### 3.5 3D Vector Algebra Method in Analysis of Parabolic Plane

For the two-dimensional unfolding calculation of the parabolic dish antenna, a MATLAB script was developed. Its core idea is to divide the paraboloid into multiple discrete planar layers to approximate the curved surface. First, the paraboloid is divided into $n_Y$ horizontal layers; within each layer, an $n$-sided polygon (which can be adjusted to refine the fitting accuracy) is inscribed. Each layer is illustrated in Figure 17.
The objective is to unfold the parabolic dish into a three-dimensional petal-shaped layout.

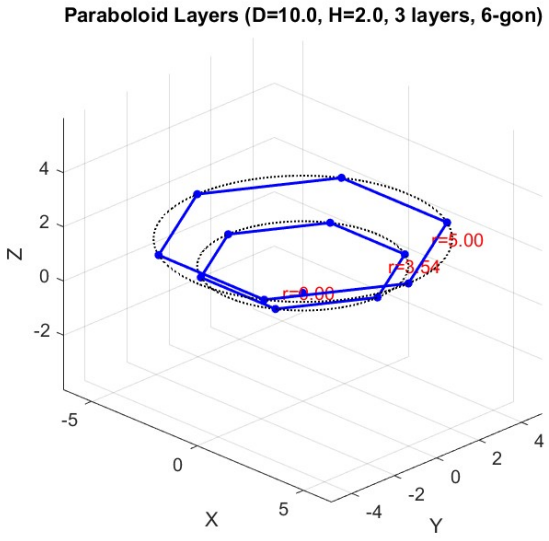



Figure 17. MATLAB illustration of the paraboloid surface whose continuous height layers are discretized and subsequently approximated by polygons.

The generatrix of the paraboloid is described by the parabola $y = a\,x^2$. First, an array

$$\mathbf{y} \;=\; \left[y_1,\, y_2,\, \ldots,\, y_{n_Y}\right]^{\mathrm{T}}$$

is defined to partition the height into $n_Y$ levels. For each level $i$, the corresponding radius $r_i$ satisfies

$$y_i \;=\; a\,r_i^2,$$

so that the array $\{r_i\}$ contains the radius of each layer.

Next, the arc length of the parabola must be computed for each layer in order to determine the edge length of each petal. This is done via the arc length formula in Equation (2), which yields the length corresponding to radius $r_i$. These lengths are used later for vector-space operations.
Afterwards, the coordinates of each vertex on the approximating polygons are computed using polar coordinates. For the $j$-th vertex in layer $i$, the polar angle is given by

$$\theta_j \;=\; \frac{2\pi\,(j-1)}{n_{\mathrm{Pieces}}}, \qquad j = 1, 2, \ldots, n_{\mathrm{Pieces}}.$$

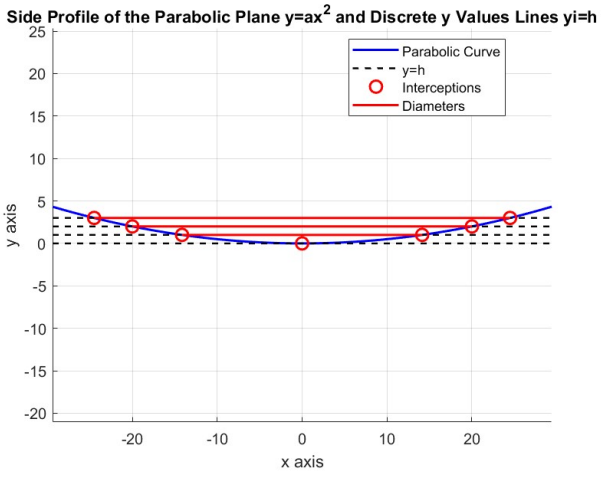



Figure 18. MATLAB illustration of the paraboloid surface whose continuous height layers are discretized and subsequently approximated by polygons.

