DEPARTMENT OF MECHANICAL ENGINEERING

ME395 PRACTICUM REPORT

ENDSEM REPORT

DATE OF SUBMISSION-30/04/2025
NAME OF OUR MENTOR:
DR. ANIRBAN BHATTACHARYA

GROUP-10

AYUSH NAGRALE –2201ME01
ADARSH RAJ – 2201ME03
AMAN KUMAR – 2201ME07
SUJIT KUMAR – 2201ME68
ADARSH RAJ – 2201ME78

ME395 – ENGINEERING PRACTICUM REPORT

Name of our Project: Development of a slicing and tool path generation module for Layered Manufacturing.

Objective: Development of an algorithm to generate slicing contours at various heights and corresponding toolpath from CAD model to reconstruct the tessellated model for sheet metal operation.

Aim: The aim of our project is to innovated the medical / healthcare facilities and manufacturing sector by enabling rapid and accurate production of complex geometry. In a critical medical situation such as the patient requires a replacement of a bone structure but due to medical technology limitation could not fulfill the need.

Our project focused on such situation by enabling rapid and accurate reconstruction of complex anatomical parts from CAD data. Similarly in manufacturing, bulk production of an intricated geometry typically requires significant time, labor force and repeated setups.

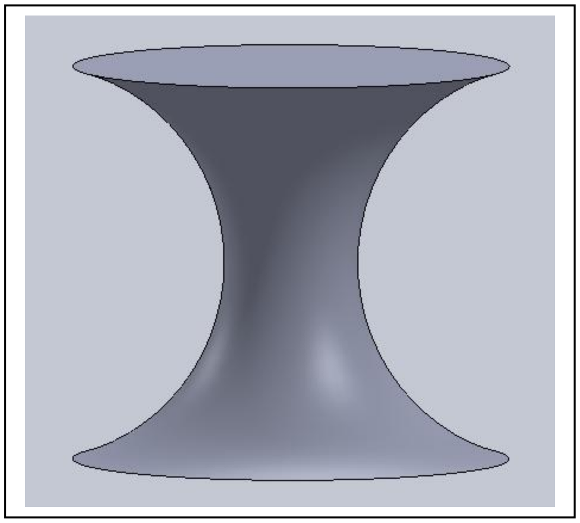
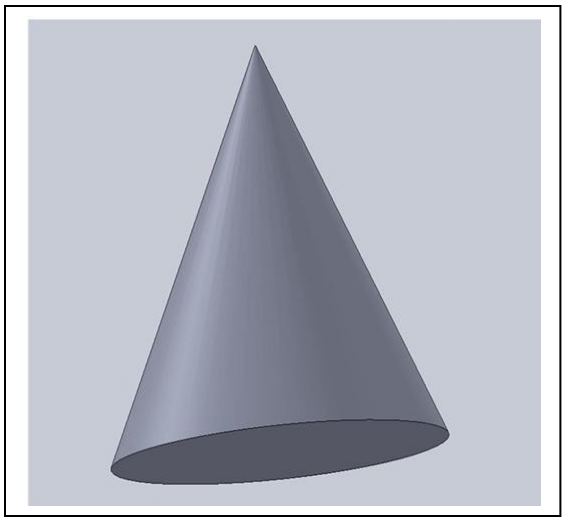
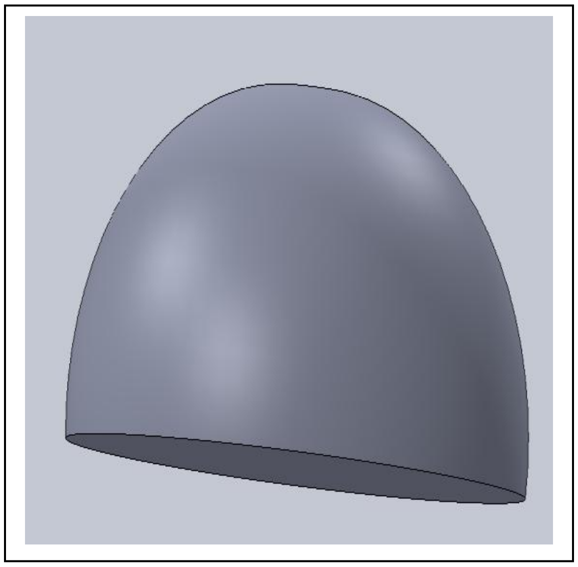
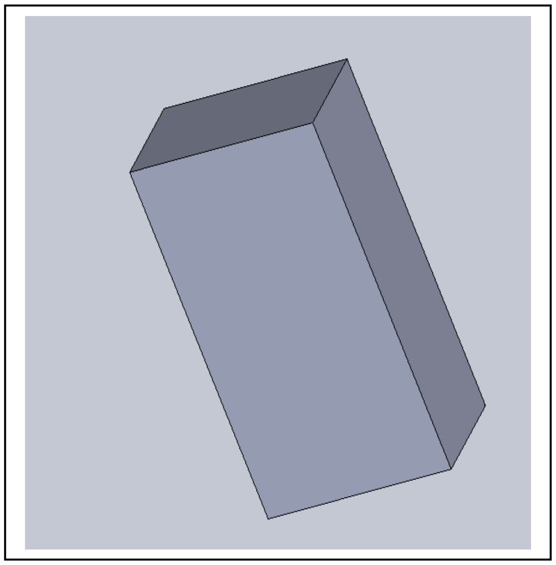
Our algorithm streamlines this by requiring only single time input which can be reused for bulk production without any manual intervention.

Key features such as adjustable step sizing , surface extraction , tool radius compensation and intersection based toolpath generation ensure high precision and adaptability making the model suitable for individual and Industry production.

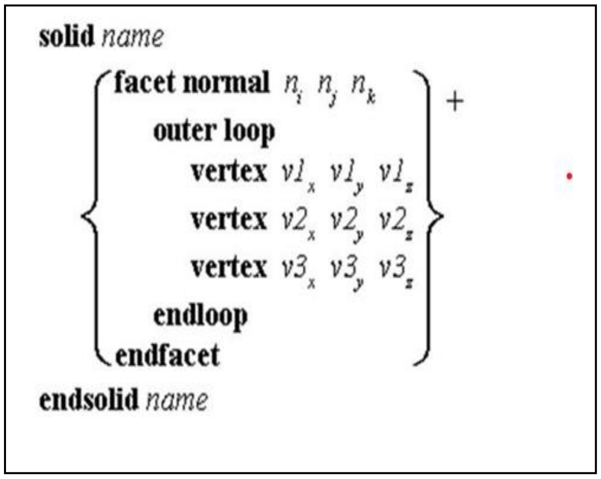
<u>PROJECT WORKFLOW</u>

1)<u>CAD Model Generation :</u> This is the first step which involves using Computer Aided Software (CAD software) to create a 3D model. The generated CAD model is suitable for layer by layer manufacturing.
It acts as a blueprint for the whole process, so it needs to be accurate and practical for production.

