International Institute of Information Technology Bangalore

Basic Computational Topology
SM 402

BCT Implementation Assignment

May 9, 2022

Group 1
Karanjit Saha (IMT2020003)
Arya Kondawar (IMT2020084)
Paras Vekariya (IMT2020547)
Anshul Madurwar(IMT2020554)

Problem Statement

Given any input simplicial complex (up to 3-dimensional), compute β_0 using the boundary matrix method.

Algorithm

We have used the formula given below in our code to calculate β_0:

$$\beta_0 = dim(H_0(K)) = dim(C_0(K)) - dim(Im(\partial_1)) \tag{1}$$

In our program we take vertices, edges and faces (it is redundant) as input (from a .gts file). We then create a matrix corresponding to the linear transformation ∂_1 and then compute $dim(Im(\partial_1))$, i.e. the $rank(\partial_1)$.

As we know,

$$\partial_1(\overline{v_1 v_2}) = \overline{v_2} - \overline{v_3}$$

As we also know $C_0(K)$ is the vector space of 0-chains $\implies$ $dim(C_0(K))$= number of vertices.

By using all the above facts we can easily calculate β_0 for a simplicial complex.

Implementation Steps

1. First we ask for filename as input from user (.gts file) (in main() function). We open the file corresponding to this filename. From here we get the number of vertices, edges and faces in the input simplicial complex, i.e. $dim(C_0(K))$.
2. In the next step we create the matrix corresponding to ∂_1 using the edges of the input.
3. In the last and the final step we calculate β_0 using equation 1.

Steps to run the code

1. Open the terminal.
2. Enter the commmand "pip3 install sympy".
3. Enter the commmand "pip3 install numpy".
4. Enter the commmand "pip3 install scipy".
5. Enter the command "python3 topo.py".
6. Enter the filename of the .gts file you want to take input from.
7. Press Enter to get the final result.

NOTE:- Here the code may take time to calculate the result for very large data (depending on the system hardware specifications).

Demo Results

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
seashell.gts
The number of vertices are:  915
The number of edges are:  2594
The number of faces are:  1680
==========
| β₀ = 1 |
==========
total execution time(in sec) =  1.2641651630401611
```

Figure 1: Testcase 1:- seashell

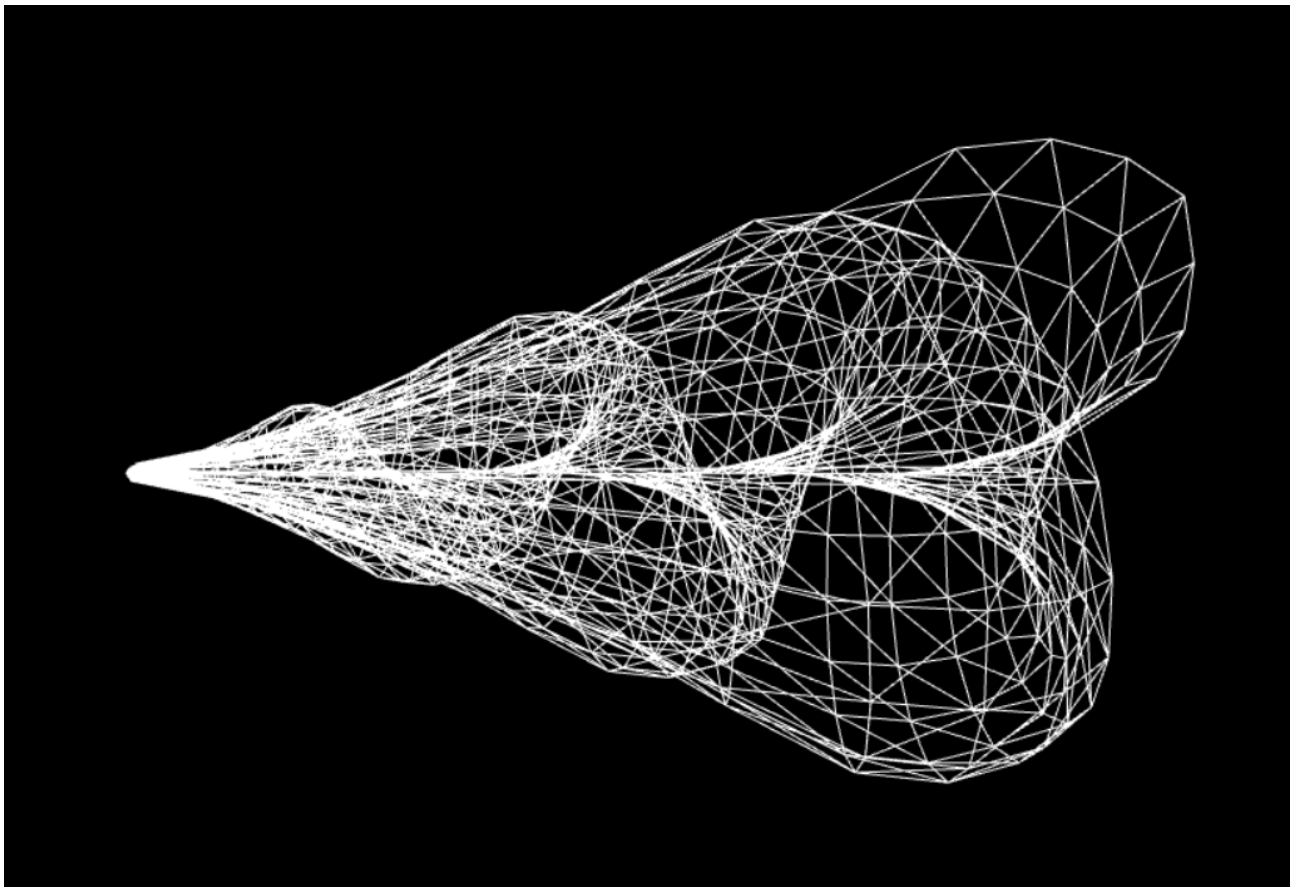

Figure 2: Testcase 1:- seashell

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
tetrahedron.gts
The number of vertices are:  4
The number of edges are:  6
The number of faces are:  4
==========
| β₀ = 1 |
==========
total execution time(in sec) =  0.028593778610229492
```

