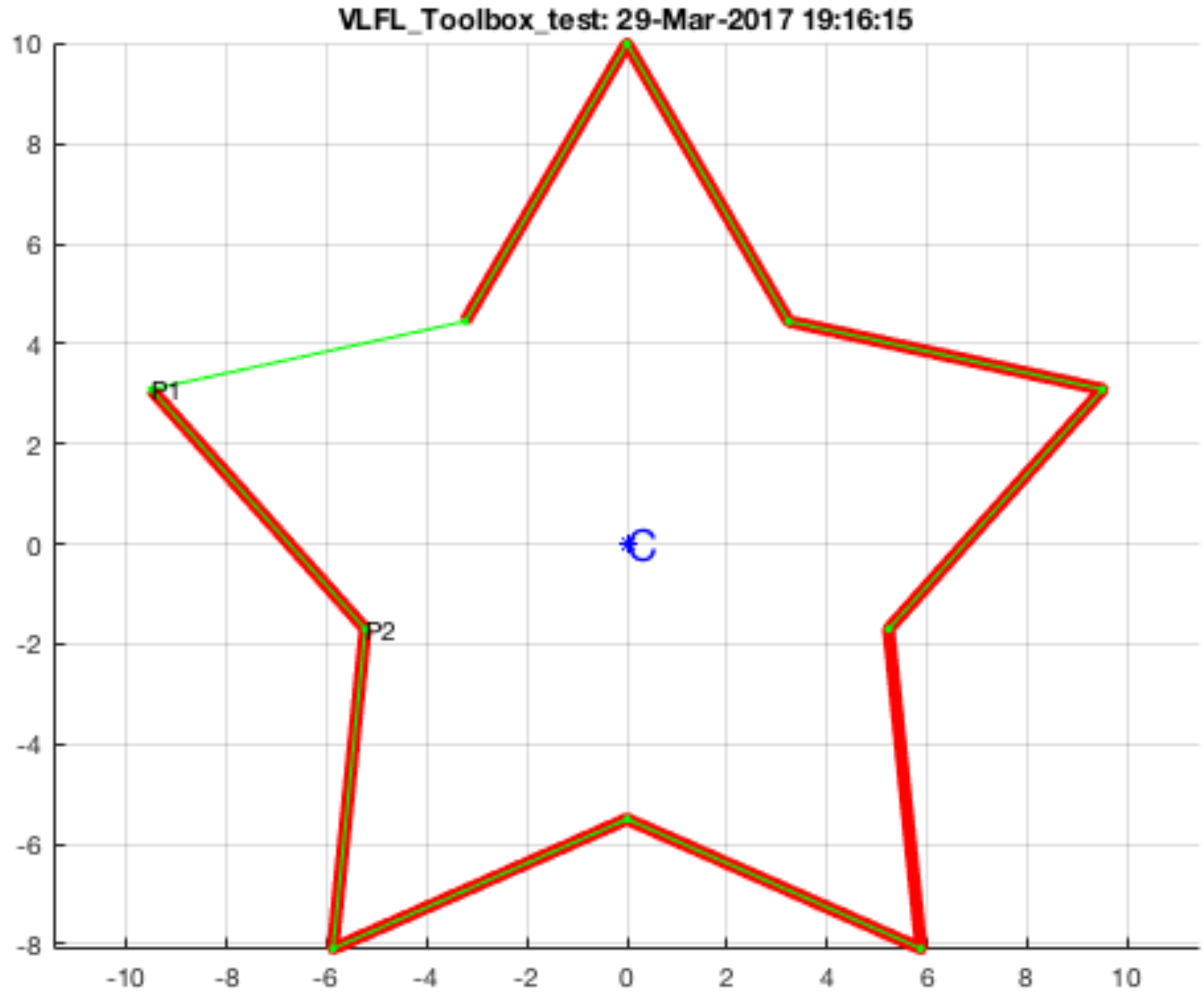
Find the minmal angle value of convex hull of a spiral

```
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;
```

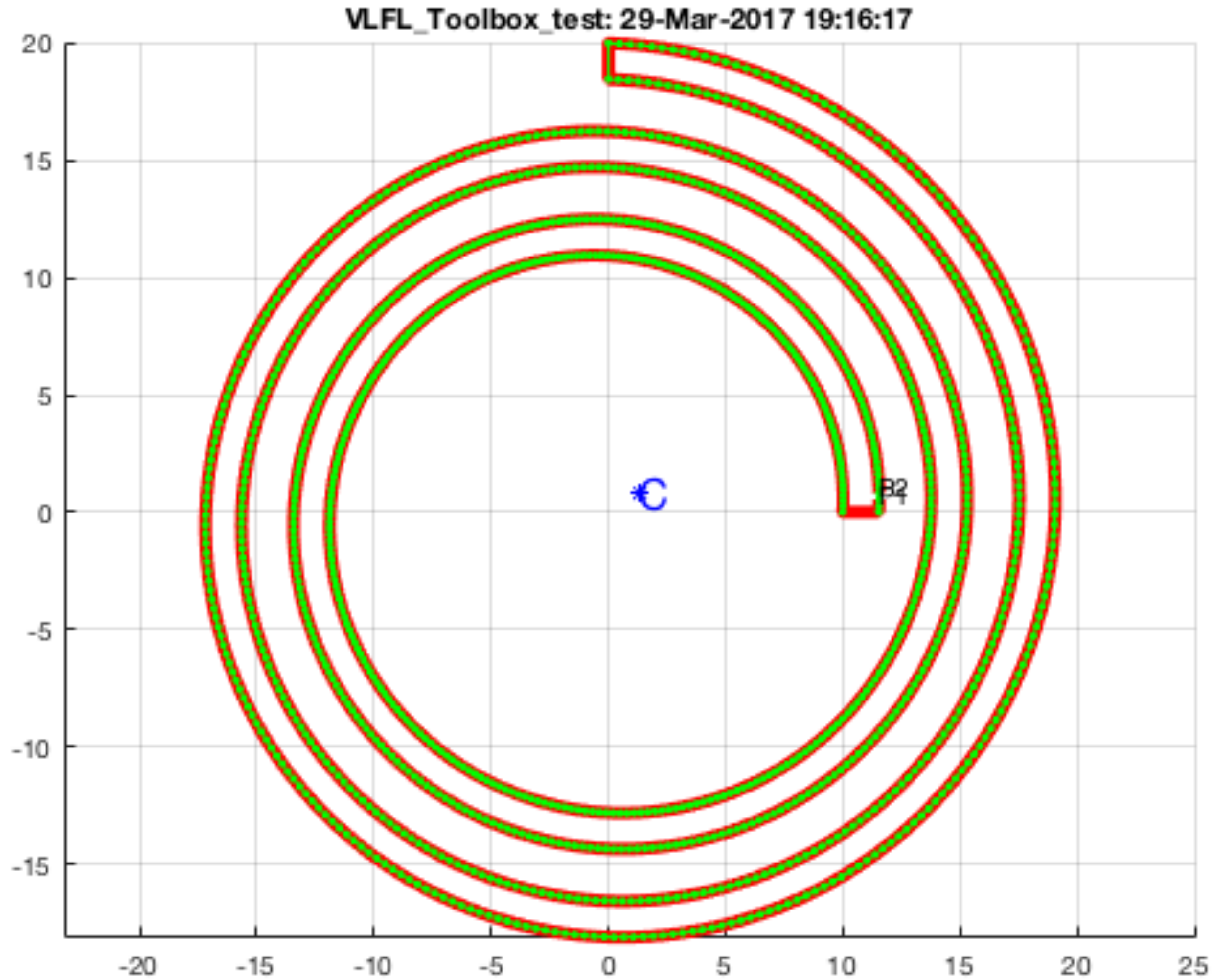


Find the angle value nearest to zero of a star

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

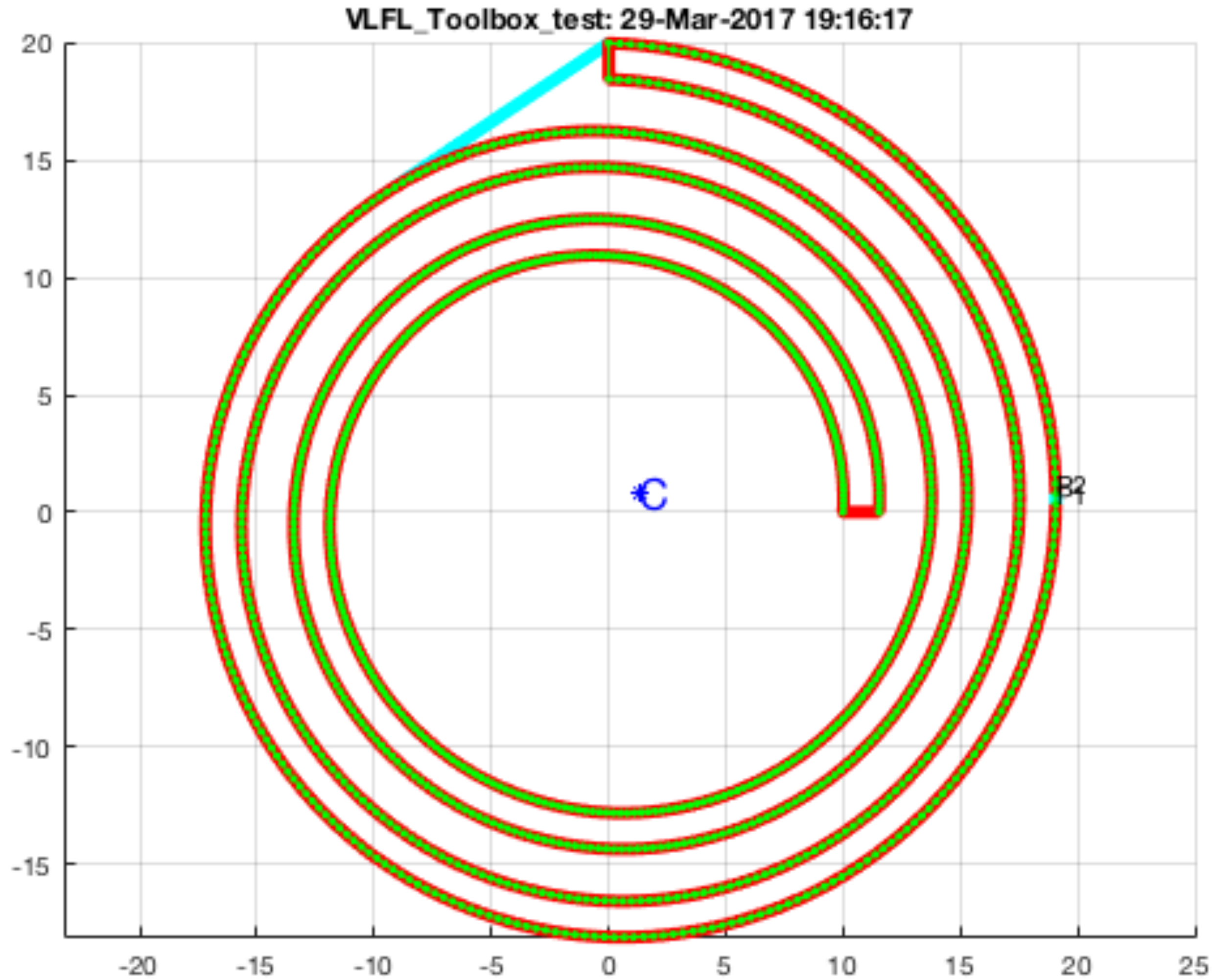
Find the angle value nearest to zero of a spiral

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



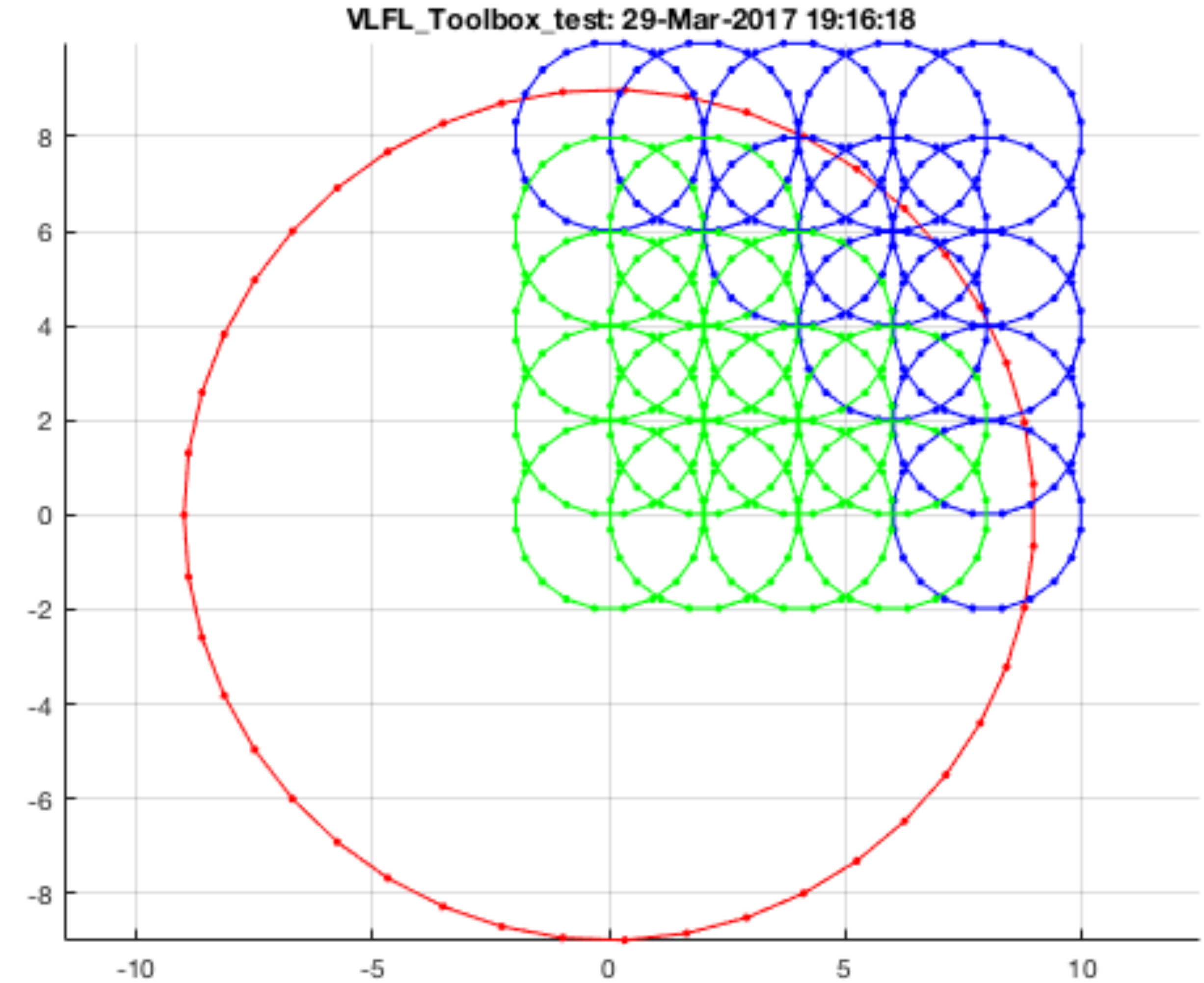
Find the angle value nearest to zero of convex hull of a spiral

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



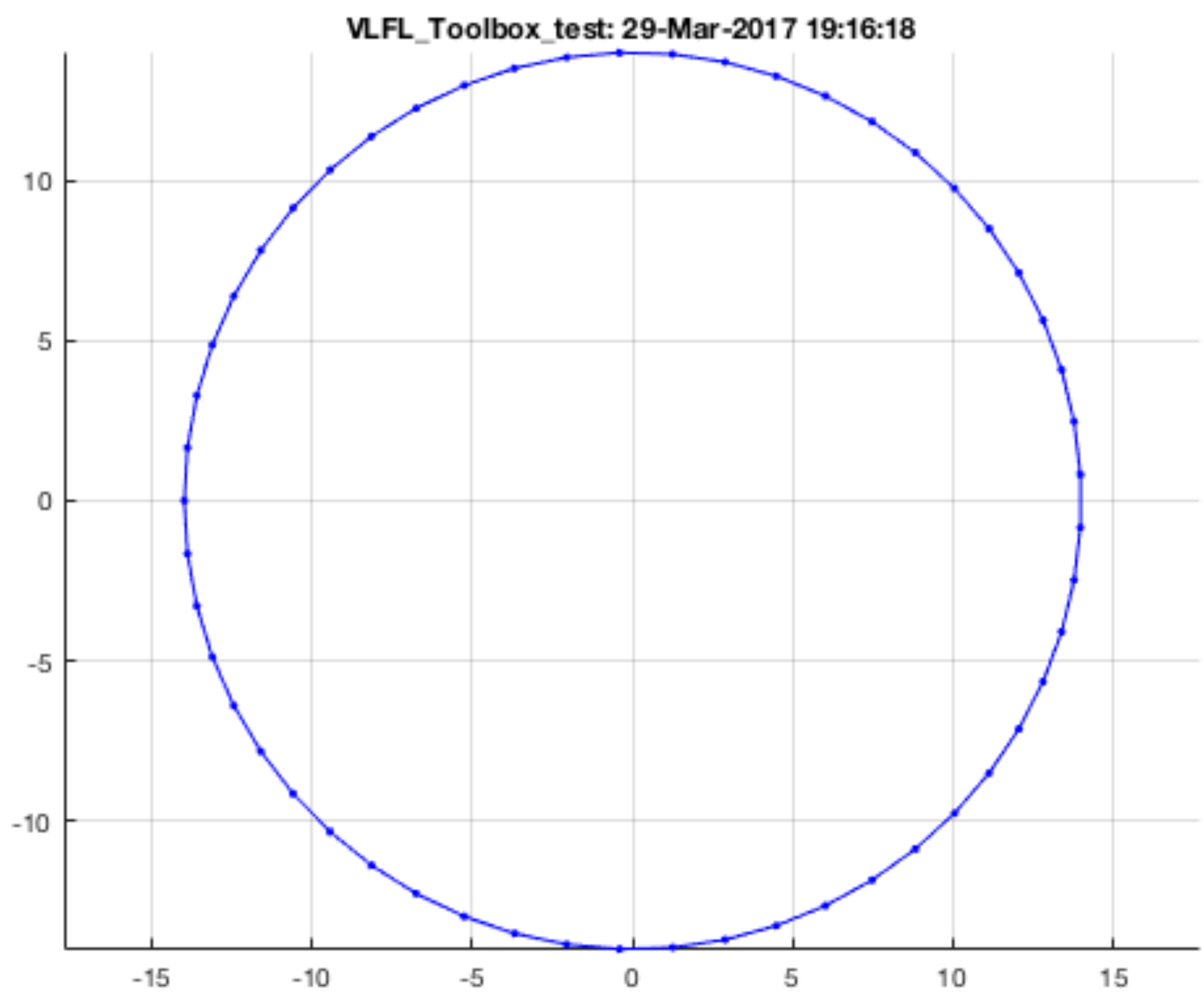
9. Informations on contours inside of others

```
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)));
```



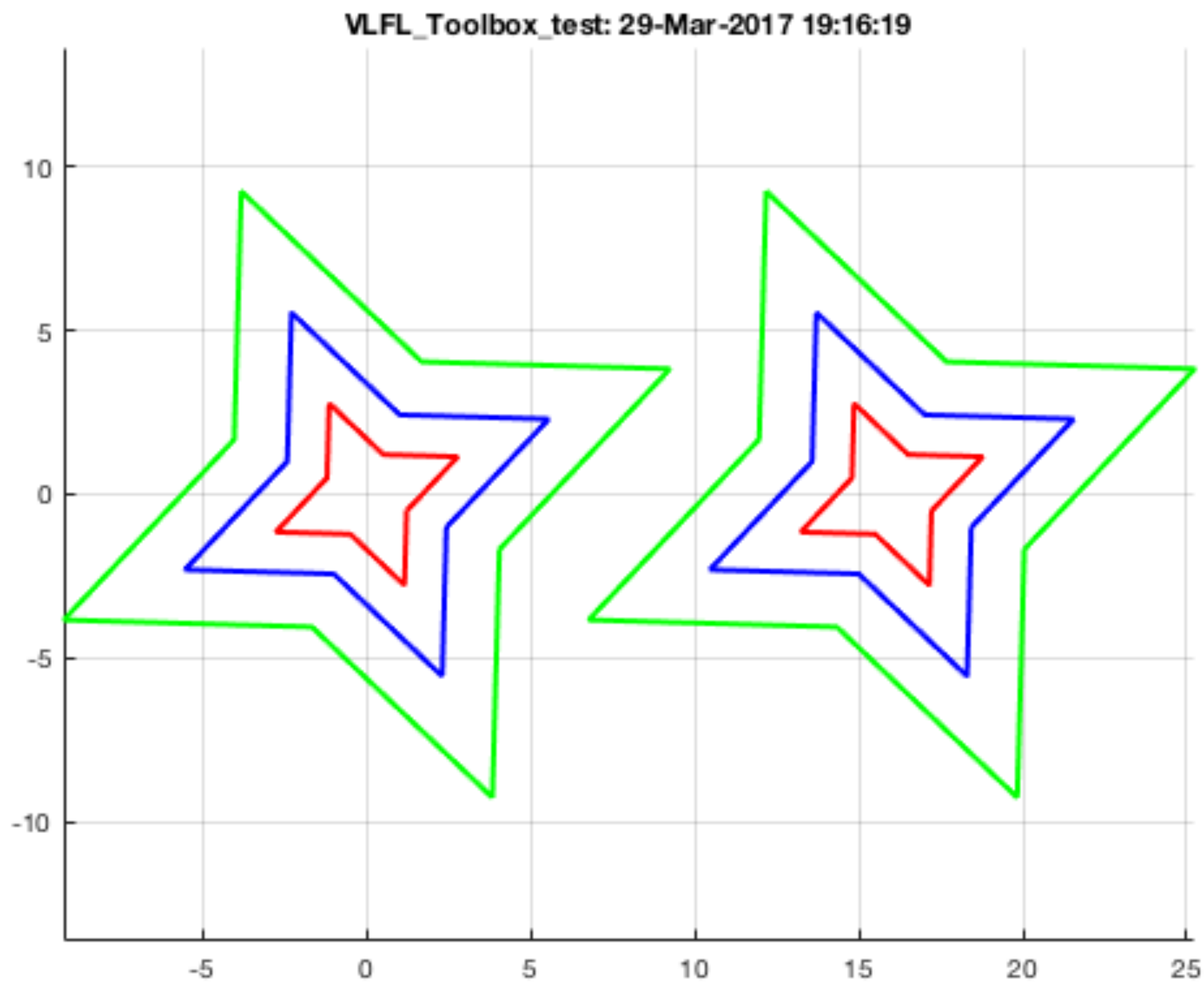
10. Order contours for the sequential plot with a laser cutter

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

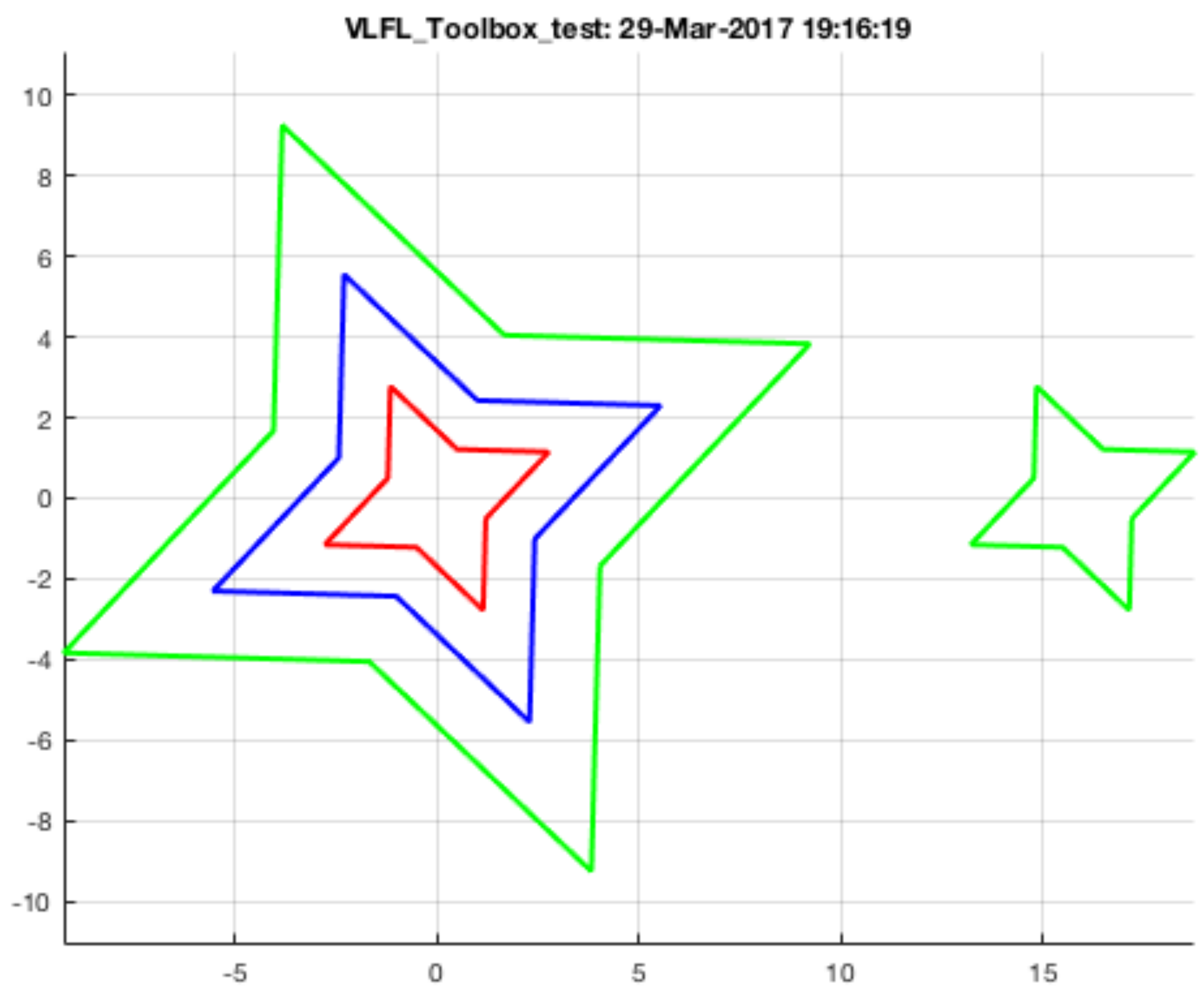
```
SGfigure; view(0,90);
[ci,CC]=CPLsortinout(CPLsample(14))
CPLsortinout(CPLsample(14));
```

```
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
```



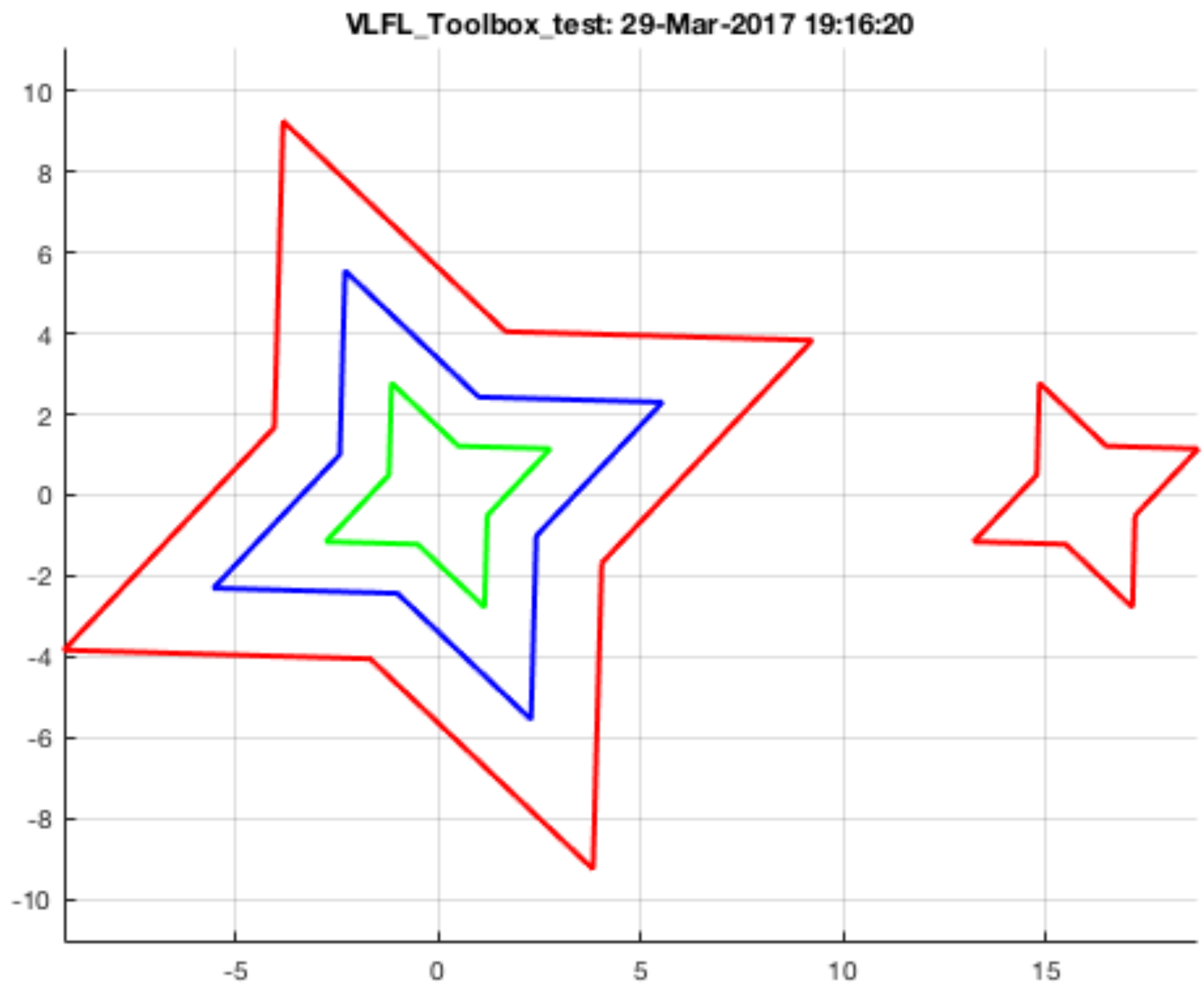
Show a selction from inner to outer for laser cutting

```
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/lueth/Desktop/VLFL_EXP14_cutter.SVG in ASCII MODE
completed.
```

Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:16:22 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 15: Create a Solid by 2 Closed Polygons

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)

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.

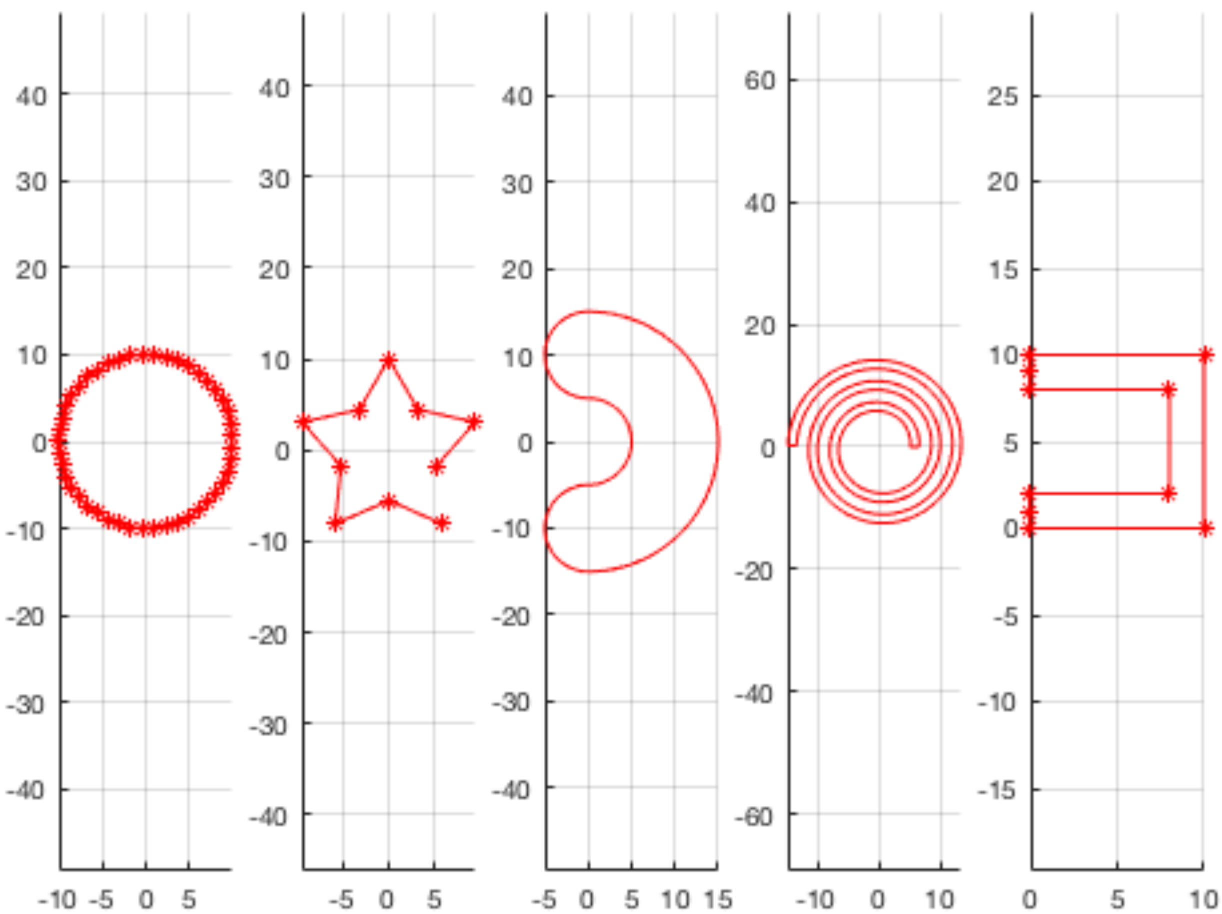
So, for convex polygons there is absolutely no problem by **stepping forward related to the strictly monotonic increasing angle sum**, after **finding a suitable start point** of both polygons (which is a challege itself).

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;
```



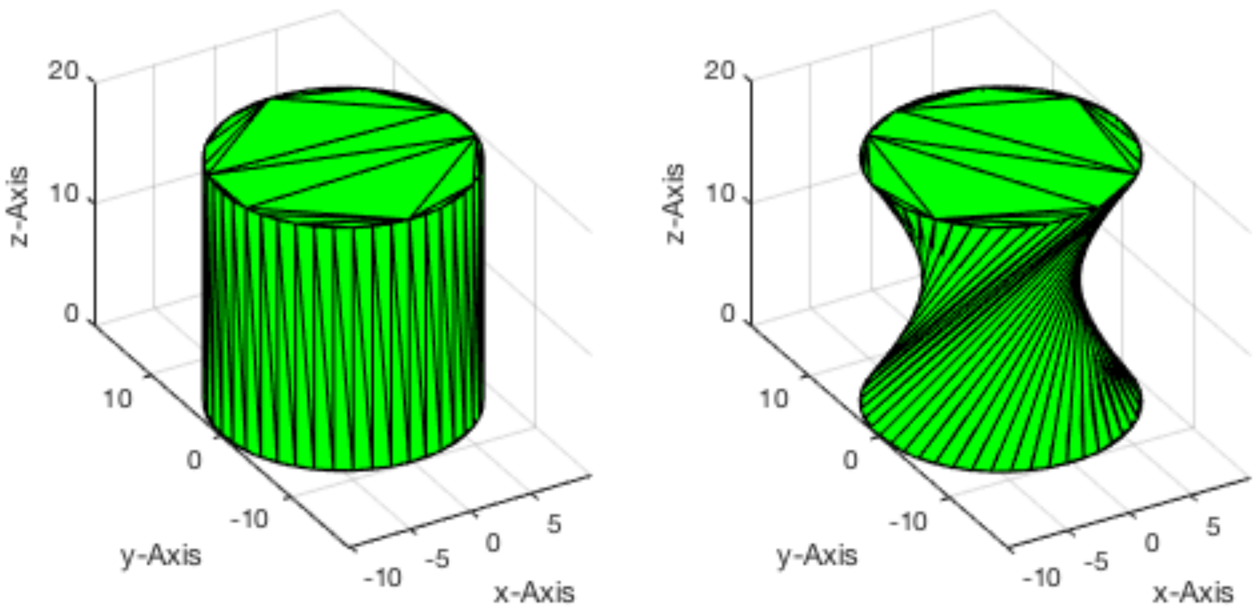
3. Solid surface generation and the importance of start point of the contour (turning)

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

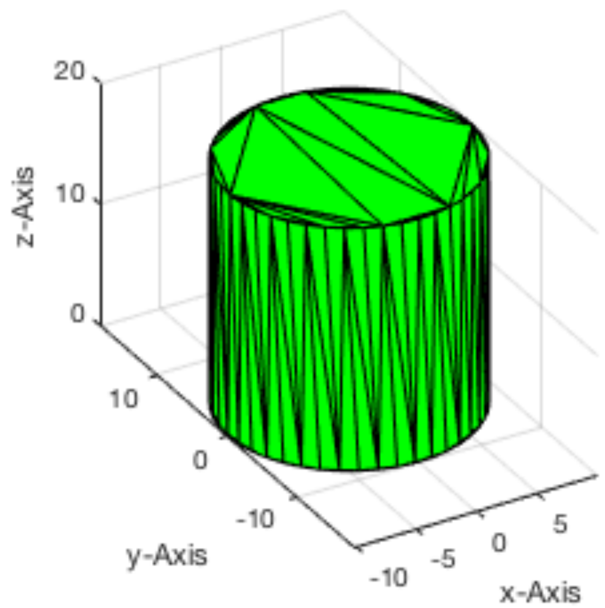
```
SGfigure; view(-30,30);
```

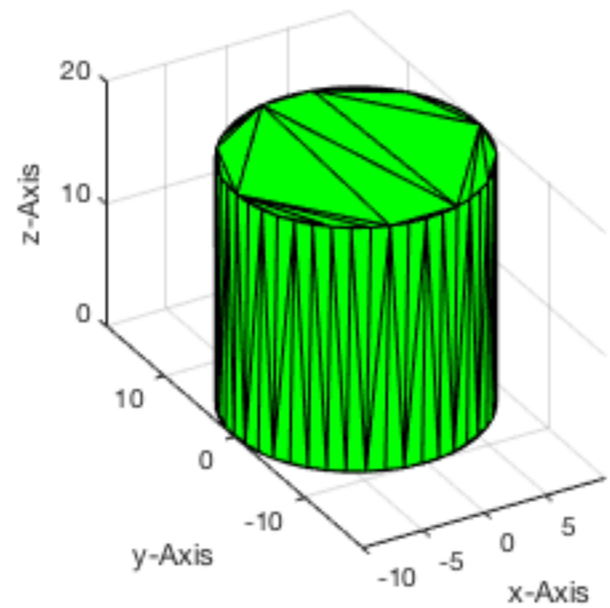
```
subplot(1,2,1);
SG=SGof2CPLz(PLcircle(10),PLcircle(10),20); SGplot(SG,'g'); view(-30,30);
subplot(1,2,2);
SG=SGof2CPLz(PLcircle(10),PLtransR(PLcircle(10),rotdeg(120)),20,[],'none');
SGplot(SG,'g'); view(-30,30);
```



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.

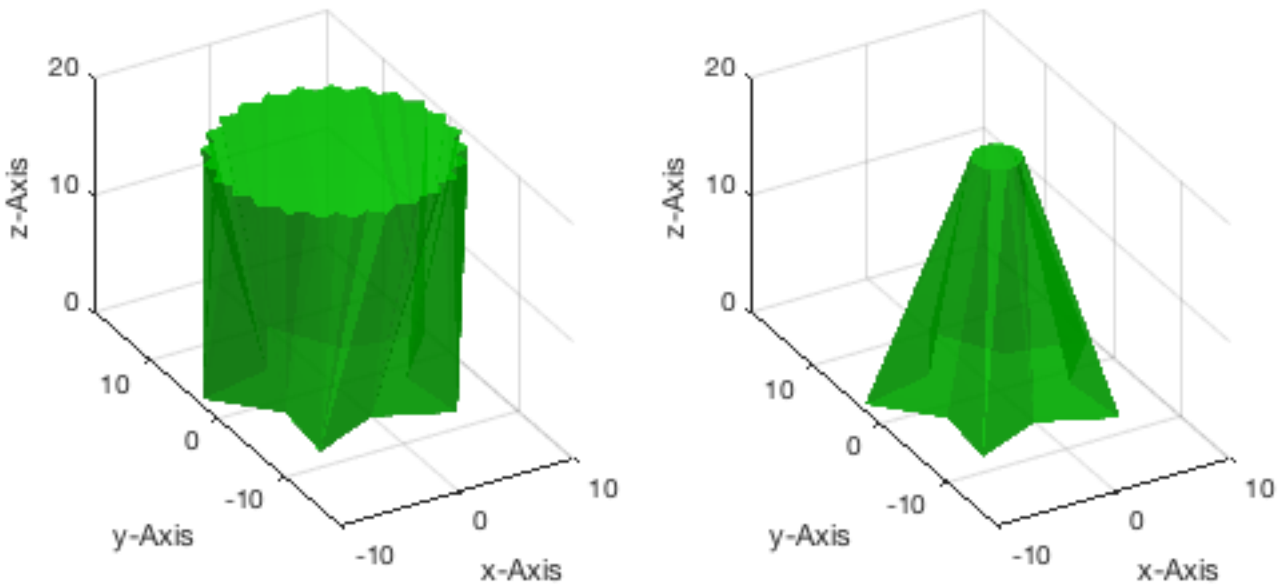
4. Solid surface generation and the importance of point assignment strategy

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'**

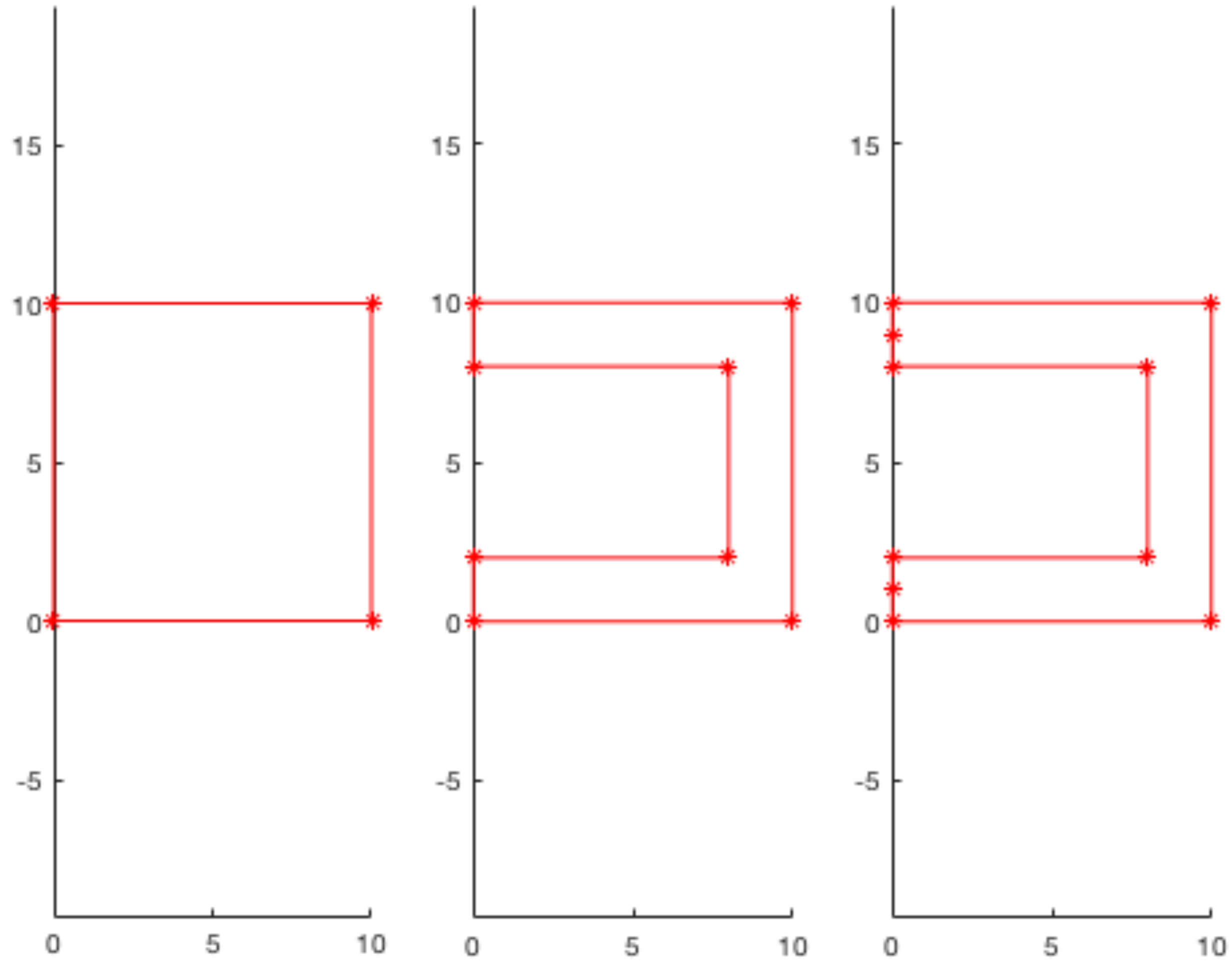
```
SGfigure; view(-30,30);
subplot(1,2,1);
SG=SGof2CPLz(PLstar(10,10),PLstar(10,50),20); SGplot(SG,'g'); view(-30,30);
VLFLplotlight(1,0.7);
subplot(1,2,2);
SG=SGof2CPLz(PLstar(10,10),PLcircle(2),20); SGplot(SG,'g'); view(-30,30);
VLFLplotlight(1,0.7);
```



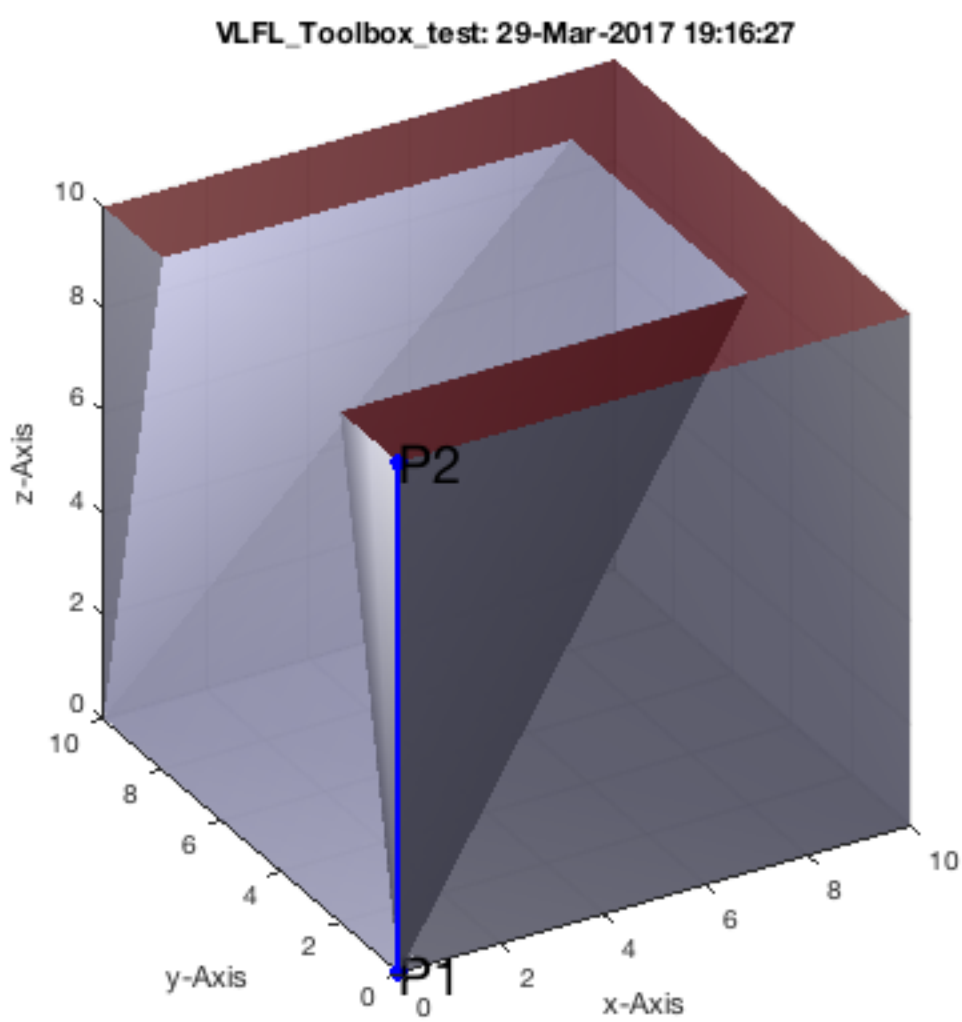
5. Solid surface generation for polygons with a small number of points

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

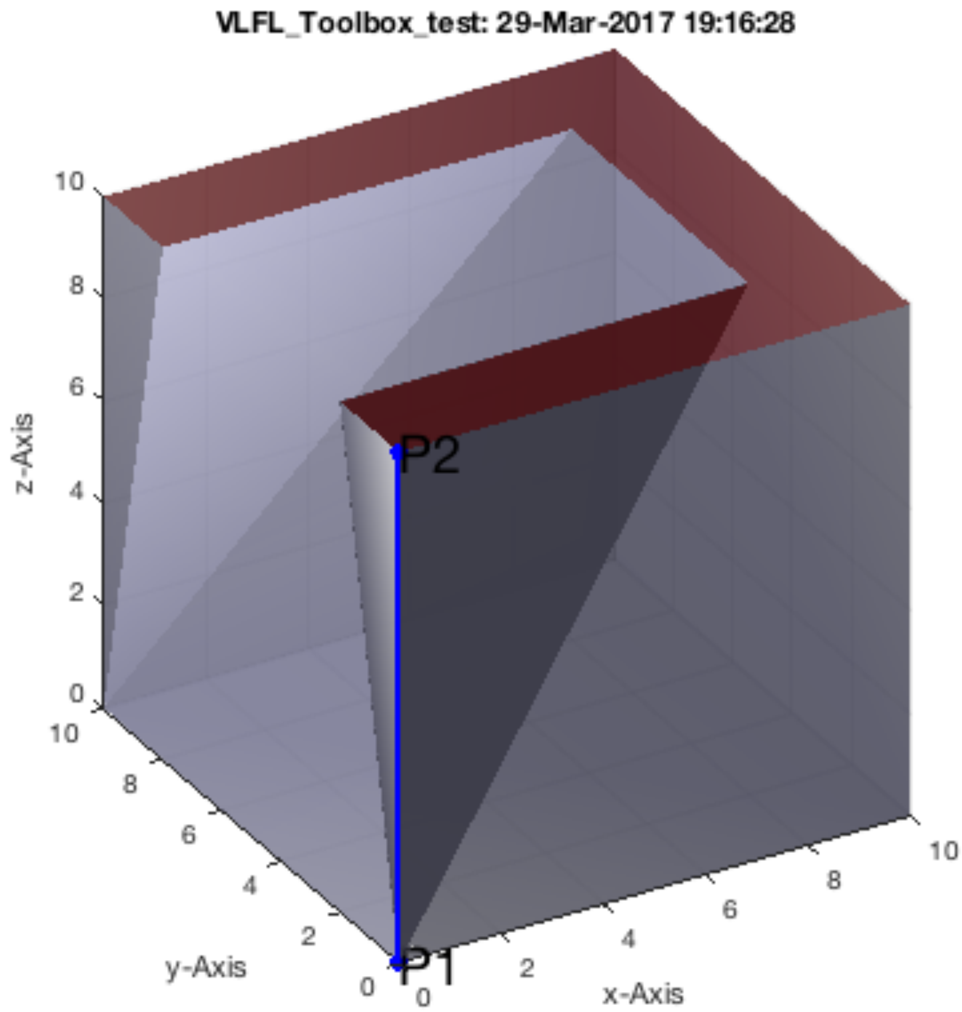
```
SGfigure;
PLU0=[0 0;10 0; 10 10; 0 10; 0 0];
PLU1=[0 0;10 0; 10 10; 0 10; 0 8; 8 8; 8 2; 0 2; 0 0];
PLU2=[0 0;10 0; 10 10; 0 10; 0 9; 0 8; 8 8; 8 2; 0 2; 0 1; 0 0];
subplot(1,3,1); CPLplot(PLU0);  view(0,90); axis equal;
subplot(1,3,2); CPLplot(PLU1);  view(0,90); axis equal
subplot(1,3,3); CPLplot(PLU2);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



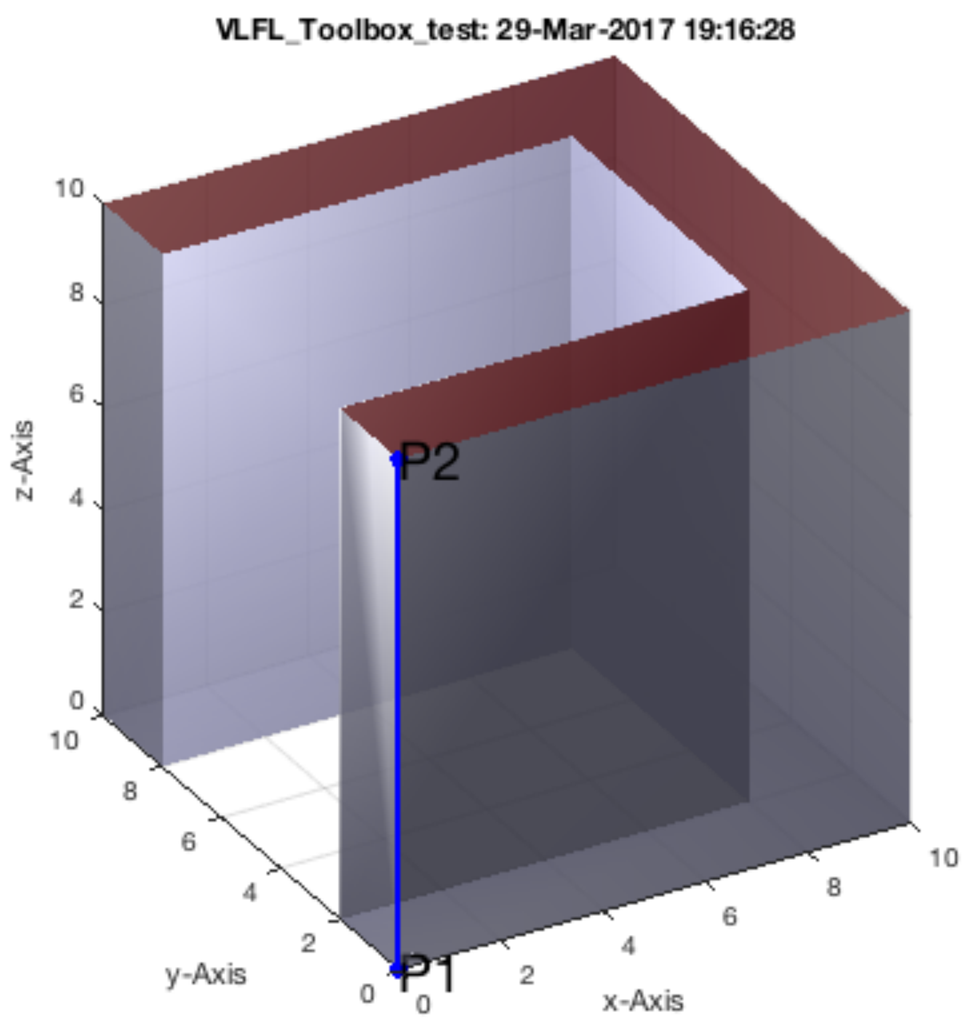

```
SGof2CPLz(PLU0,PLU2,10); VLFLplotlight(1,0.7);
```




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



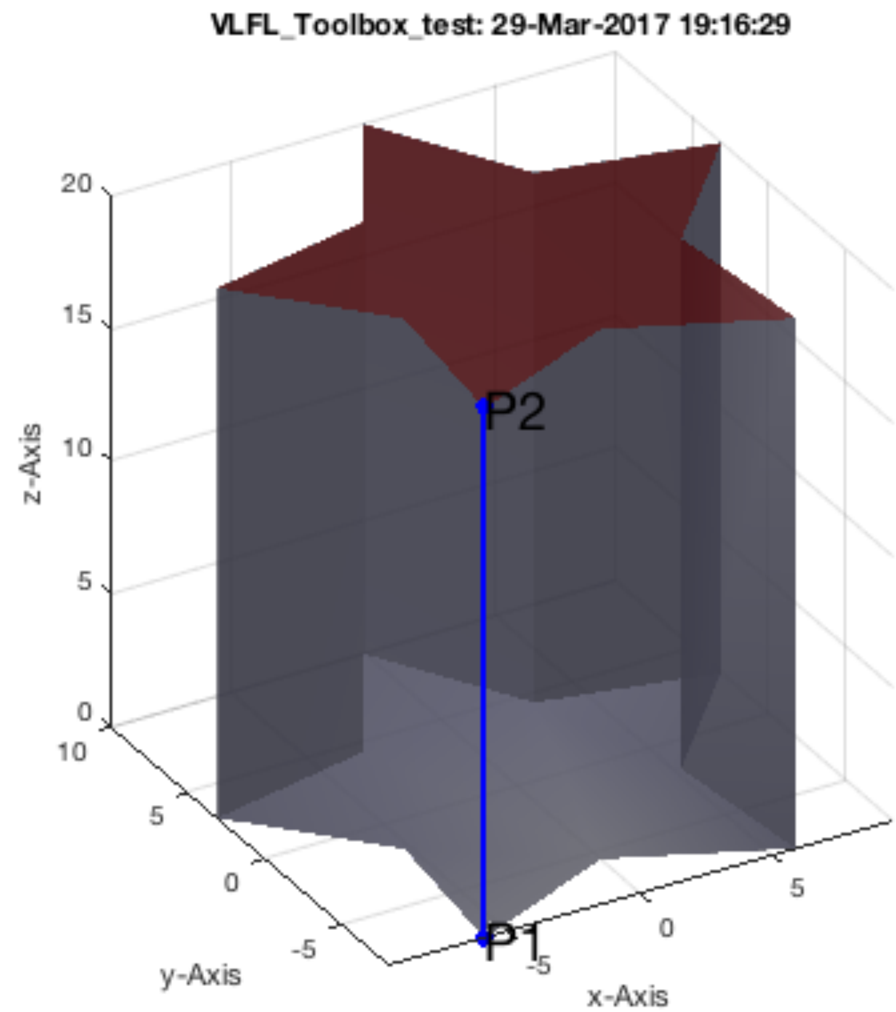

```
SGof2CPLz(PLU1,PLU2,10); VLFLplotlight(1,0.7);
```



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.

