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

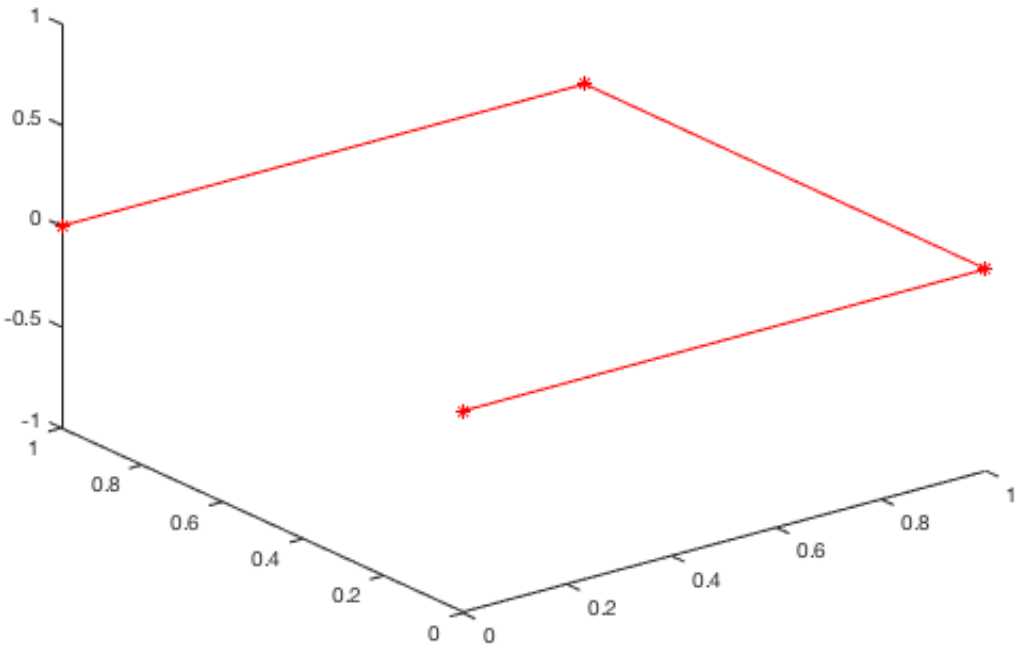
A point list is a nx2 array. The number n is the number of 2D coordinate points [x y]. In general, such a point list can be the basis for designing a boundary surface model. We start with some simple functions to display polygons:

- **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

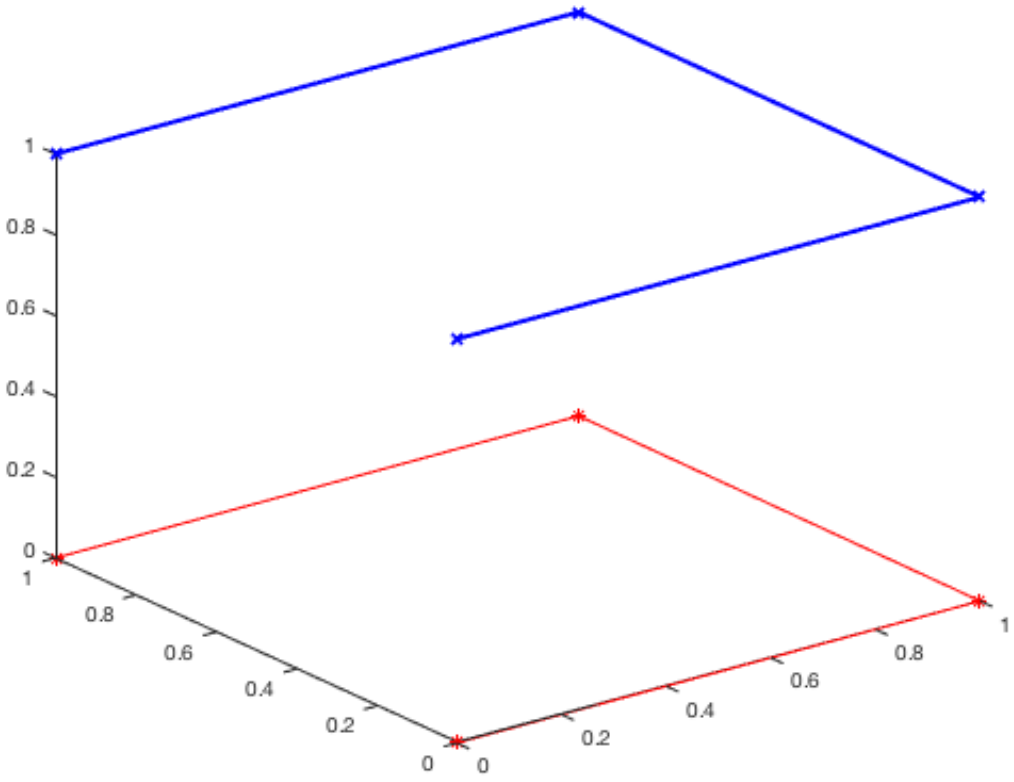
A vertex list is a nx3 array. The number n is the number of 3D coordinate points [x y z]. In fact, the point list can be transfered into a vertex list by appending a third column containing the z-coordinate such as zero or another z-coordinate.