Figure 3: Testcase 2:- tetrahedron

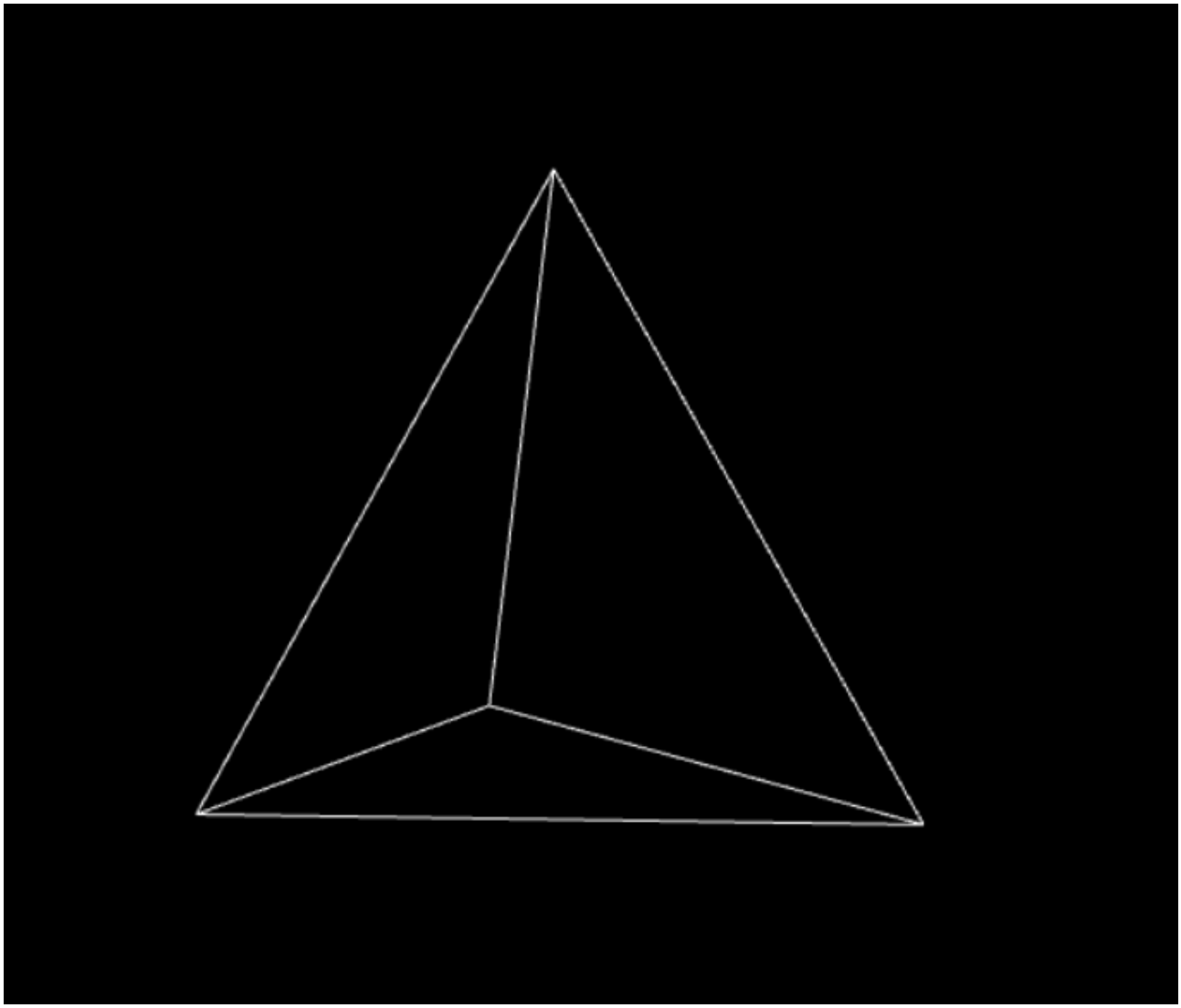

Figure 4: Testcase 2:- tetrahedron

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
scc.gts
The number of vertices are:  4
The number of edges are:  2
The number of faces are:  0
==========
| β₀ = 2 |
==========
total execution time(in sec) =  0.06108403205871582
```

Figure 5: Testcase 3:- two line segments (created by us)

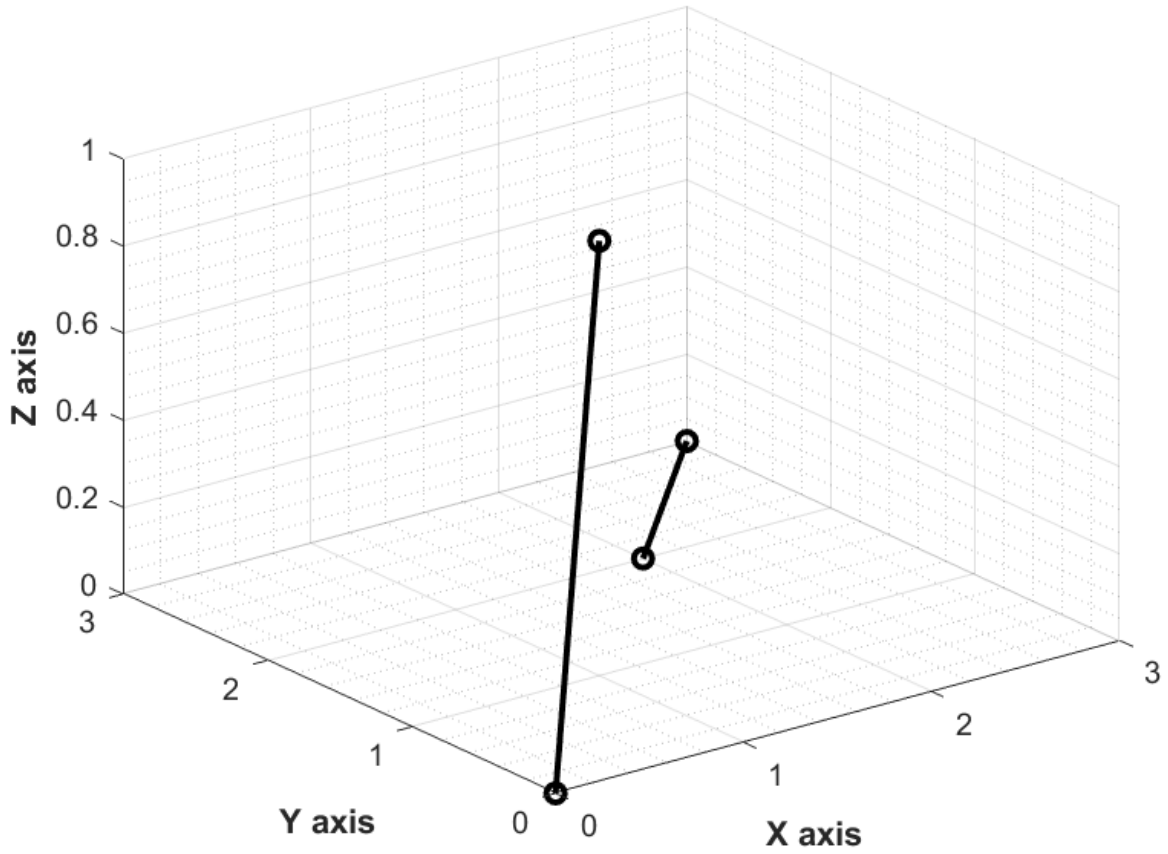

Figure 6: Testcase 3:- two line segments (created by us)

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
sphere20.gts
The number of vertices are:  4002
The number of edges are:  12000
The number of faces are:  8000
==========
| β₀ = 1 |
==========
total execution time(in sec) =  49.04568862915039
```