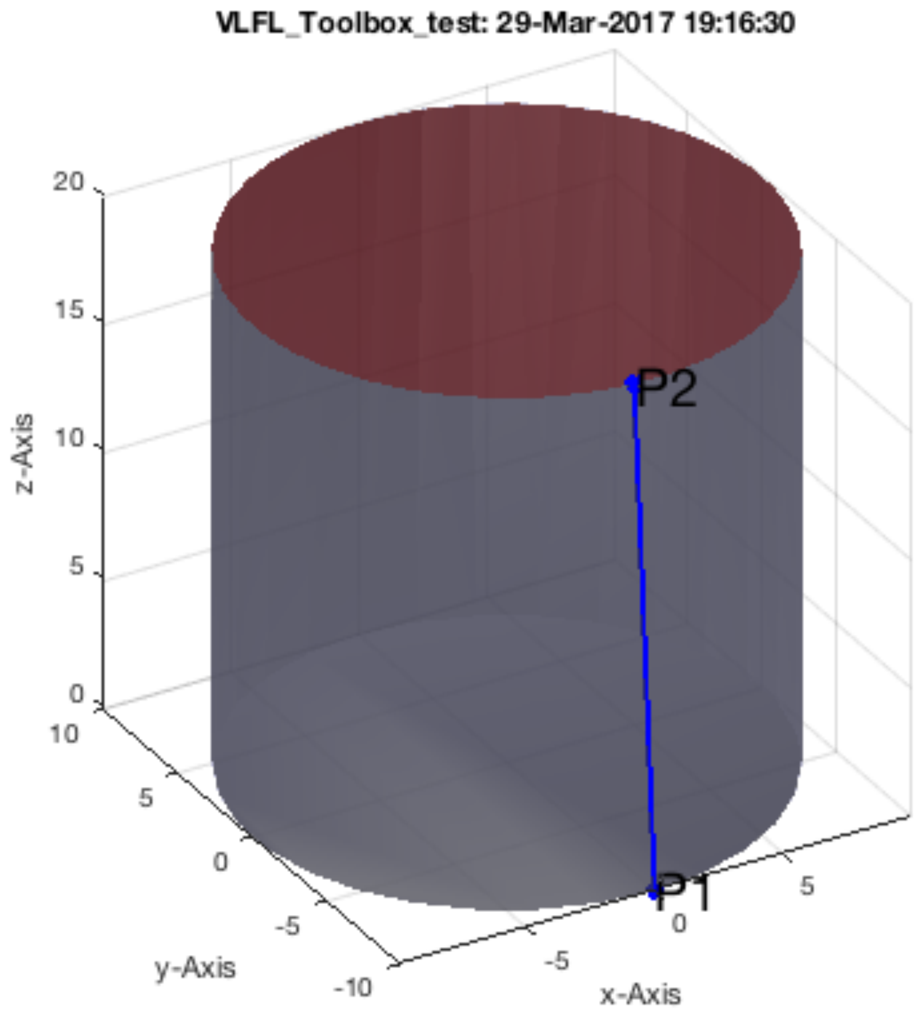
```
SGof2CPLz(PLstar(10,10),PLstar(10,10),20); VLFLplotlight(1,0.7);
```



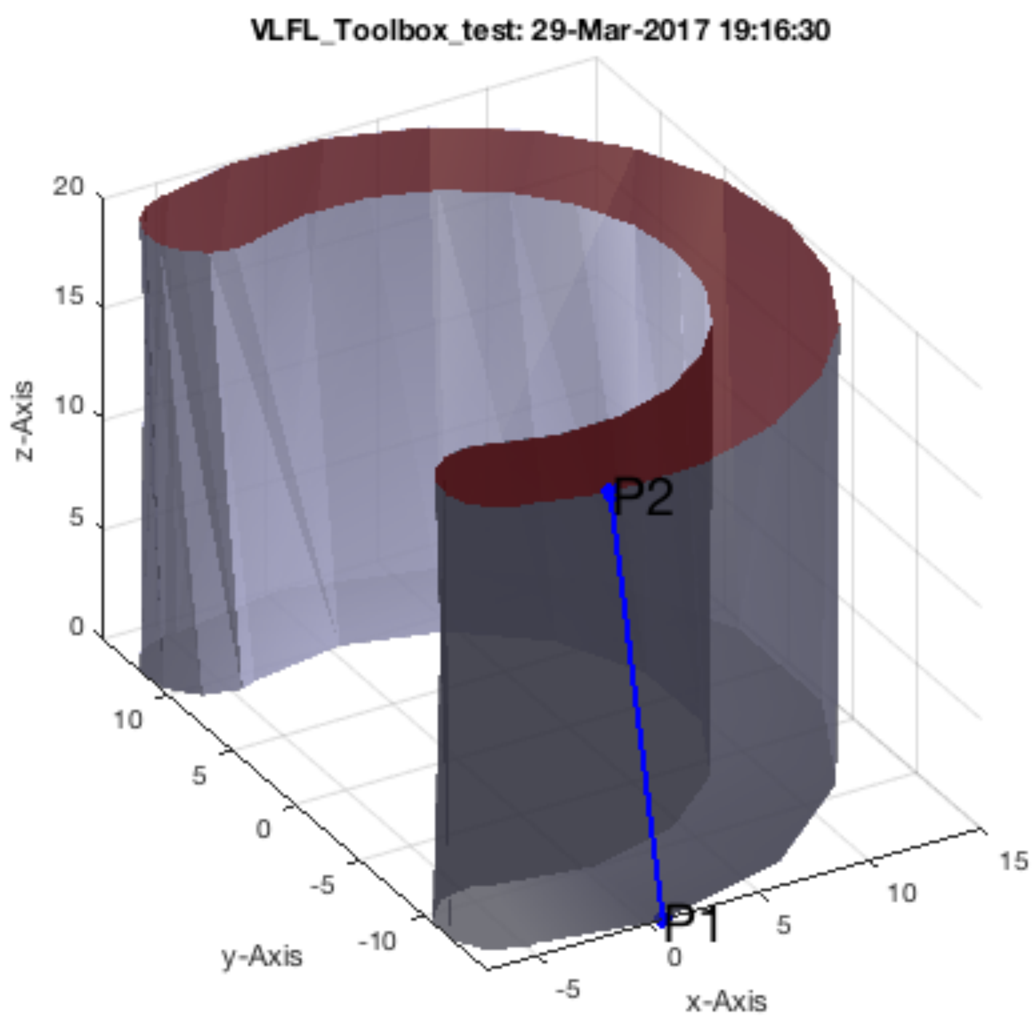
7. Two contours of the same shape with different number of points

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);
```



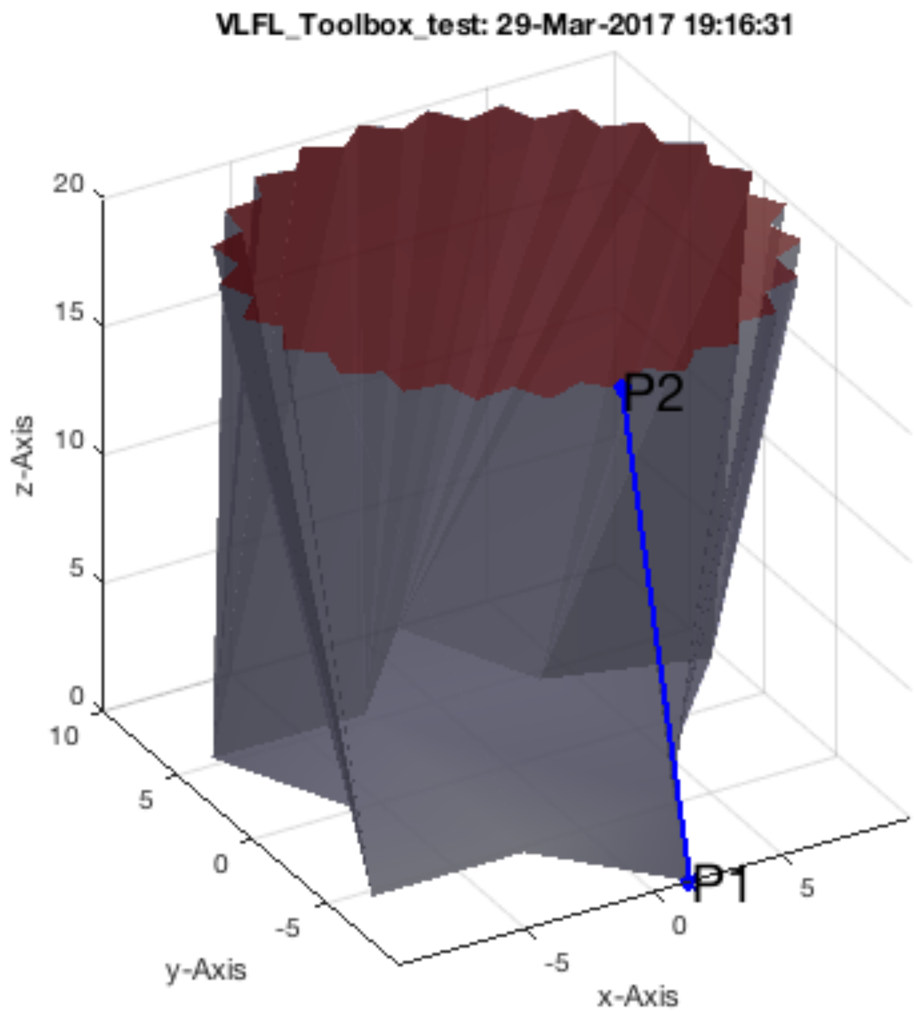

```
SGof2CPLz(PLkidney(10,15,pi/0.8,10),PLkidney(10,15,pi/0.8,15),20);
VLFLplotlight(1,0.7);
```



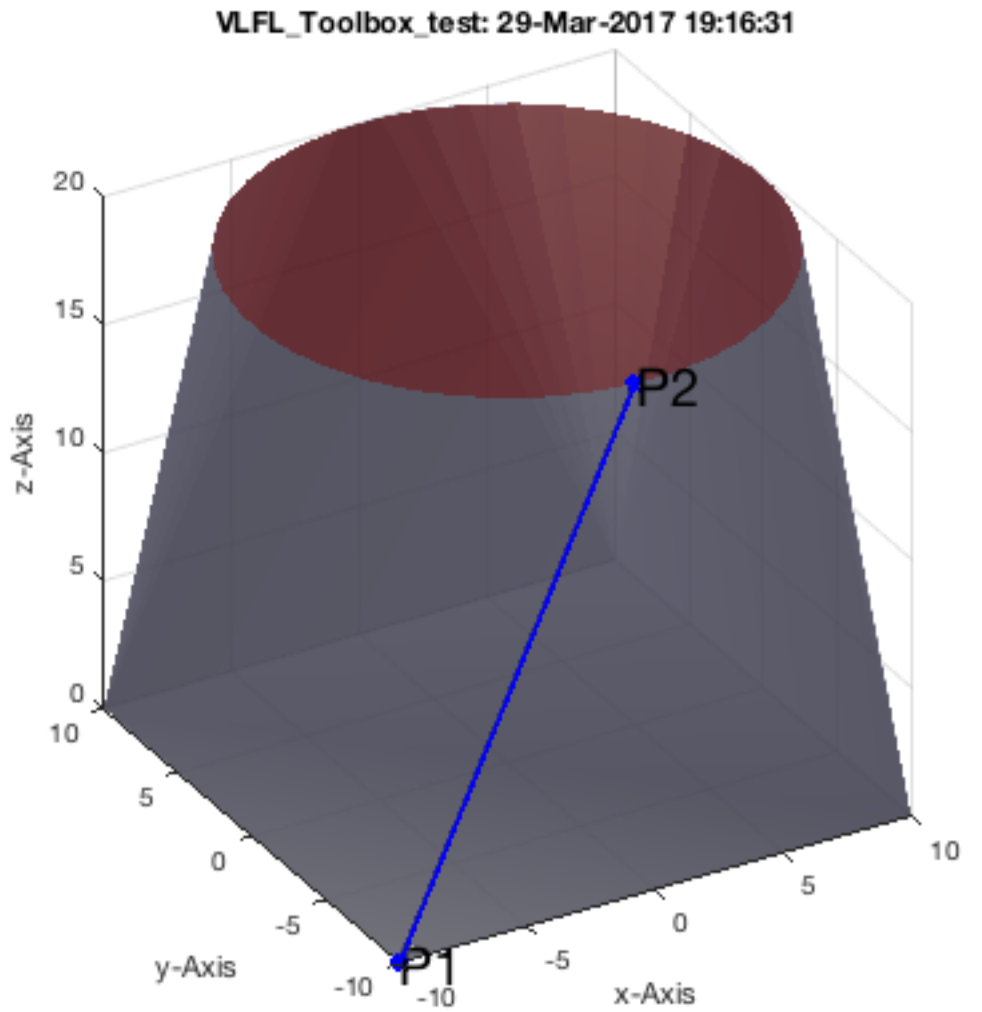
8. Two contours of the similar shape with different number of points

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.

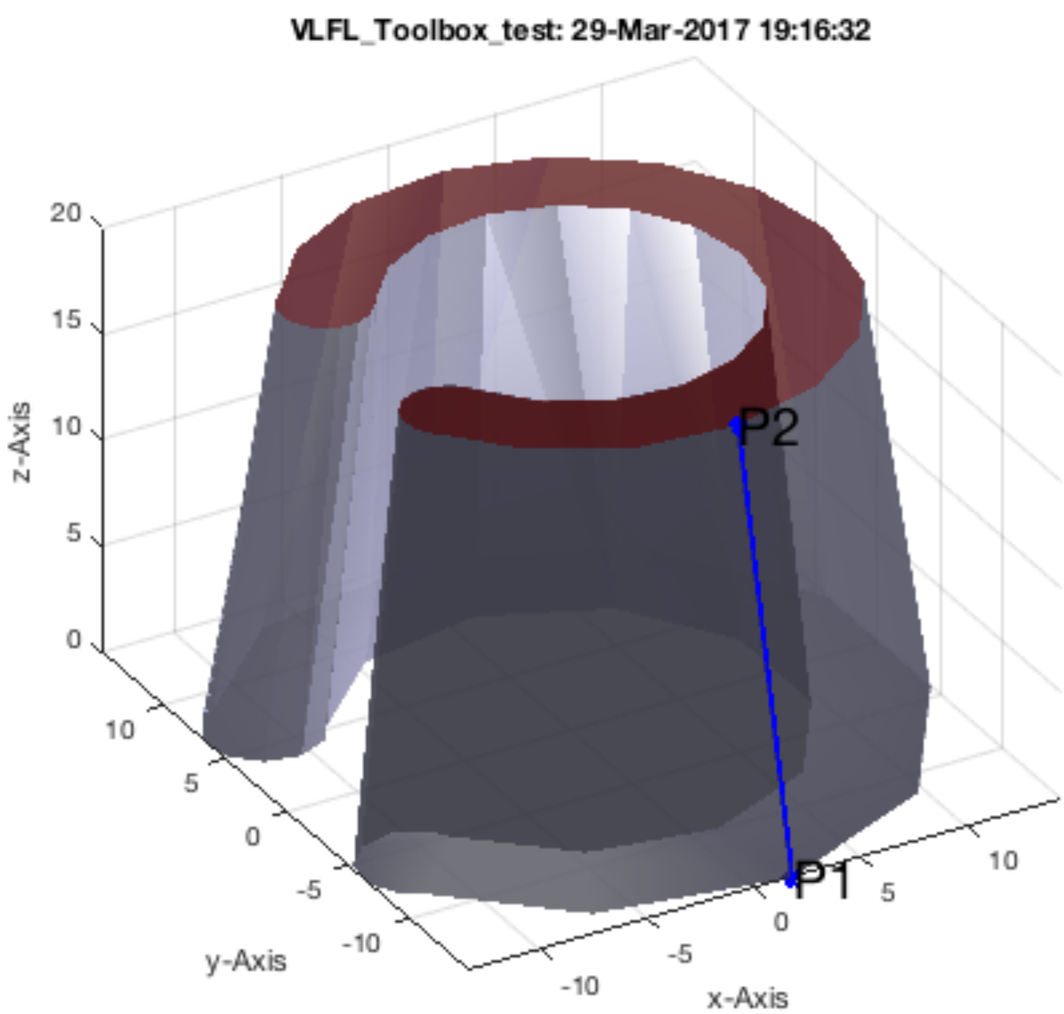
```
SGof2CPLz(PLstar(10,11),PLstar(10,50),20); VLFLplotlight(1,0.7);
```




```
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);
```

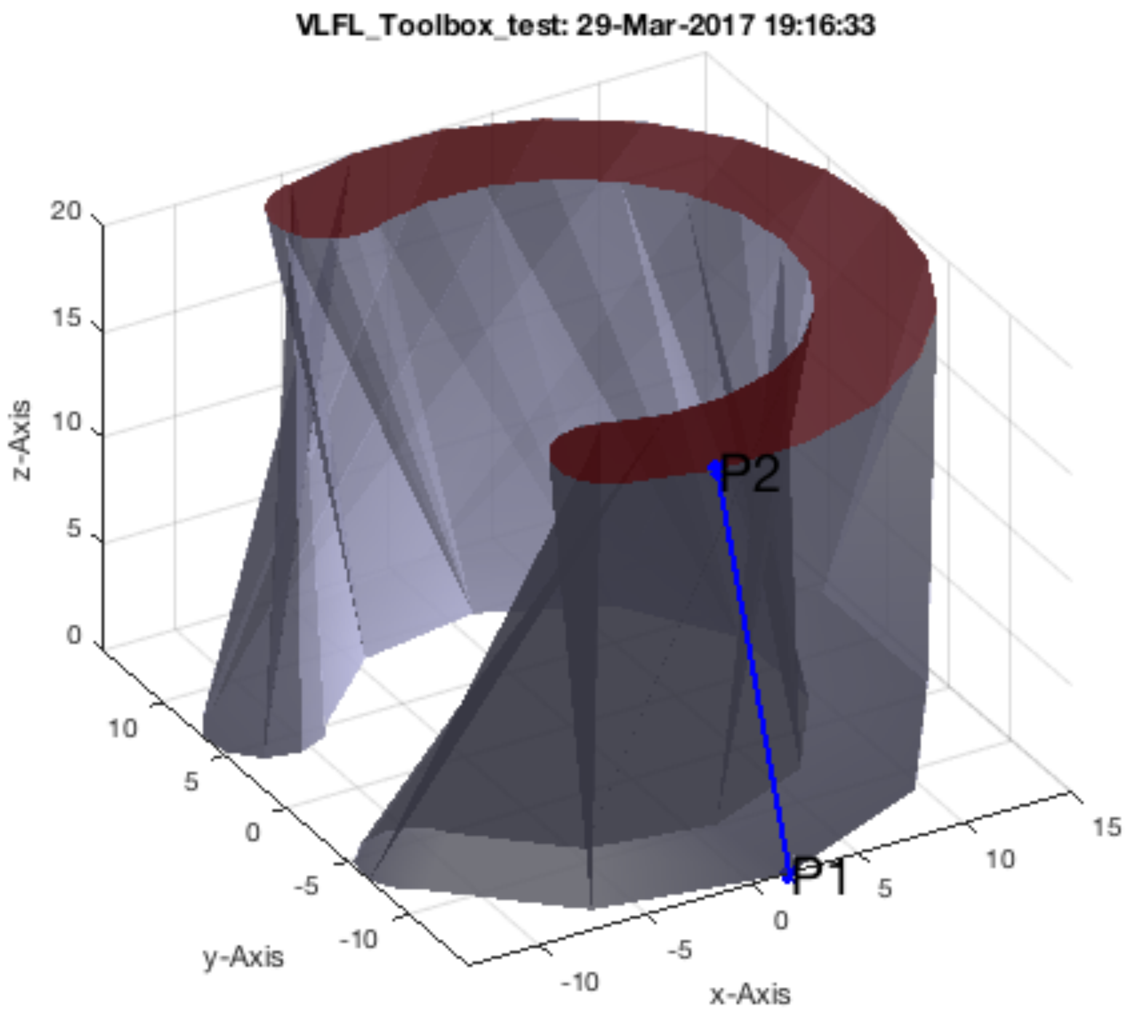


In this case we see that the kidney has different size but the same bending angle of pi/.6.

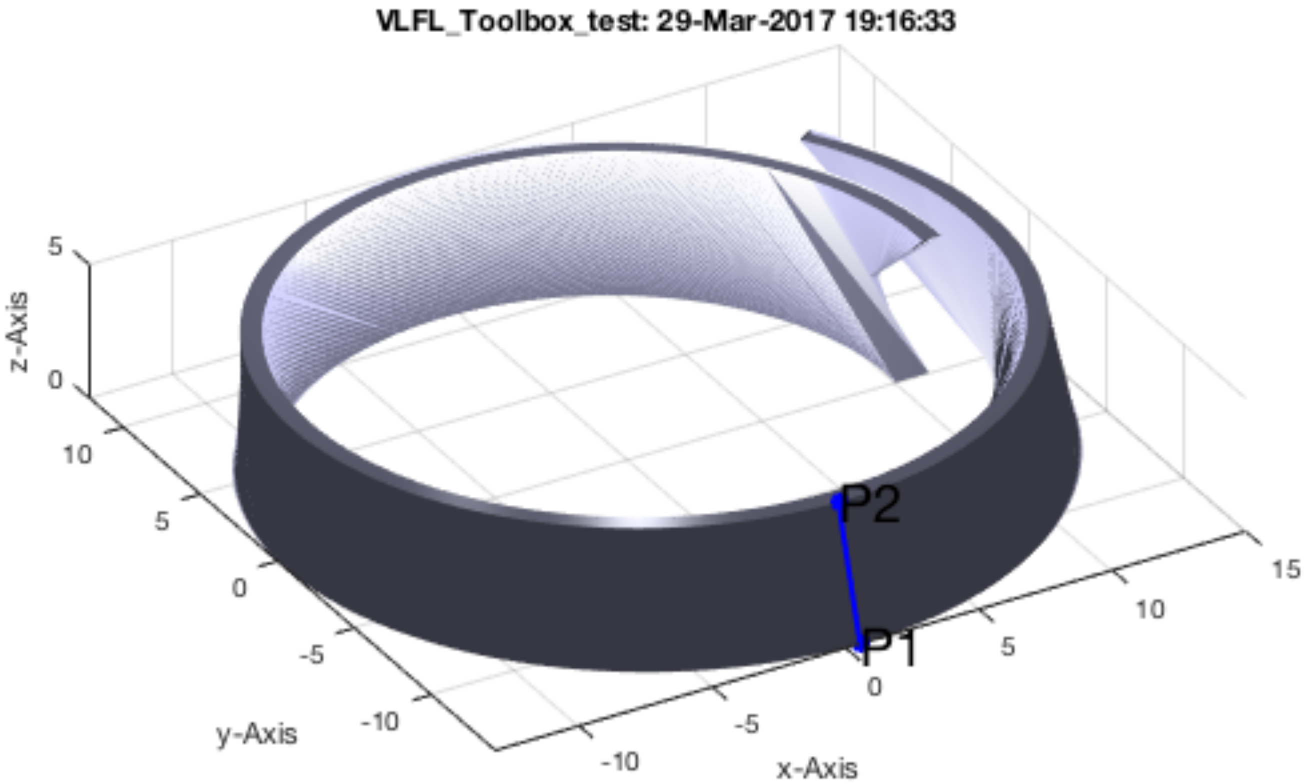
9. Two contours of the similar shape with different sum of boundary bending angle sum

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);
```

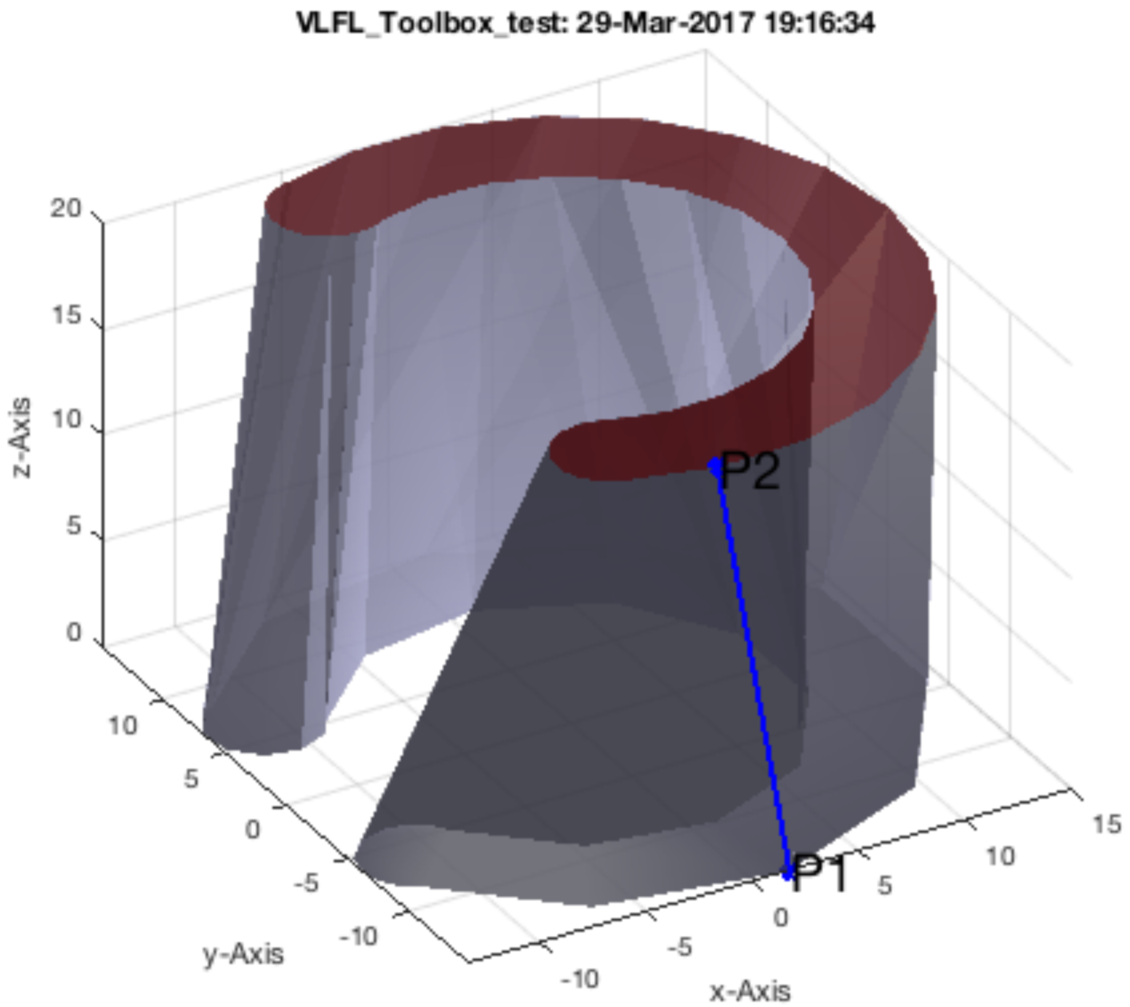



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5);
VLFLplotlight (1,1);
```

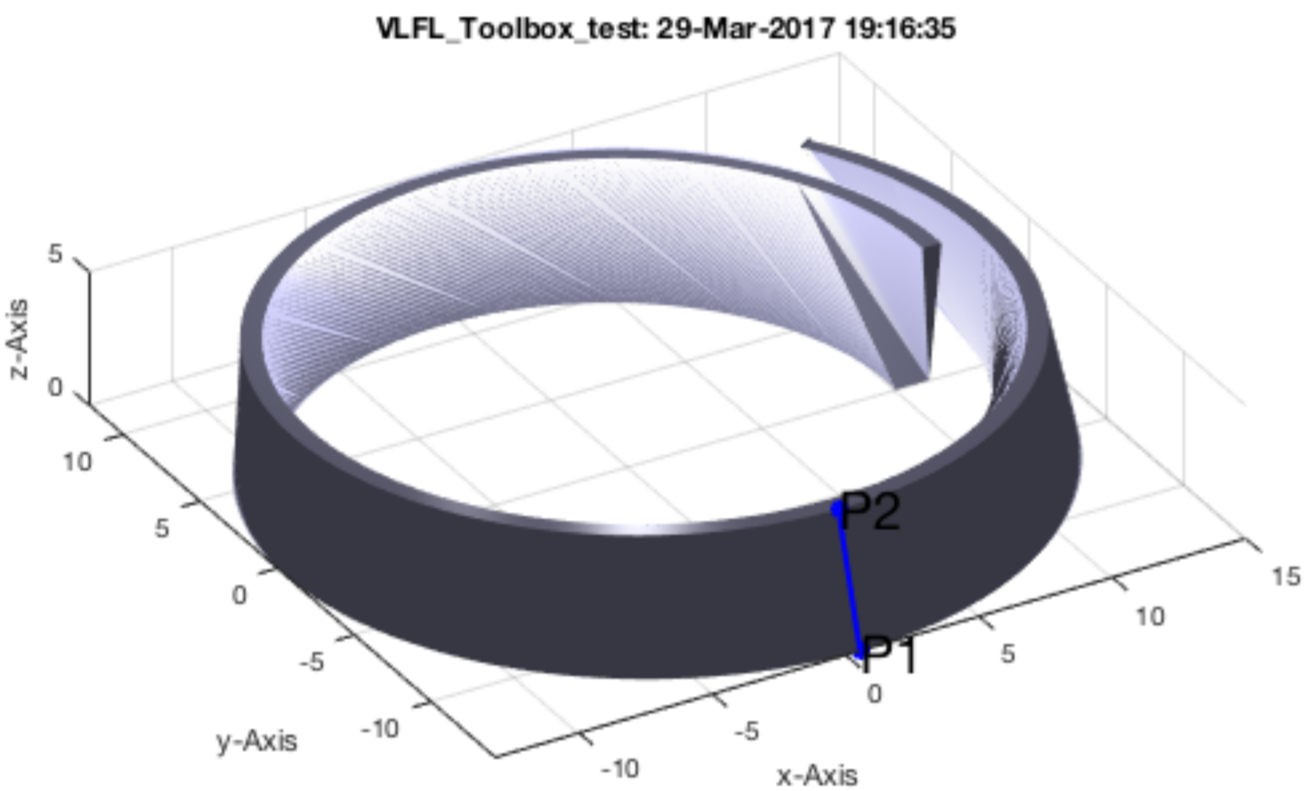


In this case we see malformed solids.

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

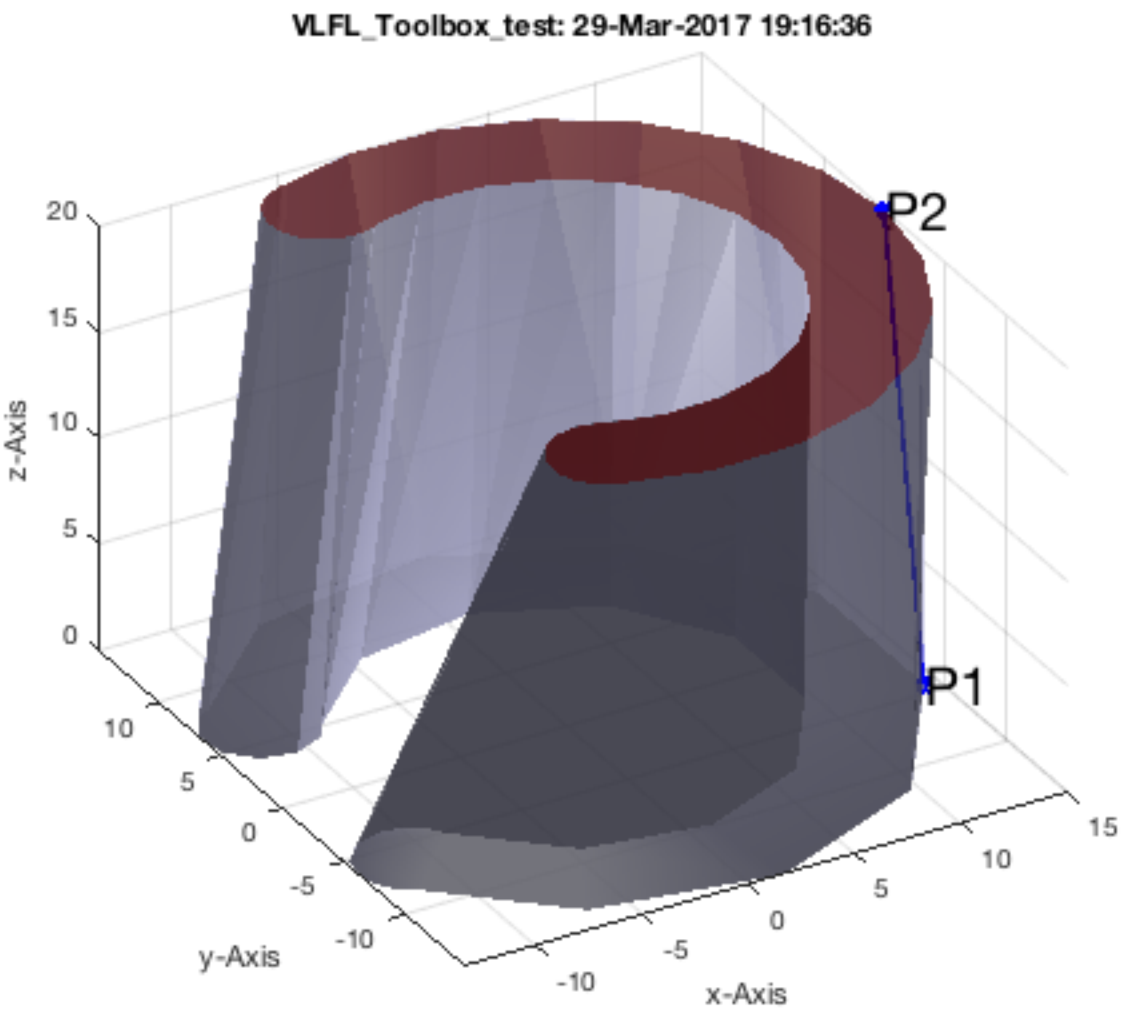



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5,'angle');
VLFLplotlight (1,1);
```

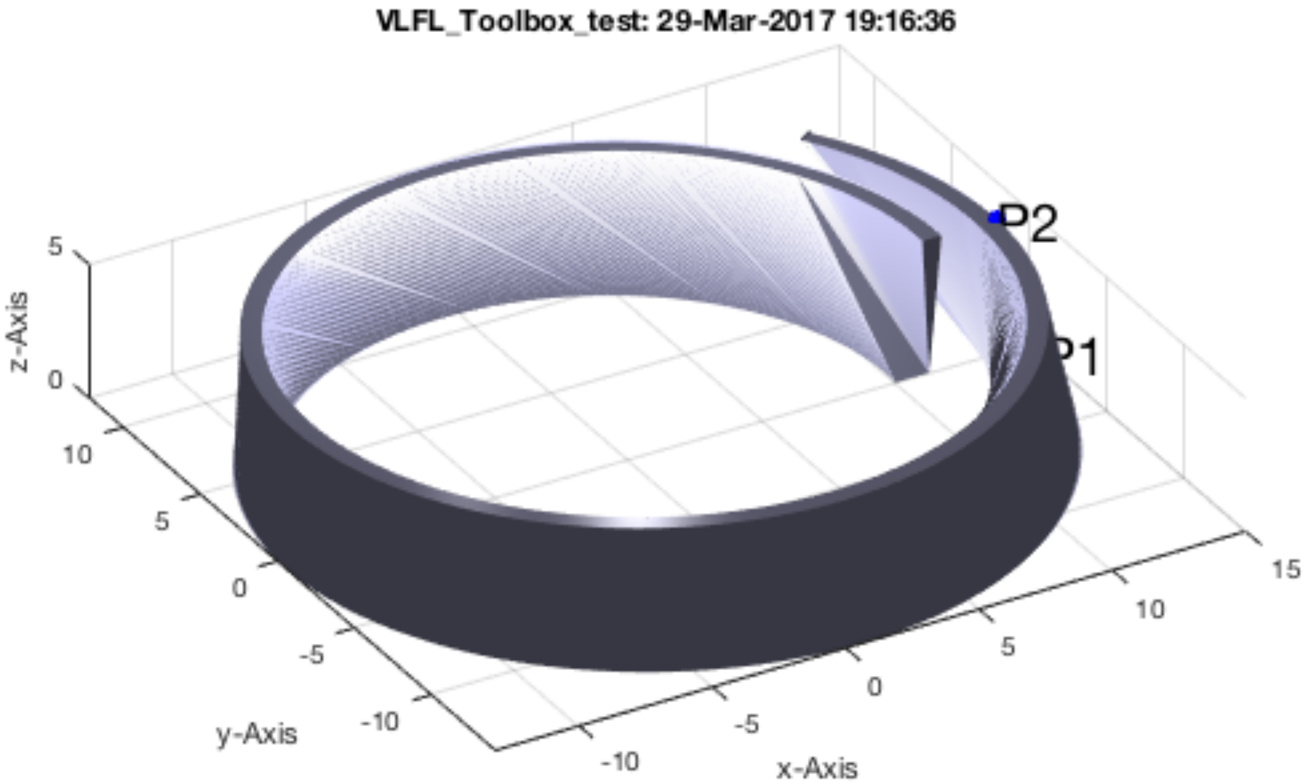


By using 'angle' instead of 'length', the solids are more what we expected.

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

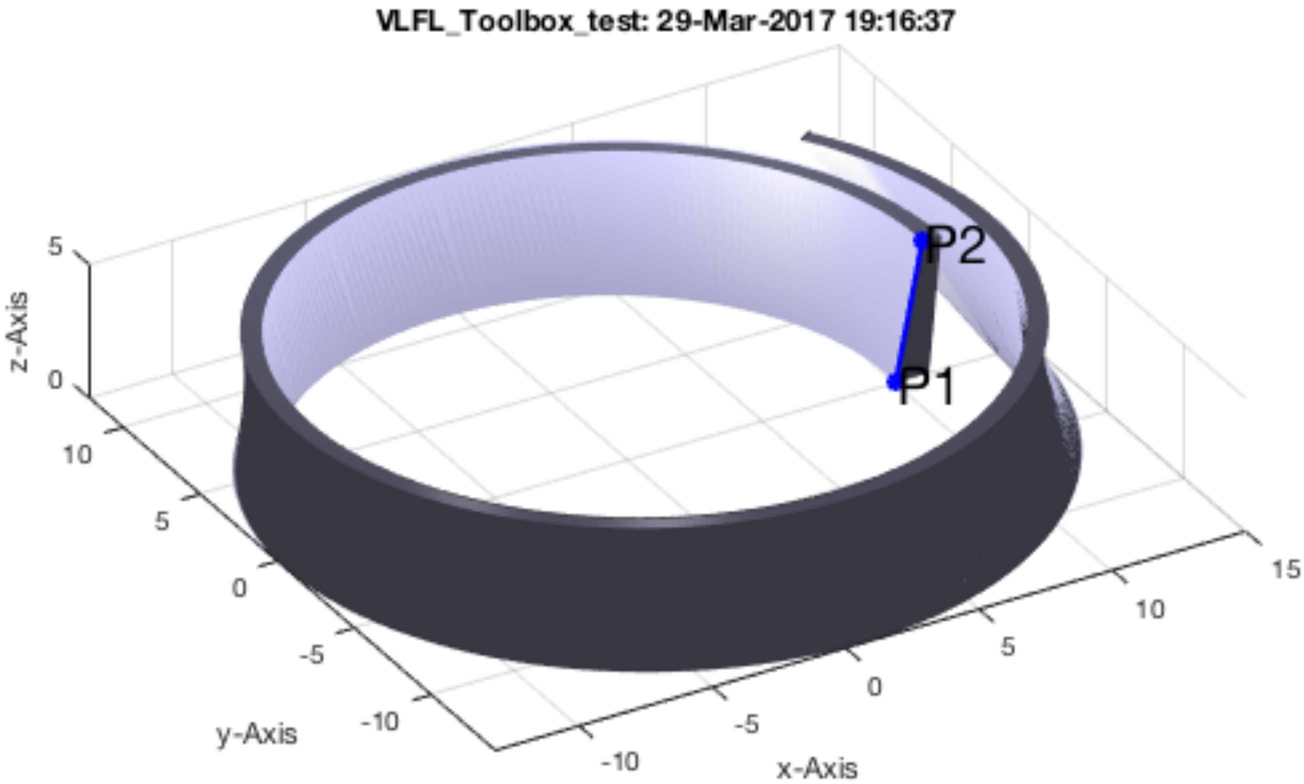



```
SGof2CPLz(CPLspiral(10,15,2*pi),CPLspiral(11,14,2.2*pi),5,'angle','rot');
VLFLplotlight (1,1);
```



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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:16:40 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 16: Create Tube-Style Solids by Succeeding Polygons

- 2015-10-04: Tim C. Lueth, MIMED - Technische Universität München, Germany
- 2016-12-04: Tim C. Lueth, VLradialEdges and SGofCPLCVLR added (URL: http://www.mimed.de)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons

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

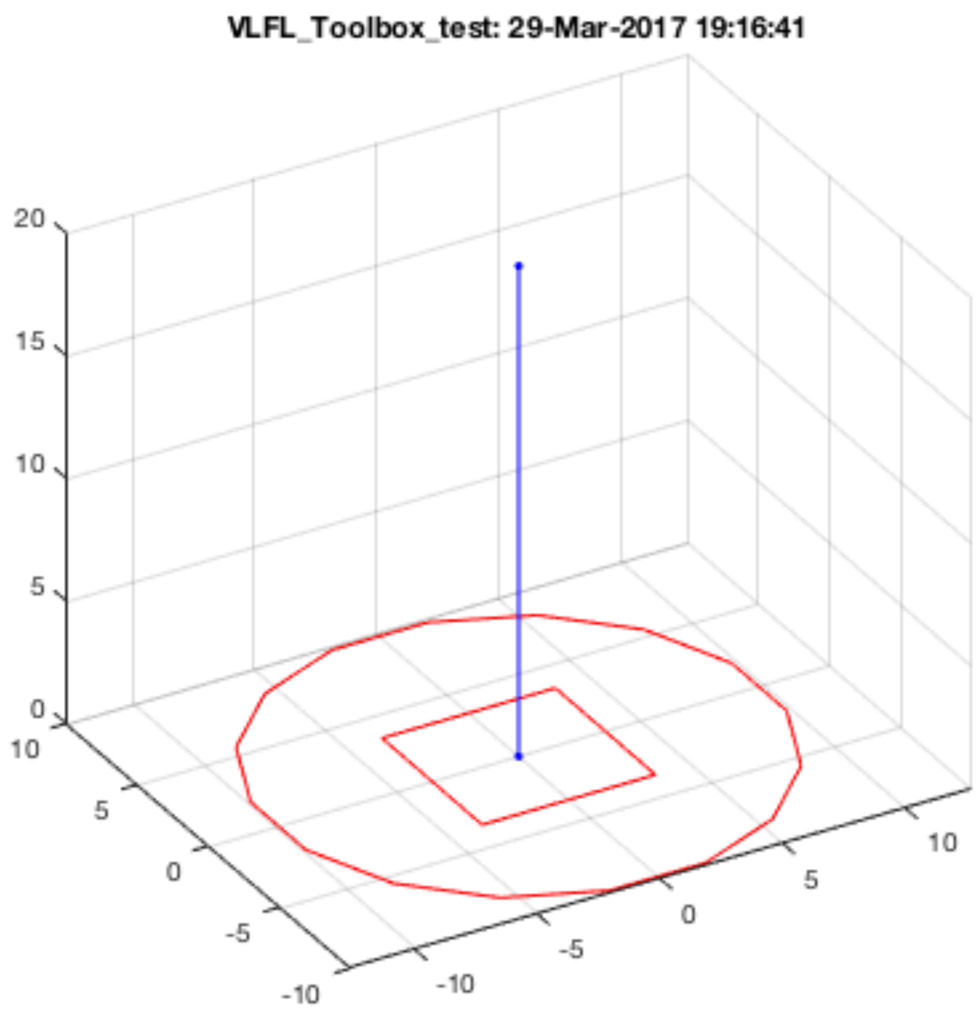
For robotics design,very often we have a wish to extrude a CPL not only in an orthognal z direction to the xy-plane, but in an any desired direction even along a path in 3D space. Intutively we expect a result, but this is not easy to achieve automatically. Anyway, for those tasks we have two functions:

- 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

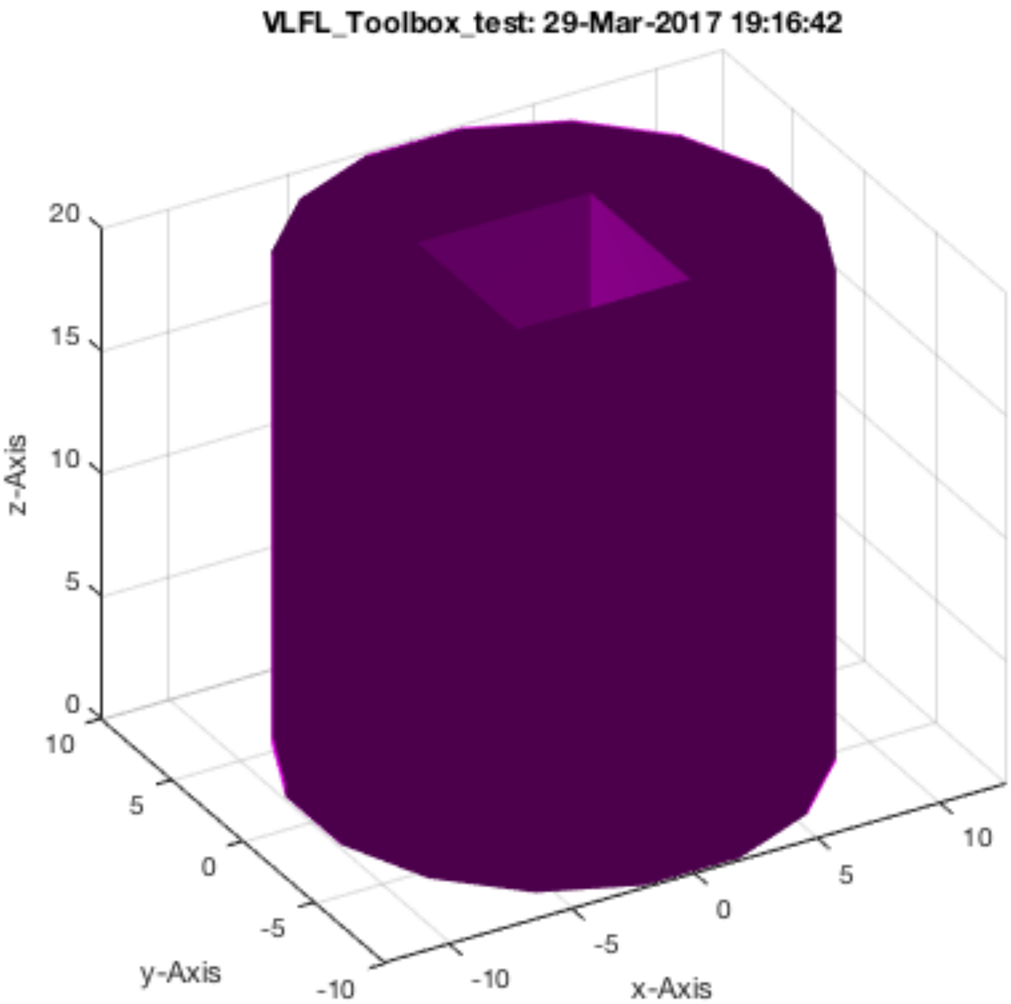
```
SGfigure; axis on ; grid on;
```

```
CPL=CPLsample(8);
CPLplot(CPL,'r-');

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



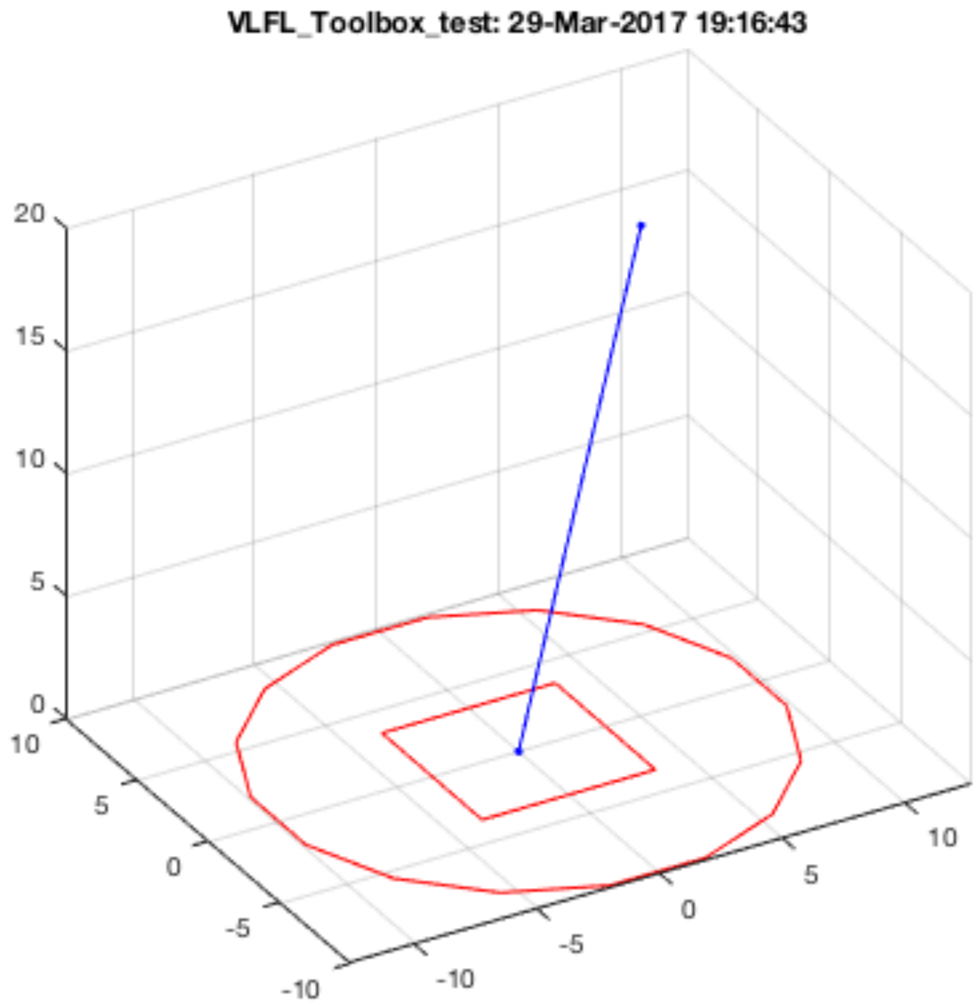
The final result looks as expected

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



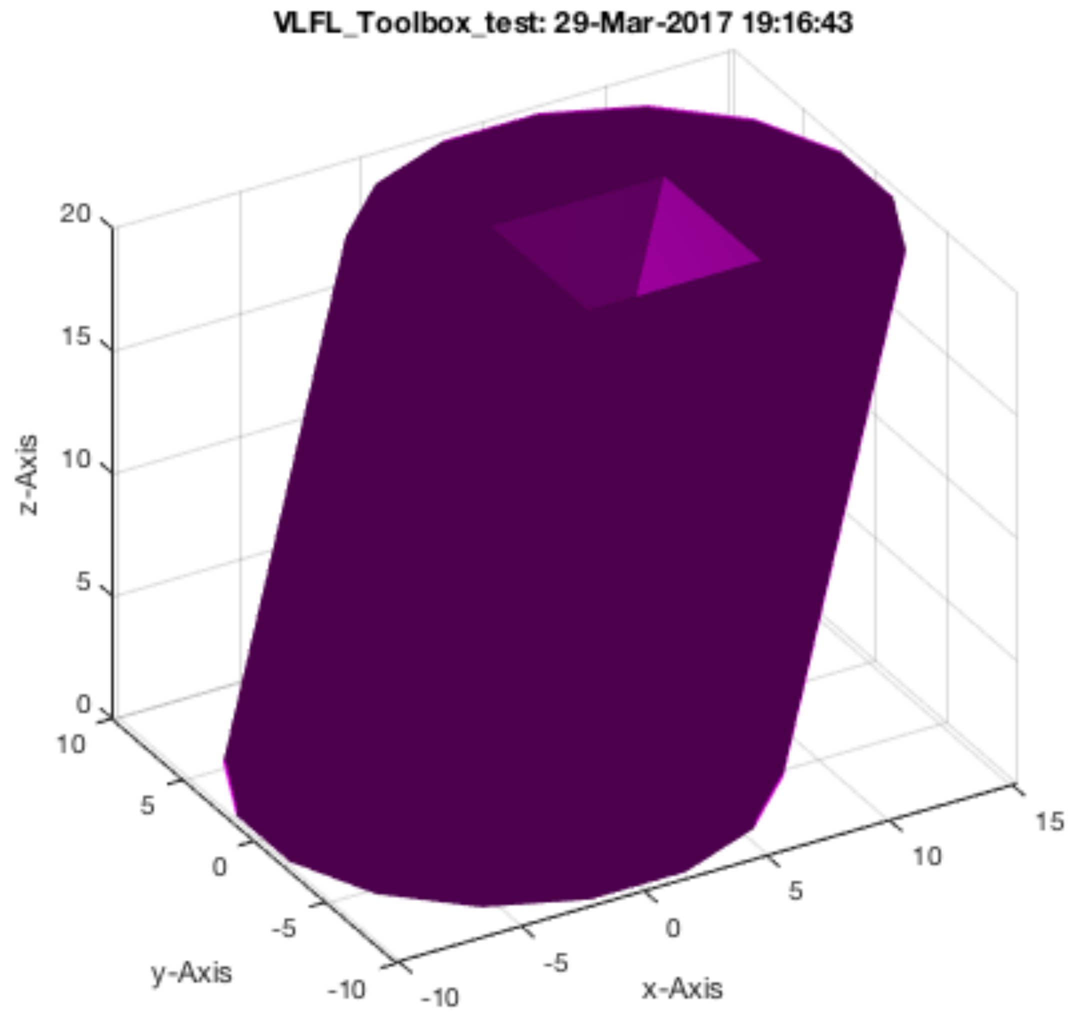
Now we change the path a little

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



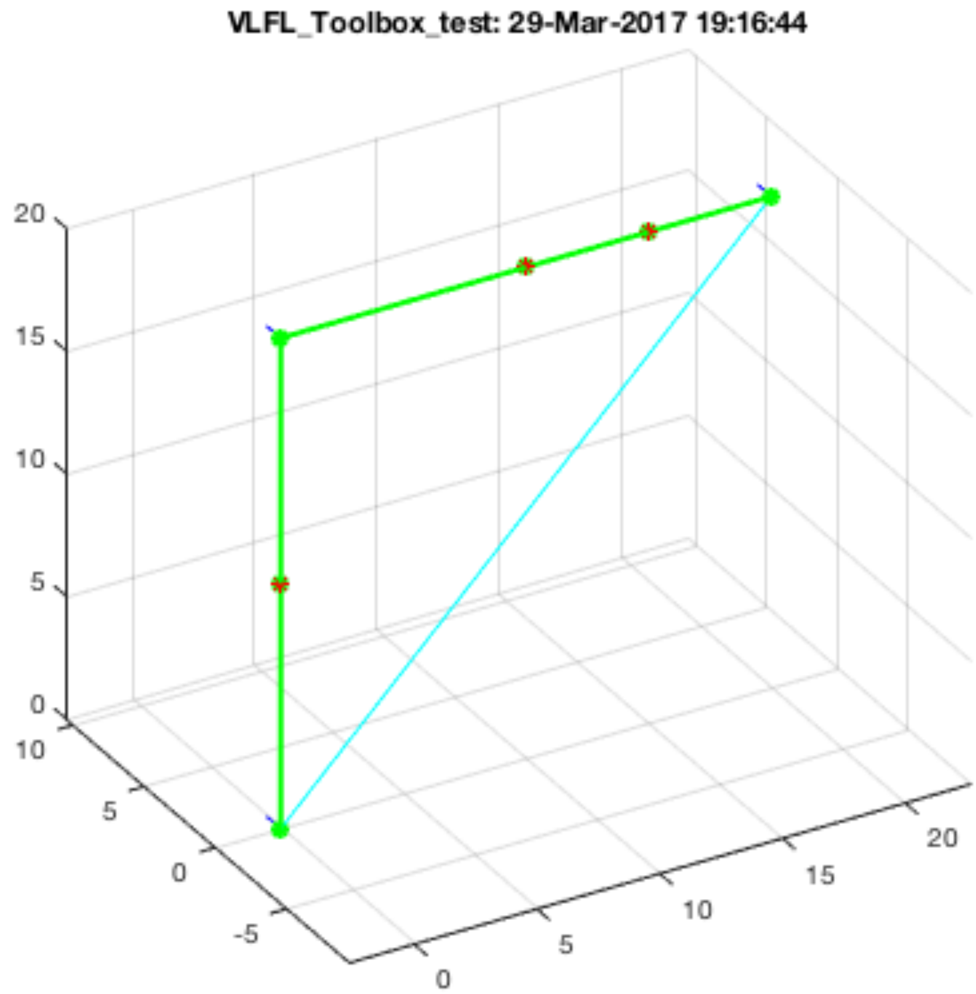
The final result may not look as expected

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



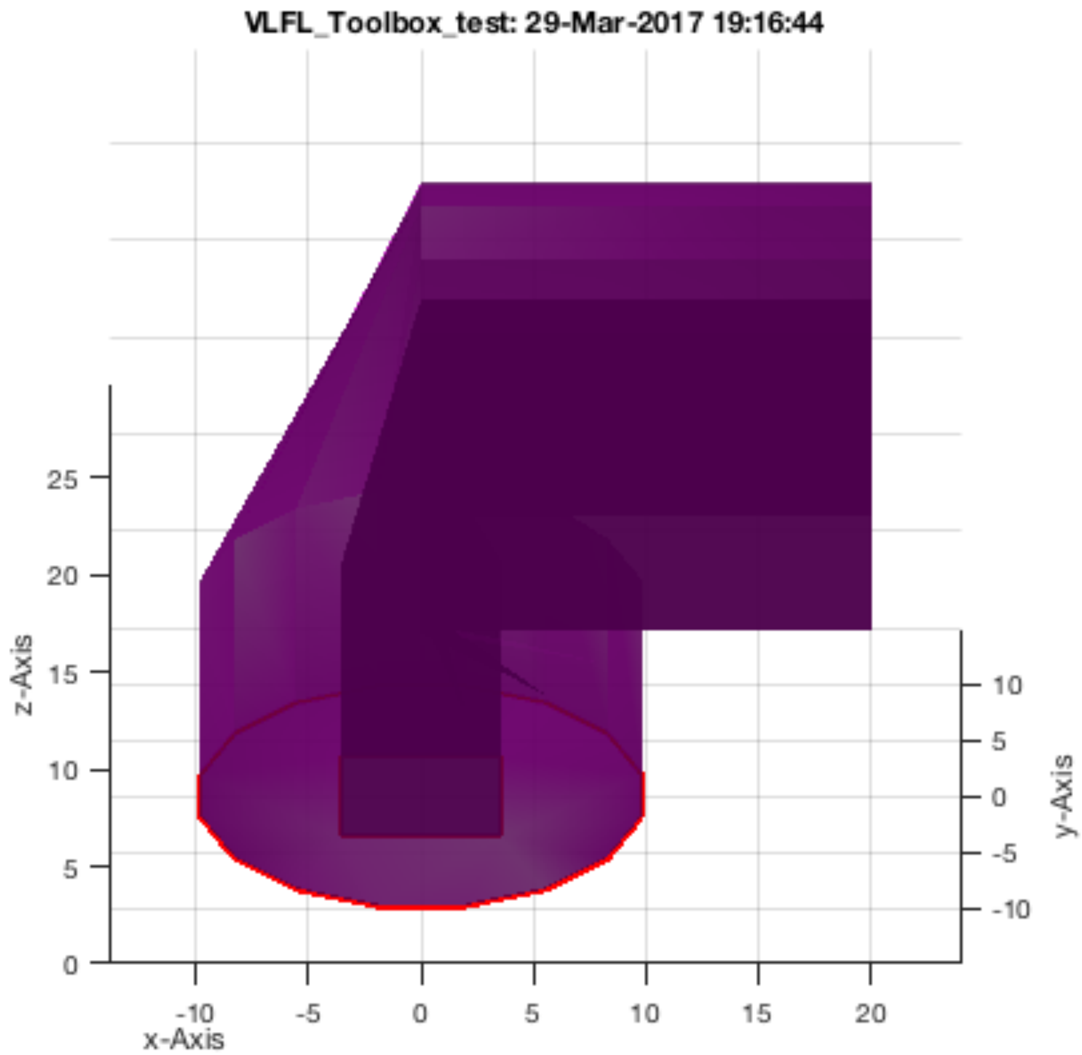
Now we change the path, still a planar one and anylyze the angle situation The cynan colored path explain that the vertex list has to be closed to analyze the first an last angle. Furthermore we see that more than one point has no defined orthogonal vector since it sits on a straight line

```
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);
```



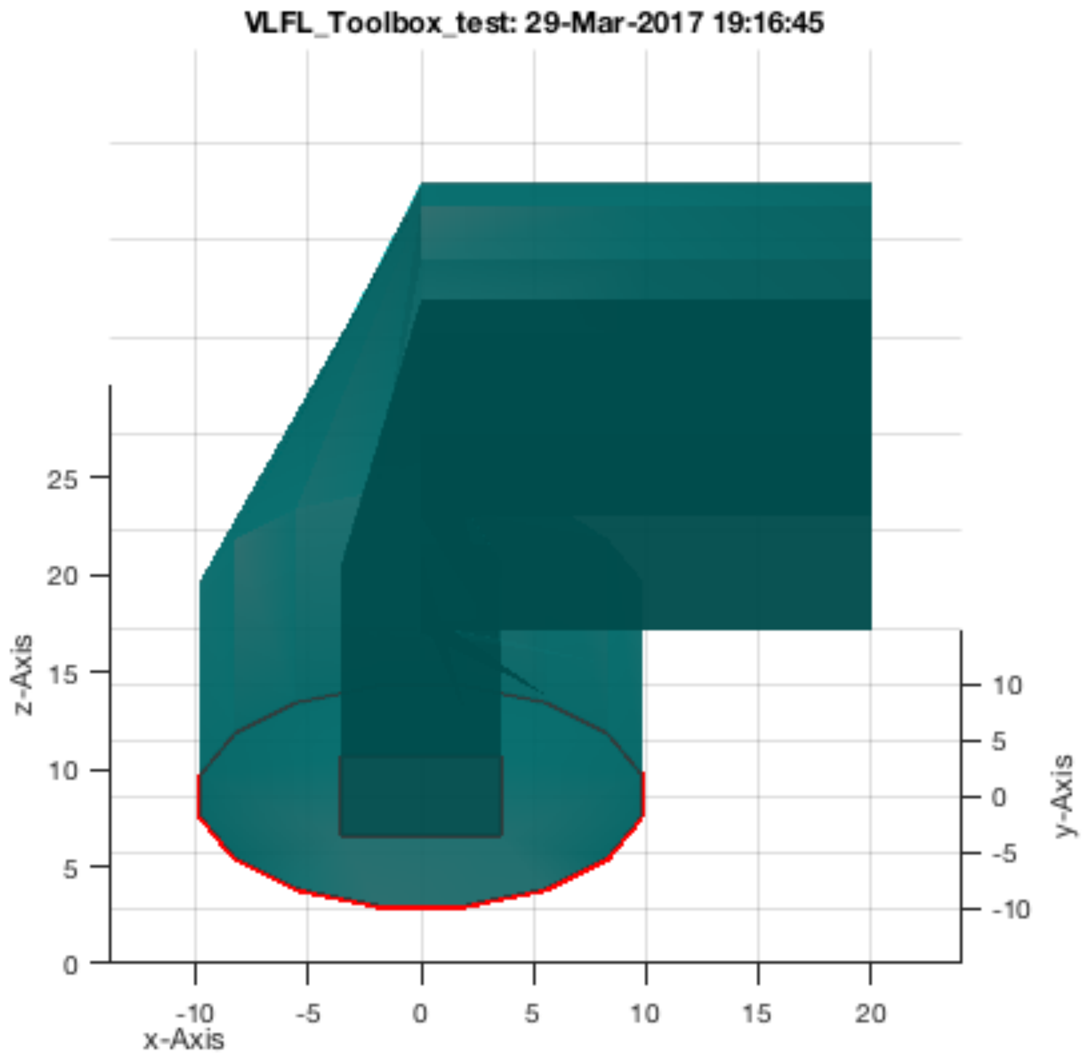
The result is not may be we expected

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

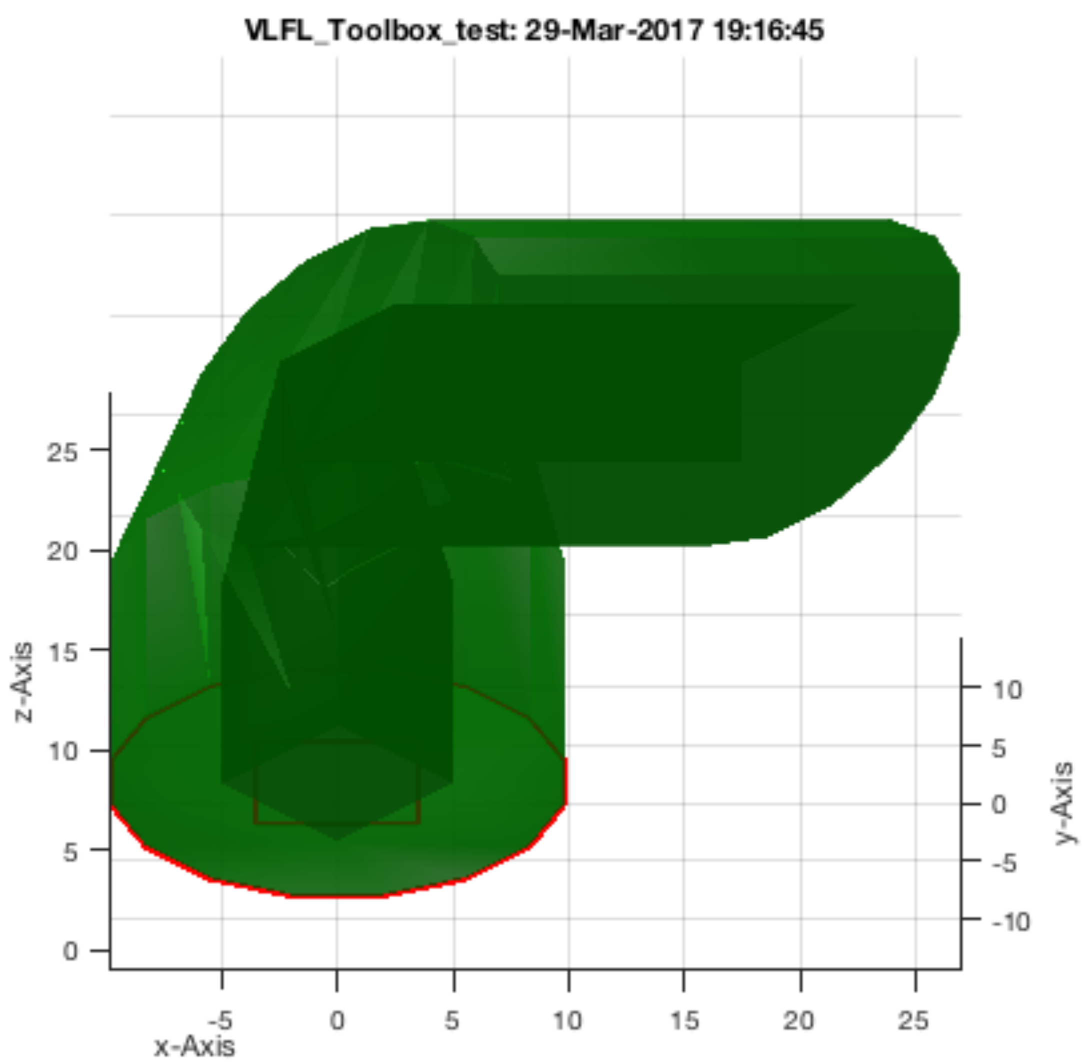


The result can be adjusted by defining the ex-vector at the start point

```
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);
```



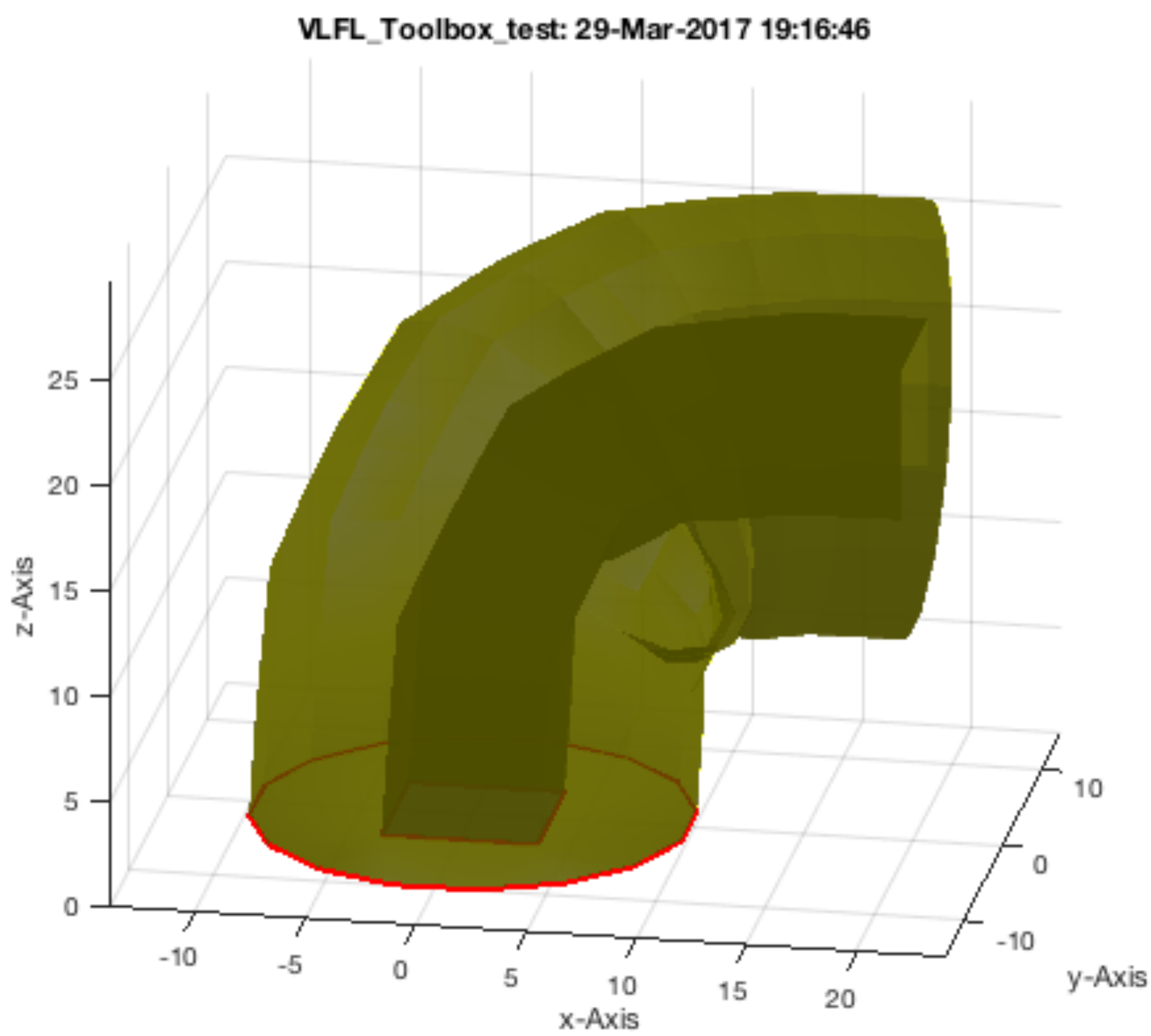

```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGcontourtube(CPL,VL,[1 1 0]); SGplot(SG,'g'); VLFLplotlight(1,0.8);view (0,30);
```



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);
```

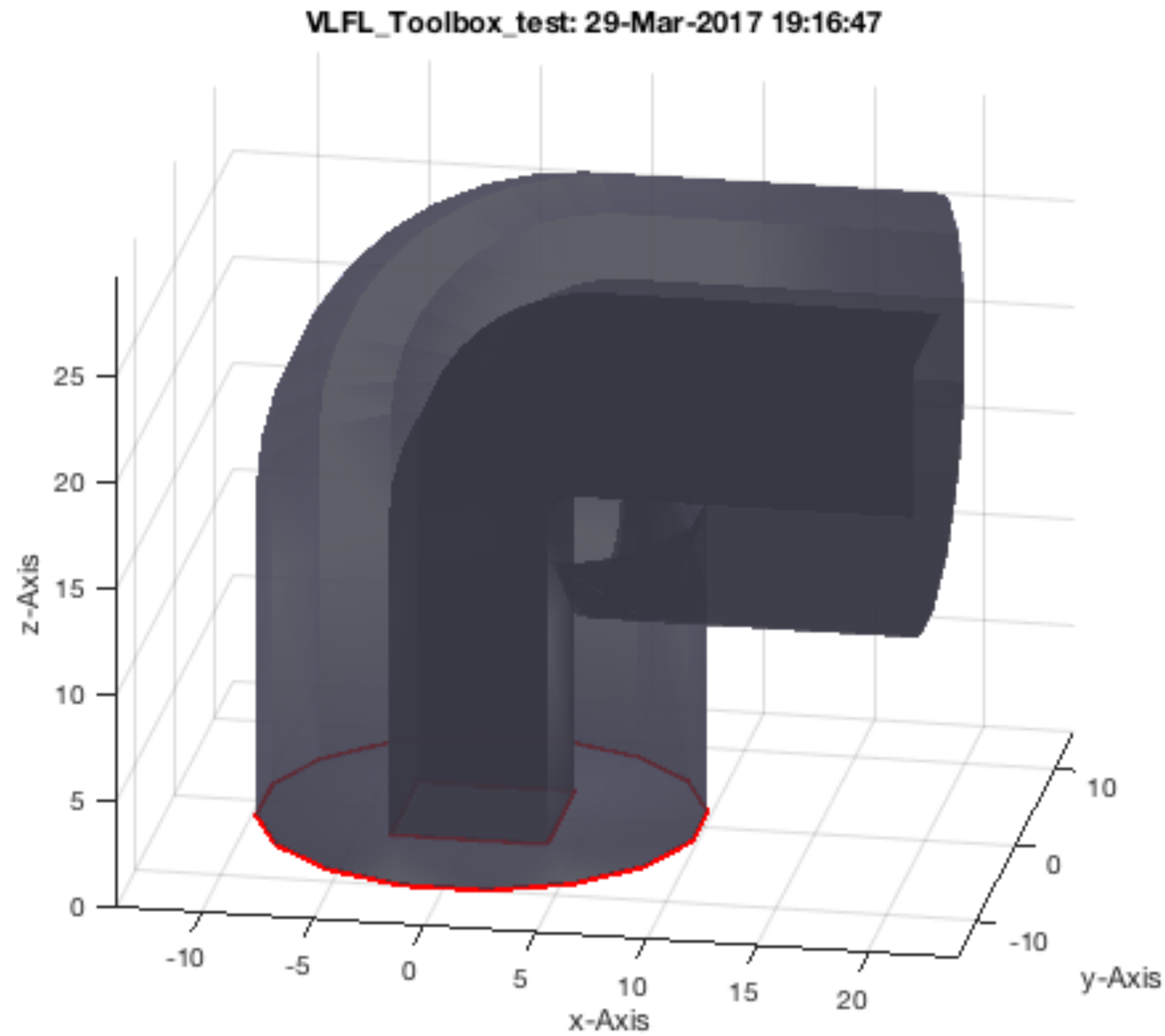
```
Warning: 1st Euler angle does not fit to vertex list path direction
Warning: Last Euler angle does not fit to vertex list path direction
Warning: 1st Euler angle does not fit to vertex list path direction
Warning: Last Euler angle does not fit to vertex list path direction
```



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);
```

```
tangentcirc: Radius 5.00 reduced to 4.76
```

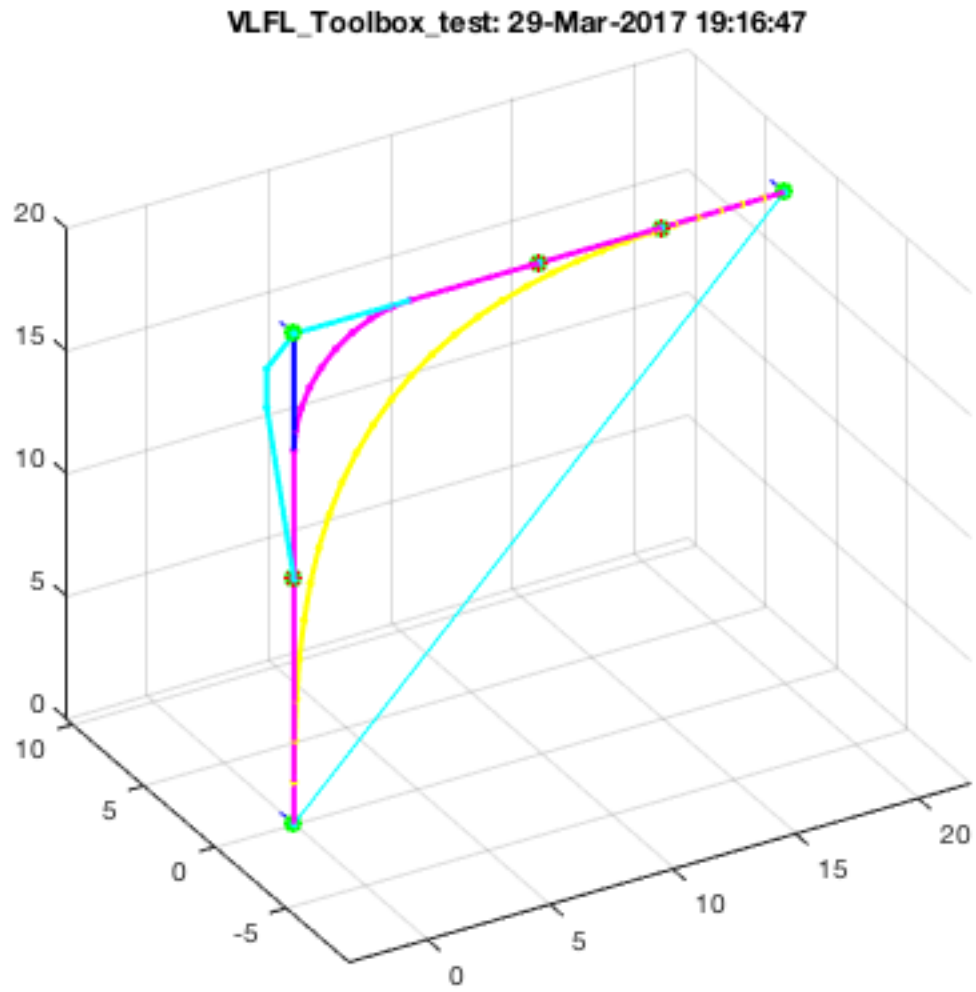


3. Tube generation using a 3D path with Bezier-curves or radial edges

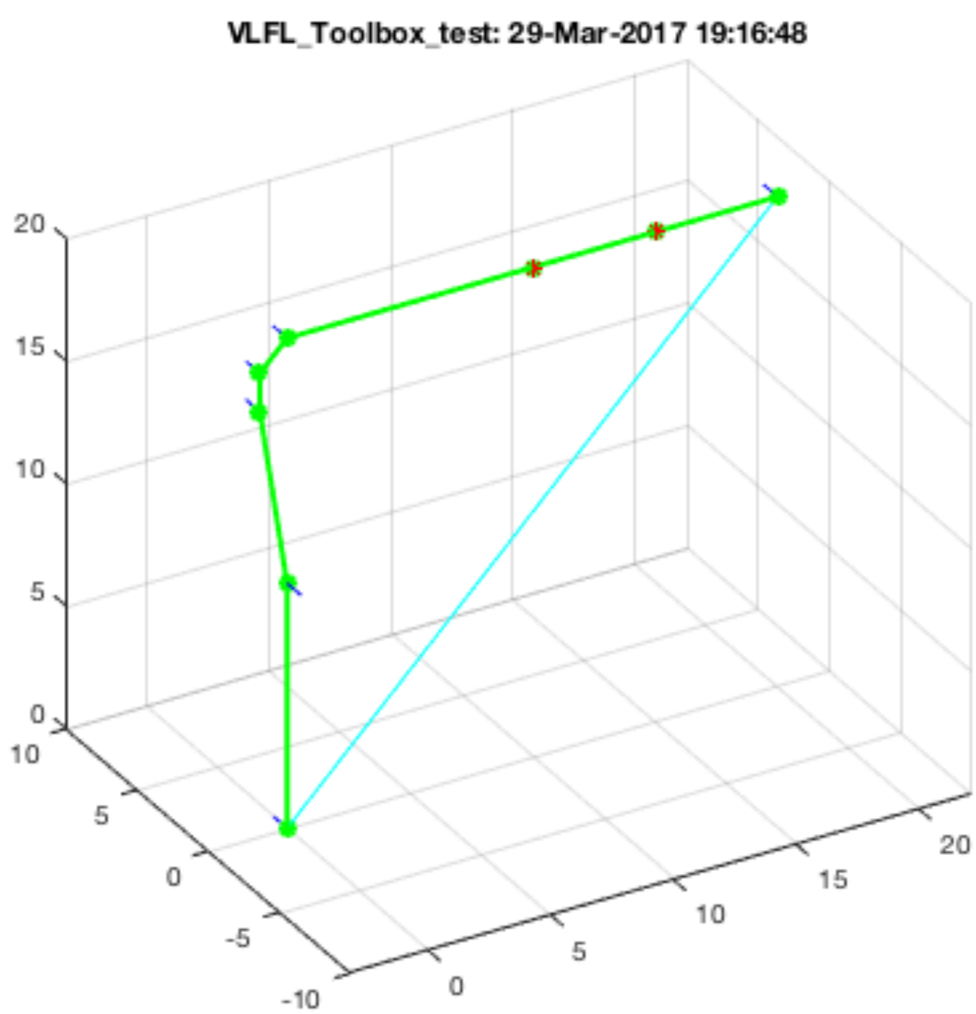
Create a 2D closed polygon line to by copied in 3D space

