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.

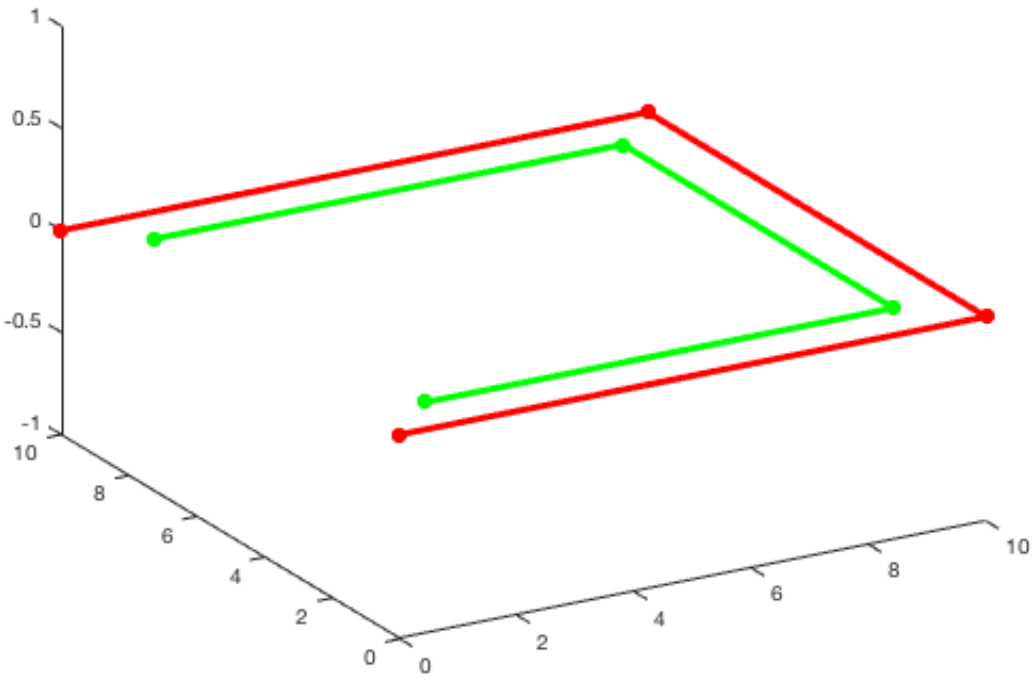
```
close all;
PLA=[0 0; 10 0; 10 10; 0 10], PLB=[1 1; 9 1; 9 9; 1 9]
PLplot (PLA,'r-*',3);PLplot (PLB,'g-*',3); view(-30,30);
```

```
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:

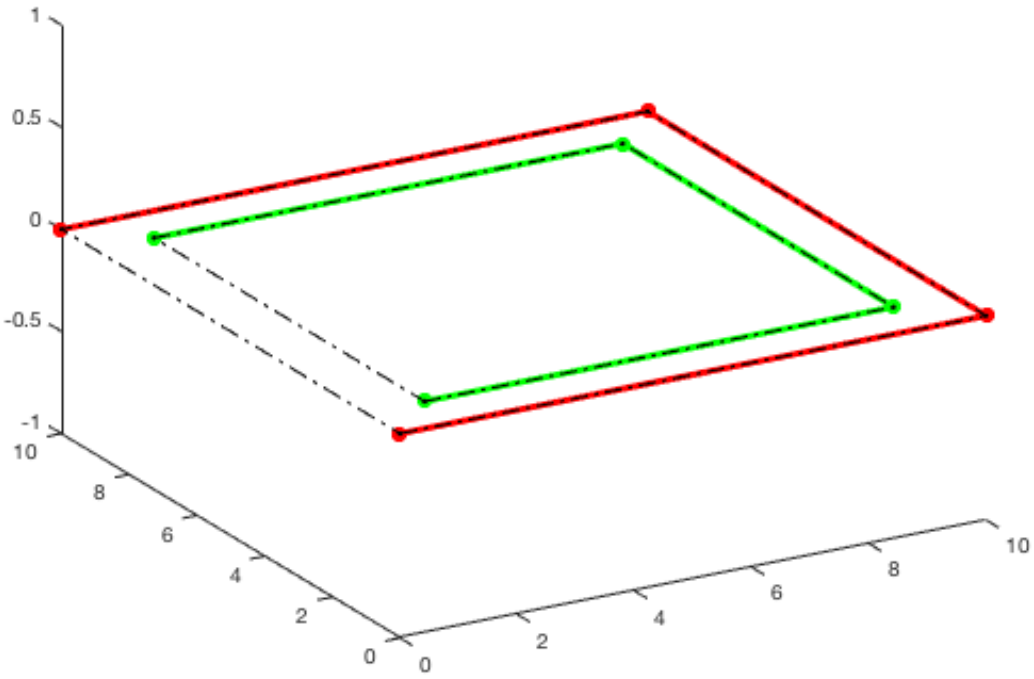
- **CPLplot** draws a closed contour polybool list (CPL).
- **PLELofCPL** draws a start point, end point and direction-arrows, when called without any output variable.

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

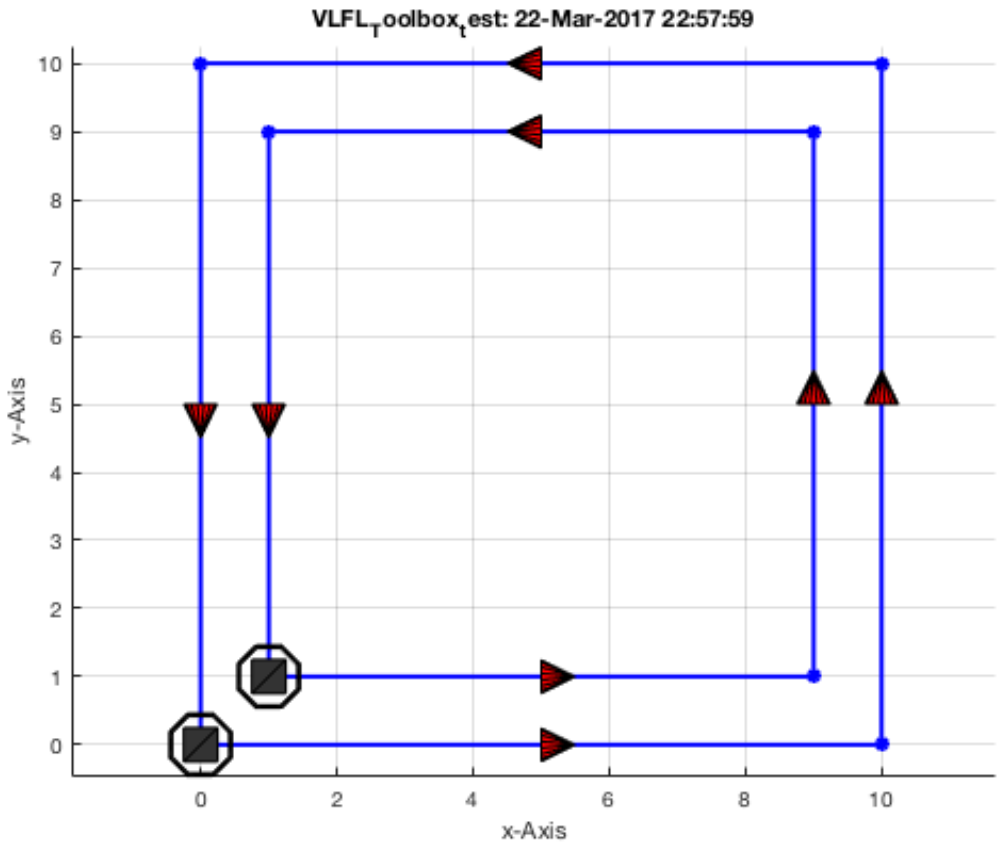
```
CPL =