Figure 7: Testcase 4:- sphere(20)

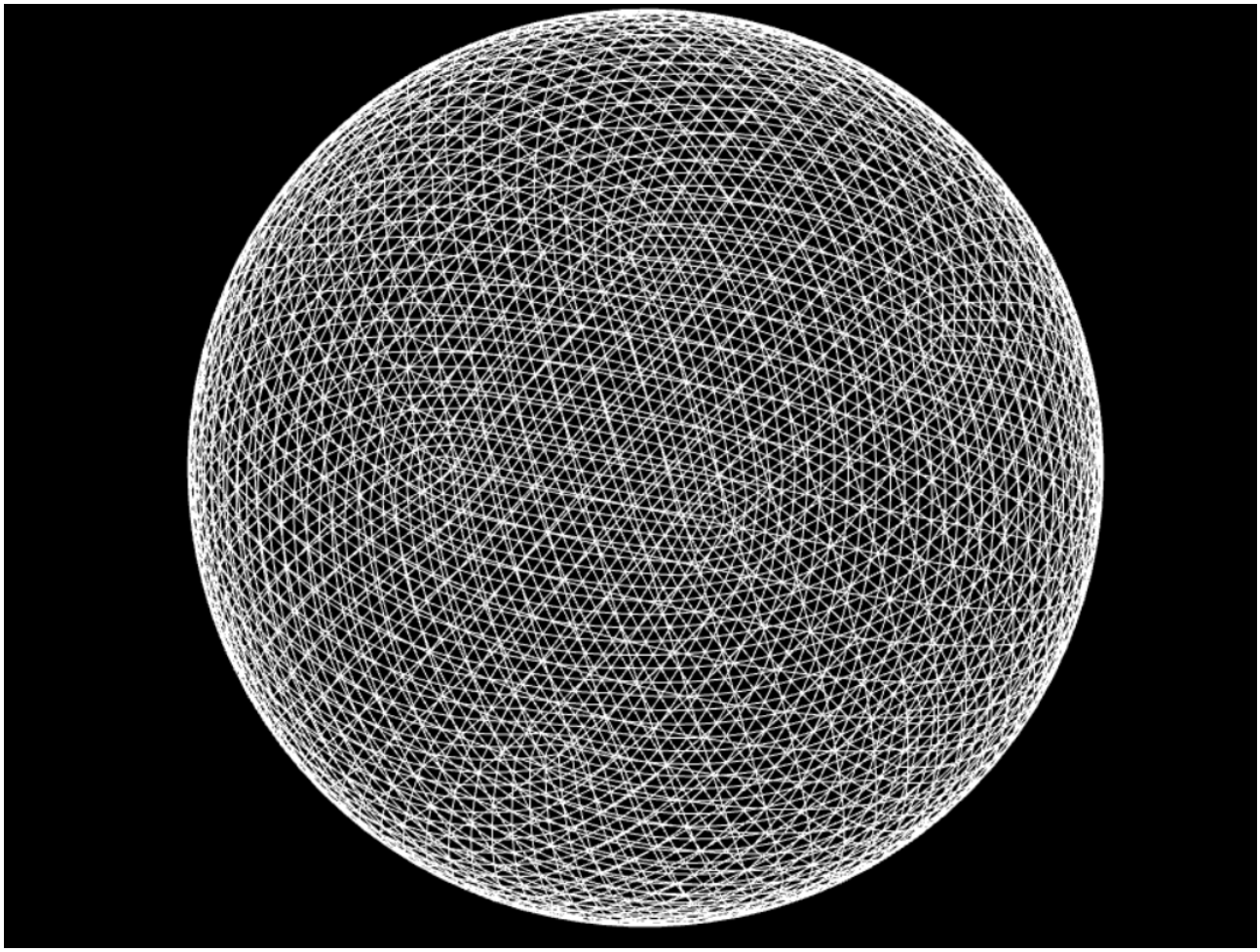

Figure 8: Testcase 4:- sphere(20)