```
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);
```

```
tangentcirc: Radius 5.00 reduced to 4.76
```

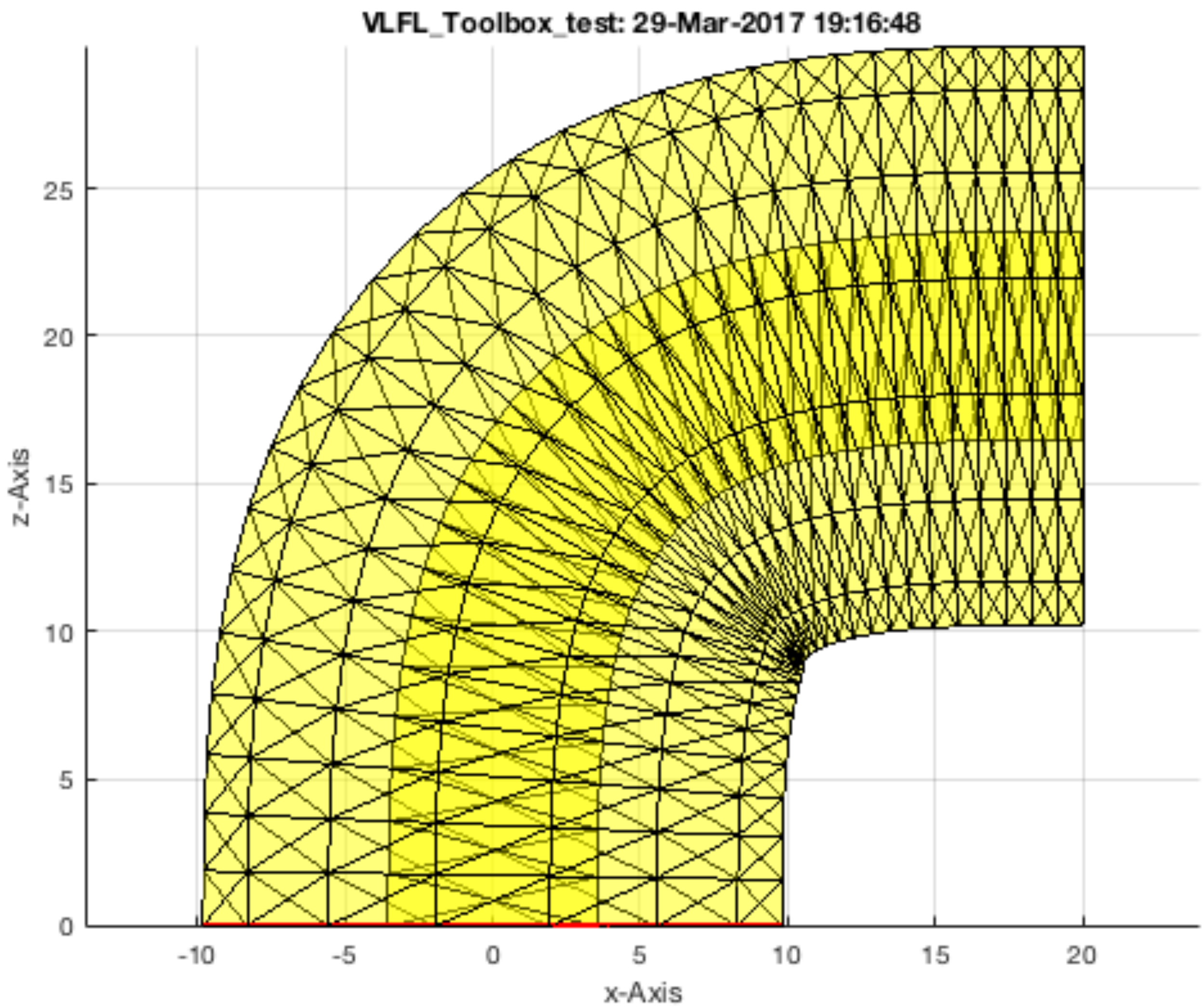



```
VLangle(VLR);
```



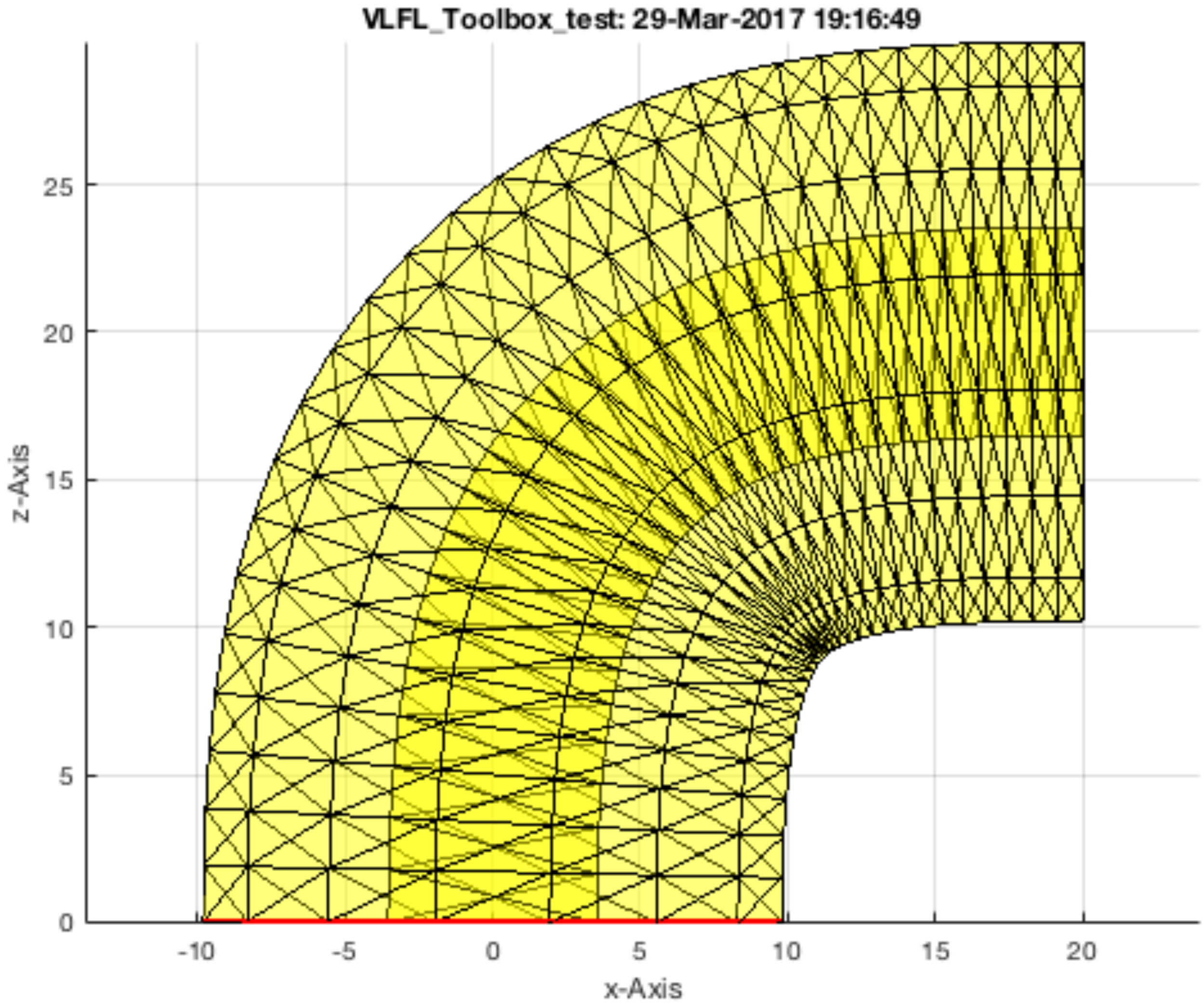
The Bezier-curve tube

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



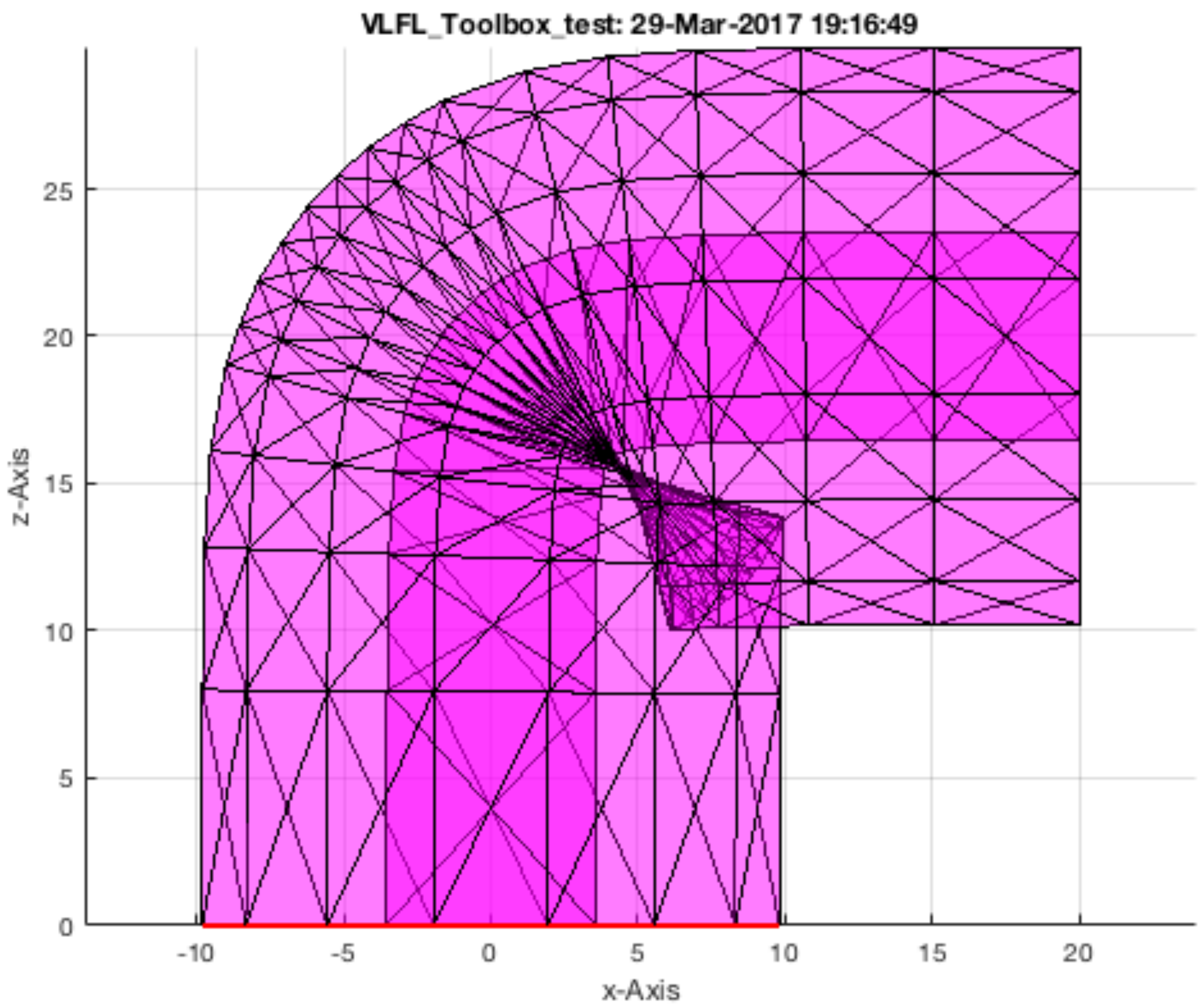
The Bezier-curve tube

```
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

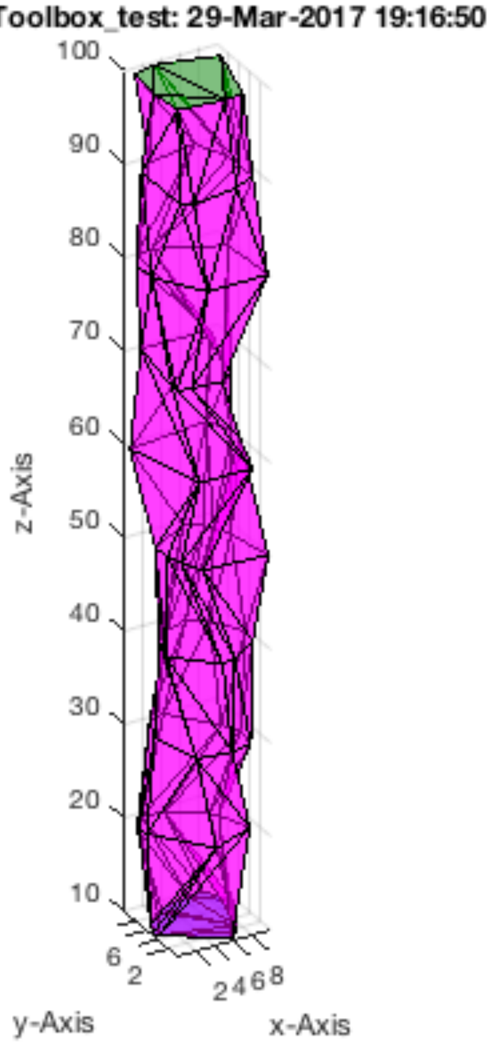
```
SGfigure; CPLplot(CPL,'r-',2);  axis on ; grid on;
SG=SGofCPLCVLR(CPL,VLr); SGplot(SG,'m'); VLFLplotlight(0,0.3);view (0,0);
```



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

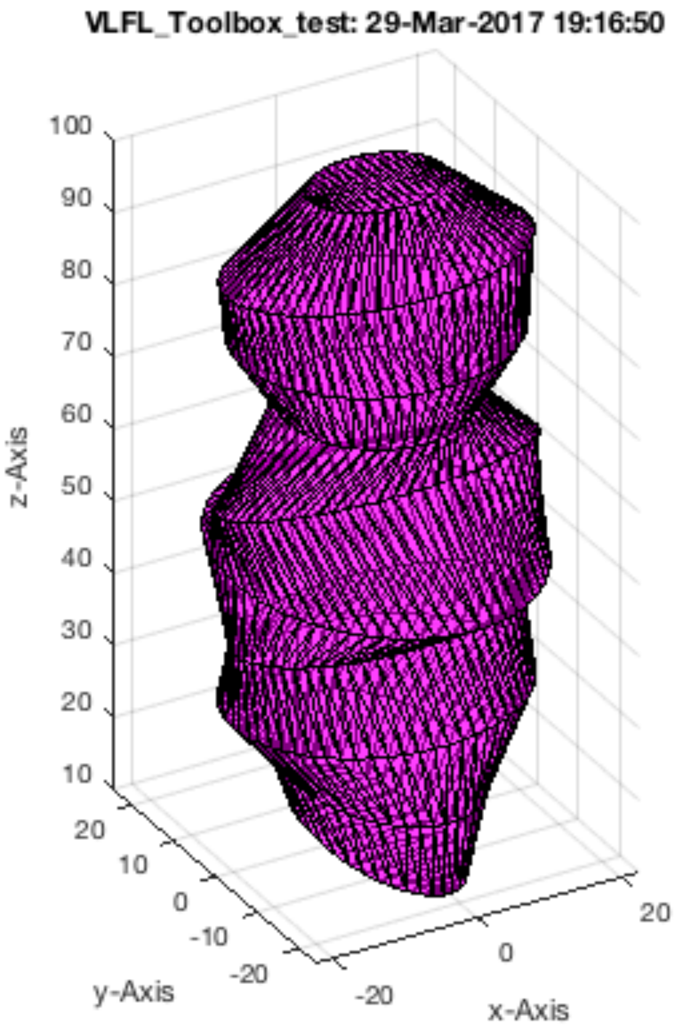
The connection of contours in different z-values works currently only with ONE contour per z-value

```
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)
```



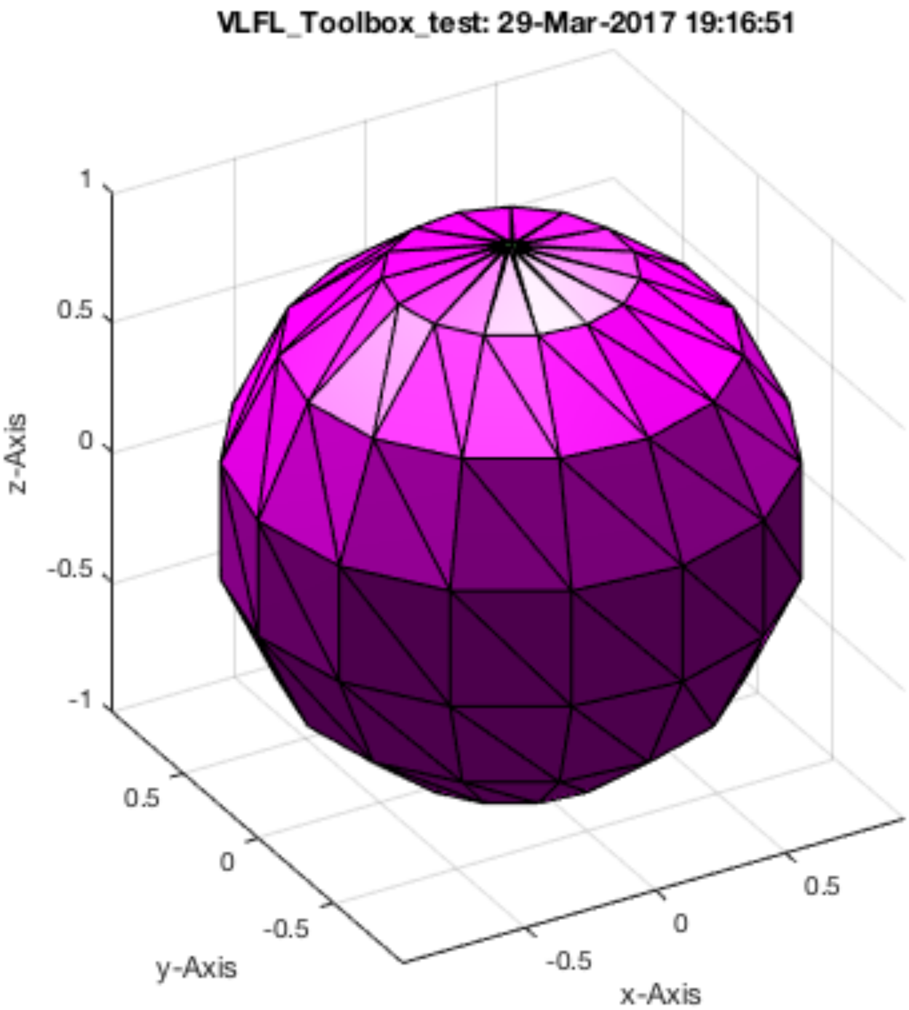
5. Creating a sphere with minmal number of points

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)
```

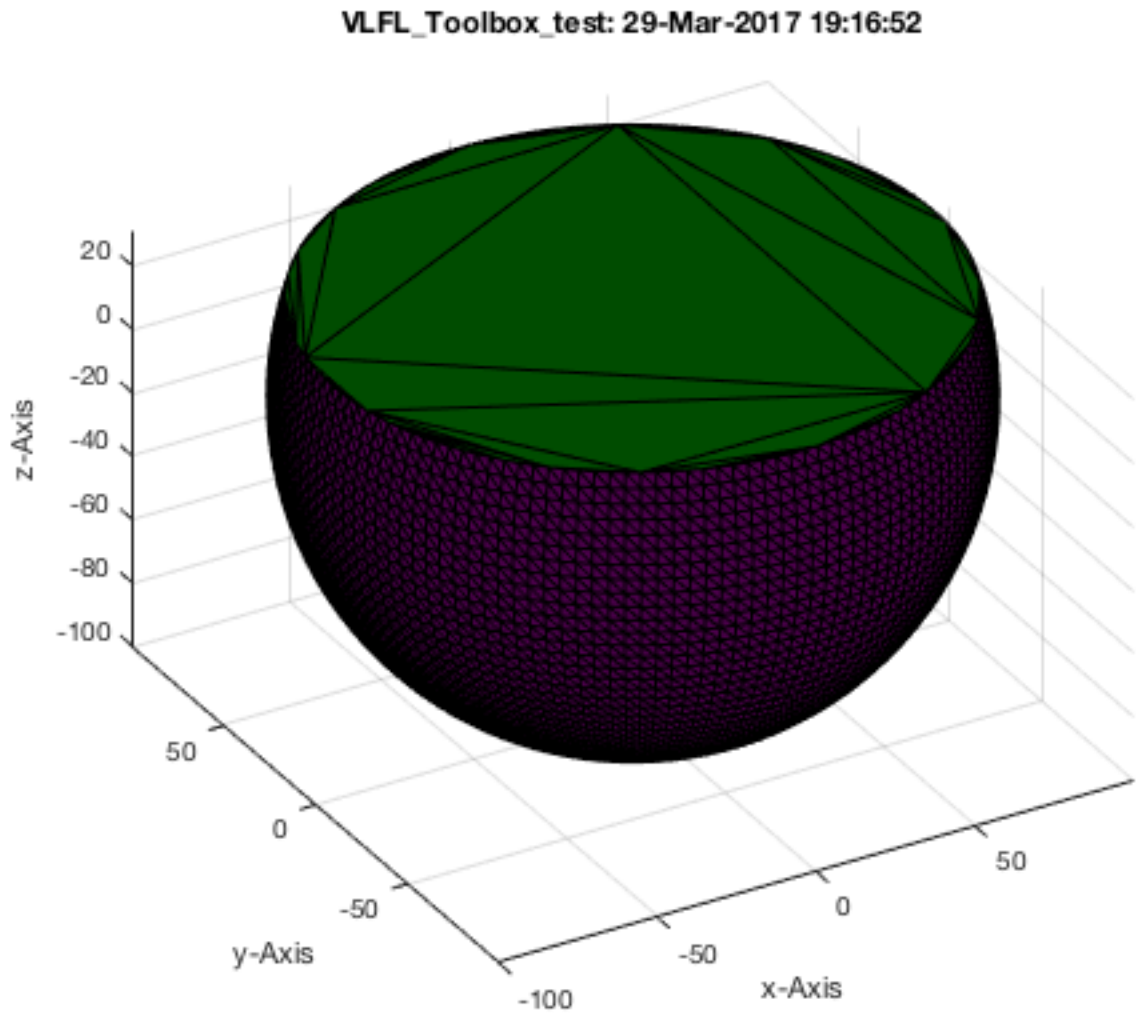
```
ans =

  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)
```

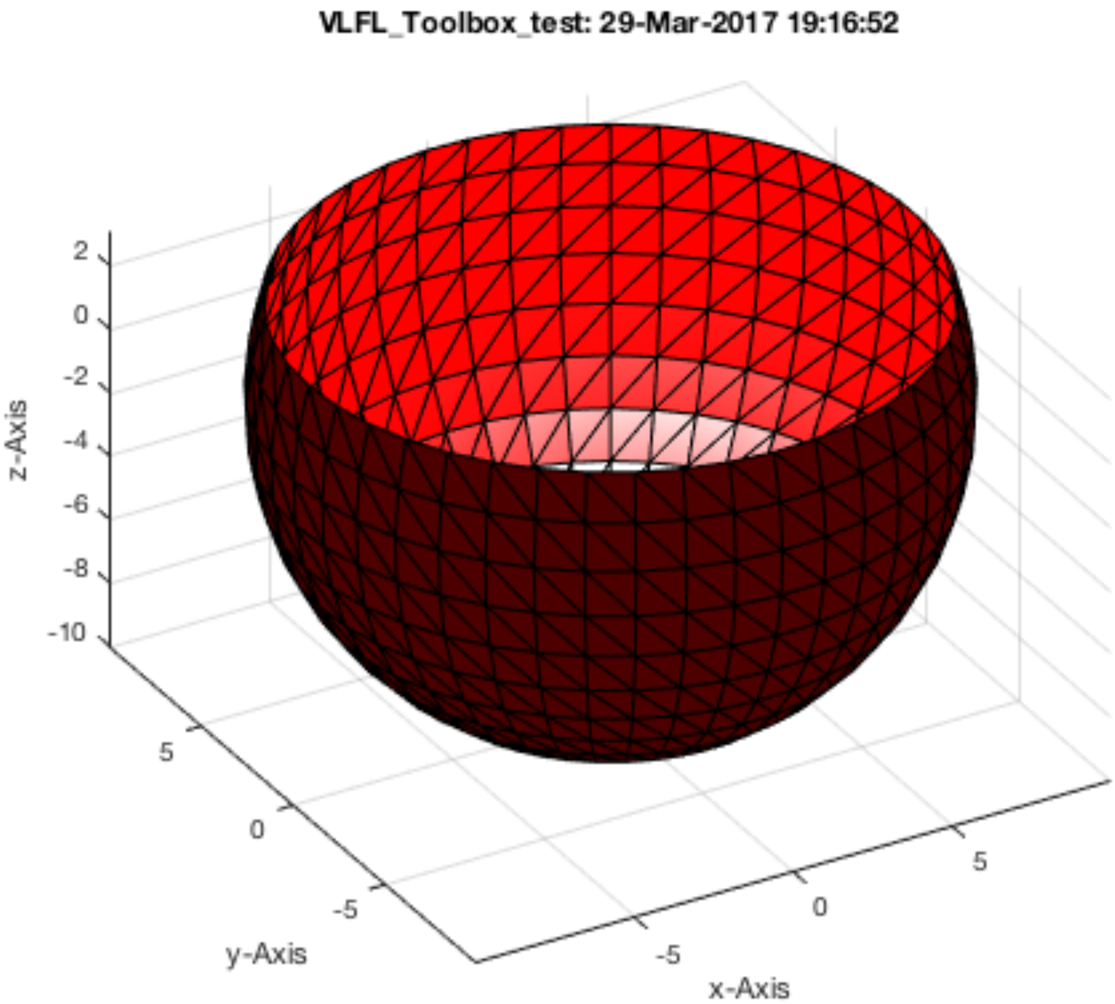
```
ans =

  struct with fields:

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



6. Creating a spherical joint

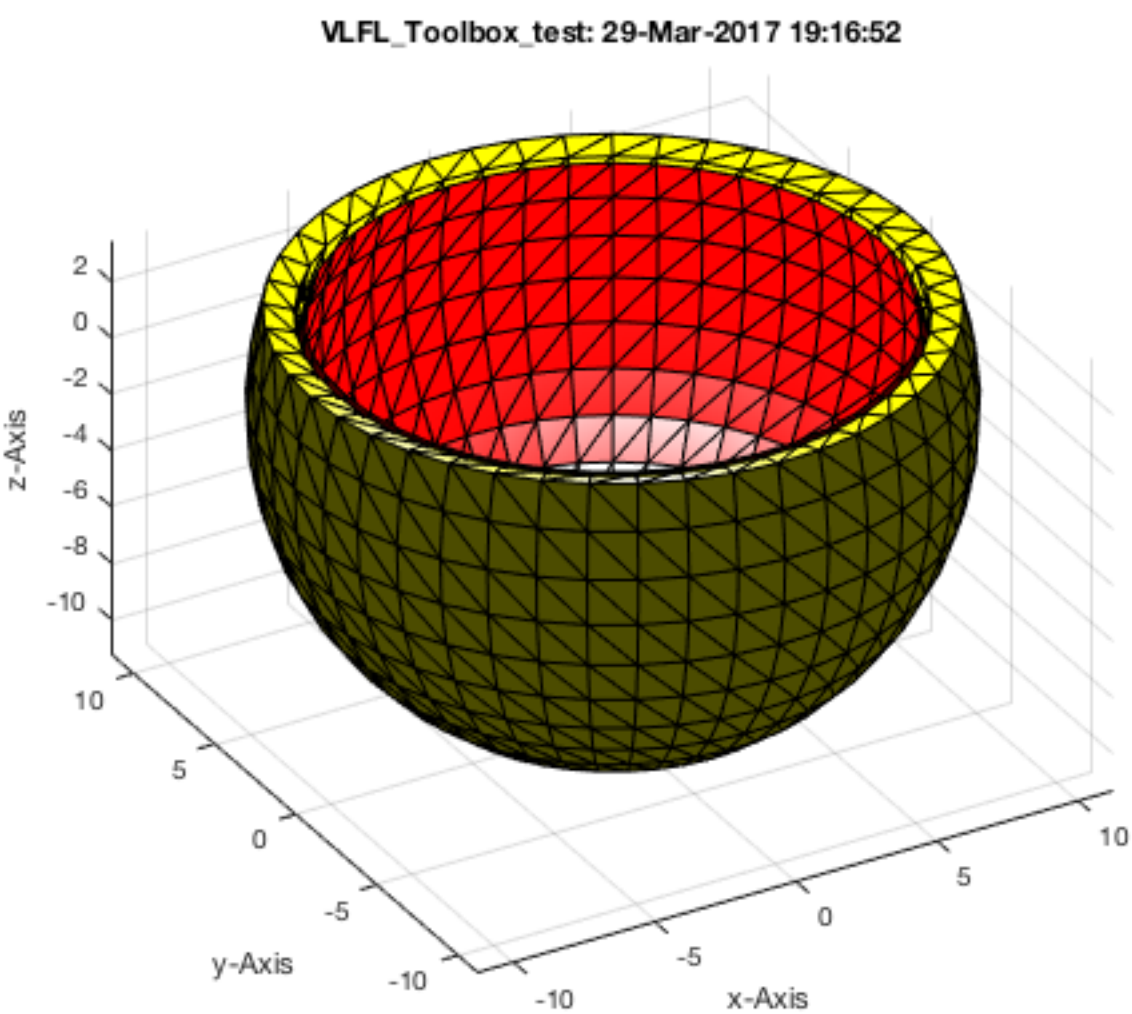
First we have to create a sphere and separate the spherical surface

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



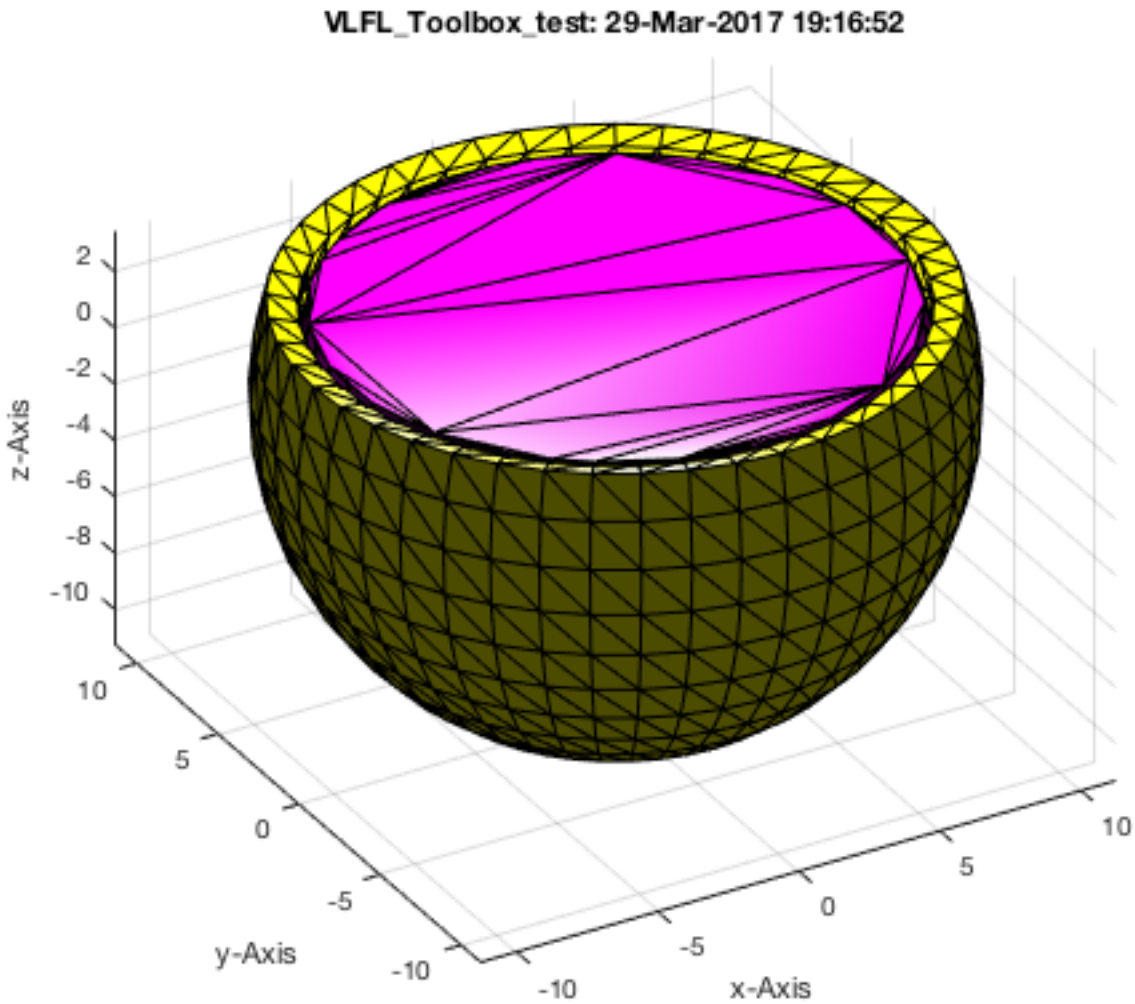
Now create the surface for the joint from the spherical surfacewith tickness 1 as bearing

```
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:16:56 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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_

Exercise 17: Filling and Bending of Polygons and Solids

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

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- Motivation
- 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

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

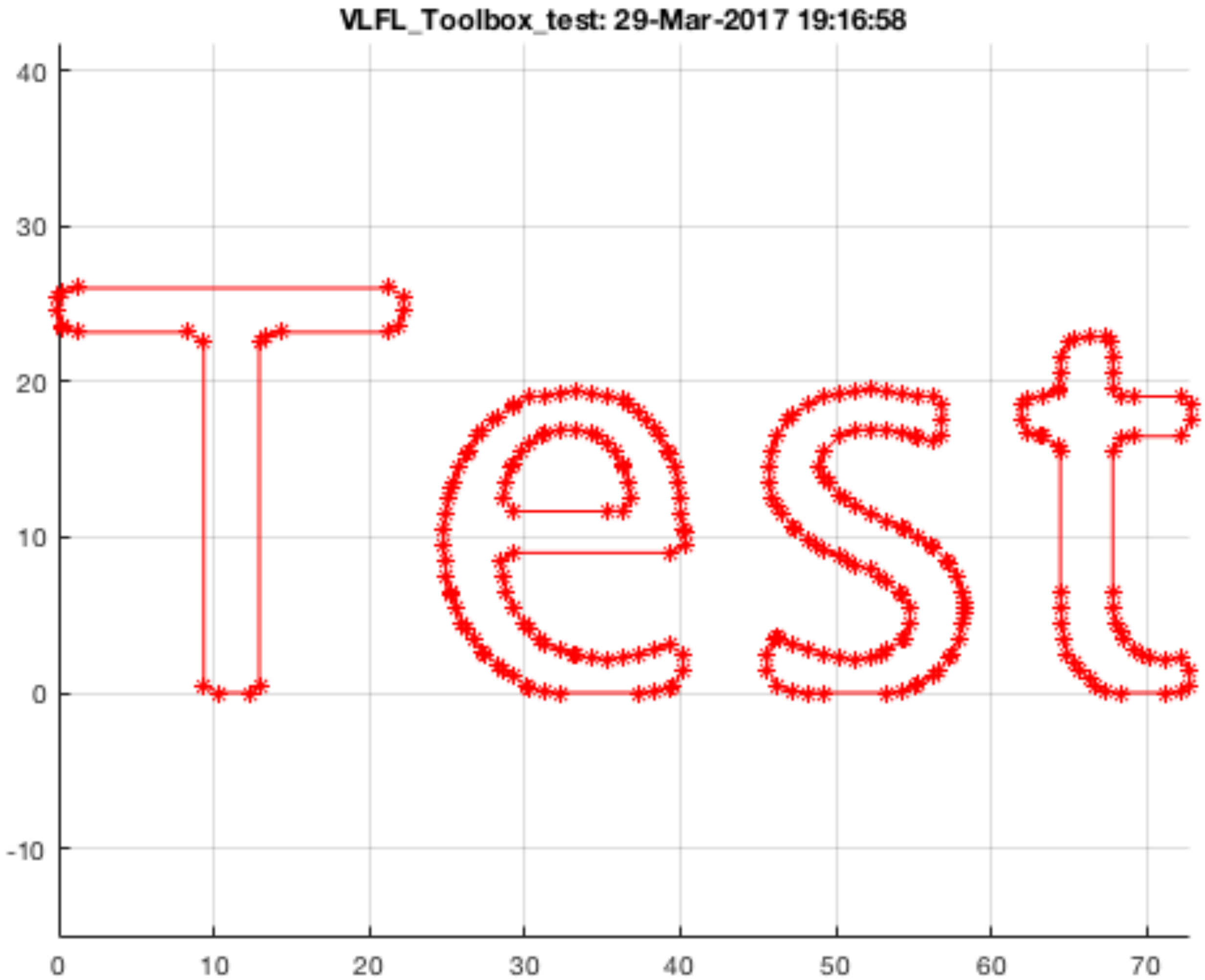
- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create Tube-Style Solids by Succeeding Polygons
- Example 17: Filling and Bending of Polygons and Solids

Motivation

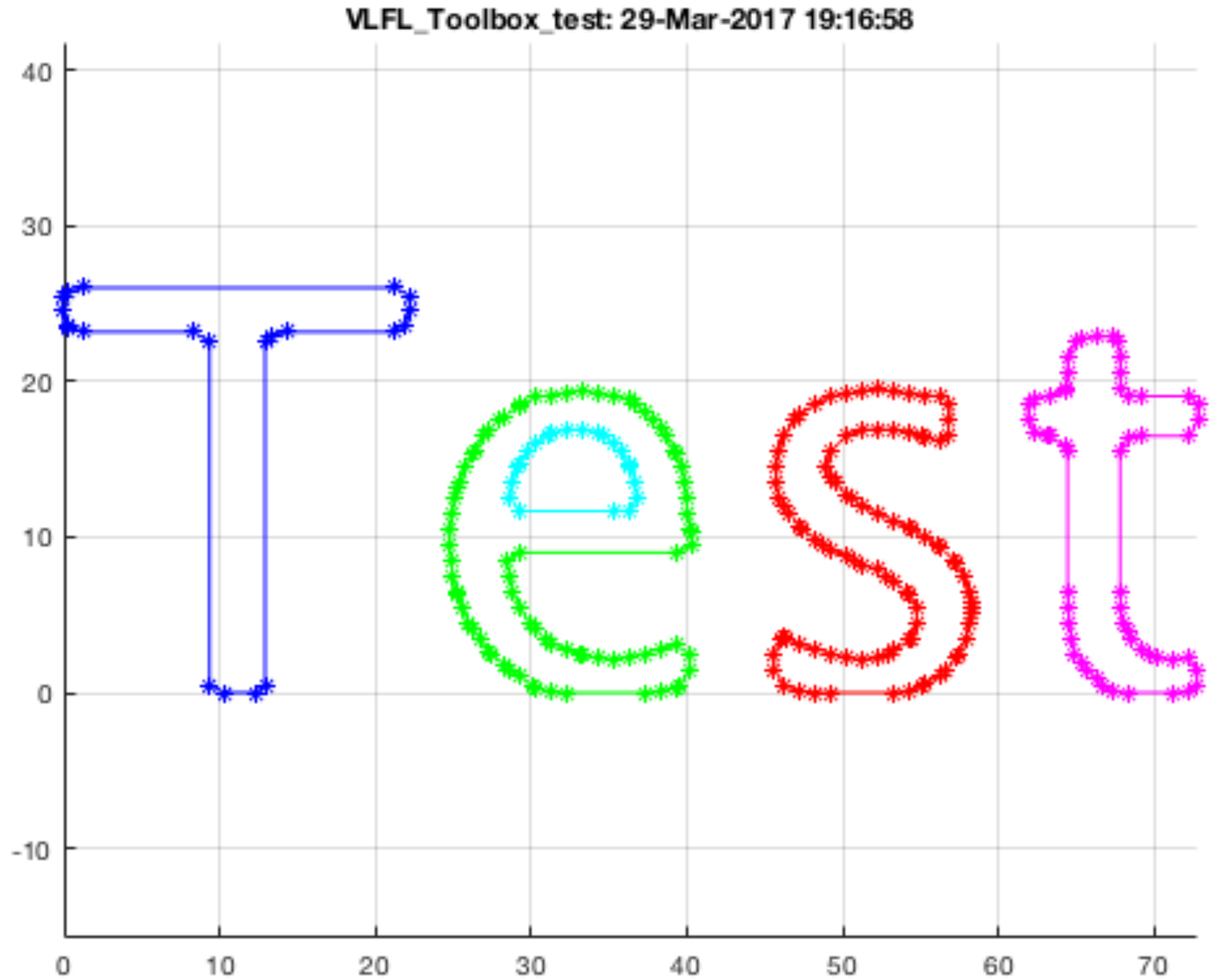
For robotics design,very often we have a wish to extrude a CPL not only in an orthognal z direction to the xy-plane, but in an any desired direction even along a path in 3D space. Intutively we expect a result, but this is not easy to achieve automatically. Anyway, for those tasks we have two functions:

1. Creating Closed Polygon Line

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

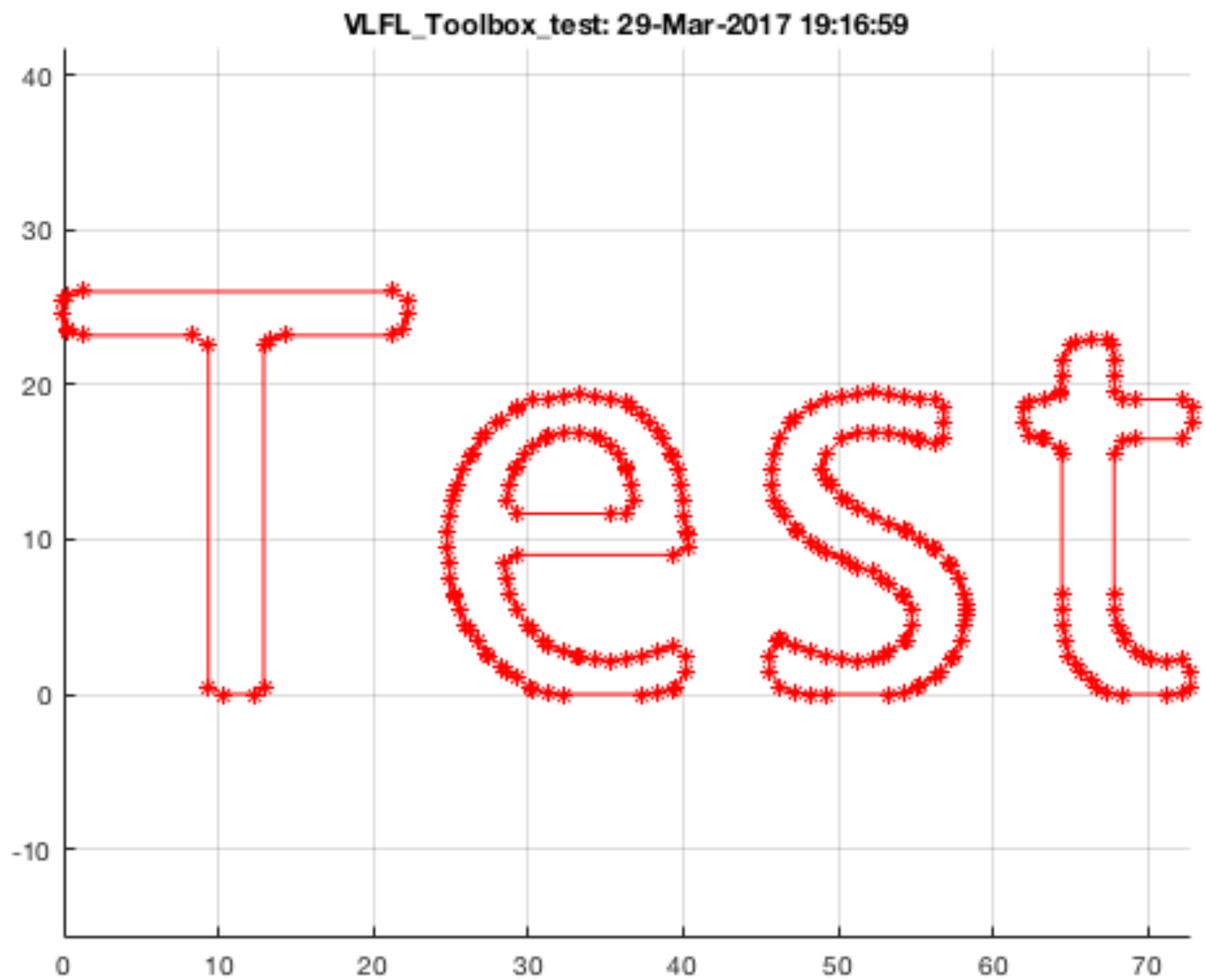



```
SGfigure; view(0,90); CVLplot(CPL);
```

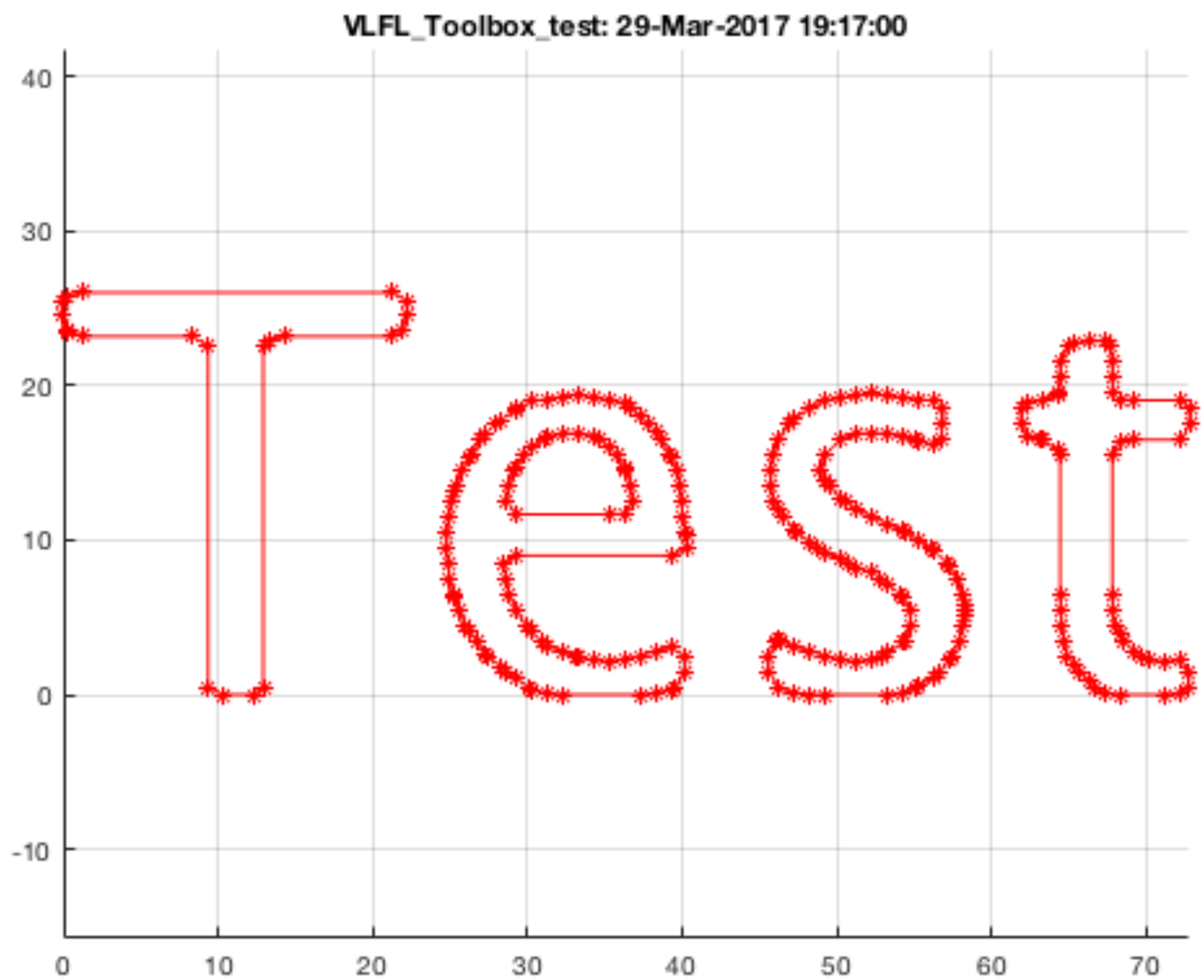


2. Converting CPL into PL EL

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

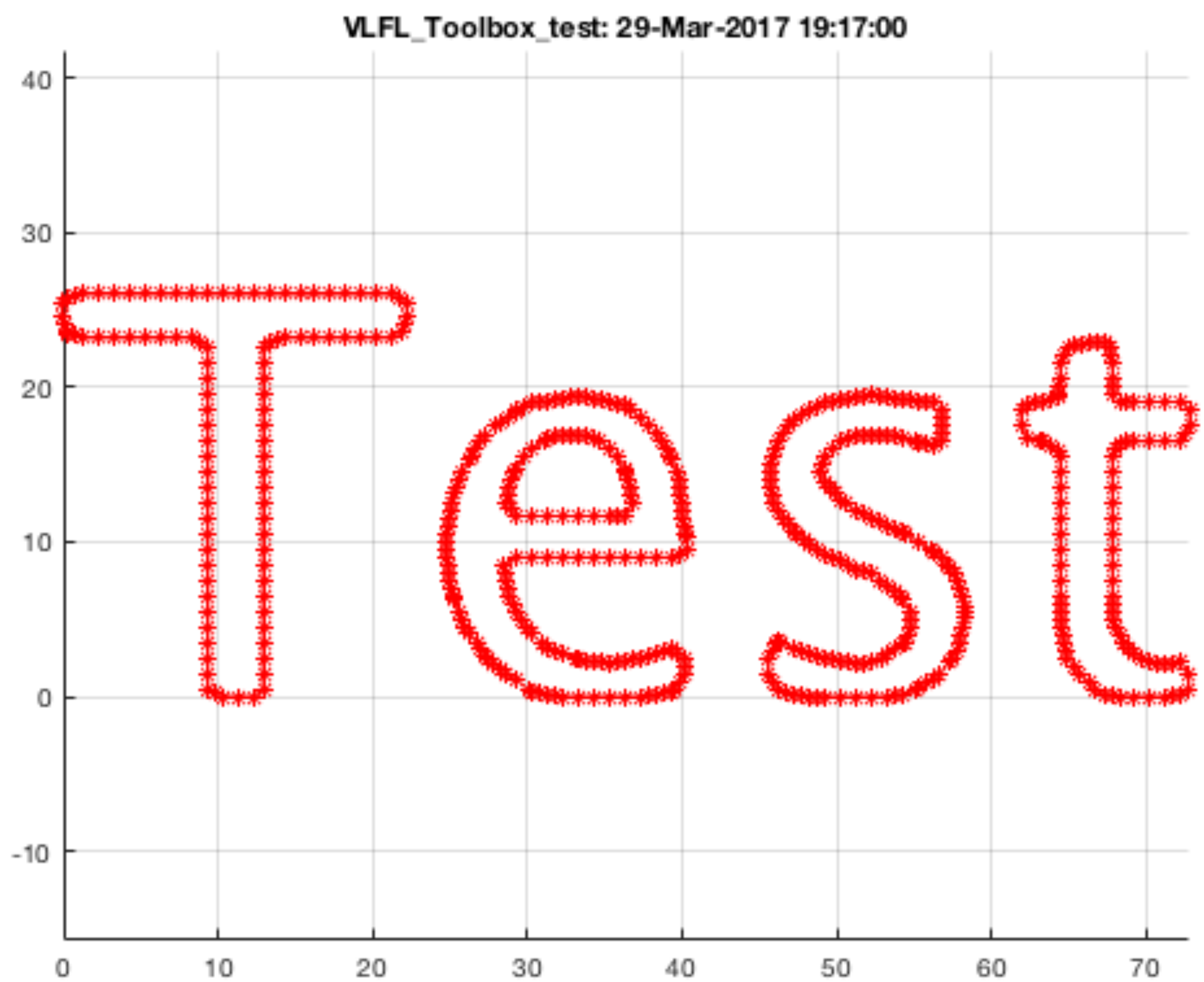



```
CPL=CPLofPLEL(PL,EL);
SGfigure; view(0,90); CPLplot(CPL);
```

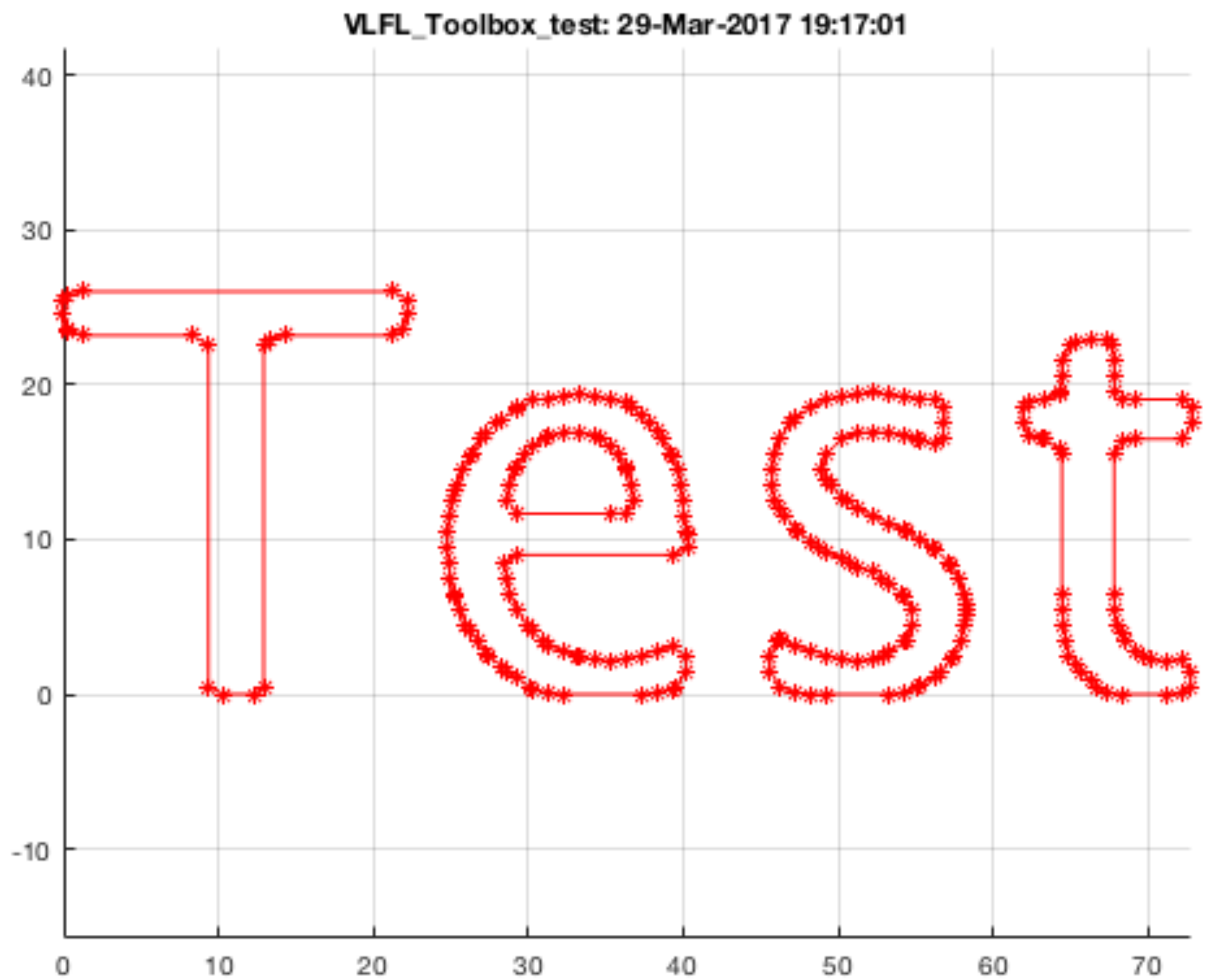


3. Adding and removing points on the contour

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

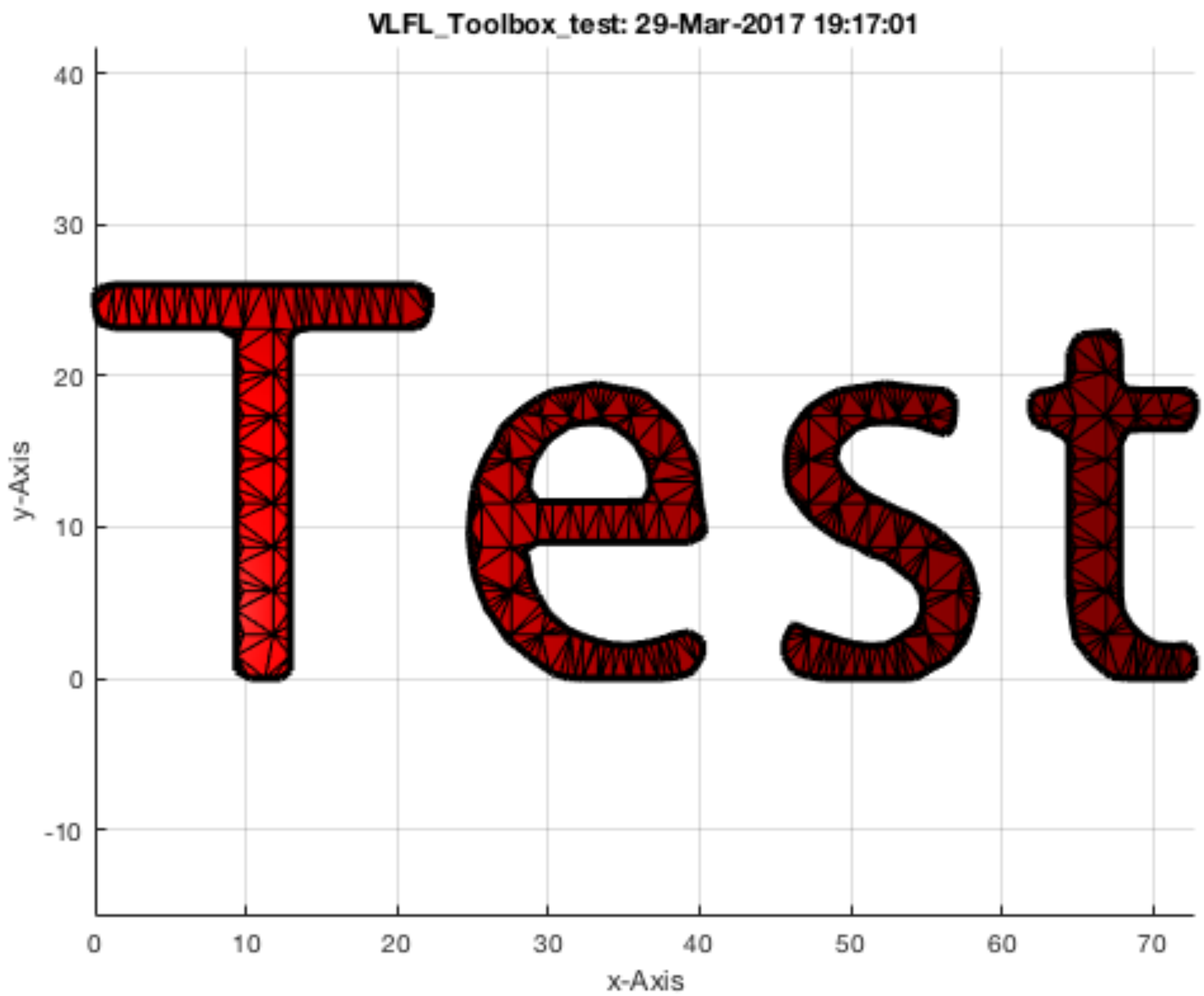



```
CPLB=CPLremstraight(CPL);
SGfigure; view(0,90); CPLplot(CPLB);
```



4. Adding and removing points inside of the contour

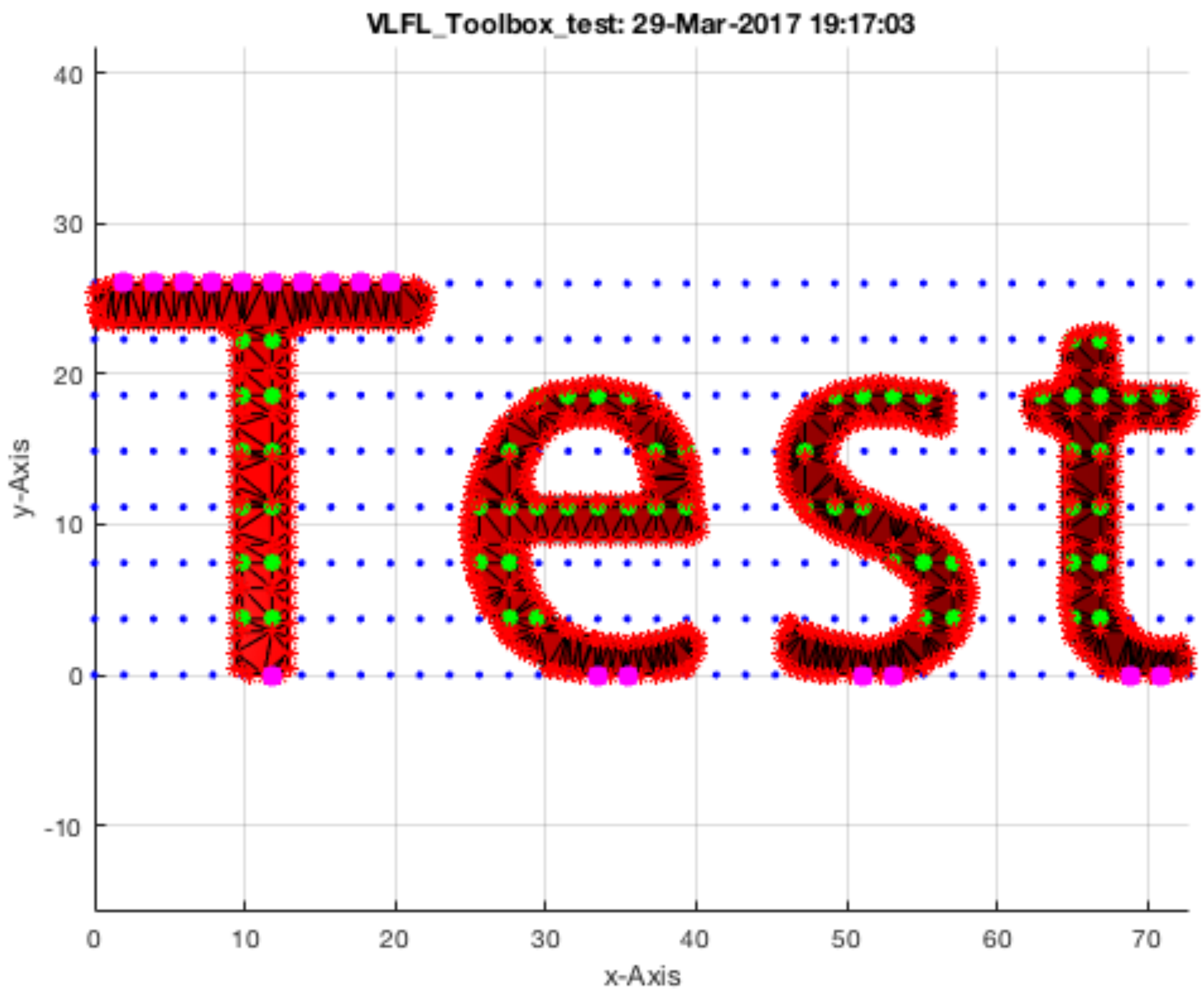
```
[PL,FL,EL]=PLFLofCPLdelaunayGrid(CPL,1,2,3);
SGfigure; view(0,90); VLFLplot(PL,FL); VLELplots(PL,EL,'k',3);
```



5. Calculate Grid Points

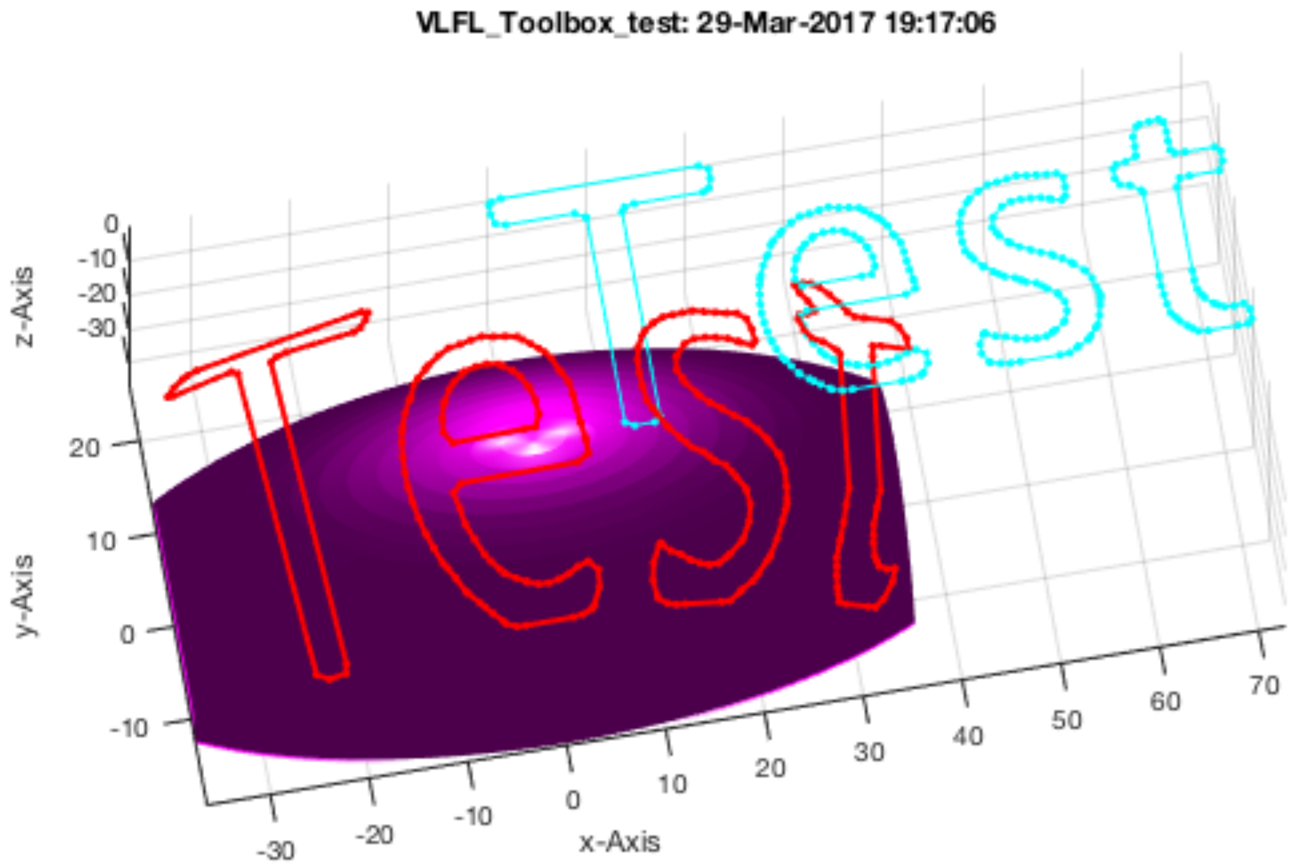
```
GPL=GPLauxgridpointsPLEL(PL,EL,2,4);
insidePLELdelaunay(PL,EL,GPL);
```