Each vertex $(x_{i,j},\, y_{i,j},\, z_{i,j})$ is then obtained by applying Equation (3) in a loop. Here, the $z$-coordinate is simply the height $y_i$, and all points are stored in a three-dimensional array $P$ of size $n_Y \times n_{\mathrm{Pieces}} \times 3$.

$$\mathbf{L} \;=\; \big[L(r_1),\, L(r_2),\, \dots\big]^{\mathrm{T}}, \qquad L(r_i) \;=\; \int_0^{r_i} \sqrt{1 + \big(2\,a\,x\big)^2}\;\mathrm{d}x. \tag{2}$$

$$\begin{pmatrix} x_{i,j} \\ y_{i,j} \\ z_{i,j} \end{pmatrix} = \begin{pmatrix} r_i\,\cos\!\big(\theta_j\big) \\ r_i\,\sin\!\big(\theta_j\big) \\ y_i \end{pmatrix}. \tag{3}$$

Once the coordinates of every point on the parabolic dish have been computed, the normal vector $\mathbf{n}$ of each face can be determined. Two types of faces are considered: the ring of triangles adjacent to the dish' s center (shown in red in Figure 20a) and the trapezoidal faces in each outer layer (shown in green in Figure 20b). For triangular faces, their normals are calculated using the three vertices $P_a$, $P_b$, $P_c$; for trapezoidal faces, any three of the four vertices suffice to compute the normal.

$$\mathbf{n} = \frac{\big(P_b - P_a\big) \times \big(P_c - P_a\big)}{\big\|\big(P_b - P_a\big) \times \big(P_c - P_a\big)\big\|}. \tag{4}$$

After obtaining the normal vector $\mathbf{n}$ for each face, the dihedral angle between adjacent faces is computed. Knowing these angles is essential for 3D modeling and for designing the 4D-printed hinges between petals, which "program" the desired recovery behavior based on data predicted by a neural network.

The angle between two adjacent faces is given by Equation (5), which calculates the angle between their normal vectors. This computation is implemented in a function:

```
function [normal] = planeNormal(A, B, C)
```

Given three points $(A, B, C)$ on a plane, the function applies the cross product in Equation (4) to produce the normal vector `normal` (a $3 \times 1$ vector). This function is then called within loops as follows:

- A loop for $k = 1, \dots, N_{\text{lev}} - 2$ computes the angle between vertically adjacent trapezoidal faces, radiating outward from the center, as shown in Figure 20a.
- Another loop, also radiating from the center, computes the angle between horizontally adjacent trapezoidal faces, as shown in Figure 20(b).
- Finally, the angle between triangular face normals and the horizontal plane normal $[0, 0, 1]$ is computed using the same function, as illustrated in Figure 20(c).

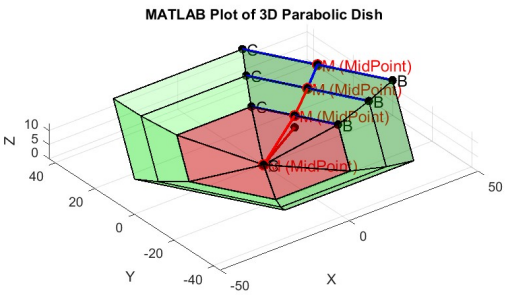
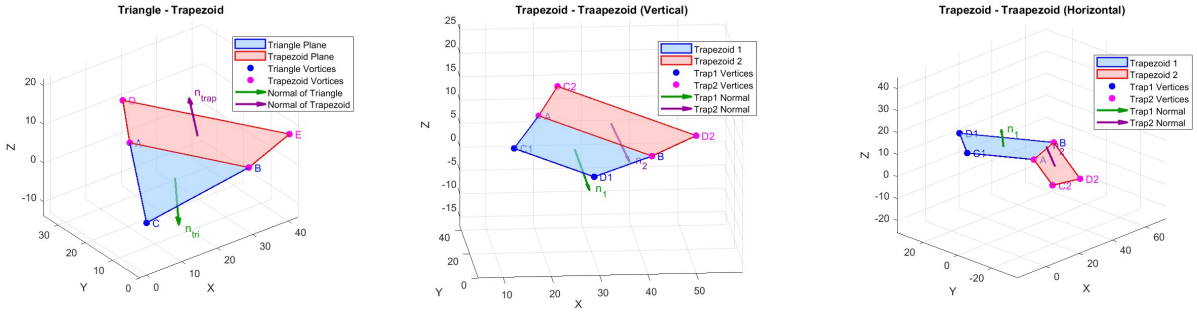



Figure 19. Plot of the paraboloid discretized into three vertical segments and six petals.