     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.

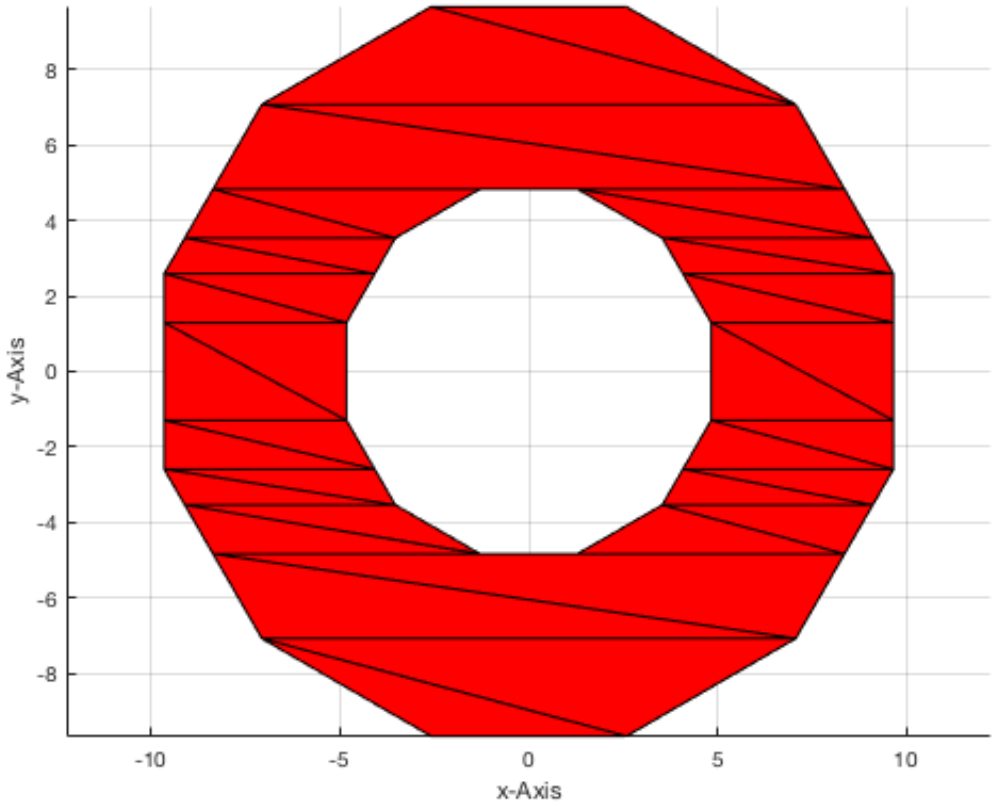
It is not unique to tesselate polygons, if they are enclosed or cross each other. There exist no general purpose solution. Nevertheless, in most cases, the Delaunay-Triangulation is preferrable, since this concept does work also in 3D. To generate a facet list (FL) for a CPL, there exist two functions. 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).

