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

2014-11-19: Tim C. Lueth, Professor at Technische Universität München, Germany (URL: http://www.SG-Lib.org) - Last Change: 2019-06-11

Contents

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

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

- Tutorial 01: First Steps Using the VLFL-Toolbox for Solid Object Design
- Tutorial 02: Using the VLFL-Toolbox for STL-File Export and Import
- Tutorial 03: Closed 2D Contours and Boolean Operations in 2D
- Tutorial 04: 2½D Design Using Boolean Operators on Closed Polygon Lists (CPL)
- Tutorial 05: Creation, Relative Positioning and Merging of Solid Geometries (SG)
- Tutorial 06: Relative Positioning and Alignment of Solid Geometries (SG)
- Tutorial 07: Rotation of Closed Polygon Lists for Solid Geometry Design
- Tutorial 08: Slicing, Closing, Cutting and Separation of Solid Geometries
- Tutorial 09: Boolean Operations with Solid Geometries
- Tutorial 10: Packaging of Sets of Solid Geometries (SG)
- Tutorial 11: Attaching Coordinates Frames to Create Kinematik Models
- Tutorial 12: Define Robot Kinematics and Detect Collisions
- Tutorial 13: Mounting Faces and Conversion of Blocks into Leightweight-structures
- Tutorial 14: Manipulation Functions for Closed Polygons and Laser Cutting (SVG)
- Tutorial 15: Create a Solid by 2 Closed Polygons
- Tutorial 16: Create Tube-Style Solids by Succeeding Polygons
- Tutorial 17: Filling and Bending of Polygons and Solids
- Tutorial 18: Analyzing and modifying STL files from CSG modeler (Catia)
- Tutorial 19: Creating drawing templates and dimensioning from polygon lines
- Tutorial 20: Programmatically Interface to SimMechanics Multi-Body Toolbox
- Tutorial 21: Programmatically Convert Joints into Drives (SimMechanics)
- Tutorial 22: Adding Simulink Signals to Record Frame Movements
- Tutorial 23: Automatic Creation of a Missing Link and 3D Print of a Complete Model
- Tutorial 24: Automatic Creation of a Joint Limitations
- Tutorial 25: Automatic Creation of Video Titels, Endtitels and Textpages
- Tutorial 26: Create Mechanisms using Universal Planar Links
- Tutorial 27: Fourbar-Linkage: 2 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 28: Fourbar-Linkage: 3 Pose Syntheses and Linkage Export for 3D Printing
- Tutorial 29: Create a multi body simulation using several mass points
- Tutorial 30: Creating graphical drawings using point, lines, surfaces, frames etc.
- Tutorial 31: Importing 3D Medical DICOM Image Data and converting into 3D Solids
- Tutorial 32: Exchanging Data with a FileMaker Database
- Tutorial 33: Using a Round-Robin realtime multi-tasking system
- Tutorial 34: 2D Projection Images and Camera Coordinate System Reconstruction
- Tutorial 35: Creation of Kinematic Chains and Robot Structures
- Tutorial 36: Creating a Patient-Individual Arm-Skin Protector-Shell
- Tutorial 37: Dimensioning of STL Files and Surface Data

- Tutorial 38: Some more solid geometry modelling function
- Tutorial 39: HEBO Modules robot design
- Tutorial 40: JACO Robot Simulation and Control
- Tutorial 41: Inserting Blades, Cuts and Joints into Solid Geometries
- Tutorial 42: Performing FEM Stress and Displacement Analysis and Structural Optimization of Solids
- Tutorial 43: Performing FEM Structural Optimization (CAO) and Topological Optimization (SKO) of Solids
- Tutorial 44: Creation of solids and kinematics from 3D curves and transformation matrices
- Tutorial 45: Creation of Solids using the SG-Coder - SGofCPLcommand
- Tutorial 46: Creating Fischertechnik compatible gear boxes using SGofCPLcommand
- Tutorial 47: Creating four-joints by 3 pose synthesis

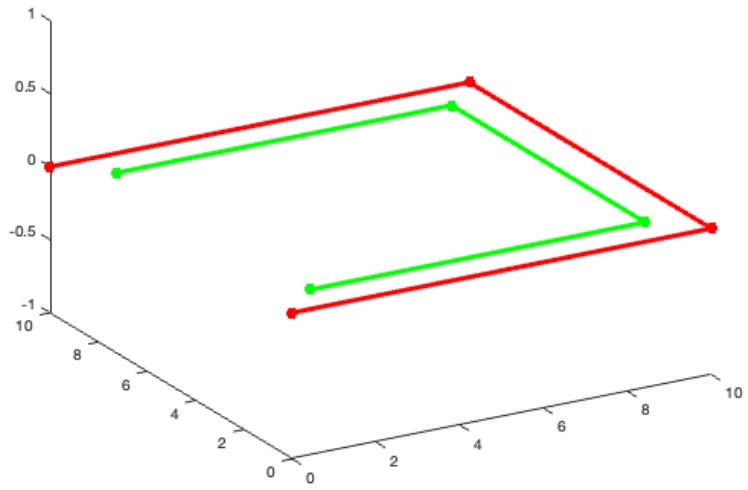
Motivation for this tutorial: (Originally SolidGeometry 1.3 required)

2. The contour polybool list (mapping toolbox)

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

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

```
PLA =
     0     0
    10     0
    10    10
     0    10
PLB =
     1     1
     9     1
     9     9
     1     9
```

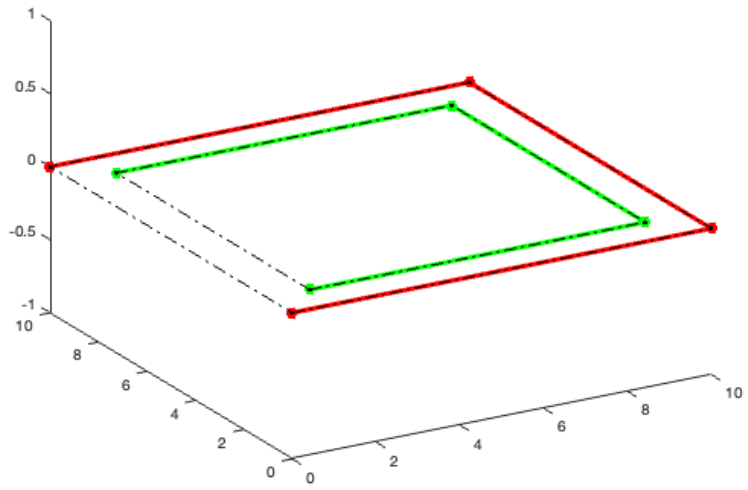


The concatenation generates then the closed polybool list (CPL). Two functions are helpful to draw a CPL and also to show the start point (black cube), the endpoint (black ring) and the direction of each edge of the CPL:

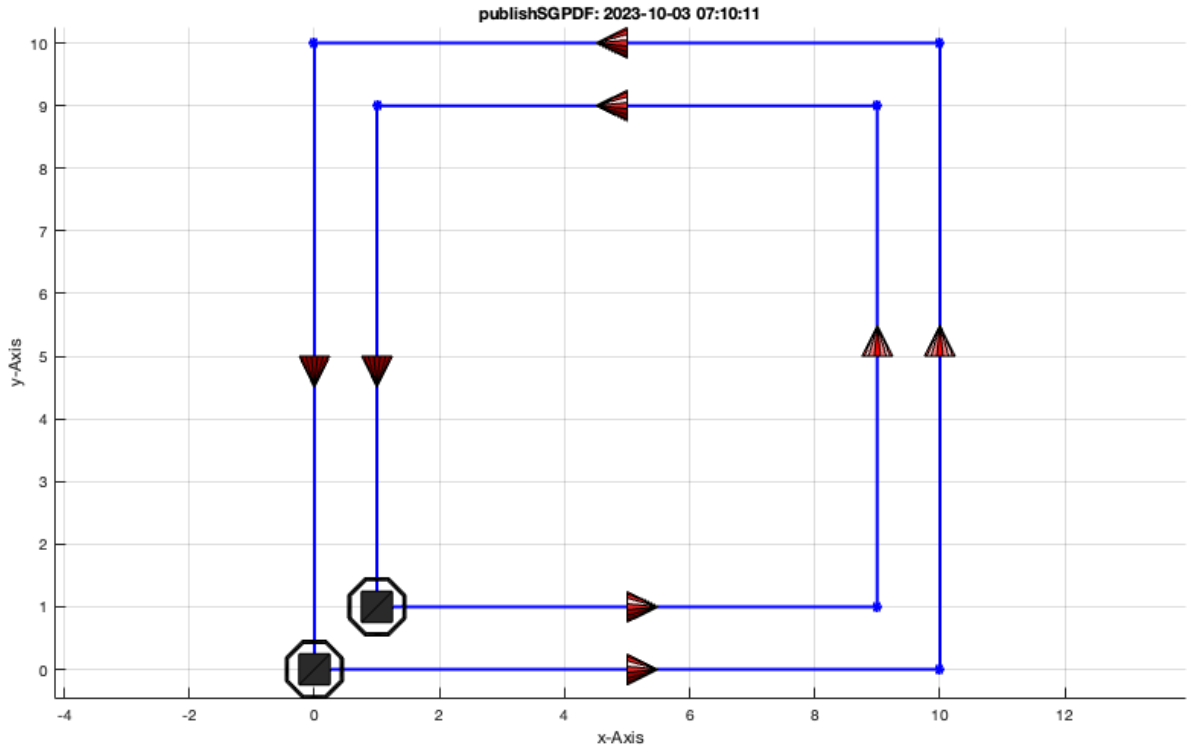
- **CPLplot** draws a closed contour polybool list (CPL).
- **PLELofCPL** draws a start point, end point and direction-arrows, when called without any output variable.

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

```
CPL =
     0     0
    10     0
    10    10
     0    10
   NaN   NaN
     1     1
     9     1
     9     9
     1     9
```




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



3. Surface tesselation for contour polybool list (CPL)

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

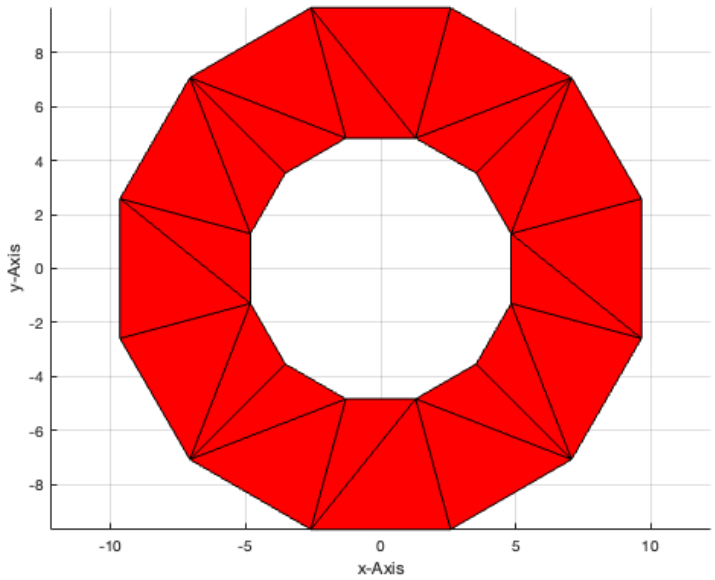
- **Row-Scanning-Triangulation** is able to handle all kinds of polygons (even enclosed), but the triangle facets are sometimes very small. Furthermore, the point contains redundant information.
- **Delaunay-Triangulation** is able to handle all kinds of polygons (even enclosed), but has problems with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

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

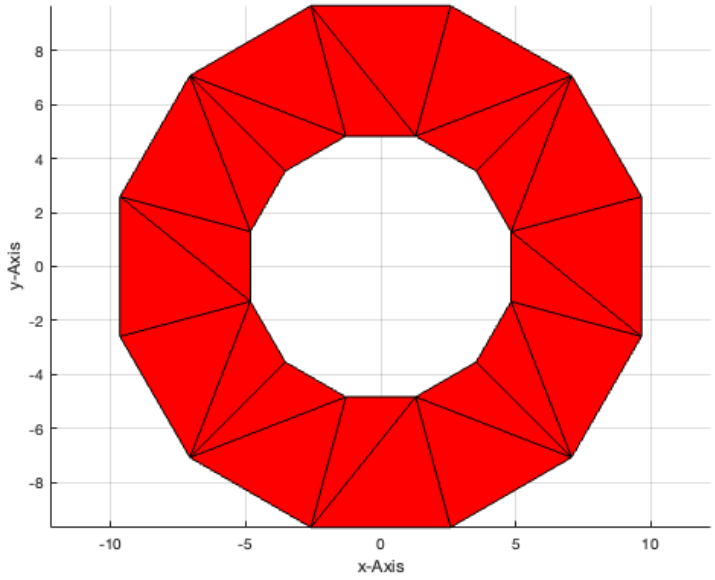
- **PLFLofCPLpoly** returns a facet tesselation by a simple y-coordinate row scanning (the points are ordered by increasing y, contour by contour, do not mix, but are redundant. Not as efficient as Delaunay, and not useful for 3D).
- **PLFLofCPLdelaunay** returns a facet tesselation by a Delaunay-triangulation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