- **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
```

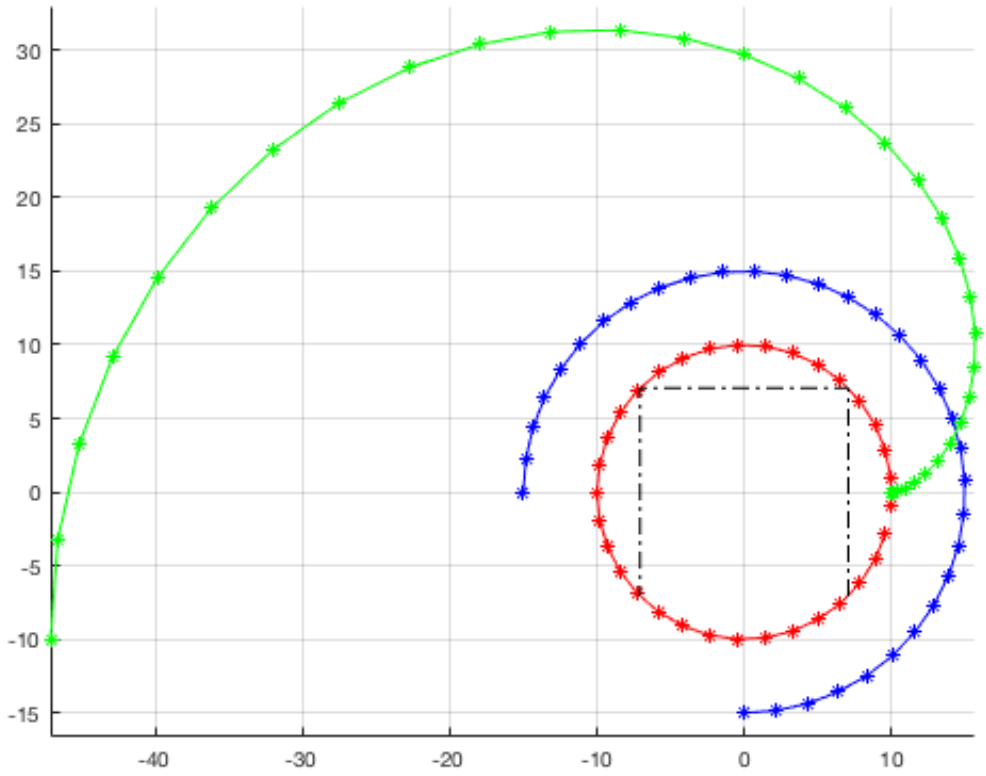


3. Predefined functions for the generation of often used planar polygons (PL)

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;
```