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
sphere5.gts
The number of vertices are:  252
The number of edges are:  750
The number of faces are:  500
==========
| β₀ = 1 |
==========
total execution time(in sec) =  0.05487823486328125
```

Figure 9: Testcase 5:- sphere(5)

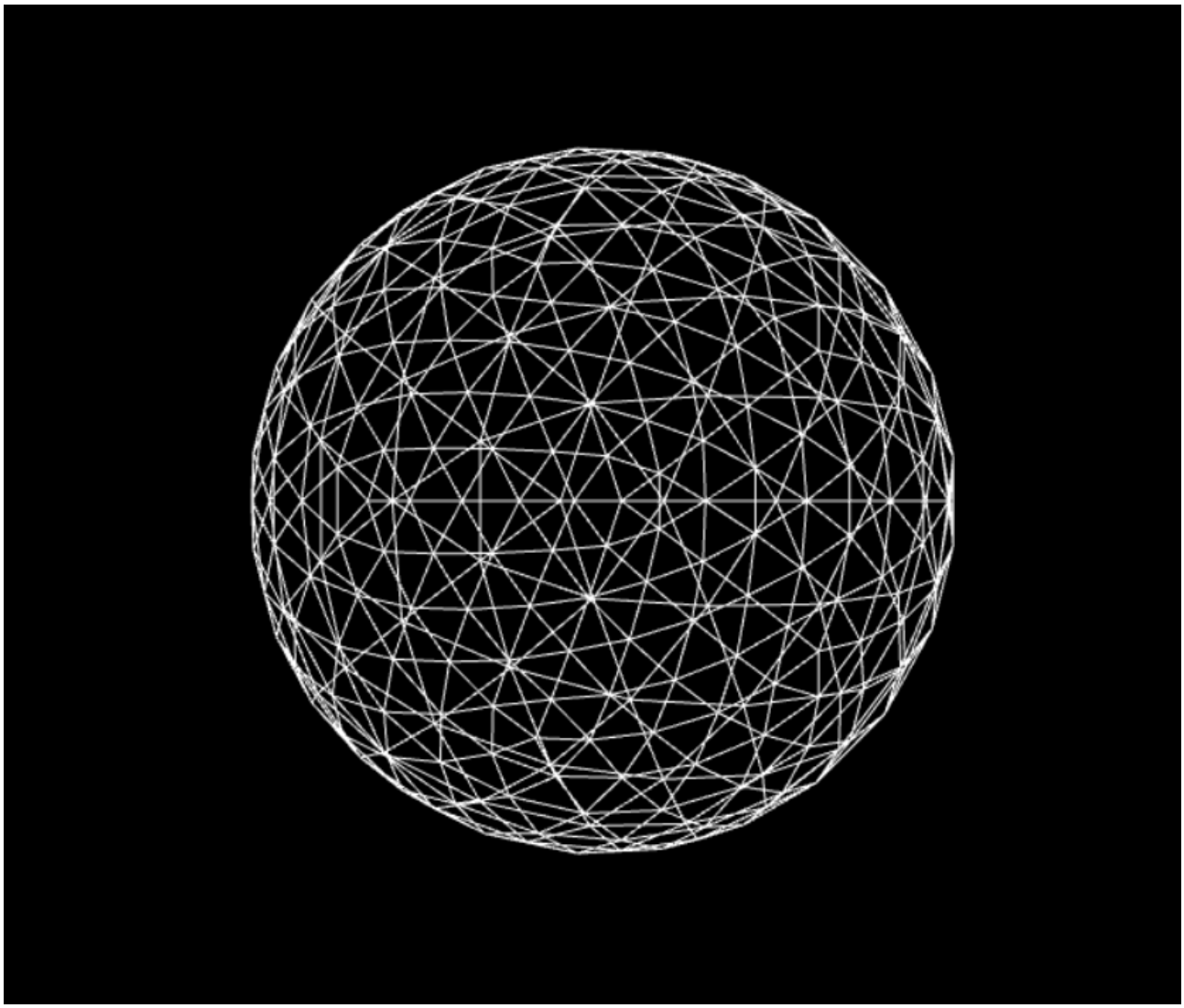

Figure 10: Testcase 5:- sphere(5)

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
icosa.gts
The number of vertices are:  12
The number of edges are:  30
The number of faces are:  20
==========
| β₀ = 1 |
==========
total execution time(in sec) =  0.024145364761352 54
```

