Estes, Mani

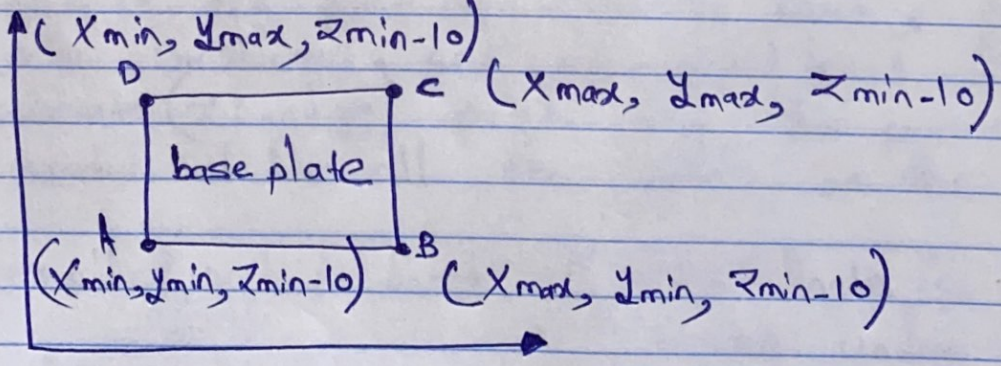
Support Generation Algorithm

1) Read the information from STL file. The file imported in matlab and Faces, Vertices and Normals were extracted.

2) Final build orientation is changed by rotating the part 30° w.r.t. x-axis and 45° w.r.t. y-axis and STL file of the rotated part were plotted

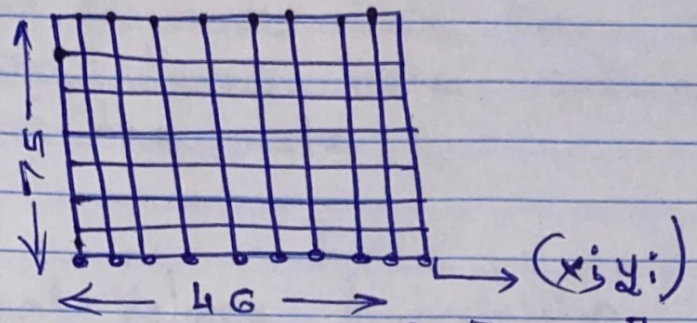
3) For the ray shooting approach, base plate was created and for that amongst all the rotated Vertices, (X_{min} Y_{min} and Z_{min}), (X_{max}, Y_{max}) have been extracted. It is given that, assume the support substrate is placed 10 mm below the lowermost z-co-ordinate.

Dimensions of the base plate are :-



Rectangular base plate is now created in the matlab.

4) Shoot rays from the baseplate in +z direction
•) points have been discretized on the base plate with the spacing of 2mm in x and y direction.

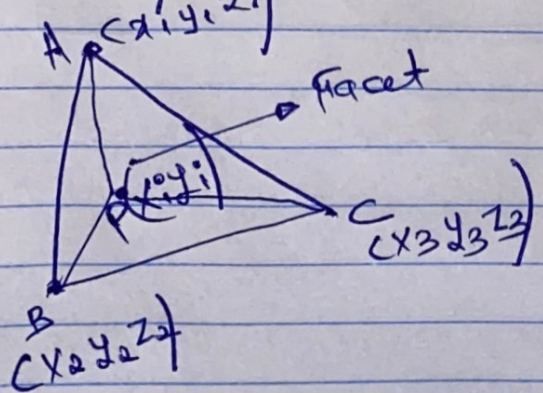


It form a matrix of [46x75]

5) This discretized plate along with the rotated part have been plotted together to initiate the ray shooting process.