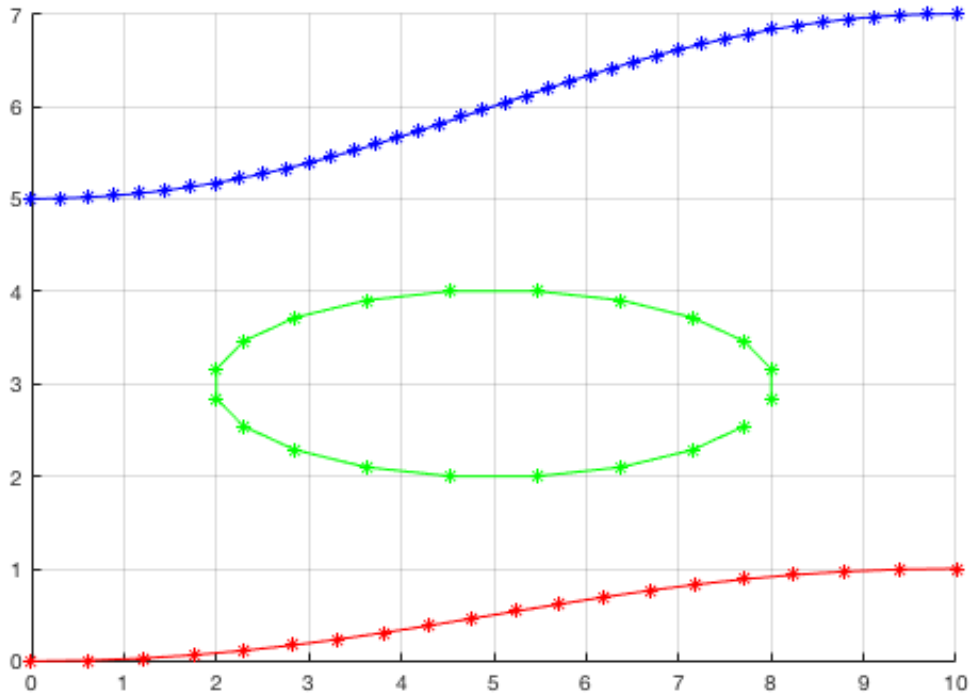


4. More predefined functions for planar polygons in 3D (VL)

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

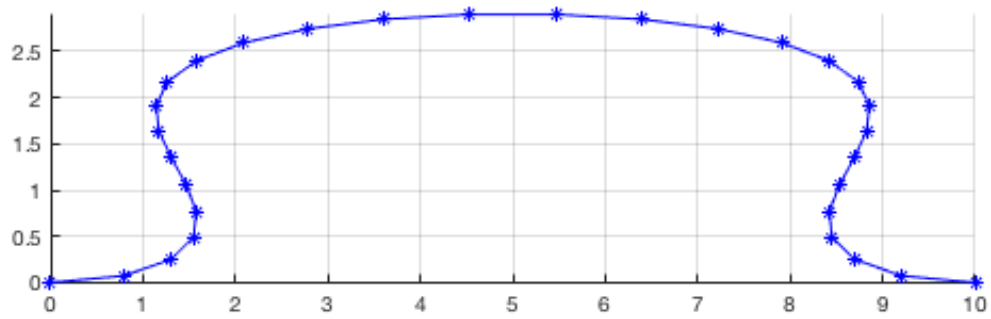
- **VLpolygon** to generate elliptic contours.
- **VLBezier4P** to generate a Bezier-curve using 4 points.
- **VLBezierC** to generate a Bezier-curve using as many points as possible.
- **VLremstraightCVL** to remove obsolete points on straight lines.

```
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
```

