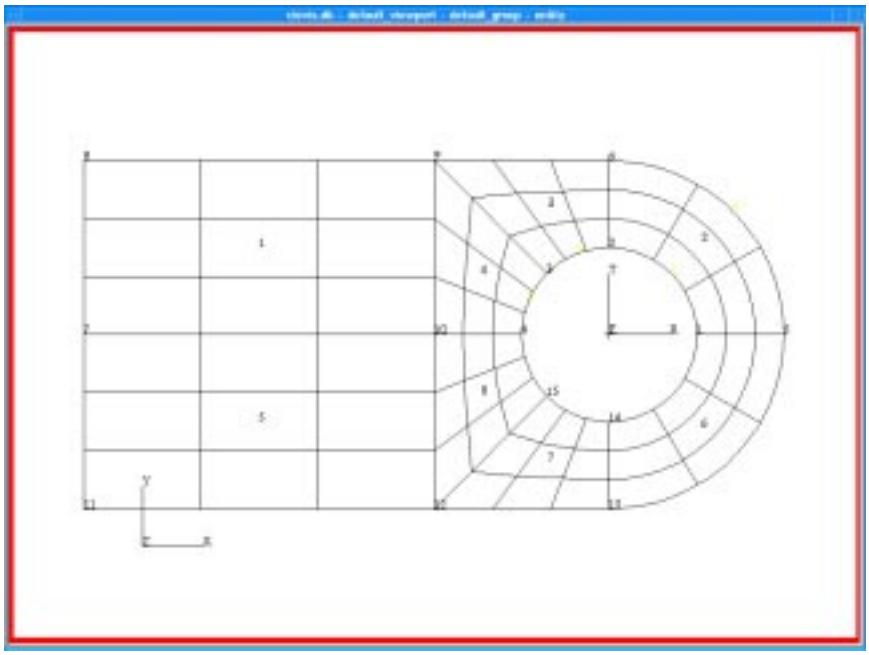
EXERCISE 4

Create Lug Geometry



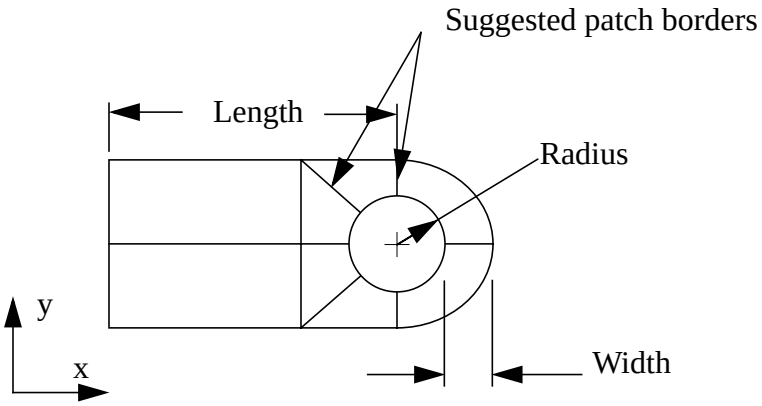
Objective:

- Write a function to create the geometry of the lug.

Create Lug Geometry

Exercise Description:

This exercise, `lug_create ( )`, creates a 2-dimensional model of a lug using parametric cubic patches. Use the global variables radius, width, and length to create the lug. Geometry is created and stored in the database as a result of this function.



Files:

All the files that used in this exercise are listed below. Each list includes the file, where it originated, and a summary of information of how it relates to the exercise.

File	Supplied/Created	Description
p3prolog.pcl	Created	Should contain the values for the variables that you are going to use in creating the lug.
lug_create.pcl	Created	This will be created from a session file that you are going to build during the exercise.

Exercise Procedure:

1. Create a file called **p3prolog.pcl**. In this file you will need to give the values of the radius, length and width of the lug. The file should look like the one shown below.

```
global real radius = 1., width = .5, length = 5.
global real thickess = .125, amplitude = 100.
```

2. Start PATRAN in the directory in which you just created the

p3prolog.pcl file.

3. Open a new database called lug.db.

File/New ...

New Database Name **lug.db**

OK

The viewport (PATRAN's graphics window) will appear along with a *New Model Preference* form. The *New Model Preference* sets all the code specific forms and options inside MSC/PATRAN.

In the *New Model Preference* form click OK.

