5: Contour Plots

In this section we will study contour plots. These are related to curves defined implicitly as the solutions of an equation $f(x,y)=0$, so we will take a look at that topic as well.

5.1 Basic Contour Plots

Matlab has several commands for generating contour plots. We will omit the ez-commands because one can't easily customize them in ways that are particularly valuable for this type of plot.

The standard contour plot commands use the same general framework as `surf`. Namely, we generate a pair of matrices holding the x and y coordinates of the grid points, then generate the z values from these using a vectorized expression. In these notes we will analyze the contour plot of the surface $z=x^3+y^3-3xy$. We describe the process step-by-step, but the commands should be written to an M-file for convenient modification.

Example 5.1: Generate a contour plot of the function $z=x^3+y^3-3xy$ over the square $-2\le x\le 2$ and $-2\le y\le 2$.

Solution:

a) Select a plotting region:

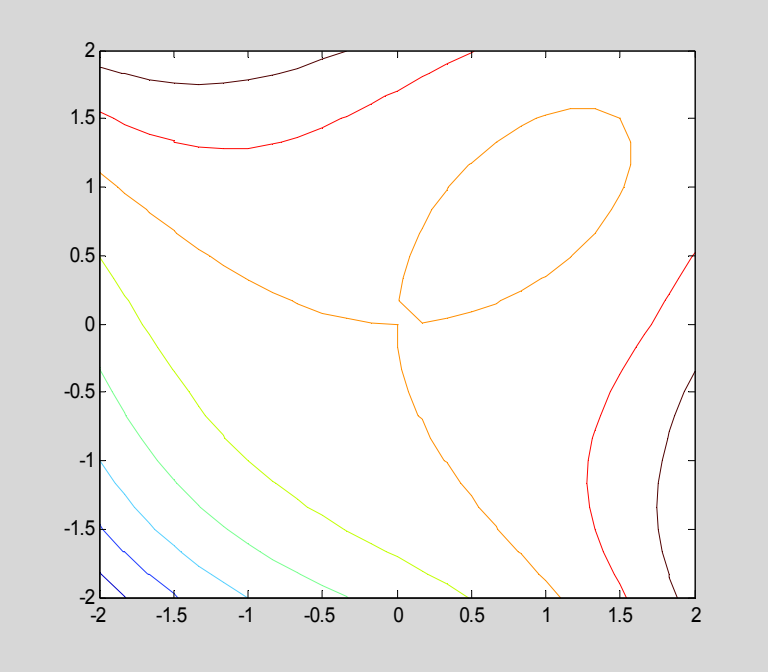
```
x=linspace(-2,2,25); y=linspace(-2,2,25);[x y]=meshgrid(x,y);
```

Now generate the z values using a vectorized expression for z.

```
z=x.^3+y.^3-3*x.*y;
```

We generate a 2D contour plot using the command `contour`.

```
contour(x,y,z)
```



■

Matlab generates a plot using a default number of level curves (contours with constant z value.) They are not labeled with the z value associated with each curve. The colors enable one to distinguish the relative heights on each curve. Reds and oranges correspond to larger z values than blues and purples. (To remember, think of the hot, red sun (high) and the cool, blue oceans (low) – OK, it helps me.)