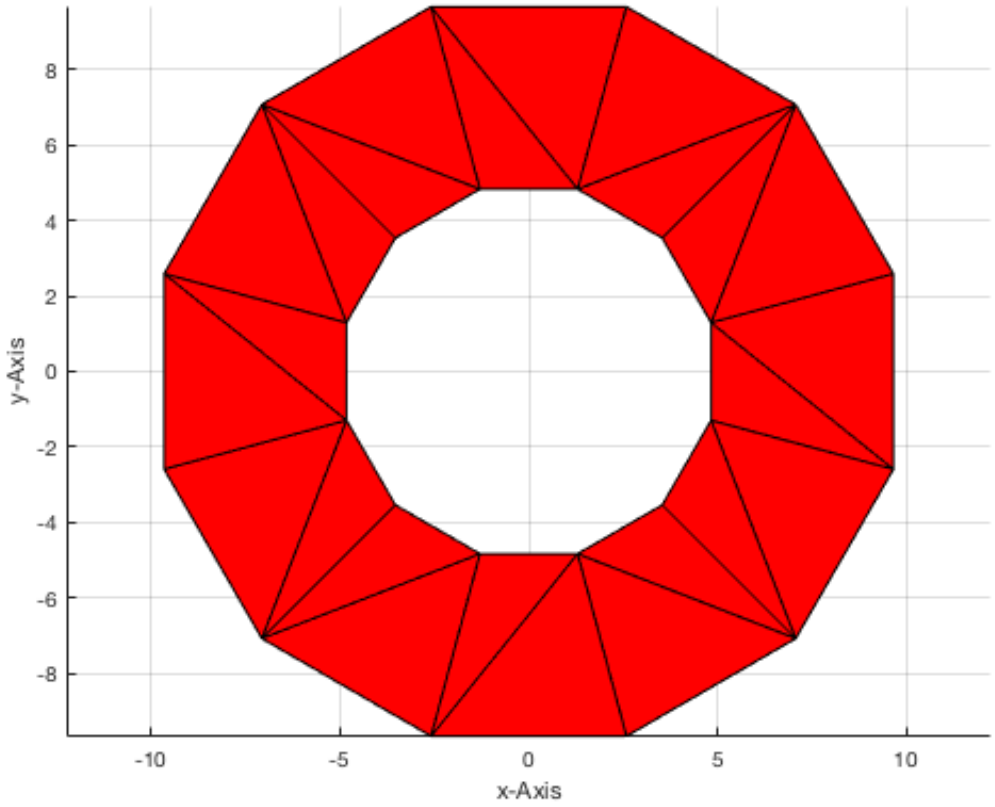
```
close all;
CPL=[PLcircle(10,12);NaN NaN;PLcircle(5,12)]
[PL,FL]=PLFLofCPLpoly(CPL); VLFLplot(PL,FL);
```

```
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
```



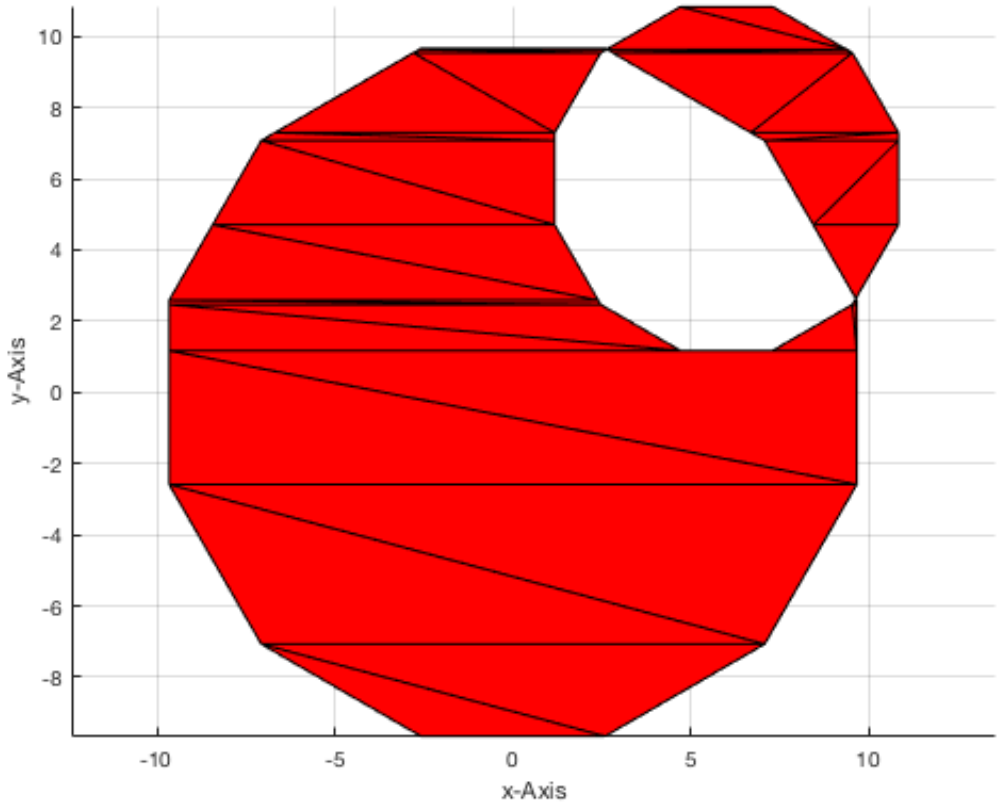

```
close all;
[PL,FL]=PLFLofCPLdelaunay(CPL); VLFLplot(PL,FL);
```