Given three points $(A, B, C)$ on a plane, the function applies the cross product in Equation (4) to produce the normal vector 'normal' (a $3 \times 1$ vector). This function is then called within loops as follows:

- A loop for $k = 1, \dots, N_{\text{lev}} - 2$ computes the angle between vertically adjacent trapezoidal faces, radiating outward from the center, as shown in Figure 20a.

- Another loop, also radiating from the center, computes the angle between horizontally adjacent trapezoidal faces, as shown in Figure 20(b).
- Finally, the angle between triangular face normals and the horizontal plane normal [0, 0, 1] is computed using the same function, as illustrated in Figure 20(c).




**(a)** Angle between triangle and adjacent trapezoid **(b)** Angle between vertically adjacent trapezoids **(c)** Angle between trapezoid and horizontal plane

**Figure 20.** Use of plane normal vector method to find the dihedral angles between faces

$$\cos\beta_k = \frac{\mathbf{n}_{\text{trap}}^{(k)} \cdot \mathbf{n}_{\text{trap}}^{(k+1)}}{\left\|\mathbf{n}_{\text{trap}}^{(k)}\right\| \left\|\mathbf{n}_{\text{trap}}^{(k+1)}\right\|}, \qquad \beta_k = \arccos\!\left(\frac{\mathbf{n}_{\text{trap}}^{(k)} \cdot \mathbf{n}_{\text{trap}}^{(k+1)}}{\left\|\mathbf{n}_{\text{trap}}^{(k)}\right\| \left\|\mathbf{n}_{\text{trap}}^{(k+1)}\right\|}\right) \times \frac{180}{\pi}. \tag{5}$$

### 3.5.1 Validation Using Analytic Geometry Methods

At the same time, an alternative geometric method is employed to verify the correctness of the model. Since each "petal" is a rotationally symmetric figure, the dihedral angle between any two adjacent faces is equal to the angle between their profiles in the plane of symmetry. Therefore, by identifying the axis of symmetry for each petal, one can compute the angle between a triangular face and the horizontal plane, the angle between a triangular face and an adjacent trapezoidal face, and the angle between two adjacent trapezoidal faces.

Consequently, two functions were defined:

- `function [M] = midpoint(C, B)`: Given two points $C$ and $B$, this function returns the midpoint $M$.
- `function [theta_deg] = angle(M, D, E)`: Given three points $M$, $D$, and $E$, this function computes the angle (in degrees) between the line segments $MD$ and $ME$, returning $\theta_{\text{deg}}$.

These functions are called within a loop to calculate each dihedral angle between adjacent faces on a single petal.The results obtained via this "midline method" precisely match those from the plane-normal vector method described earlier, confirming the validity of the angle computation algorithm.

Another function, `function [polyL] = polygonSide(nPoly, r)`, was created to calculate the side length of a regular $n$-sided polygon inscribed in a circle of radius $r$. The output `polyL` is then compared to the length computed directly from vertex coordinates stored in `nPts`. Because the results coincide with the vector-based distance between adjacent points in `nPts`, this comparison serves as an additional check for the correctness of the lateral-angle calculations.

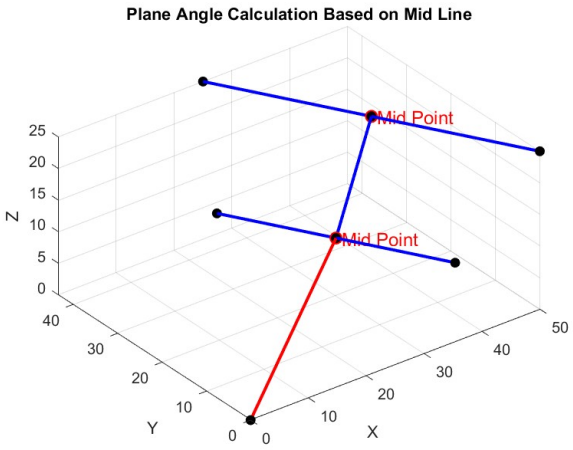



Figure 21. Plot of a paraboloid discretized with three vertical segments and six petals.