Tolerance: ◆ **Default**

Analysis Code: **MSC/NASTRAN**

Analysis Type: **Structural**

OK

4. Start recording a new session file called **lug_create.ses**

File/Session/Record ...

lug_create.ses

Apply

5. Turn on the entity labels by clicking on the Show Labels icon.

Show Labels

6. Create a point with the variable 'radius' as the X coordinate

Note: The variable is inside back tics not single quotes..

◆ **Geometry**

Action: **Create**

Object: **Point**

Method: **XYZ**

❑ **Auto Execute**

Point Coordinates List	**['radius' 0 0]**

Apply

7. Create a curve by revolving the point

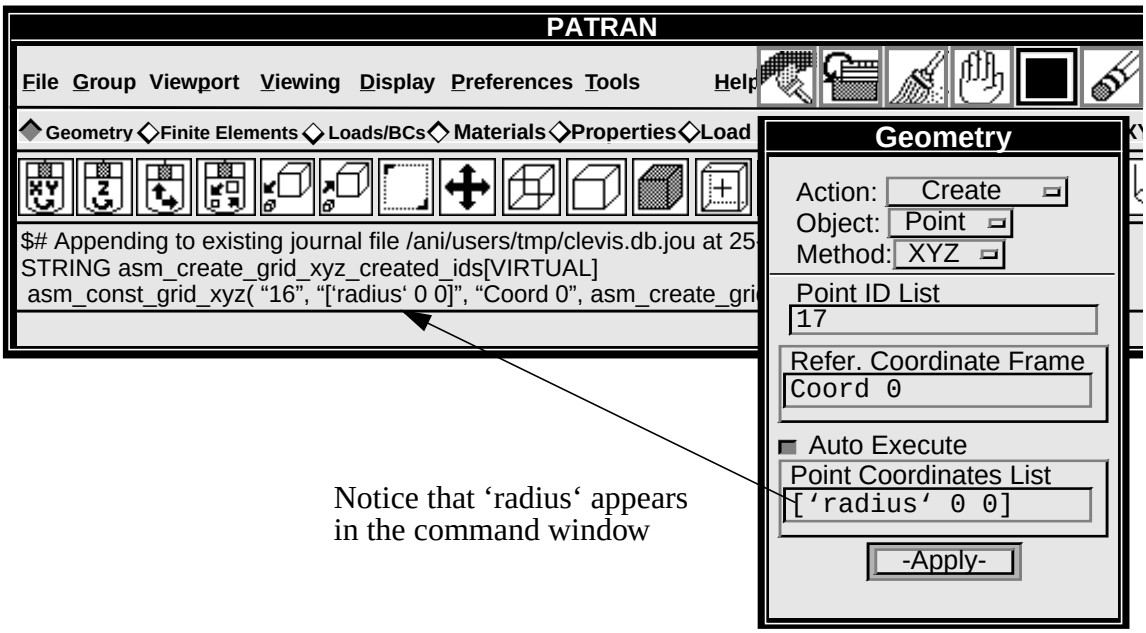
Action:	**Create**
Object:	**Curve**
Method:	**Revolve**
Total Angle	**90**

❑ **Auto Execute**

Point List	**Point 1**

Apply

For example, one could create a point as follows:



8. Now create another curve by revolving point number 2.

Action:	**Create**
Object:	**Curve**
Method:	**Revolve**

■ PATRAN 2 Convention

Total Angle	**90**
Curves per Point	**2**
Point List	**Point 2**

Apply

9. Create a cylindrical coordinate frame.

Action:	**Create**
Object:	**Coord**
Method:	**3Point**
Type:	**Cylindrical**

Apply

10. Transform curve number 1 in order to make the outer radius of the lug.

Action:	**Transform**
Object:	**Curve**
Method:	**Translate**

◆ **Curvilinear in Refer.CF**

Refer. Coordinate Frame	**Coord 1**
Translation Vector	**<‘width‘ 0 0>**

❑ **Auto Execute**

Curve List	**Curve 1**

Apply

11. Construct a surface for the lug using the variables that were defined in the p3prolog.pcl file.

Action:	**Create**
Object:	**Surface**
Method:	**XYZ**
Refer. Coordinate Frame	**Coord 0**

Vector Coordinates List	**<'length-radius-width' 'radius+width' 0>**
Origin Coordinates List	**['-length' 0 0]**