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

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

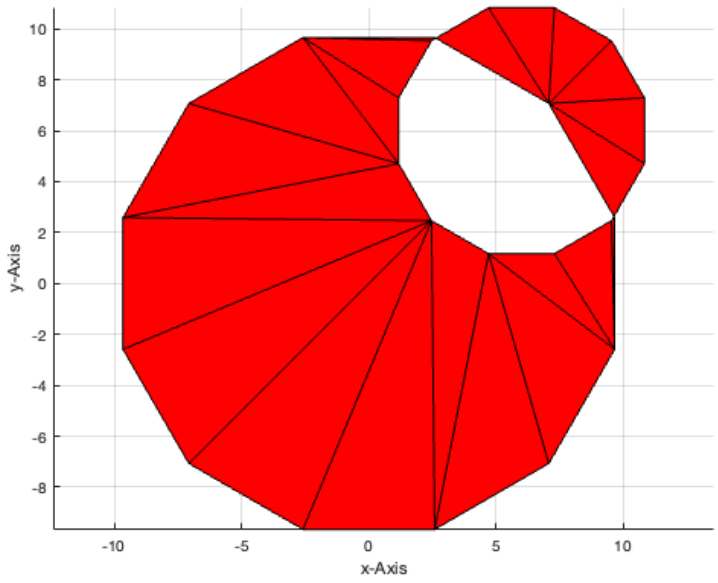


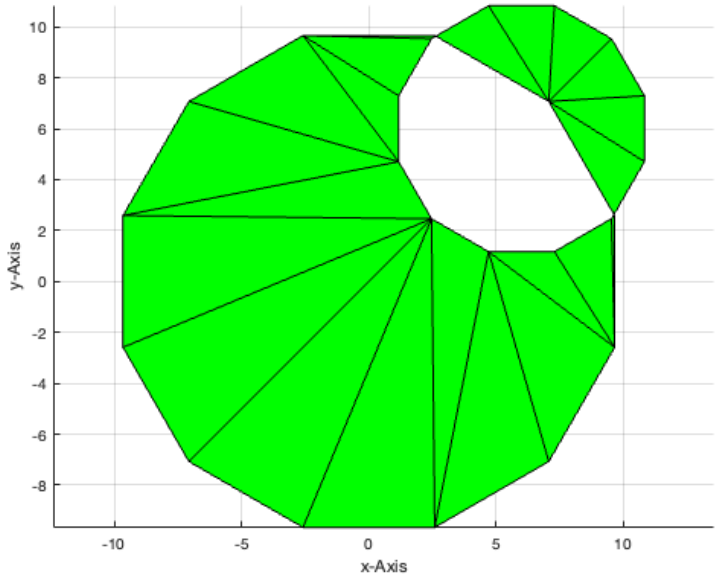

```
close all;
[PL,FL]=PLFLofCPLdelaunay(CPL); VLFLplot(PL,FL);
```



The next example shows the need for splitting contours:

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





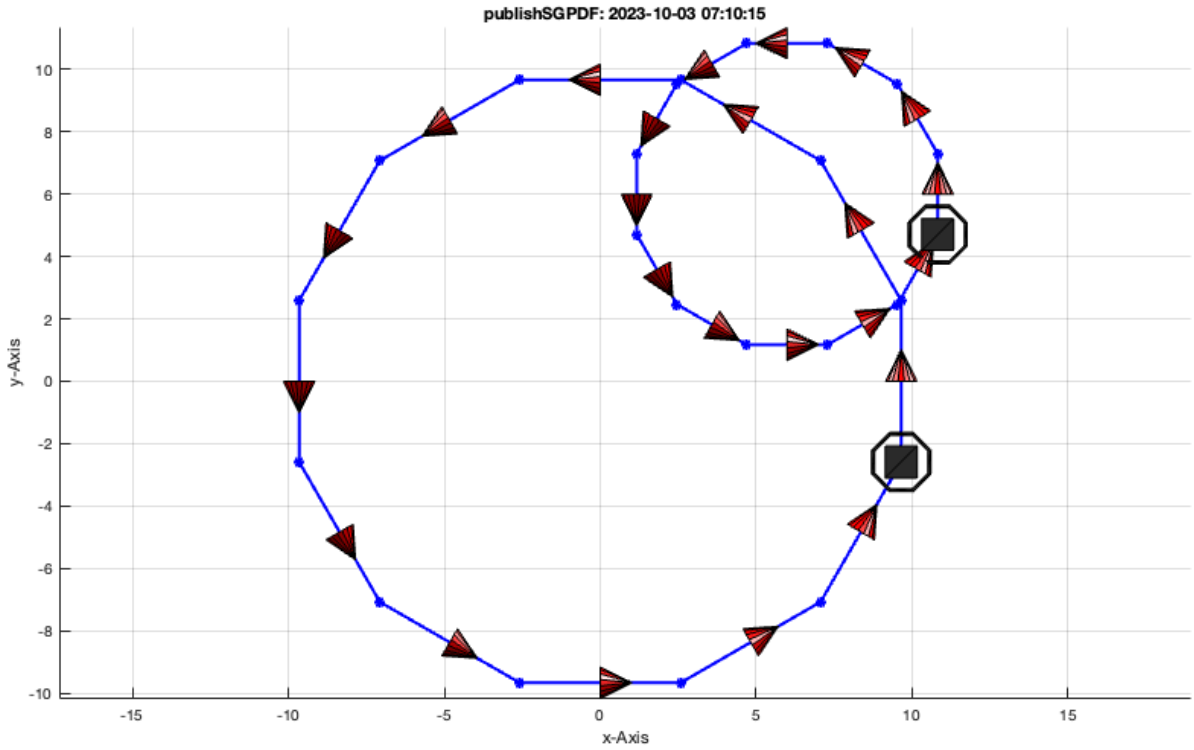
4. Orientation of outer and inner polygons of a CPL

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.

