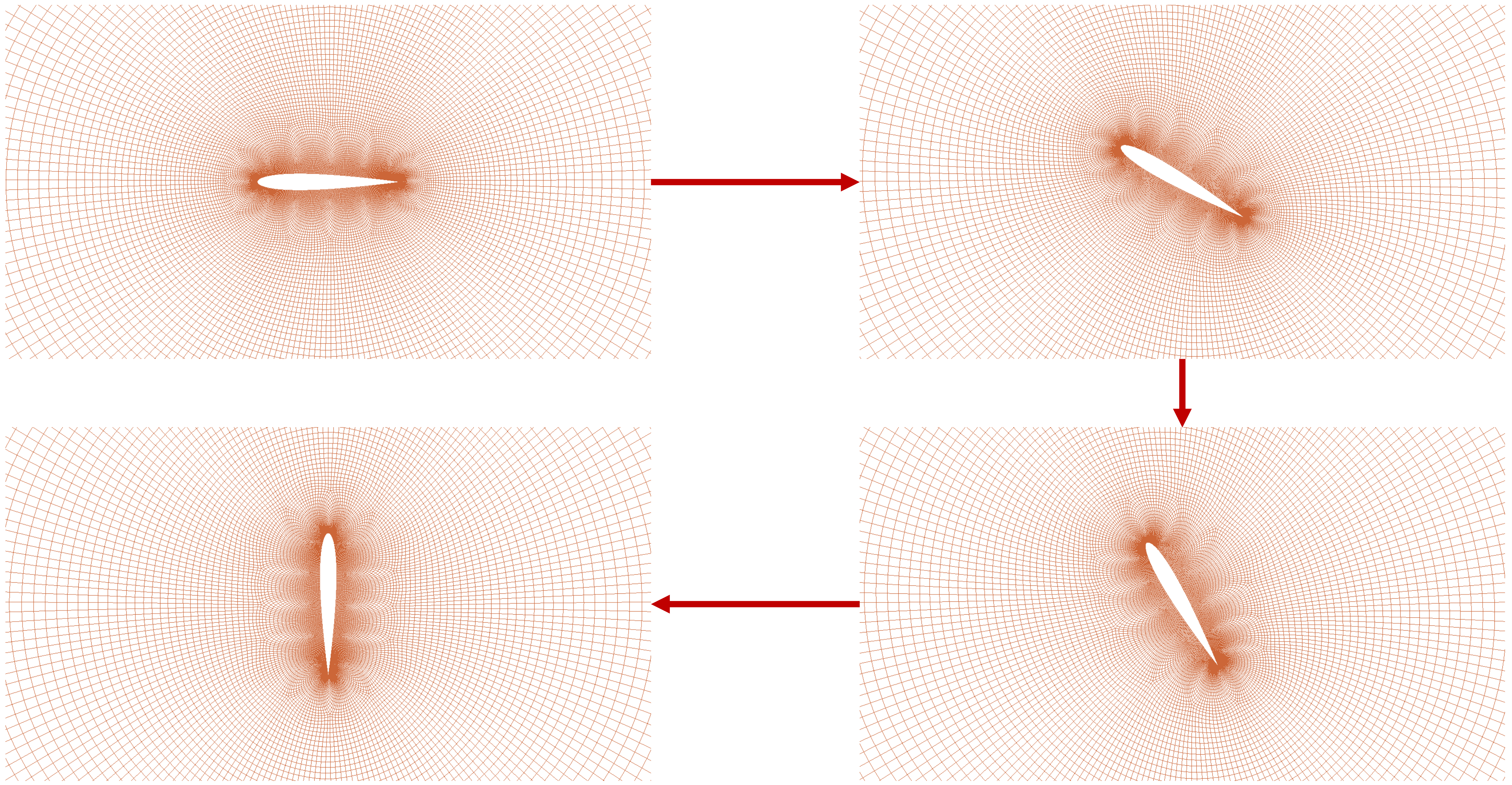


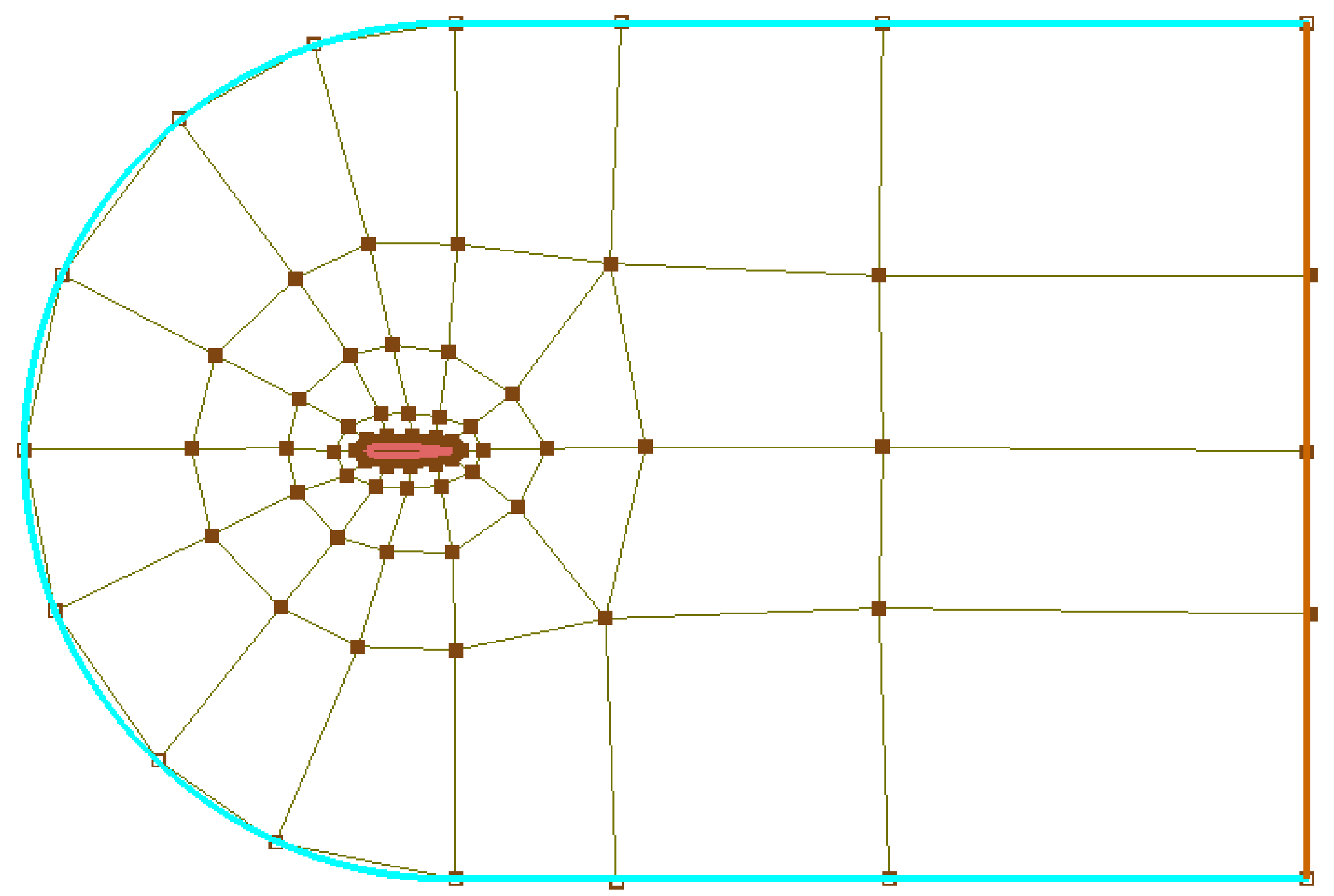
Grid*Pro API – Transforming Topology and Rebuilding Blocks*

.

API Tutorial 6

Scope of the tutorial

- In this tutorial, we will discuss about how to generate an Airfoil grid for different angles.
- Initially a template topology is generated when the airfoil is at zero angle.
- The airfoil and its corresponding topology would be rotated for the desired angle to generate the grid.
- Rigid body rotation of the topology would only work upto a certain range of angles, beyond which it will cause skew in the grid. So we would be detaching the topology closer to the airfoil and relinking the blocks with more suitable match.