Figure 11: Testcase 6:- icosahedron

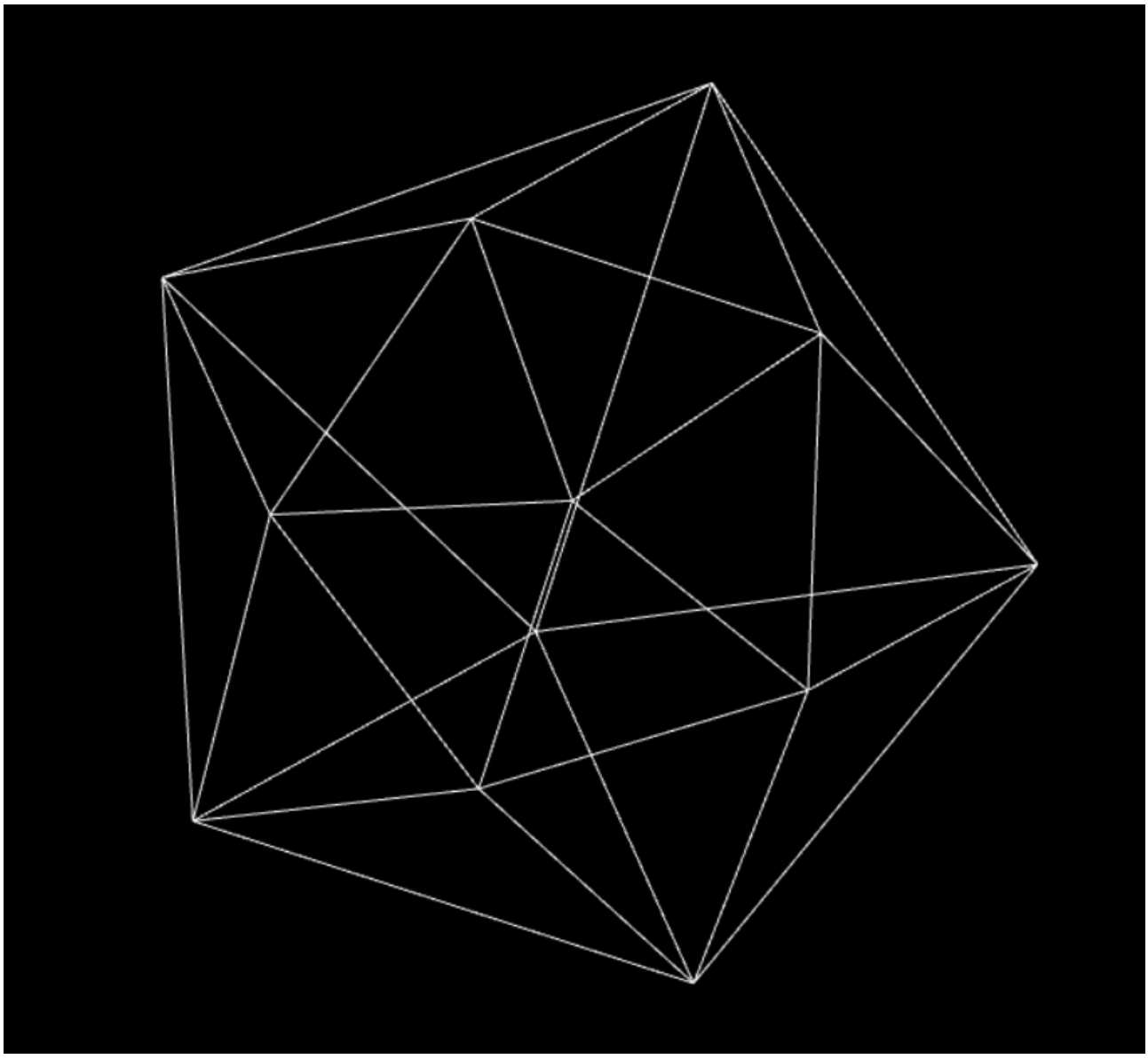

Figure 12: Testcase 6:- icosahedron

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
goblet.gts
The number of vertices are:  502
The number of edges are:  1500
The number of faces are:  1000
==========
| β₀ = 1 |
==========
total execution time(in sec) =  0.12866997718811035
```

Figure 13: Testcase 7:- goblet

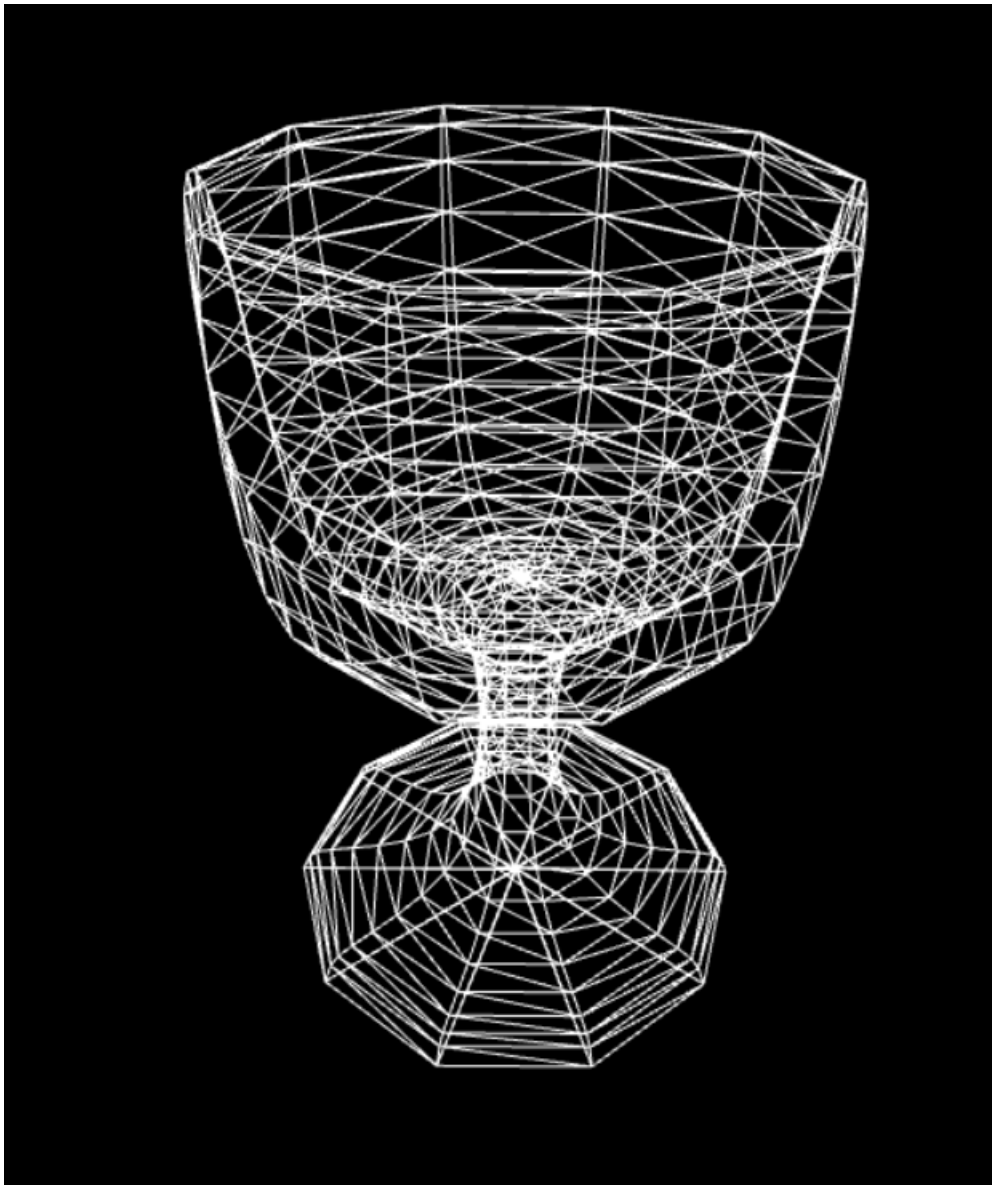

Figure 14: Testcase 7:- goblet

```
rahuljain rahul % python3 topo.py
Tangle.gtsEnter the filename for reading the data