```
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 delaunayTriangulation property X.
```

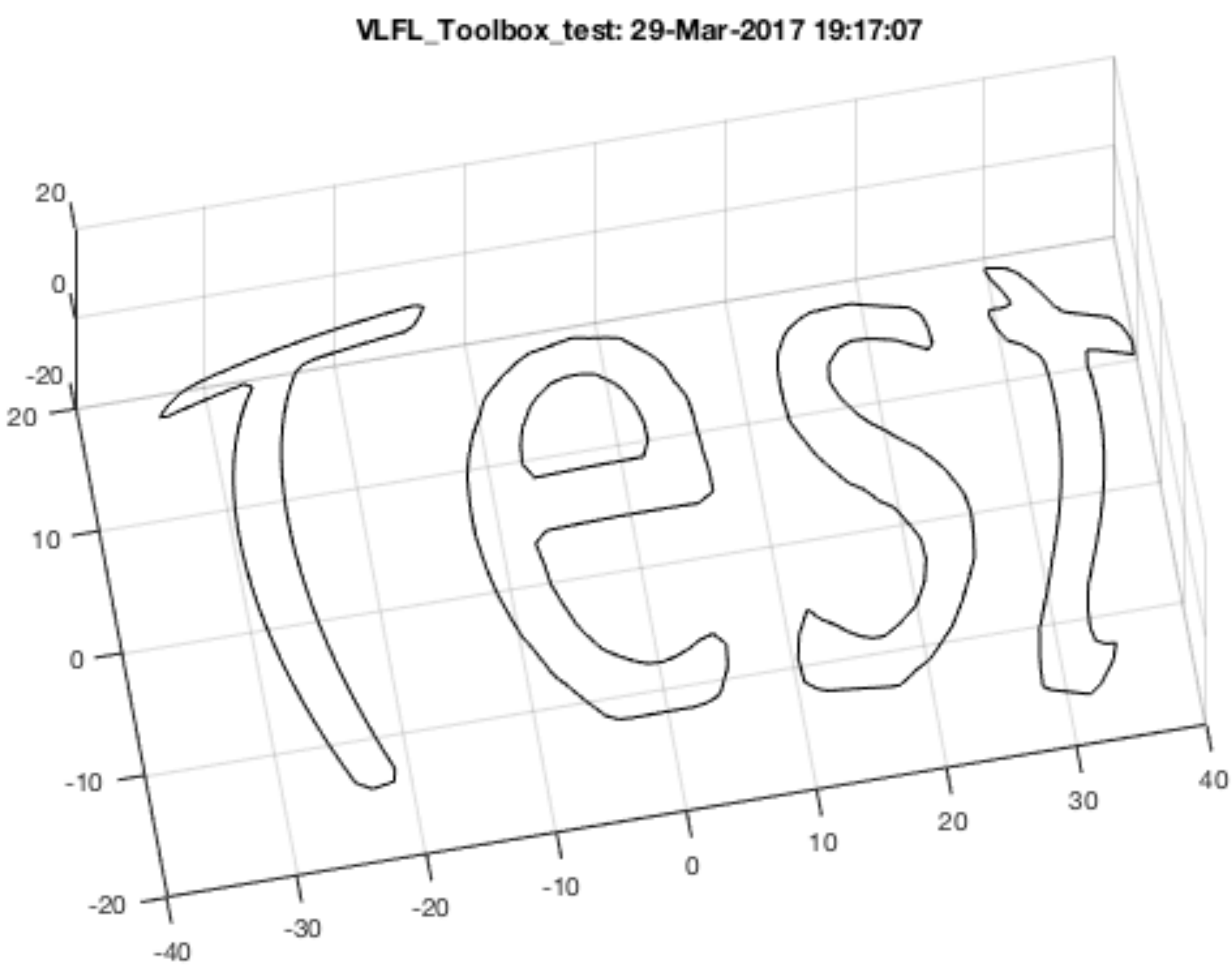


6. Bending of contours

```
% Without auxiliary points

BPL=PLbending(CPL,50,10,20);
SGfigure; view(-30,30); PLbending(CPL,50,10,20);

% With auxiliary points
```

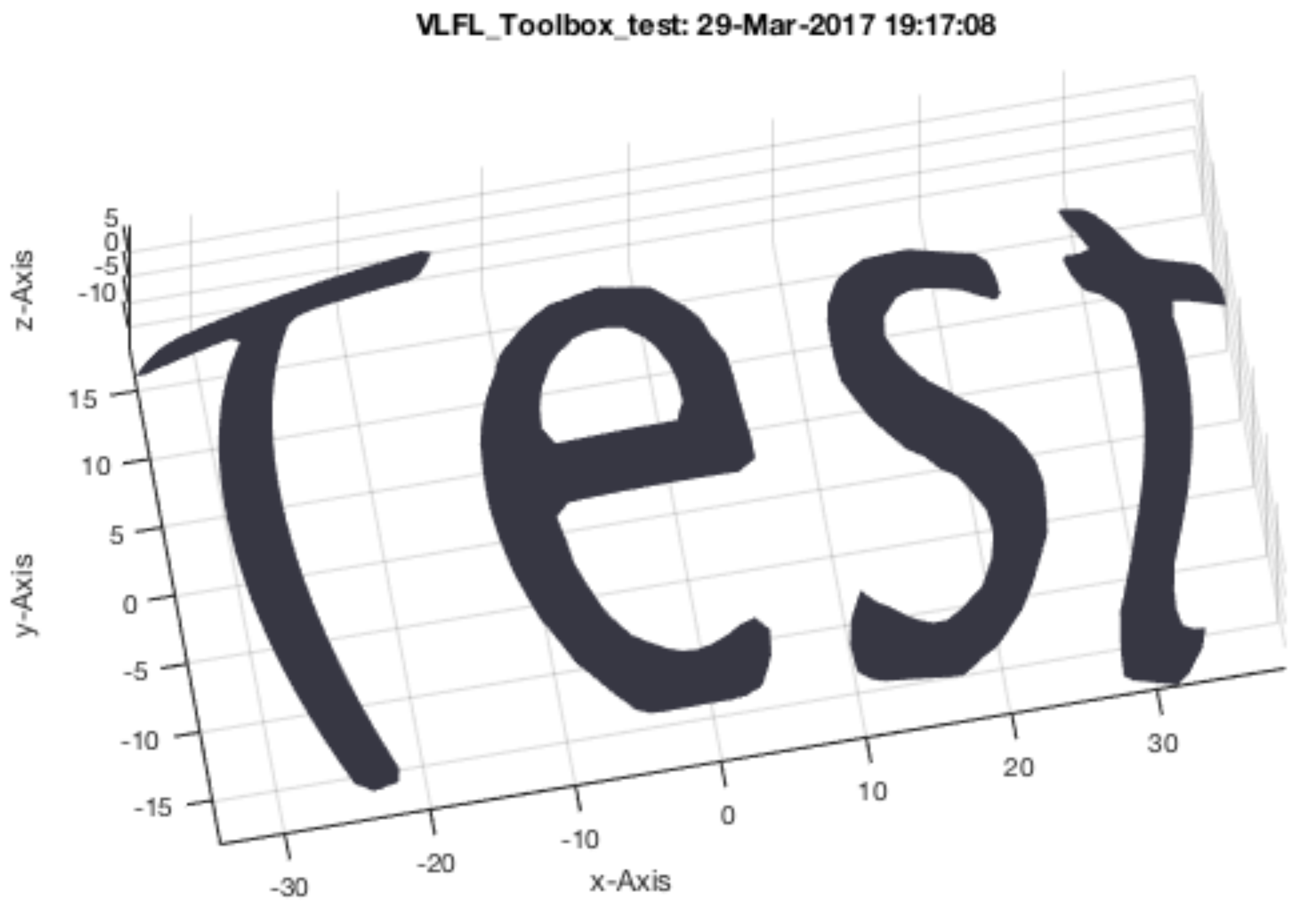



```
[PL,FL,EL]=PLFLofCPLdelaunayGrid(CPL,1,2,3);
NPL=PLbending(PL,50,10,20);
SGfigure; view(-10,70); VLELplot(NPL,EL,'k-');
```



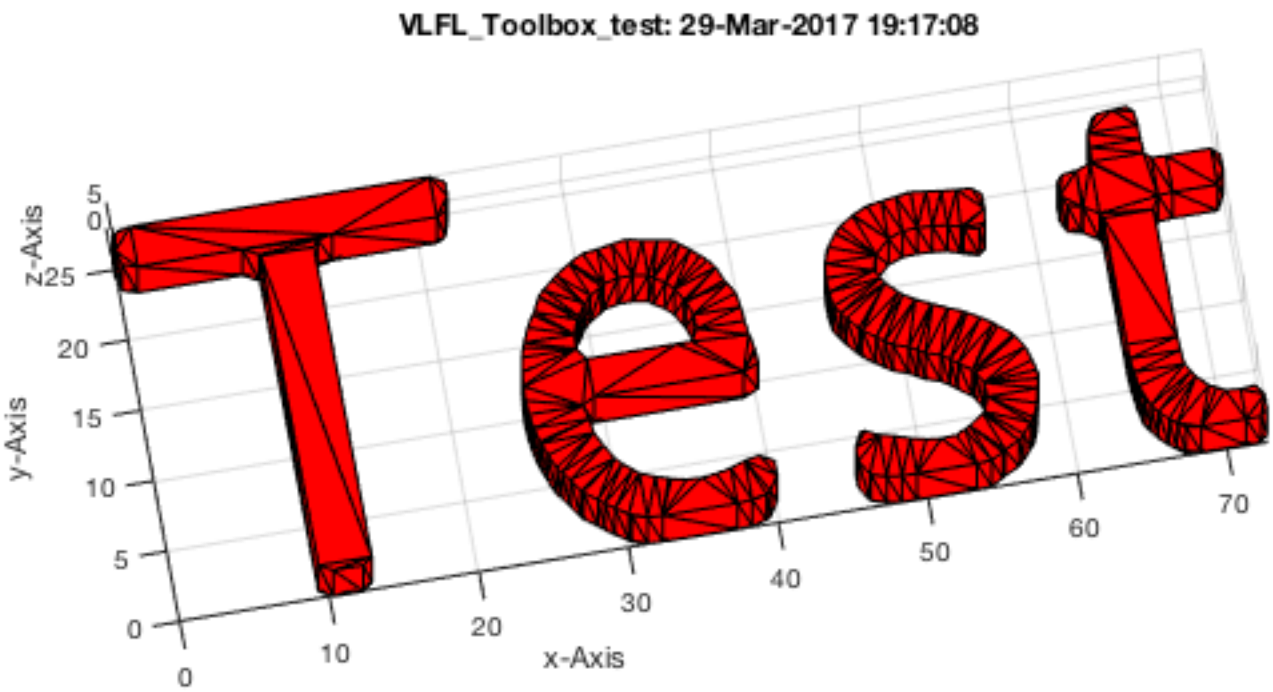
7. Bending of closed contour surfaces

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

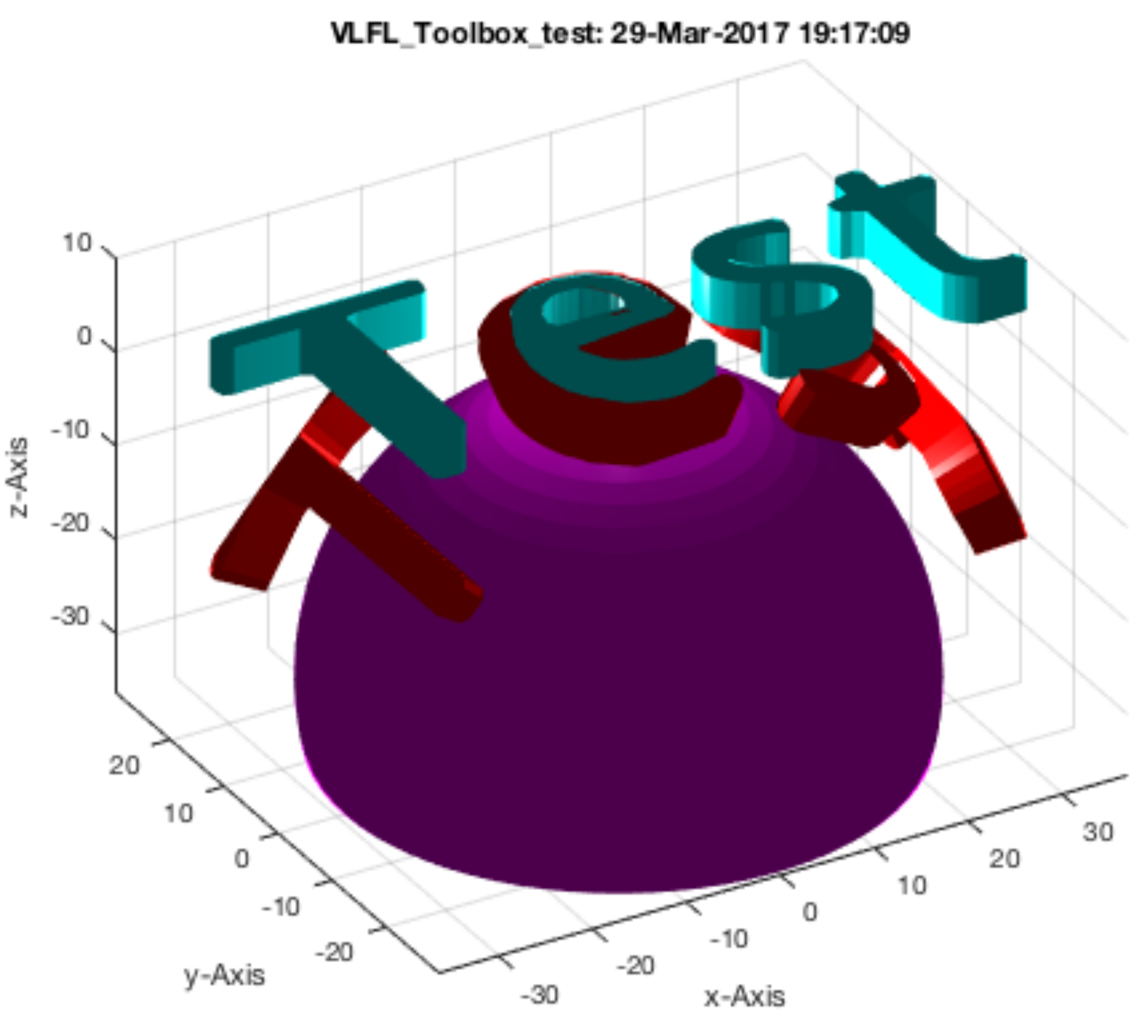


8. Bending of solid geometries

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

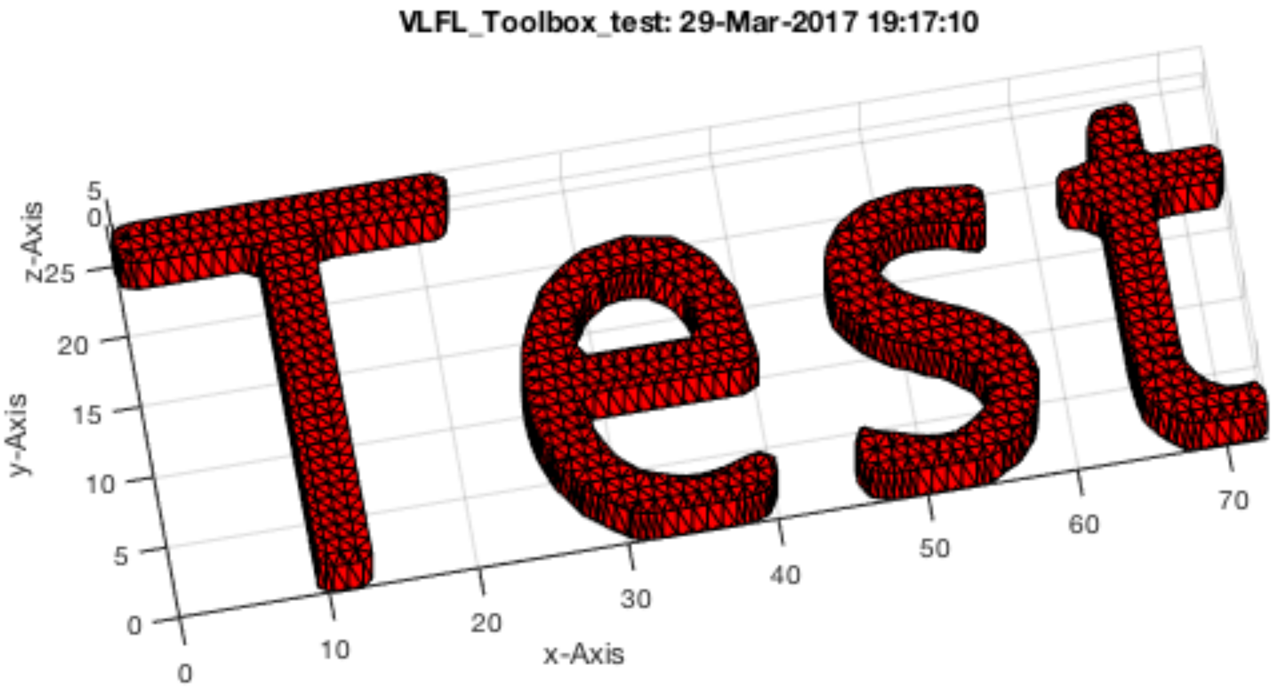



```
SGbending(SG,30,5,30); VLFLplotlight(1,1);
```

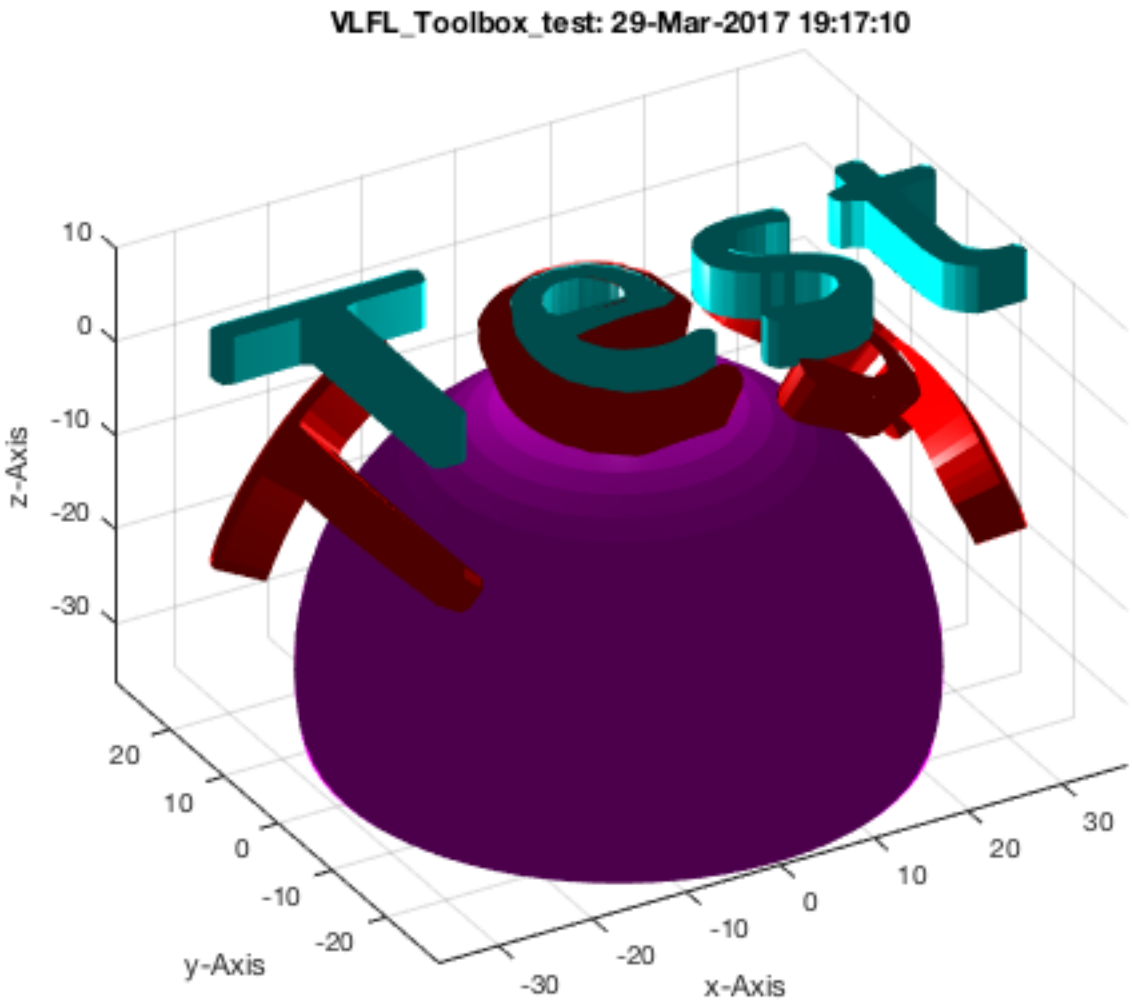



```
% With auxiliary points
```

```
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:17:13 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

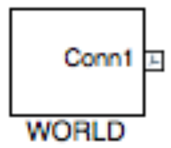
The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmatically Interface to SimMechanics Multi-Body Toolbox

2. Creating a new SimMechanics System

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

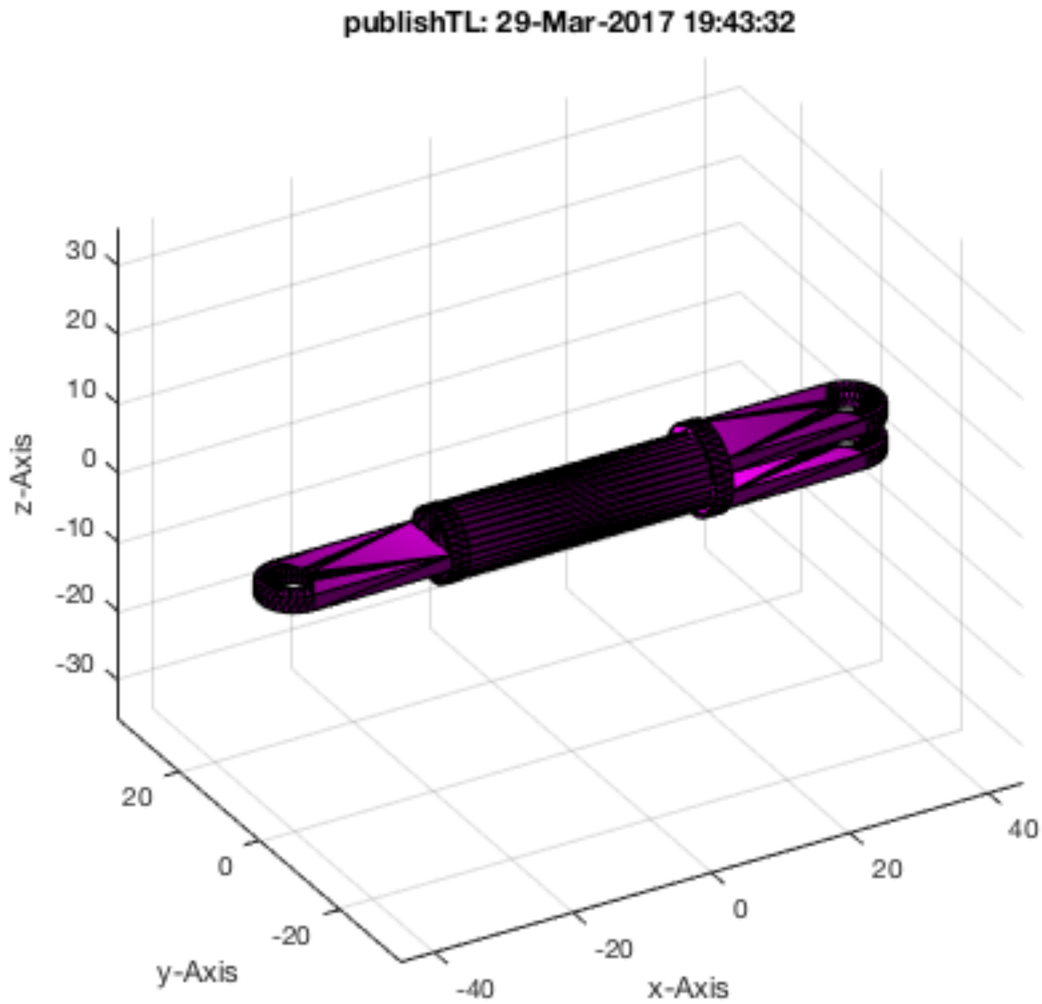
```
Creating temporary directory '/Users/lueth/Desktop/tmp_SG_LIB_EXP_20/'
```



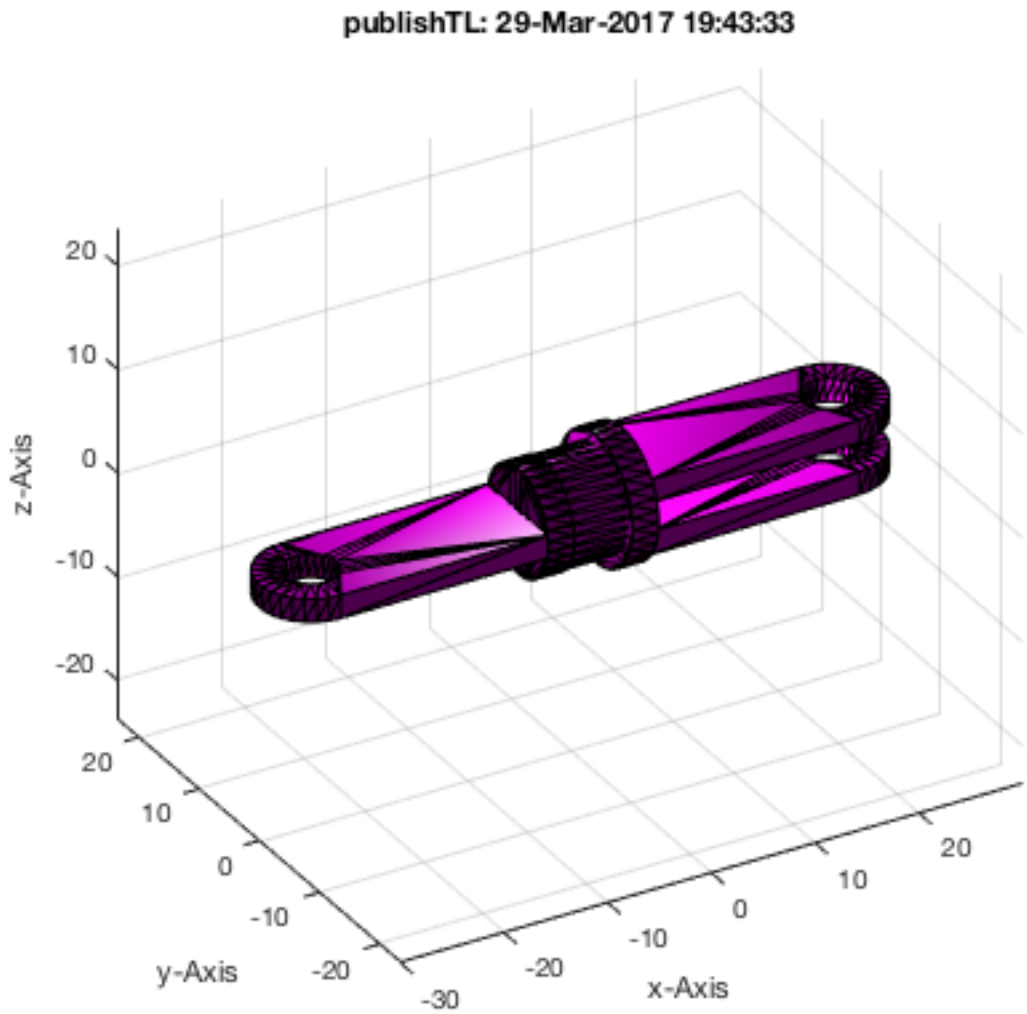
3. Create two links with length 50 and 80 and one or two mounting holes

```
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);
```

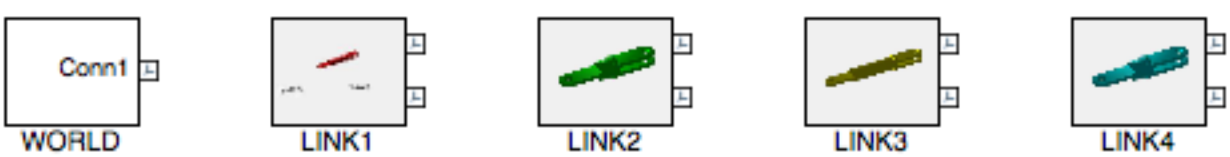



```
SGfigure(SG2); view(-30,30);
```



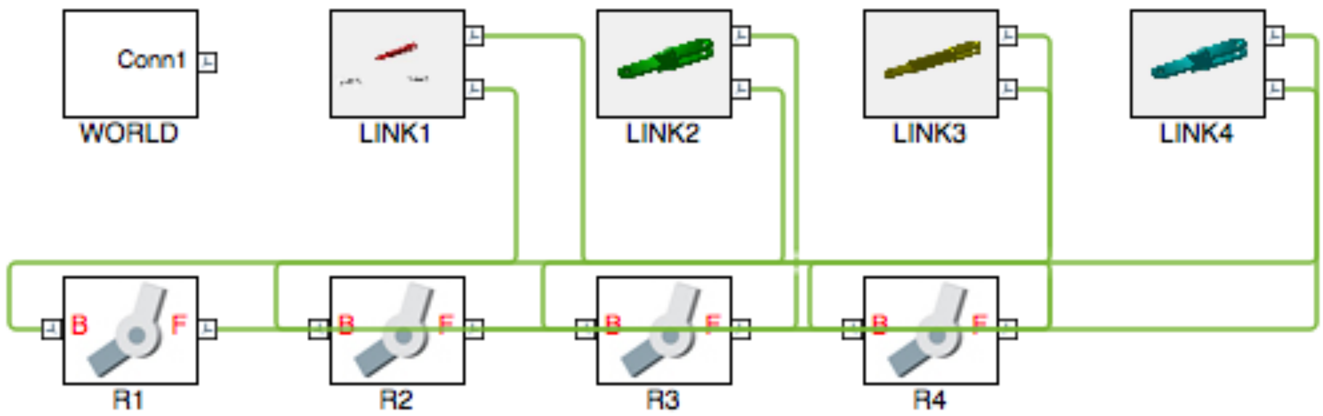
4. Create SimMechanics models for the four links in different colors

```
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;
```



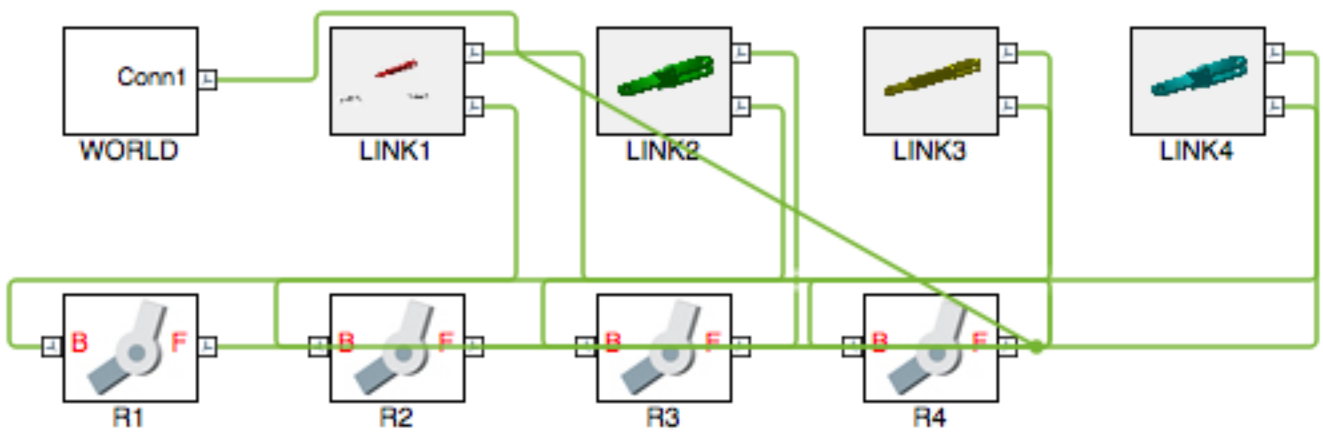
5. Create SimMechanics models for the four joint and connect them with the links

```
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;
```



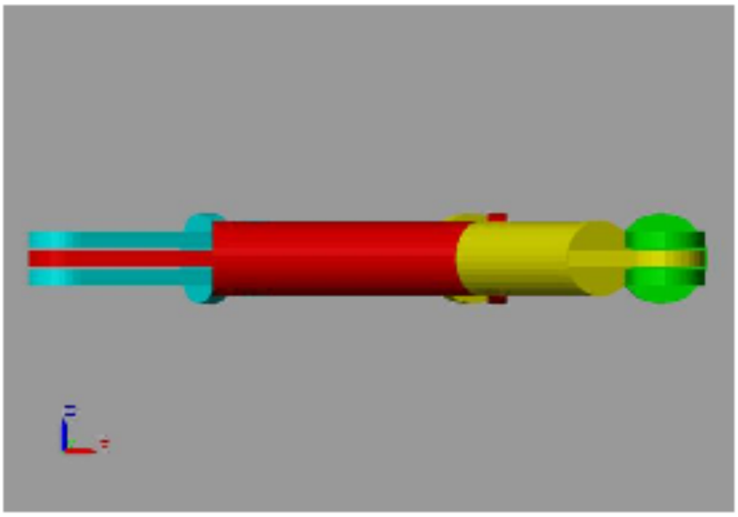
7. Run the Simulation of the Simulink/SimMechanics diagram for 1 second

```
smbSimulate(1);          % Simulate for 1 second
```

8. Create a Video of the Simualatin for 5 seconds

```
smbVideoSimulation(5);  % Show a 5 seconds video
```

....



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:44:00 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ============================
==========
database_toolbox
image_toolbox
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® R2016b

Exercise 21: Programmatically Convert Joints into Drives (SimMechanics)

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

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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. Create a SimMechanics model for a motor/drive and use a Cosinus Rotation
- 8. Create a Simulink models for a cosinus signal
- 9. Create a Video of the Simualati0n for 10 seconds
- Final remarks on toolbox version and execution date

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmatically Convert Joints into Drives (SimMechanics)

2. Creating a new SimMechanics System

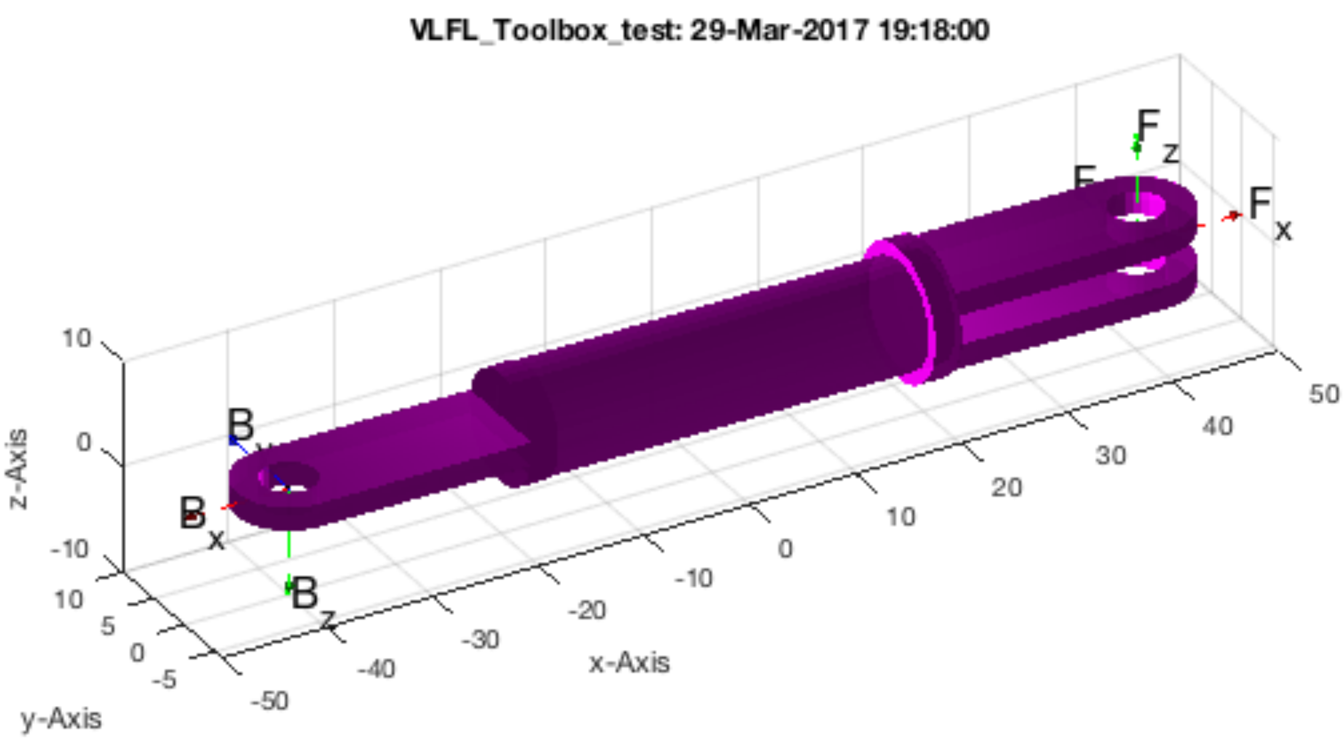
```
smbNewSystem ('SG_LIB_EXP_21');          % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/lueth/Desktop/tmp_SG_LIB_EXP_21/'
```

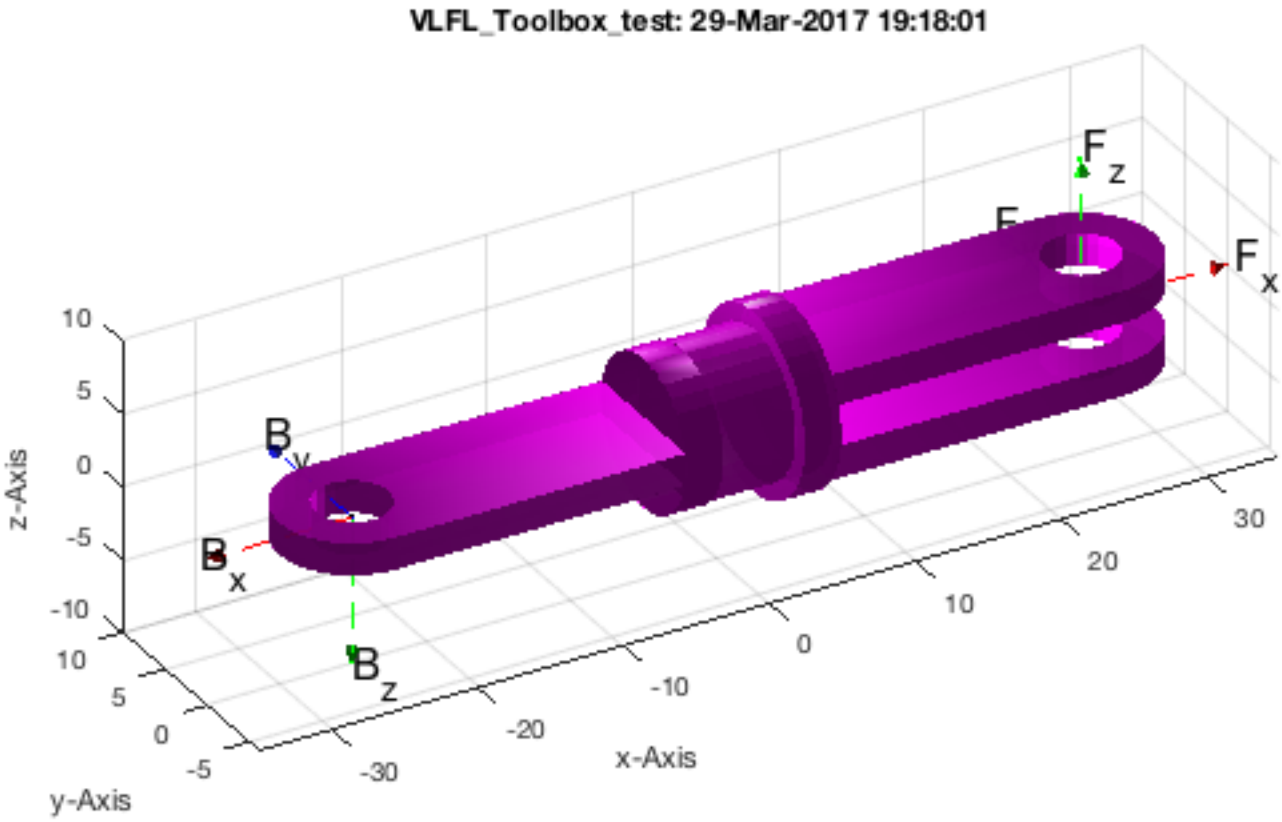
3. Create two links with length 50 and 80 and one or two mounting holes

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

```
SGfigure; view(-30,30); axis on;
SGT(SG1);
```

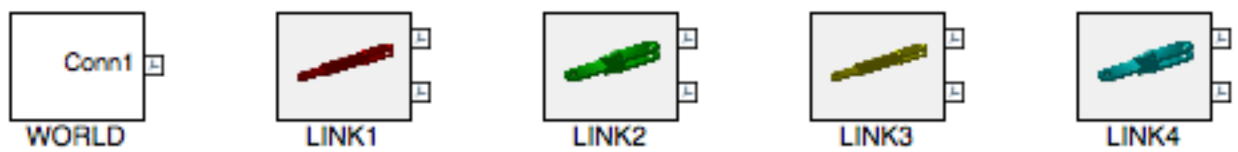



```
SGfigure; view(-30,30); axis on;
SGT (SG2);
```



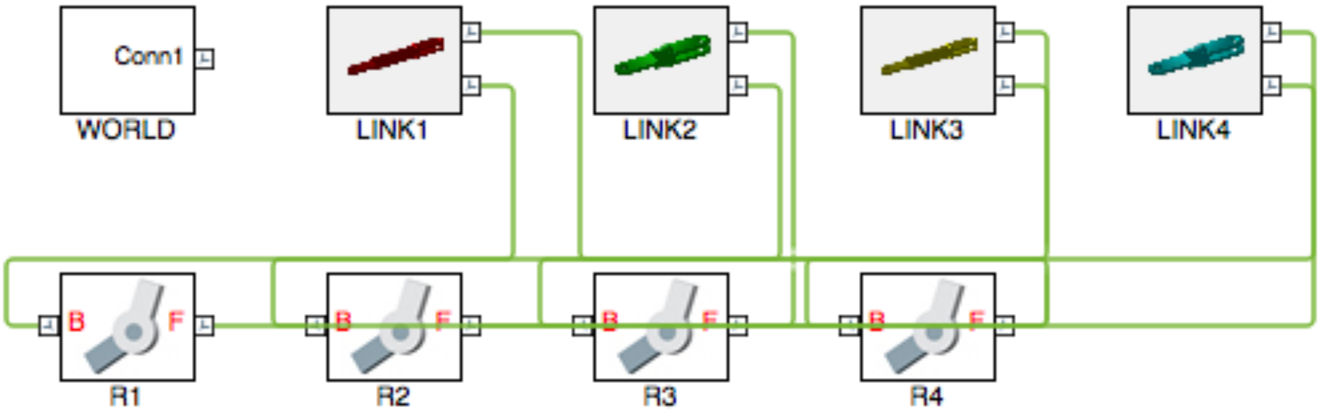
4. Create SimMechanics models for the four links in different colors

```
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;
```



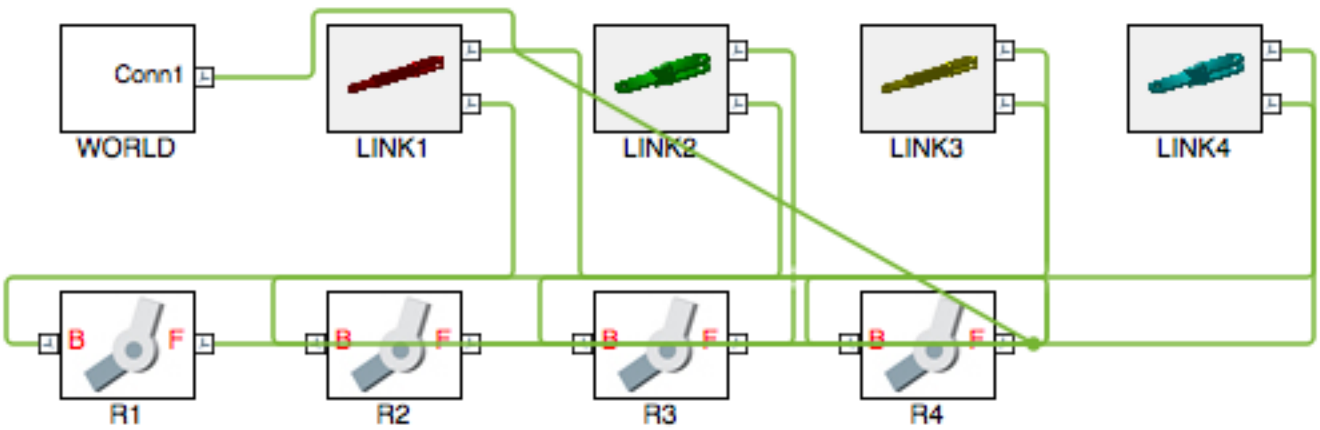
5. Create SimMechanics models for the four joint and connect them with the links

```
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;
```



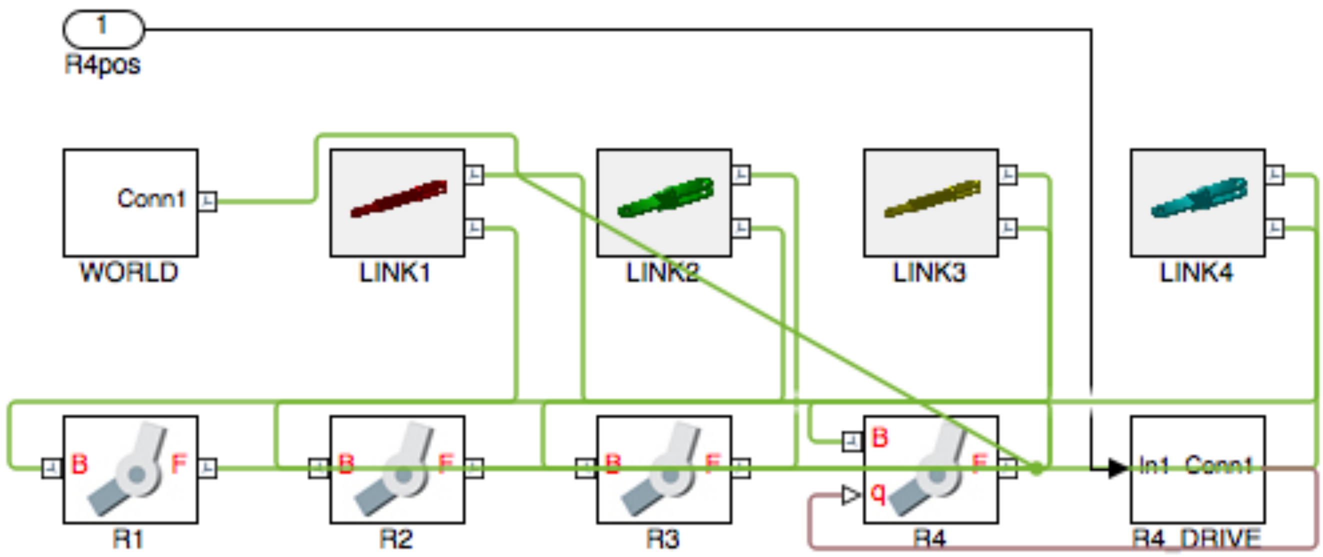
6. Connect the base frame of link1 to the world coordinate system

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



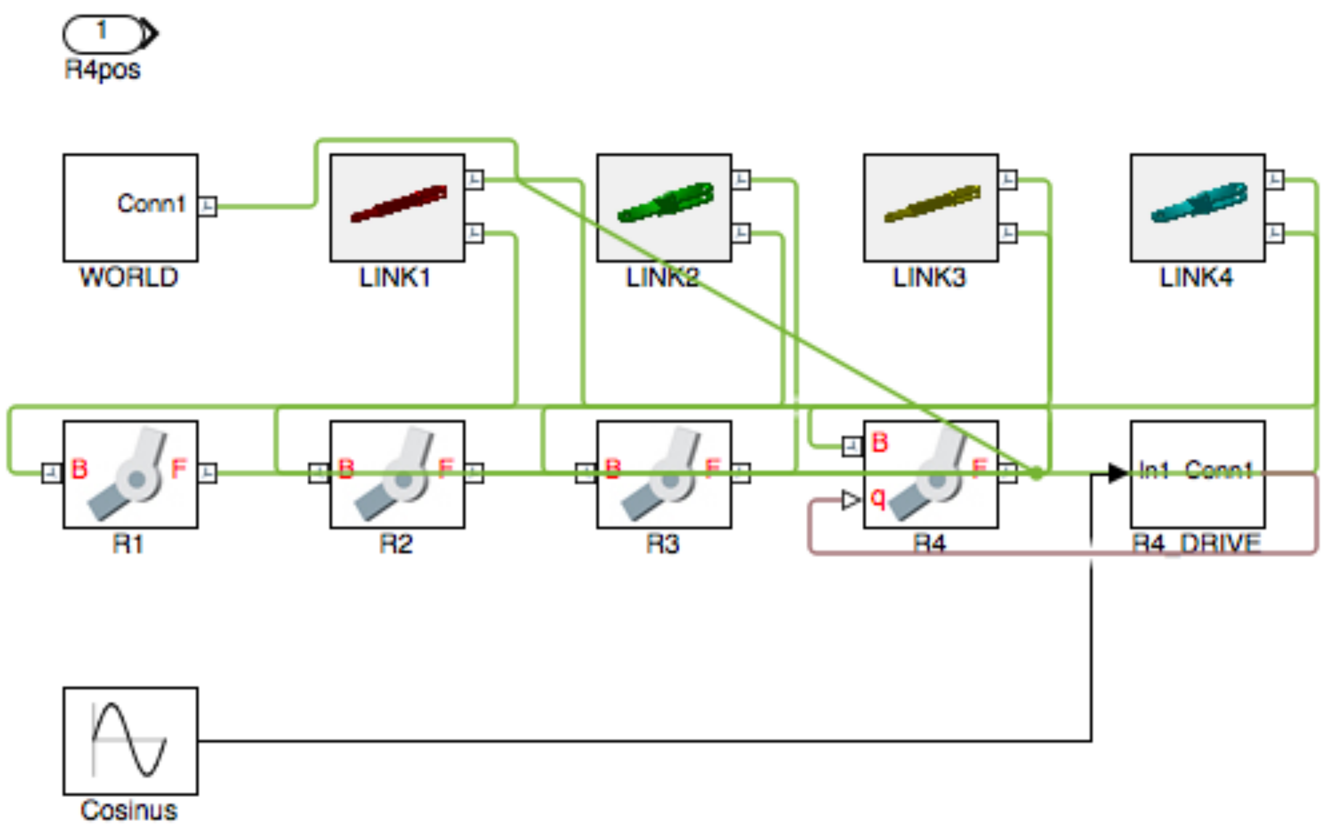
7. Create a SimMechanics model for a motor/drive and use a Cosinus Rotation

```
smbCreateDrive ('R4');                        % Convert Joint R4 into a Drive
smbDrawNow;
```



8. Create a Simulink models for a cosinus signal

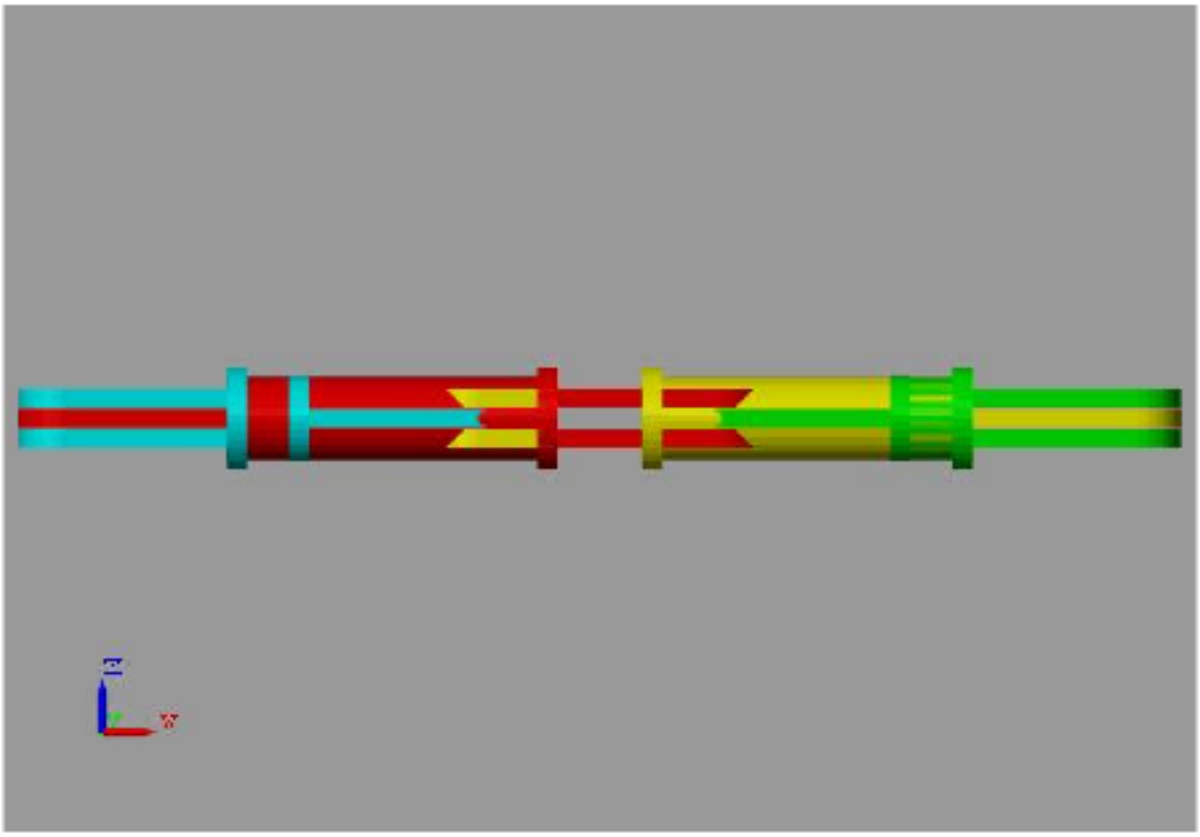
```
smbCreateSineWave ('Cosinus','R4_DRIVE/1');     % Connect a Sinus Generator to Drive
smbDrawNow;
```



9. Create a Video of the Simualati0n for 10 seconds

```
smbVideoSimulation;  % Show a 5 seconds video
```

....



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:18:35 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 22: Adding Simulink Signals to Record Frame Movements

2016-12-18: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

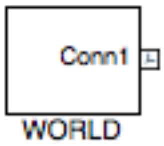
The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Adding Simulink Signals to Record Frame Movements

2. Creating a new SimMechanics System

```
smbNewSystem ('SG_LIB_EXP_22');            % Creates the mechansim diagramm
smbDrawNow;
```

```
Creating temporary directory '/Users/lueth/Desktop/tmp_SG_LIB_EXP_22/'
```

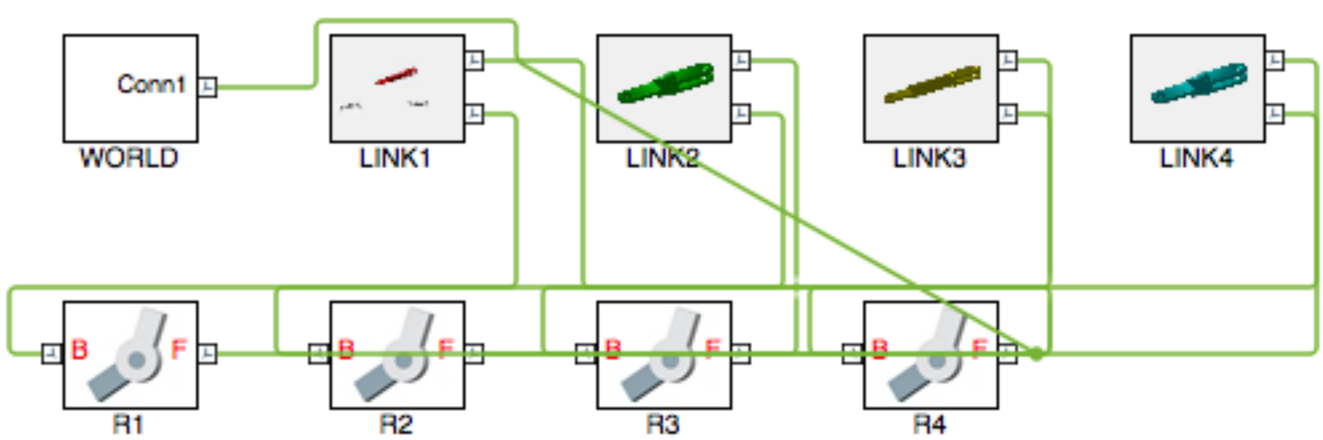


3. Create two links with length 50 and 80 and one or two mounting holes

```
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)
```

```
Simulink.SimulationOutput:

    simlog: [1×1 simscape.logging.Node]
      tout: [209×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.
```

The states contain the parameter = angles/velocity of the joints

```
xout = simOut.get('xout')
```

```
xout =

  Simulink.SimulationData.Dataset
  Package: Simulink.SimulationData

  Characteristics:
              Name: 'xout'
    Total Elements: 8

  Elements:
    1 : 'SG_LIB_EXP_22.R1.Rz.q'
    2 : 'SG_LIB_EXP_22.R1.Rz.w'
    3 : 'SG_LIB_EXP_22.R2.Rz.q'
    4 : 'SG_LIB_EXP_22.R2.Rz.w'
    5 : 'SG_LIB_EXP_22.R3.Rz.q'
    6 : 'SG_LIB_EXP_22.R3.Rz.w'
    7 : 'SG_LIB_EXP_22.R4.Rz.q'
    8 : 'SG_LIB_EXP_22.R4.Rz.w'

  -Use 'get' or 'getElement' to access elements by index or name.
  -Use 'addElement' or 'setElement' to modify elements.
```

There is no Simulink signals yet

```
sout = simOut.get('sout')
```

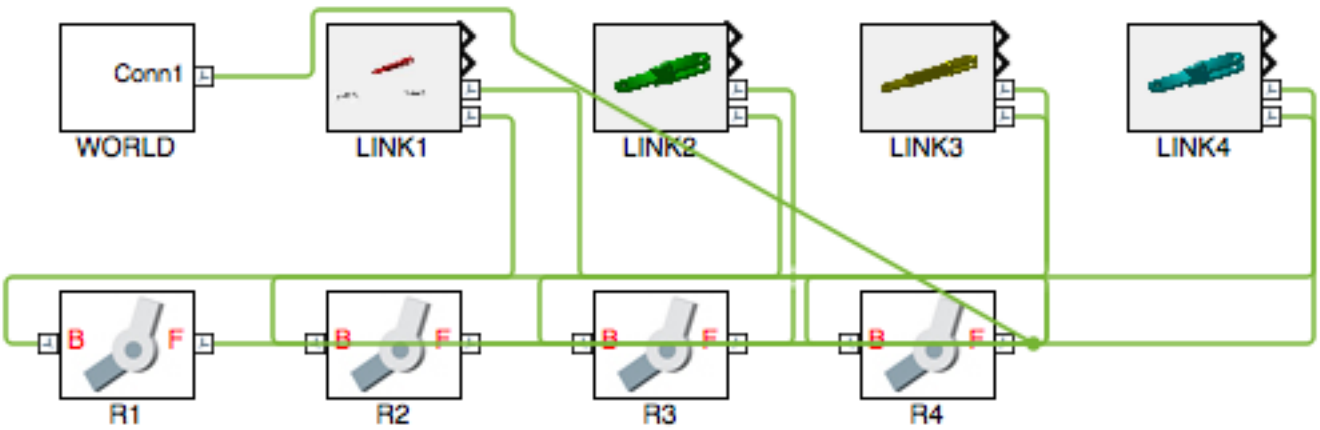
```
sout =

     []
```

7. Create Simulink signals for all the frames of the four links

```
smbAddFrameSensor ('LINK1.RF');
smbAddFrameSensor ('LINK2.RF');
smbAddFrameSensor ('LINK3.RF');
smbAddFrameSensor ('LINK4.RF');
```

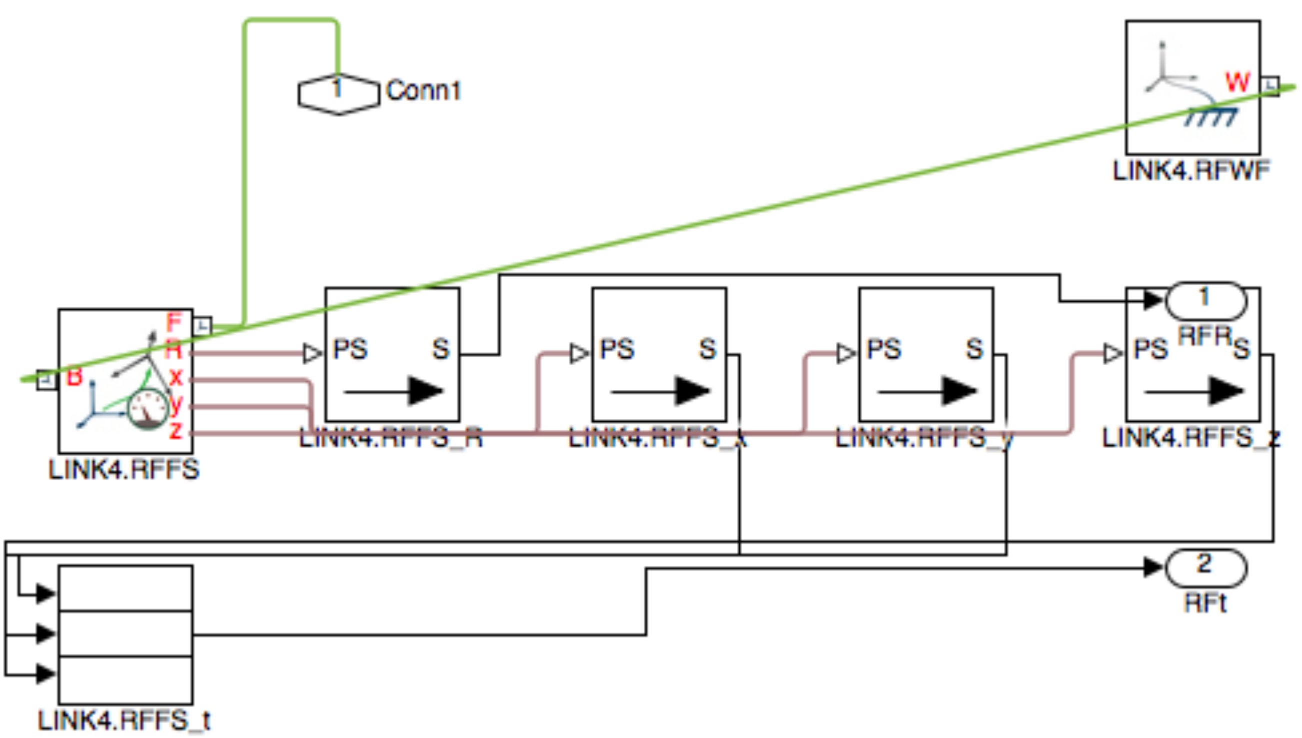
Now, all links have simulink signals and signal output for R and T of the reference frame

```
smbDrawNow;
```



The model of link4 is extendend by a transformation sensor

```
smbDrawNow ('LINK4.RF_T');
```

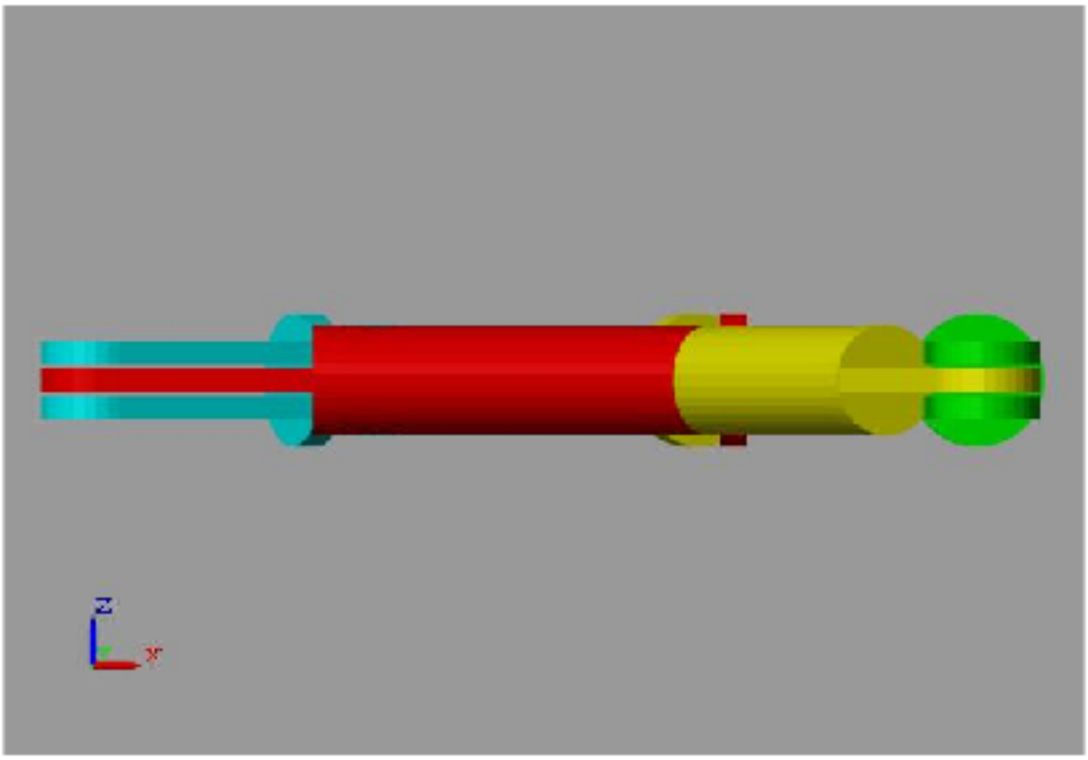


8. Simulate and record those signals too

```
simOut=smbSimulate(3)
smbVideoSimulation(3);
```

```
Simulink.SimulationOutput:
```

```
    simlog: [1×1 simscape.logging.Node]
      sout: [1×1 Simulink.SimulationData.Dataset]
      tout: [209×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.

....
```