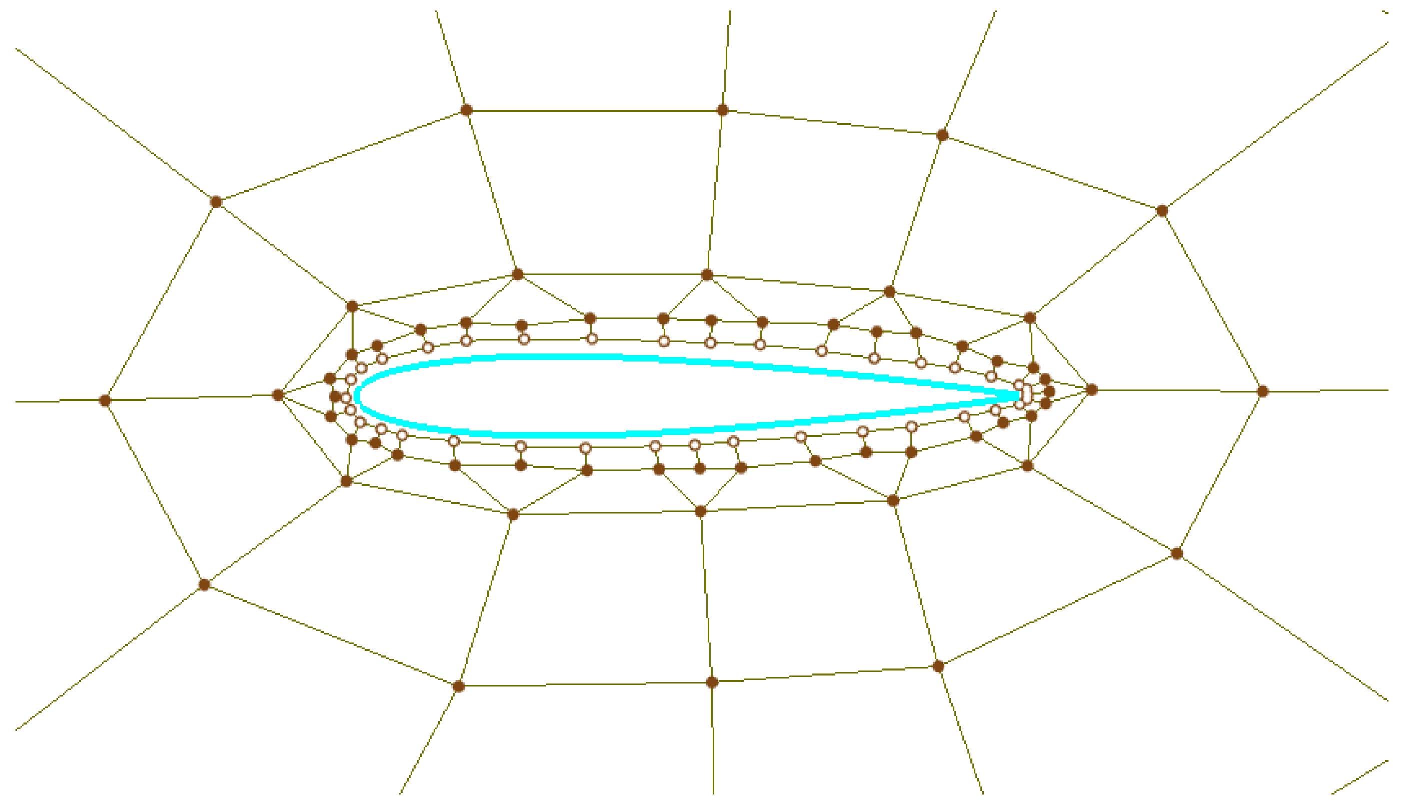
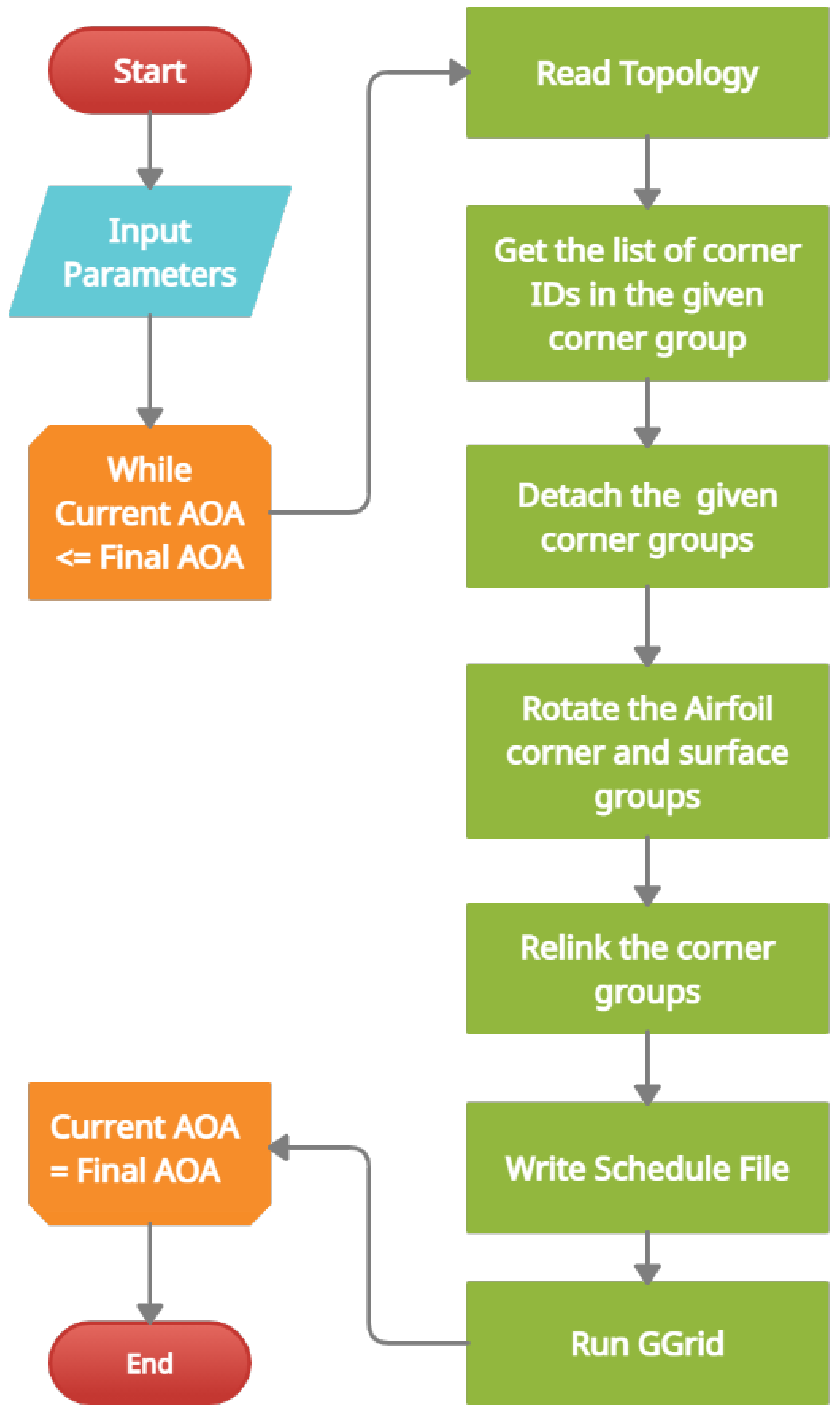