5.2 Labeled Contour Plots

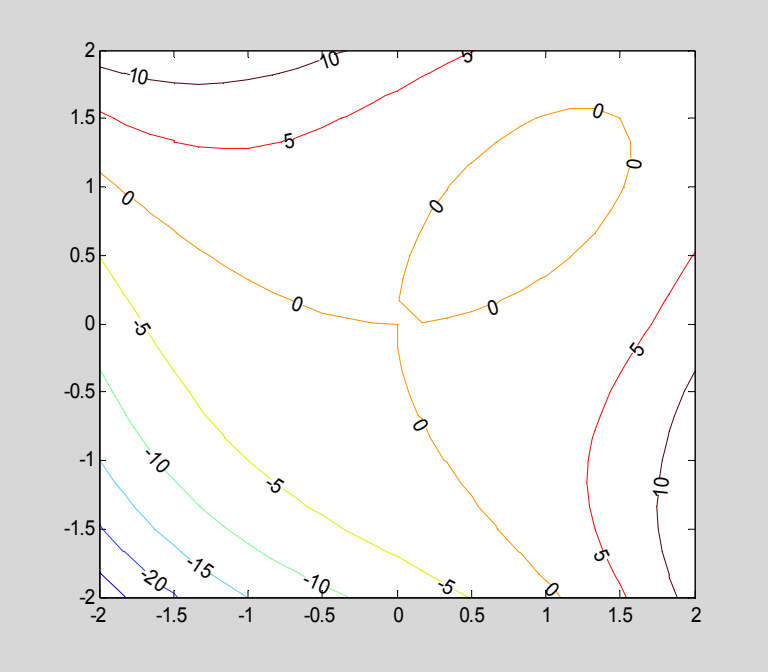
The first improvement over the basic contour plot would be to get Matlab to label the curves with the associated z value. We can do this by passing to the function `clabel` the so-called contour matrix (a matrix that holds all the data for drawing the curves) and the "handle" to the contour plot. You will recall from Chapter 2 that the handle is a name that we can ask Matlab to assign to identify a particular graphic object. We can access these bits of important information about the plot by assigning variables to the output of the `contour` command. Without going into excruciating details on all of this, let's see how it is done.

Example 5.2: Generate a labeled contour plot of the function $z = x^3 + y^3 - 3xy$ over the square $-2 \le x \le 2$ and $-2 \le y \le 2$.

Solution:

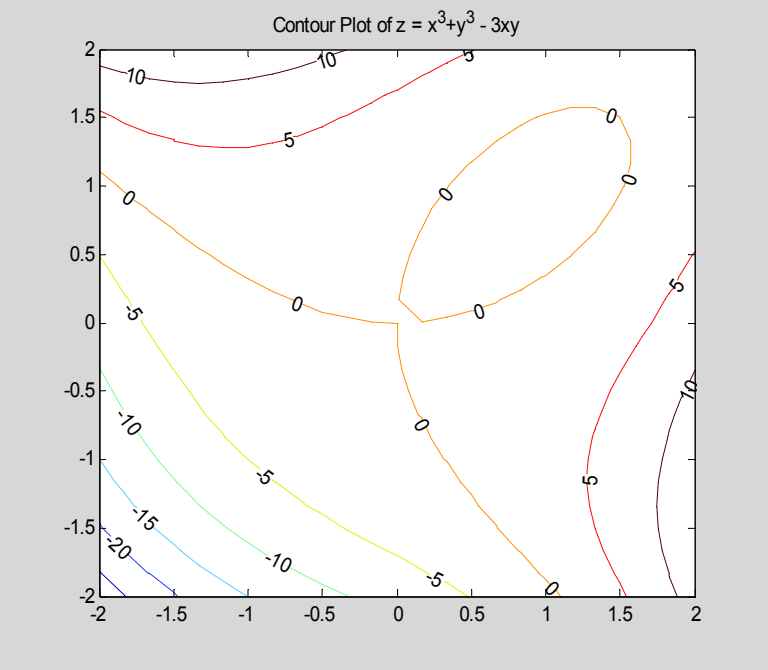
```
x=linspace(-2,2,25); y=linspace(-2,2,25);[x y]=meshgrid(x,y);
z=x.^3+y.^3-3*x.*y;
[C h]=contour(x,y,z);  %generates graph and assigns appropriate
variables to output for labeling
```

```
clabel(C,h);
```



This is much better. We can add a grid, if we wish, as well as axis labels and title in the usual way. Just to remind you, here is a nice title:

```
title('Contour Plot of z = x^3+y^3 - 3xy')
```



■

It is very instructive to explore this plot with the Data Cursor, which can be activated using an icon at the top of the figure window. (This feature may not work on some systems because of issues related to graphics card drivers.) As you move the cursor through the figure, the program displays both the coordinates of the point under the cursor and the *z* value (level value) of the point. Wouldn't it be great if you could click the mouse button and have the program generate the specific level curve that passes through a point, (in other words, connect all the *x*,*y* values at that same level)? Well, sorry, but we can't do this. However, as a consolation prize we can write our M-file to produce specific level curves.