CAD models used…

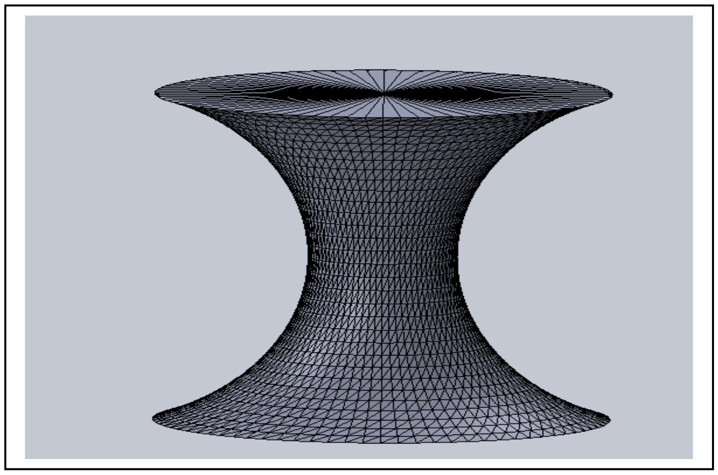
2) <u>Conversion to STL format :</u> The Generated Cad model is save in STl format. The STL format captures the surface geometry of a 3D object by breaking it down into a mesh of triangular facets. Each of these triangles is defined by its three corner points (vertices) and a normal vector, which together describe both the shape and the orientation of each part of the object's surface.



solid *name*

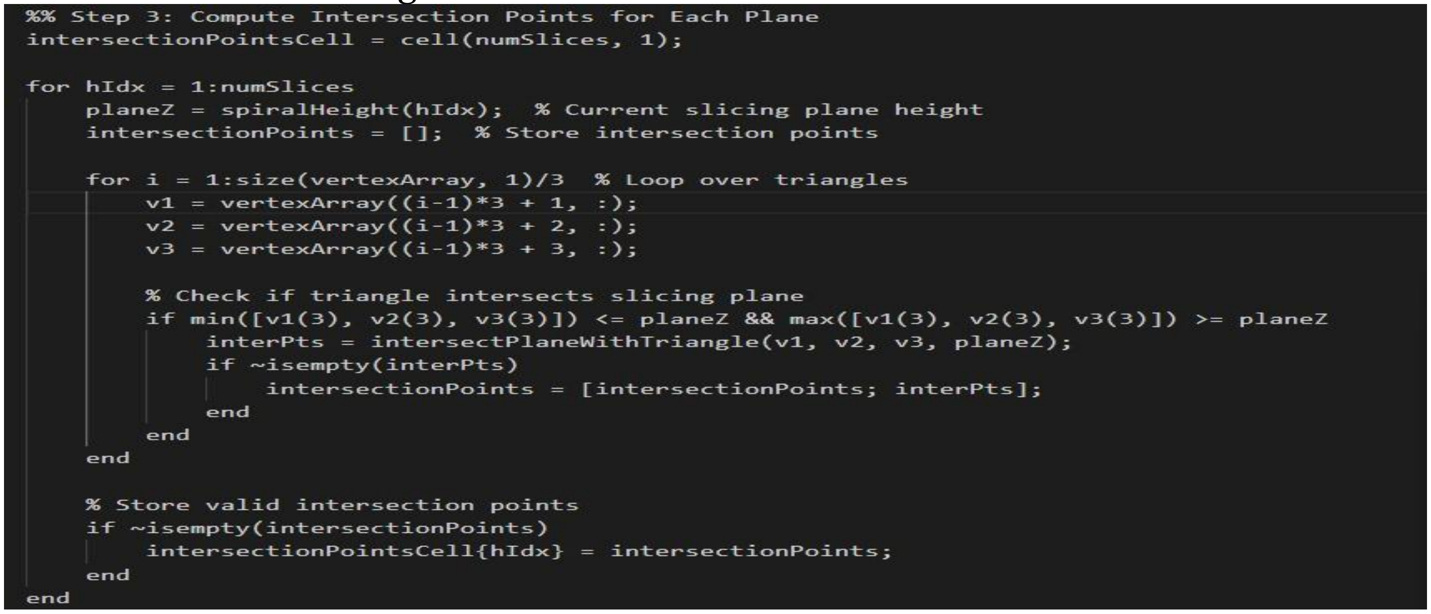
$$\left\{\begin{array}{l}\textbf{facet normal } n_i\ n_j\ n_k \\ \quad \textbf{outer loop} \\ \qquad \textbf{vertex } v1_x\ v1_y\ v1_z \\ \qquad \textbf{vertex } v2_x\ v2_y\ v2_z \\ \qquad \textbf{vertex } v3_x\ v3_y\ v3_z \\ \quad \textbf{endloop} \\ \textbf{endfacet}\end{array}\right\}^{+}$$

endsolid *name*



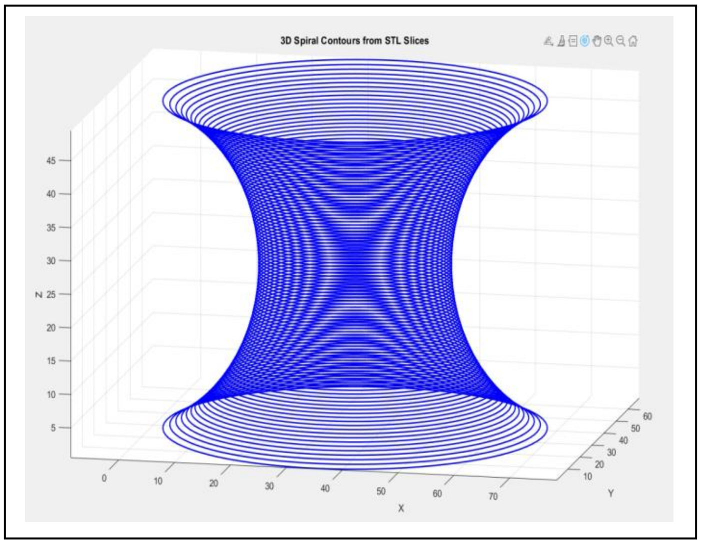
3) <u>Reconstruction in MATLAB :</u> For further processing the STL file is imported in MATLAB. Model geometry is constructed using the triangle vertices and normal extracted from STL format. These elements are essential because they define the shape and orientation of each triangular facet that makes up the surface of the 3D model. By plotting the vertices and normal in MATLAB, you can generate a visual representation of the object.