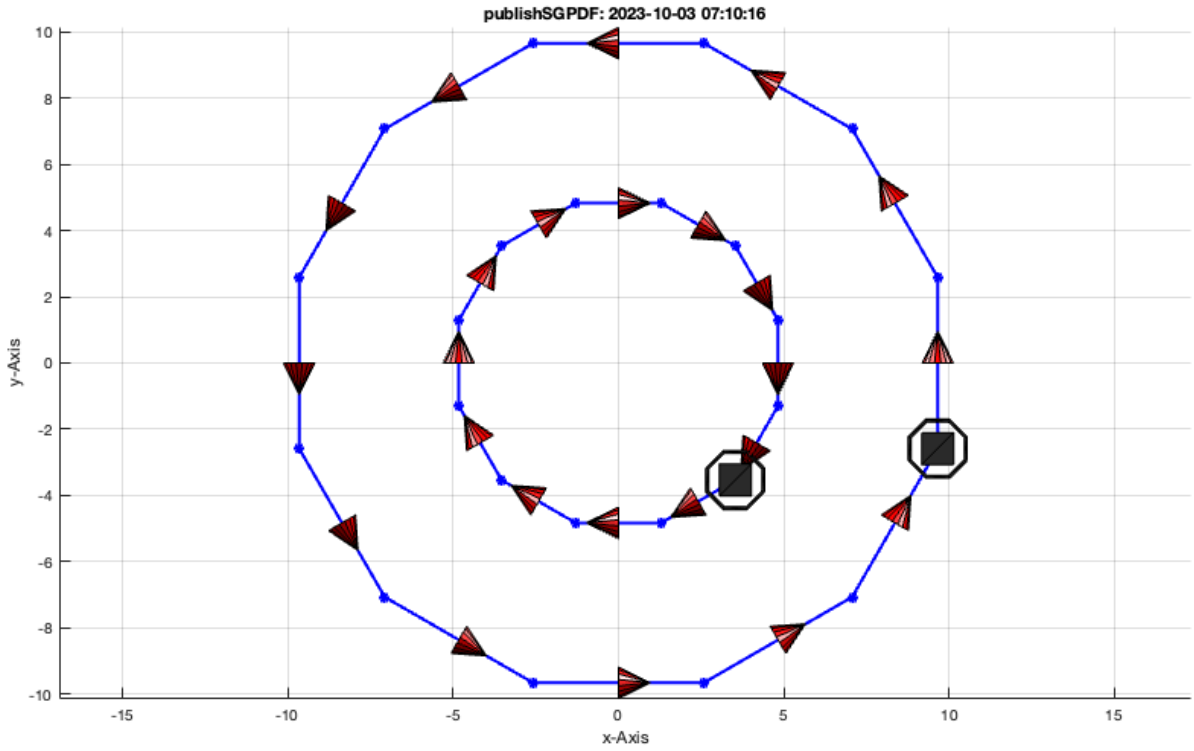
In the next figure, we see both polygons counter-clockwise:

```
PLELofCPL(CPL);
```



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

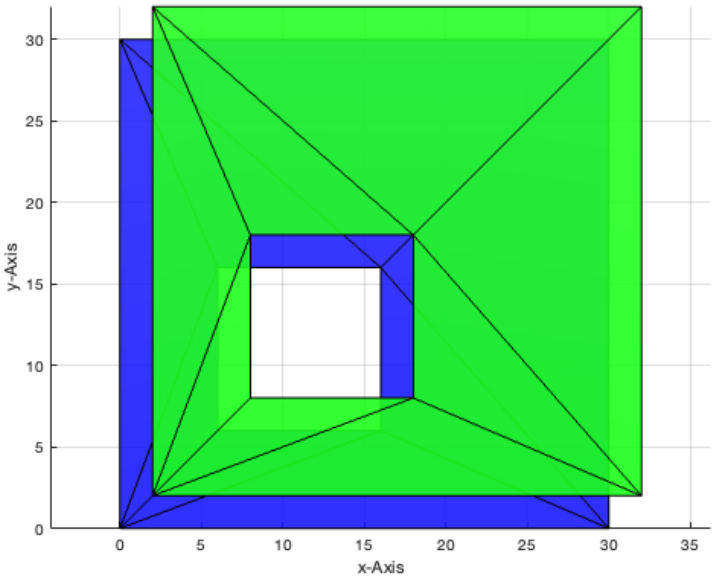
```
close all;
CPL=[PLcircle(10,12);NaN NaN;flip(PLcircle(5,12))];
PLELofCPL(CPL);
```



5. Boolean operations of contour polybool lists (CPL)

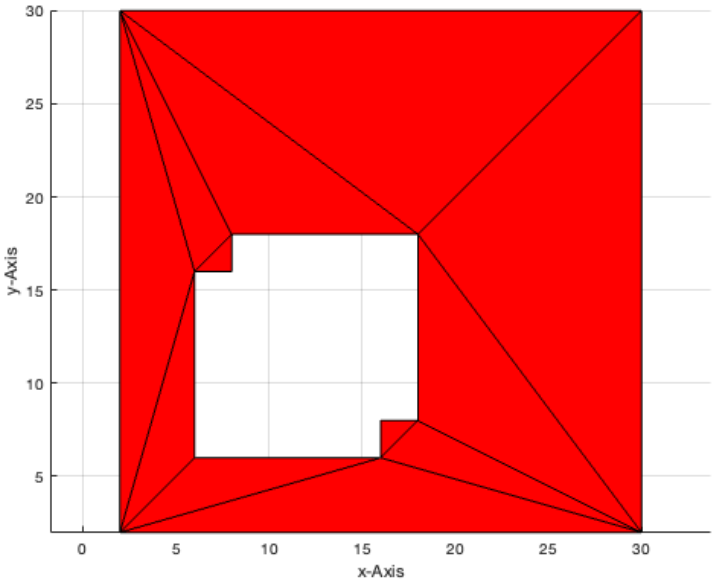
The main advantage of the CPL representation is currently the possibility to use the polybool functions of the mapping toolbox. Here, we show the use in embedded functions that help later to make the step forward to 3D modeling. We start with boolean operations of contour polybool lists (CPL).