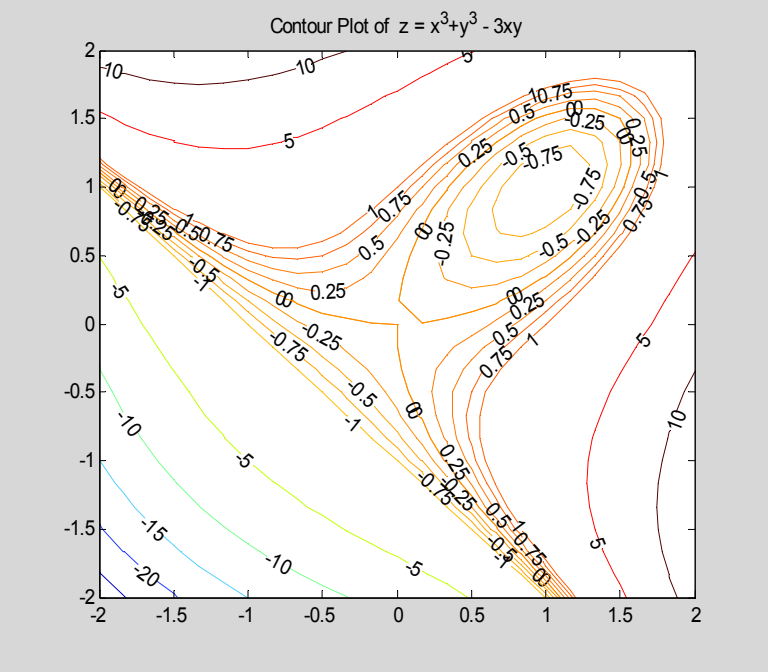
Programatically, we can get Matlab to add specific contour lines to the picture. We do this by creating a list of *z* values for which we want to see the level curves and pass this list to the `contour` command. We also pass the list to `clabel` so that the curves are labeled as well. In this case we would like to see level curves corresponding to values of *z* between -1 and +1.

Example 5.3: Add specific level curves to the contour plot generated in Example 5.2.

Solution:

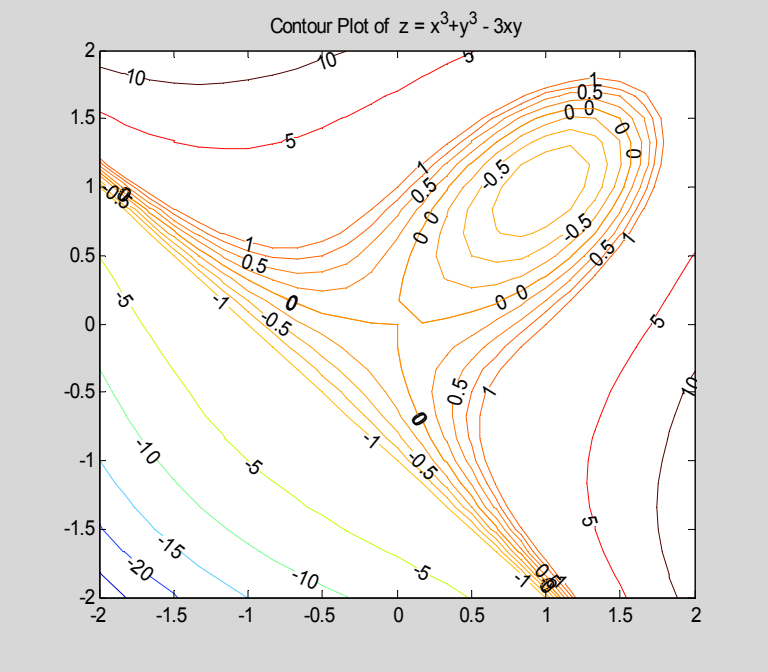
We assume the previous plot has been generated and is currently displayed. We then execute the following additional code:

```
vals=-1:.25:1; % level values to plot.  Recall the : operator described
% in the exercises of Chapter 3.
hold on
[C h]=contour(x,y,z,vals);
clabel(C,h,vals);  % add labels to the new curves
hold off
```



We seem to have generated too many labels. The picture appears cluttered. In this case we can either pass a shorter list to the `clabel` command, or we can use a neat interactive option to select the labels we care to see. The latter is done via the command `clabel(C,h,'manual')`. The help page for `clabel` contains a pretty good description of how this option works. As with all things interactive, it is most easily discussed by actually using it, which you will do in class. (N.B. This feature also has graphics driver issues, so may not work on all systems.) Here, we'll redo the plot with a truncated list of levels to be labeled.

```
vals=-1:.25:1;
hold on
[C h]=contour(x,y,z,vals);
clabel(C,h,-1:.5:1);  % fewer labels in this list
hold off
```