Figure 1: Airfoil Topology @ Zero AOA

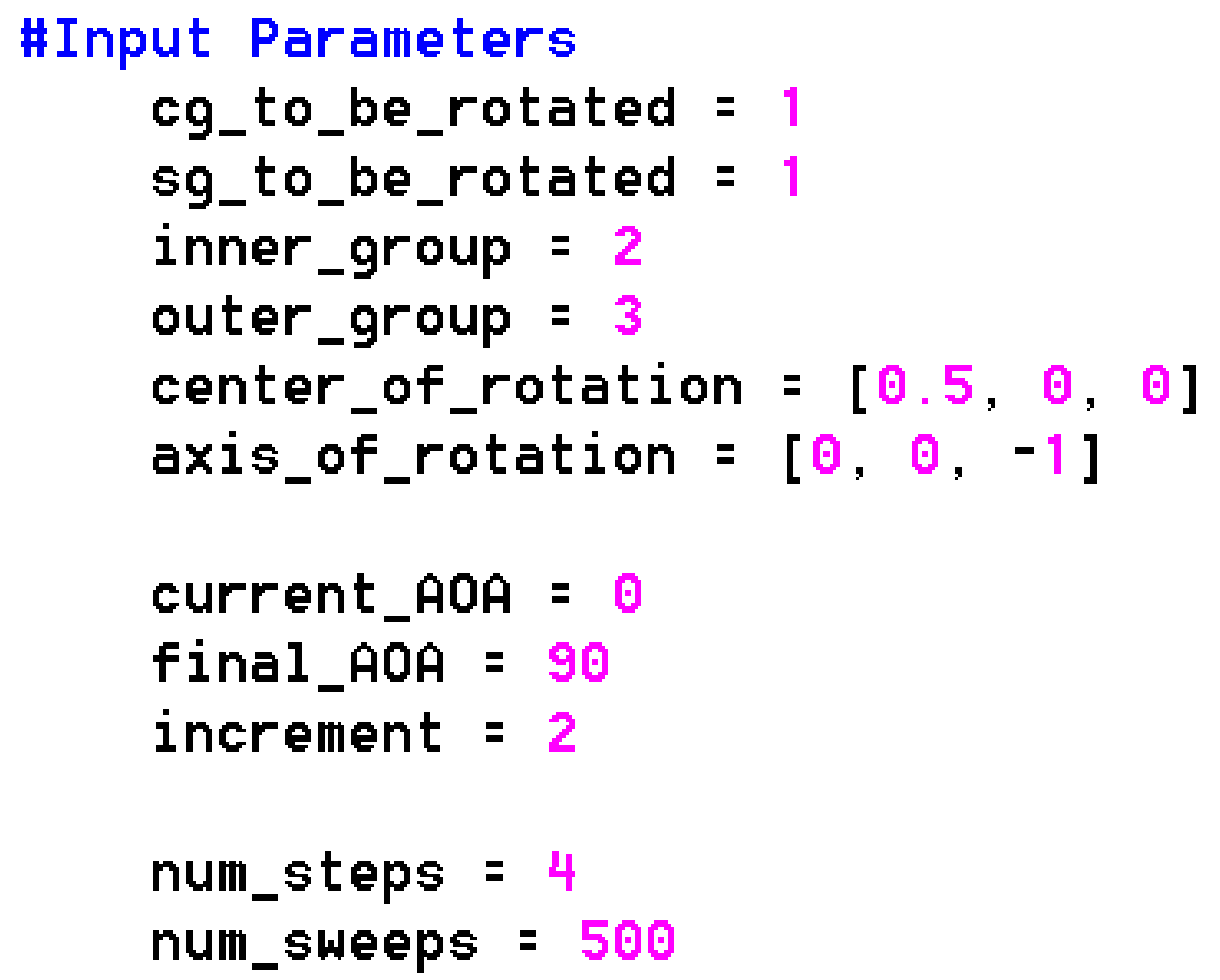
Script Workflow

- The work flow of the script goes as follows:
  1. Get the required input parameters.
  2. Read the topology.
  3. Detach the airfoil corner group from the farfield.
  4. Rotate the airfoil and the corner group to the desired AOA.
  5. Re-Link the corner group to form valid topology.
  6. Write the schedule file as required using the input parameters.
  7. Run ggrid on the topology file.
  8. Repeat the steps until the grids for all the desired AOAs are generated.
  9. End.


Start
Input Parameters
While Current AOA <= Final AOA
Read Topology
Get the list of corner IDs in the given corner group
Detach the given corner groups
Rotate the Airfoil corner and surface groups
Relink the corner groups
Write Schedule File
Run GGrid
Current AOA = Final AOA
End

Input Parameters

- Corner and surface groups to be rotated along with the center and axis of rotation.
- Corner groups between which the links have to be detached.
- Current and final angles with the desired rate of increment.
- For the schedule file, we need to input the number of steps and number of sweeps per step to run the grid generator.



```
#Input Parameters
    cg_to_be_rotated = 1
    sg_to_be_rotated = 1
    inner_group = 2
    outer_group = 3
    center_of_rotation = [0.5, 0, 0]
    axis_of_rotation = [0, 0, -1]

    current_AOA = 0
    final_AOA = 90
    increment = 2

    num_steps = 4
    num_sweeps = 500
```