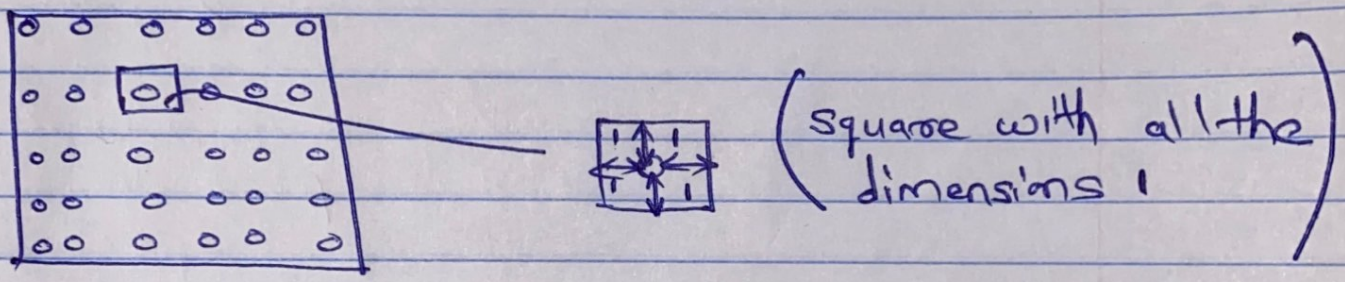
6) Now, for every discrete point on the plate, rays have been shooted on all the faces of the part and inside/outside test were performed and if the point is inside the facet, those facets were considered, and represented by 1 for that discrete (xi and yi) point and this process is carried out for all the points on the plate.

Inside/outside test (z co-ordinate is not taken under consideration)

