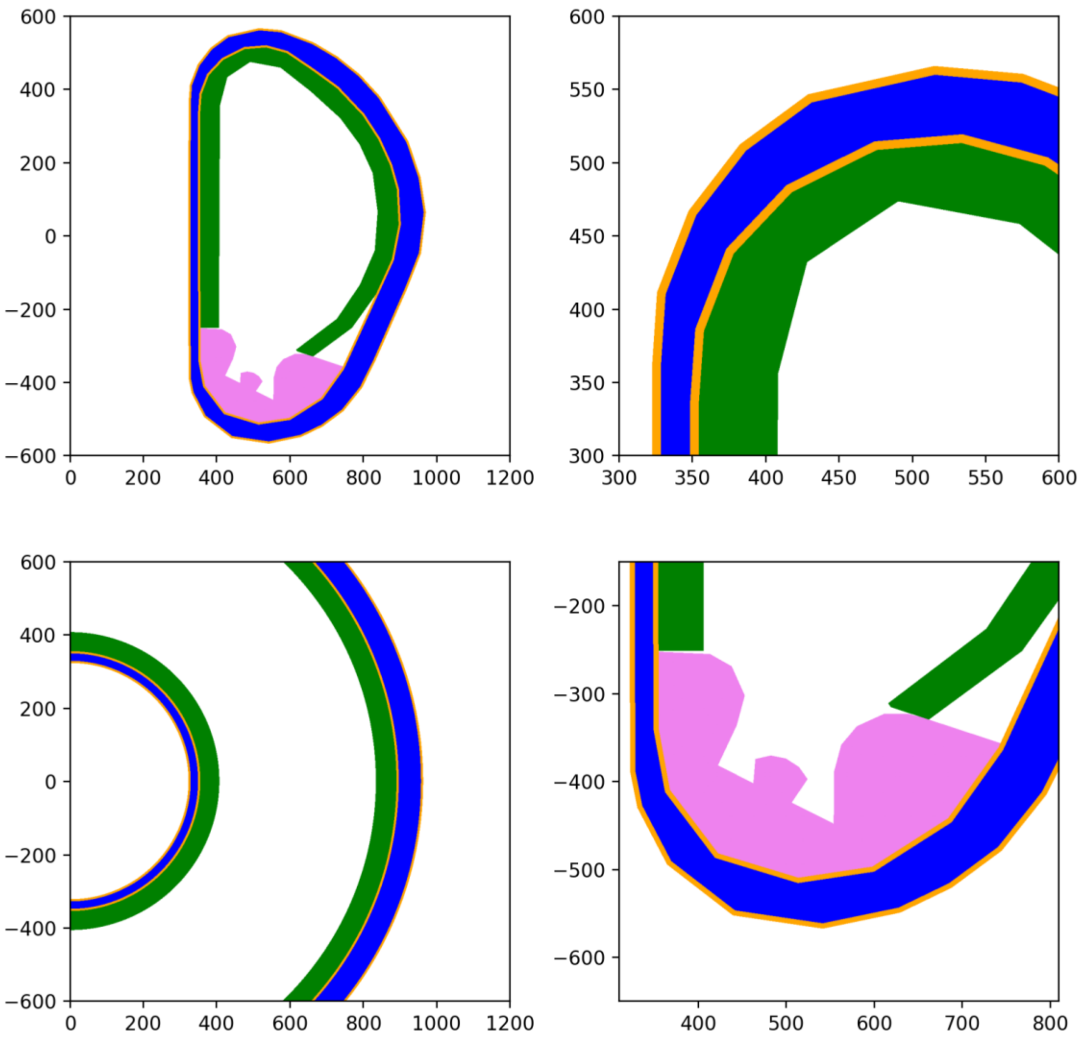
PolygonTorus

A tool for making toroidal CSG bodies

CSG – constructive solid geometry

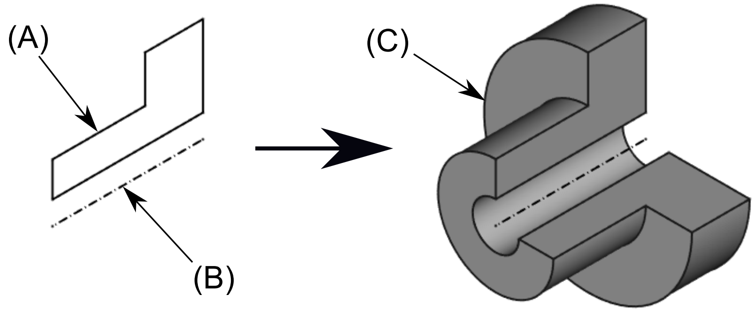


Content

1. Concept & theory
    - Revolve for CSG
    - Cone from 2D line
    - Script structure
2. Using PolygonTorus
3. Simple example
    - Making a shell
4. Future improvements
5. How NCSM was created (shown in Jupyter Notebook)