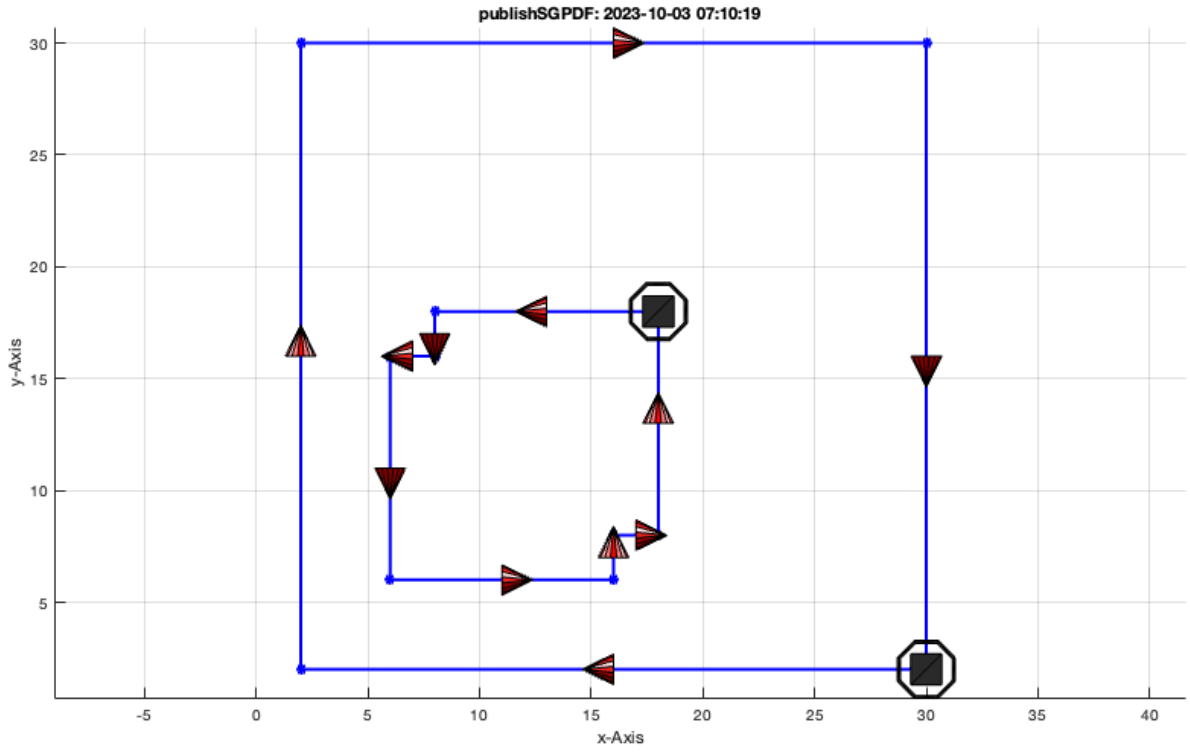
```
close all; figure;
CPL=[PLA*3;NaN NaN;(PLA)+6];
CPLA=CPL;    [PL,FL]=PLFLofCPLpoly(CPLA); VLFLplot(PL,FL,'b');
CPLB=CPL+2;  [PL,FL]=PLFLofCPLpoly(CPLB); VLFLplot(PL,FL,'g');
VLFLplotlight (0,0.9)
```



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

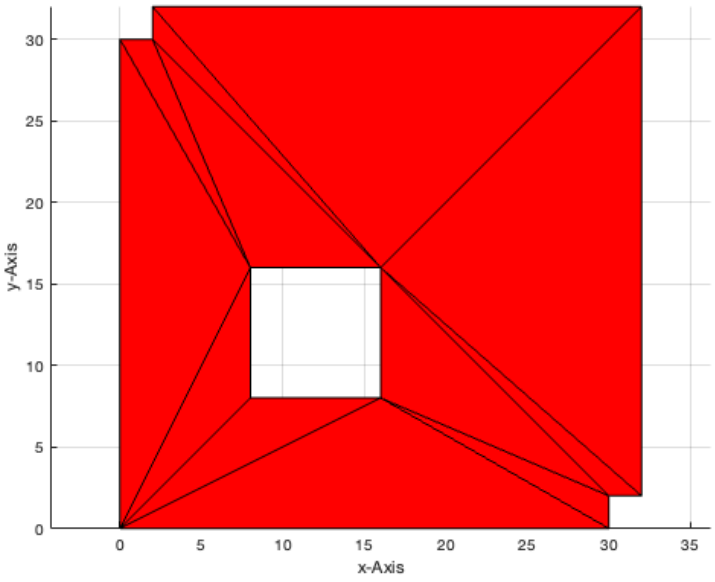
```
close all;
CPLN=CPLpolybool('and',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

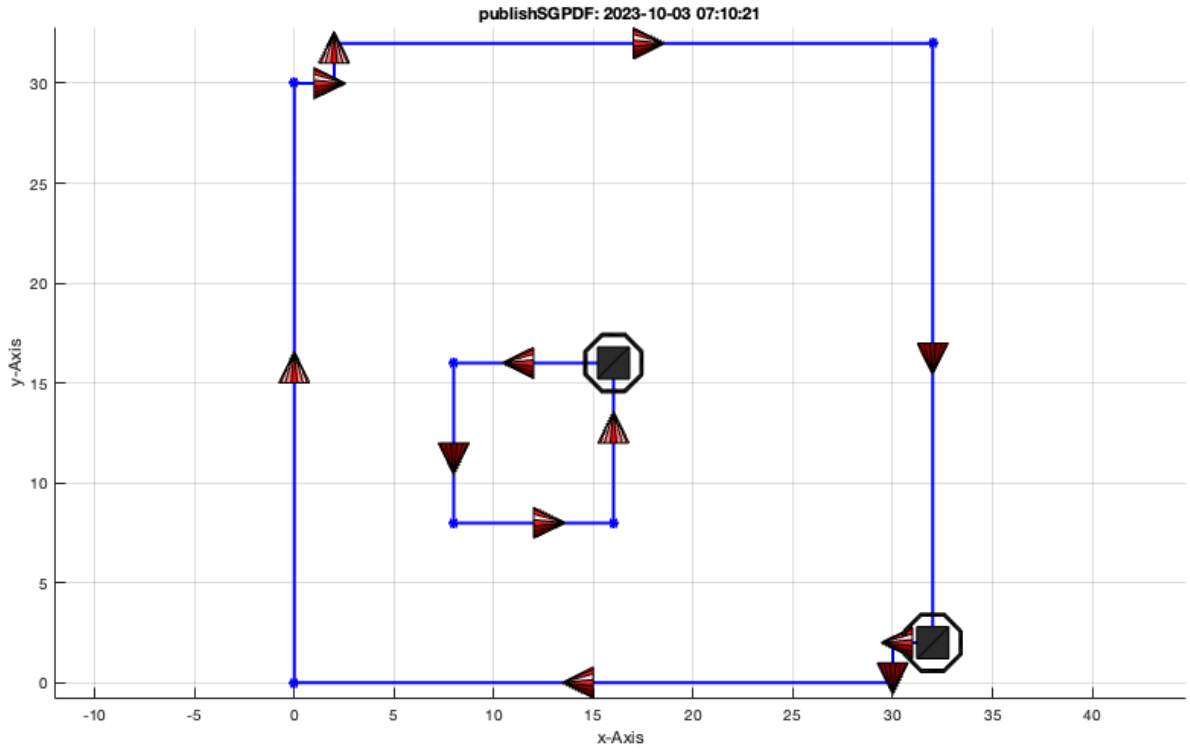




- **CPLpolybool('or',CPLA,CPLB)** delivers CPLA united with CPLB.

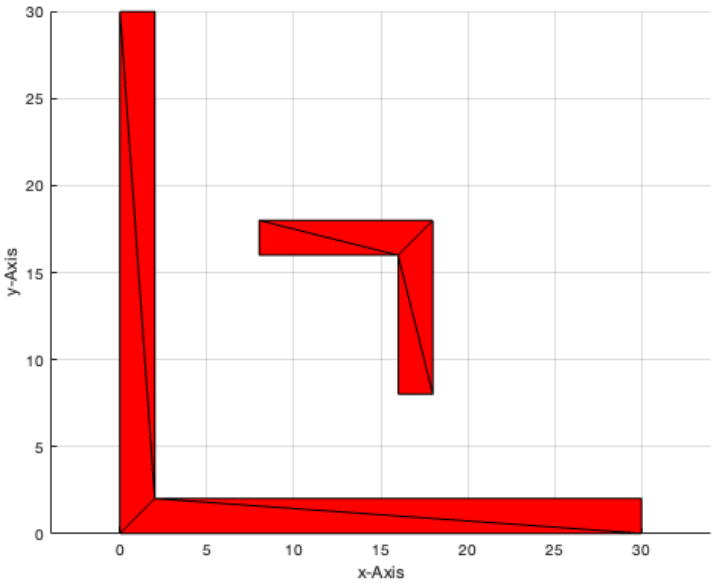
```
close all;
CPLN=CPLpolybool('or',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