### 3.5.2 3D-to-2D Coordinate Projection Algorithm

Once the 3D parameters of the model are established, the next step is to project each face from the three-dimensional coordinate system onto a two-dimensional plane for CAD usage. To that end, the function

`function [Pts2D] = trep(P1, P2, P3, P4, f)`

is defined. It takes as input the four vertices $P_1, P_2, P_3, P_4$ of a trapezoidal face and a translation vector $f$ that specifies where the trapezoid should be placed in the 2D plane. The output `Pts2D` is a structure containing the four projected 2D coordinates.

Inside `trep`, a local coordinate system is constructed on the trapezoid' s plane:

$$\mathbf{u} = \frac{\mathbf{P}_2 - \mathbf{P}_1}{\|\mathbf{P}_2 - \mathbf{P}_1\|}, \qquad \mathbf{n} = \frac{(\mathbf{P}_2 - \mathbf{P}_1) \times (\mathbf{P}_3 - \mathbf{P}_1)}{\|(\mathbf{P}_2 - \mathbf{P}_1) \times (\mathbf{P}_3 - \mathbf{P}_1)\|}, \qquad \mathbf{v} = \frac{\mathbf{n} \times \mathbf{u}}{\|\mathbf{n} \times \mathbf{u}\|}.$$

Here, $\mathbf{u}$ and $\mathbf{v}$ are orthonormal basis vectors lying in the trapezoid' s plane, and $\mathbf{n}$ is the unit normal to that plane. These vectors form a projection matrix $R$ defined as:

$$R = \begin{bmatrix} \mathbf{u}^T \\ \mathbf{v}^T \end{bmatrix} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \end{pmatrix}.$$

Next, the coordinates of the trapezoid' s four vertices are shifted so that $P_1$ becomes the origin:

$$P - P_1\mathbf{1} = \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ z_1 & z_2 & z_3 & z_4 \end{pmatrix} - \begin{pmatrix} x_1 & x_1 & x_1 & x_1 \\ y_1 & y_1 & y_1 & y_1 \\ z_1 & z_1 & z_1 & z_1 \end{pmatrix} = \begin{pmatrix} 0 & x_2 - x_1 & x_3 - x_1 & x_4 - x_1 \\ 0 & y_2 - y_1 & y_3 - y_1 & y_4 - y_1 \\ 0 & z_2 - z_1 & z_3 - z_1 & z_4 - z_1 \end{pmatrix}.$$

By left-multiplying this with the projection matrix $R$, the raw projected coordinates in the plane are obtained:

$$Q_{\text{raw}} = R\,(P - P_1\mathbf{1}) = \begin{pmatrix} u_1^{(\text{raw})} & u_2^{(\text{raw})} & u_3^{(\text{raw})} & u_4^{(\text{raw})} \\ v_1^{(\text{raw})} & v_2^{(\text{raw})} & v_3^{(\text{raw})} & v_4^{(\text{raw})} \end{pmatrix}.$$

Finally, to place the trapezoid at the correct position $\mathbf{f} = (f_u,\ f_v)$ in 2D, each column of $Q_{\text{raw}}$ is translated:

$$\begin{pmatrix} u_i^{(\text{final})} \\ v_i^{(\text{final})} \end{pmatrix} = \begin{pmatrix} u_i^{(\text{raw})} \\ v_i^{(\text{raw})} \end{pmatrix} + \begin{pmatrix} f_u \\ f_v \end{pmatrix} - R\big(P_i - P_1\big), \quad i = 1, 2, 3, 4.$$

The projection function for a triangular face follows a similar procedure.

Using a `for` loop, the triangular and trapezoidal plot functions are called successively for each layer of the petal, from the innermost to the outermost. The result for a single petal is shown in Figure 22.