Figure 2: Input Parameters

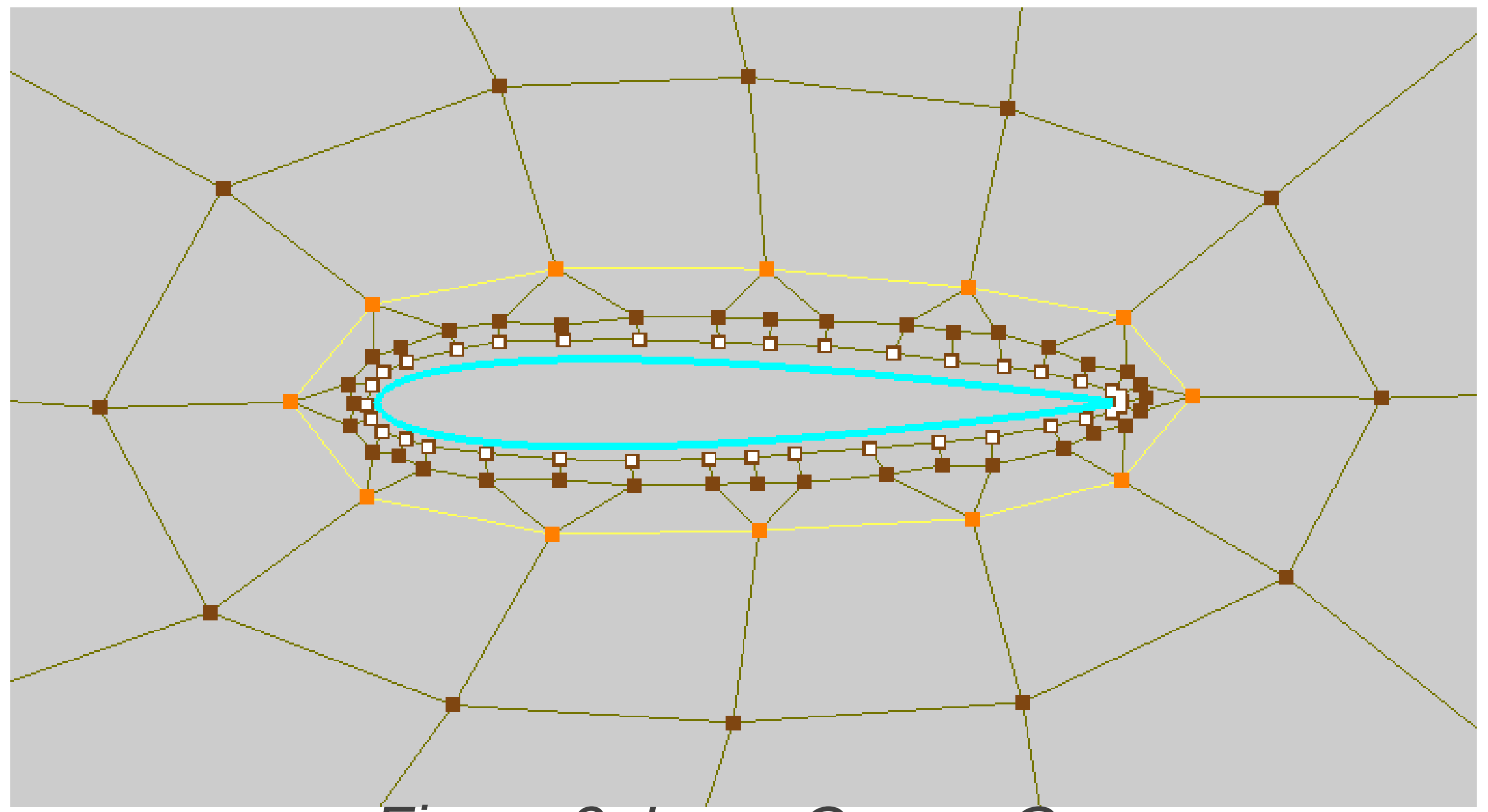


Figure 3: Inner Corner Group

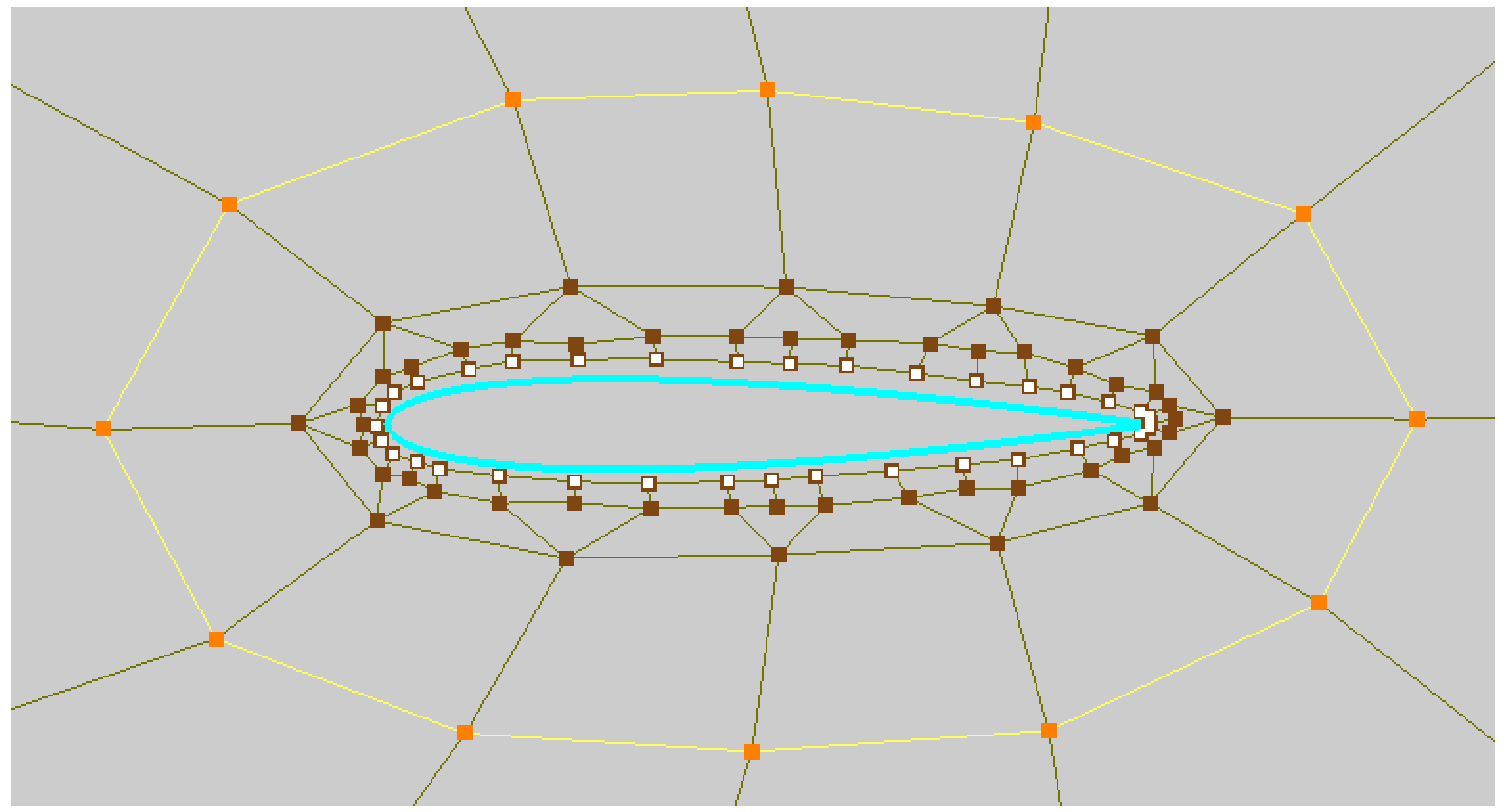


Figure 4: Outer Corner Group

Code Snippet – Topology_Transformation.py

```python
# IMPORT OPERATIONS
import os
import gp_utilities
#Find the List of corner IDs in a given corner group
def corner_group_to_id_list(dimension, corner_group):
   id_list = []
   corners = topo.corner_group(corner_group).get_all()
   num_corners = topo.get_num_corners_in_corner_group(corner_group)
   print num_corners
   if dimension == 2:
    for number in range(num_corners/2):
     id_list.append(corners[number].get_id())
   else:
    for number in range(num_corners):
     id_list.append(corners[number].get_id())
   return id_list

#Main Function
if(__name__ == '__main__'):

   topo = gp_utilities.Topology()
#Input Parameters
   cg_to_be_rotated = 1
   sg_to_be_rotated = 1
   inner_group = 2
   outer_group = 3
   center_of_rotation = [0.5, 0, 0]
   axis_of_rotation = [0, 0, -1]
   current_AOA = 0
   final_AOA = 90
   increment = 2
   num_steps = 4
   num_sweeps = 500
```

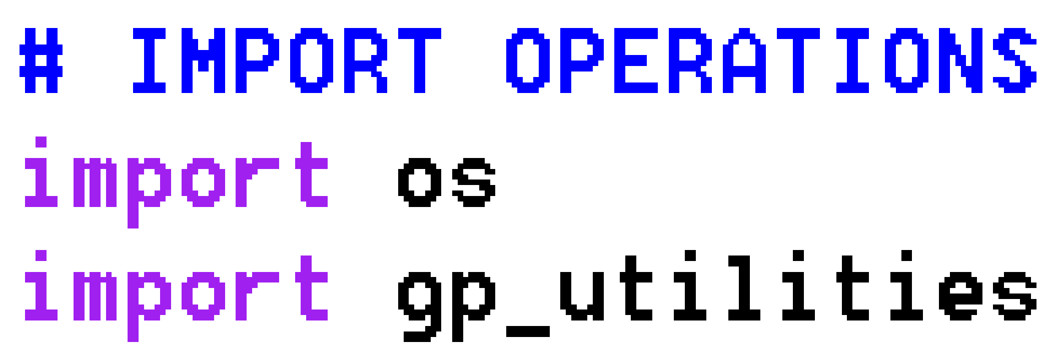
```python
#Transform Topology
while current_AOA <= final_AOA:
   topo.read_topo_file("topology.fra")
   inner_list = corner_group_to_id_list(2, inner_group)
   outer_list = corner_group_to_id_list(2, outer_group)

   #Detach Corner Groups
   for corners in inner_list:
    linked_corners_inner_list = topo.corner(corners).get_links()