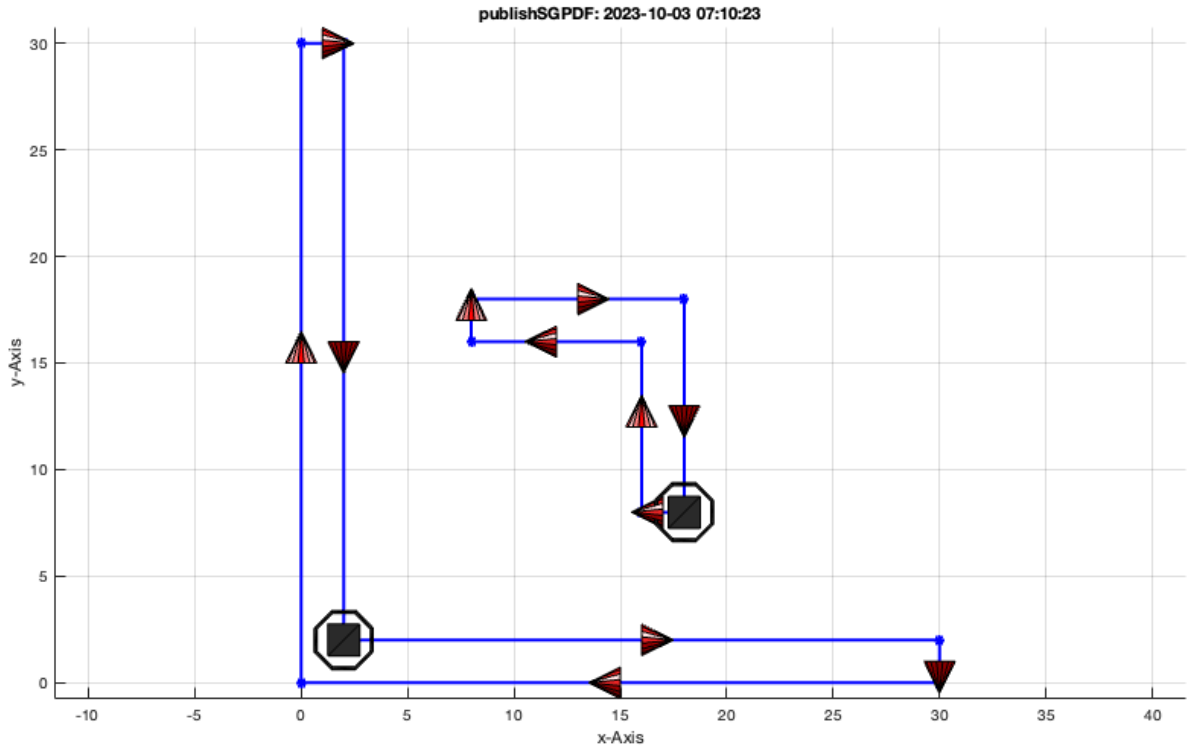




- **CPLpolybool('minus',CPLA,CPLB)** delivers CPLA minus CPLB.