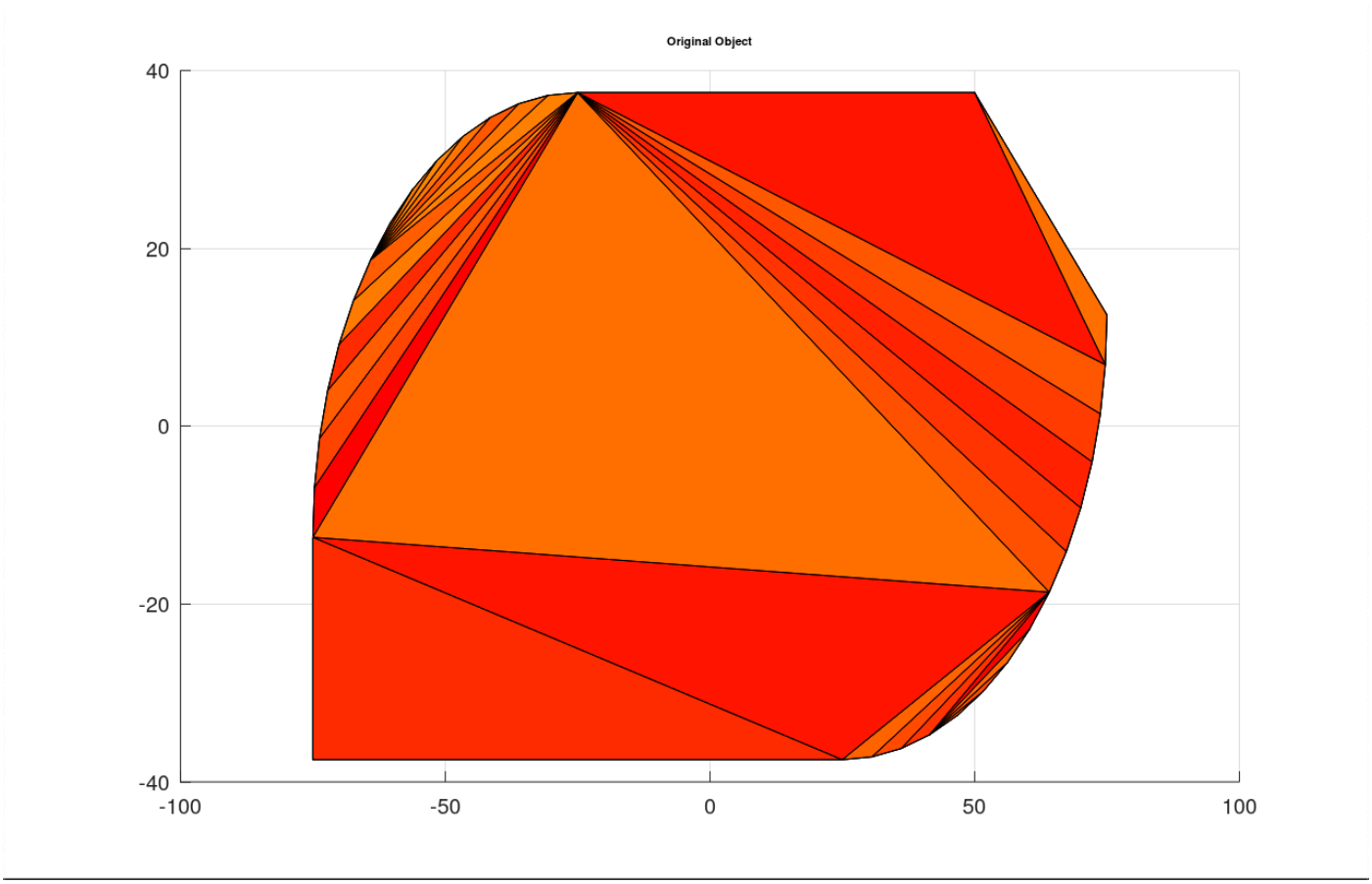


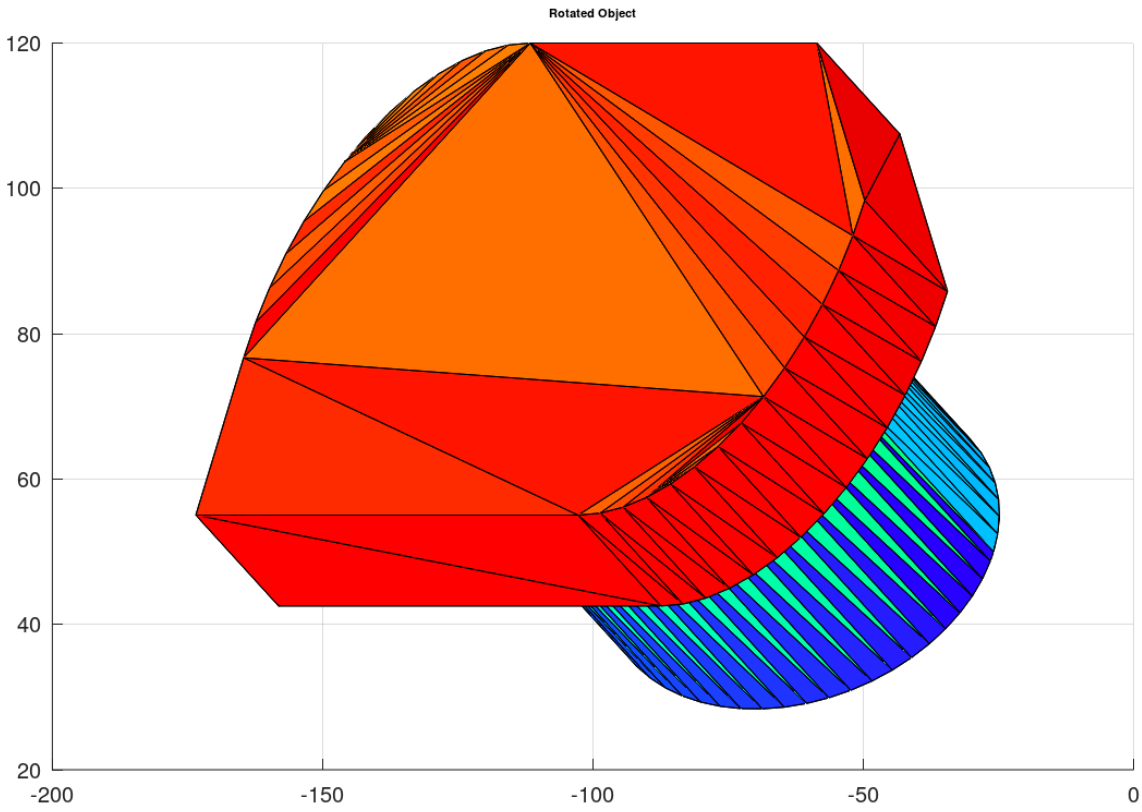
If point 'p' lies inside the triangle; then if else loop have been computed and Area of facet = Summation of areas of triangles with the tolerence have been computed and only those faces were selected for discrete point on the plate.