This is a bit less muddy. ■

Look at the picture. Do you see a point where the surface $z = x^3 + y^3 - 3xy$ might have a a maximum or minimum? What about a saddle point, where the function increases in some directions and decreases in others? By examining z values with the Data Cursor see how well you can approximate the location of these points. Later in the course, you will use methods of calculus to determine these points analytically. As we will see, such methods are often difficult to carry out, (though not in this case) so the simple technique used here in visually identifying the possible location of such points is very helpful.

There is one other strange phenomenon exhibited above. One of the level curves appears to be a straight line (which one?) You can find the equation of this line by examining the graph. Can you show algebraically that all points on this line produce the same value of z and therefore define a level curve? Try it.

5.3 Contour and Surface Plots

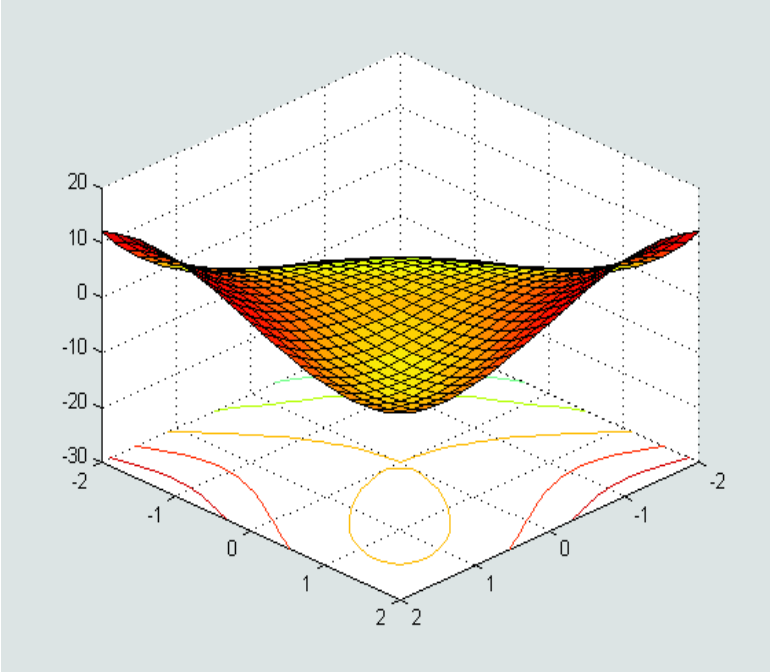
The graph of the surface can be visualized along with the contour curves using the command `surfc`. Here is the original contour set shown lying beneath the graph of the surface.

Example 5.4: Draw a combined contour and surface plot for the function $z = x^3 + y^3 - 3xy$.

Solution:

Assuming the data values x, y, and z have been obtained as above we use

```
surfc(x,y,z); view([1 1 1])
```



As usual, rotating the figure gives a clearer sense of the relationship between the level (contour) curves and the surface. In the exercises we will explore how the contour curves can be drawn on the surface. ■

