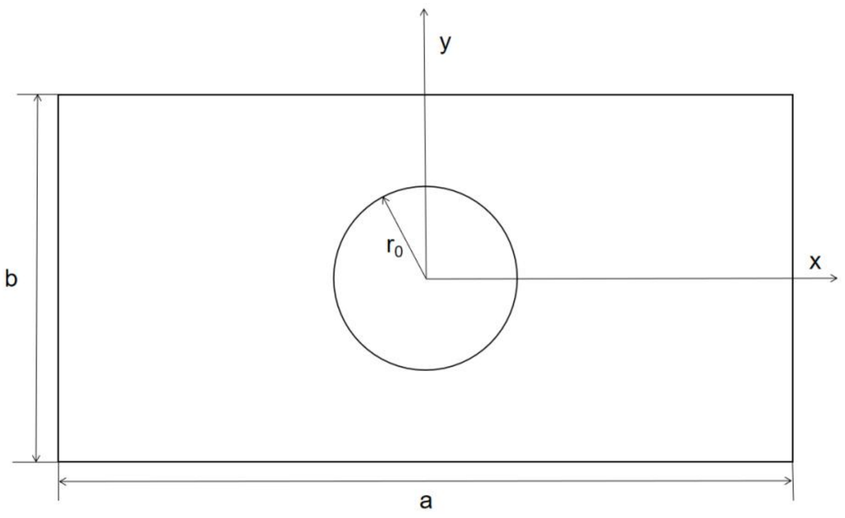
Usage Description of FEM Encapsulation Function

The Python version must be python3.8.

The following modules need to be installed：
meshpy numpy matplotlib



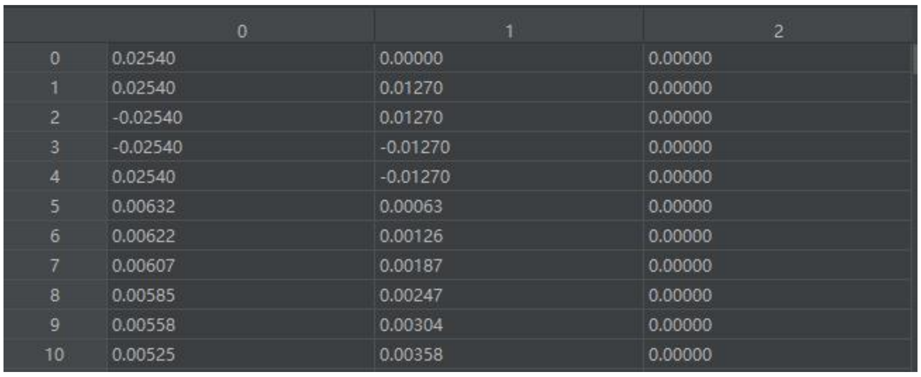
1、Function of meshing :create_mesh

Position
Mesher.pyd

Input
create_mesh(xmin, xmax, ymin, ymax, h_radius=0.0, holex=0.0, holey=0.0, maxpro=0.004):
xmin: the left boundary of the waveguide on the X-axis
xmax: the right boundary of the waveguide on the X-axis
ymin: the lower boundary of the waveguide on the Y-axis
ymax: the upper boundary of the waveguide on the Y-axis
h_radius: the radius of the inner circle, the default value is 0
holex: the center of the inner circle on the X-axis, the default value is 0
holey: the center of the inner circle on the Y-axis, the default value is 0
maxpro: the maximum proportion of the area of a single subdivided triangle to the cross-sectional area of the waveguide, the default value is 0.004

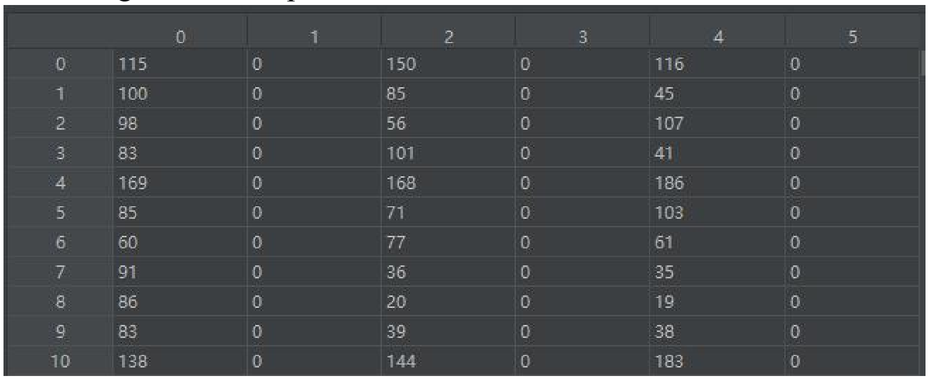
Output

1. nodes: the coordinates of the split point. Column with index 0 is the x coordinate, column with index 1 is the y coordinate, and column with index 2 is the z coordinate (0). Each point's index in the nodes plus 1 is its ID，for example, (0.0254, 0) is the first point.



