PROJECT 8:

PROBLEM STATEMENT:

Given a volumetric scalar data, compute a sub-level set corresponding to a given scalar value. Integrate with P2 and compute β1 for an increasing sequence of scalar values.

THEORY:

We calculate the sublevel set by considering all the vertices which have a scalar value <= the given scalar value.

Now coming to the complicated stuff, in order to calculate the k dimensional faces we do as follows:

we say that, if K+1 points are 'close' to each other, then they form a K simplex. Here we use an arbitrary definition of 'closeness'. For our case, data is a set of 3D coordinates sampled at 1 unit, thus we define 'closeness' for our case as:

'closeness' : 2 points are close if the Euclidean distance between the 2 is at most $\sqrt{3}$ units. ie, the farthest 2 points which we consider as 'close' lie on the body diagonal of a 1X1X1 cube.

For our problem statement, we need to calculate B1, hence we only require to find upto the two dimensional faces. We already have the zero dimensional faces (ie the vertices), in order to find the one dimensional faces(ie edges), we iterate over all the vertices to see if any 2 of them are 'close', if so, we add them to the K simplicial complex.

Similarly for the two dimensional faces, we check if 3 points are close each other, if so, we add them to the simplicial complex. We do this as follows:

We implement this by dividing our simpicial complex into the simplest form of 3 simplices (terahedron). Thus, in doing this we divide each unit cube into 6 tetrahedron. The method to divide the cube into 6 tetrahedrons is citied from the following link: http://paulbourke.net/geometry/polygonise/.

ALGORITHM:

The algorithm for calculating the vertices is straightforward, we just iterate over all the vertices and check if it is present in the sub level set.

Now moving on to edges :

For a given coordinate (i, j, k), we search for edges by checking if every vertex 'close' to (i, j, k) :

(i+1, j, k), (i, j+1, k), (i, j, k+1), (i+1, j+1, k), (i, j+1, k+1), (i+1, j, k+1), (i+1, j+1, k+1), has a scalar value <= to the given scalar value. The edges obtained from this are added to our edgelist.

Moving onto faces :

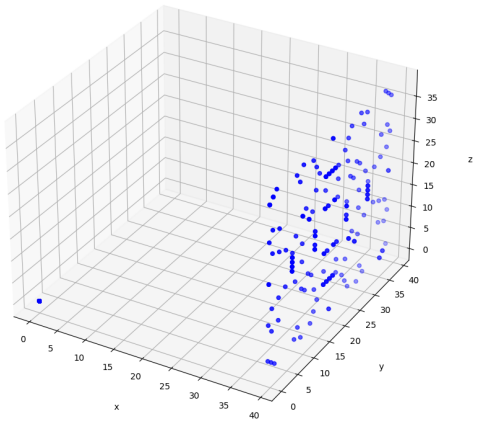
Similar to edge, for every vertex (i,j,k), we search for faces by checking if there are 2 other vertices 'close' to (i,j,k) such that their scalar value <= given scalar value. These 3 vertices form a face and hence added to our facelist.