The number of vertices are:  8856
The number of edges are:  26592
The number of faces are:  17728
==========
| β0 = 1 |
==========
total execution time(in sec) =  247.12285804748535
```

Figure 15: Testcase 8:- Tangle Cube

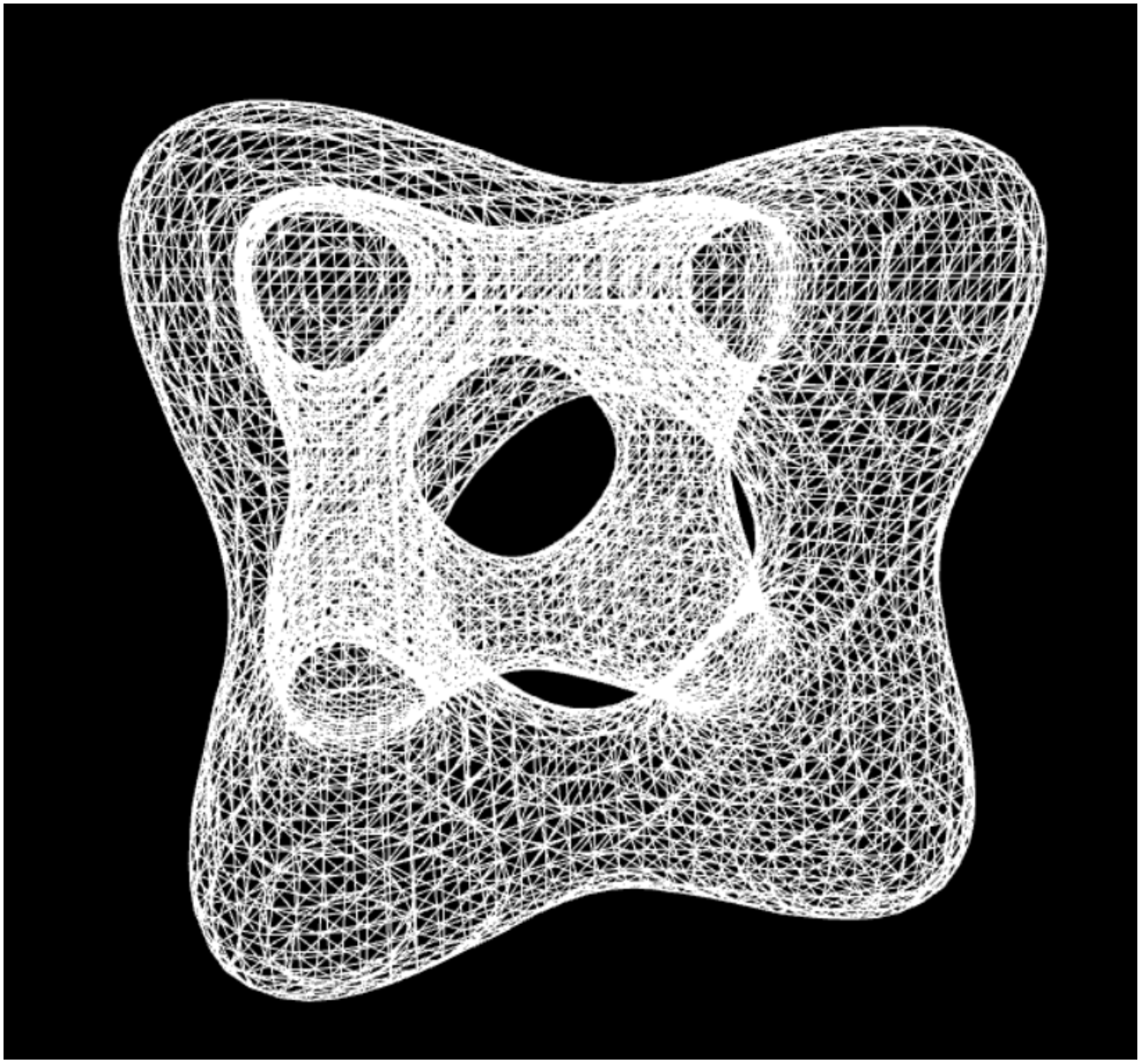

Figure 16: Testcase 8:- Tangle Cube

```
rahuljain rahul % python3 topo.py
Enter the filename for reading the data
Torus.gts
The number of vertices are:  6320
The number of edges are:  18960
The number of faces are:  12640
==========
| β0 = 1 |
==========
total execution time(in sec) =  100.028225183486 94
```

Figure 17: Testcase 9:- Torus

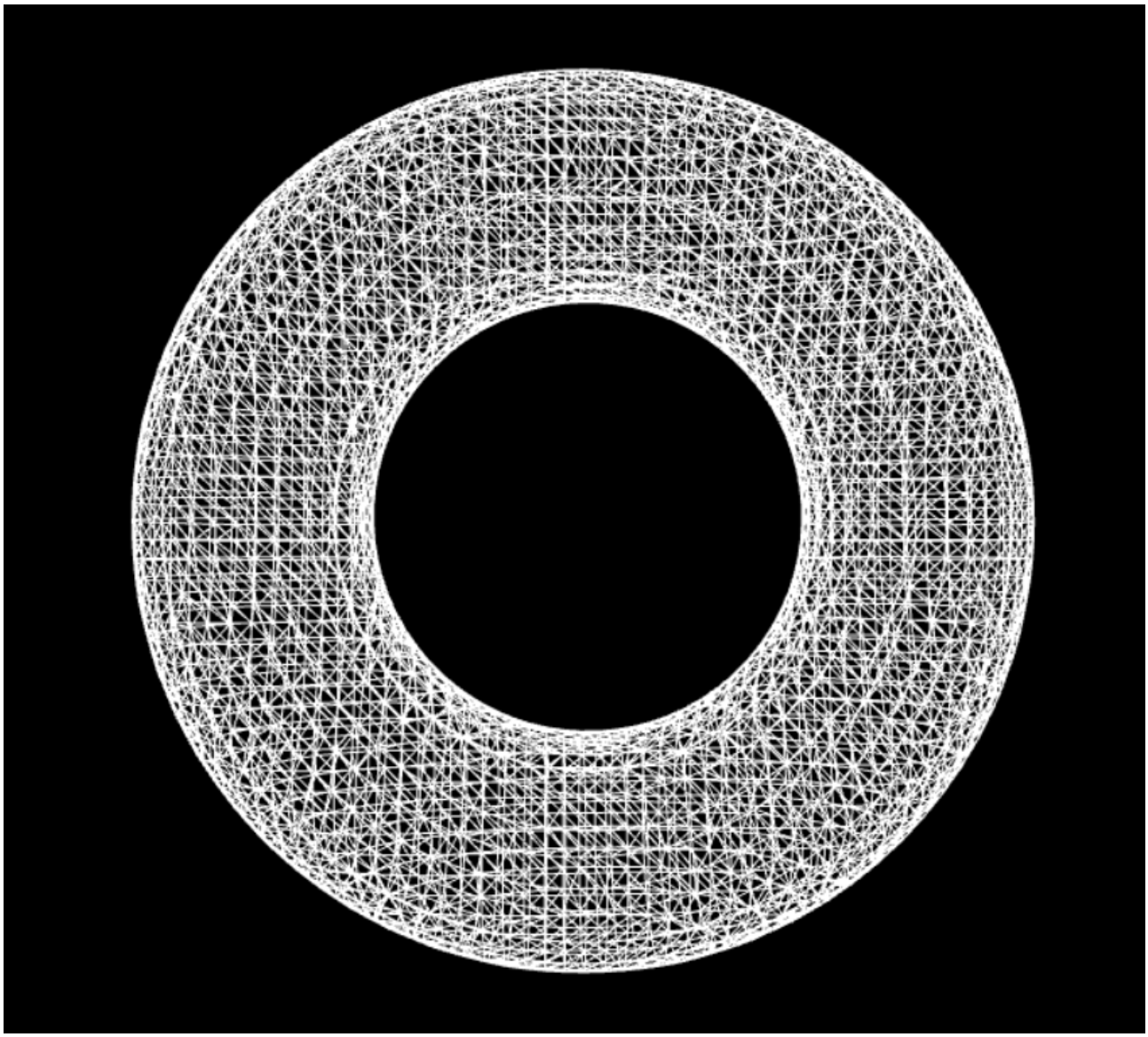

Figure 18: Testcase 9:- Torus

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
2squares.gts
The number of vertices are:  8
The number of edges are:  8
The number of faces are:  0
==========
| β₀ = 2 |
==========
total execution time(in sec) =  0.04158616065979004
```