The states contain the parameter = angles/velocity of the joints

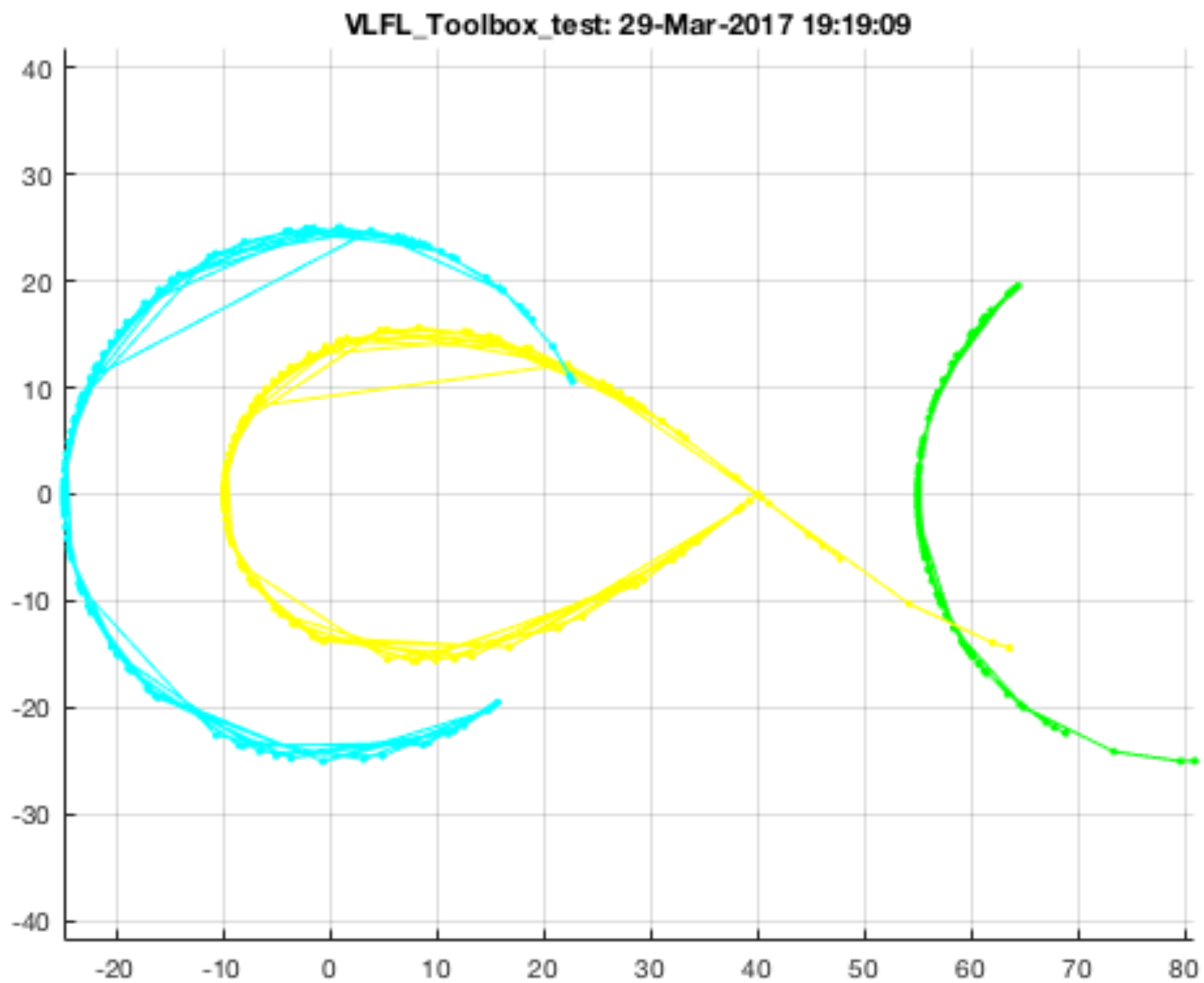
```
xout = simOut.get('xout')
```

```
xout =

  Simulink.SimulationData.Dataset
  Package: Simulink.SimulationData

  Characteristics:
              Name: 'xout'
    Total Elements: 8

  Elements:
    1 : 'SG_LIB_EXP_22.R1.Rz.q'
    2 : 'SG_LIB_EXP_22.R1.Rz.w'
    3 : 'SG_LIB_EXP_22.R2.Rz.q'
    4 : 'SG_LIB_EXP_22.R2.Rz.w'
    5 : 'SG_LIB_EXP_22.R3.Rz.q'
    6 : 'SG_LIB_EXP_22.R3.Rz.w'
    7 : 'SG_LIB_EXP_22.R4.Rz.q'
    8 : 'SG_LIB_EXP_22.R4.Rz.w'

  -Use 'get' or 'getElement' to access elements by index or name.
  -Use 'addElement' or 'setElement' to modify elements.
```

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
  Package: Simulink.SimulationData

  Characteristics:
              Name: 'sout'
    Total Elements: 8

  Elements:
    1 : 'SG_LIB_EXP_22/LINK1/LINK1.RF_T.RFR'
    2 : 'SG_LIB_EXP_22/LINK1/LINK1.RF_T.RFt'
    3 : 'SG_LIB_EXP_22/LINK2/LINK2.RF_T.RFR'
    4 : 'SG_LIB_EXP_22/LINK2/LINK2.RF_T.RFt'
    5 : 'SG_LIB_EXP_22/LINK3/LINK3.RF_T.RFR'
    6 : 'SG_LIB_EXP_22/LINK3/LINK3.RF_T.RFt'
    7 : 'SG_LIB_EXP_22/LINK4/LINK4.RF_T.RFR'
    8 : 'SG_LIB_EXP_22/LINK4/LINK4.RF_T.RFt'

  -Use 'get' or 'getElement' to access elements by index or name.
  -Use 'addElement' or 'setElement' to modify elements.
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:19:11 by 'lueth' using Matlab 9.1.0.441655 (R2016b) 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® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

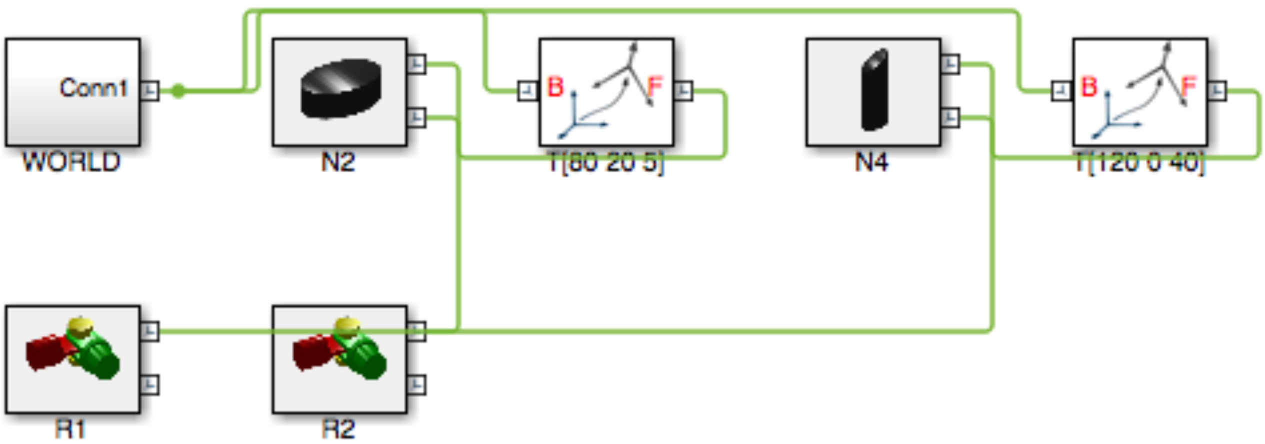
- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model

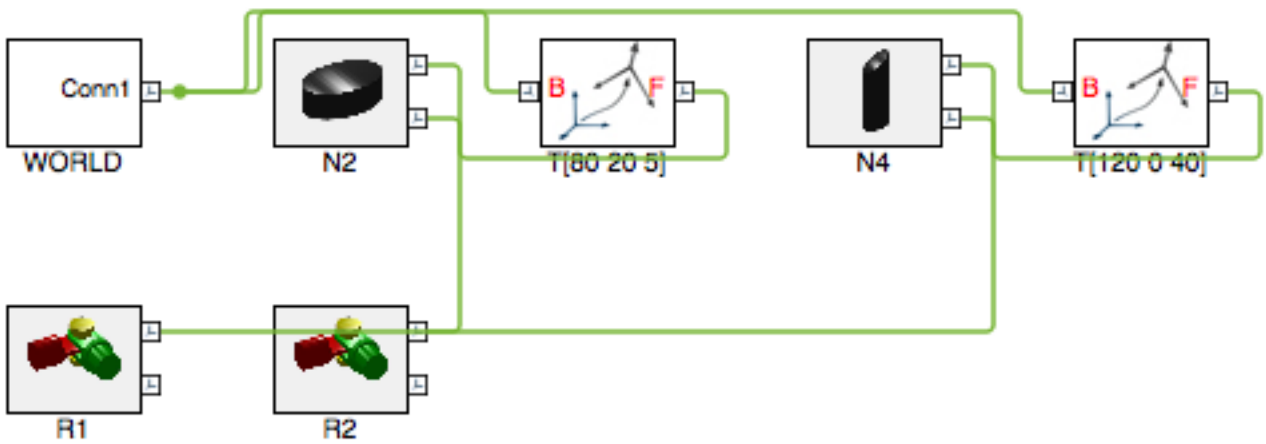
2. Open a system and create several fixed nodes and attach revolute joints

```
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/lueth/Desktop/tmp_SG_LIB_EXP_23/'
```





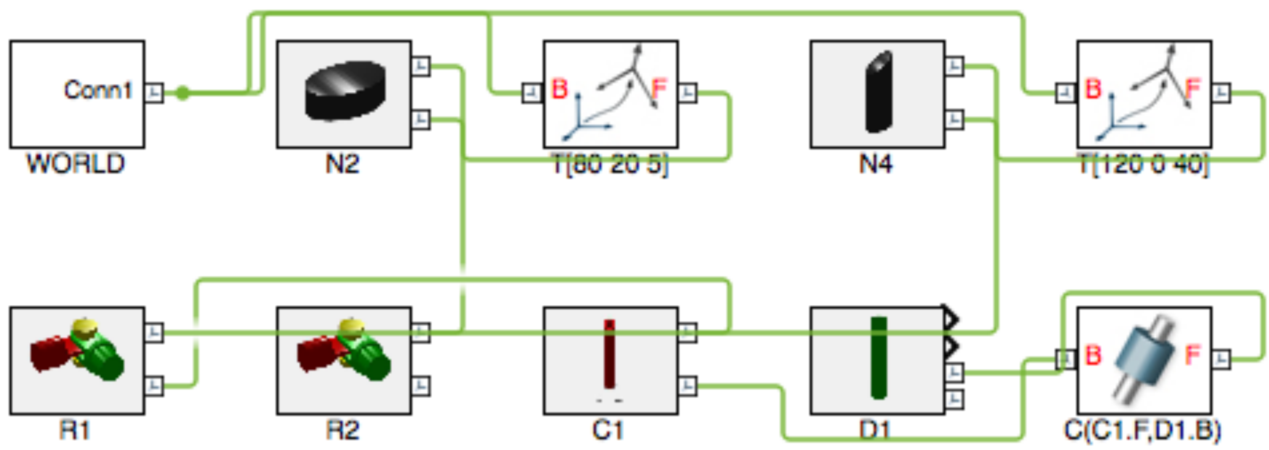
3. Create a cylindric joint from two solids an attach it to revolute joint

```
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

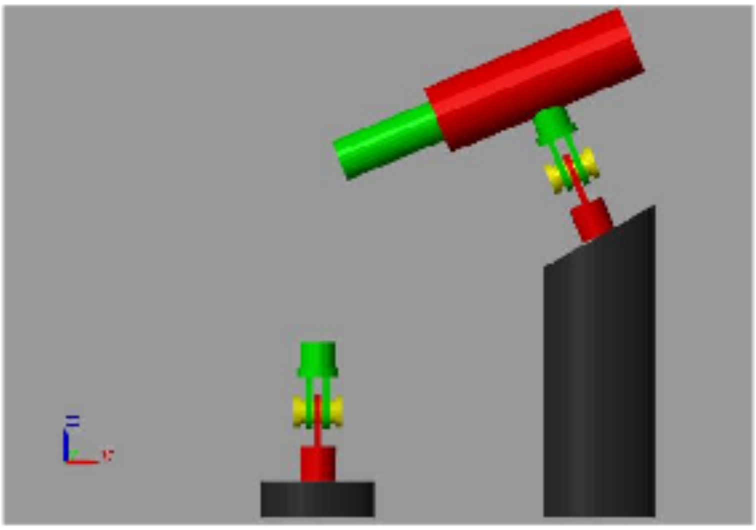
```
smbCreateSG(D1,'D1','g');
smbCreateConnection('C1.F','D1.B','C');
smbAddFrameSensor('R2_M.F');
smbAddFrameSensor('D1.F');
smbDrawNow;
```



5. Show the Simulation

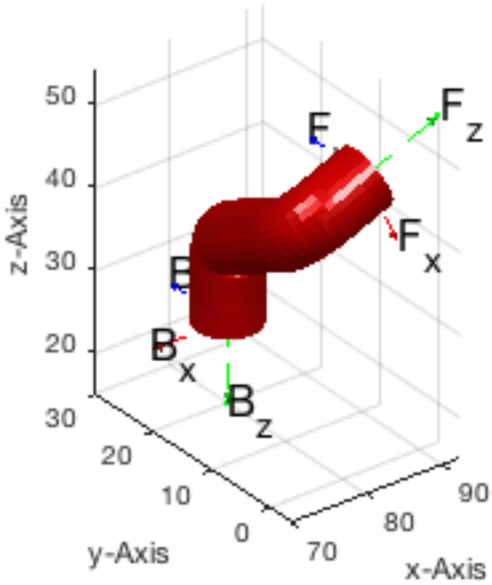
```
simOut=smbSimulate(0.1);
smbVideoSimulation(1);
```

...



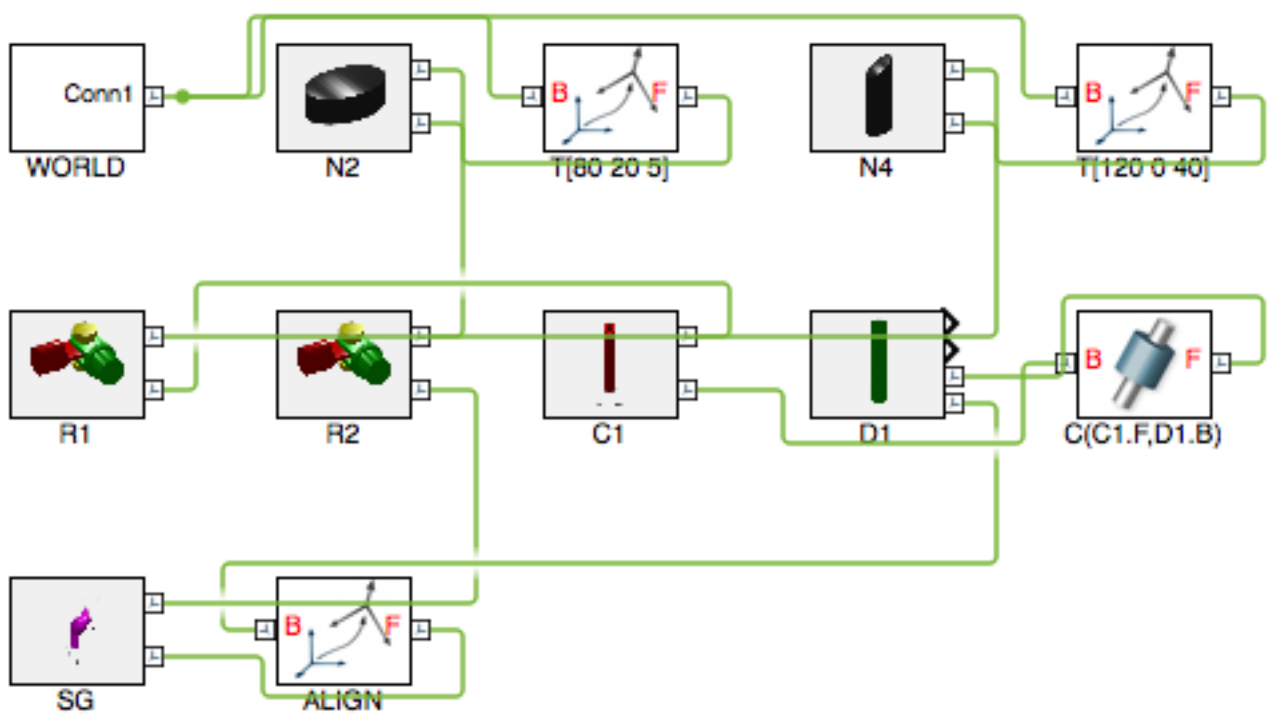
6. Now create a solid between the revolute joint and cylindric joint

```
[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);
```



7. Now connect the new solid in the model

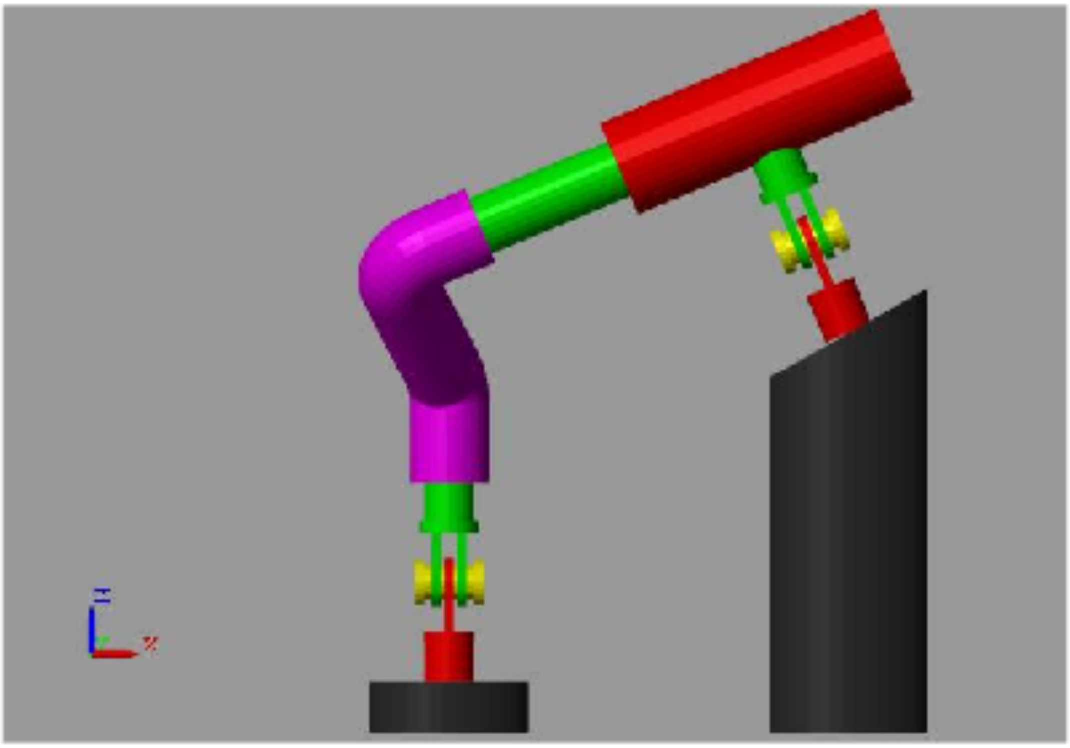
```
smbCreateSG(SG,'SG','m');
smbCreateConnection('R2_M','SG.B');
smbCreateConnection('D1.F','SG.F','align');
smbDrawNow;
```



8. Show the Simulation: The Mechnism has no Movement anymore

```
smbVideoSimulation(1);
```

...

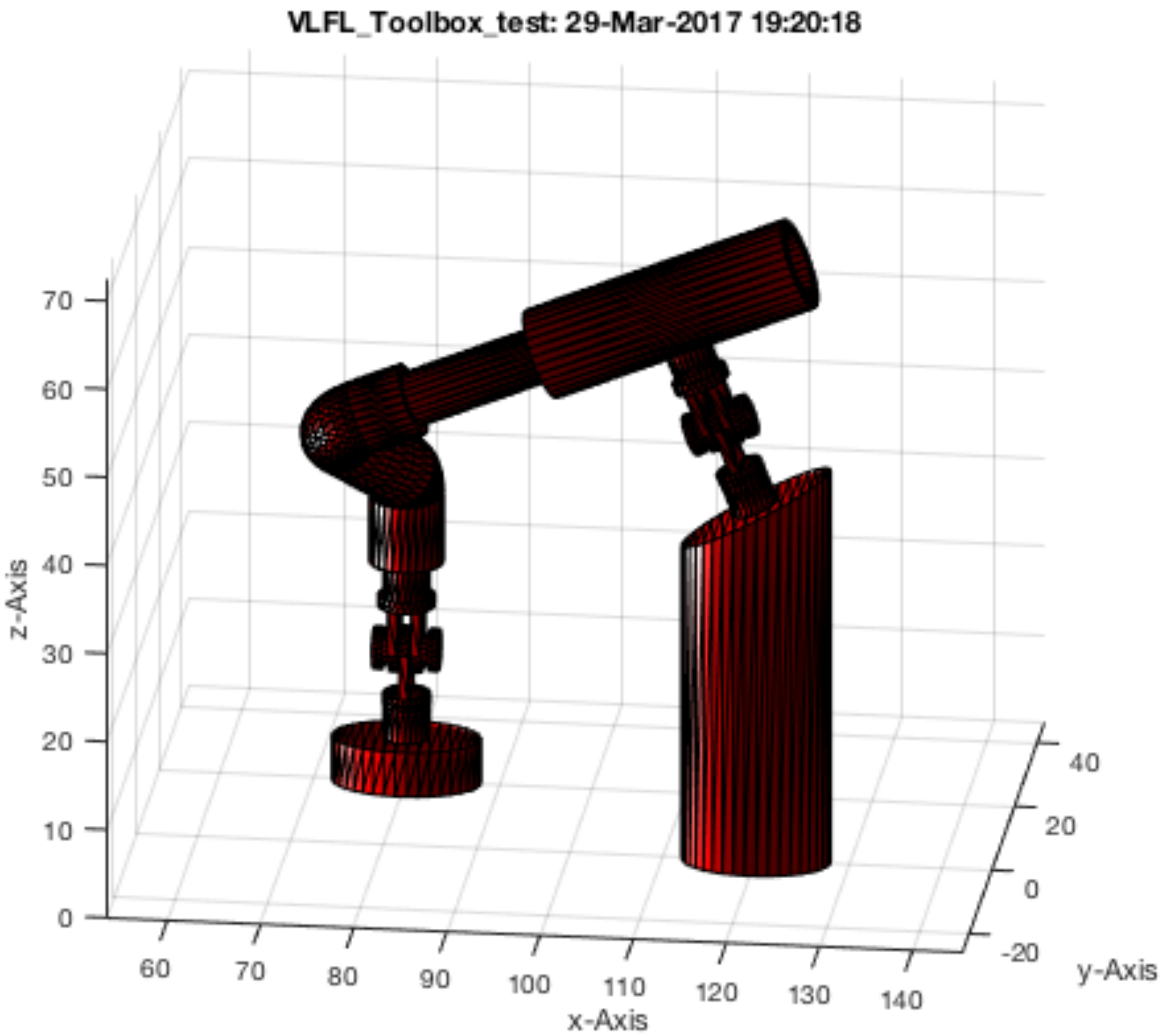


9. Now Create a Solid Model of Movement Status at Time = 0.1 Seconds

```
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'
===========================================================================================
========

Simulink.SimulationOutput:

    simlog: [1×1 simscape.logging.Node]
      sout: [1×1 Simulink.SimulationData.Dataset]
      tout: [51×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.

LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_C1.stl
Header:
```

```
Number of facets: 232
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_D1.stl
Header:
Number of facets: 96
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_N2.stl
Header:
Number of facets: 156
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_N4.stl
Header:
Number of facets: 156
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R1.FIX1.stl
Header:
Number of facets: 240
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R1_M.stl
Header:
Number of facets: 456
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R1_S.stl
Header:
Number of facets: 448
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R2.FIX1.stl
Header:
Number of facets: 240
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R2_M.stl
Header:
Number of facets: 456
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_R2_S.stl
Header:
Number of facets: 448
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_23/sbm_temp_SG.stl
Header:
Number of facets: 1214
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:20:21 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==========================================================================================
==========
```

Published with MATLAB® R2016b

Exercise 24: Automatic Creation of a Joint Limitations

2016-12-25: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations

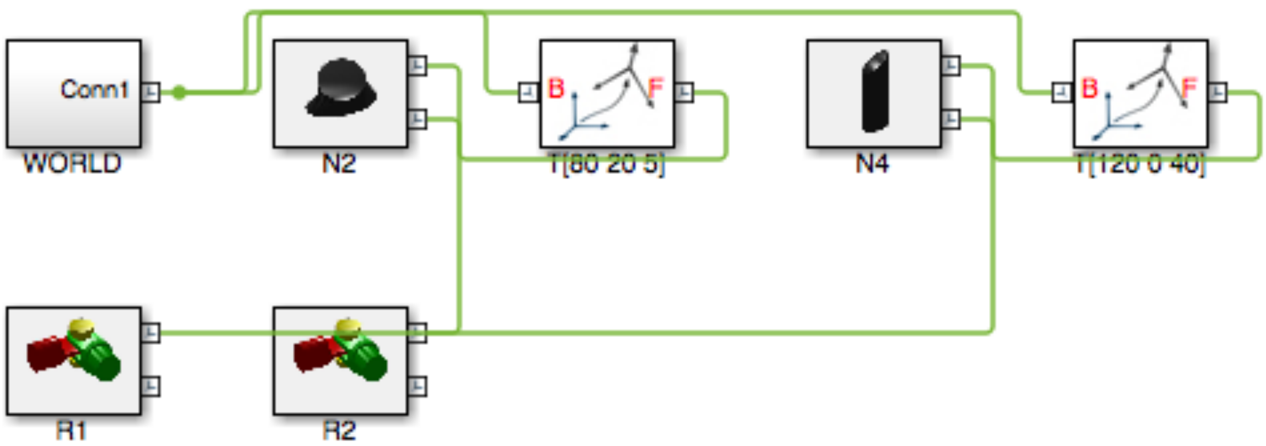
2. Open a system and create several fixed nodes and attach revolute joints

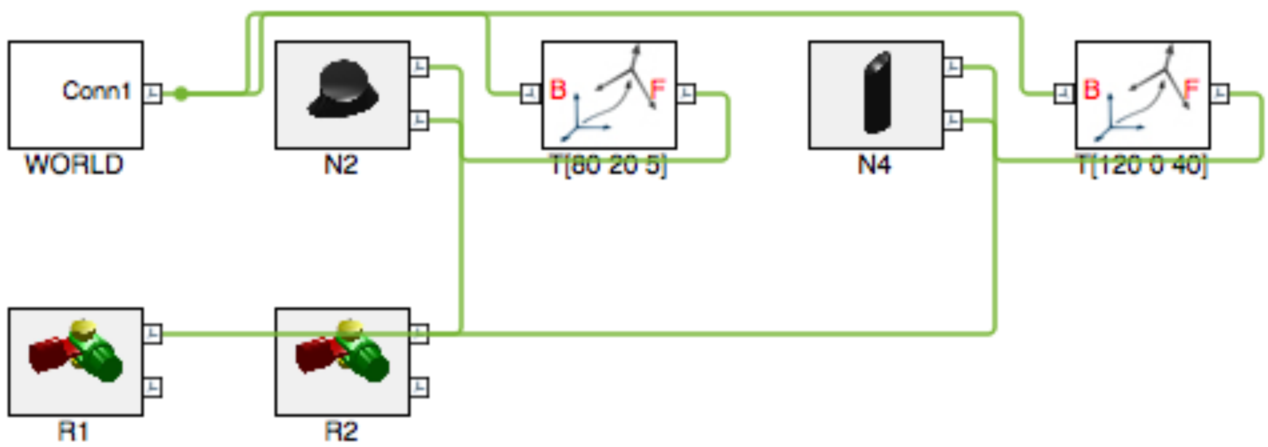
function VLFL_EXP23

```
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/lueth/Desktop/tmp_SG_LIB_EXP_24/'
```





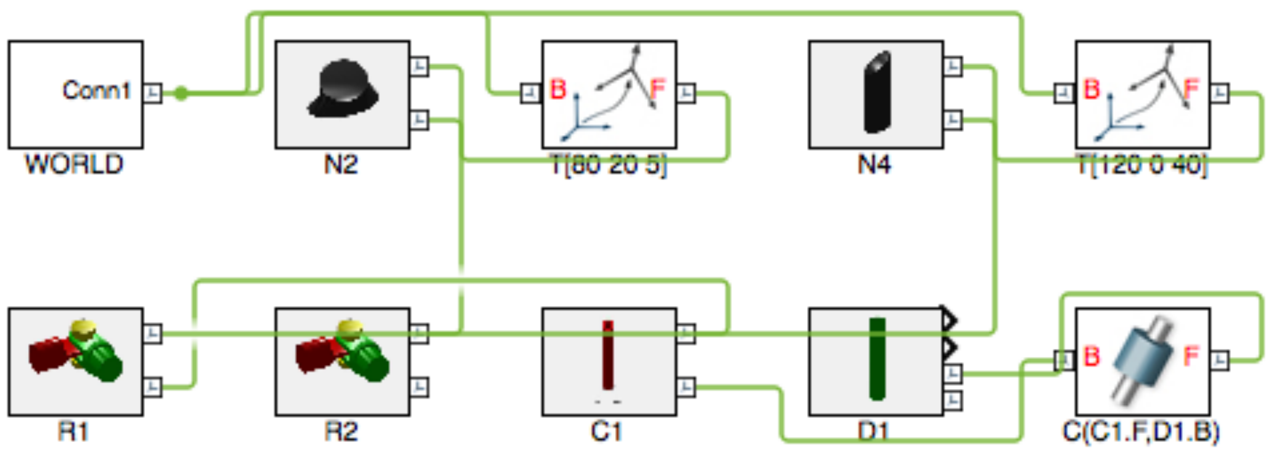
3. Create a cylindric joint from two solids an attach it to revolute joint

```
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');
```

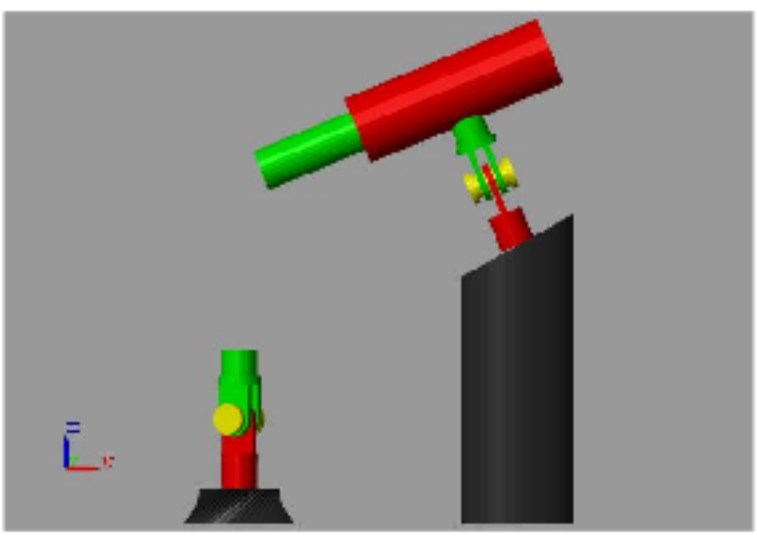
```
smbCreateConnection('C1.F','D1.B','C');
smbAddFrameSensor('R2_M.F');
smbAddFrameSensor('D1.F');
smbDrawNow;
```



5. Show the Simulation

```
simOut=smbSimulate(0.1);
smbVideoSimulation(4);
```

```
......
```

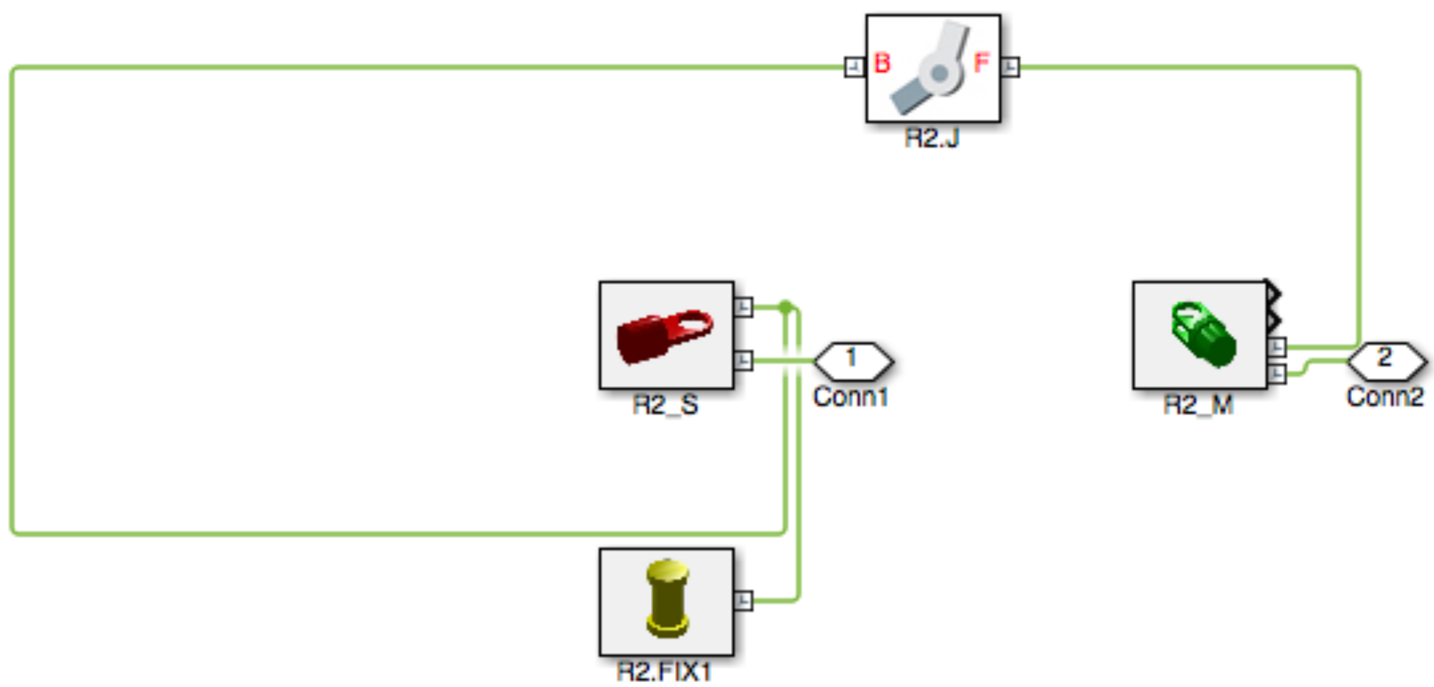


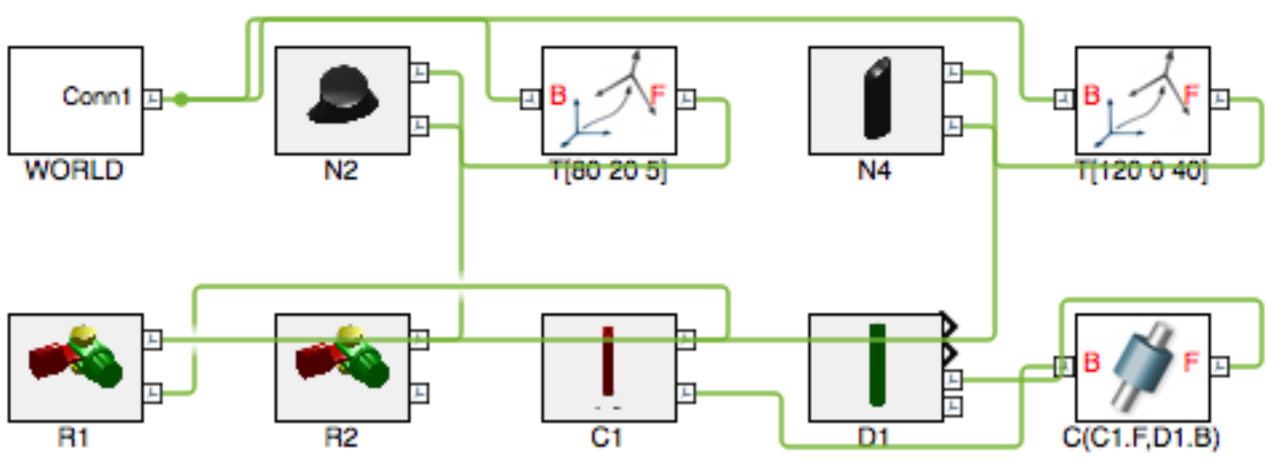
6. Install additional block funktion for joint restrictions

```
smbPSLibInstall
open_system(smbPSBlockname);
open_system(smbsys,'tab');
open_system(smbWhich('R2'),'tab');smbDrawNow;
```

```
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_friction_rot.ssc
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_friction_rot.svg
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hardstop_rot.ssc
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hardstop_rot.svg
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_friction_trans.ssc
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_friction_trans.svg
```

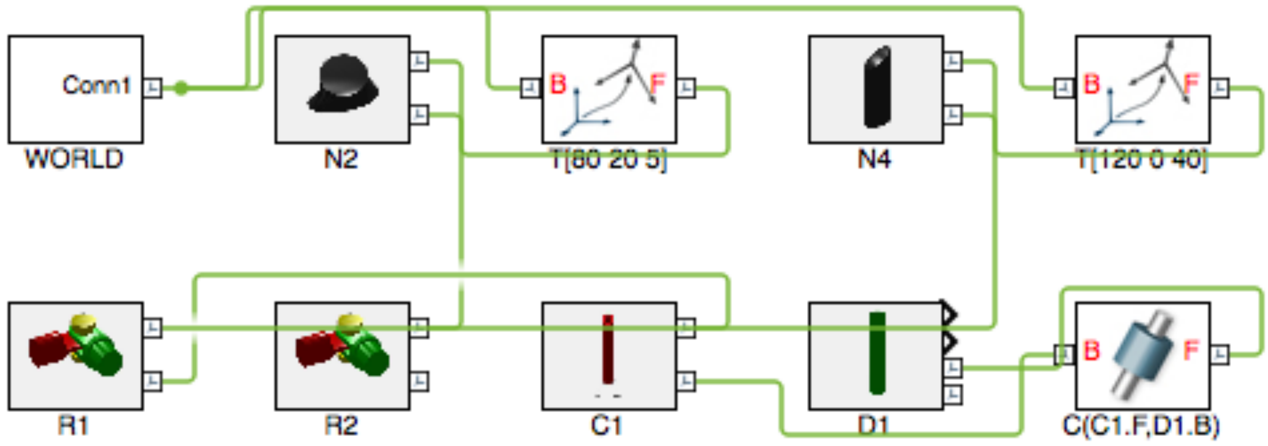
```
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hardstop_trans.ssc
Create /Users/lueth/Desktop/tmp_SG_LIB_EXP_24/+mechPS_Tim_Lueth/PS_force_hardstop_trans.svg
Generating Simulink library 'mechPS_Tim_Lueth_lib' in the current directory '/Users/lueth/Desk
top/tmp_SG_LIB_EXP_24' ...
```





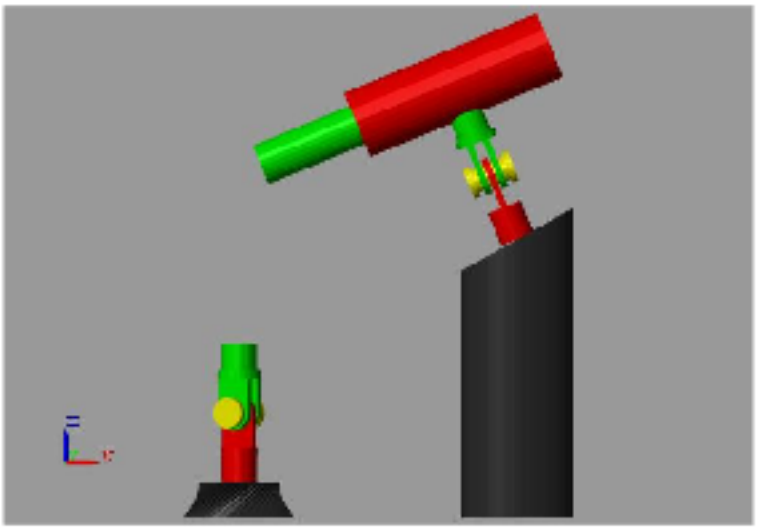
7. Create a stopp joint and copy all connections of an existing joint

```
smbCreateStopJointR ('R2stop.J',[-pi/2 +pi/2]);
smbCopyConnections  ('R2.J','R2stop.J');
smbDrawNow;
```



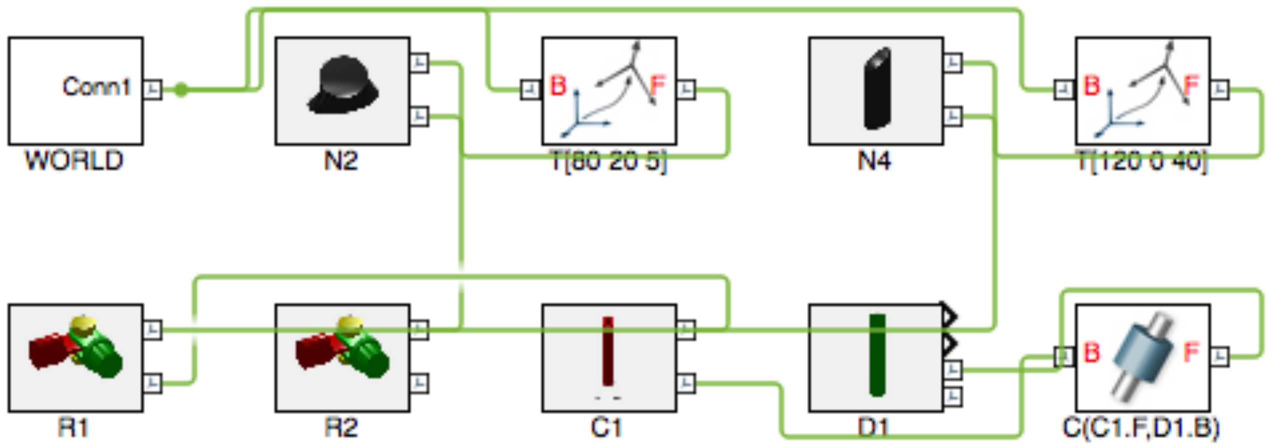

```
smbVideoSimulation(4);
```

......

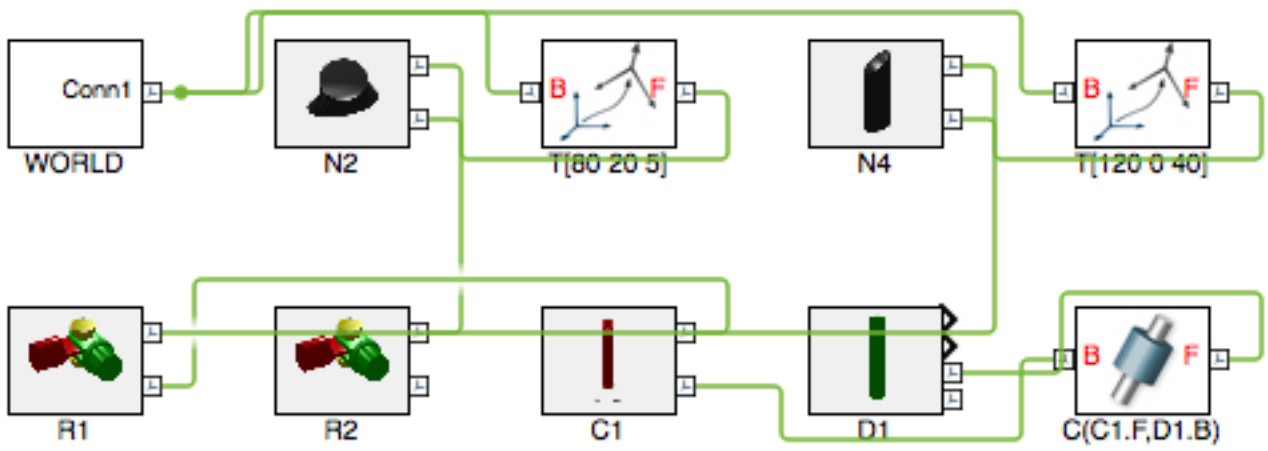


8. Create a stopp joint and replace an existing joint

```
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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:21:45 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ===============================
=======
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==============================================================================================
=======
```

Published with MATLAB® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 2. Motivation for this tutorial
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages

2. Motivation for this tutorial

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/lueth/Desktop/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.

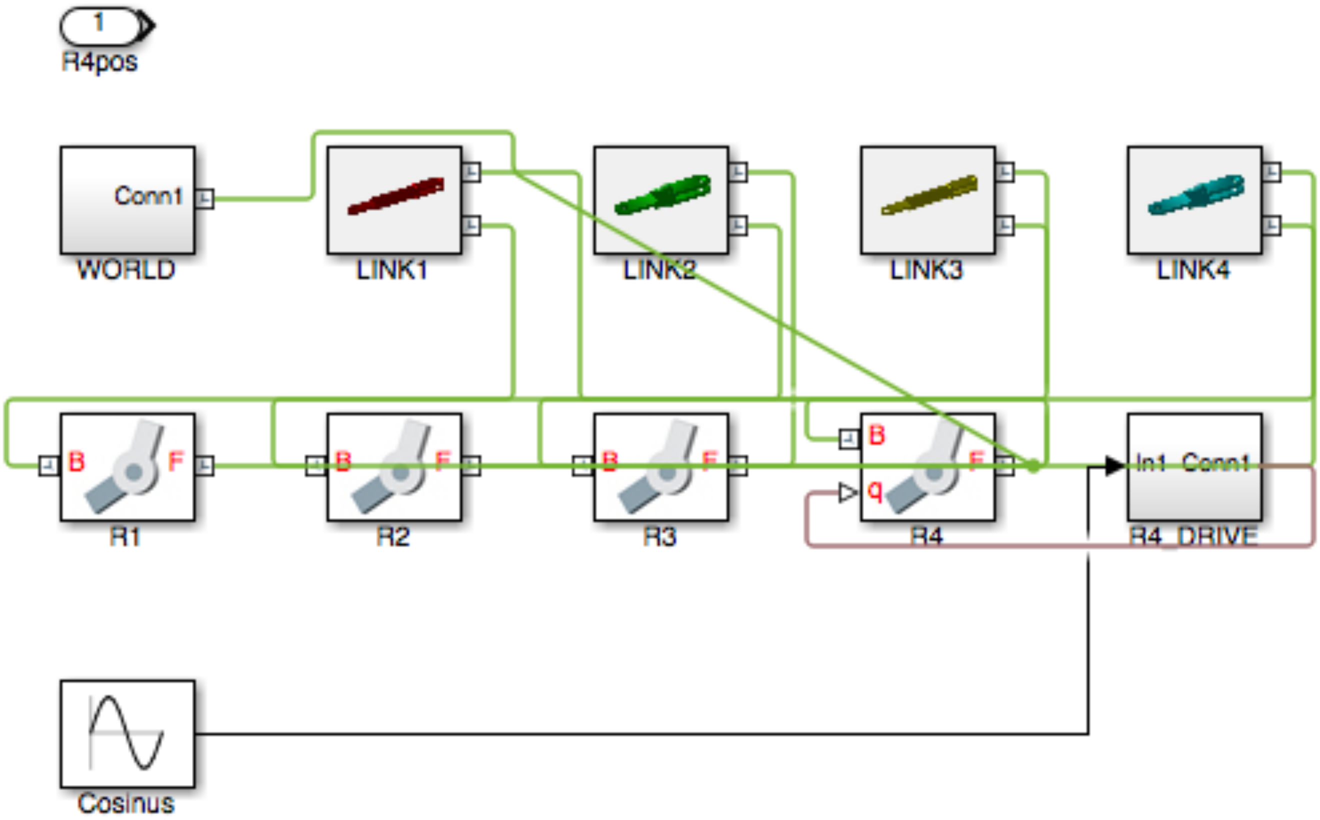
```
I=imageVideoTextPage([640 480],...
['It is also possible to name an existing video or to select it during function '...
'execution to define the size from an existing video. In addition, the '...
'background color and text color can be defined (in future also font name and '...
'font size), and furthermore times for creating a snapshot that becomes part of '...
'the title page.', char(13), '©']);
imshow(I.cdata);
```

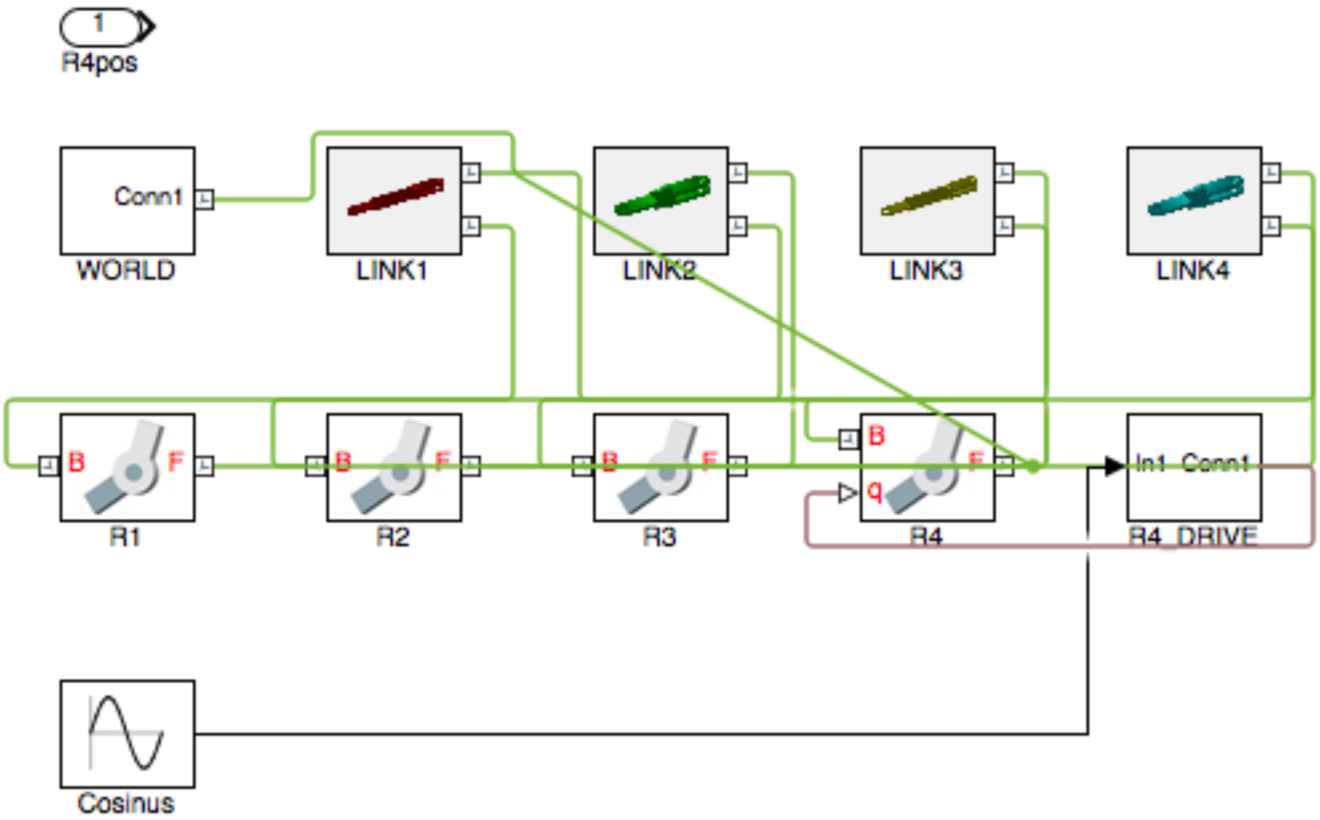
It is also possible to name an existing video or to select it during function execution to define the size from an existing video. In addition, the background color and text color can be defined (in future also font name and font size), and furthermore times for creating a snapshot that becomes part 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/lueth/Desktop/tmp_SG_LIB_EXP_25/'
```

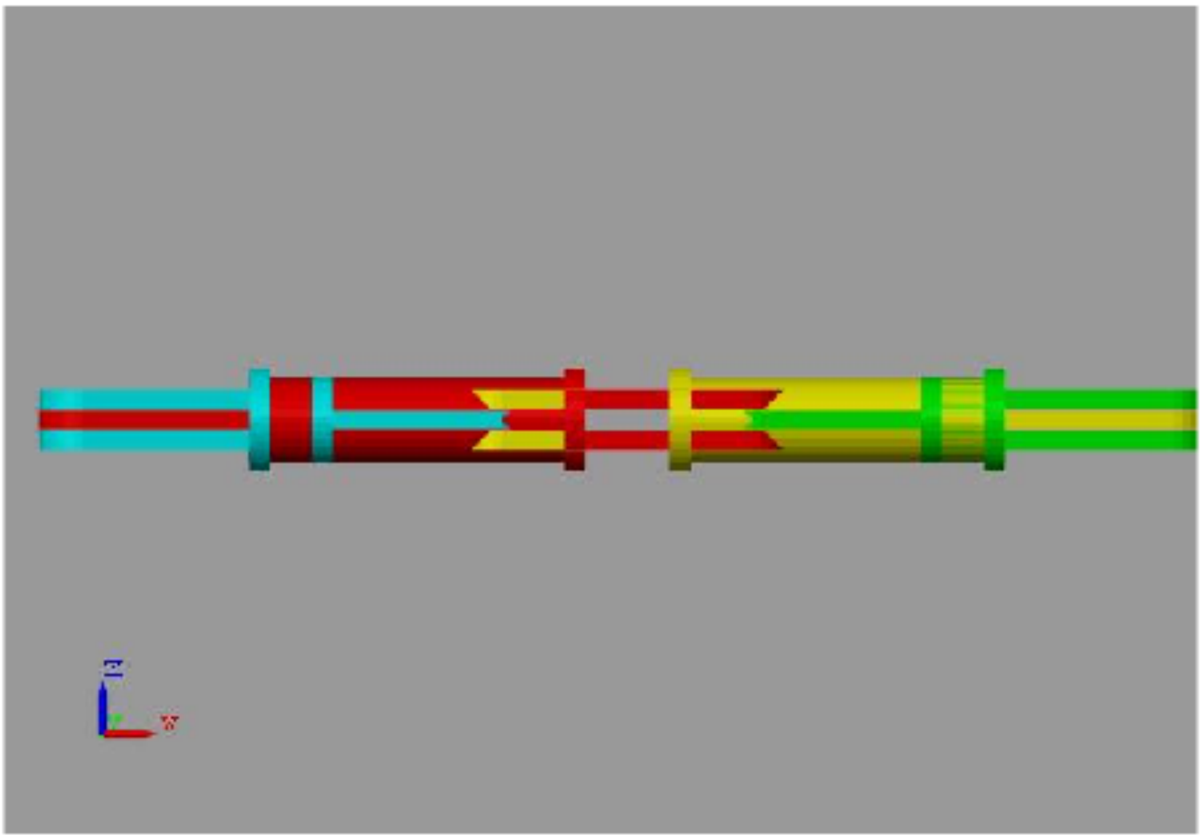




7. Create a video simulation and creates header and titles

```
[I,FN]=smbVideoSimulation (10);
```

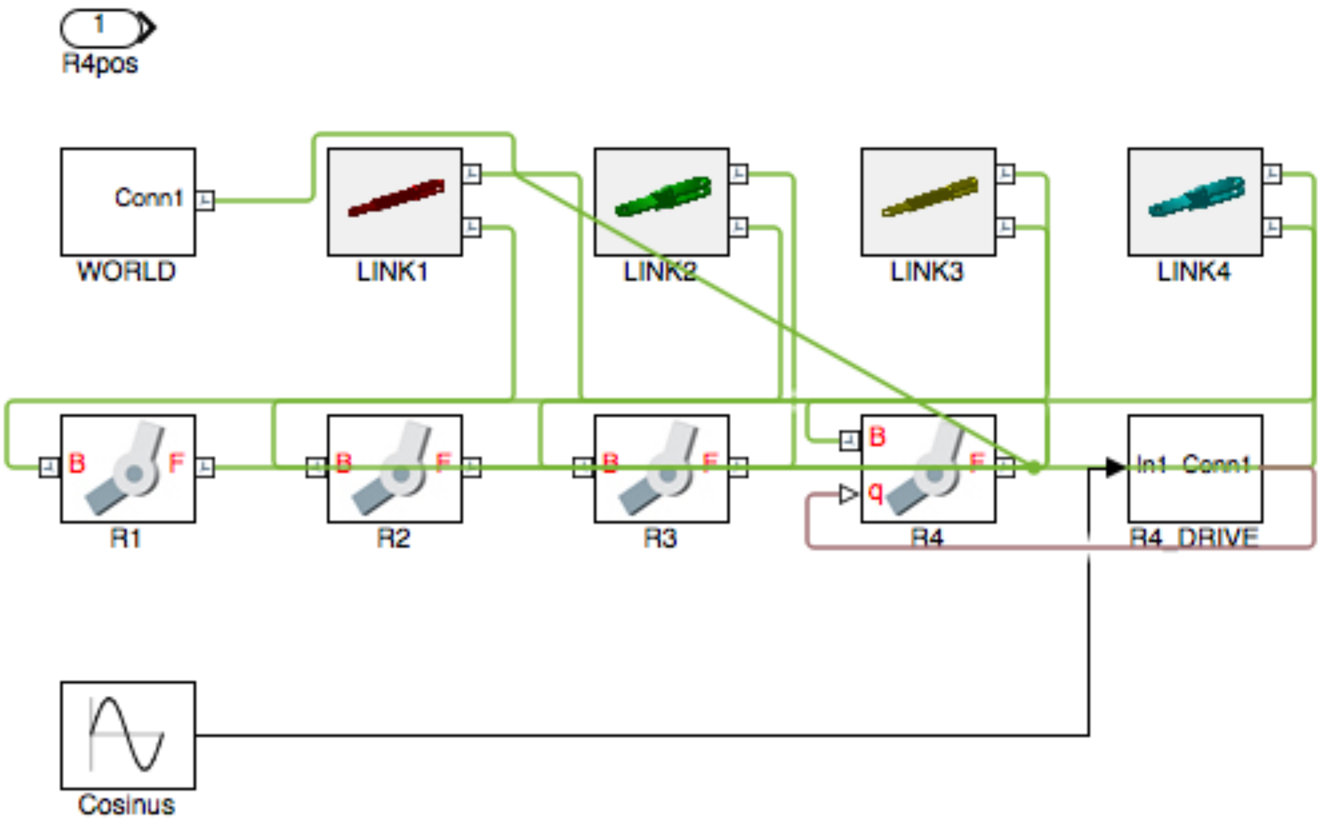
```
............
```



Now we create four small video clips in the desktopdir

```
imageVideoTitle(FN,{'SG-Lib Tutorial #25','Creating Videos Titles for SimMultiBody Videos','Tim C. Lueth
','$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/lueth/Desktop/imageVideoTitle.avi'
Creating a new video file: '/Users/lueth/Desktop/imageVideoEndtitle.avi'
Creating a new video file: '/Users/lueth/Desktop/imageVideoTextPage.avi'
Creating a new video file: '/Users/lueth/Desktop/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. Lu
eth','$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/lueth/Desktop/tmp_SG_LIB_EXP_25/Video co
mp 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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:23:09 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: =========================================
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
```

```
==================================================================================================
```

Published with MATLAB® R2016b

Exercise 26: Create Mechanisms using Universal Planar Links

2017-01-20: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- 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

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing

```
% function VLFL_EXP26
```

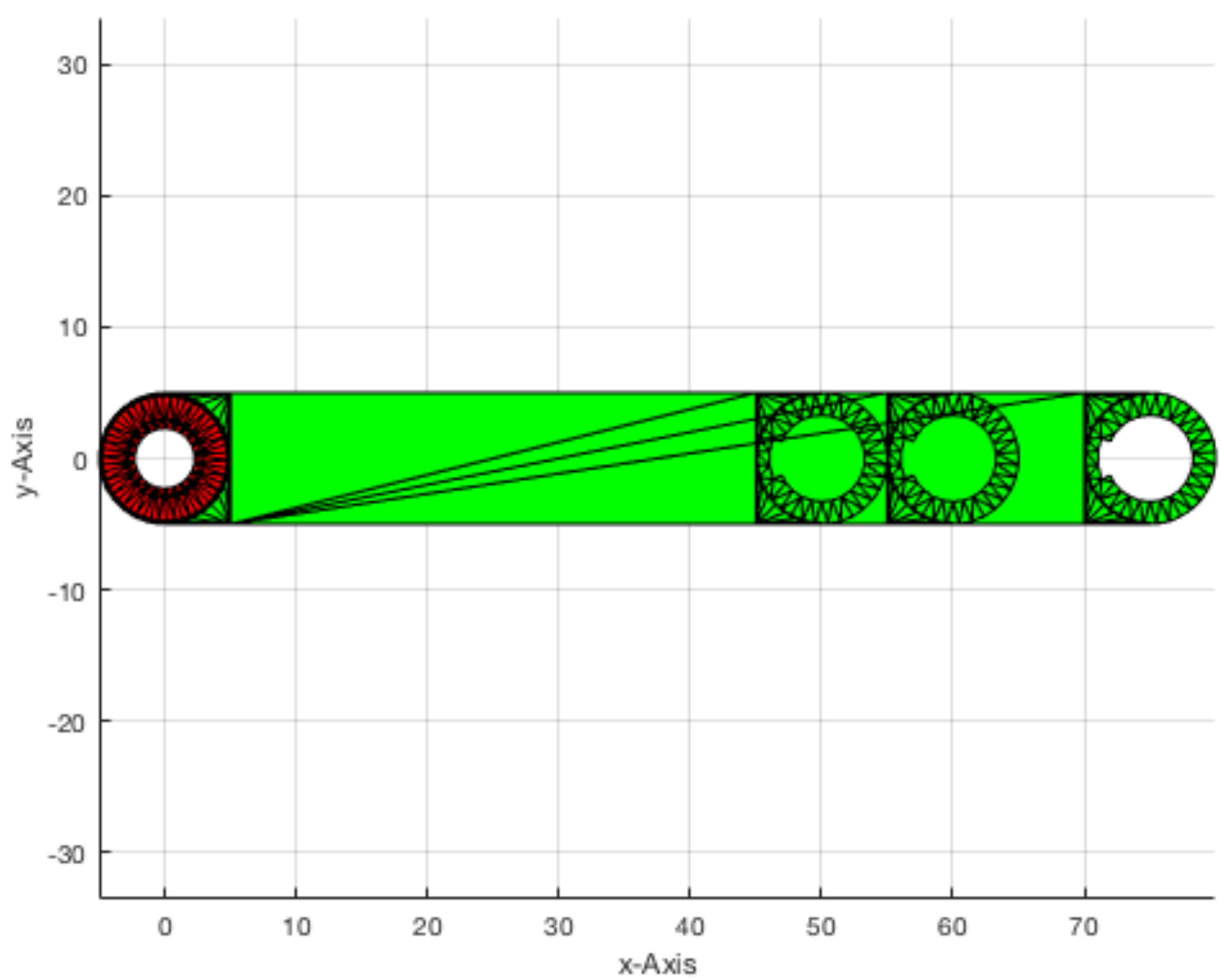
Motivation for this tutorial

Ein Mechanismus besteht aus zwei Grundelementen: a) Gelenke und b) Glieder, die diese Gelenke verbinden. Bei der "automatischen Konstruktion" von Mechanismen ist es hilfreich sich auf eine der beiden Elemente zu beschränken. Dies wurde in den vorherigen Tutorials bereits genutzt. In diesem Tutorial wird nun ein neues Verfahren vorgestellt. Es sind "Universelle Planare Glieder". Diese bestehen aus einem einfachen Gelenkglied sowie zwei Hälften eines rotatorischen Gelenks. Werden zwei Glieder an einem der Endpunkte miteinander verbunden, dann verbinden sich die beide Hälften aufgrund einer räumlichen Überlappung zu einer Drehachse und es wird automatisch ein Drehgelenk gebildet. Wird ein Gliedenden nicht verbunden, dann bleibt dort dennoch eine Drehachse erhalten. Jede Drehachse kann mit "Knöpfen oder Antriebsmechanismen relativ zum Gelenk verbunden werden und in ihrem Winkelbereich eingeschränkt werden. Die Glieder können in festen Ebenen verbunden werden. Dies erlaubt sowohl eine kollisionsfrei Bewegung unter Berücksichtigung der Glieder als auch die Berücksichtigung von Antriebselementen sofern diese in Vielfachen der Gliederhöhe ausgelegt werden. Dieses Tutorial zeigt nun die Verwendung der Universelle Planbaren Glieder an einem einfachen Beispiel.