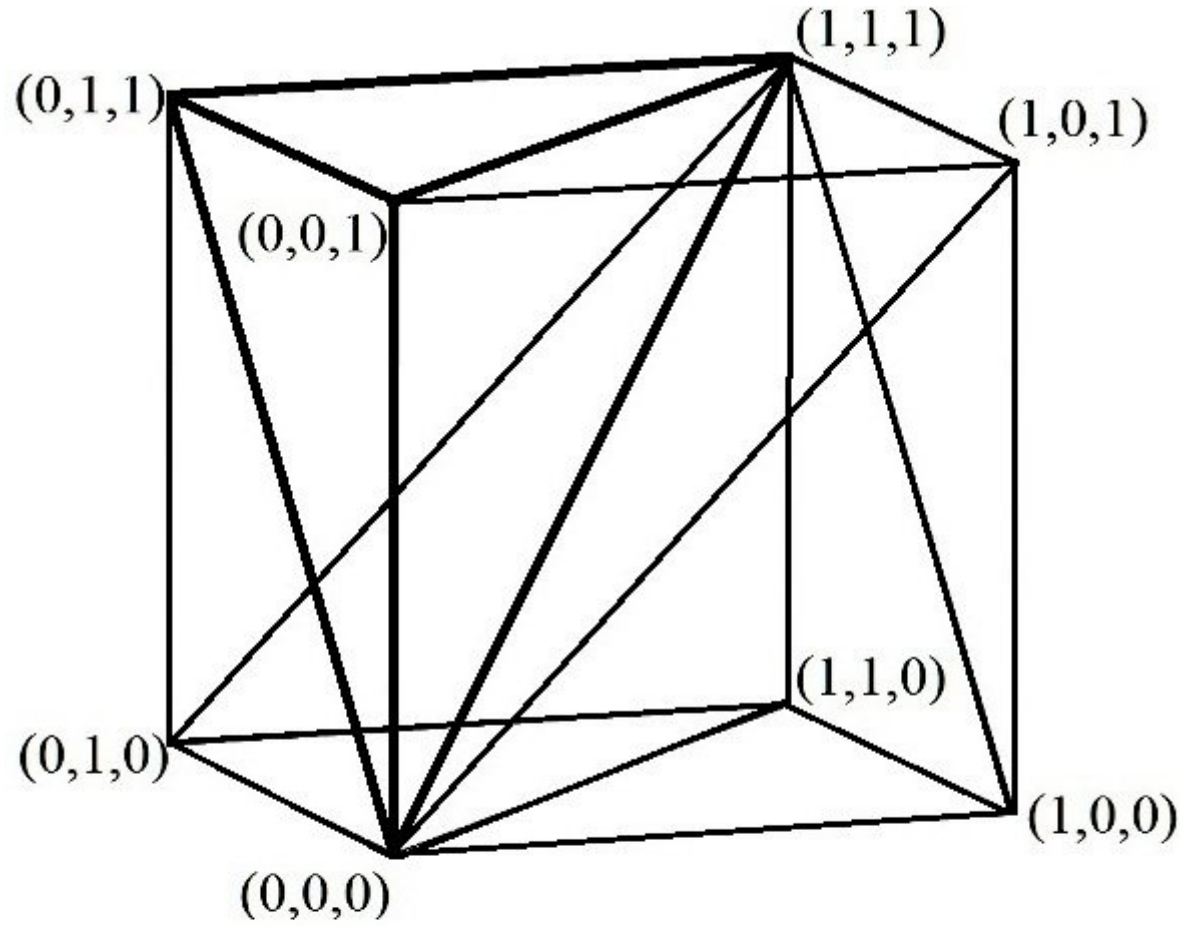
using the above, we calculate the value of B1 and report our answer.

INSTRUCTIONS TO RUN:

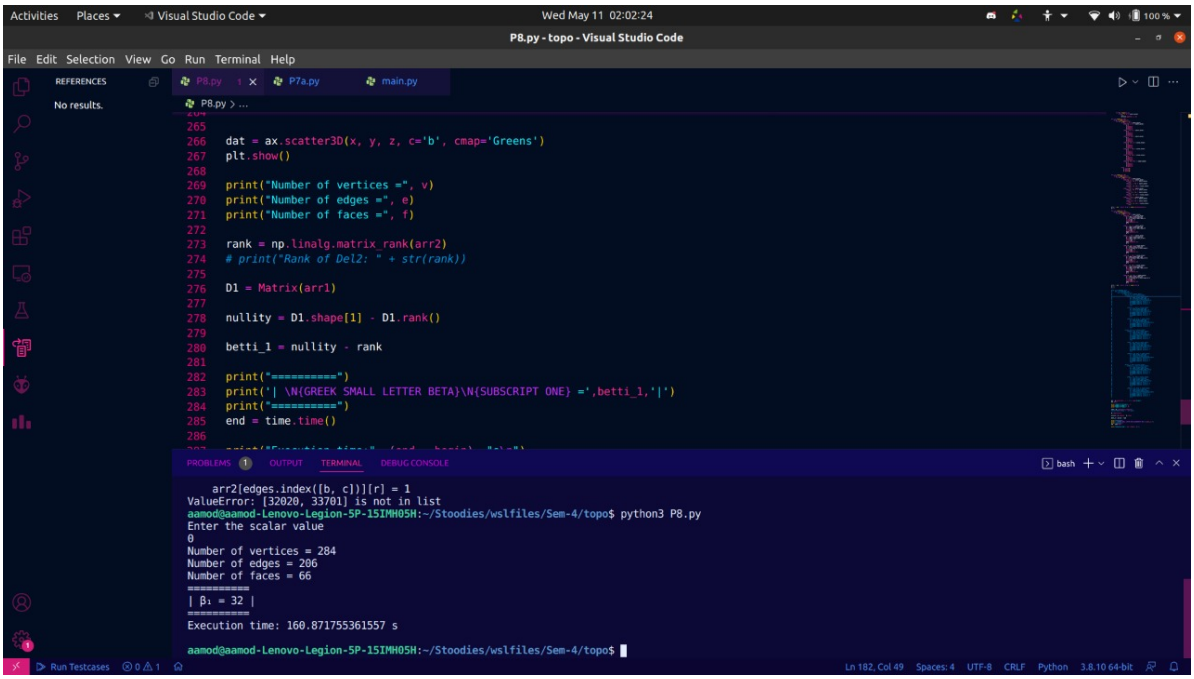
- Open P8.py
- compile and run the python script in the terminal using command : ***"Python3 P8.py"***
- The output will be displayed on the terminal
- We have hardcoded the input file, so our script will run for the file we hardcoded.