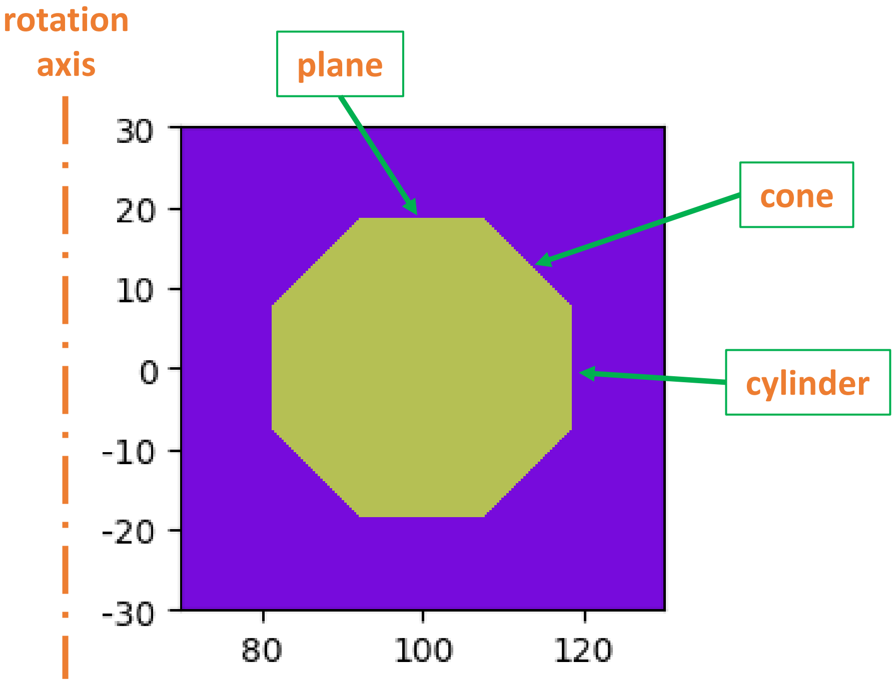
1.1. “Revolution” Command in CSG?

- CAD software “revolution / revolve” command:
    1. make profile
    2. rotate around axis
    3. get a 3D solid



- CSG:
    - Vertical* lines → cylinders
    - Horizontal* lines → planes
    - Angled* lines → cones

*with respect to rotation axis



1.2. Cone From an Angled Line

- Given points A, B we can calculate the line slope k and intersection y_0
- From those we can construct a z-axis cone:

$$(x - x_0)^2 + (y - y_0)^2 = t^2(z - z_0)^2$$

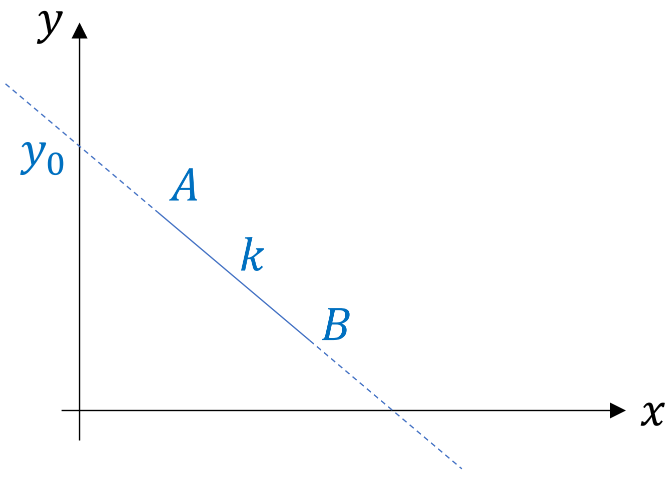
- If $(x_0, y_0) = (0,0)$ we get:

$$r^2 = t^2(z - z_0)^2$$

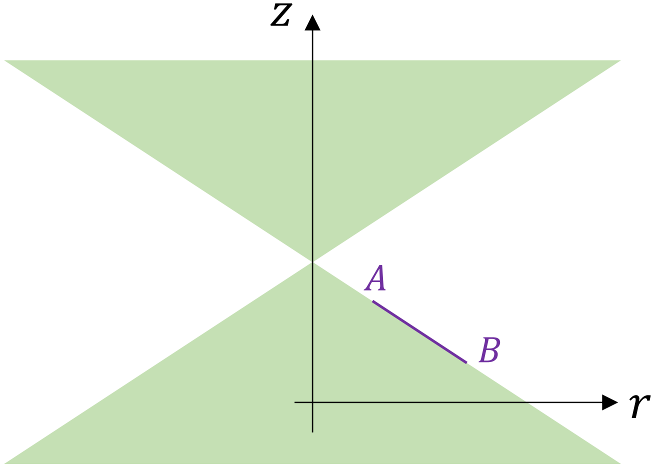
$$\Rightarrow z(r) = \pm\frac{1}{t}r + z_0$$

- Line equation: $y(x) = kx + y_0$

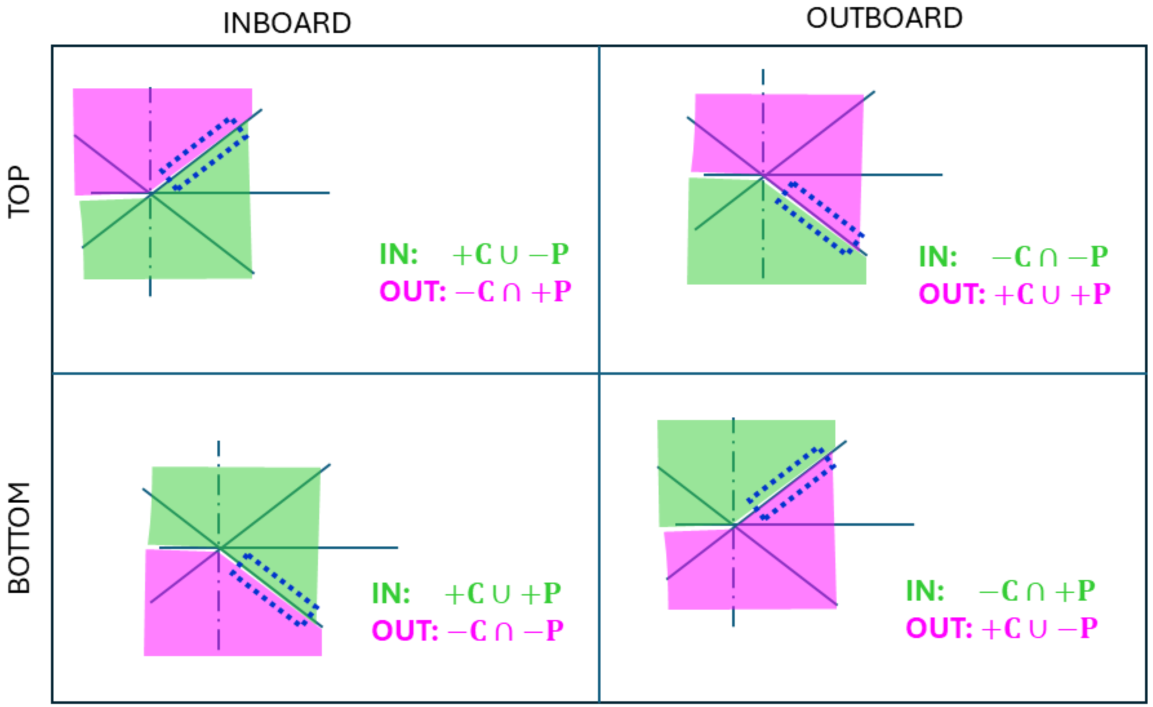
$$\boxed{\begin{aligned} t &= \frac{1}{k} \\ z_0 &= y_0 \end{aligned}}$$



$$y(x) = kx + y_0$$

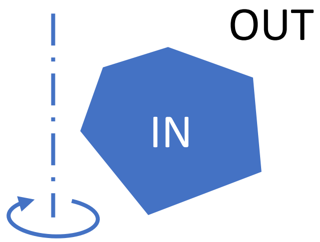


1.3. Regions from cones (C) and planes (P)

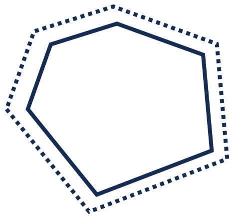


1.4. Script Structure: Object Layers

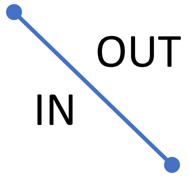
1. PolygonTorus(points)
- Makes *Polygon*
- Generates region in & out