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;

A=SGmodelLink2(L1,0,1,'BL,FL'); A.col='r';
B=SGmodelLink2(L2,0,1);         B.col='g';
C=SGmodelLink2(L3,0,-1);        C.col='y';
D=SGmodelLink2(L4,0,-1);        D.col='m';
```

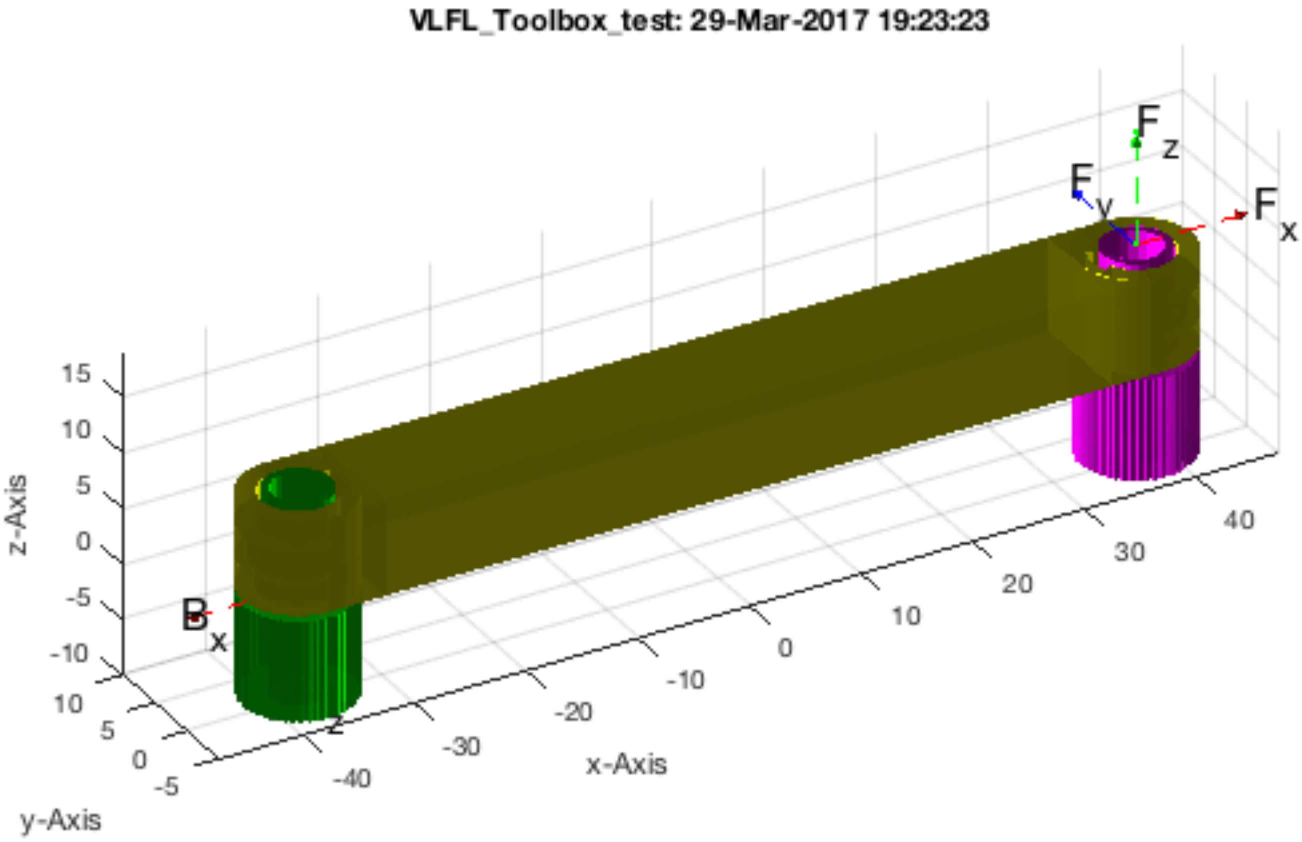
```
Creating temporary directory '/Users/lueth/Desktop/tmp_SG_LIB_EXP_26/'
```



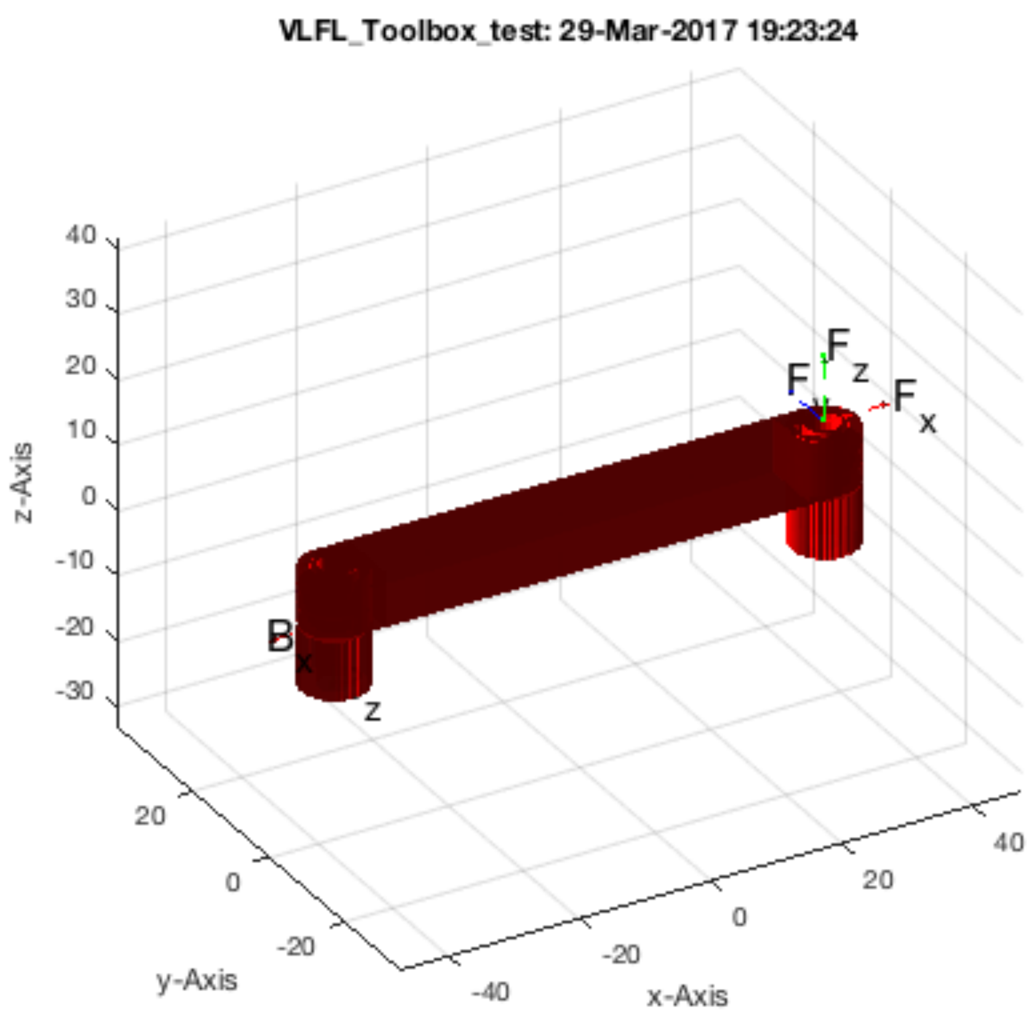
Show the components of the link

```
SGanalyzeGroupParts(A); SGTframeplot(A);
```

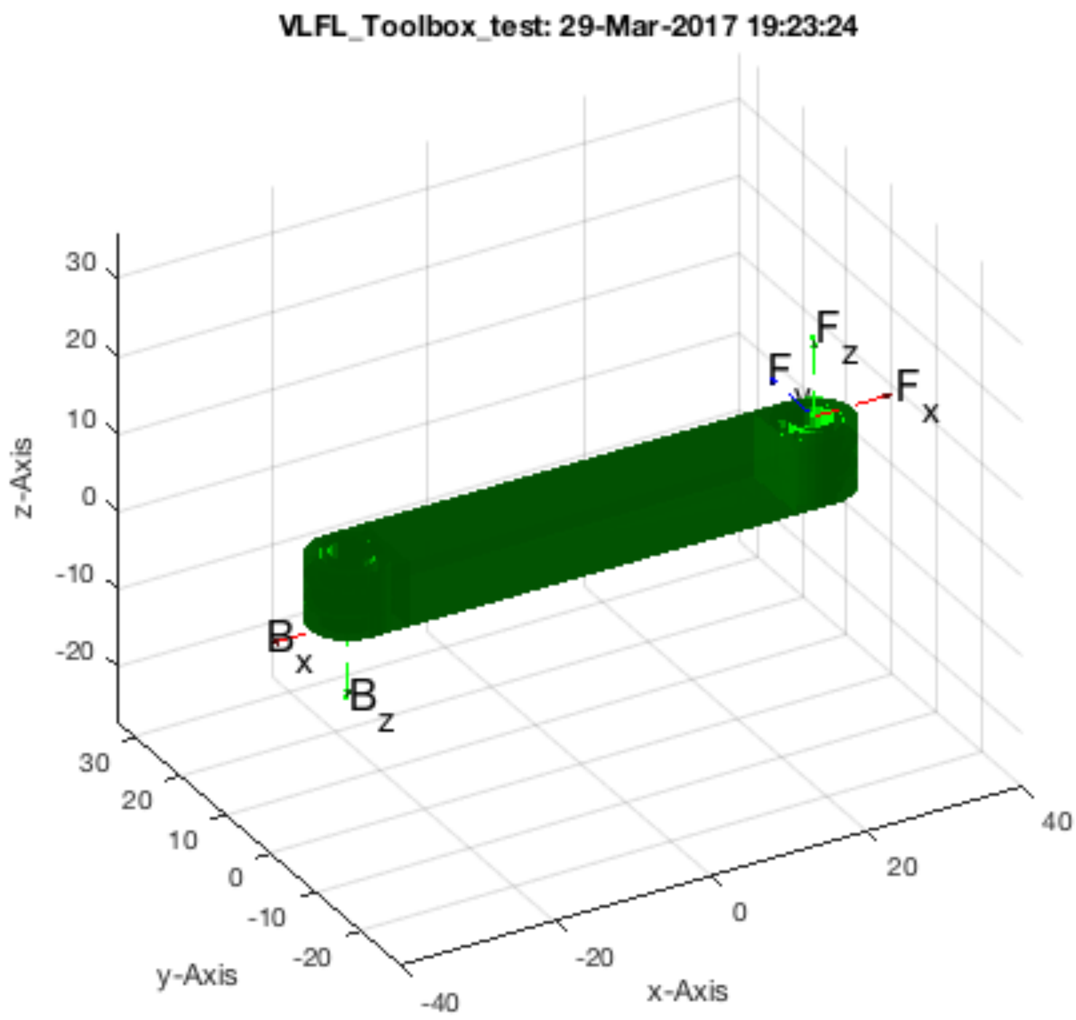
```
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 90% 95% 100%
SGanalyzeGroupParts: 3 separated parts found.
```



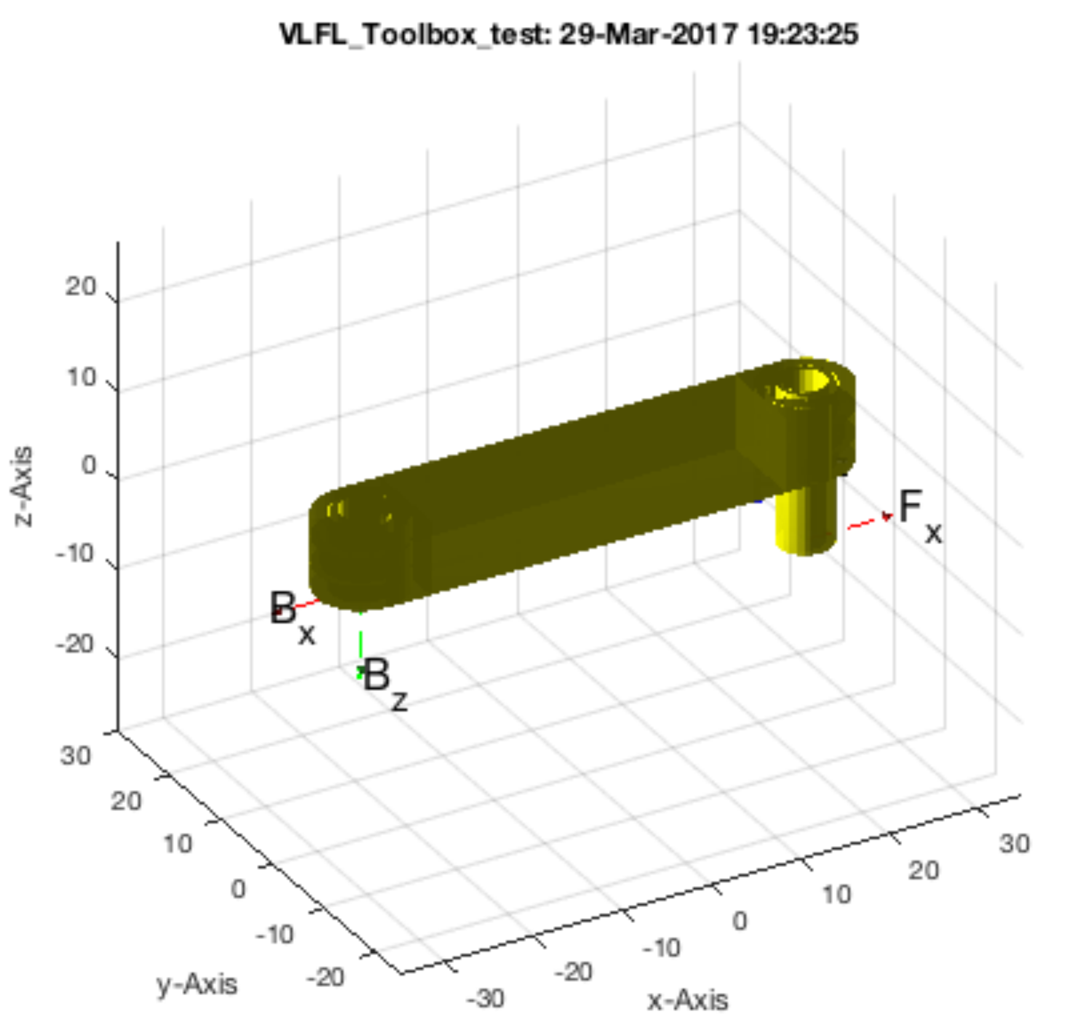

```
SGfigure; SGTplot(A); view(-30,30);
```



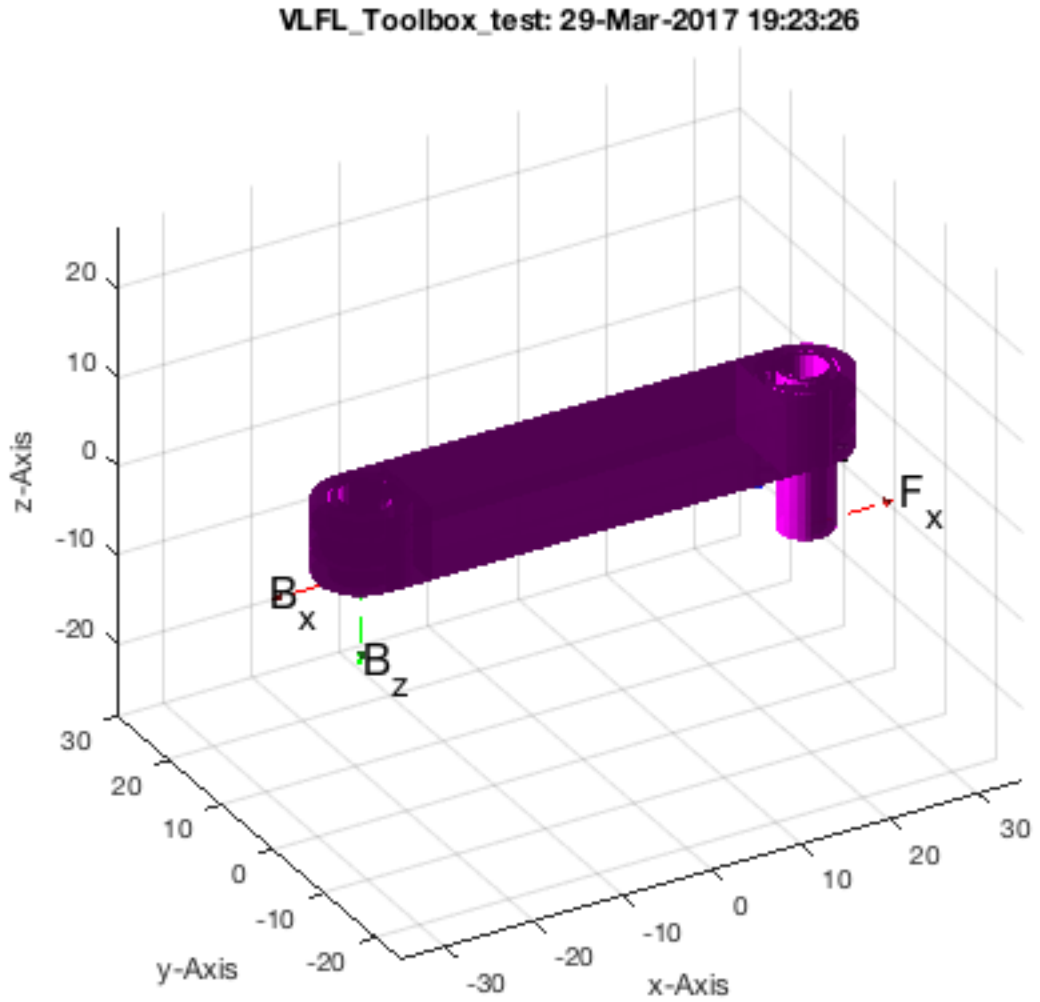

```
SGfigure; SGTplot(B); view(-30,30);
```



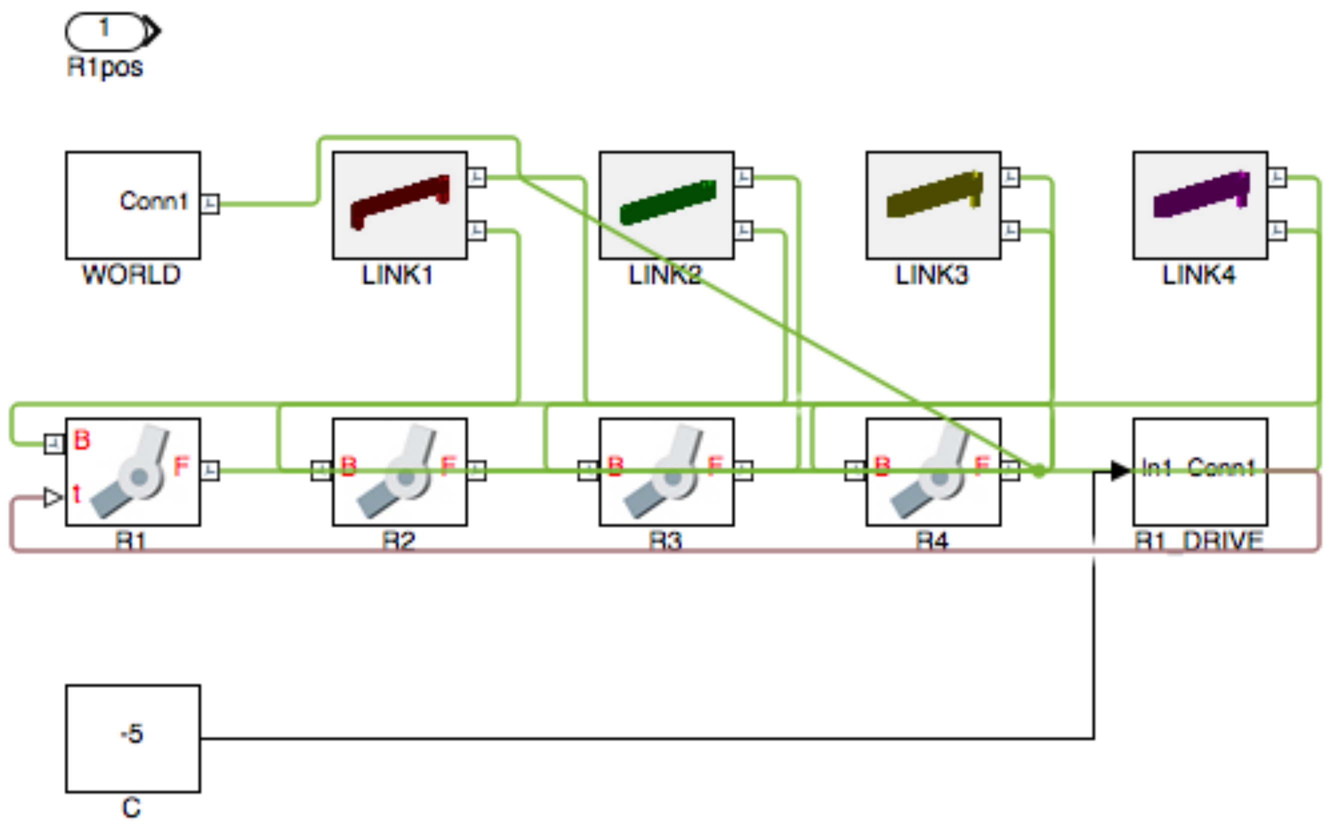

```
SGfigure; SGTplot(C); view(-30,30);
```




```
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);
```

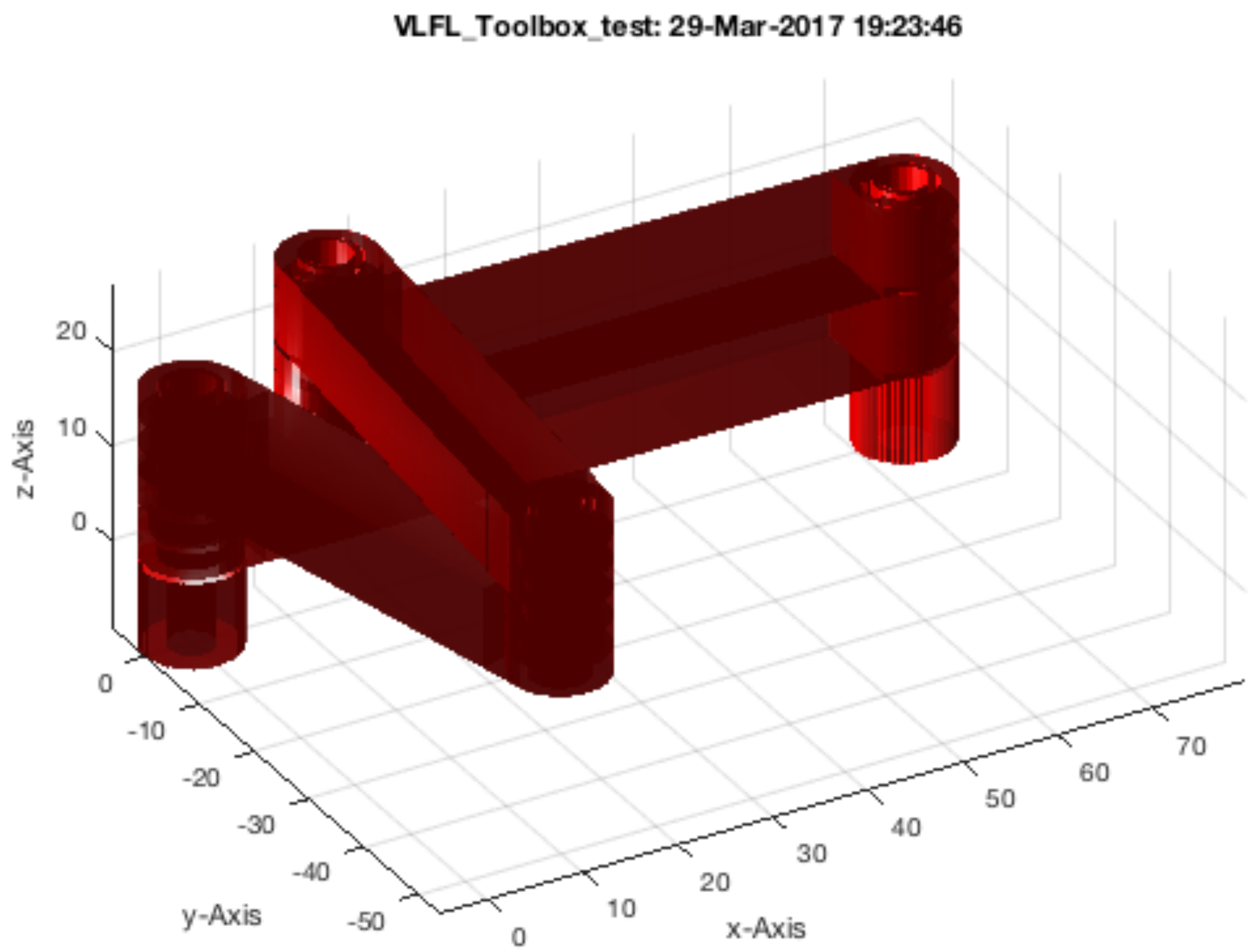


2. Now Create a Specific Configuration (Pose) and Write a STL-Files

```
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 SECONDS
=================================================================================================
=
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'
=================================================================================================
=

Simulink.SimulationOutput:

    simlog: [1×1 simscape.logging.Node]
      sout: [1×1 Simulink.SimulationData.Dataset]
      tout: [1097×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.

LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_26/sbm_temp_LINK1.stl
Header:
Number of facets: 3756
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_26/sbm_temp_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_26/sbm_temp_LINK3.stl
Header:
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_26/sbm_temp_LINK4.stl
Header:
```

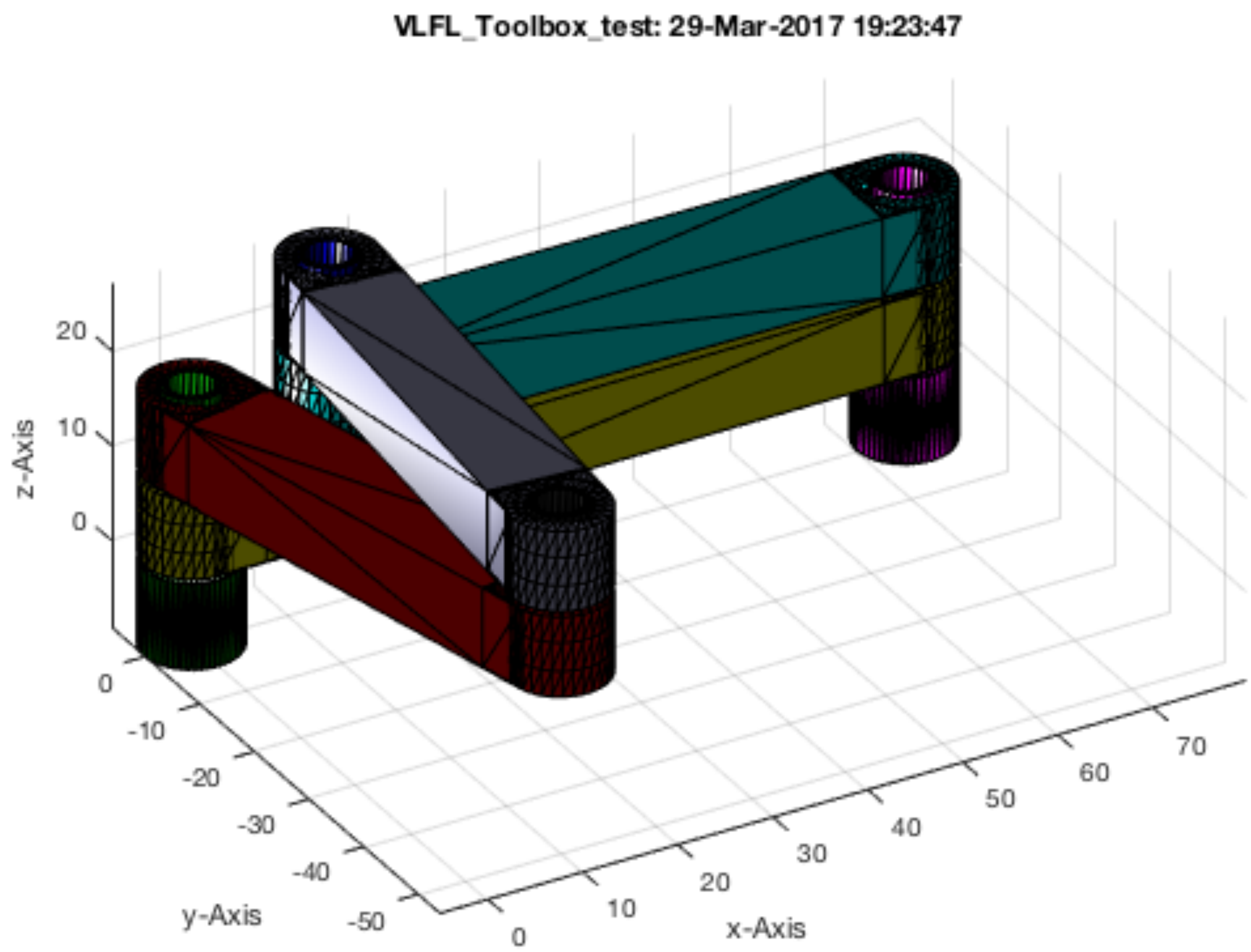
```
Number of facets: 2584
0..
CREATED A SOLID GEOMETRY OF THE FULL SIMULATION-MODEL 'SG_LIB_EXP_26' AT TIME: 5.00 SECONDS
===============================================================================================
=
1000..2000..3000..4000..5000..6000..7000..8000..9000..10000..11000..
```



3. Now Analyze the Stucture and Group the Solids to Parts

```
SGfigure; view(-30,30); SGplot(SG,'m');  SG=SGanalyzeGroupParts(SG); SGplot(SG);
```

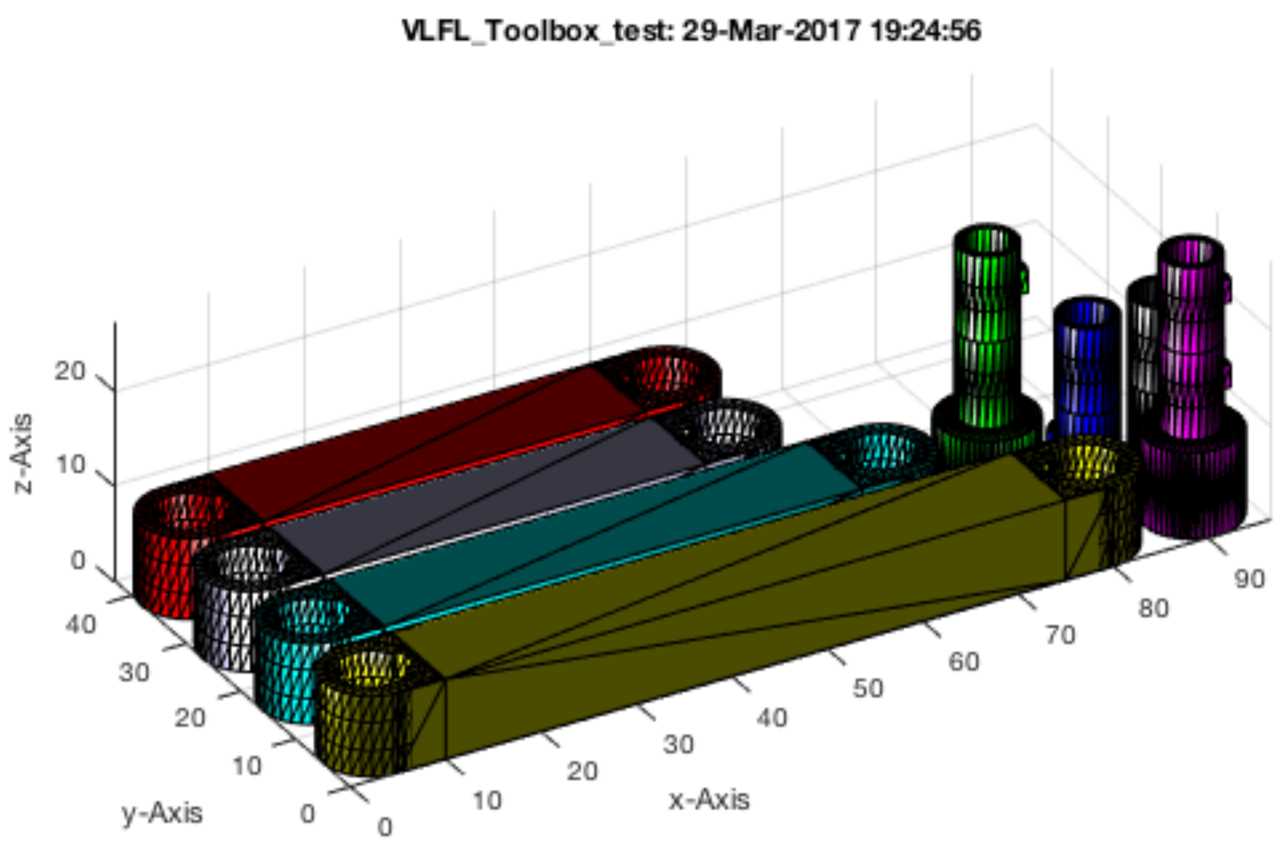
```
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
```



4. Now Arrange all Parts for Printing as Separated Solids

```
[~,SG]=SGpacking(SG); SGfigure; view(-30,30); SGplot(SG);
```

```
Packing 8 objects (h=61):
```



5. Now Write the Separated Parts into Different STL Files

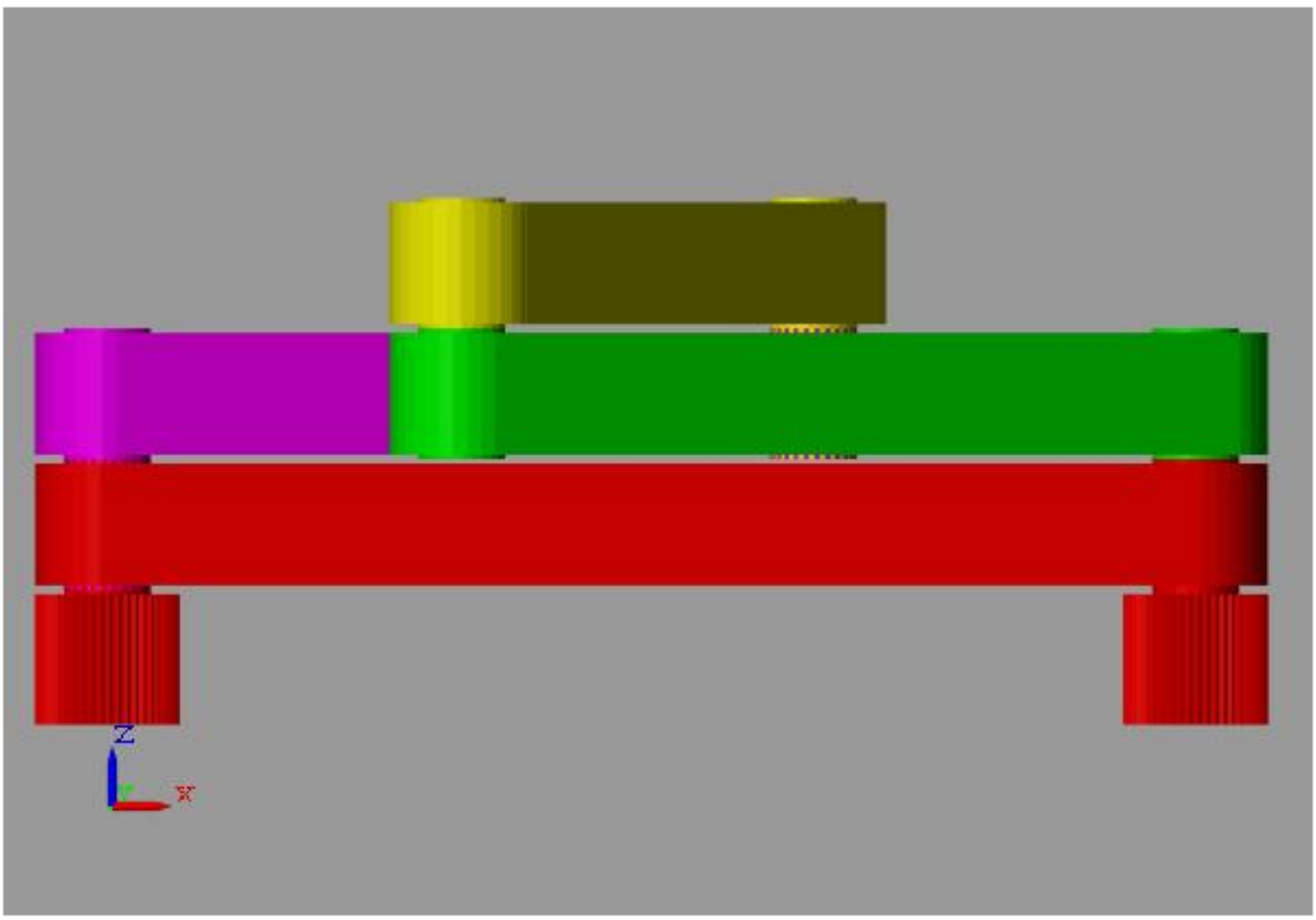
```
SGwriteSeparatedSTL(SG);
```

```
SGwriteSeparatedSTL: Writing 8 STL files in /Users/lueth/Desktop/EXP-2017-03-29/
```

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/lueth/Desktop/tmp_SG_LIB_EXP_26/Universa
l 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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
```

```
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:25:19 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: =======================================
===
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
======================================================================================================
===
```

Published with MATLAB® R2016b

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 2. Motivation for this tutorial
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing

```
function VLFL_EXP27
```

2. Motivation for this tutorial

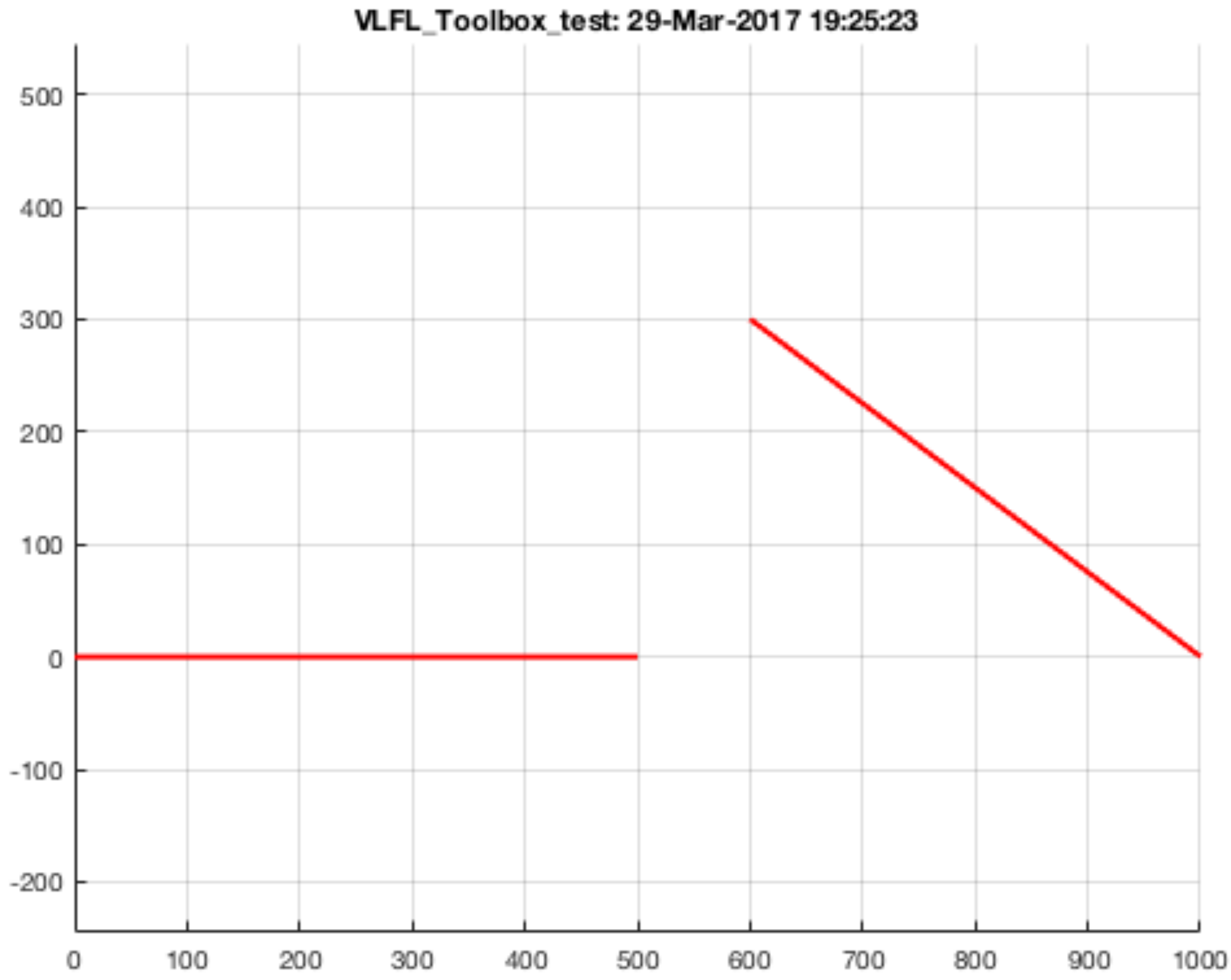
```
smbNewSystem ('SG_LIB_EXP_27');              % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/lueth/Desktop/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);
```

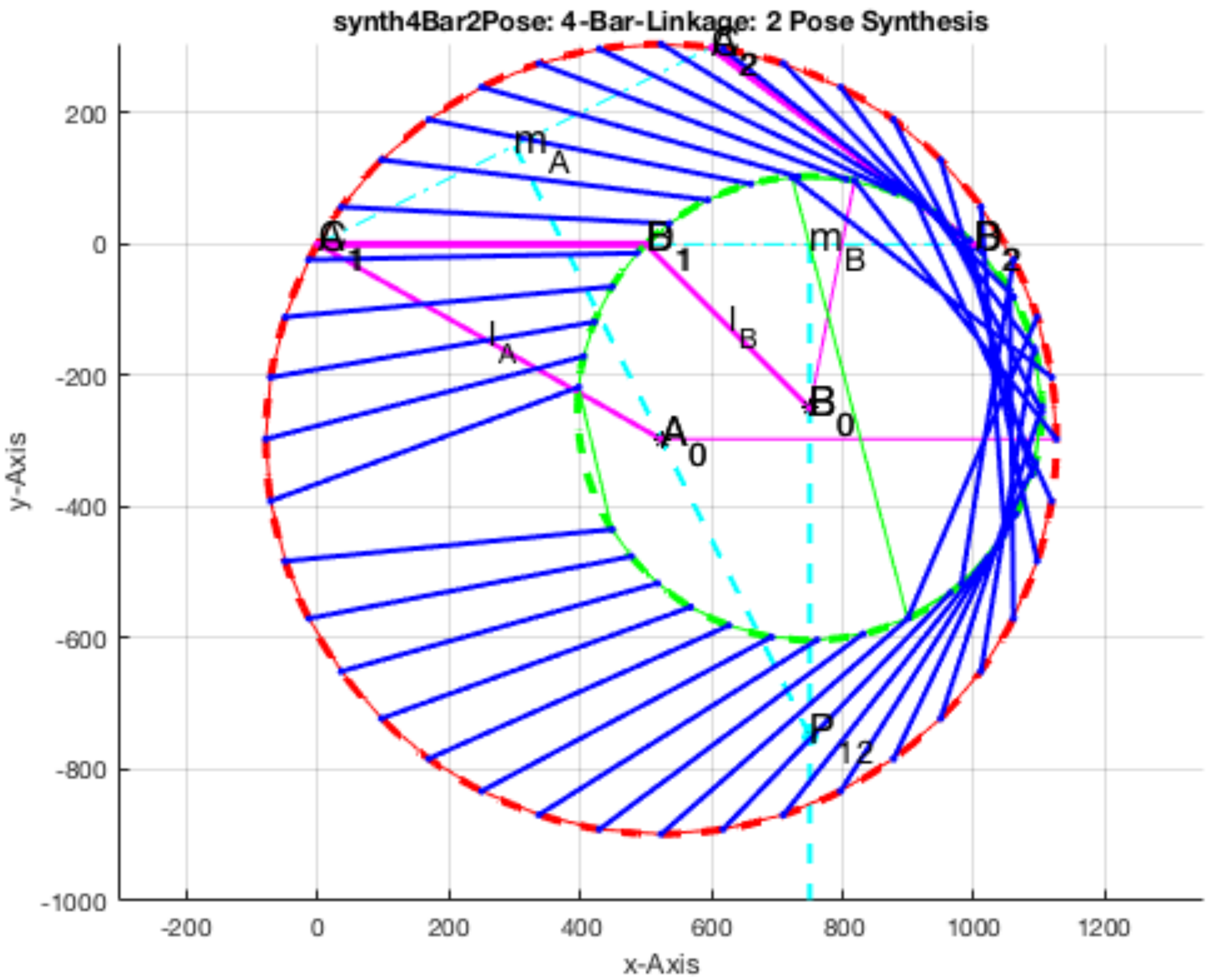


4. Find a General Solutions for the Two Pose Problem

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 =

   250     0     0


ans =

   800   150     0


ans =

   250     0     0


ans =

   800   150     0


ans =

  261.8034 -148.6068         0


ans =

   625  -125     0

imageFigureSaveMovie: Writing figure movie with 42 frames in file: /Users/lueth/Desktop/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

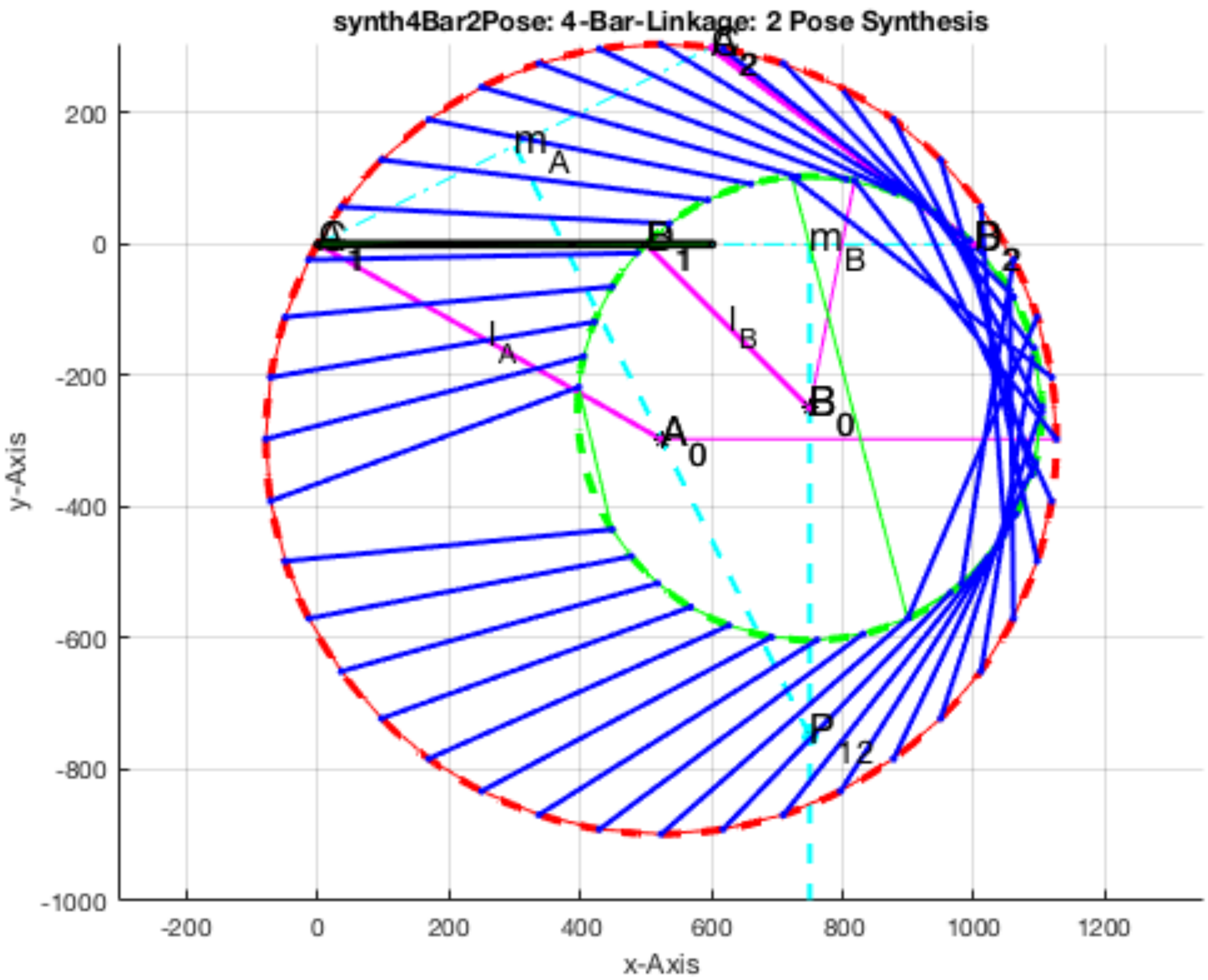
```
[A0,B0,A1,B1,P12]=synth4Bar2Pose(C1,D1,C2,D2,d);
db=P12-B0
k=(A0(2)-B0(2))/db(2)
B0=B0+k*db
A0
L1=norm(B0-A0)
L2=norm(B1-B0)
L3=norm(A1-B1)
L4=norm(A0-A1)
```

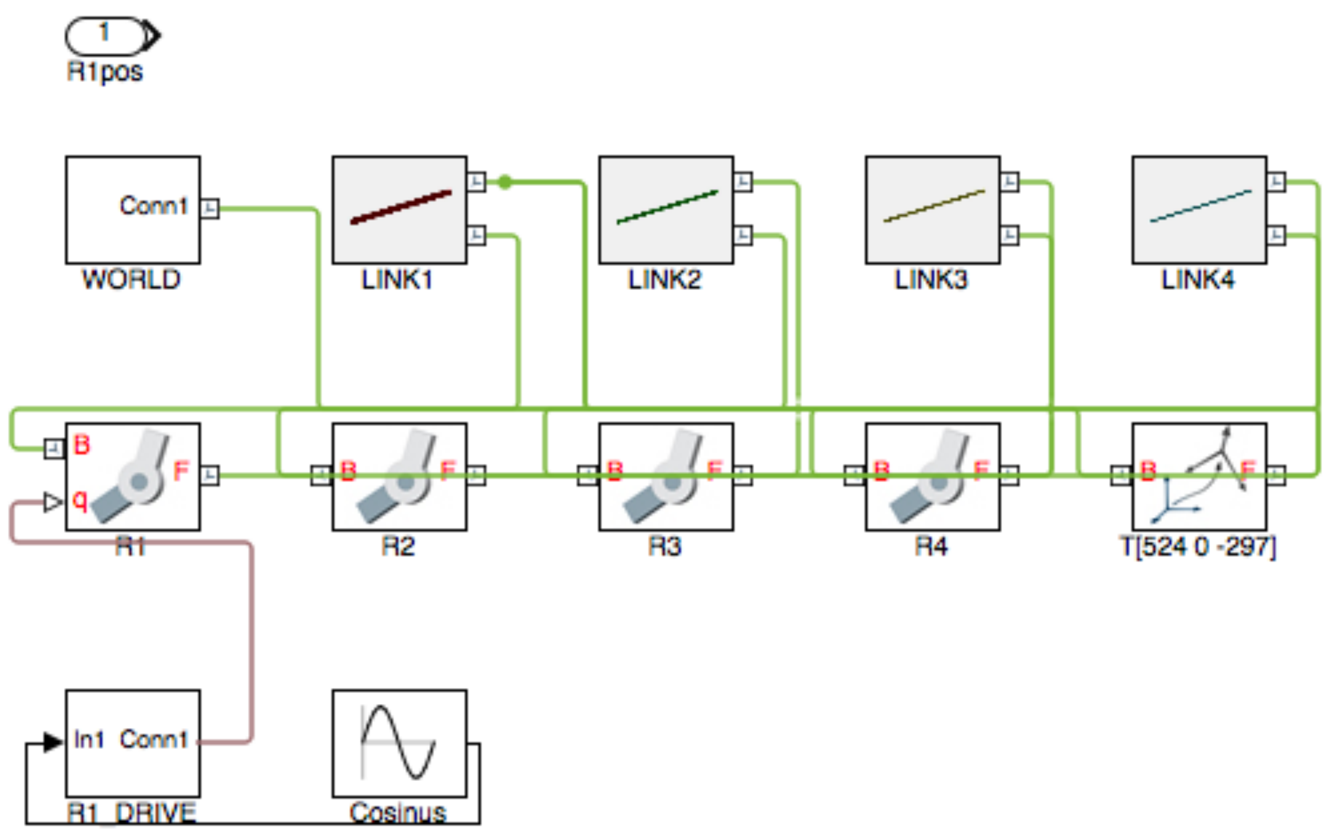
```
db =

    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 World Fr
ame
smbCreateDrive ('R1');
smbCreateSineWave ('Cosinus','R1_DRIVE/1');
ID=smbDrawNow;
```



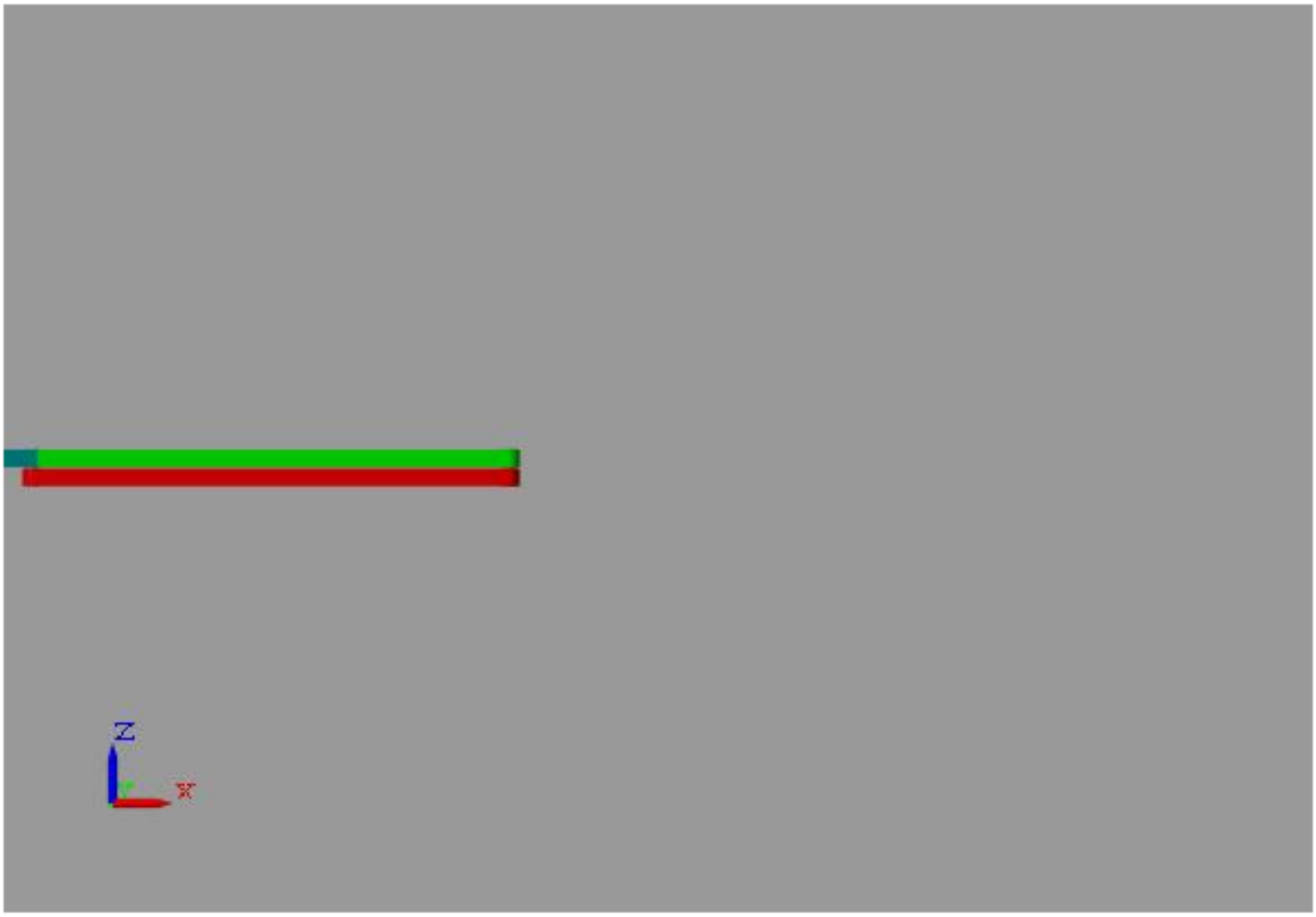


7. Show the Video of the Simulation

```
[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: Unable to reduce the step size without violating minimum step size of
```

```
7.41022264752519E-19 for 1 consecutive times at time 0.00020857922471329357.
Continuing simulation with the step size restricted to 7.41022264752519E-19 and
using an effective relative error tolerance of 0.0084742961193163945, which is
greater than the specified relative error tolerance of 0.001. This usually may
be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
7.4102226475252156E-19 for 2 consecutive times at time 0.0002085792247132943.
Continuing simulation with the step size restricted to 7.4102226475252156E-19
and using an effective relative error tolerance of 0.0012481106134613336, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
7.4102226654519361E-19 for 1 consecutive times at time 0.00020857922521788663.
Continuing simulation with the step size restricted to 7.4102226654519361E-19
and using an effective relative error tolerance of 0.011695417848713387, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
7.4102226654519621E-19 for 2 consecutive times at time 0.00020857922521788736.
Continuing simulation with the step size restricted to 7.4102226654519621E-19
and using an effective relative error tolerance of 0.010469936968904534, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
7.4102226654519881E-19 for 3 consecutive times at time 0.00020857922521788809.
Continuing simulation with the step size restricted to 7.4102226654519881E-19
and using an effective relative error tolerance of 0.00124811061648053, which is
greater than the specified relative error tolerance of 0.001. This usually may
be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
......Creating a new video file (NO SOUND/2016b): '/Users/lueth/Desktop/tmp_SG_LIB_EXP_27/2 Pose S
yntheses 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%
```



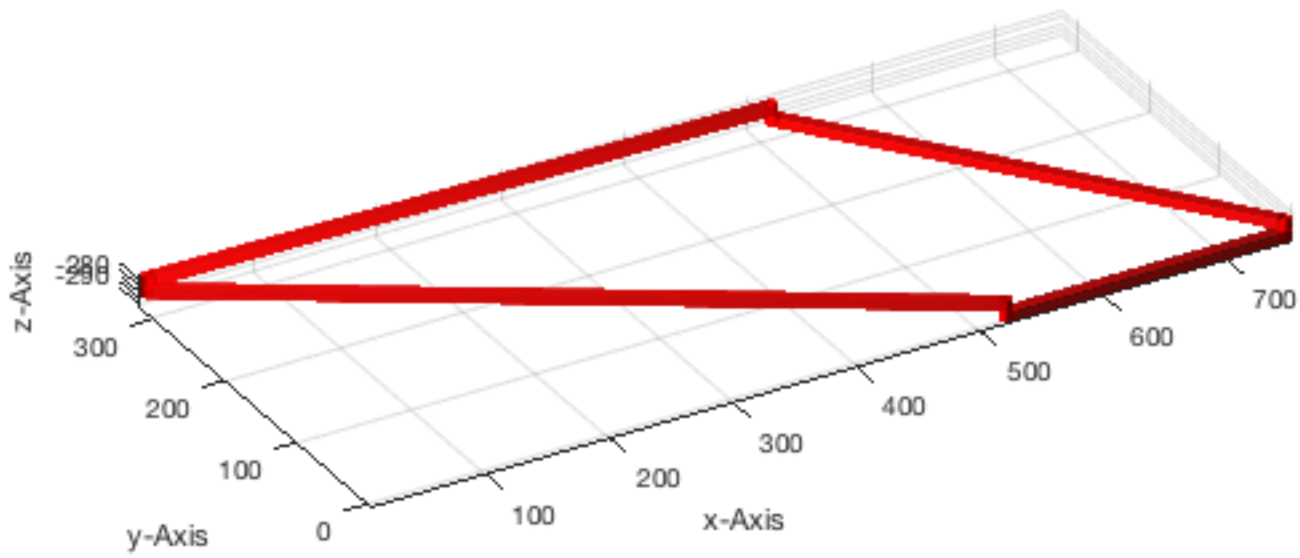
8. Now Create the Solid Geoemtry at time 0.78 seconds

```
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 SECONDS
===================================================================================================
=
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: Unable to reduce the step size without violating minimum step size of
5.3944822254360946E-19 for 1 consecutive times at time 0.00015184117587706779.
Continuing simulation with the step size restricted to 5.3944822254360946E-19
and using an effective relative error tolerance of 0.0043035671698391821, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
5.3944822645165932E-19 for 1 consecutive times at time 0.00015184117697708603.
Continuing simulation with the step size restricted to 5.3944822645165932E-19
and using an effective relative error tolerance of 0.0061589696648945867, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
```

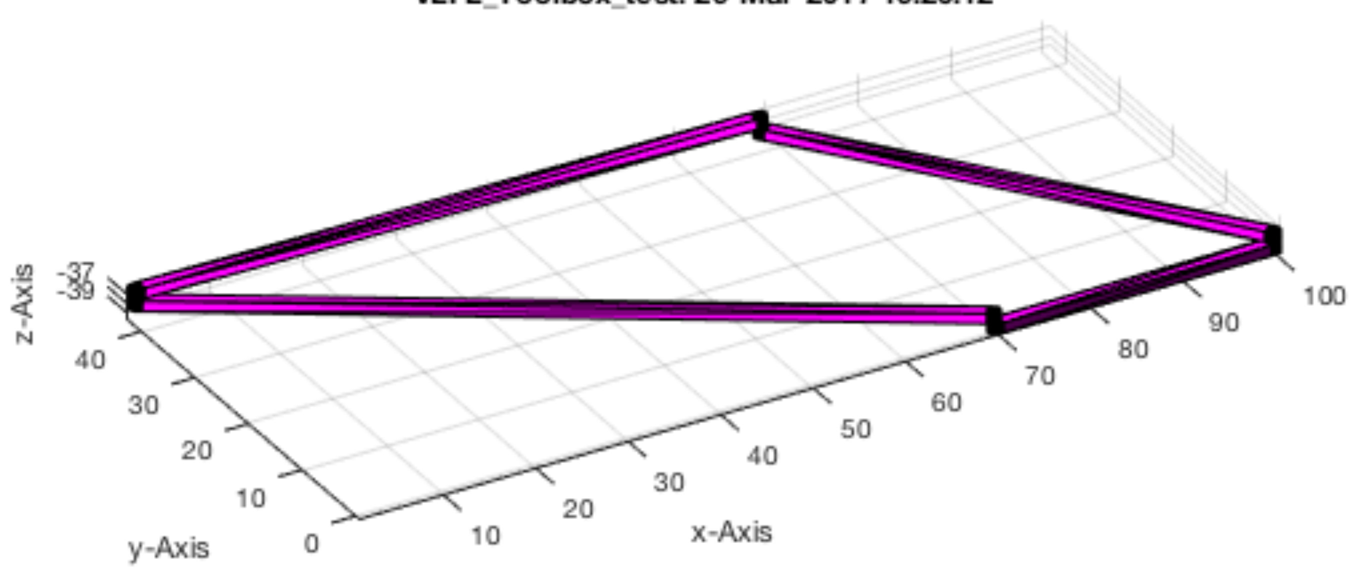
```
5.3944822645166125E-19 for 2 consecutive times at time 0.00015184117697708658.
Continuing simulation with the step size restricted to 5.3944822645166125E-19
and using an effective relative error tolerance of 0.0087521093125372748, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps
Warning: Unable to reduce the step size without violating minimum step size of
5.3944822645166317E-19 for 3 consecutive times at time 0.00015184117697708712.
Continuing simulation with the step size restricted to 5.3944822645166317E-19
and using an effective relative error tolerance of 0.0012266067701044677, which
is greater than the specified relative error tolerance of 0.001. This usually
may be caused by the high stiffness of the system. Please check the model
'SG_LIB_EXP_27' or increase Number of consecutive min steps

Simulink.SimulationOutput:

    simlog: [1×1 simscape.logging.Node]
      sout: [1×1 Simulink.SimulationData.Dataset]
      tout: [189×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.

LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_27/sbm_temp_LINK1.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_27/sbm_temp_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_27/sbm_temp_LINK3.stl
Header:
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/lueth/Desktop/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
===================================================================================================
=
```

VLFL_Toolbox_test: 29-Mar-2017 19:26:11




```
SGfigure; view(-30,30); SGplot(SG,'m'); % SGanalyzeGroupParts(SG);
```

VLFL_Toolbox_test: 29-Mar-2017 19:26:12



Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:26:15 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: =====================================
===
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
====================================================================================================
===
```

Exercise 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)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 2. Motivation for this tutorial
- 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

1. Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

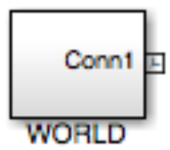
- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing

```
% function VLFL_EXP28
```

2. Motivation for this tutorial

```
smbNewSystem ('SG_LIB_EXP_28',[+5 +5 +5]);           % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/lueth/Desktop/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.