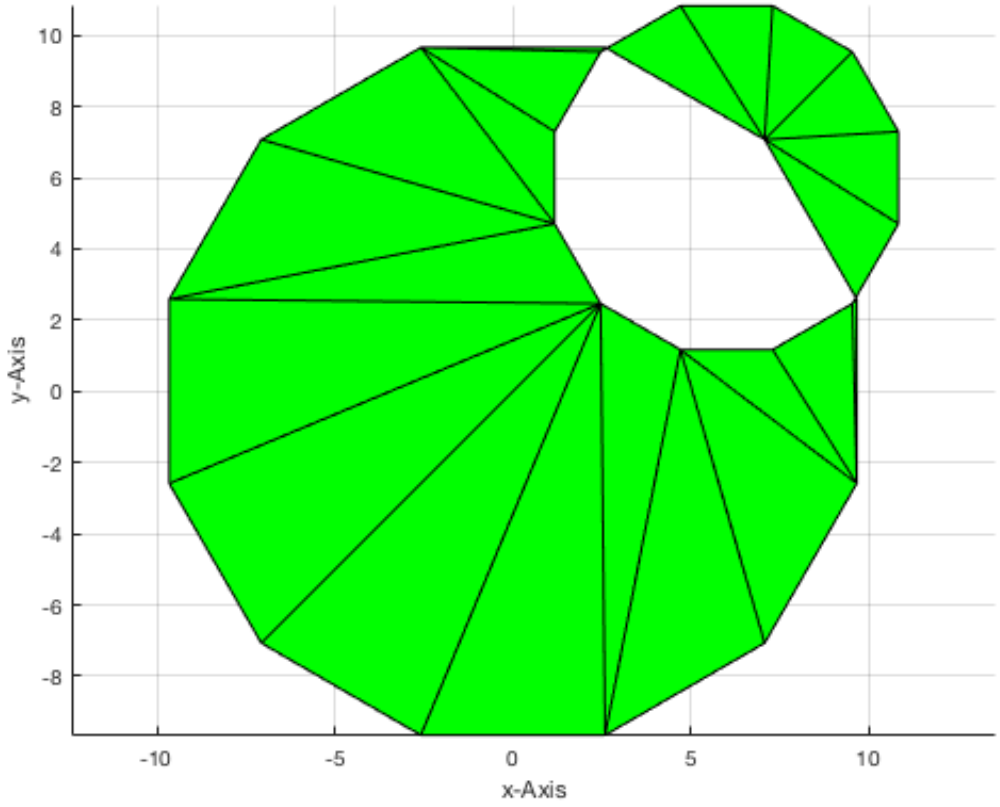


The next example shows the need for splitting contours:

```
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.
```





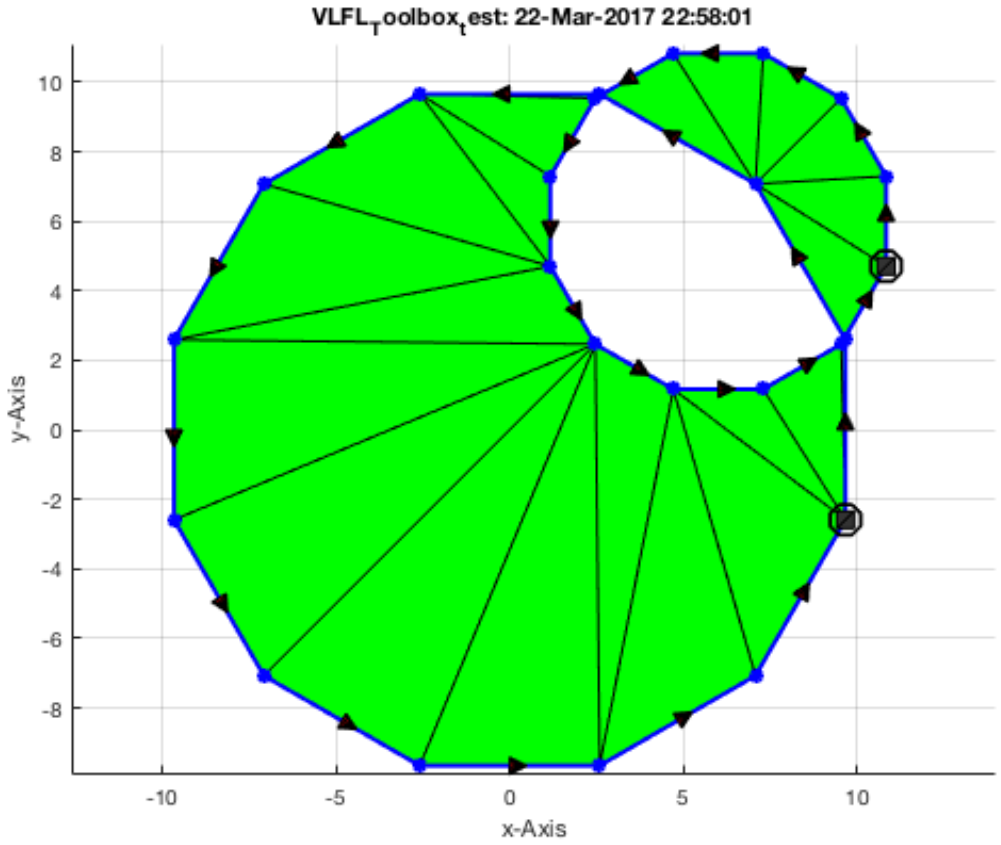
4. Orientation of outer and inner polygons of a CPL

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.