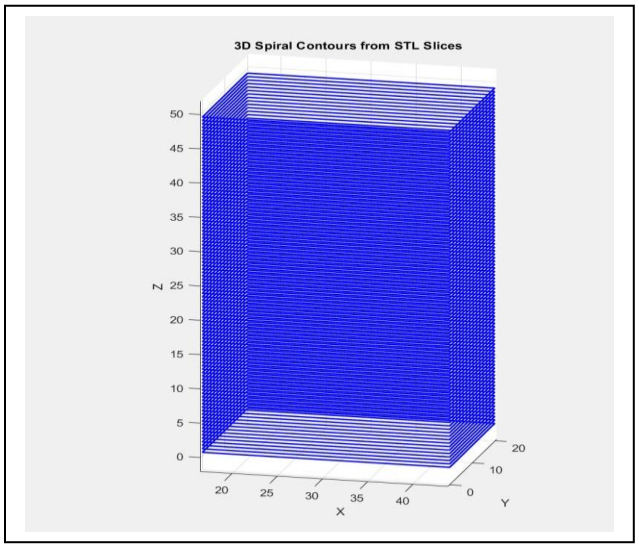
4) <u>Slicing process:</u> In LM, slicing of the CAD model of a part to be produced is one of the important steps. Slicing is the process in layered manufacturing (LM) where a 3D CAD model is divided into a series of horizontal layers or slices. These slices guide the layer-by-layer construction of a part. The thickness of each slice affects both the build time and surface finish, making slicing a critical step in achieving a balance between manufacturing efficiency and part quality.
MATLAB code for slicing....

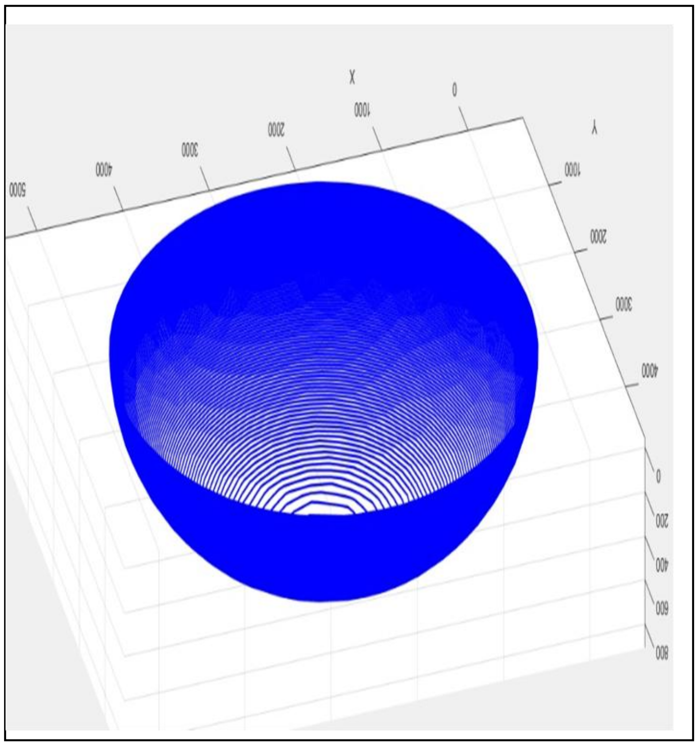


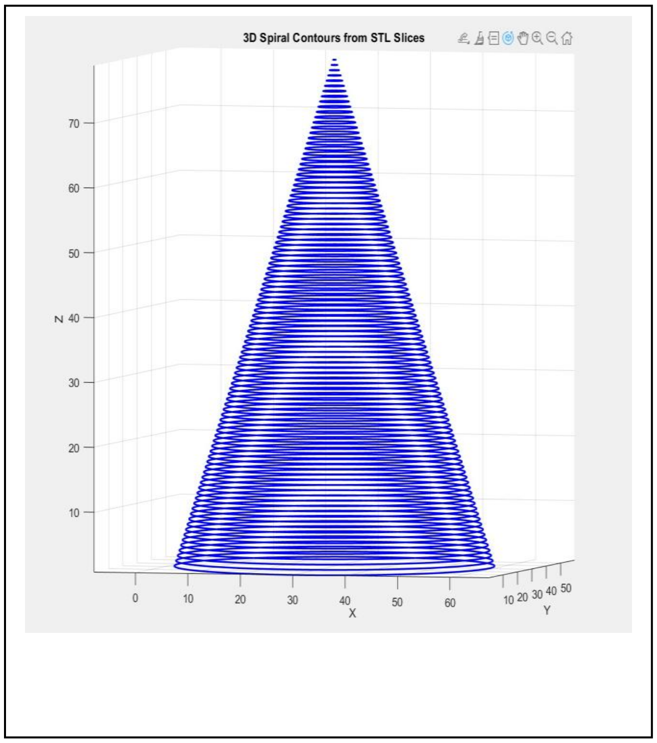
```
%% Step 3: Compute Intersection Points for Each Plane
intersectionPointsCell = cell(numSlices, 1);

for hIdx = 1:numSlices
    planeZ = spiralHeight(hIdx);  % Current slicing plane height
    intersectionPoints = [];  % Store intersection points

    for i = 1:size(vertexArray, 1)/3  % Loop over triangles
        v1 = vertexArray((i-1)*3 + 1, :);
        v2 = vertexArray((i-1)*3 + 2, :);
        v3 = vertexArray((i-1)*3 + 3, :);

        % Check if triangle intersects slicing plane
        if min([v1(3), v2(3), v3(3)]) <= planeZ && max([v1(3), v2(3), v3(3)]) >= planeZ
            interPts = intersectPlaneWithTriangle(v1, v2, v3, planeZ);
            if ~isempty(interPts)
                intersectionPoints = [intersectionPoints; interPts];
            end
        end
    end

    % Store valid intersection points
    if ~isempty(intersectionPoints)
        intersectionPointsCell{hIdx} = intersectionPoints;
    end
end
```

Sliced contour at different heights..