SCREENSHOTS:





T1	T2	T3	T4	T5	T6
(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)	(0,0,0)
(0,0,1)	(0,0,1)	(0,1,0)	(0,1,0)	(1,0,0)	(1,0,0)
(0,1,1)	(1,0,1)	(0,1,1)	(1,1,0)	(1,0,1)	(1,1,0)
(1,1,1)	(1,1,1)	(1,1,1)	(1,1,1)	(1,1,1)	(1,1,1)



CODE:

```
import numpy as np
import time
import matplotlib.pyplot as plt
from sympy import Matrix

begin = time.time()
#filename = input("Enter the filename for reading the data\n")
filename = 'marschner_lobb_41x41x41_uint8.raw.txt'

dim_x, dim_y, dim_z = 41, 41, 41
A = np.fromfile(filename, dtype='uint8', sep="")
A = A.reshape((dim_x, dim_y, dim_z))

ax = plt.axes(projection='3d')
ax.grid()
ax.set_xlabel('x', labelpad=20)
ax.set_ylabel('y', labelpad=20)
ax.set_zlabel('z', labelpad=20)

scalar_value = input("Enter the scalar value\n")
scalar_value = int(scalar_value)
v = 0
e = 0
f = 0
vertices = []
edges = []
faces = []
x = []
y = []
z = []

for i in range(dim_x):
    for j in range(dim_y):
        for k in range(dim_z):
            if (A[i, j, k] <= scalar_value):
                v+=1
                vertices.append([i, j, k])

for i in range(dim_x-1):
    for j in range(dim_y-1):
        for k in range(dim_z-1):
                if (A[i, j, k] <= scalar_value):
                    if(A[i+1, j, k] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                    if(A[i, j+1, k] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                    if(A[i, j, k+1] <= scalar_value):
                        e+=1
                        x.append(i)
```

```
                        y.append(j)
                        z.append(k)
                    if(A[i+1, j+1, k] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                    if(A[i+1, j, k+1] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                    if(A[i, j+1, k+1] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                    if(A[i+1, j+1, k+1] <= scalar_value):
                        e+=1
                        x.append(i)
                        y.append(j)
                        z.append(k)
                else:
                    x.append(0)
                    y.append(0)
                    z.append(0)


for i in range(dim_x-1):
    for j in range(dim_y-1):
        for k in range(dim_z-1):
                if (A[i, j, k] <= scalar_value):
                    if(A[i+1, j, k] <= scalar_value):
                        if(A[i+1, j+1, k] <= scalar_value):
                            f+=1
                        if(A[i+1, j, k+1] <= scalar_value):
                            f+=1
                        if(A[i+1, j+1, k+1] <= scalar_value):
                            f+=1
                    if(A[i, j+1, k] <= scalar_value):
                        if(A[i+1, j+1, k] <= scalar_value):
                            f+=1
                        if(A[i, j+1, k+1] <= scalar_value):
                            f+=1
                        if(A[i+1, j+1, k+1] <= scalar_value):
                            f+=1
                    if(A[i, j, k+1] <= scalar_value):
                        if(A[i+1, j, k+1] <= scalar_value):
                            f+=1
                        if(A[i, j+1, k+1] <= scalar_value):
                            f+=1
                        if(A[i+1, j+1, k+1] <= scalar_value):
                            f+=1

arr1 = [ [0] * int(e) for i in range(dim_x*dim_y*dim_z)]
```

```
p = 0

for i in range(dim_x-1):
    for j in range(dim_y-1):
        for k in range(dim_z-1):
                if (A[i, j, k] <= scalar_value):
                    if(A[i+1, j, k] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = (i+1)+dim_x*j+dim_z*dim_y*k
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i, j+1, k] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = i+dim_x*(j+1)+dim_z*dim_y*k
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i, j, k+1] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = i+dim_x*j+dim_z*dim_y*(k+1)