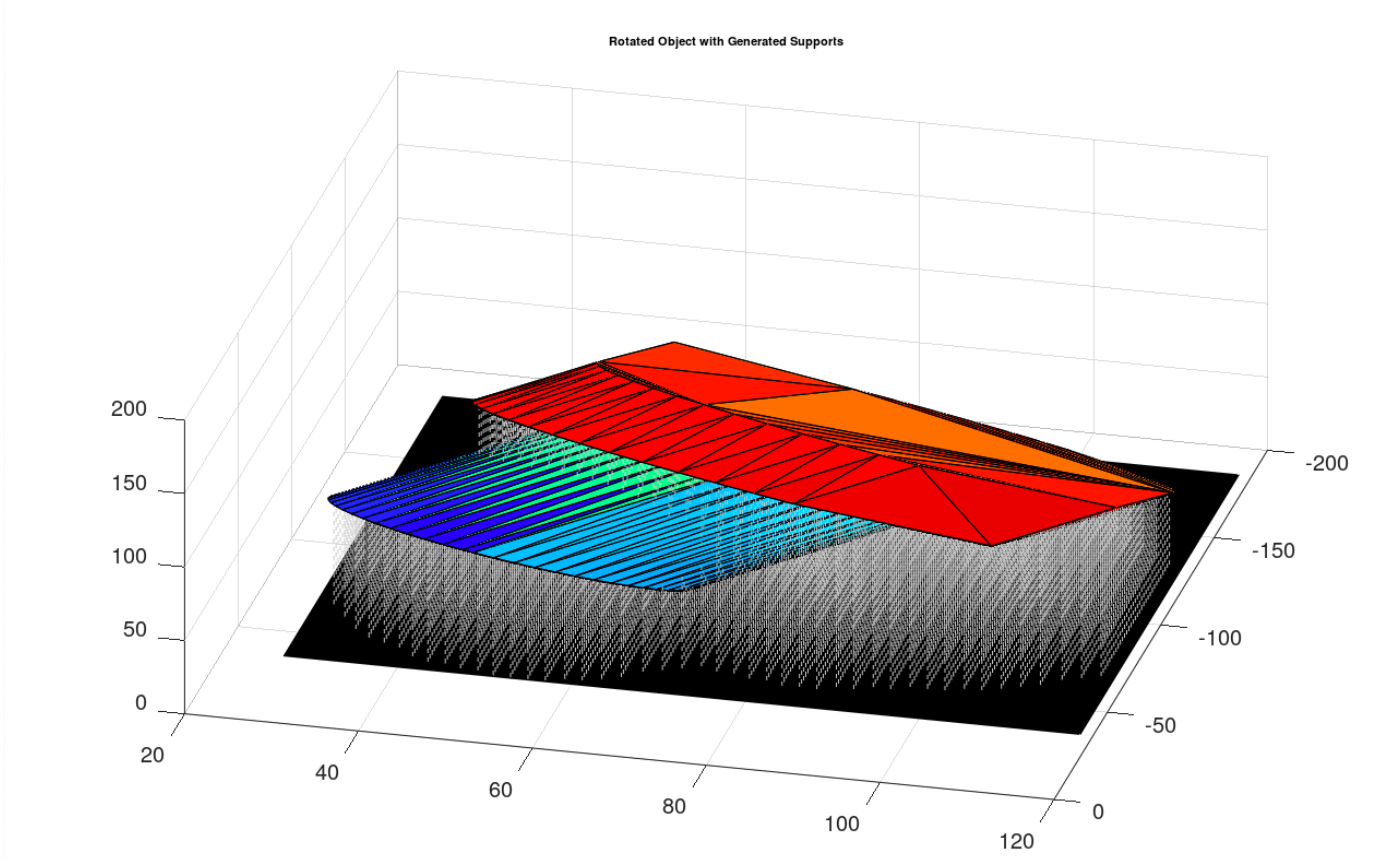
Support Volume Approximation

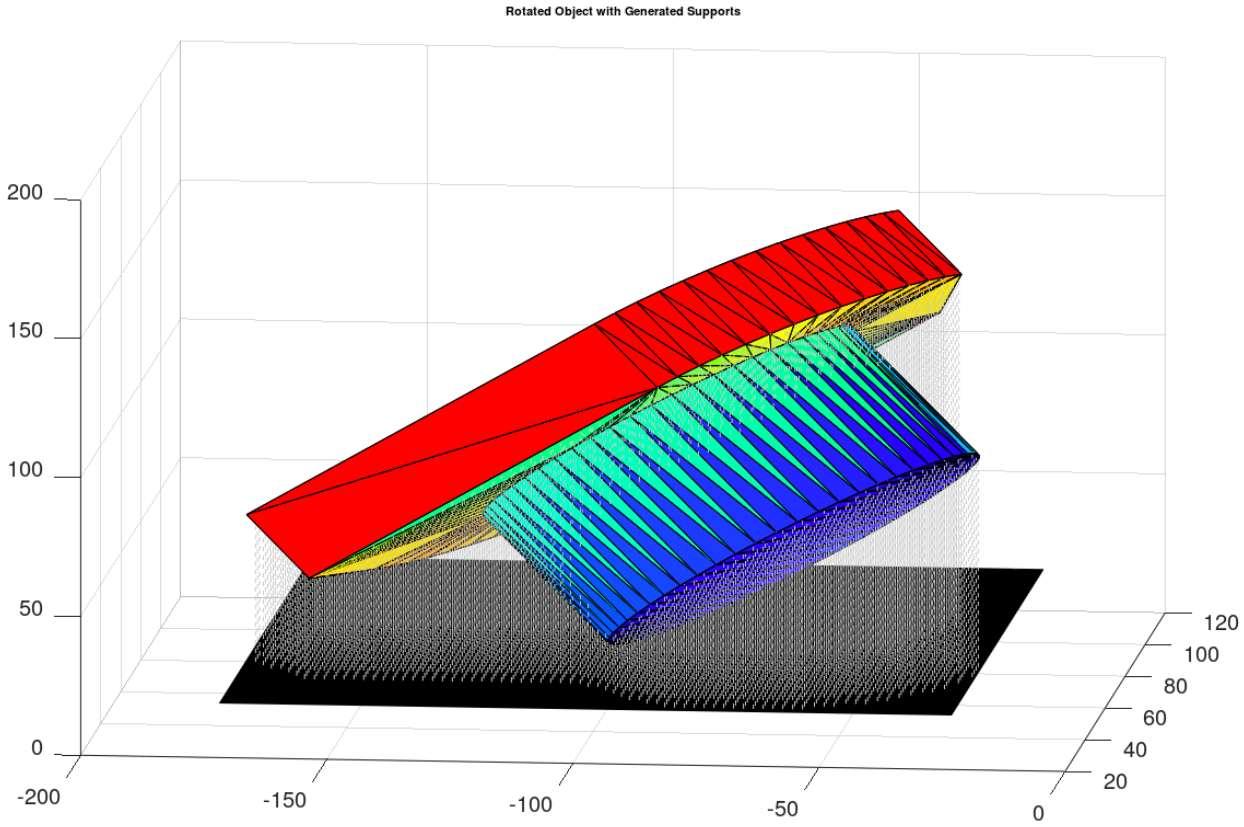


Support structure coloumn volume is the area of the square times the distance between the points of intersection.