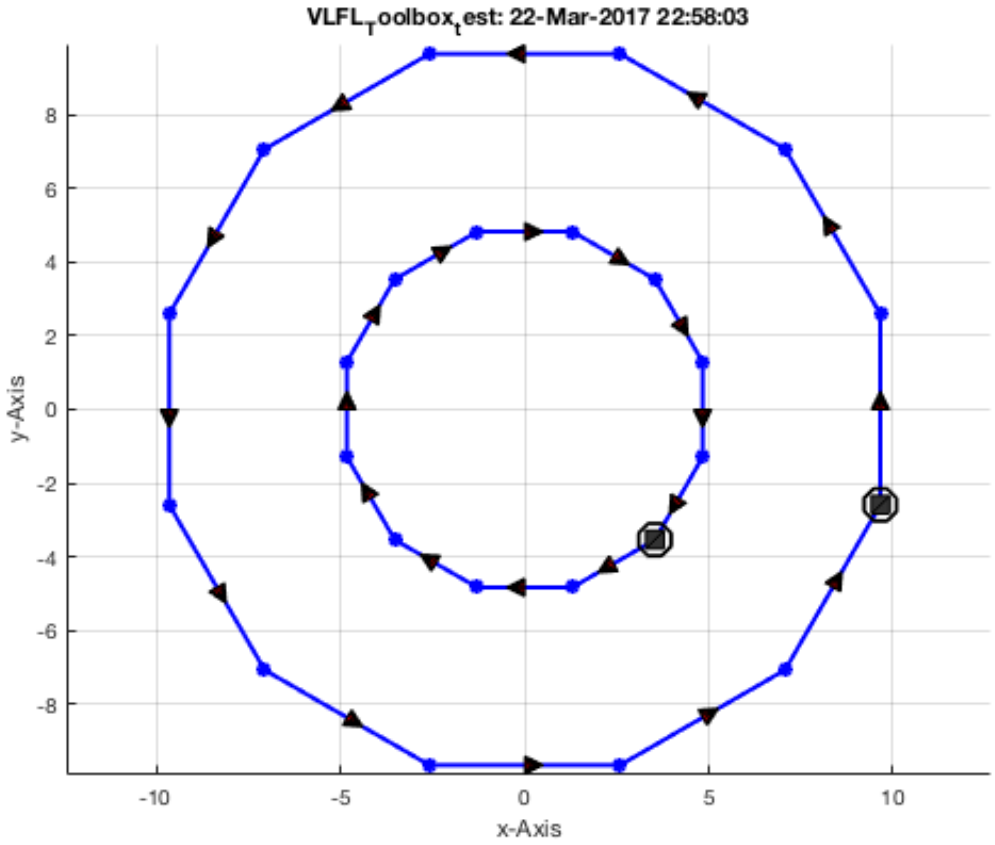
In the next figure, we see both polygons counter-clockwise:

```
PLELofCPL(CPL);
```



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

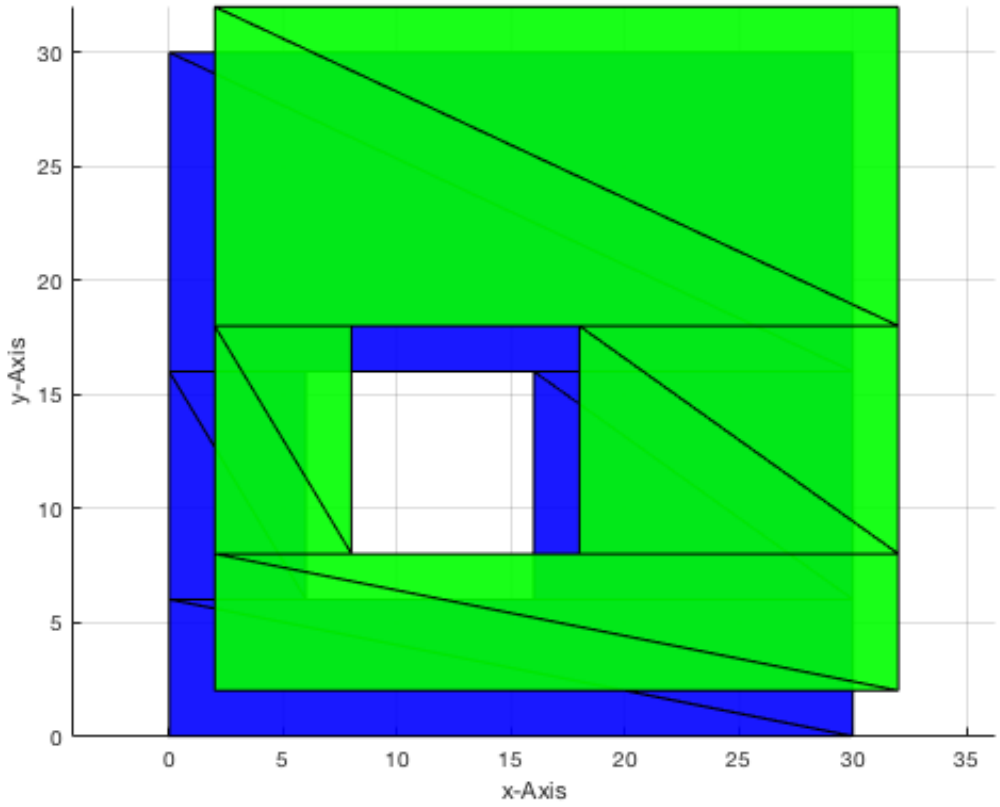
```
close all;
CPL=[PLcircle(10,12);NaN NaN;flip(PLcircle(5,12))];
PLELofCPL(CPL);
```