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i+1, j+1, k] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = (i+1)+dim_x*(j+1)+dim_z*dim_y*k
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i+1, j, k+1] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = (i+1)+dim_x*j+dim_z*dim_y*(k+1)
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i, j+1, k+1] <= scalar_value):
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = i+dim_x*(j+1)+dim_z*dim_y*(k+1)
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

                    if(A[i+1, j+1, k+1] <= scalar_value):
```

```
                        a = i+dim_x*j+dim_z*dim_y*k
                        b = (i+1)+dim_x*(j+1)+dim_z*dim_y*(k+1)
                        arr1[a][p] = -1
                        arr1[b][p] = 1
                        edges.append([a, b])
                        p+=1

arr2 = [ [0] * int(f) for i in range(int(e))]
q = 0
r = 0

for i in range(dim_x-1):
    for j in range(dim_y-1):
        for k in range(dim_z-1):
                if (A[i, j, k] <= scalar_value):
                    if(A[i+1, j, k] <= scalar_value):
                        if(A[i+1, j+1, k] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = (i+1)+dim_x*j+dim_y*dim_z*k
                            c = (i+1)+dim_x*(j+1)+dim_y*dim_z*k
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i+1, j, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = (i+1)+dim_x*j+dim_y*dim_z*k
                            c = (i+1)+dim_x*j+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i+1, j+1, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = (i+1)+dim_x*j+dim_y*dim_z*k
                            c = (i+1)+dim_x*(j+1)+dim_z*dim_y*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                    if(A[i, j+1, k] <= scalar_value):
                        if(A[i+1, j+1, k] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = i+dim_x*(j+1)+dim_y*dim_z*k
                            c = (i+1)+dim_x*(j+1)+dim_y*dim_z*k
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i, j+1, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
```

```
                            b = i+dim_x*(j+1)+dim_y*dim_z*k
                            c = i+dim_x*(j+1)+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i+1, j+1, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = i+dim_x*(j+1)+dim_y*dim_z*k
                            c = (i+1)+dim_x*(j+1)+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                    if(A[i, j, k+1] <= scalar_value):
                        if(A[i+1, j, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = i+dim_x*j+dim_y*dim_z*(k+1)
                            c = (i+1)+dim_x*j+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i, j+1, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = i+dim_x*j+dim_y*dim_z*(k+1)
                            c = i+dim_x*(j+1)+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1

                        if(A[i+1, j+1, k+1] <= scalar_value):
                            a = i+dim_x*j+dim_z*dim_y*k
                            b = i+dim_x*j+dim_y*dim_z*(k+1)
                            c = (i+1)+dim_x*(j+1)+dim_y*dim_z*(k+1)
                            arr2[edges.index([a, b])][r] = 1
                            arr2[edges.index([a, c])][r] = -1
                            arr2[edges.index([b, c])][r] = 1
                            r+=1


dat = ax.scatter3D(x, y, z, c='b', cmap='Greens')
plt.show()

end = time.time()
print("Number of vertices =", v)
print("Number of edges =", e)
print("Number of faces =", f)

rank = np.linalg.matrix_rank(arr2)
print("Rank of Del2: " + str(rank))
```

```
D1 = Matrix(arr1)

nullity = D1.shape[1] - D1.rank()

betti_1 = nullity - rank

print("==========")
print('| \N{GREEK SMALL LETTER BETA}\N{SUBSCRIPT ONE} =',betti_1,'|')
print("==========")
end=time.time()
print("Total execution time(in sec) = ",end-begin)

print("Execution time:", (end - begin), "s\n")
```

TEAM MEMBERS:

- Aamod BK
- Saket Gurjar
- Prakhar Rastogi
- Prabal Mahajan