    for links in range(len(linked_corners_inner_list)):
     linked_id = linked_corners_inner_list[links].get_id()
      if linked_id in outer_list:
       topo.unlink([(topo.corner(corners), topo.corner(linked_id))])

   topo.execute("transform_topo -g {} -sg {} -ax {} {} {} {} {} {} -a {}"
               .format(cg_to_be_rotated, sg_to_be_rotated,
               center_of_rotation[0],center_of_rotation[1],
               center_of_rotation[2], axis_of_rotation[0],
               axis_of_rotation[1], axis_of_rotation[2], current_AOA))

   cg = topo.get_corner_group(4)
   cg.append(topo.corner_group(inner_group))
   cg.append(topo.corner_group(outer_group))
   cg.rlink()
   topo.write_topology("topology.@{}.fra".format(current_AOA))

   #Run Ggrid
   topo.write_schedule_file("topology.@{}.sch".format(current_AOA),
                num_steps, num_sweeps,
   "grid.@{}.grd".format(current_AOA),       "dump.tmp")
   Ggrid = "Ggrid topology.@{}.fra".format(current_AOA)
   os.system(Ggrid)

   current_AOA = current_AOA + increment
```

API Module

- Import the following libraries:

    1. gp_utilities = GridPro's API

    2. os = To run system command



```
# IMPORT OPERATIONS
import os
import gp_utilities
```

API Module

- Function: To convert the corners in a given corner group to list of corner ids.
  1. Get the dimension of the topology and corner group to be converted.
  2. Loop through all the corners in the corner group and find it's corner id.
  3. Append the corner ids to a list.