Apply

12. Now create the surface using curves from the inside arc and the curve that was just created by translating.

Action:	**Create**
Object:	**Surface**
Method:	**Curve**

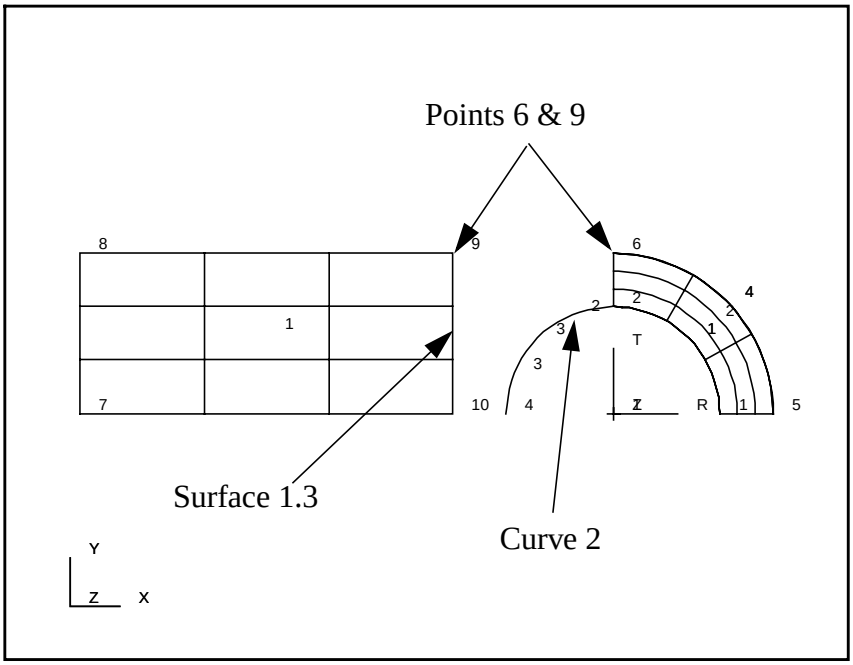
❑ **Auto Execute**

Starting Curve List	**Curve 1**
Ending Curve List	**Curve 4**

Apply

13. Create the next surface as described below. Reference the picture below for point and curve locations.

First select curve 2. Then select the two points icon and select points 6 and 9 to complete the surface.



Action:	**Create**
Object:	**Surface**
Method:	**Curve**
Starting Curve List	**Curve 2**
	Select the two points icon
	Then select the point icon
Ending Curve List	Select the two points shown above

Apply

14. Now create another surface using curve 3 and the edge of the rectangular surface.

Action:	**Create**
Object:	**Surface**

Method:	**Curve**
Starting Curve List	**Curve 3**

The select edge icon

Ending Curve List	**Surface 1.3**

Apply

15. Now mirror the top part of the lug about the X axis.

Action:	**Transform**
Object:	**Surface**
Method:	**Mirror**
Define Mirror Plane Normal	**Coord 0.2**

■ Reverse Surface

❑ **Auto Execute**

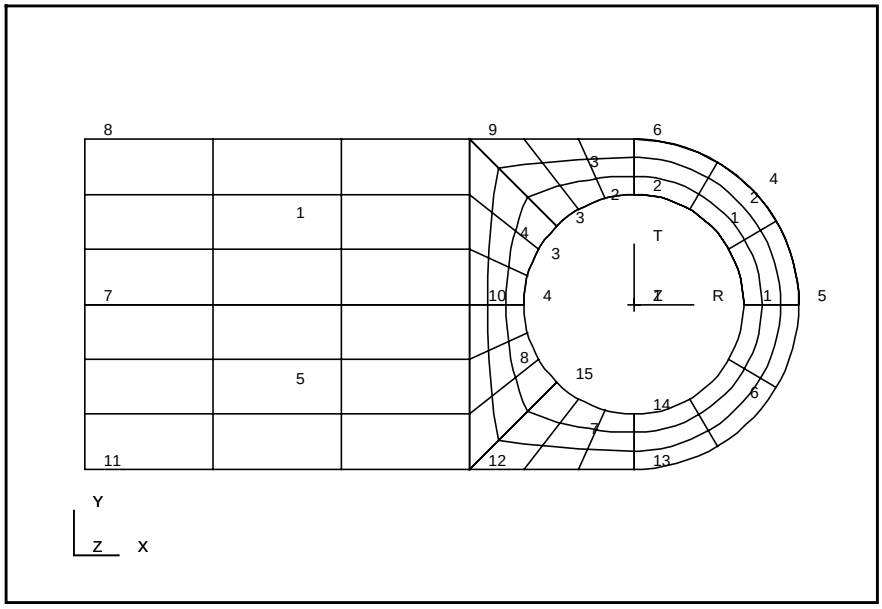
Surface List	Select All Surfaces

Apply

16. Change the number of display lines to 2 by clicking on the display lines icon in the main menu bar.

Display Lines

When complete, your lug model should look like the one shown below.