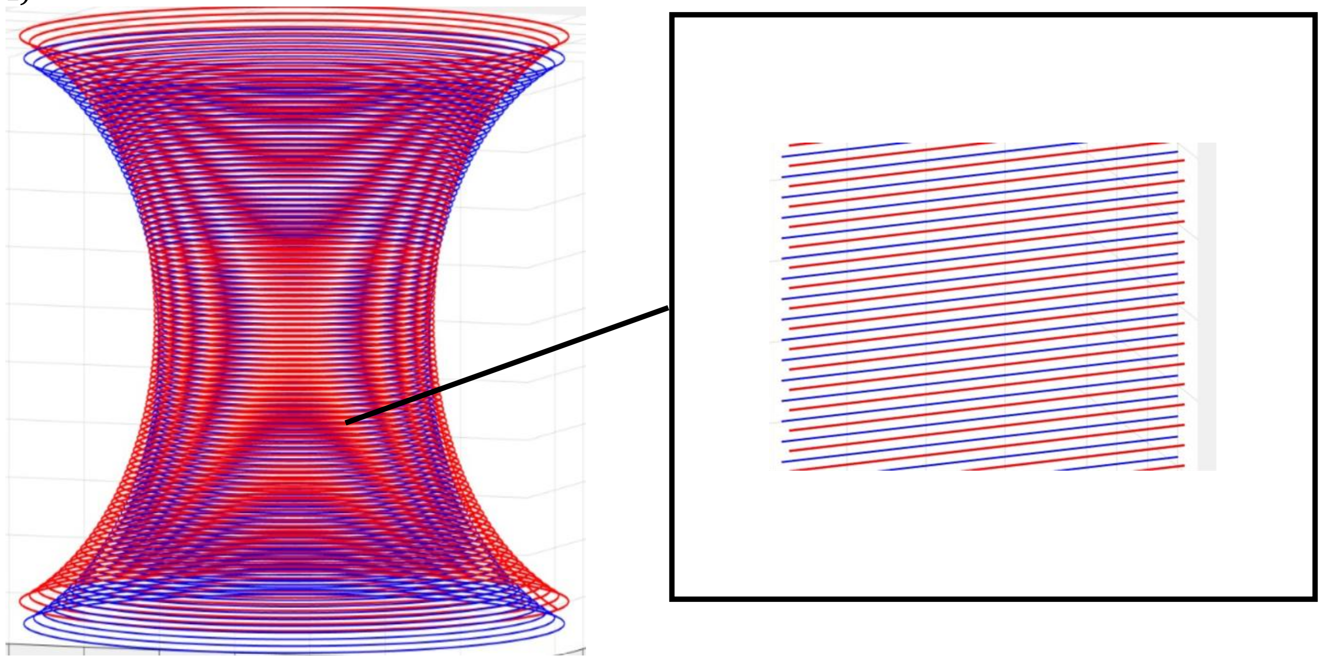


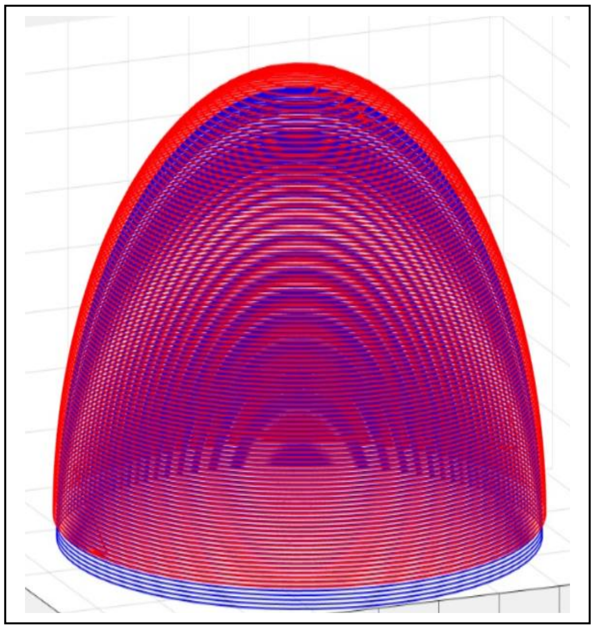


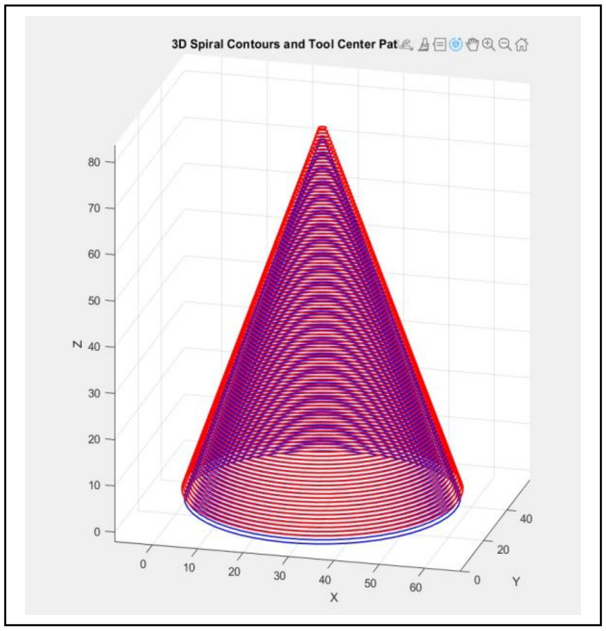
4) Toolpath Generation: This step involves creating a efficient path for deposition of material, guiding the fabrication tool as it follows the contours of each slice. Smooth deposition is ensure by optimize toolpath for reduced time, material wastage. The generated tool path is then used to program the LM machine, ensuring that each layer is fabricated precisely according to the sliced contours.

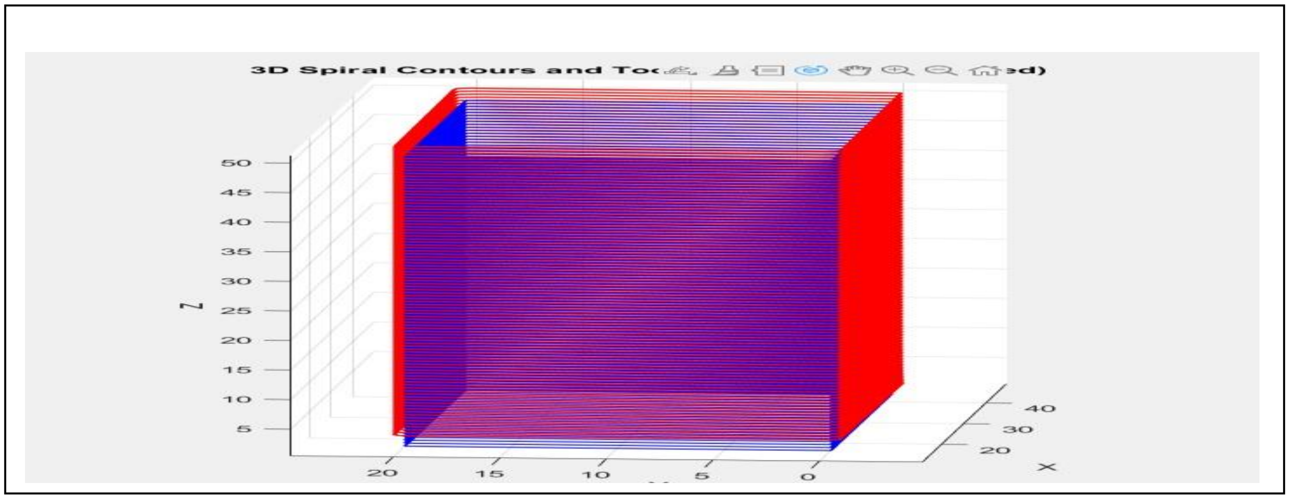
Considering,
Tool radius= 2mm
Red contour=bottom point contour
Blue contour = contact point contour

