5. MATLAB as drawing tool

MATLAB can be used as a drawing board for pictures and diagrams. Unlike other graphing software, we have an enormous library of mathematical functions at our fingertips. Here we demonstrate how MATLAB could be used to design and model objects.

MATLAB is very good in dealing with approximate data. We have encountered already the least squares approximation when we used **polyfit**. For graphical design problems, cubic splines achieve a similar smoothening effect as in polynomial fitting.

5.1 Spheres and Cylinders

MATLAB has built in commands to draw spheres and cylinders:

```
>> [x,y,z] = sphere(10);   % three 11x11 matrices   enter
>> surf(x,y,z);            % displays sphere         enter
```

Increasing the argument of the command **sphere** we can make the sphere more smooth.
Cylinders are displayed like :

```
>> [x,y,z] = cylinder(30);   % replace default by 30   enter
>> surf(x,y,z);              % to get finer picture    enter
```

Cylinders are a bit more interesting than spheres, because we can change the shape of the walls:

```
>> t = 0:0.1:2*pi;          % prepare sampling range   enter
>> r = sin(2*t);            % sample a sine            enter
>> [x,y,z] = cylinder(r);   % create cylinder          enter
>> surf(x,y,z);
```

Cones are special cylinders:

```
>> cylinder([1 0])   % displays a cone   enter
>> hold on           % keep the plot     enter
>> cylinder([0 1])
```

The cylinders produced by **cylinder** all have unit height. The argument of **cylinder** is a vector which contains the distance from the wall of the cylinder to the vertical axis at equally spaced points. This should explain why we get cones with [1 0] (distance 1 to axis at base and distance 0 at top) and why [0 1] is oriented in the opposite direction.

5.2 Cubic Spline Data Interpolation

We have seen how to fit data with a quadric or cubic curve. With **polyfit** we constructed one polynomial that best fitted the data minimizing the error, in a so-called *least squares approximation.*

Suppose we have a profile like this :

```
>> x = [1.0  1.8  4.0  4.8  6.8  7.6  8.8  9.4];   % x-coordinates of points   enter
>> y = [1.5  2.0  2.1  2.5  2.5  2.2  2.0  1.5];   % y-coordinates of points   enter
>> plot(x,y)                                       % piecewise linear plot     enter
>> axis([0 10 0 4])                                % for better viewing        enter
```

With some imagination, we recognize the profile of a car. We wish to make the profile more smooth.

The plot we made above is *piecewise linear*, MATLAB connected the points by straight lines. But instead of lines, we can connect the lines by polynomials passing through the points. Unlike the curve fitting we did in previous lecture, we will not use one single polynomial to pass through all the points, but several low degree polynomials. We partition the points and in each set of the partition a polynomial of degree three (i.e.: a cubic) is used to interpolate the data.

To make a more smooth design, we can ask MATLAB to construct a spline :

```
>> xx = 1 : 0.2 : 9.4;        % choose range to sample   enter
>> yy = spline(x, y, xx);     % construct a spline       enter
```

The output of **spline** can be seen as a collection of cubic polynomials that go through the given points. When we plot, we see that the profile is curved :

```
>> plot(x, y,'o', xx, yy);    enter
>> axis([0 10 0 4])           enter
```

The points marked with $'o'$ are the *control points*. The designer can shift these points to alter the shape of the curve.

Now we draw the wheels, using the same sample range for t as above.

```
>> hold on                                                  % keep profile              enter
>> plot(2.5 + 0.5 * cos(t), 1.5 + 0.25 * sin(t));           % contour of front wheel     enter
>> for k = 0.1 : 0.05 : 0.8                                 % fill up inside of wheel    enter
       plot(2.5 + 0.5 * k * cos(t), 1.5 + 0.25 * k * sin(t));                            enter
   end;                                                                                  enter
>> plot(8.0 + 0.5 * cos(t), 1.5 + 0.25 * sin(t));           % contour of other wheel     enter
>> for k  = 0.1 : 0.05 : 0.8                                % fill up inside of wheel    enter
       plot(8.0 + 0.5 * k * cos(t), 1.5 + 0.25 * k * sin(t));                            enter
   end;                                                                                  enter
```

Here we went through the drawing step by step, typing in the commands interactively. In practice, we will use a script for drawing. Seldomly, designs are "right" at the first time and we need to adjust frequently.