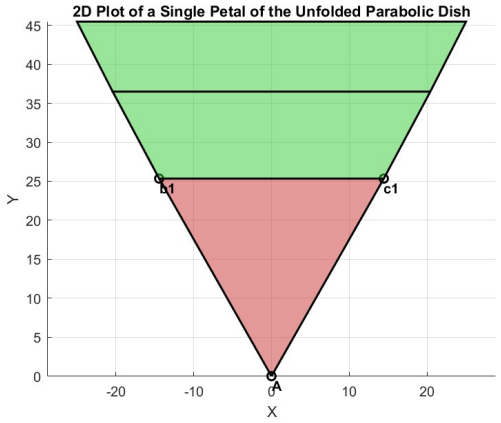



**Figure 22.** 2D Plot of a Single Petal of the Unfolded Parabolic Dish

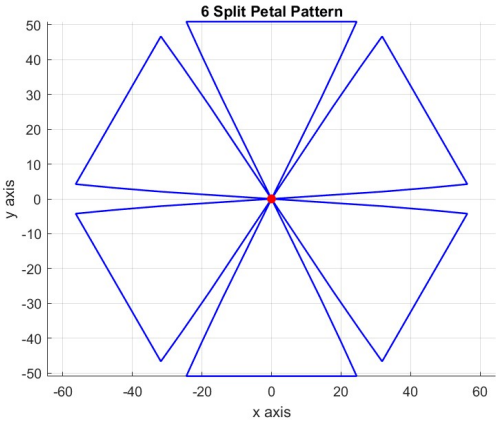



**Figure 23.** Final Unfolded 2D Pattern of the Parabolic Dish, Obtained by Rotating a Single Petal

Finally, to assemble the complete 2D pattern, each petal' s coordinate matrix $V$ is rotated by an angle $\theta = 360^\circ / n_{\text{Pieces}}$. Let

$$V' = R(\theta)\, V = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_M \\ y_1 & y_2 & \cdots & y_M \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & \cdots & x'_M \\ y'_1 & y'_2 & \cdots & y'_M \end{bmatrix}.$$

By repeating this rotation for all $n_{\text{Pieces}}$ petals, the final 2D unfolding is obtained, as illustrated in Figure 22 and Figure 23.

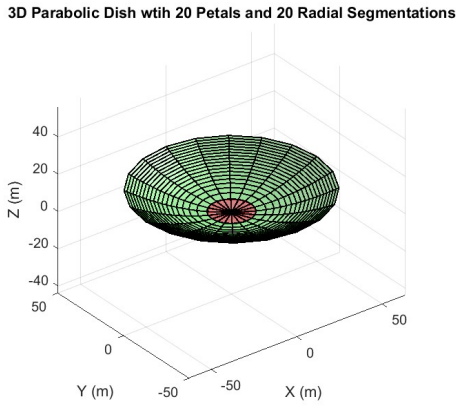



**Figure 24.** 3D Parabolic Dish with 20 Petals and 20 Radial Segments. Increasing the number of segments yields a finer approximation.

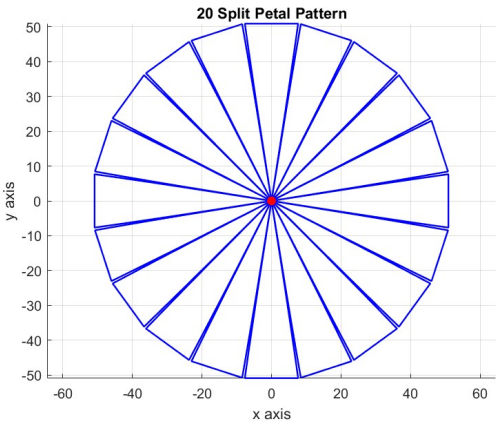



**Figure 25.** Unfolded 2D Pattern of the Parabolic Dish with 20 Petals and 20 Radial Segments.

By increasing the number of petals and the number of radial subdivisions, the 2D pattern can be made to approximate the 3D paraboloid arbitrarily closely. Figures 24 and 25 demonstrate that with 20 petals and 20 radial subdivisions, the fit is already highly accurate.

### 3.6 C-Shaped 2D Origami Design Algorithm for Parabolic Dish Antenna

The C-shaped parabolic antenna is inspired by the C-shaped ring configuration of the Thermorph structure described in An et al. (2018) [3]. The CAD model of this structure has been reproduced (Figure 44). If the bandwidth of the circular ring is increased until it becomes comparable to the radius, the ring effectively transforms into a fan-shaped sector. This sector then serves as the 2D unfolding pattern of the parabolic dish.