Original Object
40
20
0
-20
-40
-100
-50
0
50
100


Rotated Object
120
100
80
60
40
20
-200
-150
-100
-50
0


Rotated Object with Generated Supports
200
150
100
50
0
20
40
60
80
100
120
-200
-150
-100
-50
0


Rotated Object with Generated Supports
200
150
100
50
0
-200
-150
-100
-50
0
120
100
80
60
40
20

```
The volume of the generated supports is 502738.64 mm
octave:1>
```

```
%% Assignment 7
clc, clear all, close all

%% Read the STL file and rotate the part
[F, V, N] = stlread('part.stl');

figure()
patch('vertices', V, 'faces', F, 'facevertexcdata', jet(size(F, 1)), 'facecolor', 'flat')
grid on
title('Original Object')

thX = 30; % [deg]
Rx = [1, 0, 0; 0, cosd(thX), -sind(thX); 0, sind(thX), cosd(thX)]; % rotation matrix abt x-axis
thY = 45; % [deg]
Ry = [cosd(thY), 0 , sind(thY); 0,1,0; -sind(thY) 0, cosd(thY)]; % rotation matrix abt y-axis
R = Rx*Ry;
for i = 1:size(V, 1)
    V(i, :) = V(i, :)*R;
end
for i = 1:size(N, 1)
    N(i, :) = N(i, :)*R;
end

figure()
patch('vertices', V, 'faces', F, 'facevertexcdata', jet(length(F)), 'facecolor', 'flat')
grid on
title('Rotated Object')

%% Create substrate/base plate
Z_LOWER = 10; % [mm]
boundingBox = [
    min(V(:, 1)), max(V(:, 1)); % xLow, xHigh
    min(V(:, 2)), max(V(:, 2)); % yLow, yHigh
    min(V(:, 3))-Z_LOWER, max(V(:, 3)); % zLow - zDrop, zHigh
]';

figure()
ff = [1, 2, 3; 1, 5, 3; 4, 2, 3; 4, 5, 3];
patch('xdata', [boundingBox(1, 1), boundingBox(1, 1), boundingBox(2, 1), boundingBox(2, 1)], 'ydata', [boundingBox(1, 2), boundingBox(2, 2), boundingBox(2, 2), boundingBox(1, 2)], 'zdata', boundingBox(1, 3)*ones(1, 4))
grid off
title('Substrate Layer')

MESH_STEP = 2; % [mm]
[meshX, meshY] = meshgrid(boundingBox(1, 1):MESH_STEP:boundingBox(2, 1), boundingBox(1, 2):MESH_STEP:boundingBox(2, 2));

%% Find the faces that intersect each ray
SUPPORT_ANGLE = 135; % [deg]
BUILD_DIR = [0, 0, 1]; % [0, 0, 1] * [i, j, k]'
if exist('hitFaces.mat', 'file') == 2
    load('hitFaces.mat')
else
```

```
52      hitFaces = zeros(size(meshX, 1), size(meshX, 2), size(F, 1), 2); # dims: [x, y,
    face, face_indx, dist]
53      for i = 1:size(meshX, 1) % meshX and meshY are same size
54          for j = 1:size(meshX, 2)
55              for k = 1:size(F, 1)
56                  vv = V(F(k, :), :); % 3x3
57                  xBounds = [min(vv(:, 1)), max(vv(:, 1))];
58                  yBounds = [min(vv(:, 2)), max(vv(:, 2))];
59
60                  if meshX(i, j) < xBounds(1) || meshX(i, j) > xBounds(2) || meshY(i, j)
    < yBounds(1) || meshY(i, j) > yBounds(2) % ray not inside square bounding box
61                      continue
62                  end
63
64                  A = vv(1, :);
65                  B = vv(2, :);
66                  C = vv(3, :);
67                  xx = [A(1), B(1), C(1)]; % x values
68                  yy = [A(2), B(2), C(2)]; % y values
69                  triangle_area = polyarea(xx, yy); % compute area
70
71                  A = vv(1, :);
72                  B = vv(2, :);
73                  C = [meshX(i, j), meshY(i, j)];
74                  xx = [A(1), B(1), C(1)]; % x values
75                  yy = [A(2), B(2), C(2)]; % y values