5.3 Three Dimensional Modeling

We can draw polyhedra with MATLAB. The faces of the polyhedra are called *patches*. For example

```
>> x1 = [0  1  1]; y1 = [0  0  1]; z1 = [0  0  0];   % define vertices          enter
>> patch(x1, y1, z1,'r')                              % show patch in red        enter
>> x2 = [0  1  0]; y2 = [0  1  1]; z2 = [0  0  1];   % define vertices          enter
>> patch(x2, y2, z2,'g')                              % show patch in green      enter
>> view(3), grid                                      % 3-D view with grid       enter
```

More complicated polyhedra can be made by specifying the relations between vertices and faces. The coordinates of the vertices are defined in the rows of a matrix :

```
>> v = [ 0  0  0     % vertex 1   enter
         1  1  0     % vertex 2   enter
         1 -1  0     % vertex 3   enter
         1  0 -1     % vertex 4   enter
         1  0  1     % vertex 5   enter
         1  0  0 ];  % vertex 6   enter
```

Faces are formed joining vertices, indexed by their labels.

```
>> f = [ 1  2  6    % face spanned by vertices 1, 2, and 6   enter
        1  6  3    % face spanned by vertices 1, 6, and 3   enter
        1  4  6    % face spanned by vertices 1, 4, and 6   enter
        1  6  5 ]; % face spanned by vertices 1, 6, and 5   enter
```

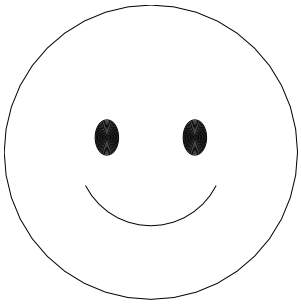
Then the object is rendered by

```
>> patch('vertices',v,'faces',f,'facecolor','g')   % draw faces in green      enter
>> view(3), grid                                   % default view with grid   enter
```

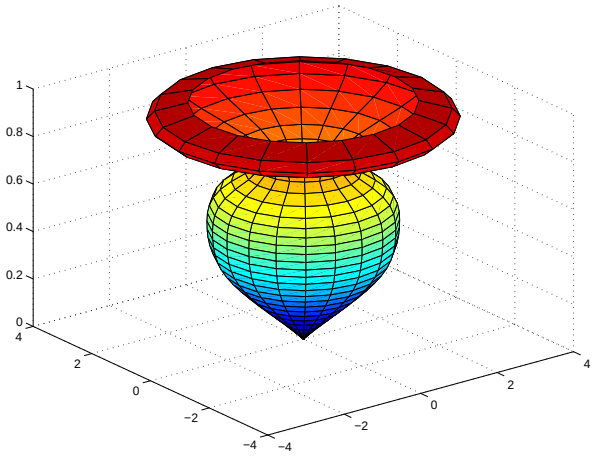
To explore this image, use **rotate3d** or click on the rightmost button of the toolbar.

5.4 Assignments

1. At the end of section one, we displayed two cylinders with **cylinder([1 0])** and **cylinder([0 1])**. What do you need to do so that the two cylinders only meet at their top?
2. Finish the drawing of the profile of a car by putting in windows, door, etc...
3. Use **patch** to make a plot of truncated pyramid. The base of the pyramid is a square, the lengths of the sides of the top of the pyramid are half as long as the sides of the base.
4. Make a plot of a happy face :



5. The curve defined by a spline using data `x = [1.0 2.0 3.0 4.0 5.0 5.5 6.0 7.0]` and `y = [0.0 1.0 1.8 2.0 1.5 1.0 2.0 2.4]` can be used in the **cylinder** command to make a pot shaped like in the following figure:



Give all MATLAB commands to make this design.