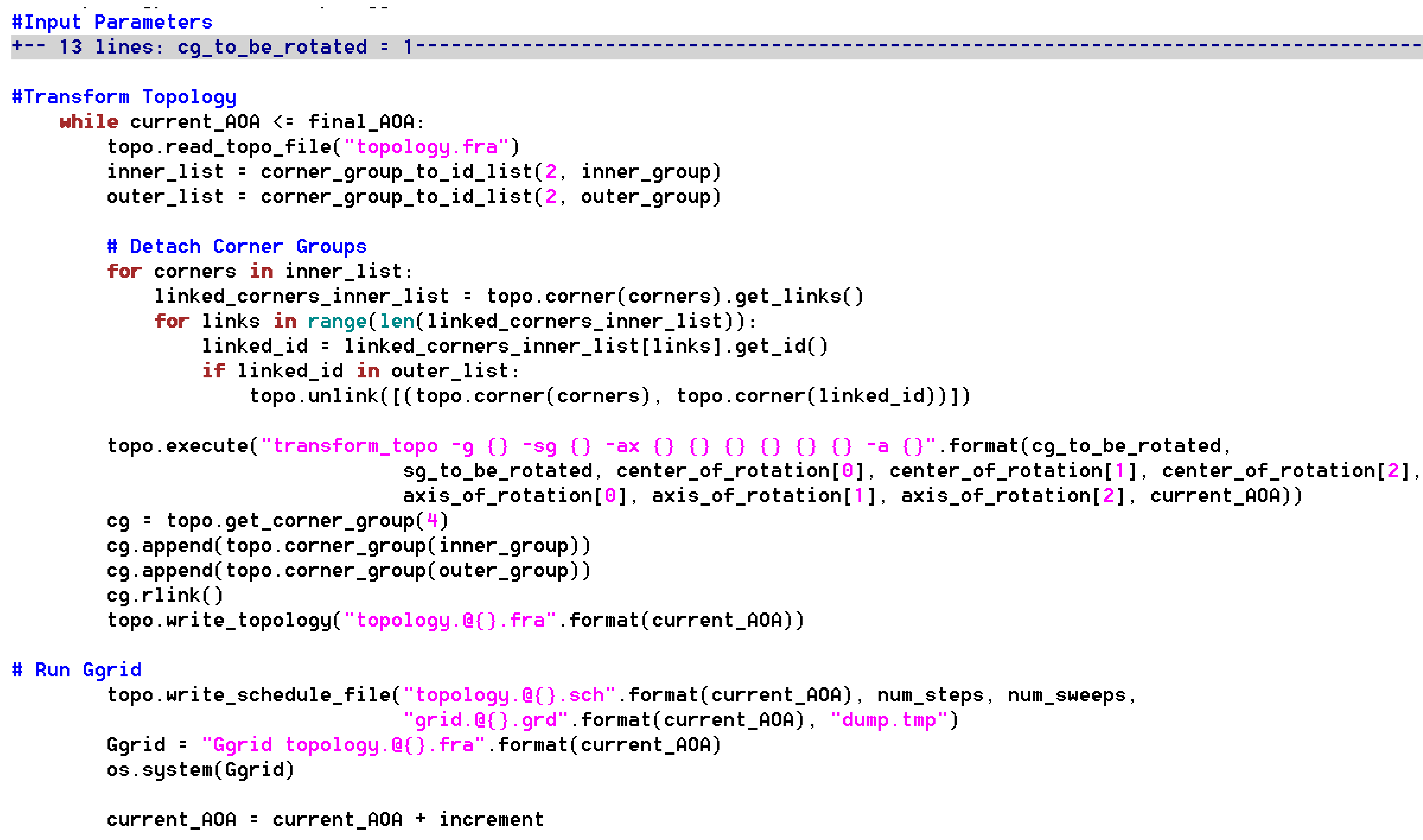
Note: If the dimension of the topology is 2, the number of corners would be twice the actual number. Because in GridPro, for every corner in 2D, there will be a duplicate corner @Z=1 unit.

```
#Find the List of corner IDs in a given corner group
def corner_group_to_id_list(dimension, corner_group):
    id_list = []
    corners = topo.corner_group(corner_group).get_all()
    num_corners = topo.get_num_corners_in_corner_group(corner_group)
    print num_corners
    if dimension == 2:
        for number in range(num_corners/2):
            id_list.append(corners[number].get_id())
    else:
        for number in range(num_corners):
            id_list.append(corners[number].get_id())
    return id_list
```