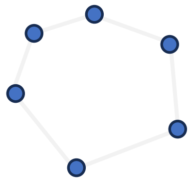
2. Polygon(points)
- Makes *Cycle* of *Points*
- Makes *Edges*
- Functions:
  - offset
  - remove non-convex



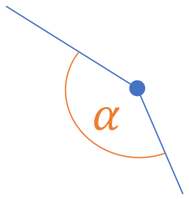
3.a Edge(point1, point2)
- Determines type of edge:
  - Inboard, Outboard, Top, Bottom, TI, TO, BI, BO
- Calculates region inside & outside edge



3.b Cycle(points)
- Makes cyclic list of *Points*



4. Point(xy)
- Has inside angle information



2. Using PolygonTorus

1. Provide list of points: *points = [[x1,y1], [x2,y2], ...]*
   - Points must follow clockwise orientation
   - All concave or in-line points will be removed automatically
   - First and last point will be connected
2. Generate PolygonTorus: *PT = PolygonTorus(points)*
3. Extract regions: *PT.region_in* and *PT.region_out*
   - Use for further model building

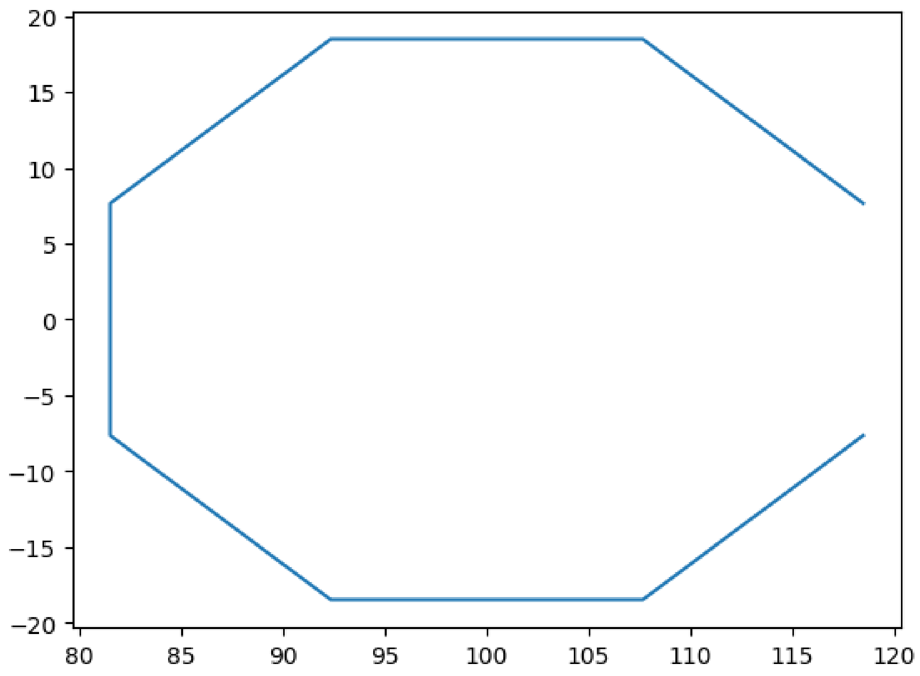
- Optional: Make offset shells:
  - Inward shell ($d < 0$): May result in no region if $|d|$ too thick
  - Outward shell: Up to z-axis!
- Complex geometries
  - Combine different PT regions with union or intersection operations

2.1. Using WebPlotDigitizer for point creation

- Tool for extracting data from images
- GitHub
- Online version
- How to use:
    1. Upload photo
    2. Align axes
    3. Make collection of points
    4. Export data to file or copy to script

3. Simple Example: Octagon Torus

```
points = array([[118.47759065,  -7.65366865],
       [107.65366865, -18.47759065],
       [ 92.34633135, -18.47759065],
       [ 81.52240935,  -7.65366865],
       [ 81.52240935,   7.65366865],
       [ 92.34633135,  18.47759065],
       [107.65366865,  18.47759065],
       [118.47759065,   7.65366865]])
[<matplotlib.lines.Line2D at 0x7fdbc65d00d0>]
```




```
PT = pt.PolygonTorus(points)
```