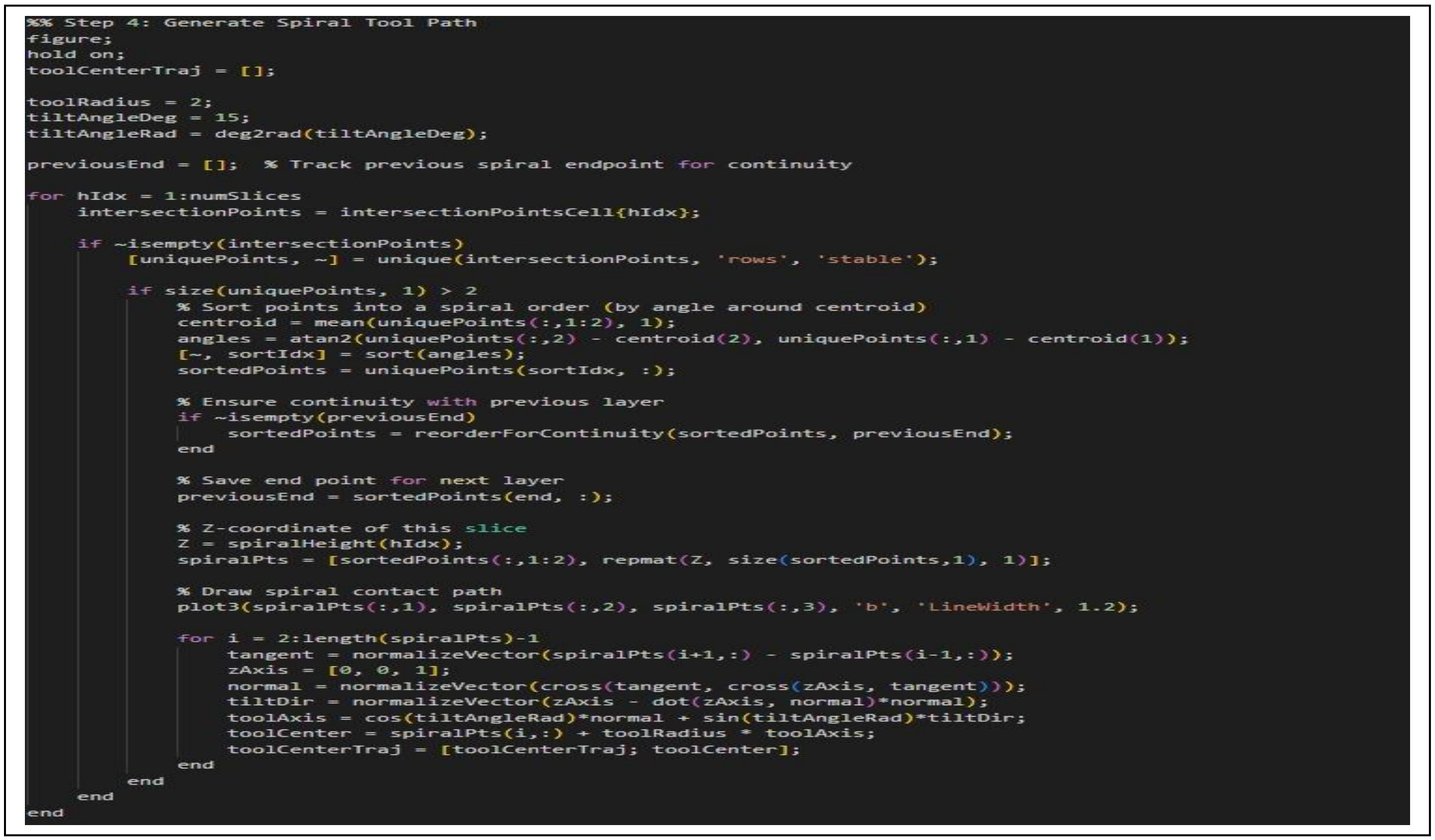
1)











```
%% Step 4: Generate Spiral Tool Path
figure;
hold on;
toolCenterTraj = [];

toolRadius = 2;
tiltAngleDeg = 15;
tiltAngleRad = deg2rad(tiltAngleDeg);

previousEnd = [];  % Track previous spiral endpoint for continuity

for hIdx = 1:numSlices
    intersectionPoints = intersectionPointsCell{hIdx};

    if ~isempty(intersectionPoints)
        [uniquePoints, ~] = unique(intersectionPoints, 'rows', 'stable');

        if size(uniquePoints, 1) > 2
            % Sort points into a spiral order (by angle around centroid)
            centroid = mean(uniquePoints(:,1:2), 1);
            angles = atan2(uniquePoints(:,2) - centroid(2), uniquePoints(:,1) - centroid(1));
            [~, sortIdx] = sort(angles);
            sortedPoints = uniquePoints(sortIdx, :);

            % Ensure continuity with previous layer
            if ~isempty(previousEnd)
                sortedPoints = reorderForContinuity(sortedPoints, previousEnd);
            end

            % Save end point for next layer
            previousEnd = sortedPoints(end, :);

            % Z-coordinate of this slice
            Z = spiralHeight(hIdx);
            spiralPts = [sortedPoints(:,1:2), repmat(Z, size(sortedPoints,1), 1)];

            % Draw spiral contact path
            plot3(spiralPts(:,1), spiralPts(:,2), spiralPts(:,3), 'b', 'LineWidth', 1.2);

            for i = 2:length(spiralPts)-1
                tangent = normalizeVector(spiralPts(i+1,:) - spiralPts(i-1,:));
                zAxis = [0, 0, 1];
                normal = normalizeVector(cross(tangent, cross(zAxis, tangent)));
                tiltDir = normalizeVector(zAxis - dot(zAxis, normal)*normal);
                toolAxis = cos(tiltAngleRad)*normal + sin(tiltAngleRad)*tiltDir;
                toolCenter = spiralPts(i,:) + toolRadius * toolAxis;
                toolCenterTraj = [toolCenterTraj; toolCenter];
            end
        end
    end
end
```