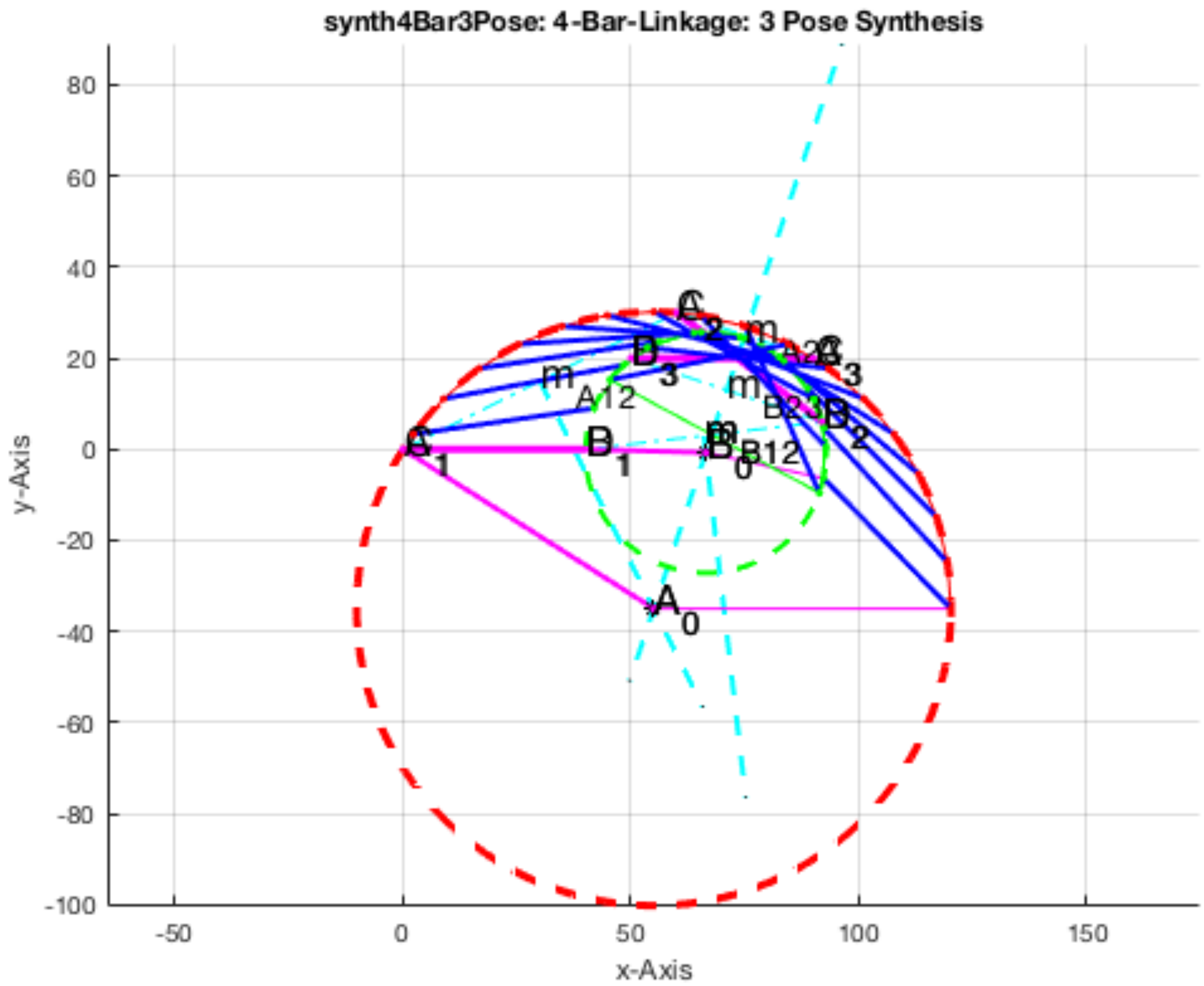
```
d=[0 0];
C1=[ 0 0];
D1=[40 0];
C2=[60 30];
D2=[100 0];
C3=[90 20];
D3=[50 20];

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 =

    20     0     0


ans =

    76    18     0


ans =

    70    20     0


ans =

   27.5000  -17.5000         0


ans =

   53.2143   -0.3571         0

imageFigureSaveMovie: Writing figure movie with 42 frames in file: /Users/lueth/Desktop/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
```

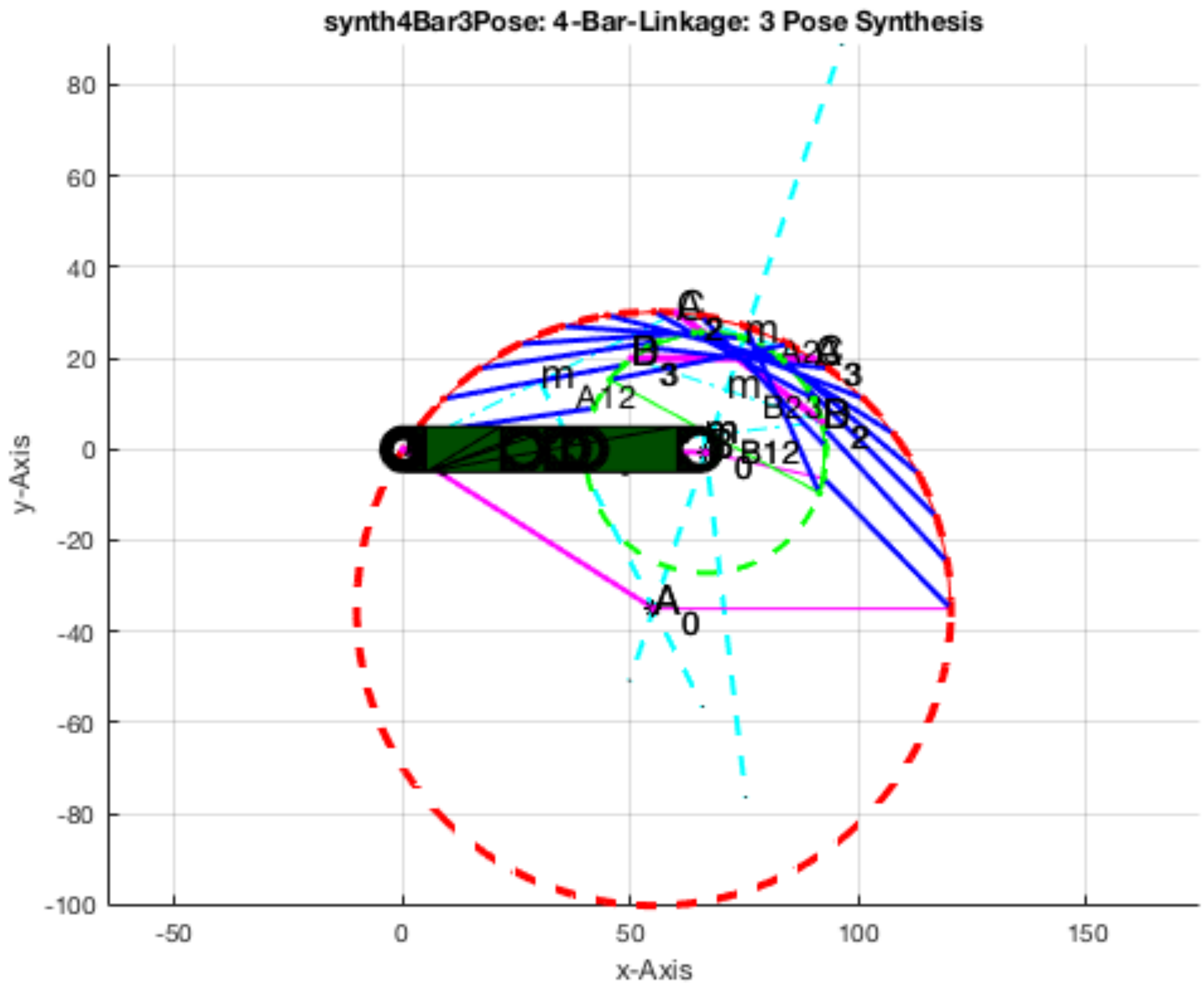
```
A0 =

   55.0000  -35.0000


B0 =

   66.4286   -0.7143


L1 =
```

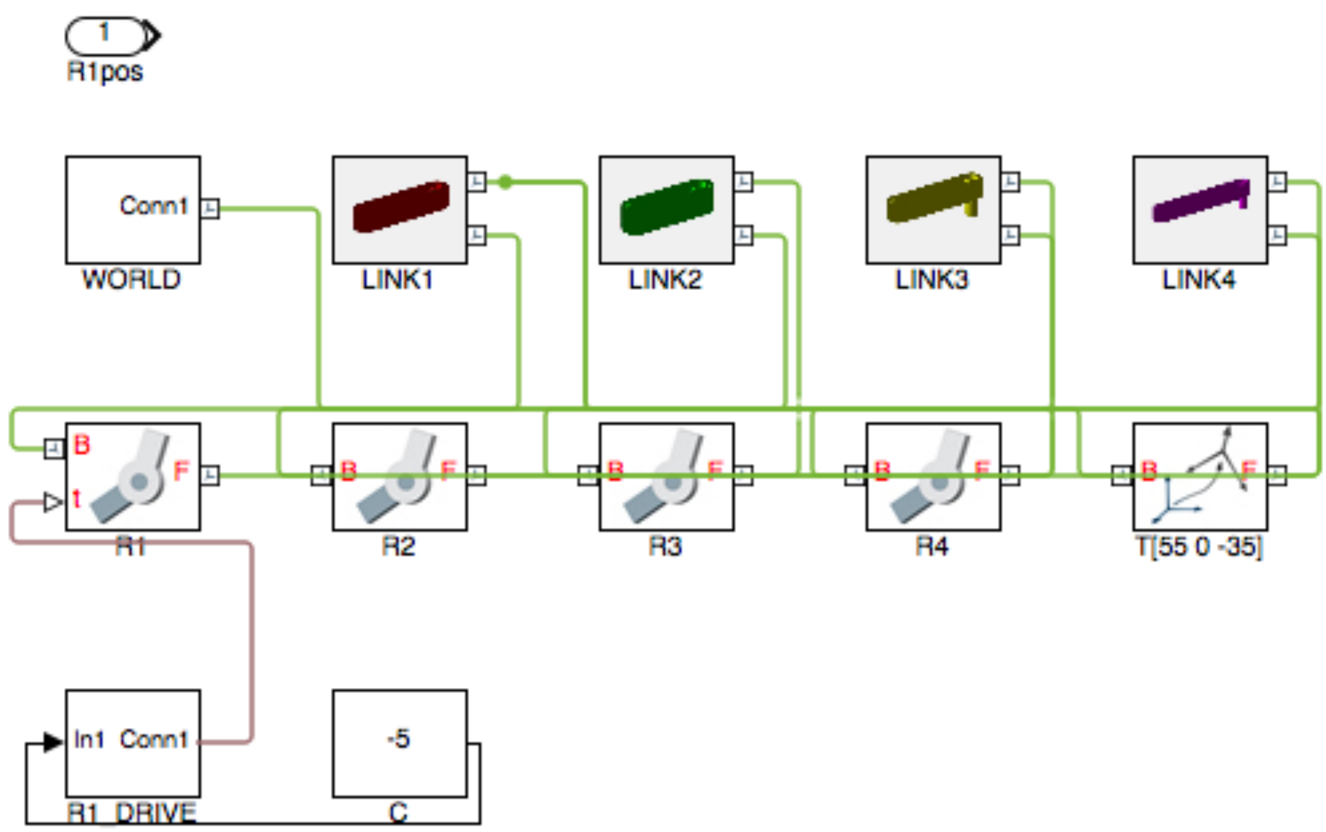
```
   36.1403


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 Linkag
e to World Frame
smbCreateDrive ('R1');
smbSetJointInputTorque('R1');
smbCreateBlockConst('C','R1_DRIVE/1',-5)
ID=smbDrawNow;
smbSimulate(4);
```

```
phi =

    1.2490
```



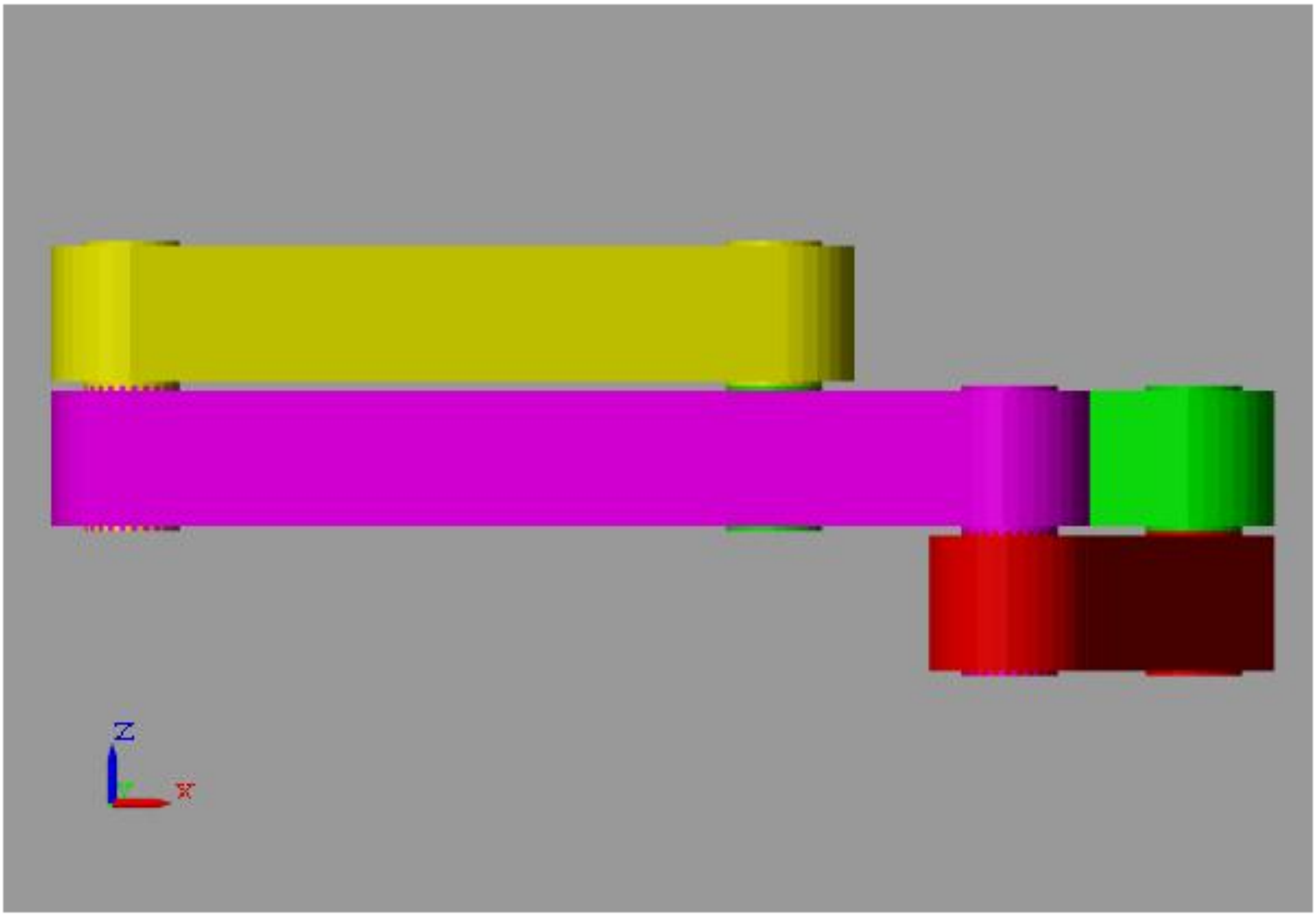


7. Show the Video of the Simulation

```
[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/lueth/Desktop/tmp_SG_LIB_EXP_28/3 Pose S
```

```
yntheses 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%
```

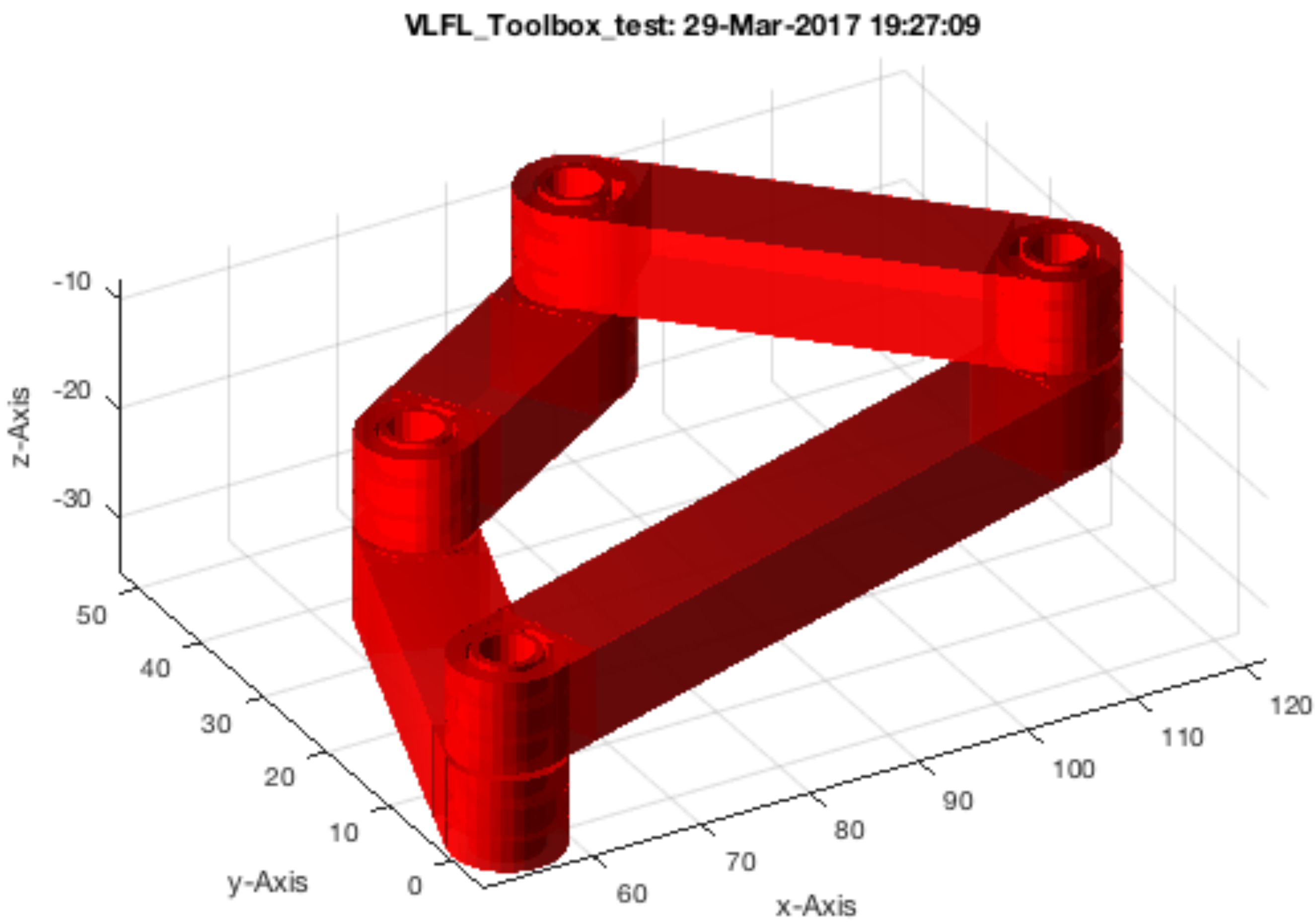


8. Now Create the Solid Geoemtry at time 0.78 seconds

```
SG=smbFullModelSimulation(0.74);
% SG=SGmagnifyVL(SG,'',[100 100 100]);
SGwriteSTL(SG,smbFilename('3-Pose-Synth'));
```

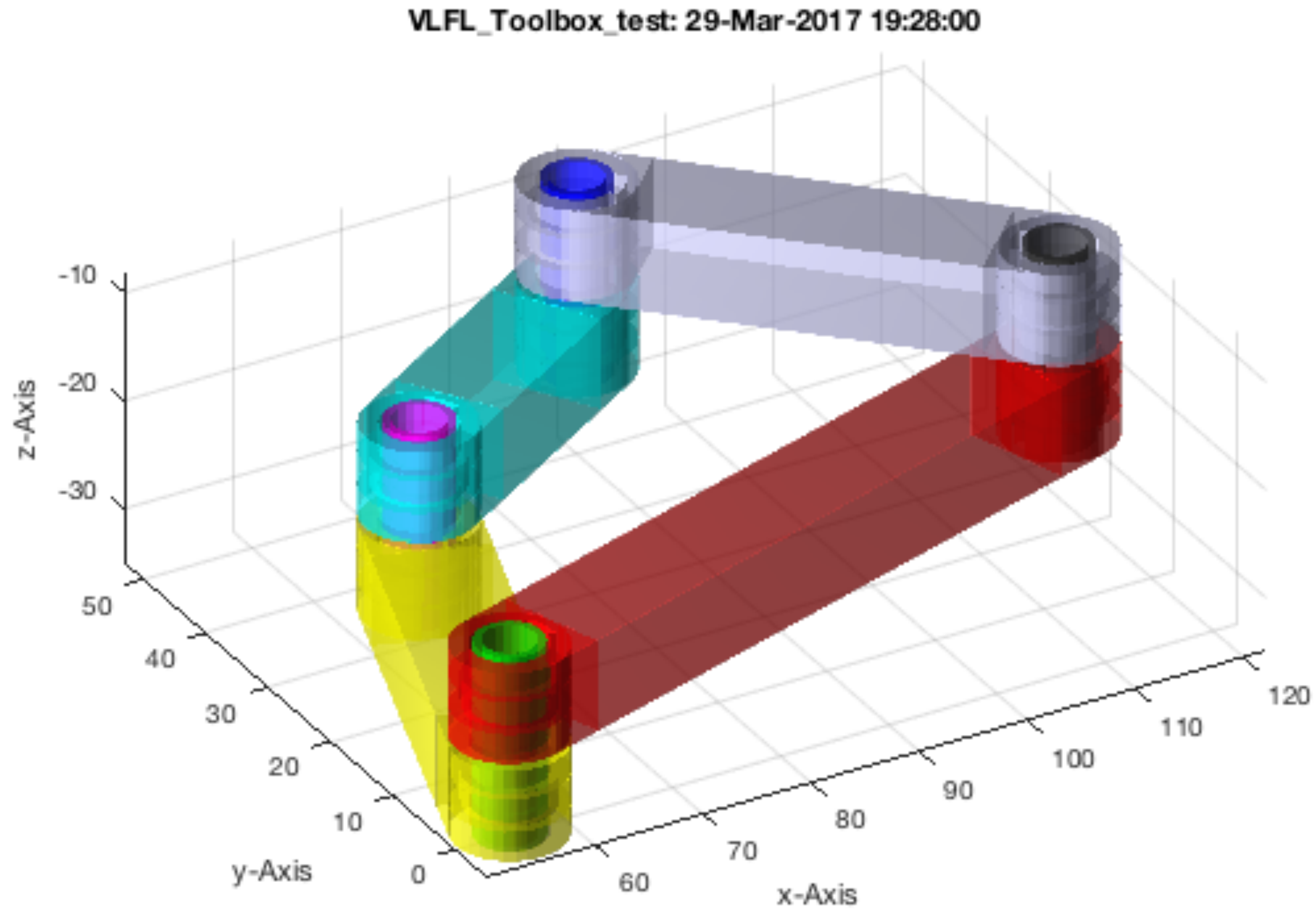
```
CREATING A FULL SOLID-MOVEMENT SIMULATION-MODEL 'SG_LIB_EXP_28' THAT RUNS At LEAST 0.74 SECONDS
===================================================================================================
=
Adding frame sensors for all solids of the model
Add frame sensors for 'LINK1.SG'
Add frame sensors for 'LINK2.SG'
Add frame sensors for 'LINK3.SG'
Add frame sensors for 'LINK4.SG'
===================================================================================================
=

Simulink.SimulationOutput:

    simlog: [1×1 simscape.logging.Node]
      sout: [1×1 Simulink.SimulationData.Dataset]
      tout: [222×1 double]
      xout: [1×1 Simulink.SimulationData.Dataset]

Use 'get' to access a variable by name.
Use 'getSimulationMetadata' to access metadata about the simulation.
```

```
LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_28/sbm_temp_LINK1.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_28/sbm_temp_LINK2.stl
Header:
Number of facets: 2404
0..LOADING BINARY STL-File: /Users/lueth/Desktop/tmp_SG_LIB_EXP_28/sbm_temp_LINK3.stl
Header:
Number of facets: 2584
0..LOADING BINARY STL-File: /Users/lueth/Desktop/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);
```

```
5% 10% 15% 20% 25% 30% 35% 40% 45% 50% 55% 60% 65% 70% 75% 80% 85% 90% 95% 100%
SGanalyzeGroupParts: 8 separated parts found.
```



Final Remarks

```
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:28:03 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ===================================
===
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==================================================================================================
===
```

Published with MATLAB® R2016b

Example 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)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- 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

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Example 29: Create a multi body simulation using several mass points

```
% function VLFL_EXP29
```

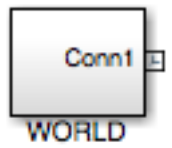
Motivation for this tutorial

Showing a finite element mass spring system

1. Create a SimMultiBody system for a Mass - Spring - Damper - System

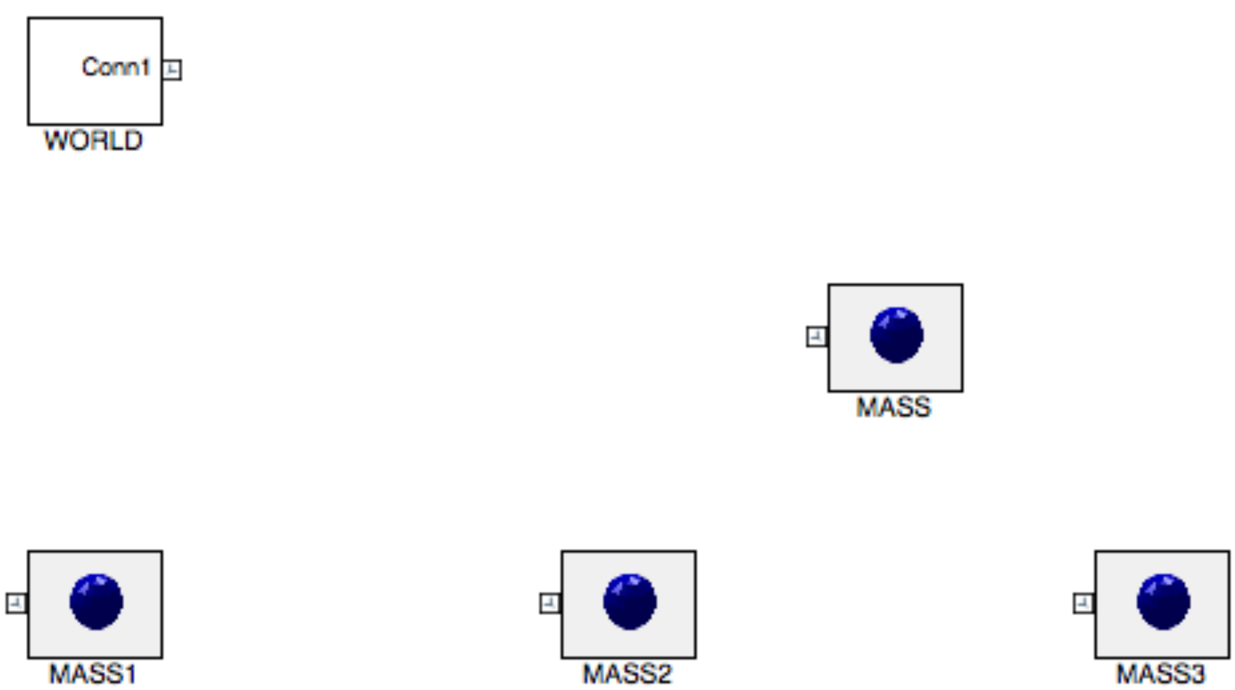
```
smbNewSystem ('SG_LIB_EXP_29',[0 0 -9.81])    % Creates the mechansim diagramm
```

```
Creating temporary directory '/Users/lueth/Desktop/tmp_SG_LIB_EXP_29/'
```



2 Create four mass points

```
smbCreateSGMass;
smbCreateSGMass;
smbCreateSGMass;
smbCreateSGMass;
smbDrawNow;
```

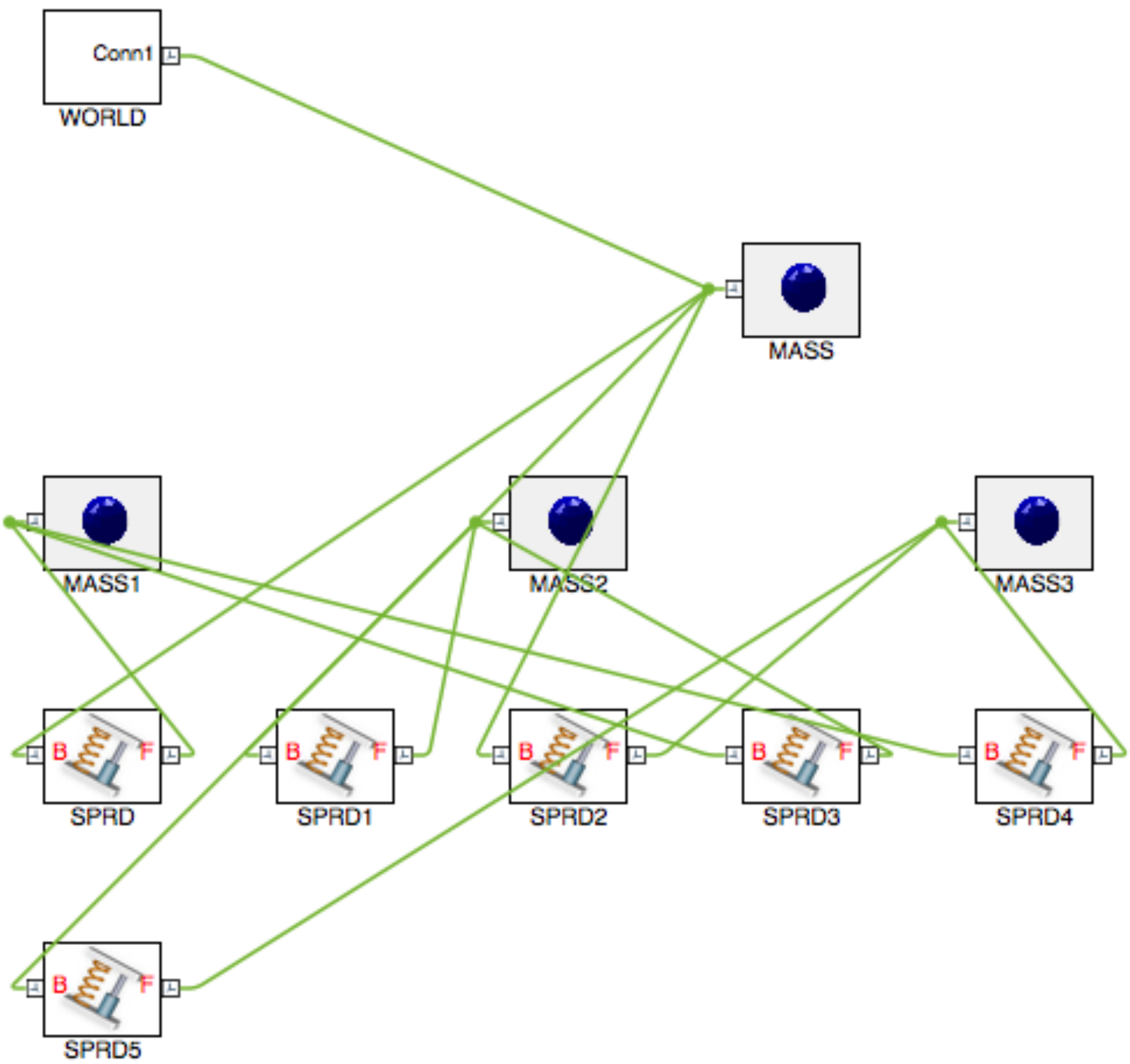


2 Create six springs between the masses

```
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;
```

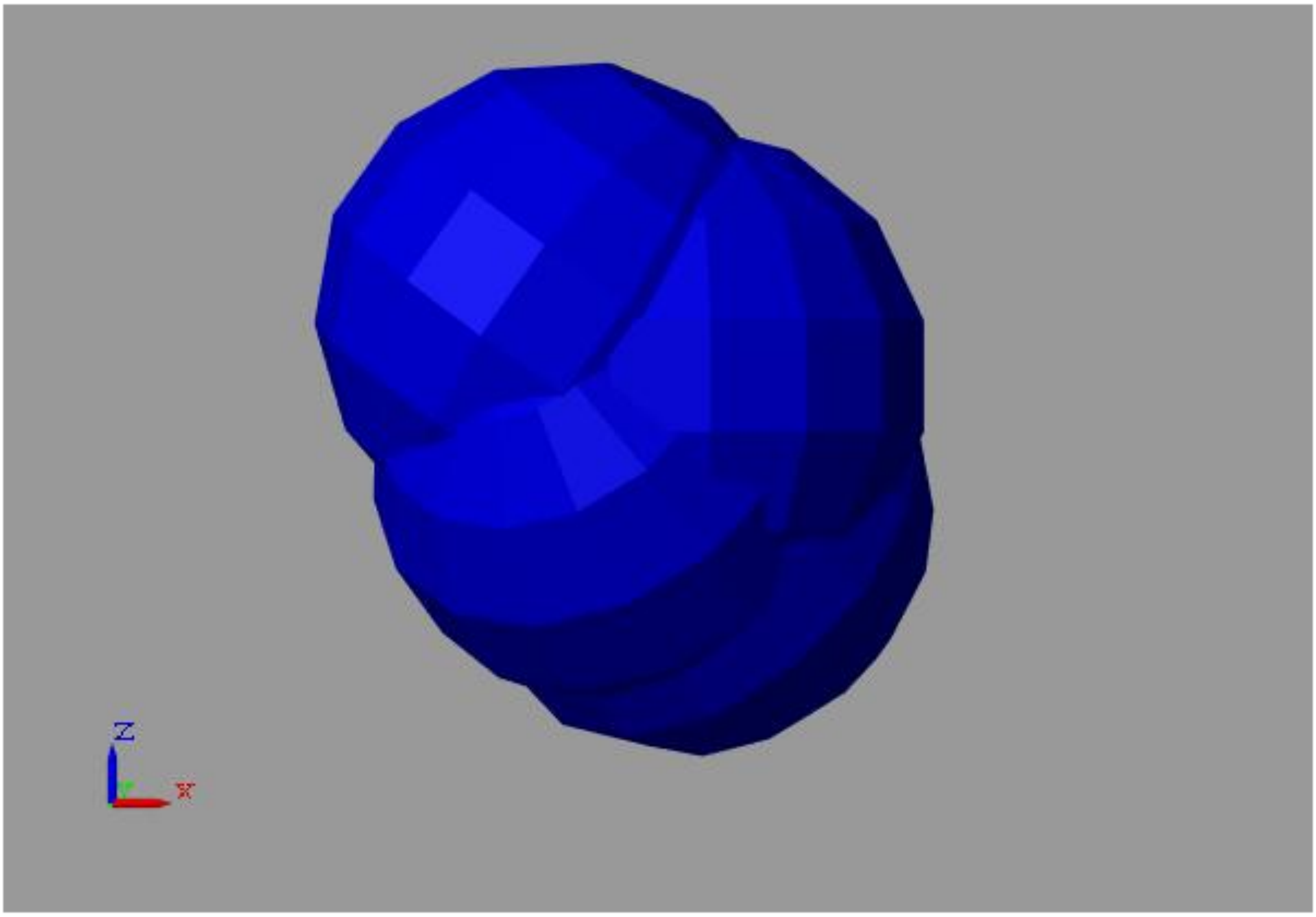


4. Show the Simulation

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);
```

```
......Creating a new video file (NO SOUND/2016b): '/Users/lueth/Desktop/tmp_SG_LIB_EXP_29/SG-Lib T
utorial #29-Mass-Spring-Nets.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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:28:36 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ===================================
===
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==================================================================================================
===
```

Published with MATLAB® R2016b

Exercise 30: Create Mechanisms using Universal Planar Links

2017-02-10: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- Motivation for this tutorial
- 1. Plotting points (PL) and vertices (3D)
- 2. Plotting lines
- 3. Plotting angles
- 4. Plotting coordinate
- 8. Adding text to the drawings
- 9. Helpful generic polygons for
- Final Remarks

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing

- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing

Motivation for this tutorial

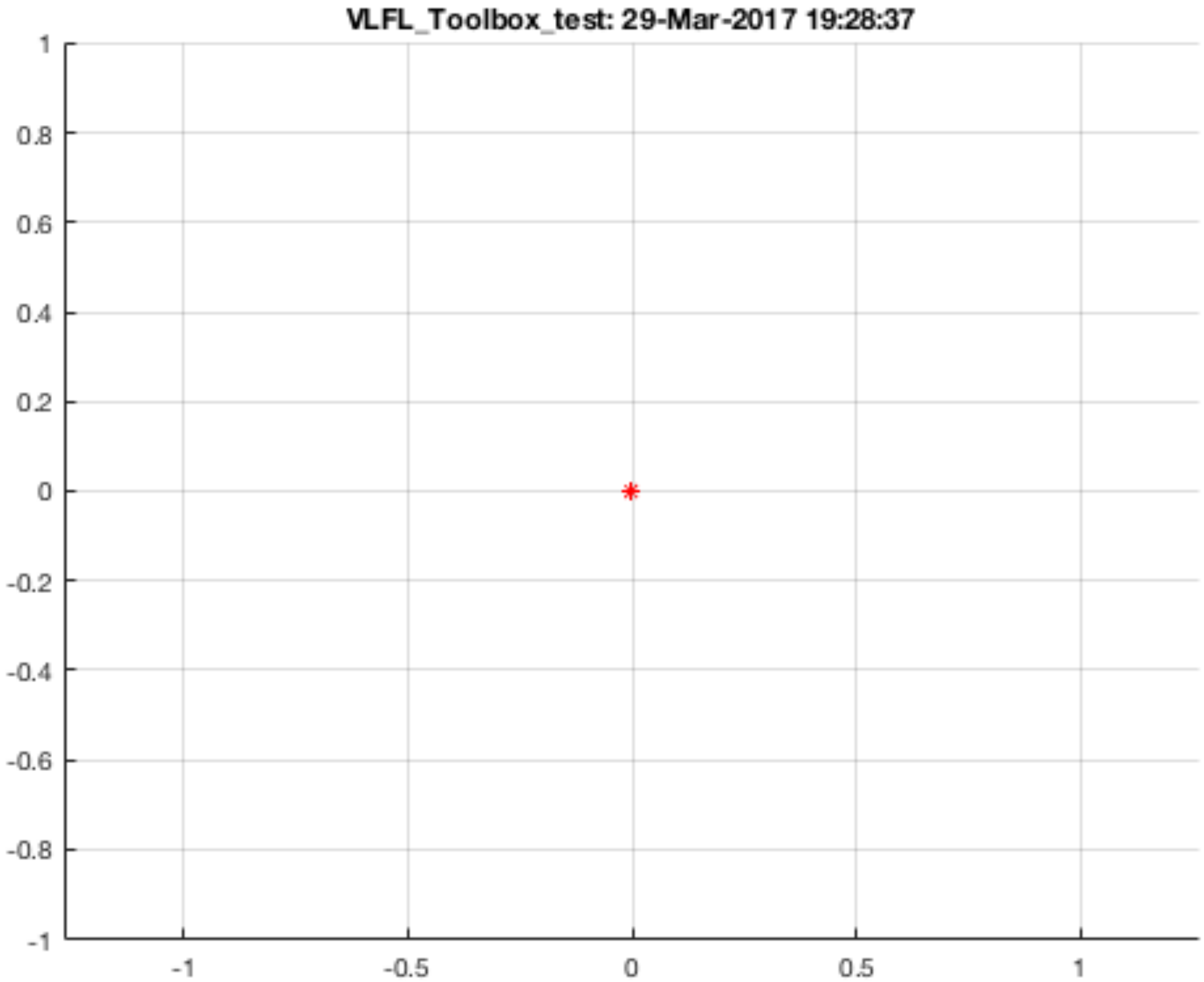
In this tutorial, features are introduced that allow you to create drawings for the descriptive geometry using Matlab. The goal is to display lines, planes, spaces and polygons, surfaces and surface-bounded volumes and to label them mathematically (Tex-style). some functions are based in individual points 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,

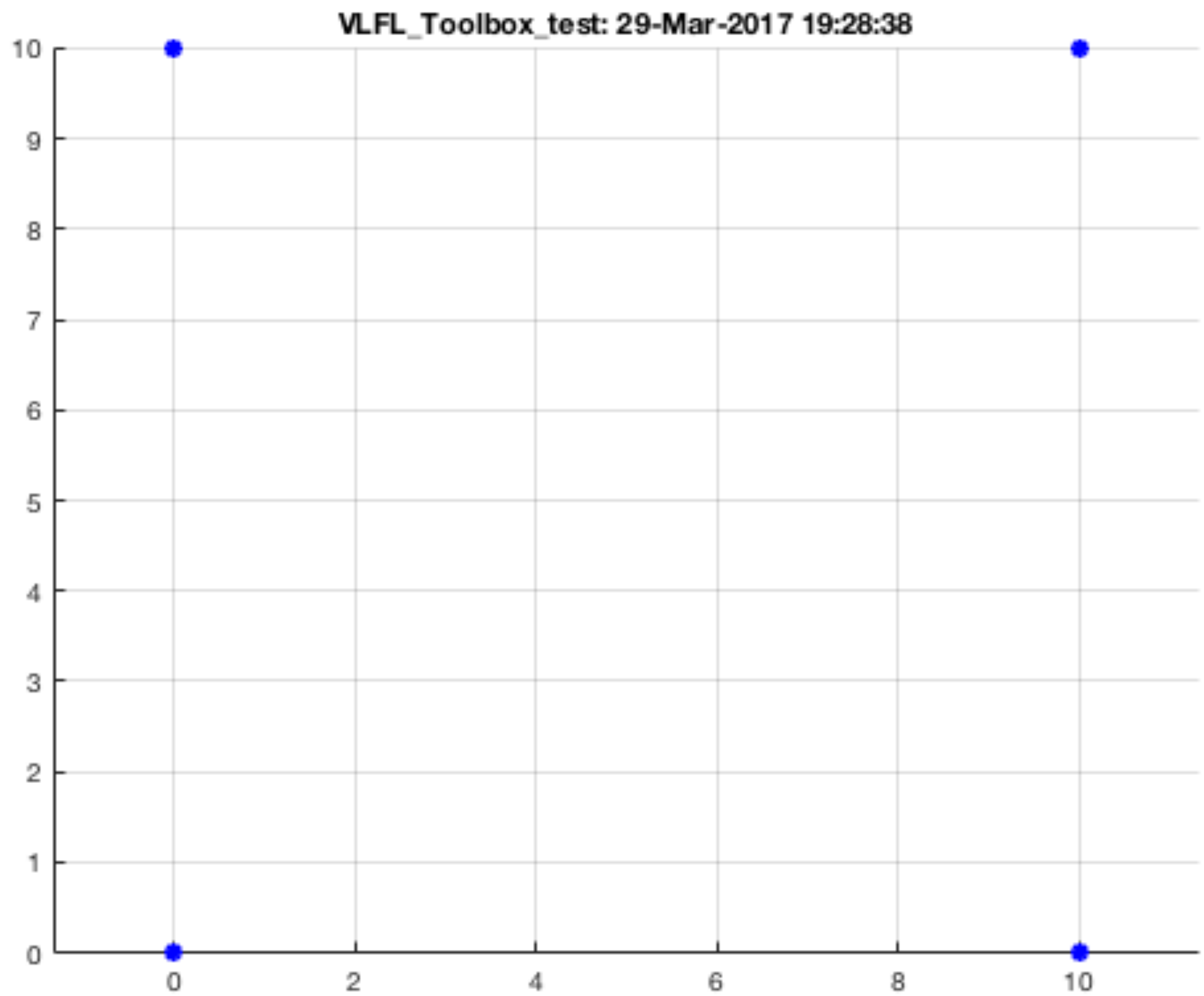
1. Plotting points (PL) and vertices (3D)

```
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);
```

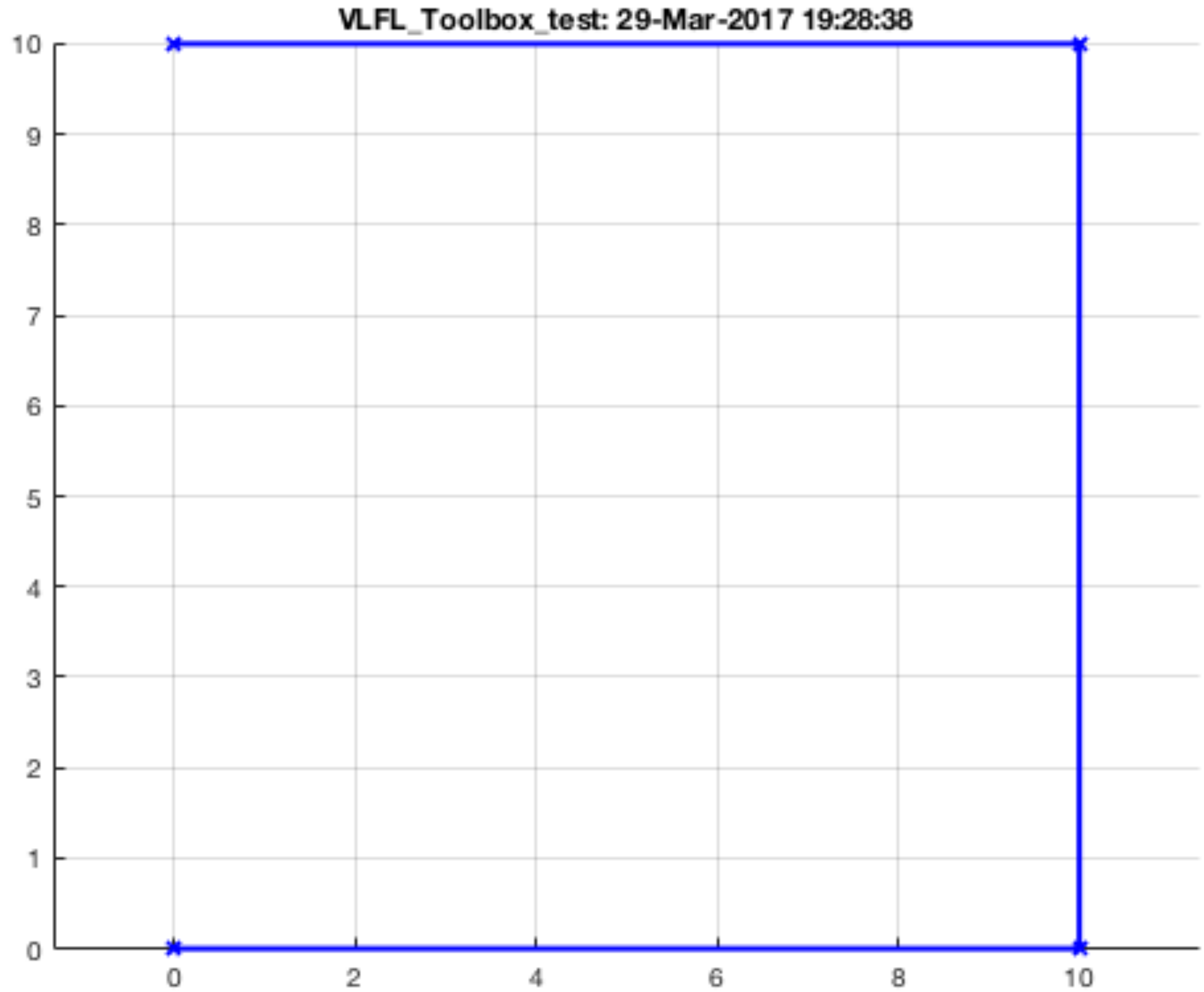
```
p =

     0     0


p =

     0
     0


v =

     0     0     0


v =

     0
     0
     0
```



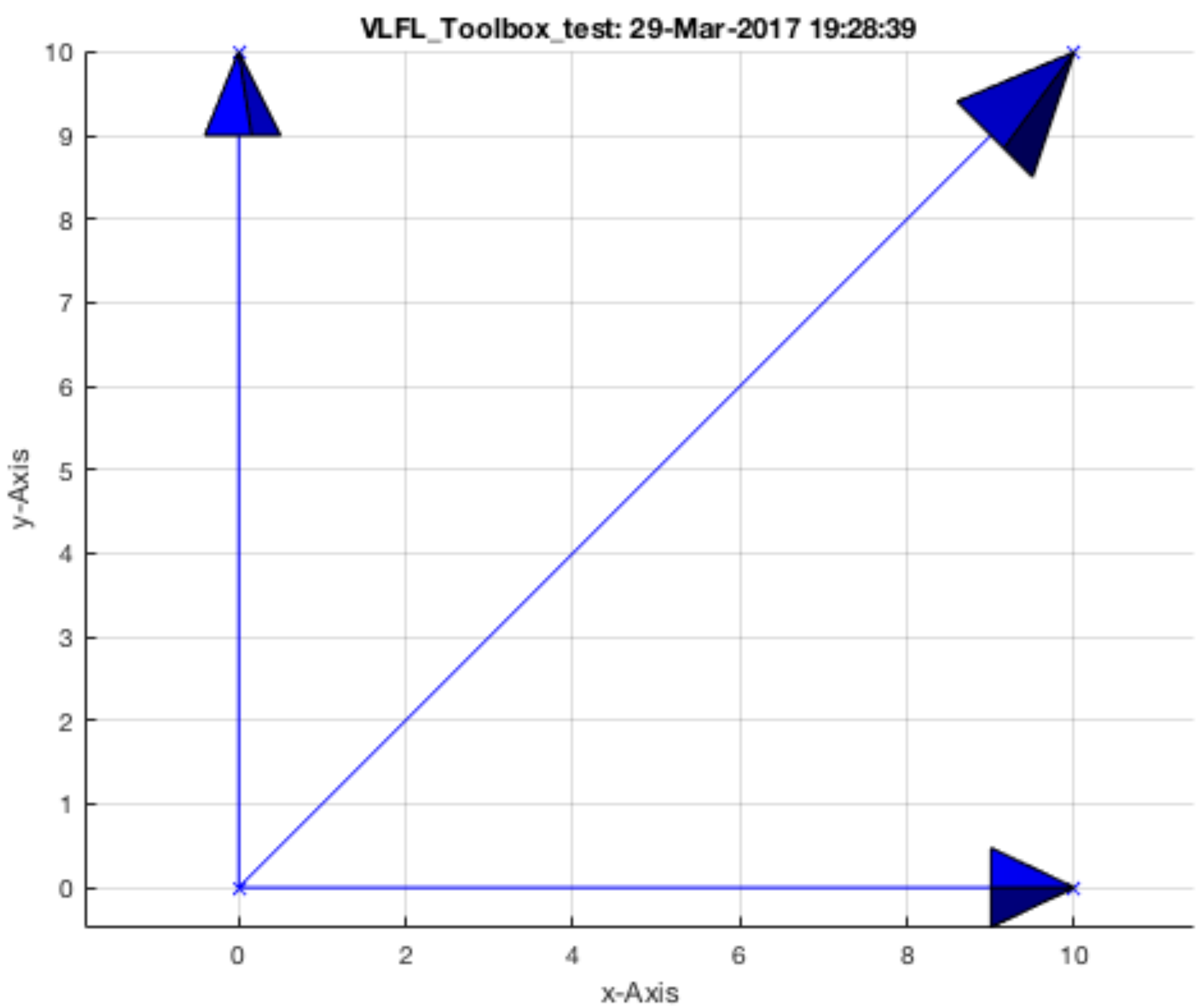

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'b*',2);          % point plot of point list
```



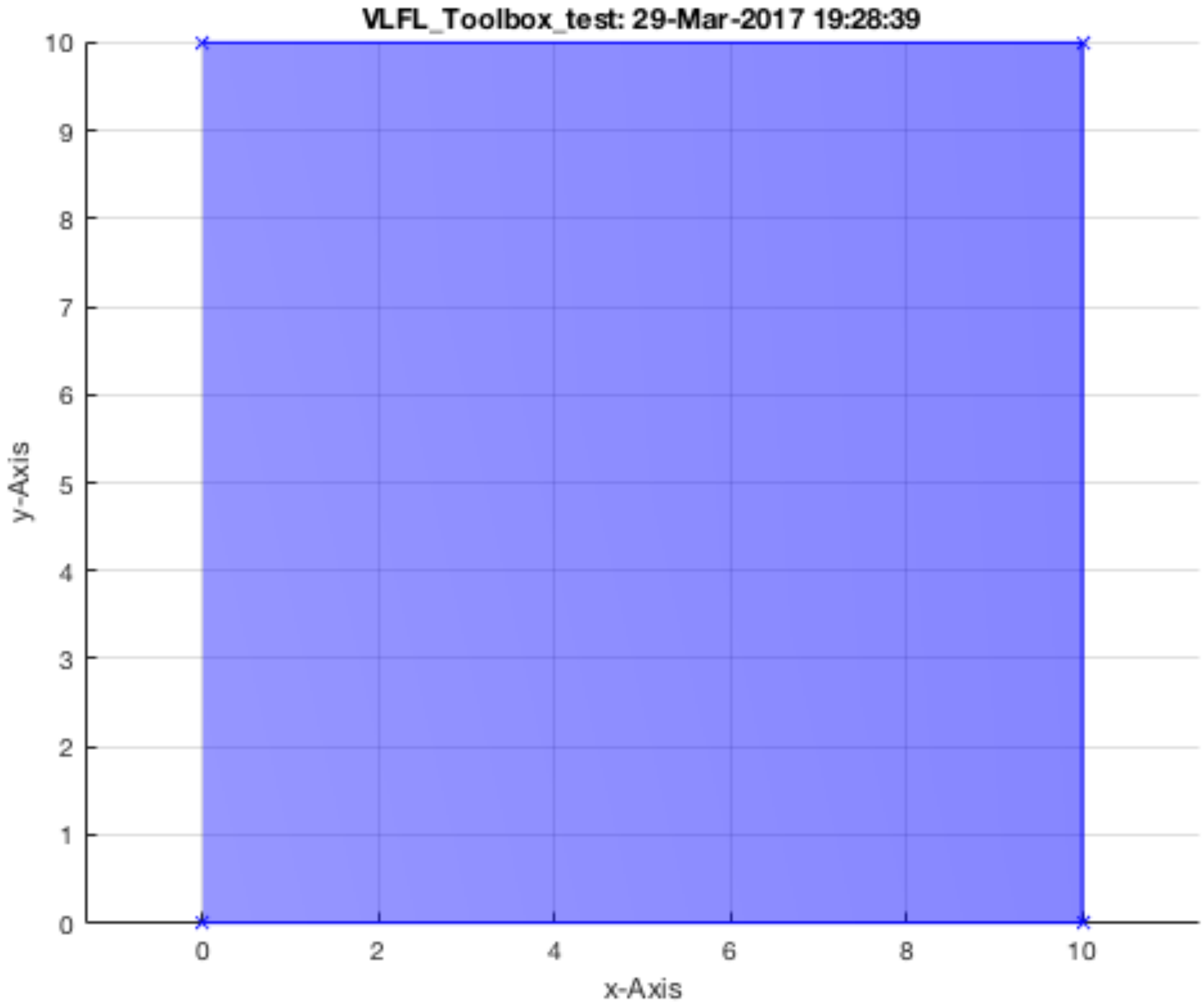

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',2);         % Line plot of point list
```



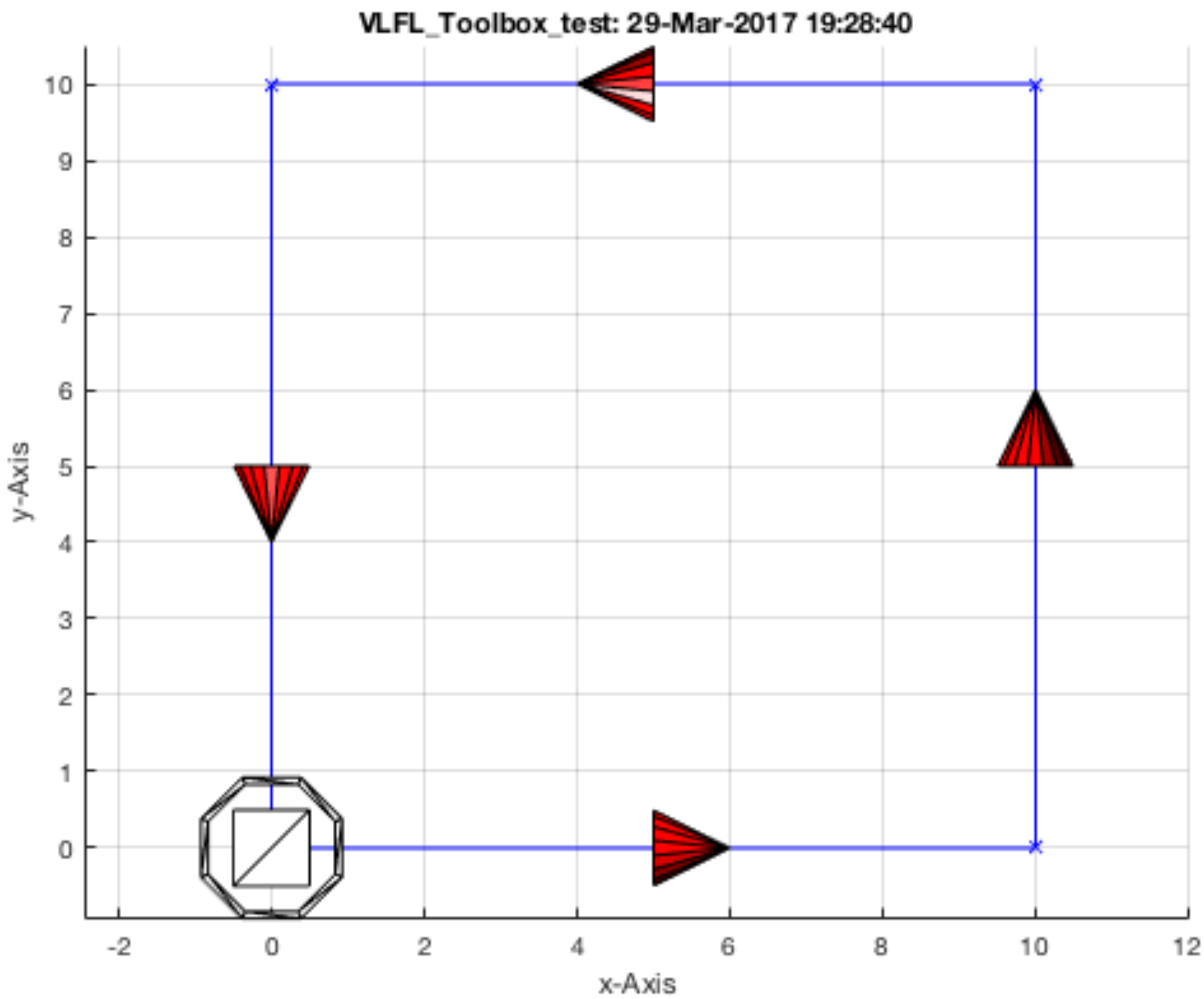

```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',1,1);          % Vector plot of point list
```




```
SGfigure; PLplot([0 0;10 0;10 10; 0 10],'bx-',1,'',0.5);     % Surface enlosed by point list
```

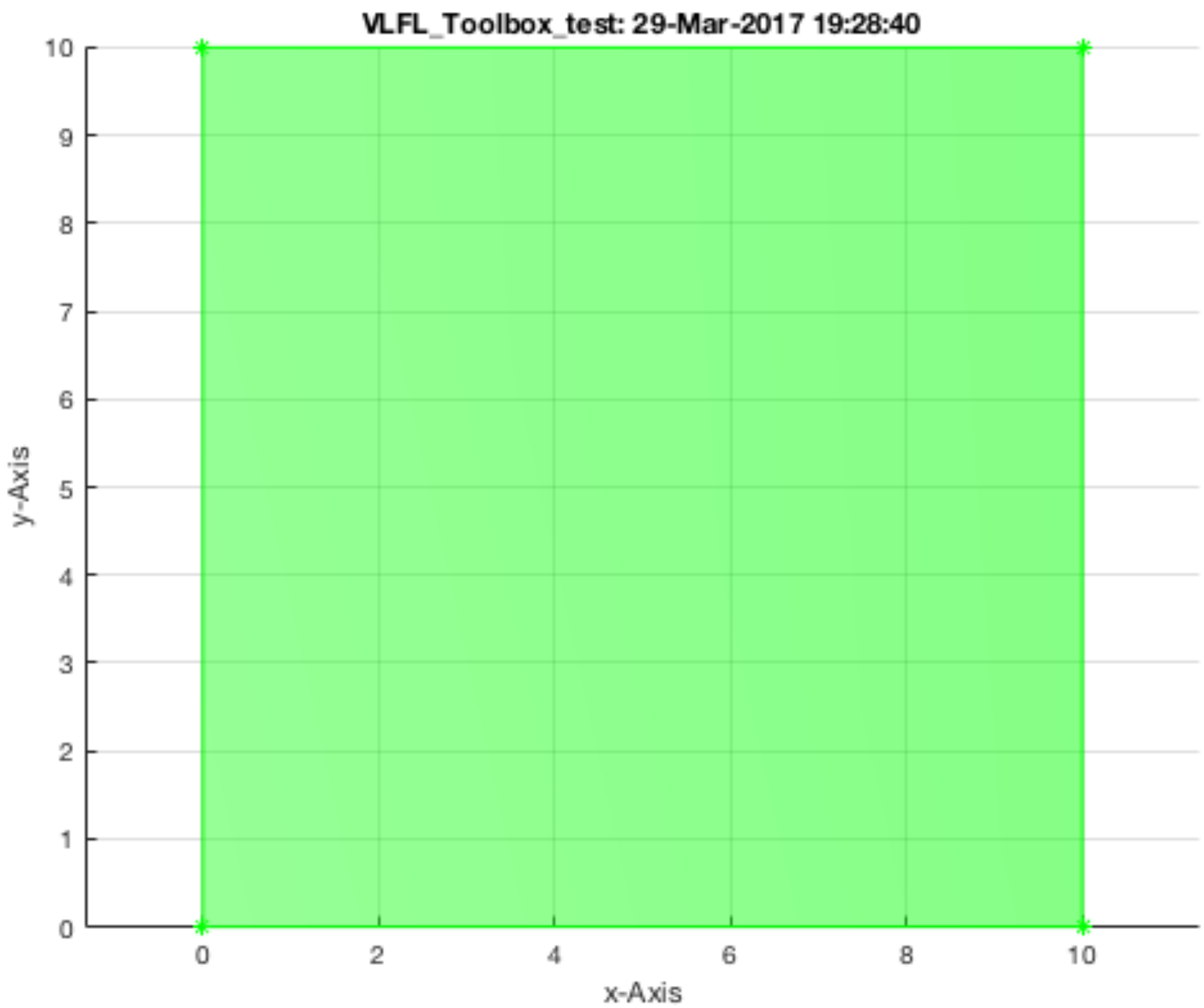



```
SGfigure; CPLplot([0 0;10 0;10 10; 0 10],'bx-',1,1,1,1);    % Plotting closed polygon
```



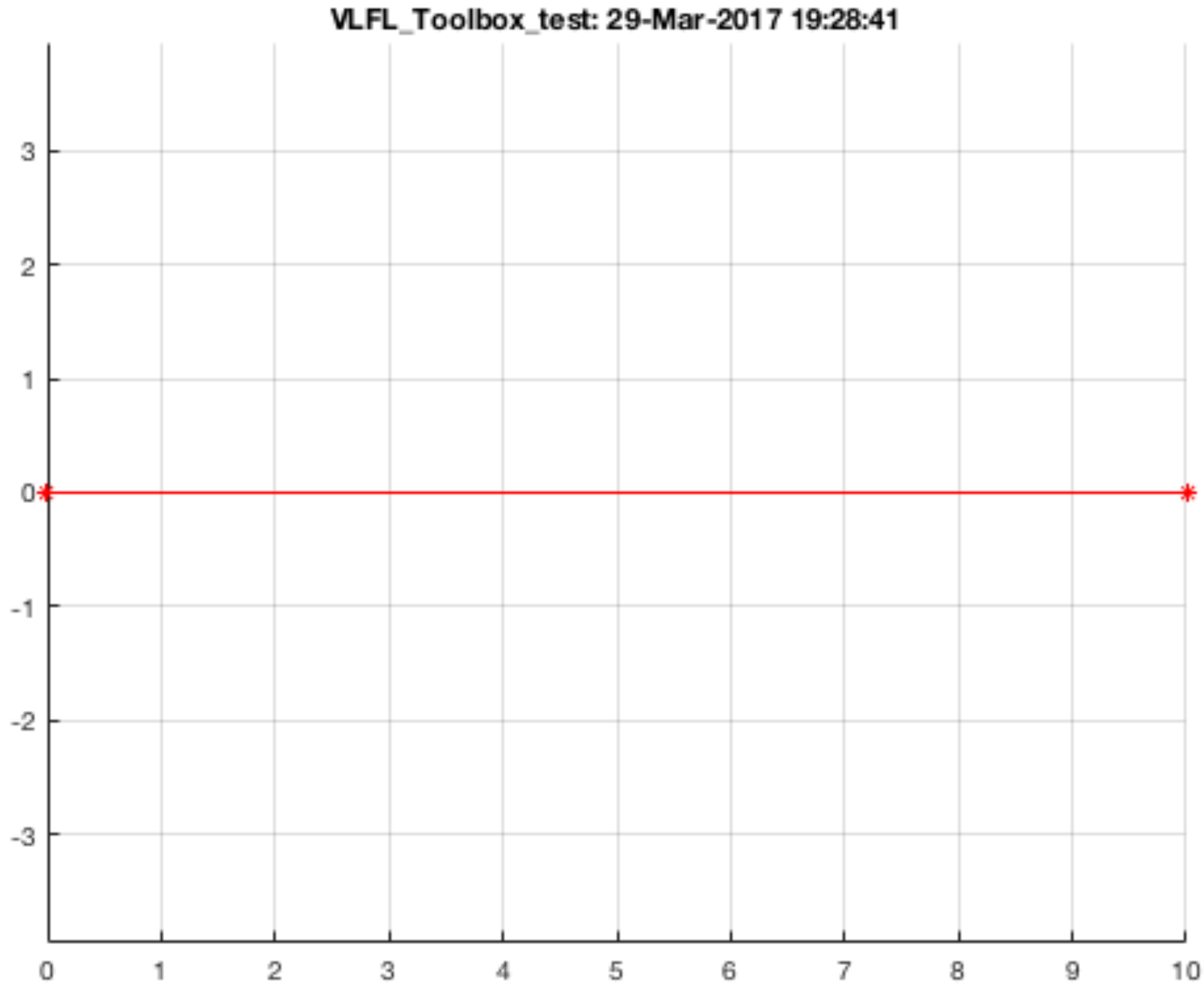

```
SGfigure; CPLfaceplot([0 0;10 0;10 10; 0 10],'g*-',1,0.5);  % Plotting closed polygon surfa
```

```
ces
```

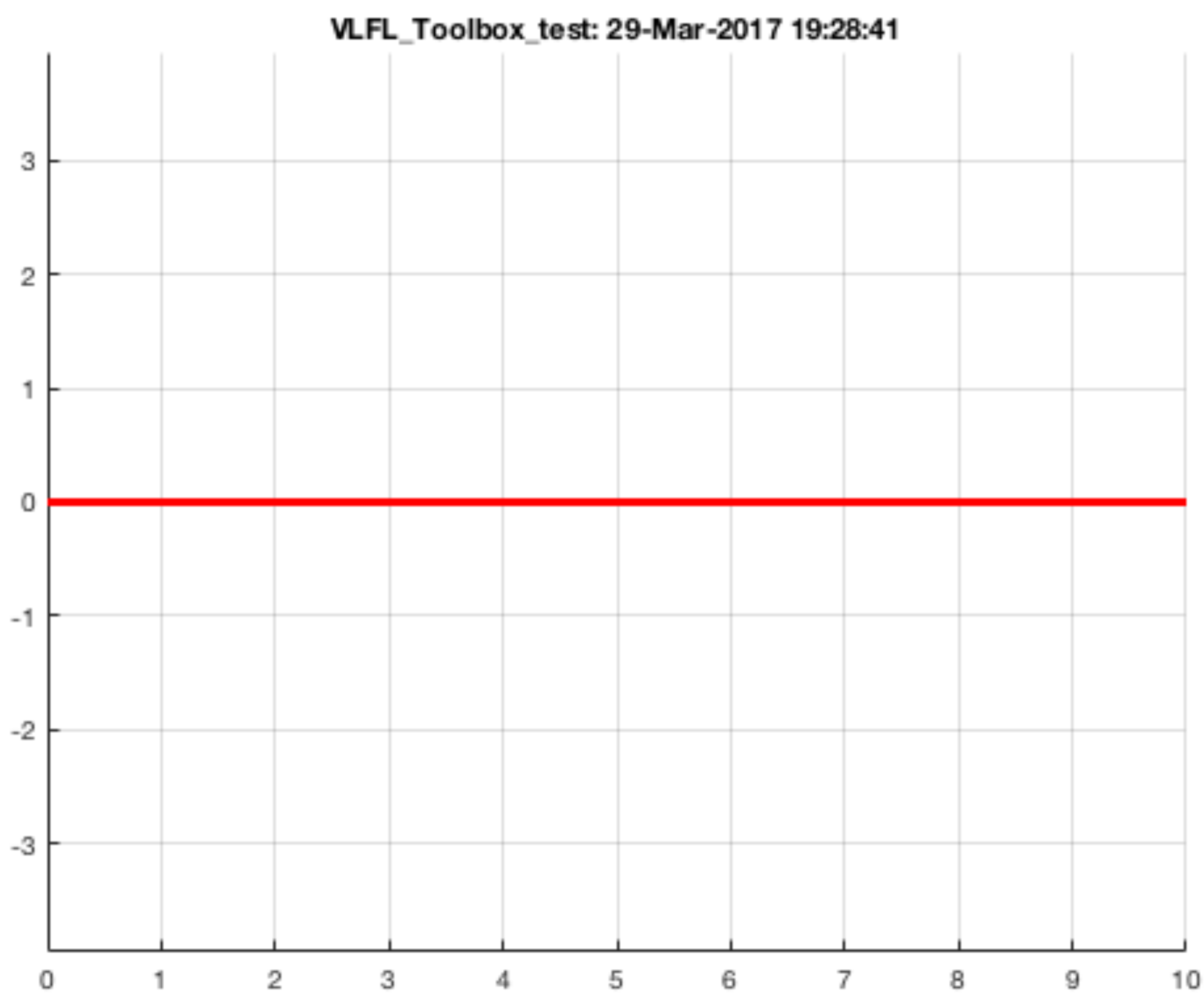


2. Plotting lines

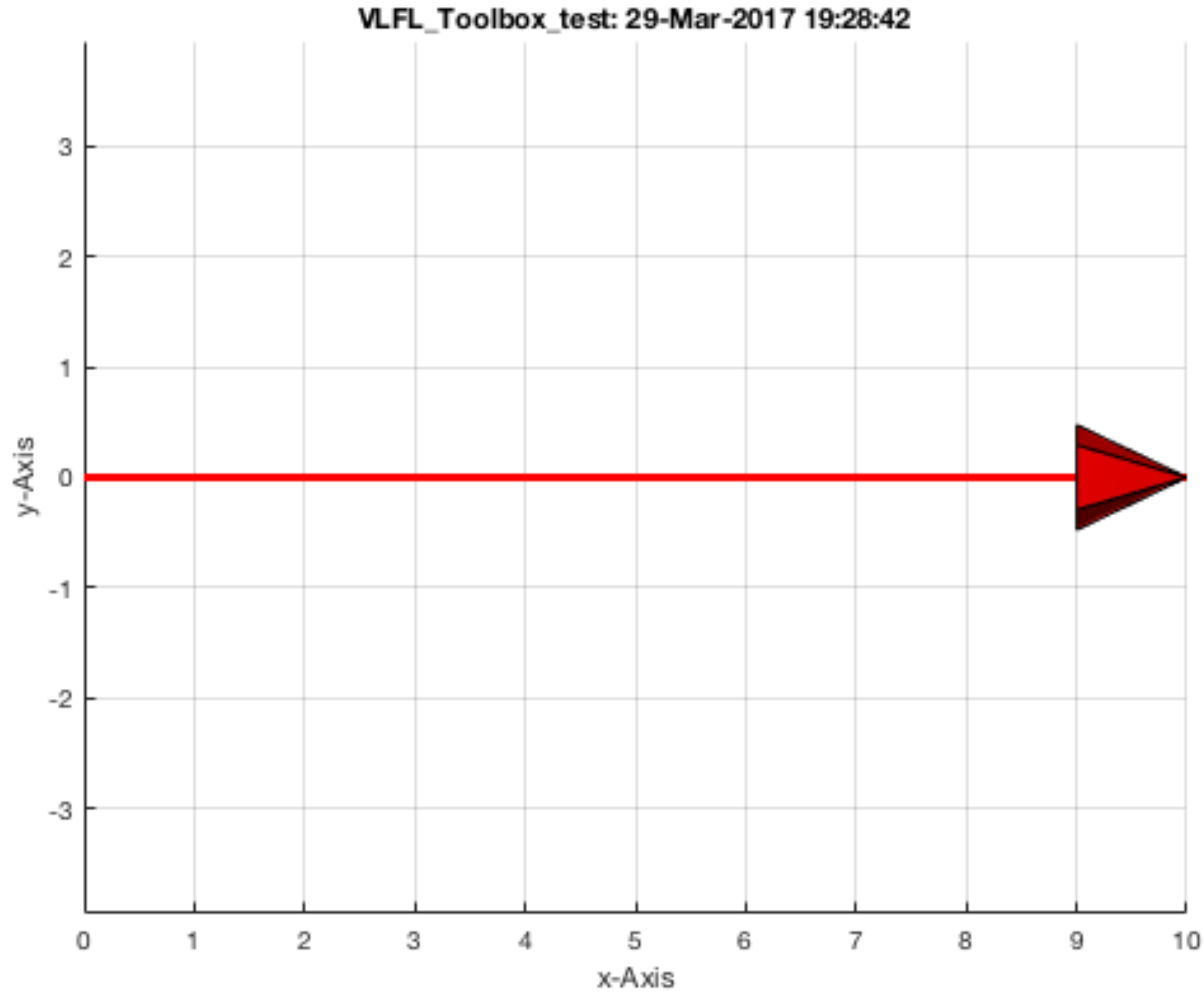
```
SGfigure; lplot([0 0],[10 0],'r*-');
```



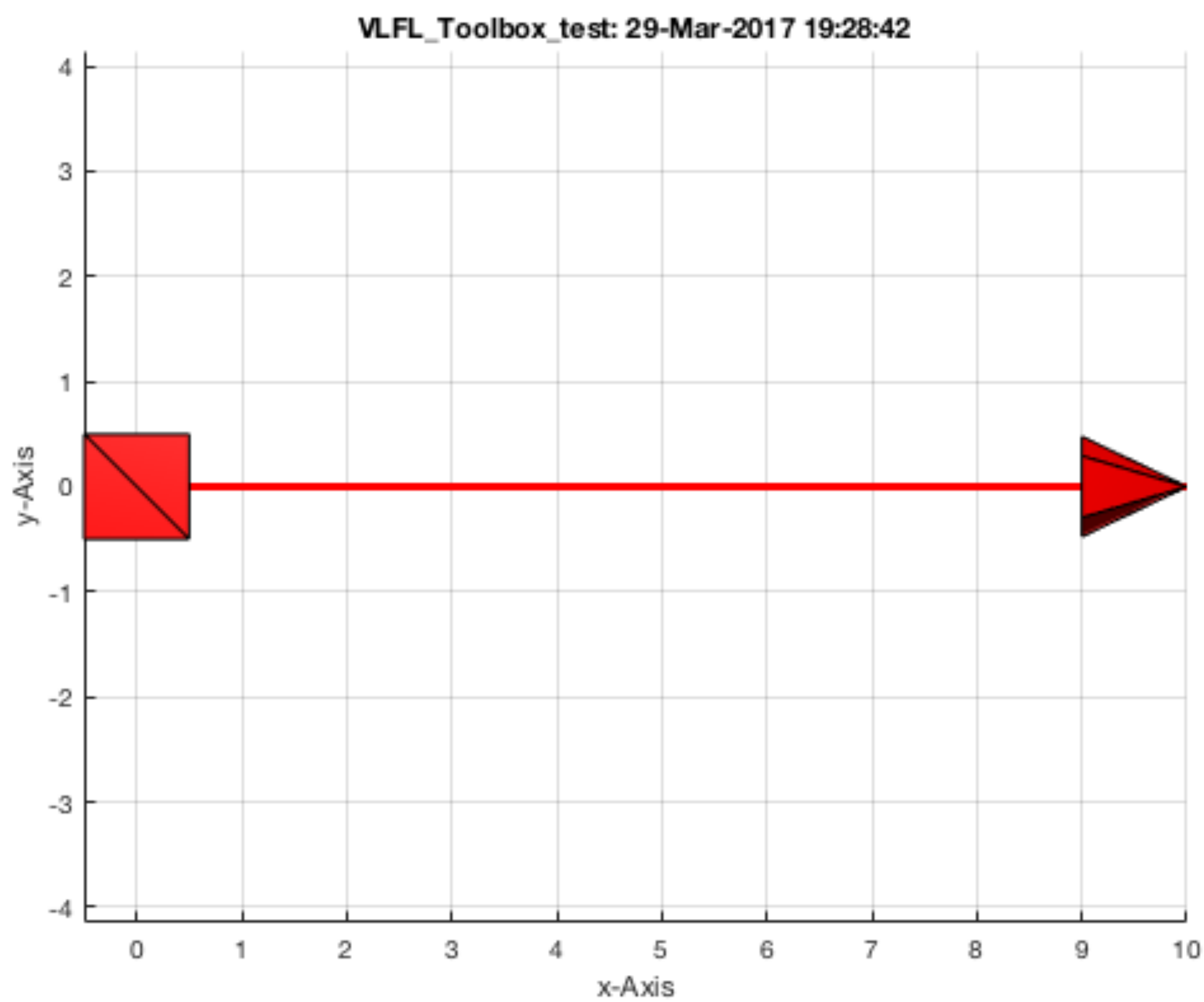

```
SGfigure; lplot([0 0],[10 0],'r-',3);
```




```
SGfigure; lplot([0 0],[10 0],'r-',3,1);
```




```
SGfigure; lplot([0 0],[10 0],'r-',3,1,1);
```




```
SGfigure; slplot([0 0],[10 0],'r-',3,1,1);
```

```
Undefined function 'getvar' for input arguments of type 'cell'.