76                  triangle_area1 = polyarea(xx, yy); % compute area
77
78                  A = vv(2, :);
79                  B = vv(3, :);
80                  C = [meshX(i, j), meshY(i, j)];
81                  xx = [A(1), B(1), C(1)]; % x values
82                  yy = [A(2), B(2), C(2)]; % y values
83                  triangle_area2 = polyarea(xx, yy); % compute area
84
85                  A = vv(3, :);
86                  B = vv(1, :);
87                  C = [meshX(i, j), meshY(i, j)];
88                  xx = [A(1), B(1), C(1)]; % x values
89                  yy = [A(2), B(2), C(2)]; % y values
90                  triangle_area3 = polyarea(xx, yy); % compute area
91
92                  multiTriangleArea = triangle_area1 + triangle_area2 + triangle_area3;
93                  if abs(multiTriangleArea - triangle_area) > 1e-5 % areas not equal; ray
    not in triangle
94                      continue
95                  end
96                  % hitFaces(i, j, k, 1) = 1;
97
98                  % A = vv(1, :);
99                  % B = vv(2, :);
100                 % C = vv(3, :);
101                 % AB = B - A;
102                 % BC = C - B;
103                 % n = cross(AB, BC);
104                 n = N(k, :);
105                 alpha = dot(BUILD_DIR, n);
106                 while alpha < 0
```

```
                    alpha = alpha + 360;
                end

                buildPlatePt = [meshX(i, j), meshY(i, j), boundingBox(1, 3)];
                d = dot( (A - buildPlatePt), n) / dot(BUILD_DIR, n);
                intersectPt = buildPlatePt + BUILD_DIR * d;
                hitFaces(i, j, k, 2) = intersectPt(3);
                if alpha < 135
                    hitFaces(i, j, k, 1) = 1;
                else
                    hitFaces(i, j, k, 1) = 2;
                end
            end
        end
    end
    save('hitFaces.mat', 'hitFaces')
end

figure(), grid on, hold on
title('Rotated Object with Generated Supports')
patch('xdata', [boundingBox(1, 1), boundingBox(1, 1), boundingBox(2, 1), boundingBox(2, 1)], 'ydata', [boundingBox(1, 2), boundingBox(2, 2), boundingBox(2, 2), boundingBox(1, 2)], 'zdata', boundingBox(1, 3)*ones(1, 4))
patch('vertices', V, 'faces', F, 'facevertexcdata', jet(length(F)), 'facecolor', 'flat')

vol = 0; % [mm]
for i = 1:size(hitFaces, 1)
    for j = 1:size(hitFaces, 2)
        intersectPtIndx = find(hitFaces(i, j, :, 1));
        overhangPtIndx = find(hitFaces(i, j, :, 1) == 2); % 1: face intersection with ray; 2: alpha > 135
        if size(overhangPtIndx, 1) > 0
            for k = 1:size(overhangPtIndx)
                if k == 1
                    bottomZ = boundingBox(1, 3);
                else
                    bottomZ = hitFaces(i, j, intersectPtIndx(k), 2);
                end
            topZ = hitFaces(i, j, overhangPtIndx(k), 2);
            vol = vol + MESH_STEP^2*(topZ - bottomZ);
            plot3(meshX(i,j)*ones(1, 2), meshY(i, j)*ones(1, 2), [bottomZ, topZ], 'color', [.8, .8, .8], 'linestyle', '--')
            end
        end
    end
end
hold off

fprintf('The volume of the generated supports is %.2f mm\n', vol)
```