17. Stop recording the session file lug_create.ses.

File/Session/Record ...

lug_create.ses

Stop

Cancel

18. Use a text editor such as vi or jot to edit the lug_create.ses file. Create a PCL function from the body of the session file. Call the function **lug_create()**. At the end of the main body of the session file make sure to end the function.

19. Change the name of the session file to **lug_create.pcl**. In the PATRAN command window type:

!!input lug_create.pcl

20. Now delete all the geometry in the model.

Action:	**Delete**
Object:	**Any**
Geometric Entity List:	Select all geometry on screen

Apply

21. Refresh the graphics on the screen.

The repaint icon

22. At the PATRAN command window type in the following command:

lug_create()

The model that was just deleted should appear once again. The next steps in the exercise are to change the radius, length, and width to different sizes. You can change these variables by typing them directly into the PATRAN command line. For example:

```
global real radius = 3.
```

After you delete the geometry and then execute the function again the radius of the the lug should be the size that you just entered.

Before continuing to the next exercise change the global variables back to their original values.

Sample Solution:

```
FUNCTION lug_create( )

/*
 * Purpose:
 * Create a 2-dimensional model of a lug with
 * parametric cubic patches.
 *
 * INPUT:
 *       none
 *
 *
 *
 * OUTPUT:
 *       none
 *
 * Side effects:
 * A 2D lug is created from the specified dimensions
 */

 STRING asm_create_grid_xyz_created_ids[VIRTUAL]
 STRING curve_1[VIRTUAL]
 STRING curve_4[VIRTUAL]
 STRING asm_sweep_line_arc_created_ids[VIRTUAL]
 STRING asm_create_cord_3po_created_ids[VIRTUAL]
 STRING asm_create_patch_xy_created_ids[VIRTUAL]
 STRING asm_patch_2curve_created_ids[VIRTUAL]
 STRING asm_transform_patch_created_ids[VIRTUAL]

           /*
           * Create the constructions grid
           */

           asm_const_grid_xyz( “1”, “[‘radius‘ 0 0]”, “Coord 0”, @
           asm_create_grid_xyz_created_ids )

           /*
           * Create the line describing the inner radius of the lug
           */

           asm_sweep_line_arc( “1”, “{[0 0 0][0 0 1]}”, 90., 0., @
                               “Coord 0”, 1, “Point “ //@
                               “1 “, curve_1 )
           asm_sweep_line_arc( “2”, “{[0 0 0][0 0 1]}”, 90., 0., @
                               “Coord 0”, 2, “Point “ //@
                               “2 “, asm_sweep_line_arc_created_ids )
           asm_const_coord_3point( “1”, “Coord 0”, 2, “[0 0 0]”, @
                               “[0 0 1]”, “[1 0 0]”,@
                               asm_create_cord_3po_created_ids )

           /*
           * Construct the outer diameter of the lug
           */

           asm_transform_line_translate( “4”, “<‘width‘ 0 0>”, “Coord 1“, @
                               1, TRUE, FALSE, @
                               curve_1, curve_4 )

           /*
           * Construct the patches
           */
```

```
        asm_const_patch_xyz( "1", @
                             "<'length-radius-width' 'radius+width' 0>", @
                             "['-length' 0 0]", "Coord 0", @
                             asm_create_patch_xy_created_ids )

        asm_const_patch_2curve_v1( "2", curve_1, curve_4, 0, @
                             "", TRUE, @
                             asm_patch_2curve_created_ids )

        asm_const_patch_2curve_v1( "3", "Curve 2 ", @
                             "Construct 2PointCurve(" //@
                             "Evaluate Geometry(Point 9 ))"//@
                             "(Evaluate Geometry(Point 6 ))", 0, @
                             "", TRUE, @
                              asm_patch_2curve_created_ids )

        asm_const_patch_2curve_v1( "4", "Curve 3 ", "Surface 1.3 ", 0, @
                             "", TRUE, @
                             asm_patch_2curve_created_ids )

        asm_transform_patch_mirror( "5", "Coord 0.2 ", 0., TRUE, FALSE, @
                             "Surface 1:4 ", @
                             asm_transform_patch_created_ids )

END FUNCTION
```