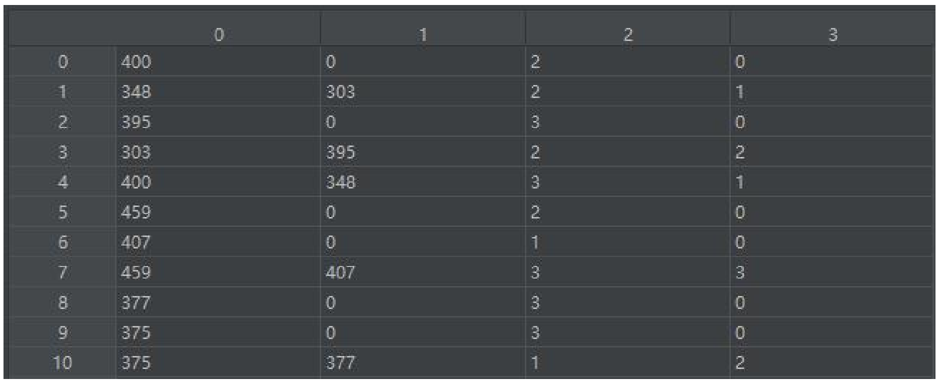
	0	1	2
0	0.02540	0.00000	0.00000
1	0.02540	0.01270	0.00000
2	-0.02540	0.01270	0.00000
3	-0.02540	-0.01270	0.00000
4	0.02540	-0.01270	0.00000
5	0.00632	0.00063	0.00000
6	0.00622	0.00126	0.00000
7	0.00607	0.00187	0.00000
8	0.00585	0.00247	0.00000
9	0.00558	0.00304	0.00000
10	0.00525	0.00358	0.00000

2. facet: the ID of the three vertices of the meshing triangle. The columns with indexes 0, 2, and 4 represent the ids of the first, second, and third vertices that make up the triangle. Each triangle's index in the facets plus 1 is its ID. For example, the first triangle is made up of vertices 115, 150, and 116.



	0	1	2	3	4	5
0	115	0	150	0	116	0
1	100	0	85	0	45	0
2	98	0	56	0	107	0
3	83	0	101	0	41	0
4	169	0	168	0	186	0
5	85	0	71	0	103	0
6	60	0	77	0	61	0
7	91	0	36	0	35	0
8	86	0	20	0	19	0
9	83	0	39	0	38	0
10	138	0	144	0	183	0

3. edge: set of edges, columns with indexes 0 and 1 are the ids of triangles, and columns with indexes 2 and 3 represent the vertices in the triangle. For example, the second side is the side corresponding to the second vertex of the 348th triangle (connecting the first vertex to the third vertex), and the side corresponding to the first vertex of the 303rd triangle; the first side is the side corresponding to the second vertex of the 400th triangle, this side is on the boundary, no common-sided triangle.



	0	1	2	3
0	400	0	2	0
1	348	303	2	1
2	395	0	3	0
3	303	395	2	2
4	400	348	3	1
5	459	0	2	0
6	407	0	1	0
7	459	407	3	3
8	377	0	3	0
9	375	0	3	0
10	375	377	1	2

4. max: the number of meshing points, sides and triangles is stored respectively

A section of code to test is provided below

```
import matplotlib.pyplot as plt
from matplotlib.patches import Polygon
from matplotlib.collections import PatchCollection
 nodes, facet, edge, max = create_mesh(xmin=-0.0254, xmax=0.0254,
ymin=-0.0127, ymax=0.0127, h_radius=0.00635)
 fig = plt.figure()
 ax = fig.add_subplot(111, aspect="equal")
 x, y = nodes[:, 0], nodes[:, 1]
 patches = []
 facet1 = facet[:, [0, 2, 4]]
 for i in range(0, facet1.shape[0]):
     L = [list(x[facet1[i] - 1]), list(y[facet1[i] - 1])]
     polygon = Polygon(list(map(list, zip(*L))), True)
     patches.append(polygon)
 pc = PatchCollection(patches, color="w", edgecolor="k")
 ax.add_collection(pc)
 ax.set_xlim(-0.027, 0.027)
 ax.set_ylim(-0.0135, 0.0135)
 plt.show()
```

2、Function of assemble the matrix: assemble

Position

assemble.pyd

Input

assemble(k,nodes,facet,edge,max)

k: working wave number, it needs to be larger than the maximum wave number solved

nodes,facet,edge,max: the output of the function creat_mesh

Output

A: the A matrix shown below

B: the B matrix shown below

liste: A collection of generated triangles

$$\underbrace{\boxed{\begin{bmatrix} A_{tt} & 0 \\ 0 & 0 \end{bmatrix}}}_{\mathbf{A}} \begin{Bmatrix} e_t \\ e_z \end{Bmatrix} = -\beta^2 \underbrace{\boxed{\begin{bmatrix} B_{tt} & B_{tz} \\ B_{zt} & B_{zz} \end{bmatrix}}}_{\mathbf{B}} \begin{Bmatrix} e_t \\ e_z \end{Bmatrix}$$

Tips: have to impose boundary conditions to solve the general eigenvalue equation

3、Function of getting the electric field at some point: f

Position
DrawFun.py

Input
f (x, y, liste, x0)
x: the x-coordinate set of all the field points that you want to get
y**:** the y-coordinate set of all the field points that you want to get
liste: the output liste in function assemble
x0: solution x in general eigenvalue equation

Output
zx: the x-direction electric field for all the field points that you want to get
zy: the y-direction electric field for all the field points that you want to get