Figure 19: Testcase 10:- 2 disconnected squares(created by us)

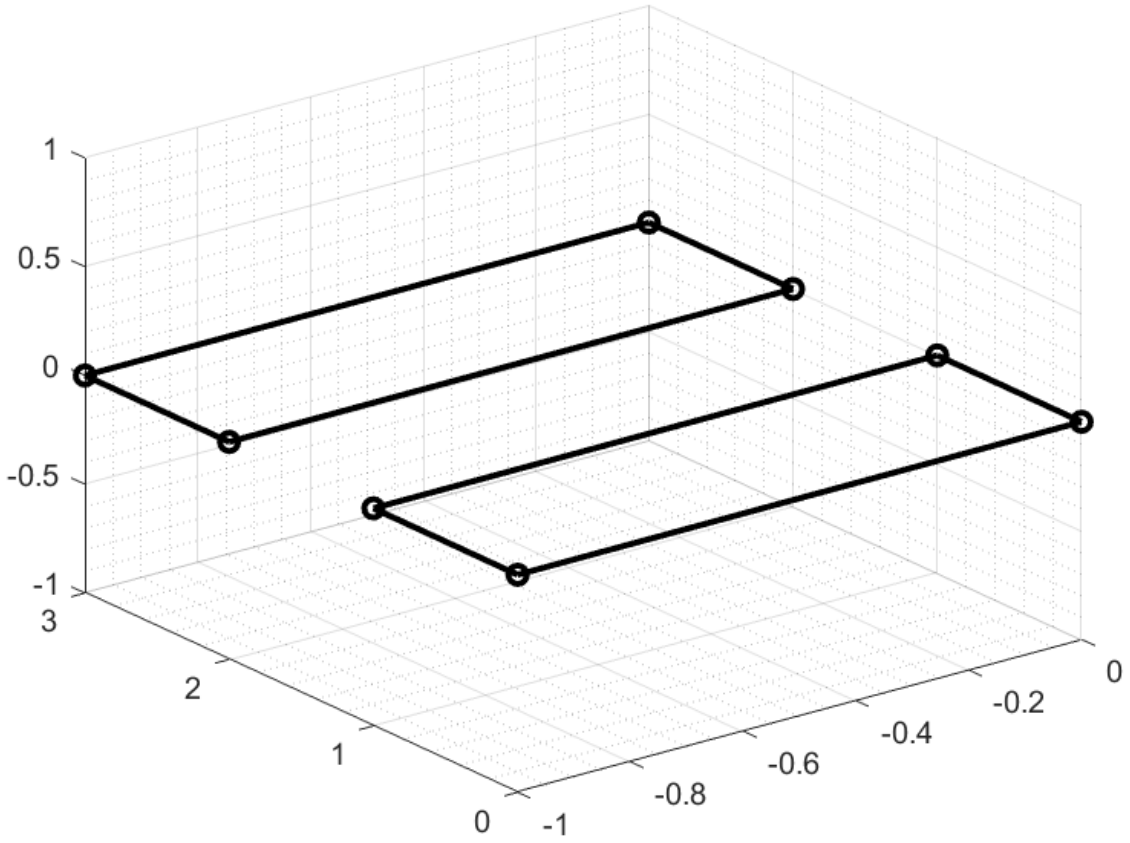

Figure 20: Testcase 10:- 2 disconnected squares(created by us)

```
(base) karanjitsaha@pop-os:~/Desktop/TOPOLOGY PROJECT$ python3 topo.py
Enter the filename for reading the data
demotestcase1.gts
The number of vertices are:  8
The number of edges are:  5
The number of faces are:  0
==========
| β₀ = 4 |
==========
total execution time(in sec) =  0.03651547431945801
```

Figure 21: Testcase 11:- 2 disjoint vertices, a line segment and a square(created by us)