Error in slplot

Error in VLFL_EXP30 (line 81)
SGfigure; slplot([0 0],[10 0],'r-',3,1,1);
```

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);
```

```
SGfigure; CPLsample;
```

```
SGfigure; CPLsample(5);
```

Final Remarks

```
close all
VLFLlicense
```

Published with MATLAB® R2016b

Exercise 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)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- Motivation for this tutorial
- 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 SGofVMmarchcub
- 4 Show the Voxel model in quadrant 1-2-4 and surface model in quadrant 3
- 5. Create a surface model and convert it into a Voxel model
- 6. Plot the surface model in 4 quadrant plot
- 7. Write the surface model as DICOM directory on disk
- 8. Select Point in 3D
- Final Remarks

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages

- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Exercise 30: Create Mechanisms using Universal Planar Links
- Exercise 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids

Motivation for this tutorial

- 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)

```
[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
```

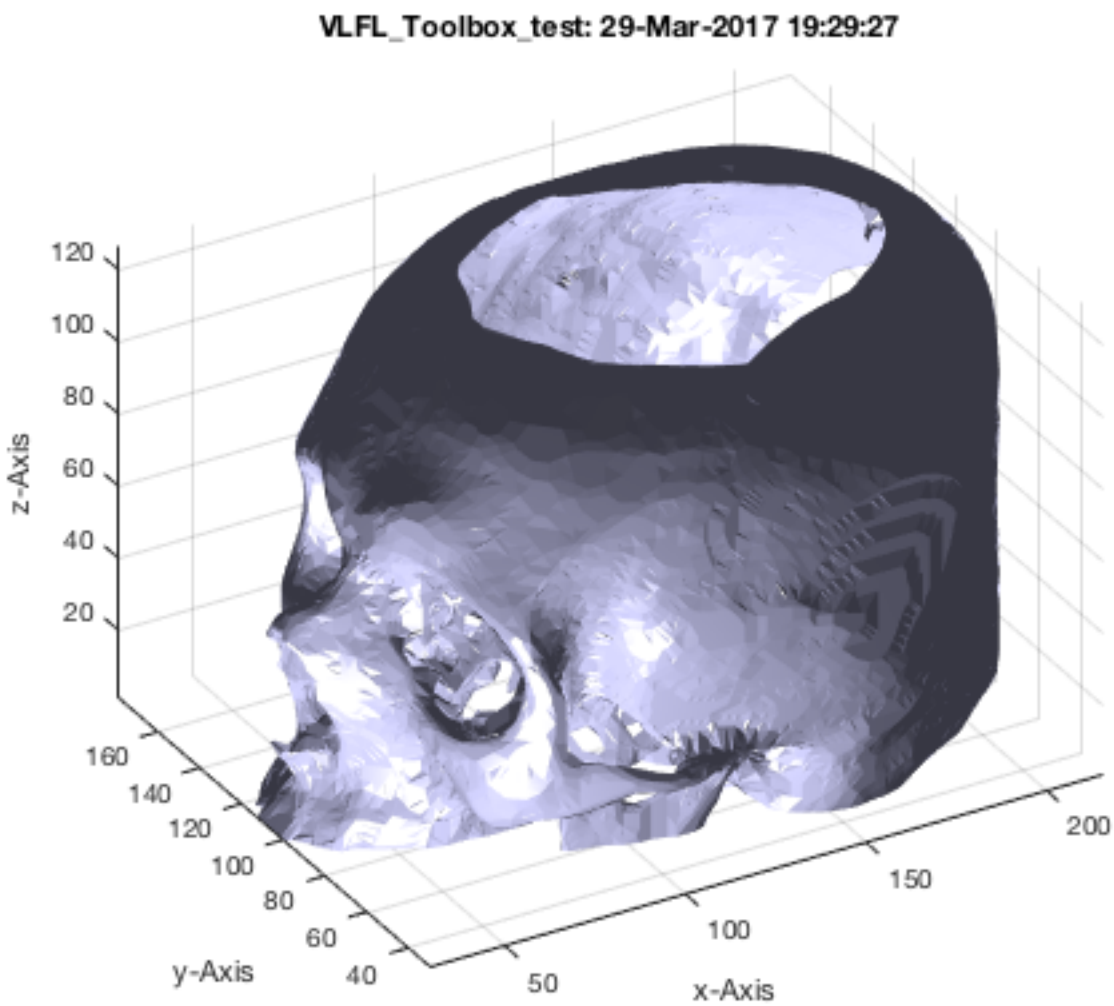
```
VMreaddicomdir: Analyzing 129 entries in path: /Volumes/LUETH-WIN/WIN AIM Matlab Libraries/
VLFL-Lib/AIM_DICOMFILES
.............................................................................................
......................................
Stack of 126 DICOM images read from disk.

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
```

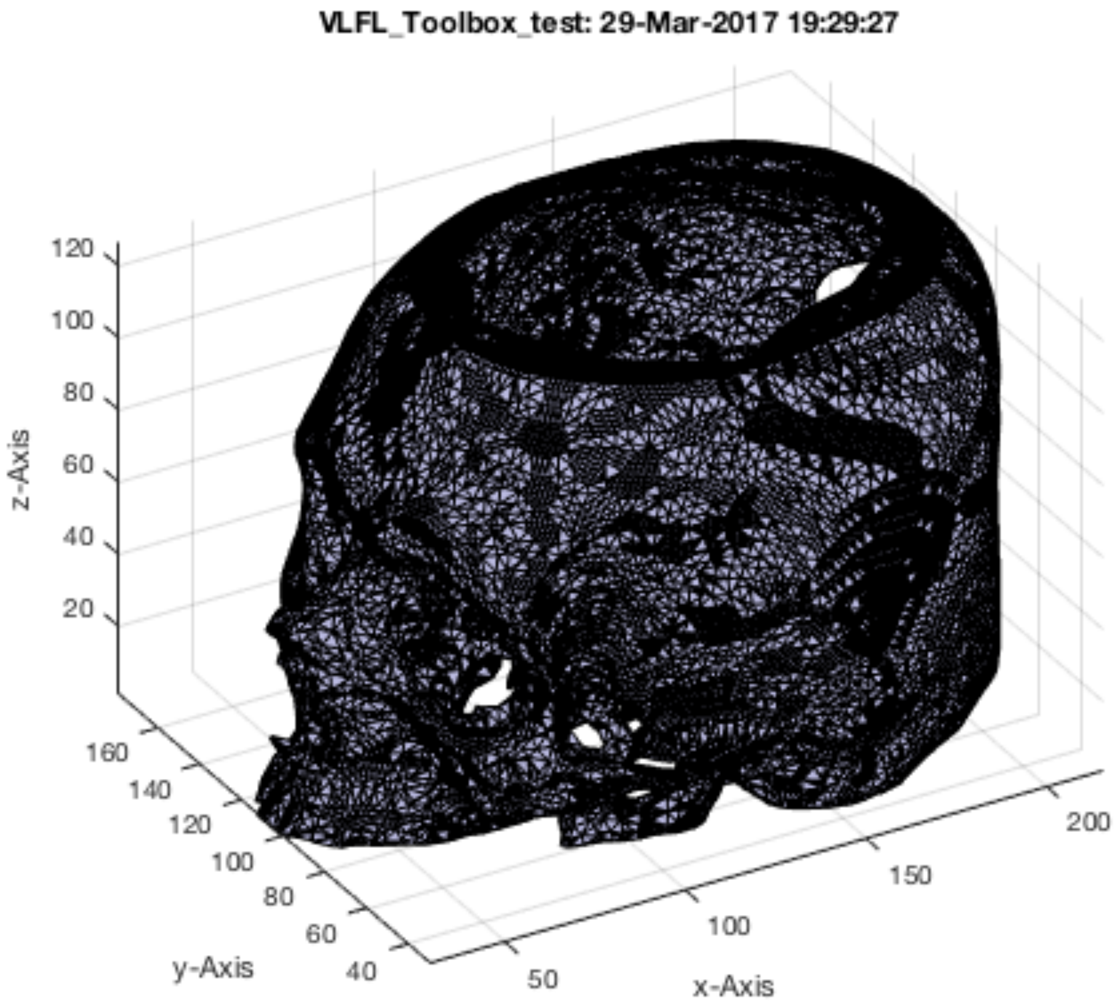
2. Solid skull bone reconstruction using SGofVMdelaunay

```
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 6.266982 seconds.
Elapsed time is 23.285550 seconds.
```



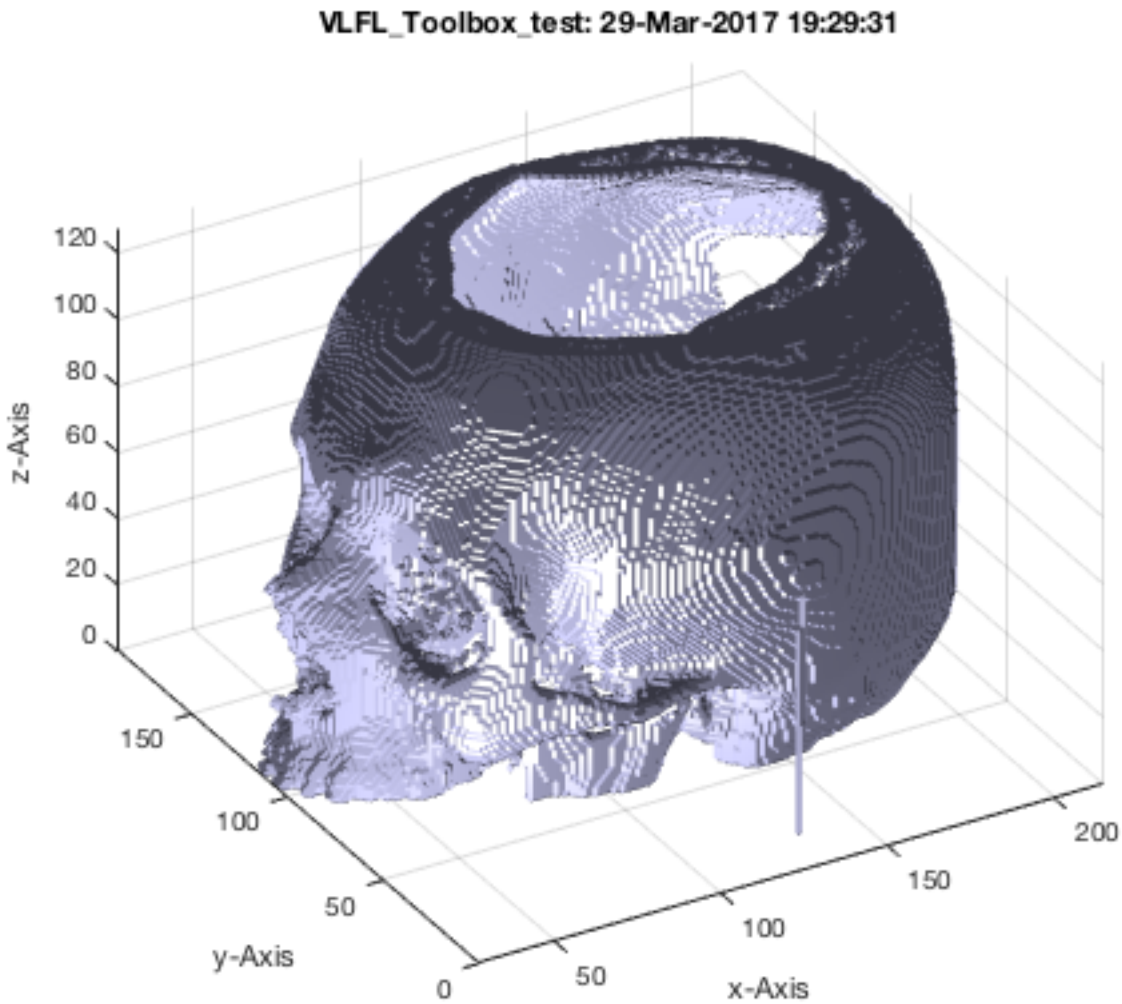

```
VLFLplotlight(0,1);
```



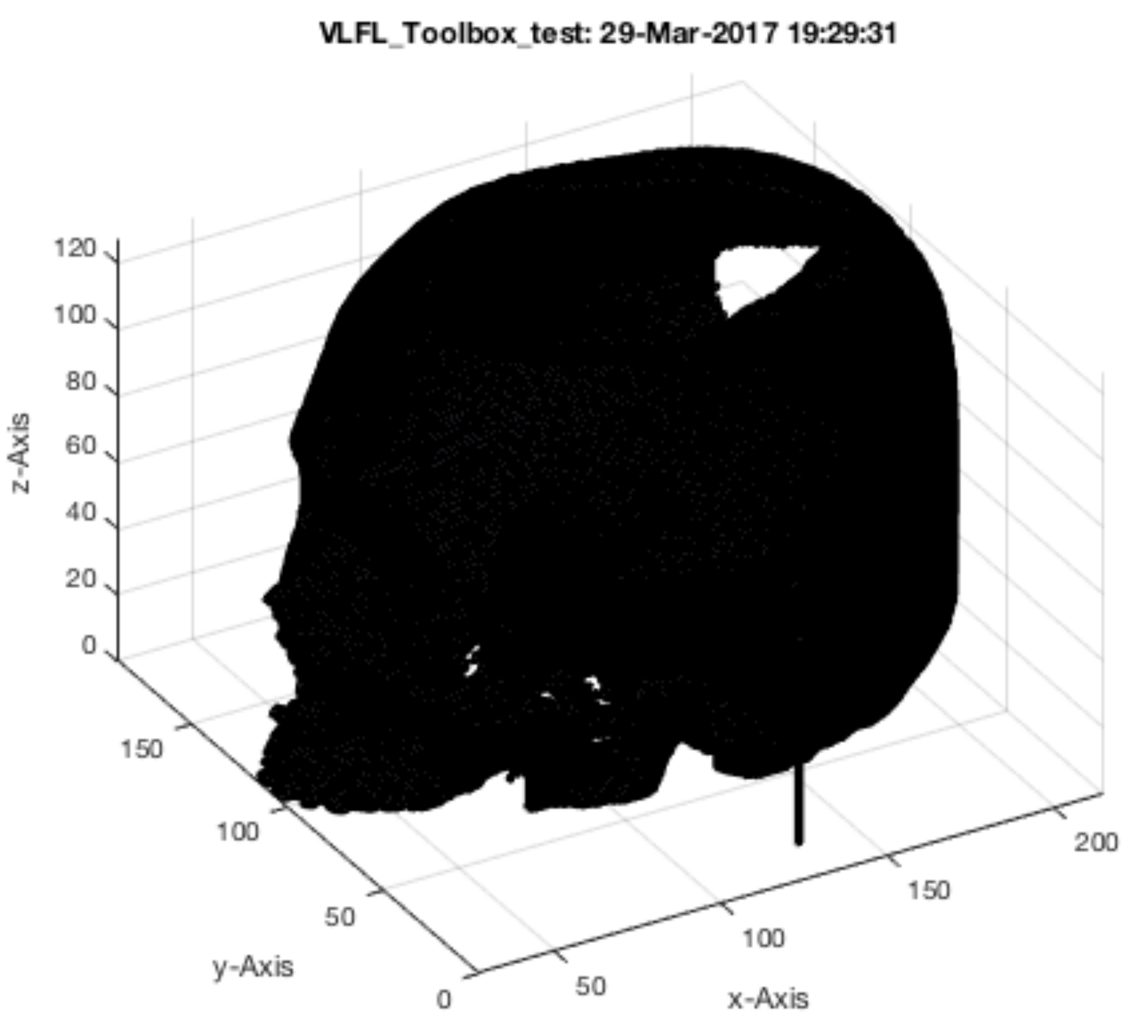
3. Solid skull bone reconstruction using SGofVMmarchcub

```
tic
SG2=SGofVMmarchcube(a>1400,as);        % Takes about 2 seconds
toc
SGfigure; VLFLplotlight (1,1); view(-30,30);
SGplot(SG2,'w');VLFLplotlight(1,1)
```

```
Elapsed time is 2.555966 seconds.
```

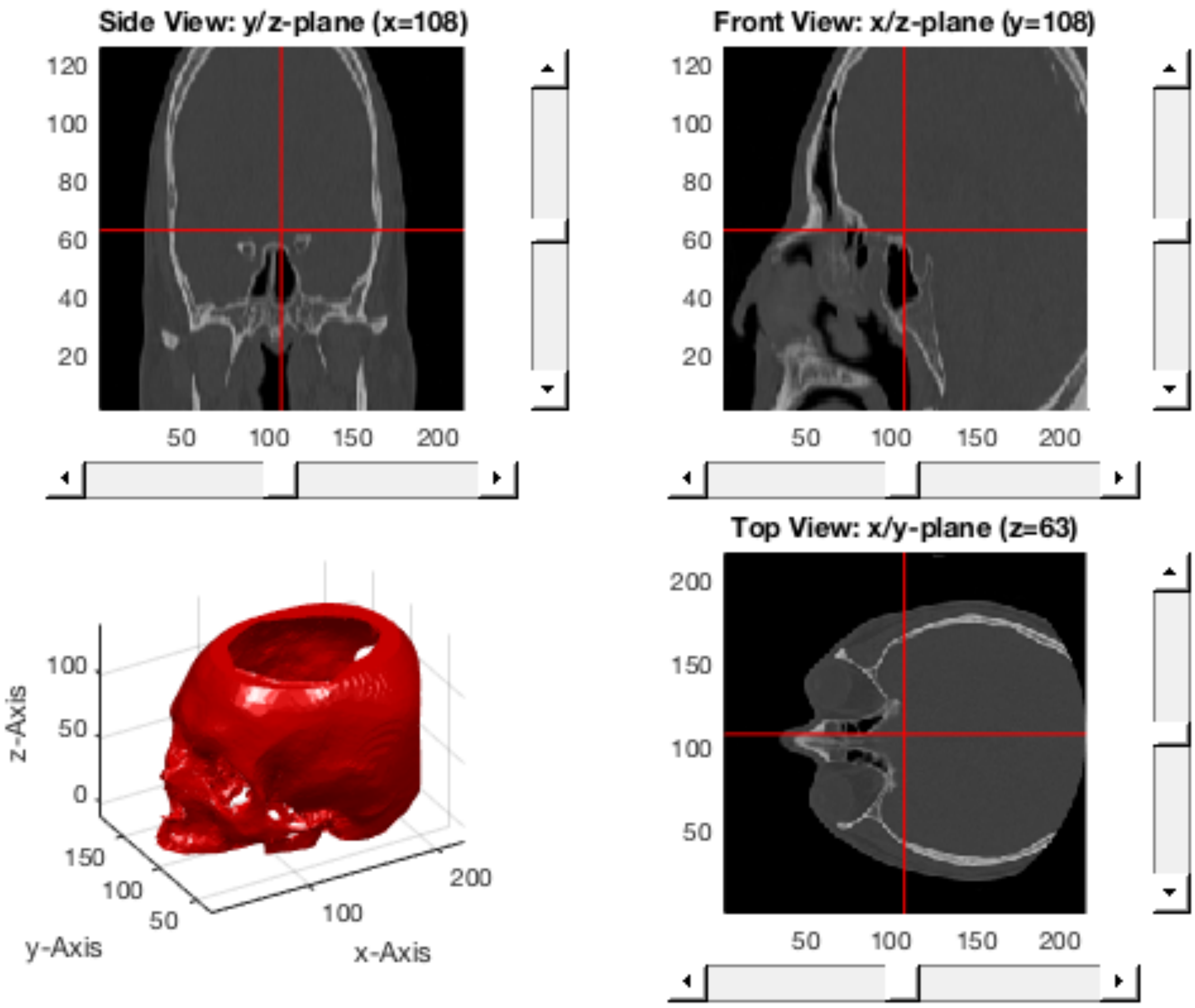



```
VLFLplotlight(0,1);
```



4 Show the Voxel model in quadrant 1-2-4 and surface model in quadrant 3

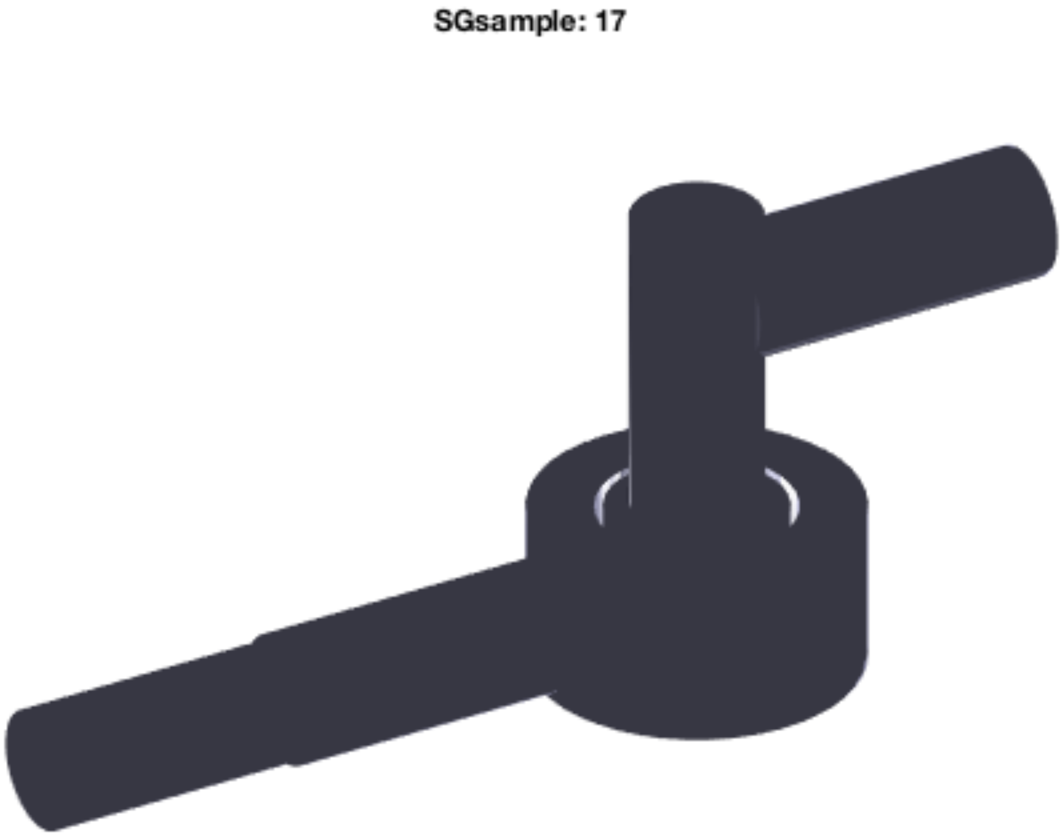
```
VMplot(a,'',SG1)
```



5. Create a surface model and convert it into a Voxel model

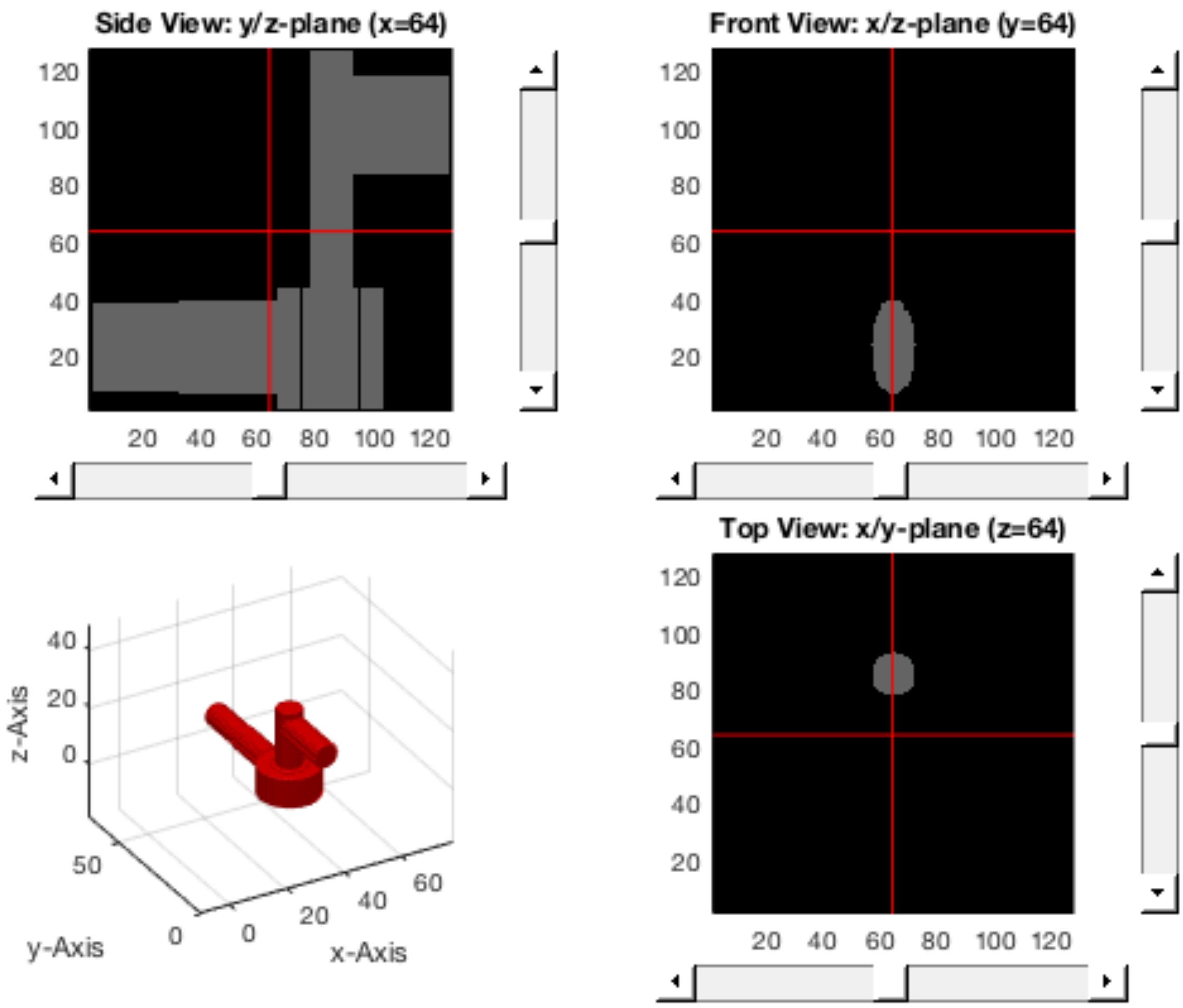
```
SGsample(17); VLFLplotlight(1,1);

[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]
```



6. Plot the surface model in 4 quadrant plot

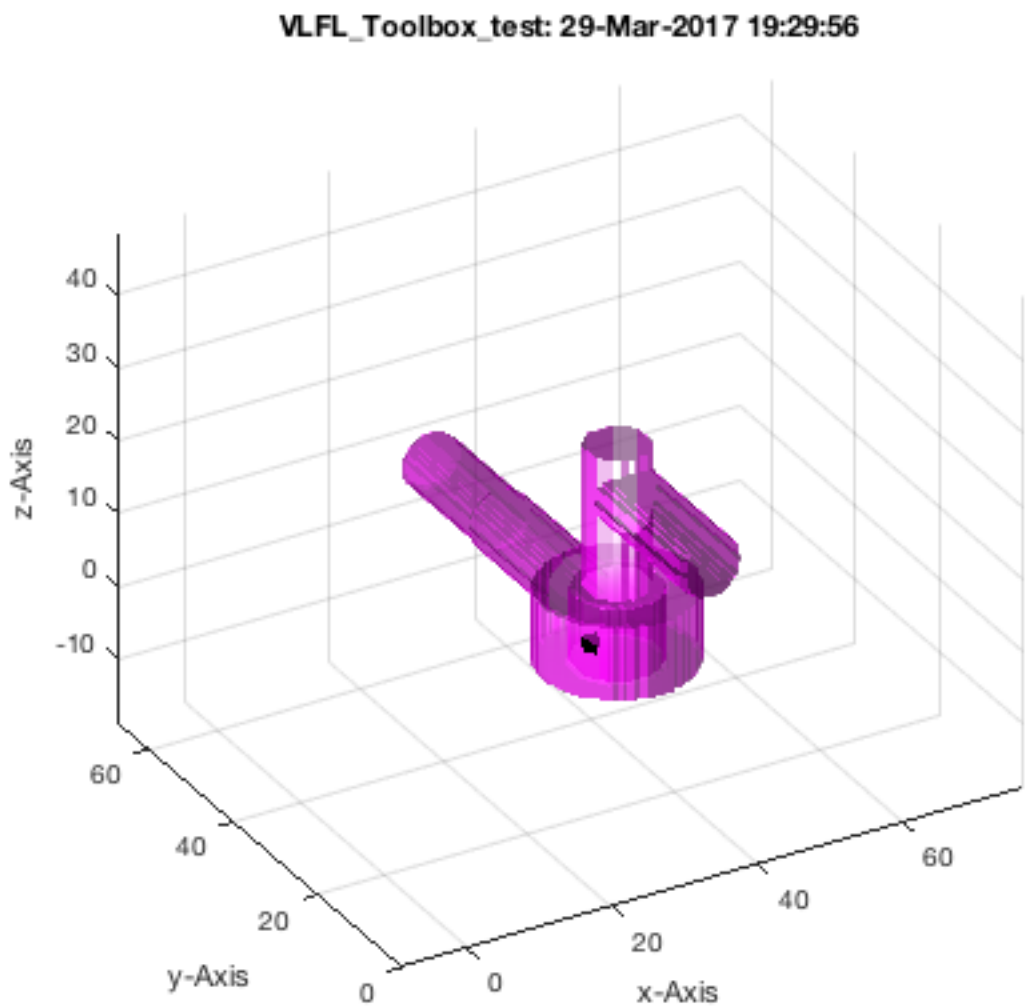
```
VMplot(VM,'',SG3)
```



7. Write the surface model as DICOM directory on disk

8. 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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:41:31 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ==============================
==========
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
=============================================================================================
==========
```

Exercise 32: Exchanging Data with a FileMaker Database

2017-03-01: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- Motivation for this tutorial
- 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

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations
- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing

- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Exercise 30: Create Mechanisms using Universal Planar Links
- Exercise 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Exercise 32: Exchanging Data with a FileMaker Database

Motivation for this tutorial

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')

  Generated by pcodeTL: 29-Mar-2017 17:12:12 using 9.1.0.441655 (R2016b)
```

2. Open a Database and establishes a connection

```
conn=FMopen('FileMakerTestBase.fmp12')
```

```
conn =

  connection with properties:

       Instance: ''
       UserName: 'Admin'
         Driver: 'com.filemaker.jdbc.Driver'
            URL: 'jdbc:filemaker://localhost/FileMakerTestBase'
    Constructor: [1×1 com.mathworks.toolbox.database.databaseConnect]
        Message: []
         Handle: [1×1 com.filemaker.jdbc3.J3Connection]
        TimeOut: 0
     AutoCommit: 'on'
           Type: 'Database Object'
```

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 Fragencode'     [1065098]    'Prüfungsfragen'
    'FragenCode'              [1065094]    'Vorlesungsinhalte'
    'Prüfungsfragen'          [1065089]    'Prüfungsfragen'
    'SAME FragenCode'         [1065099]    'Prüfungsfragen'
    'Same Thema'              [1065093]    'Prüfungsfragen'
    'Vorlesungsinhalte'       [1065091]    'Vorlesungsinhalte'

  Columns 4 through 5

    'FileMakerTestBas...'    [ 25]
    'FileMakerTestBas...'    [103]
    'FileMakerTestBas...'    [ 25]
    'FileMakerTestBas...'    [103]
    'FileMakerTestBas...'    [103]
    'FileMakerTestBas...'    [103]
    'FileMakerTestBas...'    [ 25]


ans =

  169×7 cell array

  Columns 1 through 3

    'Doppelte Themen'         'Dopplestundennummer'    'decimal'
    'Doppelte Themen'         'Themennummern'          'decimal[30]'
    'Doppelte Themen'         'Themen'                 'varchar[30]'
    'Doppelte Themen'         'REL Doppelstunde...'    'decimal'
    'Doppelte Themen'         'Anzahl der Themen'      'varchar'
    'Doppelte Themen'         'CHECK Themennummer'     'varchar'
    'Doppelte Themen'         'VAR Fragencode'         'global decimal'
    'Doppelte Themen'         'CALC Fragen zu D...'    'decimal'
    'Doppelte Themen'         'Fragenrecords'          'decimal[30]'
    'Doppelte Themen'         'REL Fragenrecord'       'decimal'
    'Doppelte Themen'         'REL Doppelstunde...'    'varchar'
    'Frage zu Fragencode'     'REC Creation Date'      'date'
    'Frage zu Fragencode'     'REC Nr'                 'decimal'
    'Frage zu Fragencode'     'REC Change Date'        'date'
    'Frage zu Fragencode'     'REC Change Time'        'time'
    'Frage zu Fragencode'     'REC Creation Time'      'time'
    'Frage zu Fragencode'     'REC Creation Author'    'varchar'
    'Frage zu Fragencode'     'REC Change Author'      'varchar'
    'Frage zu Fragencode'     'Fragen'                 'varchar[10]'
    'Frage zu Fragencode'     'Fragenskizzen'          'binary[10]'
    'Frage zu Fragencode'     'CALC Themenkomplex'     'varchar'
    'Frage zu Fragencode'     'CALC Fragennummern'     'decimal[10]'
    'Frage zu Fragencode'     'VAR i'                  'global decimal'
    'Frage zu Fragencode'     'VAR n'                  'global decimal'
    'Frage zu Fragencode'     'Doppelstunden'          'global varchar[15]'
    'Frage zu Fragencode'     'VAR j'                  'global decimal'
    'Frage zu Fragencode'     'Vorlesungstitel'        'global varchar'
    'Frage zu Fragencode'     'CALC Doppelstund...'    'global decimal[15]'
    'Frage zu Fragencode'     'Nummer der Doppe...'    'decimal'
    'Frage zu Fragencode'     'Nummer des Themas'      'decimal'
    'Frage zu Fragencode'     'REL Thema'              'varchar'
    'Frage zu Fragencode'     'CALC Vorlesungsk...'    'decimal'
    'Frage zu Fragencode'     'CALC Thema'             'decimal'
```

```
'Frage zu Fragencode'    'CALC Fragekode'          'decimal'
'Frage zu Fragencode'    'CALC Anzahl der ...'     'decimal'
'Frage zu Fragencode'    'VAR k'                   'global decimal'
'Frage zu Fragencode'    'CALC Themen und ...'     'varchar'
'Frage zu Fragencode'    'Antworten'               'varchar[10]'
'Frage zu Fragencode'    'CALC Folie Title'        'decimal'
'Frage zu Fragencode'    'CALC Folie Unter...'     'decimal'
'Frage zu Fragencode'    'CHECK Doppelt'           'varchar'
'Frage zu Fragencode'    'STAT TEST'               'decimal'
'Frage zu Fragencode'    'Fragen als Bild'         'binary[10]'
'Frage zu Fragencode'    'Uebungstext'             'varchar'
'Frage zu Fragencode'    'Einfache Beschri...'     'binary'
'FragenCode'             'Dopplestundennummer'     'decimal'
'FragenCode'             'Themennummern'           'decimal[30]'
'FragenCode'             'Themen'                  'varchar[30]'
'FragenCode'             'REL Doppelstunde...'     'decimal'
'FragenCode'             'Anzahl der Themen'       'varchar'
'FragenCode'             'CHECK Themennummer'      'varchar'
'FragenCode'             'VAR Fragencode'          'global decimal'
'FragenCode'             'CALC Fragen zu D...'     'decimal'
'FragenCode'             'Fragenrecords'           'decimal[30]'
'FragenCode'             'REL Fragenrecord'        'decimal'
'FragenCode'             'REL Doppelstunde...'     '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 Author'     'varchar'
'Prüfungsfragen'         'REC Change Author'       'varchar'
'Prüfungsfragen'         'Fragen'                  'varchar[10]'
'Prüfungsfragen'         'Fragenskizzen'           'binary[10]'
'Prüfungsfragen'         'CALC Themenkomplex'      'varchar'
'Prüfungsfragen'         'CALC Fragennummern'      'decimal[10]'
'Prüfungsfragen'         'VAR i'                   'global decimal'
'Prüfungsfragen'         'VAR n'                   'global decimal'
'Prüfungsfragen'         'Doppelstunden'           'global varchar[15]'
'Prüfungsfragen'         'VAR j'                   'global decimal'
'Prüfungsfragen'         'Vorlesungstitel'         'global varchar'
'Prüfungsfragen'         'CALC Doppelstund...'     'global decimal[15]'
'Prüfungsfragen'         'Nummer der Doppe...'     'decimal'
'Prüfungsfragen'         'Nummer des Themas'       'decimal'
'Prüfungsfragen'         'REL Thema'               'varchar'
'Prüfungsfragen'         'CALC Vorlesungsk...'     '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 Unter...'     'decimal'
'Prüfungsfragen'         'CHECK Doppelt'           'varchar'
'Prüfungsfragen'         'STAT TEST'               'decimal'
'Prüfungsfragen'         'Fragen als Bild'         'binary[10]'
'Prüfungsfragen'         'Uebungstext'             'varchar'
'Prüfungsfragen'         'Einfache Beschri...'     '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 Author'     'varchar'
'SAME FragenCode'         'REC Change Author'       'varchar'
'SAME FragenCode'         'Fragen'                  'varchar[10]'
'SAME FragenCode'         'Fragenskizzen'           'binary[10]'
'SAME FragenCode'         'CALC Themenkomplex'      'varchar'
'SAME FragenCode'         'CALC Fragennummern'      'decimal[10]'
'SAME FragenCode'         'VAR i'                   'global decimal'
'SAME FragenCode'         'VAR n'                   'global decimal'
'SAME FragenCode'         'Doppelstunden'           'global varchar[15]'
'SAME FragenCode'         'VAR j'                   'global decimal'
'SAME FragenCode'         'Vorlesungstitel'         'global varchar'
'SAME FragenCode'         'CALC Doppelstund...'     'global decimal[15]'
'SAME FragenCode'         'Nummer der Doppe...'     'decimal'
'SAME FragenCode'         'Nummer des Themas'       'decimal'
'SAME FragenCode'         'REL Thema'               'varchar'
'SAME FragenCode'         'CALC Vorlesungsk...'     '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 Unter...'     '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 Beschri...'     '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 Author'     'varchar'
'Same Thema'              'REC Change Author'       'varchar'
'Same Thema'              'Fragen'                  'varchar[10]'
'Same Thema'              'Fragenskizzen'           'binary[10]'
'Same Thema'              'CALC Themenkomplex'      'varchar'
'Same Thema'              'CALC Fragennummern'      'decimal[10]'
'Same Thema'              'VAR i'                   'global decimal'
'Same Thema'              'VAR n'                   'global decimal'
'Same Thema'              'Doppelstunden'           'global varchar[15]'
'Same Thema'              'VAR j'                   'global decimal'
'Same Thema'              'Vorlesungstitel'         'global varchar'
'Same Thema'              'CALC Doppelstund...'     'global decimal[15]'
'Same Thema'              'Nummer der Doppe...'     'decimal'
'Same Thema'              'Nummer des Themas'       'decimal'
'Same Thema'              'REL Thema'               'varchar'
'Same Thema'              'CALC Vorlesungsk...'     '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 Unter...'     '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 Beschri...'       'binary'
    'Vorlesungsinhalte'         'Dopplestundennummer'       'decimal'
    'Vorlesungsinhalte'         'Themennummern'             'decimal[30]'
    'Vorlesungsinhalte'         'Themen'                    'varchar[30]'
    'Vorlesungsinhalte'         'REL Doppelstunde...'       'decimal'
    'Vorlesungsinhalte'         'Anzahl der Themen'         'varchar'
    'Vorlesungsinhalte'         'CHECK Themennummer'        'varchar'
    'Vorlesungsinhalte'         'VAR Fragencode'            'global decimal'
    'Vorlesungsinhalte'         'CALC Fragen zu D...'       'decimal'
    'Vorlesungsinhalte'         'Fragenrecords'             'decimal[30]'
    'Vorlesungsinhalte'         'REL Fragenrecord'          'decimal'
    'Vorlesungsinhalte'         'REL Doppelstunde...'       '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 ingenieur...'
    'Beschreiben Sie die unterschiedlichen Aufgaben der Biologie, Pharmazie...'
    'Nennen Sie mehrere Lehrberufe und akademische Berufe in der Zahnheilku...'
    'Welches sind die drei Säulen eines Klinikums und wie heißen die Leitun...'
    'Welche Aufgabe hat ein Wellness-Hotel oder Kurhotel? Wer besucht diese...'
    '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 sei...'
    '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 um...'
    '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 u...'
    '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 t...'
    'Welches sind die einzelnen Schritte von einer Patentanmeldung bis hin ...'
    'Gibt es Richtlinien, wie gute wissenschaftliche Praxis aussieht und we...'
    'Was geschieht bei der Registrierung von 2 Koordinatensystemen bzw. was...'
    'Beschreiben Sie das RMS-Maß für die Genauigkeitsangabe einer Registrie...'
    'Welche Effekte führen zu einem Positionsmessfehler bei der Verwendung ...'
    'Welches ist die Aufgabe bei der Registrierung von Patient und Bilddate...'
    'Nennen Sie 6 Regeln für die Auswahl von Landmarken am Patienten/Messob...'
    'Nennen Sie das Gesetz mit Nummer, dass die Anwendung spezieller  Norme...'
    '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 Röntgenstrahl...'
    '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-B...'
'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 si...'
'Welche Aufgabe hat das Betriebssystem Windows XP?'
'Welche Aufgaben hat ein Echtzeitbetriebssystem und können dort mehrere...'
'Wie werden Eingabegeräte angeschlossen. Nennen Sie Bussysteme (je 2)'
'Wie können unterschiedliche Programme/Prozesse/Tasks untereinander Dat...'
'Unterstützt XP den Zugriff auf Datenträger?'
'Wie erzeugt man eine feste Taktfrequenz für das Aufrufen/Abarbeiten vo...'
'Wie ist ein Bildspeicher organisiert und wie kommen die Bilddaten auf ...'
'Welche Formen der 3D-Darstellung kennen Sie?'
'Zeichnen Sie in ein schematisches Volumenbild V(u,v,w) die Schnittbild...'
'Zeichnen Sie ein schematisches Bild, wie die Tiefeninformation aus dem...'
'Zeichnen Sie schematisch das Konzept der Mischung von 3D-Bildern mit S...'
'Beschreiben Sie eine Koordinaten-Transformation mit der DH-Matrix. Was...'
'Geben Sie die drei Einheitsvektoren des kartesischen Koordinatensystem...'
'Geben Sie formal die Translation von Punkt p1 nach Punkt p2 an. Geben ...'
'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 n...'
'Wie lautet die Inverse einer homogenen Transformationsmatrix?'
'Wie werden homogene Transformationsmatrizen verkettet?'
'Wieviele Möglichkeiten der Winkeldarstellung einer Rotationsmatrix sin...'
'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 K...'
'Was ist ein stereotaktischer Rahmen und wie wird dieser benutzt?'
'Zeichnen Sie die minimalen Komponenten, die für ein Navigationssystem ...'
'Wozu dient die Kamera und wie häufig misst diese die Position der Mess...'
'Wie funktionieren aktive Messmarken und Zeilenkameras? Wie viele Zeile...'
'Welche Bildschirme sind bei klinischen Navigationssystemen bekannt? Ne...'
'Geben Sie die 6 Schritte der bildgestützten Navigation an.'
'Was bedeutet Echtzeit-Navigation und worin unterscheidet Sie sich grun...'
'Aus welchen 7 Schritten besteht die planungsbasierte Navigation? Worin...'
'Bei der Planung von dentalen Implantaten sollen zwei Kriterien erfüllt...'
'Was ist ein Navigationsbogen und welche drei Aufgaben erfüllt dieser B...'
'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 Roll...'
'Was geschieht bei der Kalibration des Instruments?'
'Auf welche Weise erfolgt die Patientenregistrierung mit der Bissschien...'
```

```
'Was genau sieht man am Fadenkreuz und warum ist die Anzeige so wichtig?'
'Geben Sie die homogene Transformationsmatrix für die Pose des Patiente...'
'Welches elektrisches Interface benutzt eine IR-Stereomesskamera? Welch...'
'Zeichnen Sie eine Kamera und einen Messmarkentracker mit 3 Punkten!'
'Zeichnen Sie eine Tabelle zur Definition der Geometrien von Messmarken...'
'Welche 4 Formen der Befestigung von Patiententracker am Patienten gibt...'
'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 Verwen...'
'Unterscheiden Sie bitte qualitative Aussagen von quantitativen Aussage...'
'Was bedeuten die Begriffe: Eigenschaft, Messgröße, Wert und Einheit?'
'Können Sie aus einer neuen Innovativen Idee die daraus vermutlich ents...'
'Wie sieht der zeitliche Zusammenhang aus zwischen der „Erwartung von V...'
'Warum spielt der Zusammenhang zwischen einem Messwert und einer daraus...'
'Dürfen Sie ein Gerät bauen, damit Messwerte aufnehmen, daraus eine For...'
'Nennen Sie ein Beispiel für ein Hypothese, die einen Gerätevorteil bes...'
'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 ...'
'Welche Vorteile hat es, alle Messungen grundsätzlich mit einem Compute...'
'Wozu dient der f-Test?'
'Was ist der t-Test?'
'Welches sind die 5 Verfahren für systematische Belegung von Erkenntnis...'
'Welche drei Aussagen können bei Evidence-Based-Research getroffen werden?'
'Welche Einrichtung überwacht in Deutschland die Wirksamkeit neuer Medi...'
'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 zugäng...'
'Beschreiben Sie das V-Modell in Entwicklung und Dokumentation? '
'Welche Dokumente müssen Sie mindestens besitzen, wenn Sie ein neues Ge...'
'Wozu dient eine Funktionsbeschreibung und wer soll diese in erster Lin...'
'Wozu dient die Systembeschreibung und wer soll diese in erster Linie v...'
'Was ist eine Bauteileliste?'
'Was ist eine Gebrauchsanweisung und welche 2 wichtigen Dinge werden du...'
'Was ist ein Montagehandbuch und welche 2 Dinge werden durch dieses def...'
'Was ist eine Risikoanalyse warum wird dies durchgeführt? Wie lautet di...'
'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 Medizinprod...'
'Was ist die CE-Kennzeichnung und müssen Medizinprodukte diese haben'
'Beschreiben Sie in kurzen Worten, wie man ein Medizingerät zulässt. Ge...'
'Was ist der Unterschied zwischen dem Zulassungsverfahren in Europa und...'
'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....'
'Welche Keramiken kommen zum Einsatz. nennen Sie 3 Keramiken für 3 unte...'
'Welche Metalle kommen in der Instrumententechnik zum Einsatz. Nennen S...'
'Nennen Sie 7 Gruppen von Kunststoffen, die in der Medizintechnik zum E...'
```

```
'Was sind biokompatible Materialien?'
'Was sind sterilisierbare Materialien?'
'Welche Materialien sind implantierbar?'
'Was sind bioabsorbierbare Materialien? Welche Einheiten und Werte kenn...'
'Welche Materialien werden als Nahtmaterial eingesetzt?'
'Was sind die Anforderungen an Gehäuse, die dauerhaft im Körper verblei...'
'Nennen Sie 3 typische Materialien für Schläuche und Katheter, die im K...'
'Nennen Sie je 2 Kunststoffe mit Eigenschaften, Werten und Einheiten di...'
'Welche Kunststoffe werden als Steriler Überzug eingesetzt und welche 3...'
'Welche Materialien werden zur Züchtung von Zellen eingesetzt? Nennen S...'
'Welche Oberflächenbeschichtungsverfahren werden eingesetzt zur Erzielu...'
'Nennen Sie 3 Werkstoffe zur Herstellung autoklavierbarer Instrumente.'
'Nennen Sie 5 Materialien, die in Zahnfüllen und als Zahnimplantate zum...'
'Aus welchem Material sind Trokkare (jeweils ein Beispiel für Metall un...'
'Aus welchem Material sind Stents (Material, Eigenschaft, Wert, Einheit)?'
'Welche Elemente werden für Seeds verwendet und welche Halbwertszeiten ...'
'Nennen Sie 3 Materialien aus denen Grafts hergestellt werden mit ihren...'
'Nennen Sie 3 Knochenersatzmaterialien (mit EWE). Je eines für die Anwe...'
'Nennen Sie 3 Knochenzementsorten mit ihre Eigenschaften, Werten und Ei...'
'Welche Materialzertikate benötigen Instrumentenwerkstoffen, Hautkontak...'
'Welche Magnete kommen in MRT Systemen sowie in den Sterotaxis-Systemen...'
'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'
'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'
'null'
'Welches sind die typischen Antriebsformen von Bohrmaschinen im OP-Saal...'
'Welche Drehzahl und welches Drehmoment erreichen elektrische Fräser in...'
'Was ist der Unterschied zwischen einem Koagulator und einem HF-Messer ...'
'Was ist Lithotripsie und in welchen klinischen Anwendungen kommt es zu...'
'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'
'null'
'null'
'null'
'null'
'null'
'null'
'null'
'null'
'null'
'null'
```

```
    'Was ist eine Landmarkenkupplung? Welche zwei Komponenten werden dabei ...'
    'Geben Sie an, wie Sie aus einem Volumenbild I (x,y,z) eine orthogonale...'
    'Geben Sie an, wie Sie aus einem Volumenbild I (x,y,z) eine orthogonale...'
    'Welche Koordinatensysteme sind für die Kalibrierung von Instrumenten e...'
    'Welche Komponenten sind zur Registrierung starrer Instrumente über ein...'
    'Welche Komponenten sind zur Kalibration der Pose  starrer Instrumente ...'
    'Welche Komponenten sind zur Registrierung starrer Werkzeugspitzen oder...'
    'Welche zwei Möglichkeiten bestehen zur Realisierung einer Instrumenten...'
    'Welches Risiko und welche Gefahr besteht grundsätzlich bei der Registr...'
    'Welches Risiko und welche Gefahr besteht grundsätzlich bei der Registr...'
    'Wie viele Menschen leben gegenwärtig in Deutschland? Wie viele sind da...'
    '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 ...'
    'Was bedeuten die Begriffe Diagnose, Anamnese, Befundung und von wem we...'
    'Was macht eine künstliche Niere? Wie groß ist dieses Gerät? Wieviele M...'
    'Welche Aufgaben hat das Gebiet des Clinical Engineering in der Medizin...'
    'Nennen Sie bildgebende Systeme in der Medizin und gegen Sie an, wievie...'
    'Wie kommunizieren bisher Geräte in der Medizintechnik? '
    'Nennen Sie die häufigsten Datenaustauschformate in Kliniken? Welche Ge...'
    '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 Inten...'
    'Geben Sie eine formale Schreibweise für ein Voxelbild mit nx=512, ny=5...'
    '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. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 20:02:48 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ===============================
==========
database_toolbox
image_toolbox
map_toolbox
```

```
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
==============================================================================================
==========
```

Published with MATLAB® R2016b

Exercise 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)

Contents

- Summary of the already explained aspects of the VLFL-Toolbox
- Motivation for this tutorial
- 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

Summary of the already explained aspects of the VLFL-Toolbox

The previous examples of this series covered the following topics:

- Example 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Example 02: Using the VLFL-Toolbox for STL-File Export and Import
- Example 03: Closed 2D Contours and Boolean Operations in 2D
- Example 04: 2.5D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Example 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Example 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Example 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Example 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Example 09: Boolean Operations with Solid Geometries
- Example 10: Packaging of Sets of Solid Geometries (SG)
- Example 11: Attaching Coordinates Frames to Create Kinematik Models
- Example 12: Define Robot Kinematics and Detect Collisions
- Example 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Example 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Example 15: Create a Solid by 2 Closed Polygons
- Example 16: Create a Solid along a contour
- Example 20: Programmaticaly Interface to SimMechanics Multi-Body Toolbox
- Example 21: Programmaticaly Design of SimMechanisms Mechanisms and Simulink Signals
- Example 22: Creation of fixed nodes for the assembly of joints
- Example 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Example 24: Automatic Creation of a Joint Limitations

- Example 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Example 26: Create Mechanisms using Universal Planar Links
- Example 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Example 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Exercise 30: Create Mechanisms using Universal Planar Links
- Exercise 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Exercise 32: Exchanging Data with a FileMaker Database
- Exercise 33: Using a Round-Robin realtime multi-tasking system

Motivation for this tutorial

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 29-Mar-2017 19:56:10 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========>
> END =======
RRrun>>
TERMINATED by User
================LOOP ENDS 29-Mar-2017 19:56:10 USING 0.10 SECONDS =========================
```

```
RRrun 'RRstop=RRcputime+1;'  % Quit after 1 second
```

```
RRkeyboardLine =

RRstop=RRcputime+1;

====LOOP STARTS 29-Mar-2017 19:56:11 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========>
> END =======
RRrun>>
================LOOP ENDS 29-Mar-2017 19:56:12 USING 1.20 SECONDS =========================
```

```
RRrun 'LIST \r QUIT' % show the tasks and quit
```

```
RRkeyboardLine =

LIST  QUIT

====LOOP STARTS 29-Mar-2017 19:56:12 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 29-Mar-2017 19:56:12 USING 0.20 SECONDS =========================
```

```
RRrun 'whos \r QUIT' % show the variables and quit
```

```
RRkeyboardLine =
```

```
whos  QUIT

====LOOP STARTS 29-Mar-2017 19:56:13 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 29-Mar-2017 19:56:13 USING 0.20 SECONDS =========================
```

```
RRrun ('fprintf(''%.2f\n'',RRcputime) \rQUIT') % show the cputime and quit
```

```
RRkeyboardLine =

fprintf('%.2f\n',RRcputime) QUIT

====LOOP STARTS 29-Mar-2017 19:56:13 for 600 SECONDS with CYCLETIME 0.100 SECONDS===========>
> END =======
RRrun>>876.15


TERMINATED by User
================LOOP ENDS 29-Mar-2017 19:56:13 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 cyc
le and quit after 1 second
```

```
RRkeyboardLine =

ADD fprintf('%.2f\n',RRcputime)  RRstop=RRcputime+1;

====LOOP STARTS 29-Mar-2017 19:56:14 for 600 SECONDS with CYCLETIME 0.100 SECONDS===========>
> END =======
```

```
RRrun>>876.74

876.84
876.94
877.04
877.14
877.24
877.34
877.44
877.54
877.64
877.74
877.84
================LOOP ENDS 29-Mar-2017 19:56:15 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 29-Mar-2017 19:56:15 for 600 SECONDS with CYCLETIME 0.100 SECONDS===========>
> END =======
RRrun>>878.36

RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {' fprintf('%.2f\n',RRcputime)'  [Inf]  [1]}


878.41

878.46
878.51
878.56
878.61
878.66
878.71
878.76
878.81
878.86
878.91
878.96
879.01
879.06
879.11
879.16
879.21
879.26
879.31
879.36
```

```
879.41
879.46
================LOOP ENDS 29-Mar-2017 19:56:17 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=RRcputim
e+1;'
```

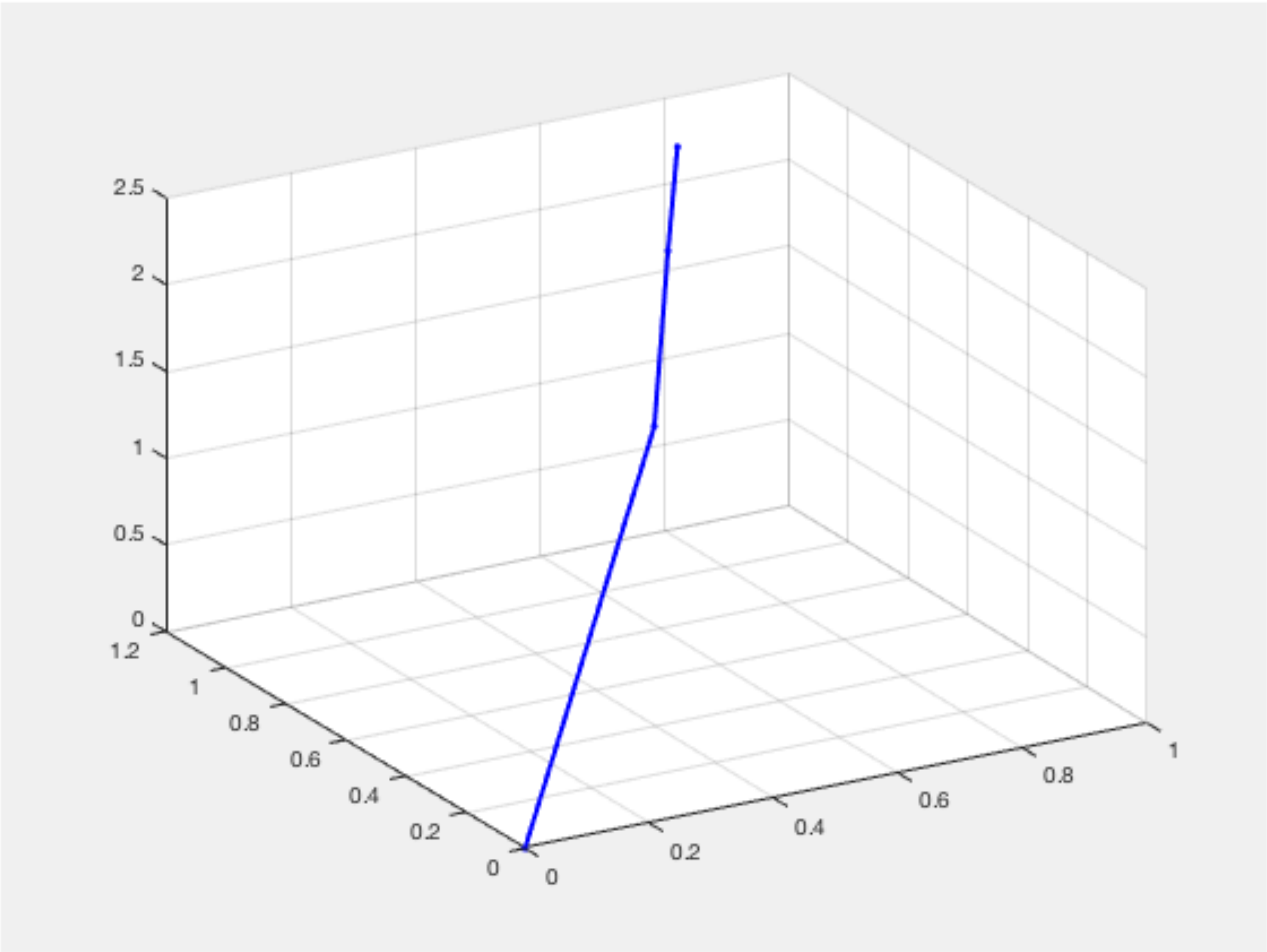
```
RRkeyboardLine =

ADD fprintf('%.2f\n',RRcputime)  RRtasklist.t0=0.05  KILLLAST  RRstop=RRcputime+1;

====LOOP STARTS 29-Mar-2017 19:56:17 for 600 SECONDS with CYCLETIME 0.100 SECONDS===========>
> END =======
RRrun>>879.94

RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {' fprintf('%.2f\n',RRcputime)'  [Inf]  [1]}


879.99

RRtasklist =

  struct with fields:

       t0: 0.0500
    twarn: 0.1000
    tstop: 1
      cnt: 2
    tlist: {0×3 cell}


================LOOP ENDS 29-Mar-2017 19:56:18 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 copyplot
\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 29-Mar-2017 19:56:18 for 600 SECONDS with CYCLETIME 0.100 SECONDS===========>
```

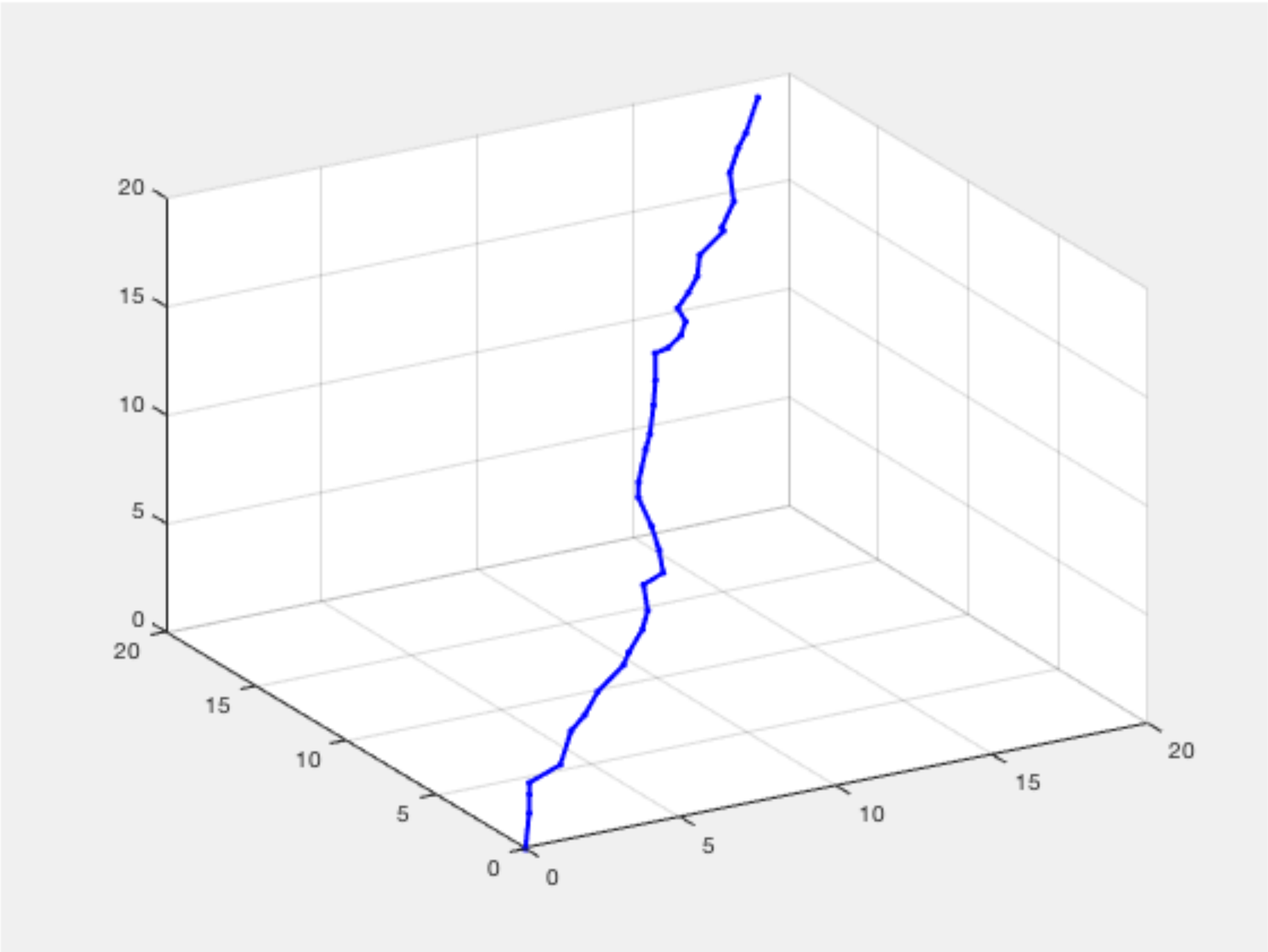
```
> END =======
RRrun>>
RRtasklist =

  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 257 milliseconds

================LOOP ENDS 29-Mar-2017 19:56:23 USING 4.09 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 29-Mar-2017 19:56:23 for 600 SECONDS with CYCLETIME 0.100 SECONDS==========>
> END =======
RRrun>>
New RRtasklist loaded from RRtasklist.mat


RRrun: realtime condition broken by 194 milliseconds

===============LOOP ENDS 29-Mar-2017 19:56:27 USING 3.72 SECONDS ========================
```

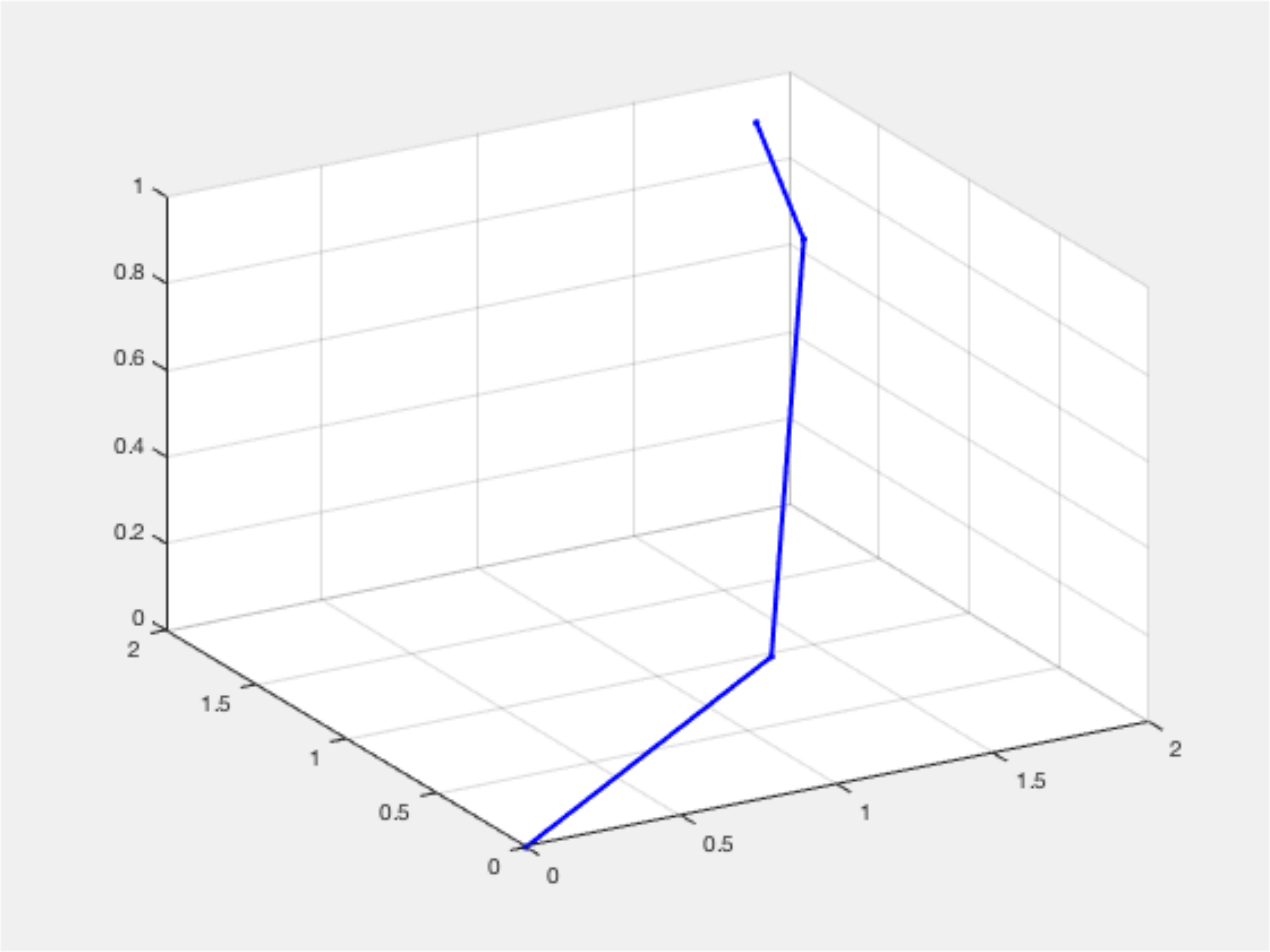


7. Execute a command file

```
RRrun 'execute RRrun_testcommands.txt'
```

```
RRkeyboardLine =

execute RRrun_testcommands.txt

====LOOP STARTS 29-Mar-2017 19:56:27 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=[P...'    [Inf]    [1]
    ' global PL; delet...'    [Inf]    [3]

RRtasklist currently stopped. Use "START" or "STEP" to start task execution.

Elapsed time is 0.009365 seconds.



RRrun: realtime condition broken by 193 milliseconds

RRrun>>================LOOP ENDS 29-Mar-2017 19:56:31 USING 4.02 SECONDS ====================
=====
```



8. Edit a command file

```
edit 'RRrun_testcommands.txt'
```

Final Remarks

```
close all
VLFLlicense
```

```
This VLFL-Lib, Rel. 3.7 (2017-Mar-30), is for limited non commercial educational use only!
Licensee: 3.7 Tim Lueth for academic and education 29-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.7) license will exceed at 25-Mar-2018!
Executed 29-Mar-2017 19:56:34 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ==============================
========
database_toolbox
image_toolbox
map_toolbox
matlab
robotics_system_toolbox
simmechanics
simscape
simulink
=============================================================================================
========
```