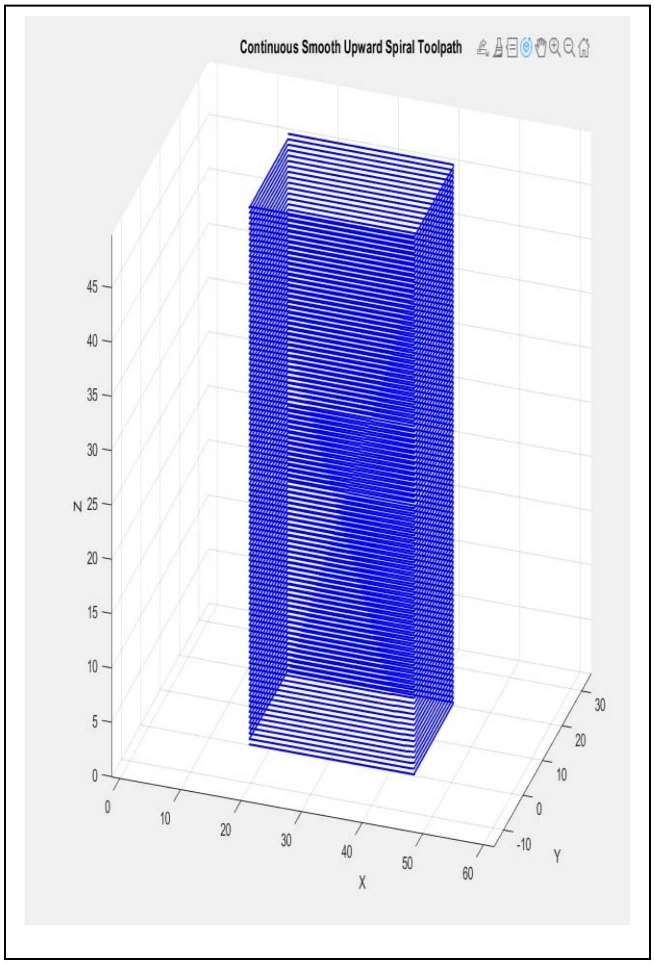
5) **<u>Spiral path generation:</u>** A 3D spiral (helix) toolpath is created by linearly interpolating between corresponding points on successive contour slices. This is done for all contour pairs to form a continuous spiral path. The interpolation uses a weighted average based on arc length between points.
The offset curves play a key role as the guidelines for generating spiral path. A spiral path is generated on the sculptured surface as a diagonal curve between two offset curves. The jth point of the kth spiral path segment is calculated by the equation given below..