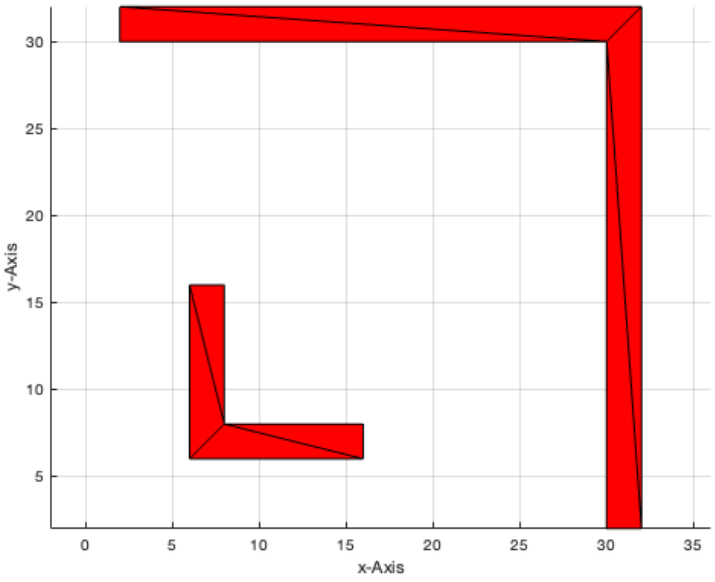
```
close all;
CPLN=CPLpolybool('minus',CPLA,CPLB);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

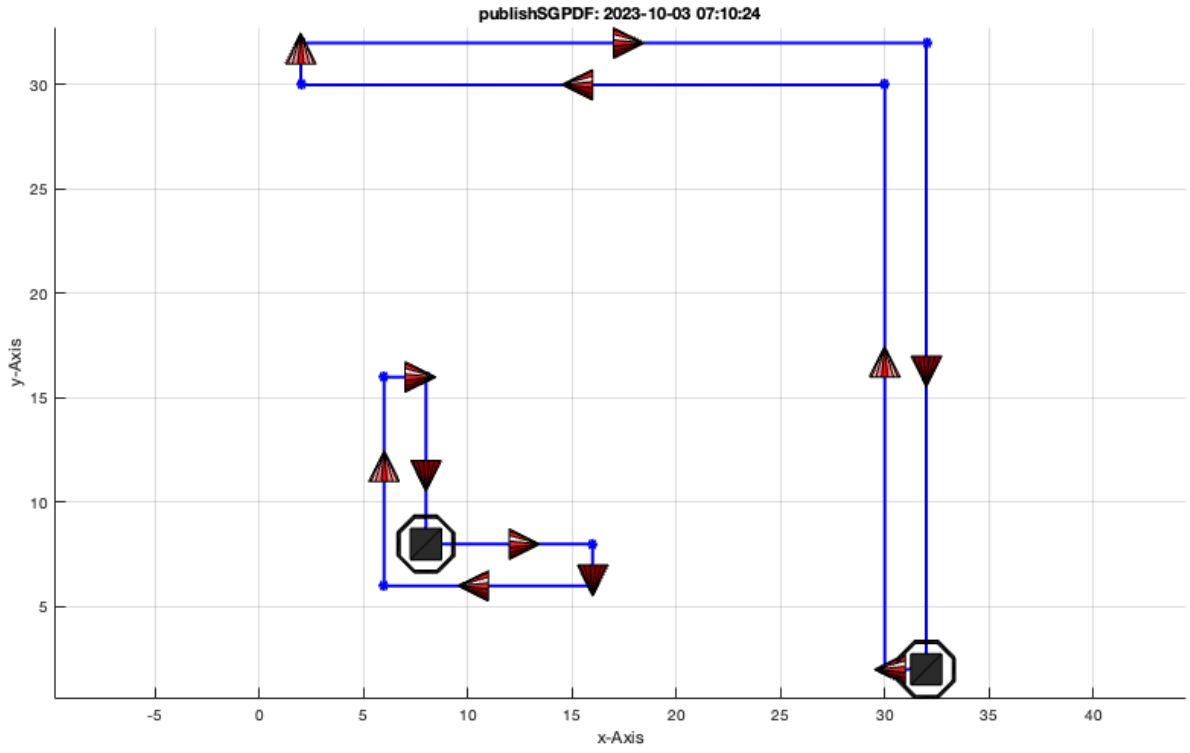




- **CPLpolybool('minus',CPLB,CPLA)** delivers CPLB minus CPLA.

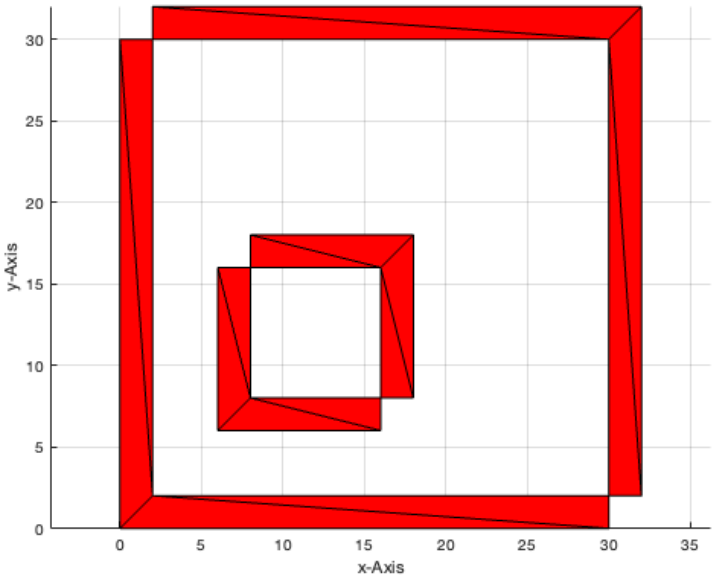
```
close all;
CPLN=CPLpolybool('minus',CPLB,CPLA);
[PL,FL]=PLFLofCPLdelaunay(CPLN); VLFLplot(PL,FL); PLELofCPL(CPLN);
```

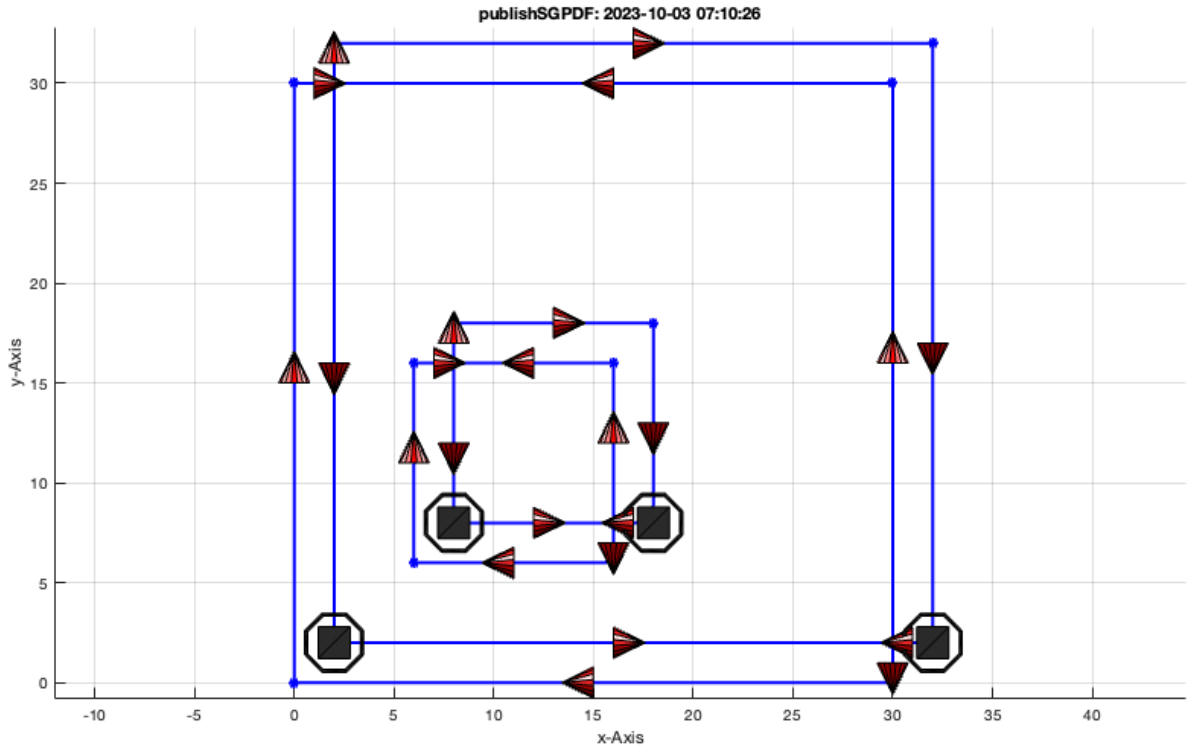




- **CPLpolybool('xor',CPLA,CPLB)** delivers CPLA exclusiveor CPLB.