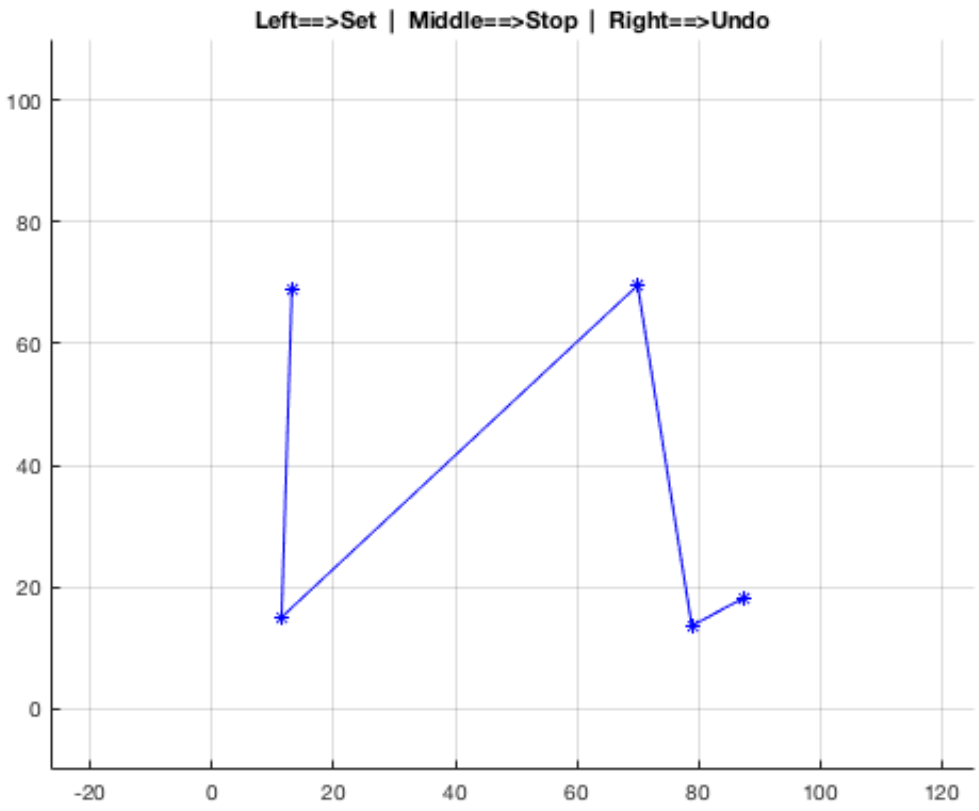


- **VLui** as an user interface to enter points by mouse clicks.

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

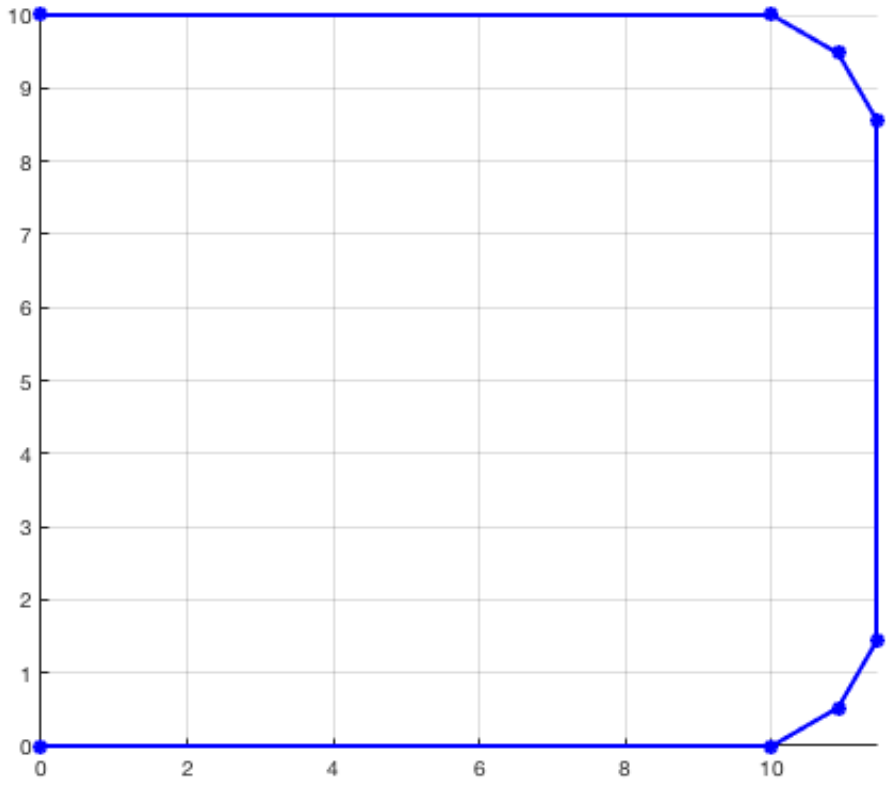
```
VL =

   13.1000   68.9000         0
   11.4000   15.0000         0
   70.0000   69.6000         0
   78.9000   13.6000         0
   87.5000   18.2000         0
   87.5000   18.2000         0
```