5. Boolean operations of contour polybool lists (CPL)

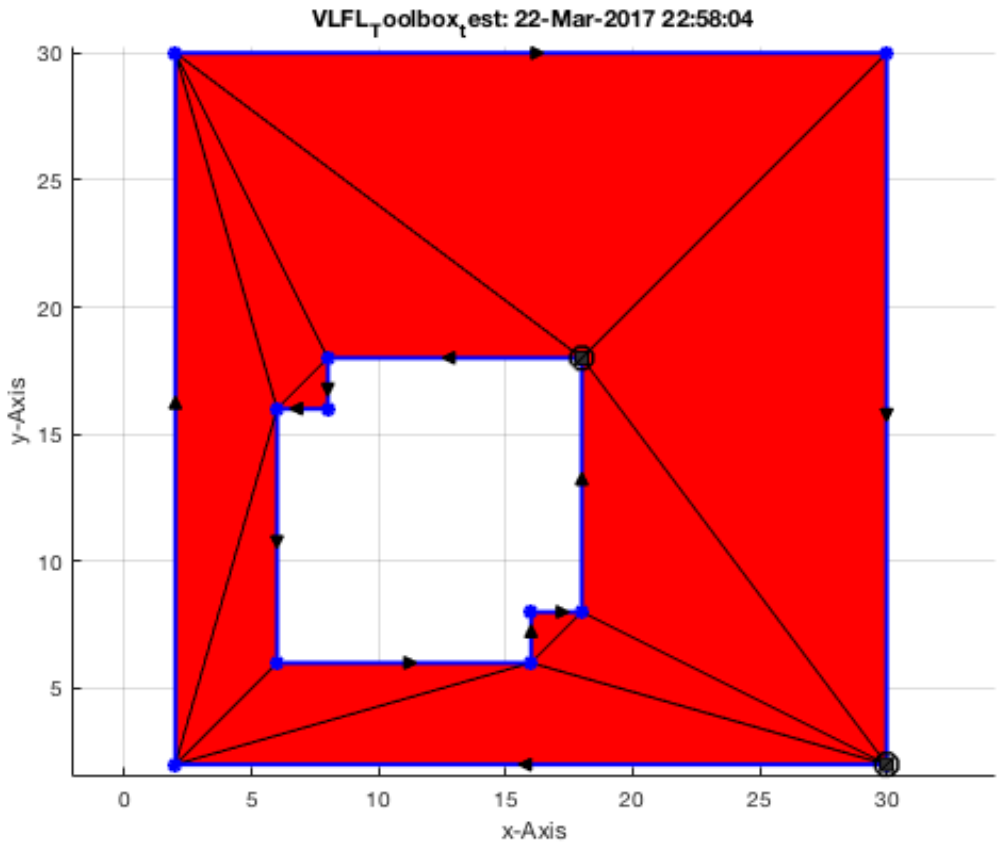
The main advantage of the CPL representation is currently the possibility to use the polybool functions of the mapping toolbox. Here, we show the use in embedded functions that help later to make the step forward to 3D modeling. We start with boolean operations of contour polybool lists (CPL).

```
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)
```



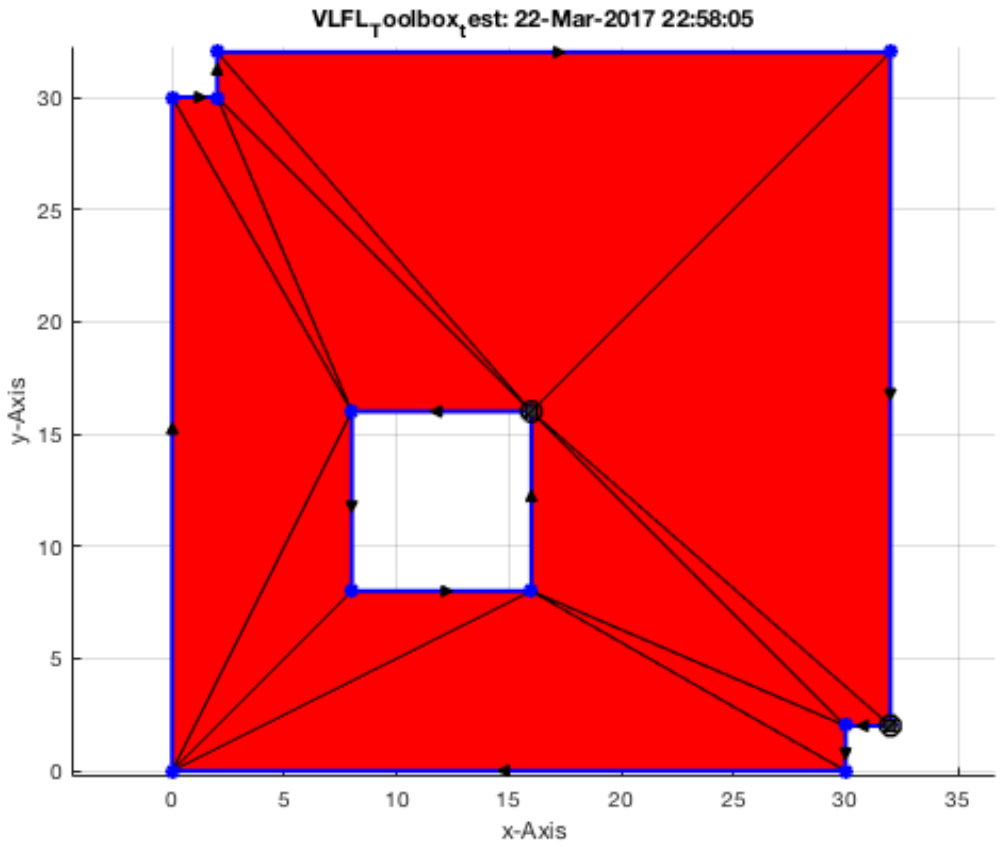
- **CPLpolybool('and',CPLA,CPLB)** delivers CPLA intersecting CPLB.