```
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
```

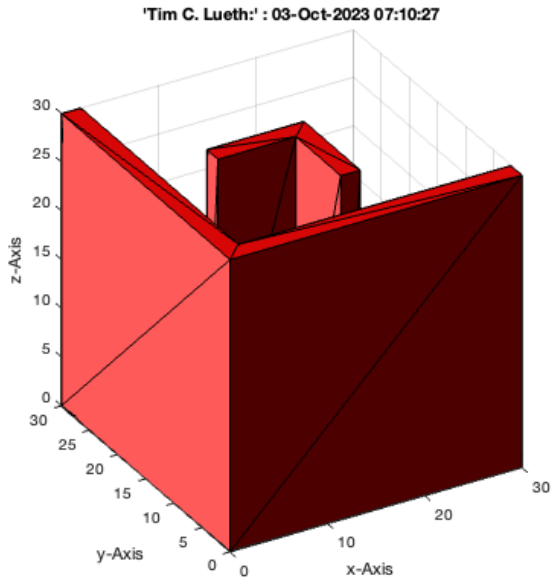
```
     2     3
     3     4
     4     5
     5     6
     6     1
     7     8
     8     9
     9    10
    10    11
    11    12
    12     7
EL =
     7    12
    12    11
    11    10
    10     9
     9     8
     8     7
     1     6
     6     5
     5     4
     4     3
     3     2
     2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

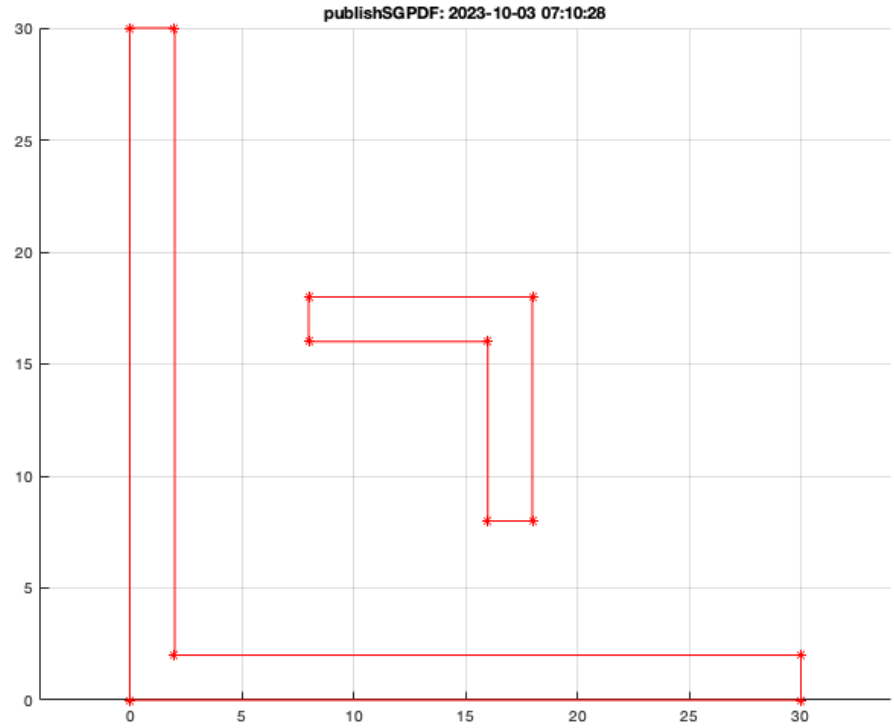
Finally, as it was possible to convert a CPL into and point list and an edge list, there is a function for the opposite.

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

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

```
ans =
    18     8
    18    18
     8    18
```

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



Final remarks on toolbox version and execution date

```
VLFLlicense
```

```
This VLFL-Lib, Rel.  (2023-Oct-03), is for limited non commercial educational use only!
Licensee: Tim Lueth (Development Version)!
Please contact Tim Lueth, Professor at TU Munich, Germany!
WARNING: This VLFL-Lib (Rel. ) license will exceed at 06-Jul-2078 07:10:29!
Executed 03-Oct-2023 07:10:31 by 'timlueth' on a MACI64 using Mac OSX 13.6 | R2023a Update 5 | SG-Lib 5.4
======================================== Used Matlab products: ========================================
database_toolbox
distrib_computing_toolbox
fixed_point_toolbox
image_toolbox
map_toolbox
matlab
optimization_toolbox
pde_toolbox
phased_array_system_toolbox
signal_blocks
signal_toolbox
simmechanics
simscape
simulink
statistics_toolbox
================================================================================================
```

- *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® R2023a