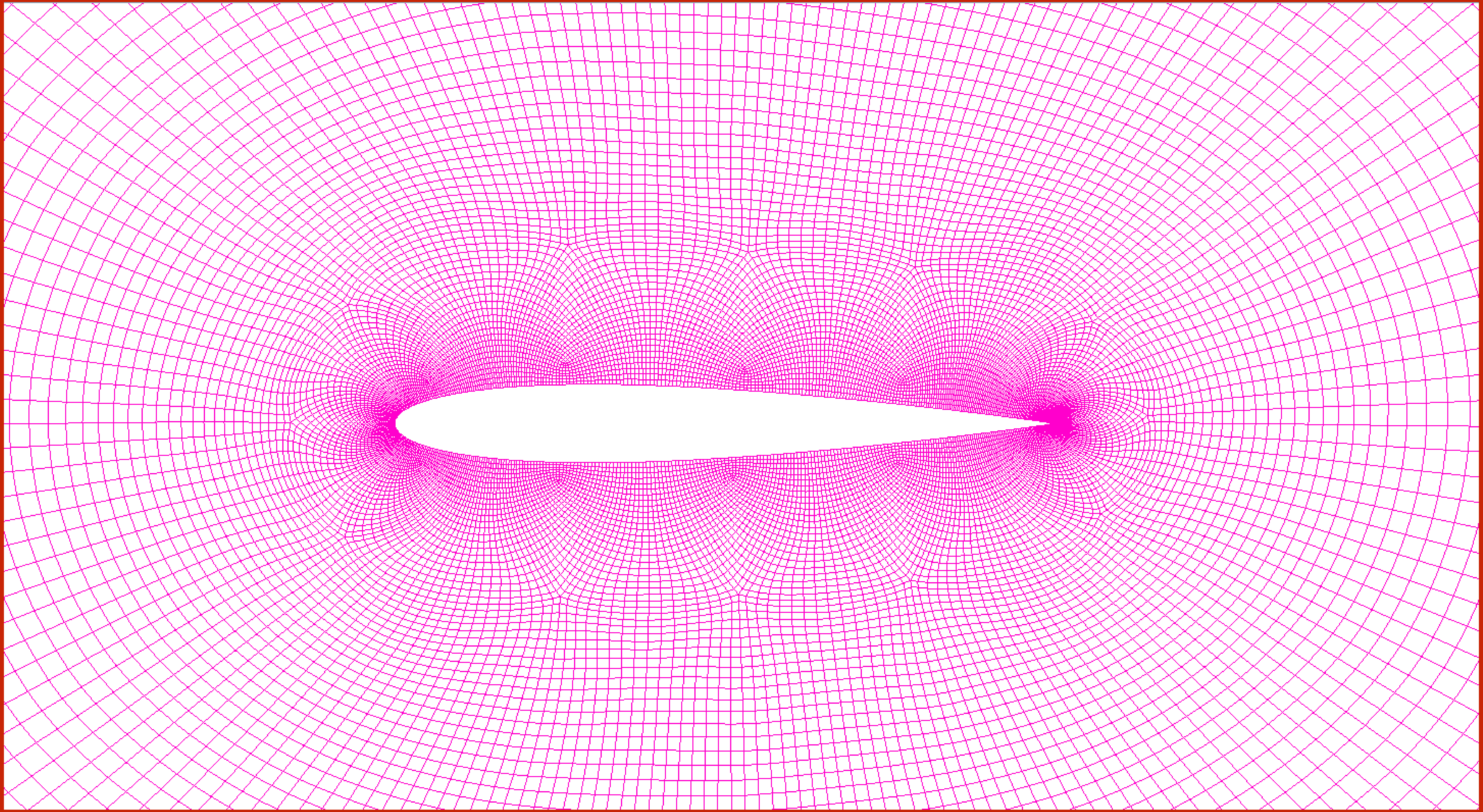
API Module

- Main Function:

  1. Collect the input parameters.
  2. Convert the corners in corner group to list of corner ids.
  3. Detach the topology at the given corner groups.
  4. Rotate the corner and surface group to the current AOA.
  5. Write the schedule file and run the grid generator.
  6. Repeat the steps until the grid is generated for the desired AOA(Final).



```
#Input Parameters
+-- 13 lines: cg_to_be_rotated = 1------------------------------------------------------------------------------------

#Transform Topology
    while current_AOA <= final_AOA:
        topo.read_topo_file("topology.fra")
        inner_list = corner_group_to_id_list(2, inner_group)
        outer_list = corner_group_to_id_list(2, outer_group)

        # Detach Corner Groups
        for corners in inner_list:
            linked_corners_inner_list = topo.corner(corners).get_links()
            for links in range(len(linked_corners_inner_list)):
                linked_id = linked_corners_inner_list[links].get_id()
                if linked_id in outer_list:
                    topo.unlink([(topo.corner(corners), topo.corner(linked_id))])

        topo.execute("transform_topo -g {} -sg {} -ax {} {} {} {} {} {} -a {}".format(cg_to_be_rotated,
                             sg_to_be_rotated, center_of_rotation[0], center_of_rotation[1], center_of_rotation[2],
                             axis_of_rotation[0], axis_of_rotation[1], axis_of_rotation[2], current_AOA))
        cg = topo.get_corner_group(4)
        cg.append(topo.corner_group(inner_group))
        cg.append(topo.corner_group(outer_group))
        cg.rlink()
        topo.write_topology("topology.@{}.fra".format(current_AOA))

# Run Ggrid
        topo.write_schedule_file("topology.@{}.sch".format(current_AOA), num_steps, num_sweeps,
                             "grid.@{}.grd".format(current_AOA), "dump.tmp")
        Ggrid = "Ggrid topology.@{}.fra".format(current_AOA)
        os.system(Ggrid)

        current_AOA = current_AOA + increment
```



End of the Tutorial