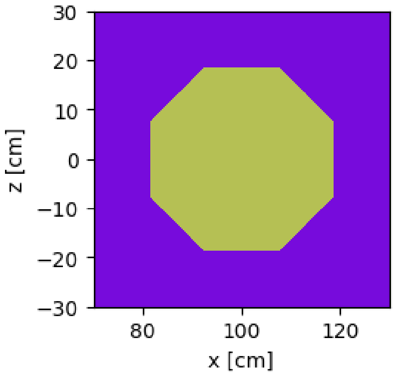
```
print(PT.region_in)
print(PT.region_out)
```

```
((-1 2) 3 (4 | 5) 6 (7 | -8) (9 | -10) (-11 -12) (-13 -14))
((1 | -2) | -3 | (-4 -5) | -6 | (-7 8) | (-9 10) | (11 | 12) | (13 | 14))
```

```
cell_inside = openmc.Cell(region=PT.region_in)
cell_outside = openmc.Cell(region=PT.region_out)
universe = openmc.Universe( cells=[cell_inside, cell_outside] )
print(universe)
```

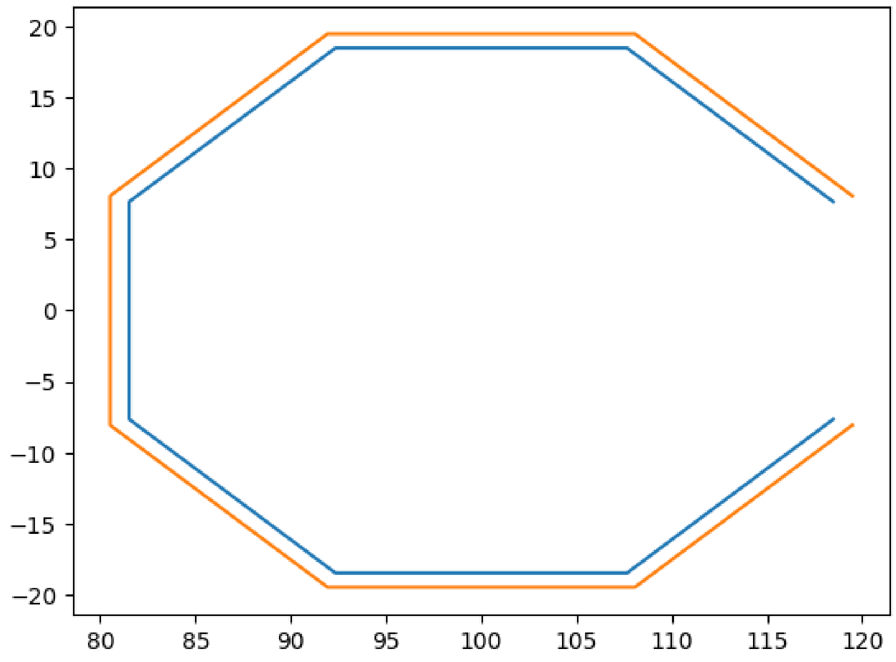
```
Universe
        ID              =       1
        Name            =
        Geom            =       CSG
        Cells           =       [1, 2]
```

```
universe.plot(origin = (r0, 0, z0), basis='xz', width=(3*minor_radius, 3*minor_radius))
plt.savefig('1_simple_octagon_torus.png')
```

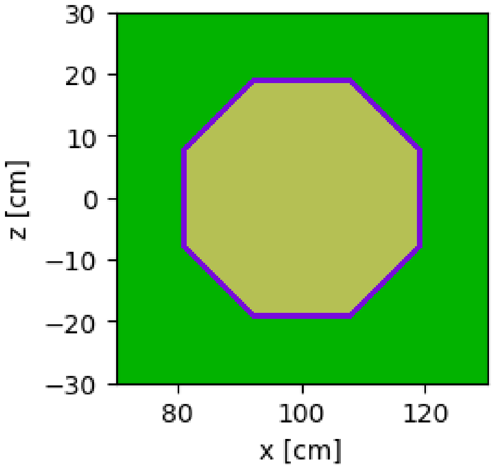


3.1. Simple Example: Making a Shell

```
PT_offset = PT.Offset(d=1)
PT.polygon.PlotPolygon(mpl_axes=plt)
PT_offset.polygon.PlotPolygon(mpl_axes=plt)
```




```
inside = openmc.Cell(region=PT.region_in)
shell = openmc.Cell(region= PT.region_out & PT_offset.region_in)
outside = openmc.Cell(region= PT_offset.region_out)
universe = openmc.Universe( cells=[inside, shell, outside] )
universe.plot(origin = (r0, 0, z0), basis='xz', width=(3*minor_radius, 3*minor_radius))
plt.savefig('1_shell.png')
```



4. Future Improvements

- Translate Polygon in r-z plane
- Rotate polygon around point in r-z plane
- Usage of torus surface → smooth curvatures
  - Can be used to read .DXF files (directly from CAD)
- Automatic correction of CCW to CW
- Segmentation at concave point → more complex shapes
- Auto correction of self-crossing path