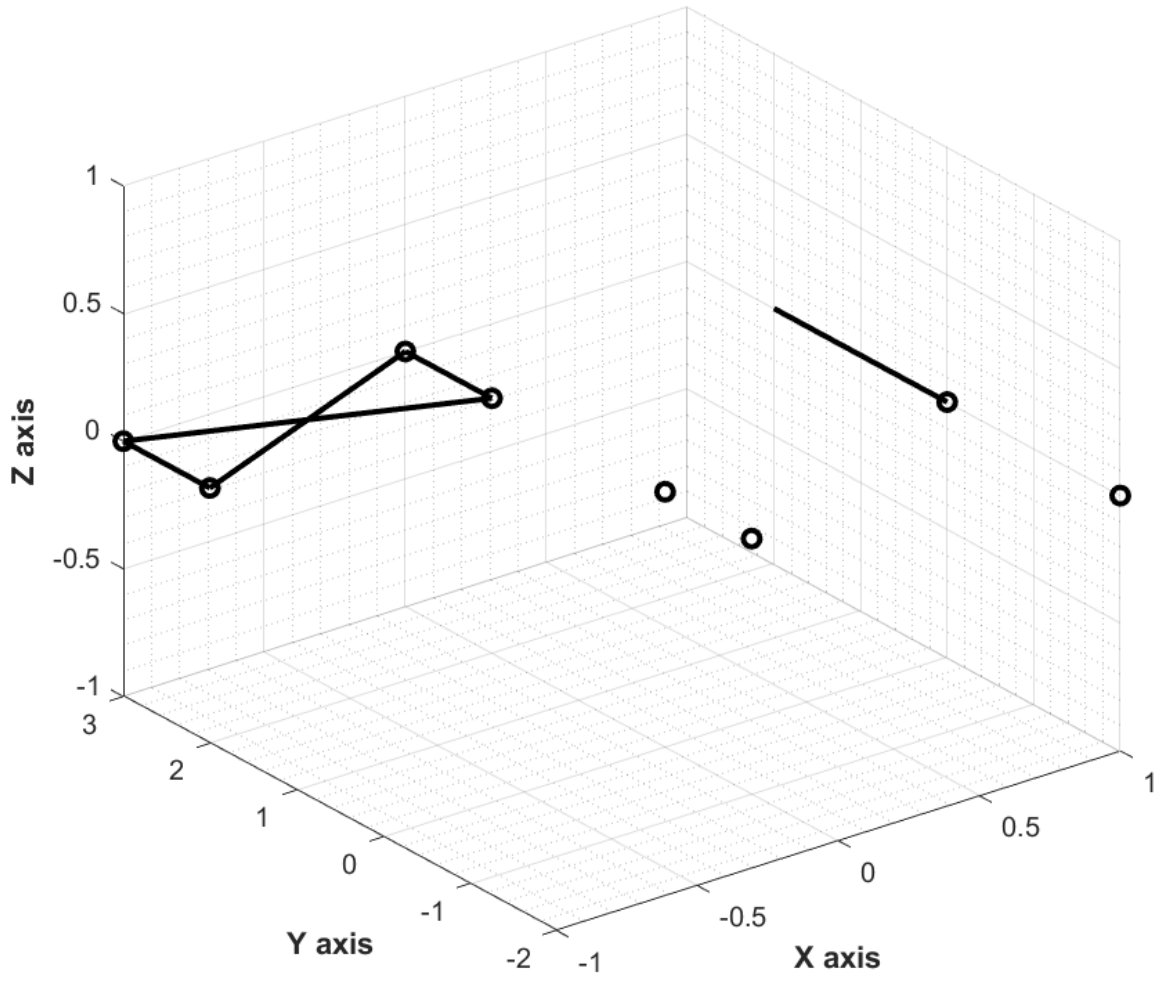

Figure 22: Testcase 11:- 2 disjoint vertices, a line segment and a square(created by us)

Python Code

Python code for calculating β_0

```
import sympy as sym
import numpy as np
import scipy.linalg.interpolative as sli
import time


def readfile(filename):
    f = open(filename, "r+");
    line = (f.readline());
    list_ver_edg_fac = line.split(' ');
    num_ver = int(list_ver_edg_fac[0]);
    print("The number of vertices are: ", num_ver);
    num_edg = int(list_ver_edg_fac[1]);
    print("The number of edges are: " ,num_edg);
    num_fac = int(list_ver_edg_fac[2]);
    print("The number of faces are: " ,num_fac);


    edges = []

    for i in range(0, num_ver):
        j = f.readline()

    for i in range(0, num_edg):
        edge = f.readline();
        edge_1 = list(map(int, edge.split(' ')))
        edges.append(edge_1)
    Return = []
    Return.append(num_ver)
    Return.append(num_edg)
    Return.append(edges)
    return Return


def image_space(edges, num_ver, num_edg):
    img_space = []
    for i in range(0, num_ver):

        temp1=[0]*num_edg
        a=False
        b=False
        for j in range(0, num_edg):
            if(edges[j][0] == (i+1)):
                temp1[j]=-1
                a=True

            elif(edges[j][1] == i+1):
                temp1[j]=1
                b=True
            elif(a==True and b==True):
                continue

        img_space.append(temp1)
    return img_space

def calculate_betti_0(img_space, num_ver):
    begin=time.time()
    rank_matrix = np.array(img_space)
    rank=np.linalg.matrix_rank(rank_matrix)
    betti_0 = num_ver - rank

    print("==========")
    print('| \N{GREEK SMALL LETTER BETA}\N{SUBSCRIPT ZERO} =',betti_0,'|')
    print("==========")
    end=time.time()
    print("total execution time(in sec) = ",end-begin)

def main():
```

```
    file_name = input("Enter the filename for reading the data\n")
    Return_list = readfile(file_name)
    num_ver = Return_list[0]
    num_edg = Return_list[1]
    edges_list = Return_list[2]
    img_space_list = image_space(edges_list, num_ver, num_edg)
    calculate_betti_0(img_space_list, num_ver)

main()
```

GitHub Link

Please visit this for the source code.
https://github.com/KaranjitSaha/TOPOLOGY-PROJECT

References

1. http://web.cse.ohio-state.edu/~wang.1016/courses/788/Lecs/lec7-qichao.pdf
2. https://www.math.rug.nl/~gert/documents/2006/rv_ecg_book_chapter7.pdf
3. https://jeremykun.com/2013/04/10/computing-homology/
4. https://jeremykun.com/2014/01/23/fixing-bugs-in-computing-homology/
5. https://en.wikipedia.org/wiki/Quotient_space_(linear_algebra
6. https://en.wikipedia.org/wiki/Rank%E2%80%93nullity_theorem
7. https://en.wikipedia.org/wiki/Rank_(linear_algebra)
8. https://en.wikipedia.org/wiki/Simplicial_complex
9. http://gts.sourceforge.net/samples.html (for getting the testcases to test out code)