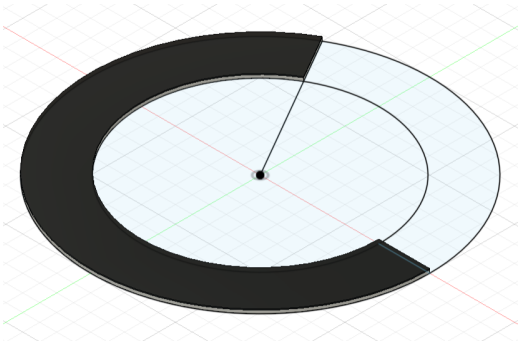


Figure 26. Replication of the C-Shaped Thermorph Structure

The derivation of the 2D unfolding algorithm is as follows:

1. Compute the circumference of the circle at each height layer and store these values in an array `levCirc`.

2. Use the previously computed arc length array `levArc`—which corresponds to the 2D "radius" of each layer—and relate it to the circumference via

$$\theta_i = \frac{\text{levCirc}_i}{\text{levArc}_i}, \qquad i = 1, 2, \ldots, (\text{number of height layers}).$$

   Here, $\theta_i$ represents the central angle of a circular sector whose arc length equals $\text{levCirc}_i$.

3. Split the angle $\theta_i$ equally on either side of the $x$-axis:

$$\theta_{2,i} = \frac{2\pi - \theta_i}{2}, \qquad \theta_{3,i} = 2\pi - \theta_{2,i}.$$

   Here, $\theta_{2,i}$ and $\theta_{3,i}$ define the lower and upper extents of the sector around the $x$-axis.

4. Convert these polar coordinates to Cartesian:

$$x_i = \text{levArc}_i \cos(\theta_{k,i}), \qquad y_i = \text{levArc}_i \sin(\theta_{k,i}), \quad k \in$$

By connecting these points sequentially, the final C-shaped 2D unfolding is obtained.

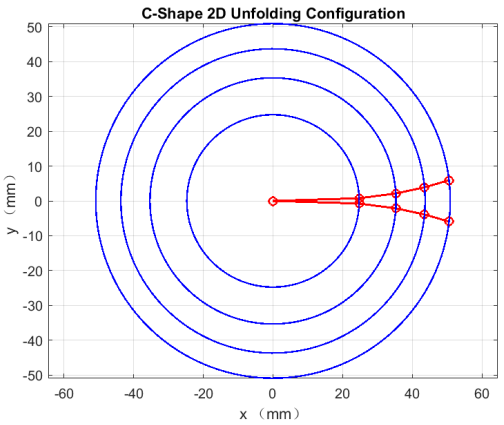



Figure 27. C-Shaped Unfolded 2D Pattern of the Parabolic Dish

### 3.7 Export of Nodal Coordinate Matrix to CAD

The final MATLAB script writes the assembled 2D coordinate matrix of all unfolded points to a `.csv` file. Using the Import CSV File feature in Fusion 360, the `.csv` is then converted into a point cloud within the CAD environment.

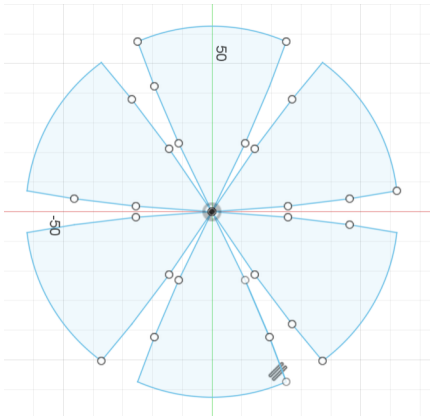



**Figure 28.** Two-Dimensional Point Arrays Imported from MATLAB into Fusion 360 for the Petal-Patterned Layout Design.

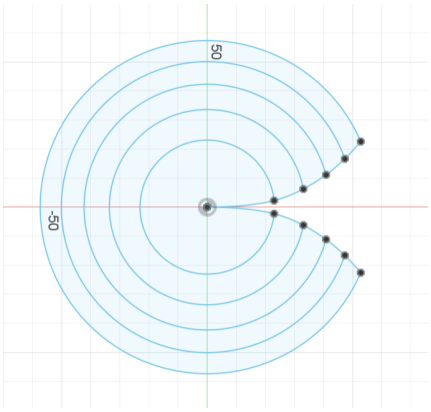



**Figure 29.** Two-Dimensional Point Arrays Imported from MATLAB into Fusion 360 for the C-Shaped Layout Design.