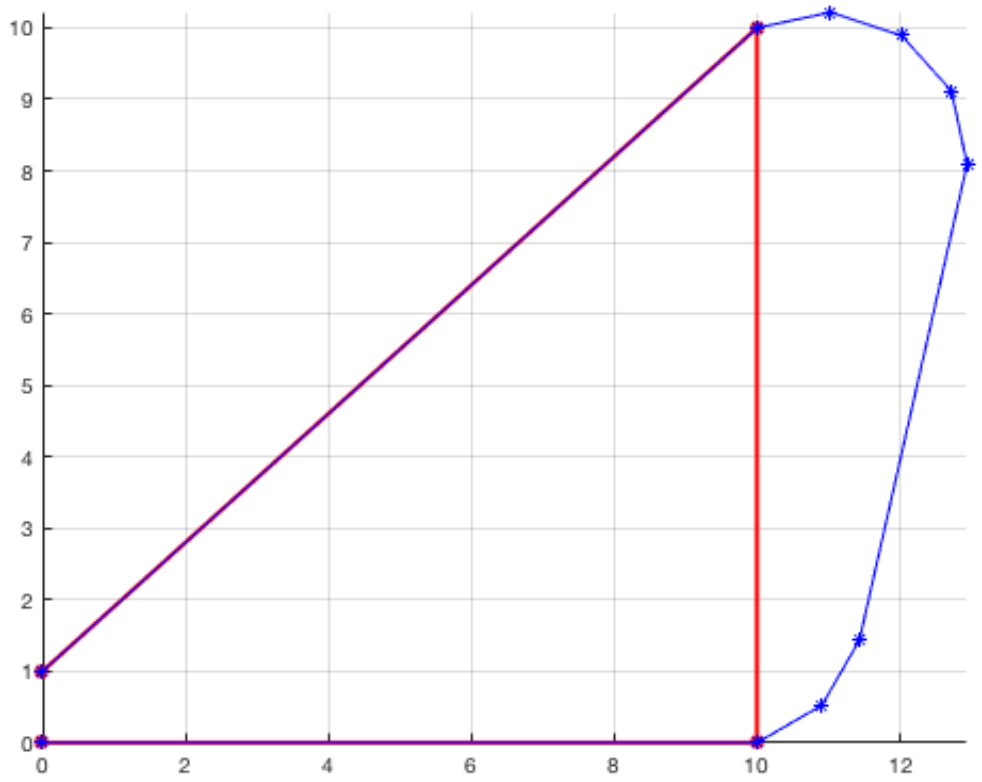
- **VLRadius4P** for inserting points to generate radial curves instead of corners.

```
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

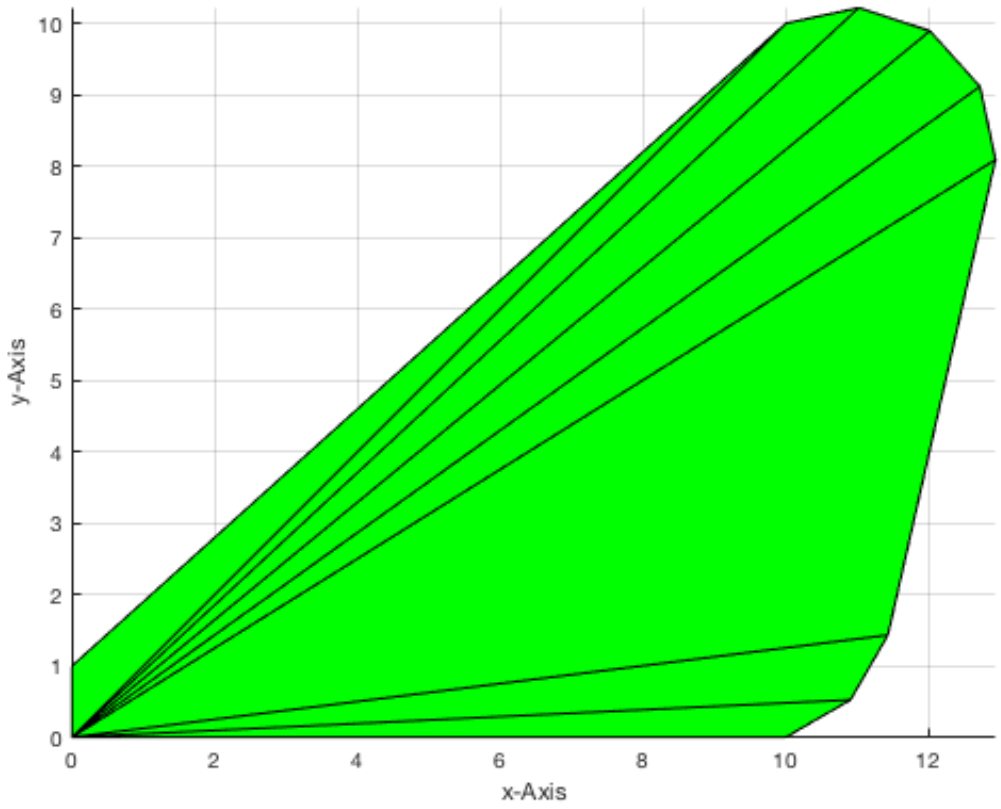
If we have a closed convex polygon, it is possible to generate a surface desciption by a facet list (FL) describing triangle facets. This is called tesselation of the closed polygon/surface. For closed convex polygons, the simplest form are facets from the 1st to the 2nd and 3rd points [1 2 3], then 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 =

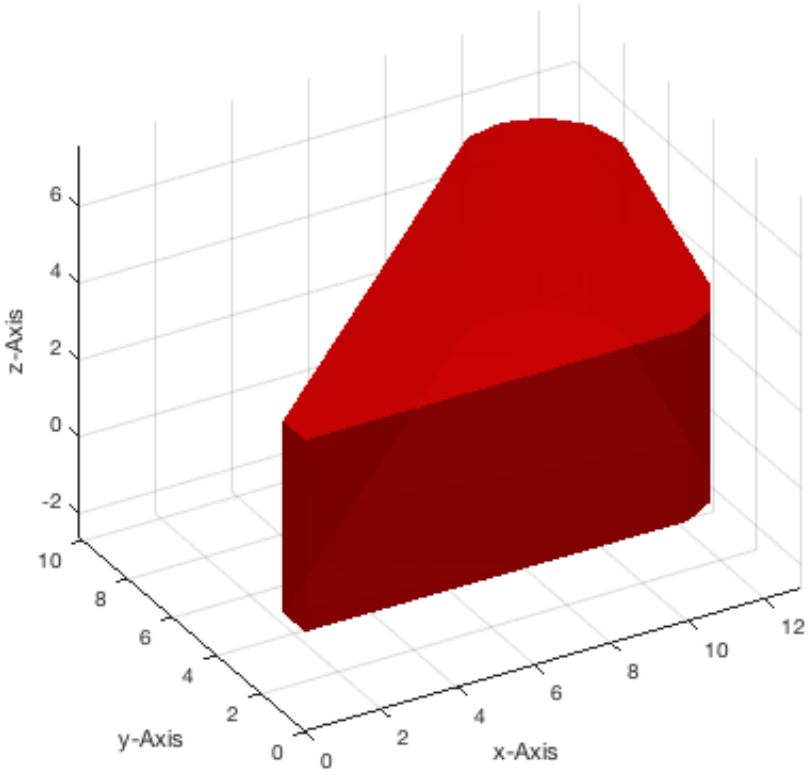
     1     2     3
     1     3     4
     1     4     5
     1     5     6
     1     6     7
     1     7     8
     1     8     9
     1     9    10
```



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;
```



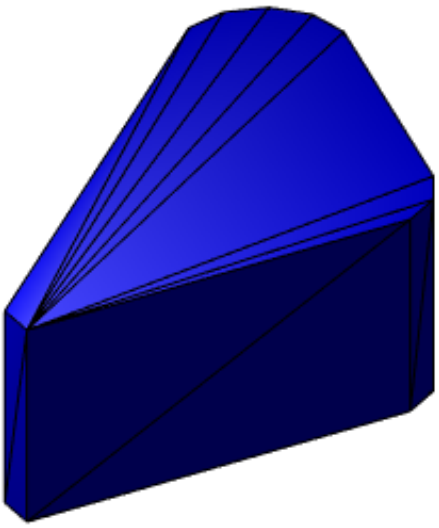
7. Graphical user interface for STL import, export, and viewing

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' : 22-Mar-2017 22:57:41



```
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. 3.6 (2017-Mar-23), is for limited non commercial educational use only!
Licensee: 3.6 Tim Lueth for academic and education 05-Mar-2017 !
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. 3.6) license will exceed at 18-Mar-2018!
Executed 22-Mar-2017 22:57:44 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simmechanics
simscape
simulink
================================================================================================
==========
```

Published with MATLAB® R2016b