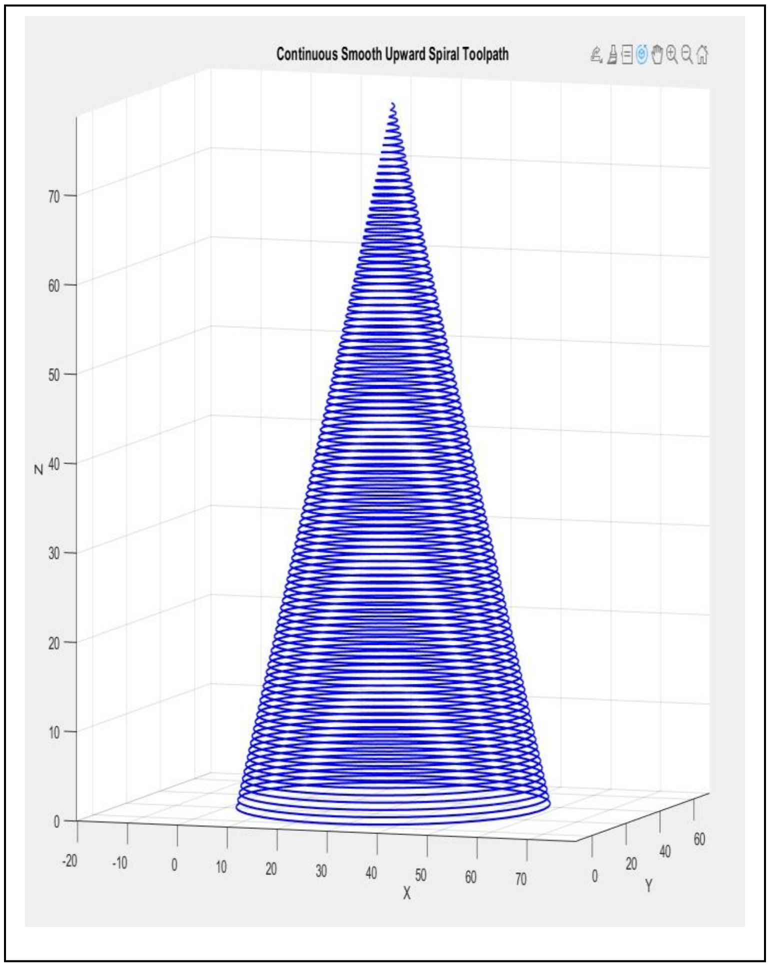
$$x_j = s_j p_j + (1 - s_j) p_j^0$$

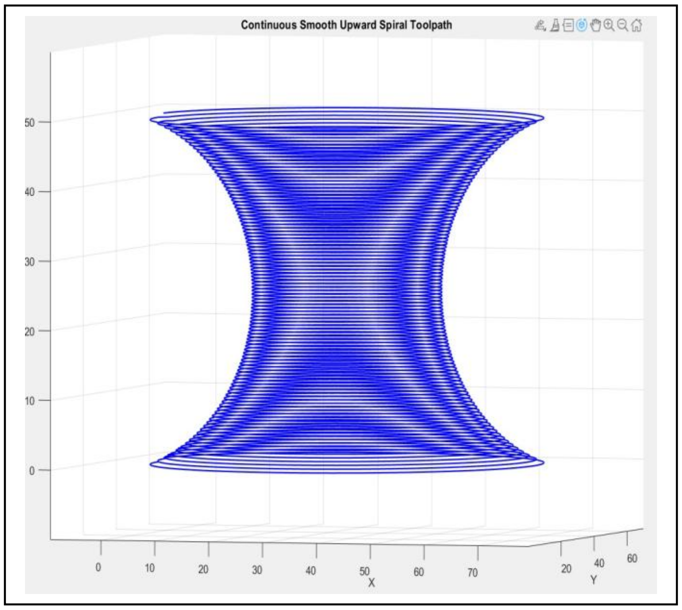
where

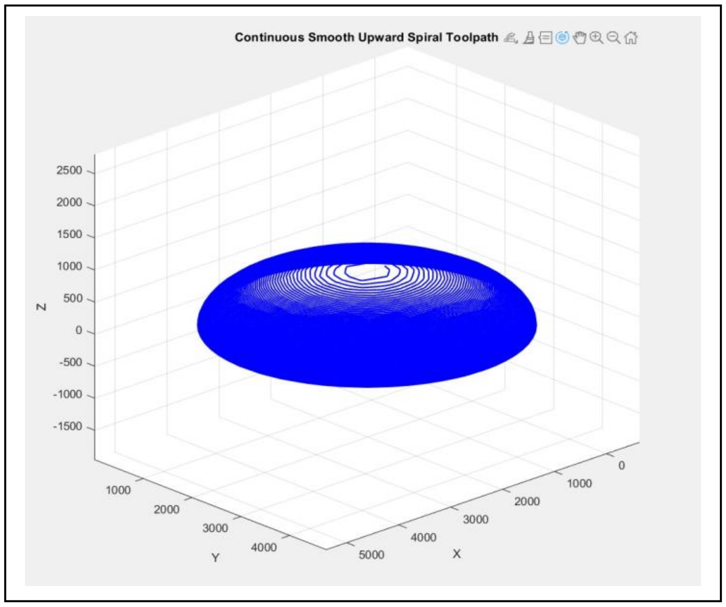
$$s_j = \frac{\sum_{l=1}^{j-i} |p_{l+1} - p_l|}{\sum_{l=1}^{N-1} |p_{l+1} - p_l|}$$

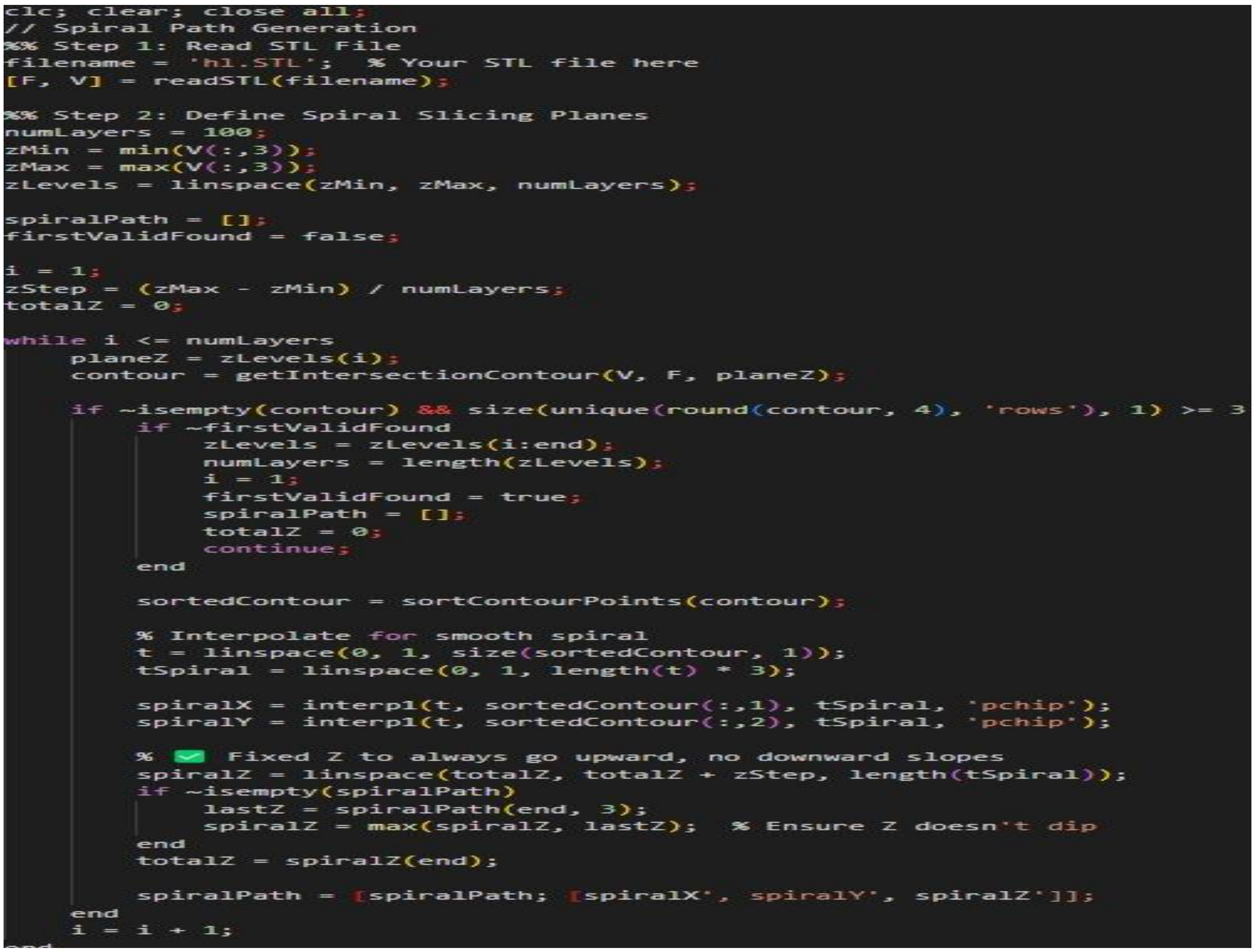
where p_l is the lth point on the ith slice, p_l^0 is the lth point on the $(i-1)$th slice, and N is the number of points on each contour slice.