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



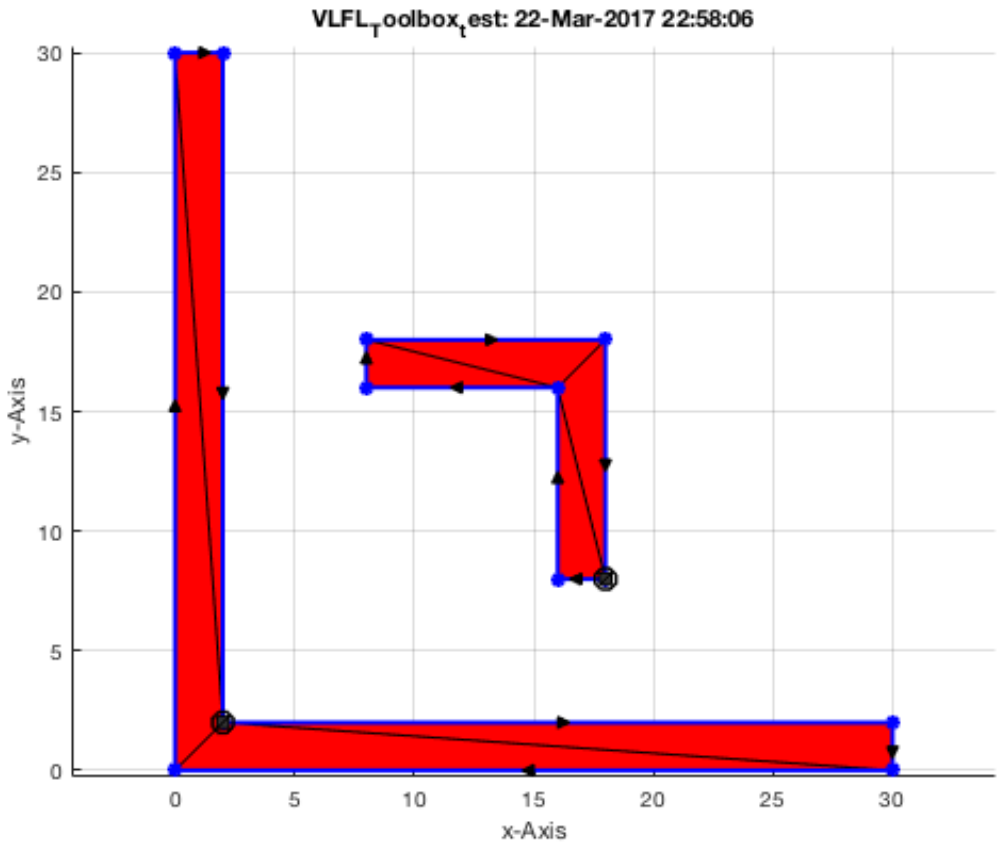
- **CPLpolybool('or',CPLA,CPLB)** delivers CPLA united with CPLB.

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



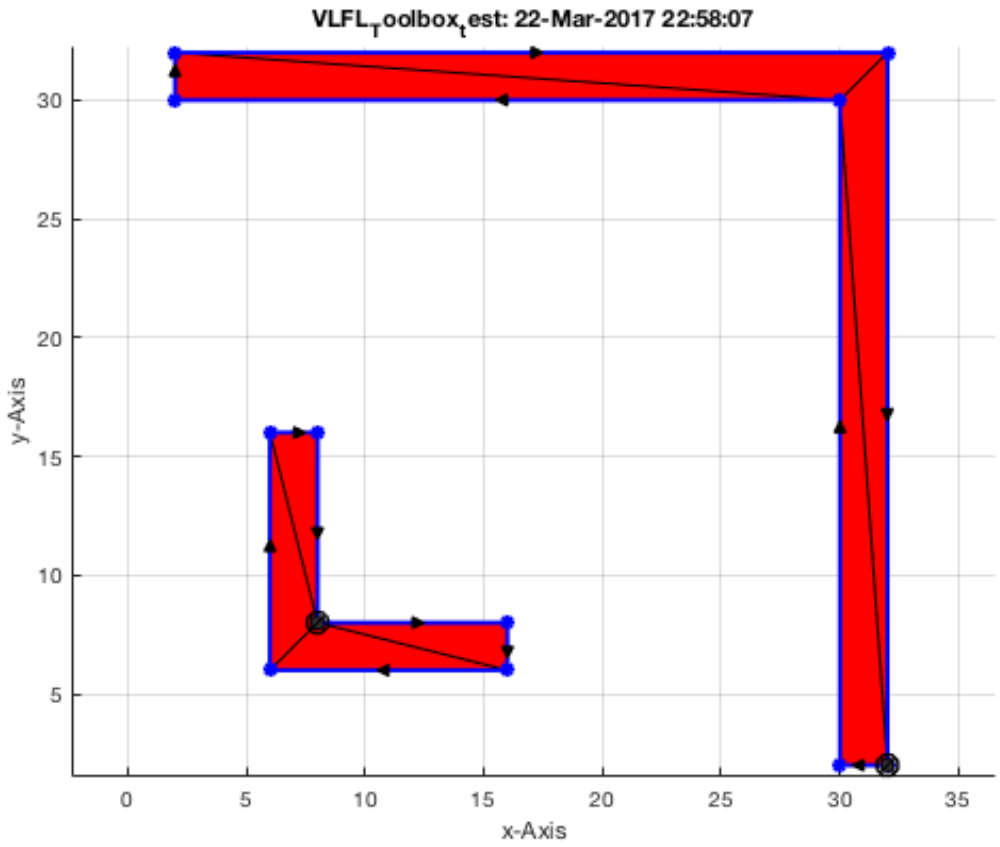
- **CPLpolybool('minus',CPLA,CPLB)** delivers CPLA minus CPLB.