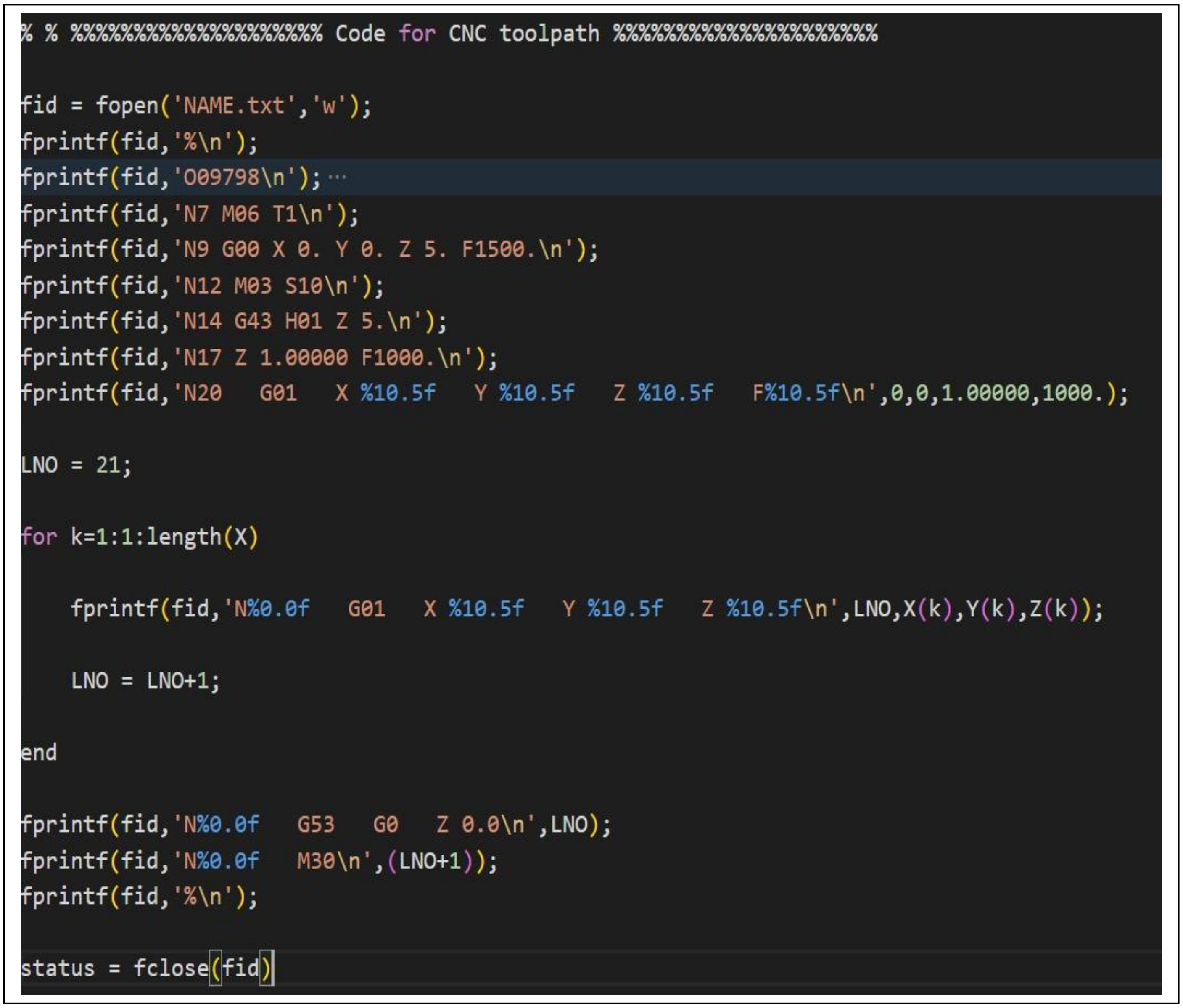






```
clc; clear; close all;
// Spiral Path Generation
%% Step 1: Read STL File
filename = 'h1.STL';   % Your STL file here
[F, V] = readSTL(filename);

%% Step 2: Define Spiral Slicing Planes
numLayers = 100;
zMin = min(V(:,3));
zMax = max(V(:,3));
zLevels = linspace(zMin, zMax, numLayers);

spiralPath = [];
firstValidFound = false;

i = 1;
zStep = (zMax - zMin) / numLayers;
totalZ = 0;

while i <= numLayers
    planeZ = zLevels(i);
    contour = getIntersectionContour(V, F, planeZ);

    if ~isempty(contour) && size(unique(round(contour, 4), 'rows'), 1) >= 3
        if ~firstValidFound
            zLevels = zLevels(i:end);
            numLayers = length(zLevels);
            i = 1;
            firstValidFound = true;
            spiralPath = [];
            totalZ = 0;
            continue;
        end

        sortedContour = sortContourPoints(contour);

        % Interpolate for smooth spiral
        t = linspace(0, 1, size(sortedContour, 1));
        tSpiral = linspace(0, 1, length(t) * 3);

        spiralX = interp1(t, sortedContour(:,1), tSpiral, 'pchip');
        spiralY = interp1(t, sortedContour(:,2), tSpiral, 'pchip');

        % ✅ Fixed Z to always go upward, no downward slopes
        spiralZ = linspace(totalZ, totalZ + zStep, length(tSpiral));
        if ~isempty(spiralPath)
            lastZ = spiralPath(end, 3);
            spiralZ = max(spiralZ, lastZ);   % Ensure Z doesn't dip
        end
        totalZ = spiralZ(end);

        spiralPath = [spiralPath; [spiralX', spiralY', spiralZ']];
    end
    i = i + 1;
```

6)Creation of GM code-



```
% % %%%%%%%%%%%%%%%%%%%% Code for CNC toolpath %%%%%%%%%%%%%%%%%%%%%

fid = fopen('NAME.txt','w');
fprintf(fid,'%\n');
fprintf(fid,'O09798\n');...
fprintf(fid,'N7 M06 T1\n');
fprintf(fid,'N9 G00 X 0. Y 0. Z 5. F1500.\n');
fprintf(fid,'N12 M03 S10\n');
fprintf(fid,'N14 G43 H01 Z 5.\n');
fprintf(fid,'N17 Z 1.00000 F1000.\n');
fprintf(fid,'N20   G01   X %10.5f   Y %10.5f   Z %10.5f   F%10.5f\n',0,0,1.00000,1000.);

LNO = 21;

for k=1:1:length(X)

    fprintf(fid,'N%0.0f   G01   X %10.5f   Y %10.5f   Z %10.5f\n',LNO,X(k),Y(k),Z(k));

    LNO = LNO+1;

end

fprintf(fid,'N%0.0f   G53   G0   Z 0.0\n',LNO);
fprintf(fid,'N%0.0f   M30\n',(LNO+1));
fprintf(fid,'%\n');

status = fclose(fid)
```

Conclusion-

This project successfully developed a practical system to convert 3D designs into instructions for machines to build objects layer by layer. Our work helps in two important ways: first, by enabling fast production of custom medical parts like bone replacements, and second, by simplifying the manufacturing of complex industrial components.

The key achievement is creating a process that automatically turns CAD models into precise cutting paths, while adjusting for tool size and optimizing material use. This saves both time and costs compared to traditional methods. The system works reliably for various shapes and can be reused for mass production without extra setup.

Looking ahead, there's potential to handle even more complex designs and connect this technology with smart factories for better efficiency. Our results show how smart engineering solutions can make manufacturing more accessible and responsive, particularly for urgent needs in healthcare and industry.

References-

https://www.researchgate.net/publication/263212613_Analysis_of_STL_files

https://www.researchgate.net/publication/242341408_Efficient_slicing_for_layered_manufacturing

https://asmedigitalcollection.asme.org/manufacturingscience/article/132/6/061003/433501/Automatic-3D-Spiral-Toolpath-Generation-for-Single

https://www.researchgate.net/publication/241441853_Slicing_procedures_in_layered_manufacturing_A_review

Contour offset approach to spiral tool path generation with constant scallop height by Eungki Lee.