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



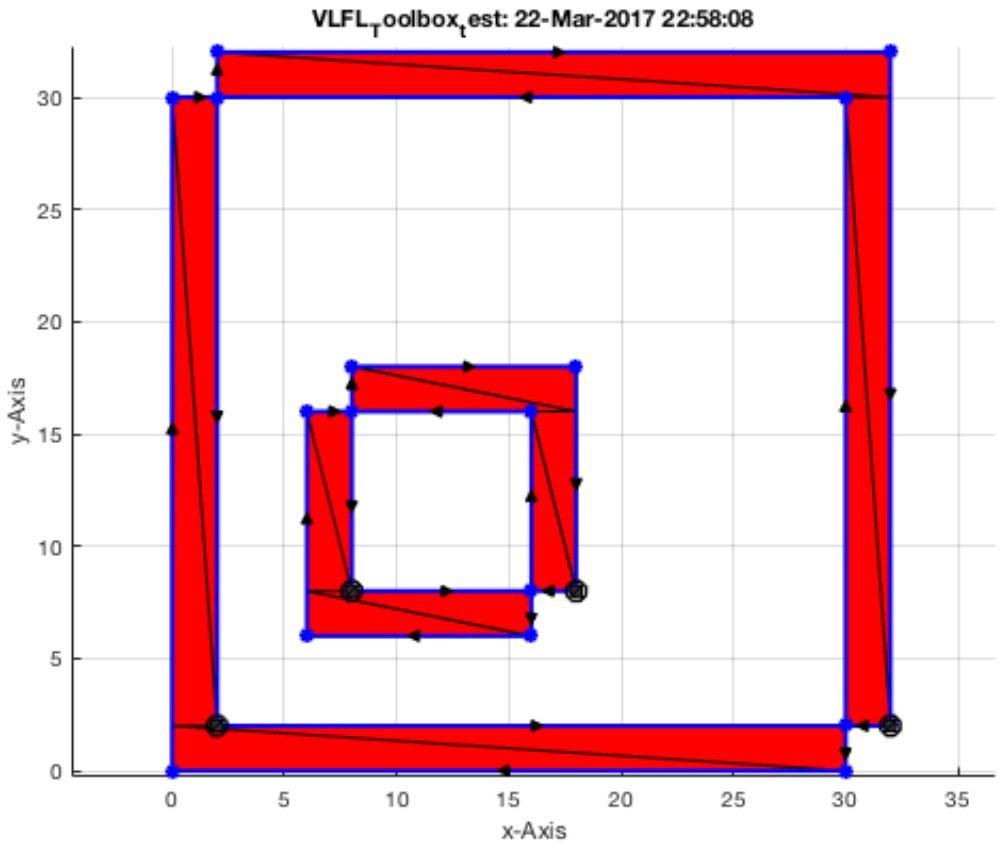
- **CPLpolybool('minus',CPLB,CPLA)** delivers CPLB minus CPLA.

```
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 =

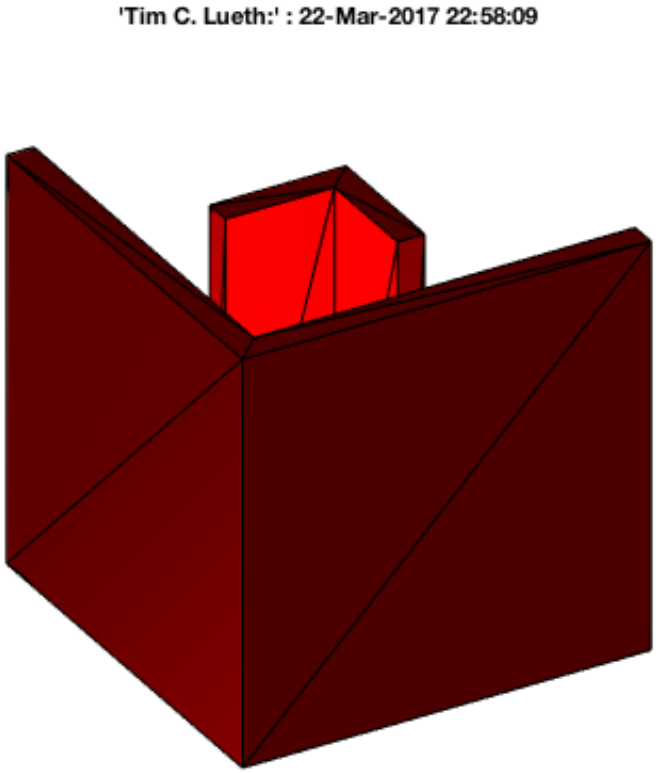
     7    12
    12    11
    11    10
    10     9
     9     8
     8     7
     1     6
     6     5
     5     4
     4     3
     3     2
     2     1
```

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.

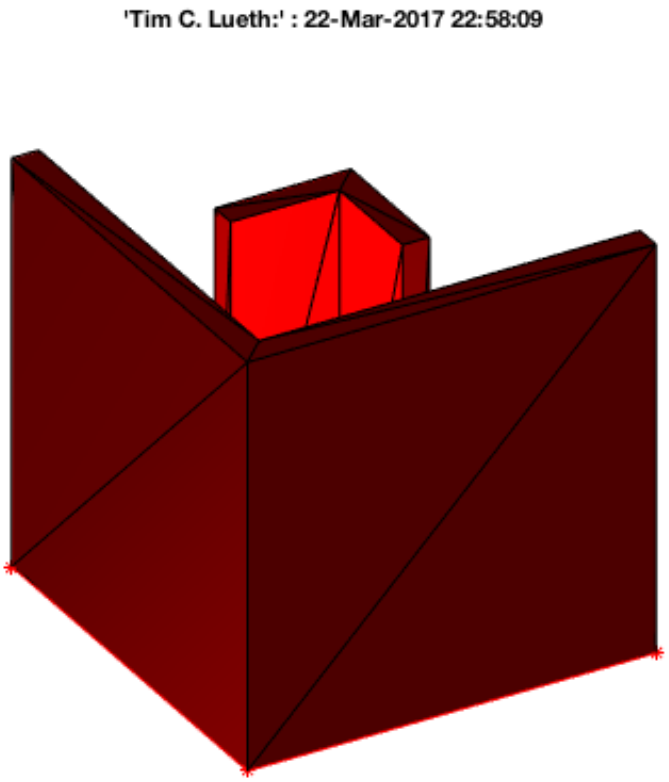
- **CPLofPLEL** converting a point list (PL) and an edge list (EL) into a closed polybool list.

```
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
```



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
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 22:58:11 by 'lueth' using Matlab 9.1.0.441655 (R2016b) on a MACI64
======================================== Used Matlab products: ============================
==========
map_toolbox
matlab
simmechanics
simscape
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