5.4 Implicit Plots

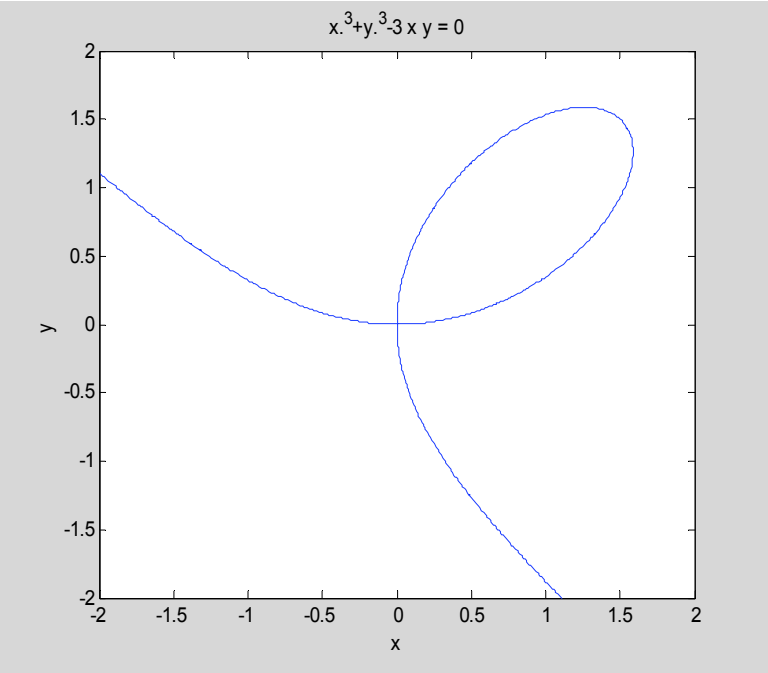
The level curves shown in a countour plot of $z = f(x, y)$ are plane curves given by setting z equal to a constant c. In other words, they describe the location of points where $f(x, y) = c$. In the example considered above these are the curves $x^3 + y^3 - 3xy = c$. For most such curves, we cannot solve for x or y as functions of the other variable and we say the curves are defined by implicit equations (remember implicit differentiation). The contour plot can be viewed as a collection of such implicit curves for different values of c. Sometimes, however, we would like a picture of simply one or two such curves and the mechanism of generating a contour plot is not the right way to achieve this. In fact, sometimes Matlab errs in the shape of curves it forms in the contour plots because of the limited information with which it is working.

Suppose for example that we want to plot the single curve $x^3 + y^3 - 3xy = 0$, which is one of the curves in the contour plot generated earlier. We can actually get Matlab to plot this using our old friend `ezplot`. When we used `ezplot` before, we passed symbolic expressions to it. However, since we are working here with numeric variables, we use an alternate method – passing the expression as a string.

Example 5.5: Draw a plot of the implicitly defined curve $x^3 + y^3 - 3xy = 0$.

Solution:

```
ezplot('x.^3+y.^3-3*x.*y', [-2 2])
```



The string should be the vectorized form of the expression $f(x, y)$. Matlab assumes you want to plot $f(x, y) = 0$. If you want $f(x, y) = c$, just move the constant to the left side and include it in the string. The domain is indicated by the entry [-2 2]. Matlab assumes that both x and y will vary over the same bounds. In other words, you are limited to sketching over a square region. You should compare the picture above with the zero level curve in the contour plot. You will see that Matlab had trouble in the latter plot in analyzing the proper shape of the curve near the origin. ■

5.5 Summary

We introduced three new commands, but each has a number of options with which you need to be acquainted to make effective use of them. Of course, everything we've said, and more, can be found in Help.

`contour`: - draw a 2D contour plot of a function of two variables

`clabel`: - generate labels for a 2D contour plot – various options for specifying the labeled curves.

`surfc`: - generate a surface plot and a contour plot lying below it.

`ezplot`: - generate a plot of an implicit curve $f(x, y) = c$

5.6 Exercises

Exercise 5.1. Let $f(x,y) = 2x^2 + xy + y^4$. Write an M-file that uses `subplot` to generate a 1 x 2 array of plots with the following specifications.

a) The left plot should use `surfc` to generate a plot of the graph $z = f(x,y)$ and contours over the region $-1 \le x \le 1$ and $-1 \le y \le 1$. Use a 25 x 25 grid. Display **grid** lines, **labels** on all axes, and a **title** that includes the formula for the function. (Use a multi-line title if necessary.)

b) The right plot should

i) display the default contour plot of $f(x,y)$ over the square [-1,1]x[-1,1] using a 25 x 25 grid, with labels on all contour lines.

ii) add five additional contours at levels between 0 and 0.5.

iii) label the additional contours using `clabel`.

iv) include **labels** on the axes and a **title**

c) Does the function have any maximum or minimum values in the domain $[-1,1]\times[-1,1]$? If so use either the Data Cursor or the coordinate grid to estimate the x and y values at the maximum/minimum points. Specify these estimates in a comment line in your M-file.

Publish your M-file (include your name in a comment). The total should be at most two pages.

Exercise 5.2 Suppose $f(x,y) = \sin(3y - x^2 + 1) + \cos(2y^2 - 2x)$.

a) Write an M-file that will produce a labeled contour plot for $f(x,y)$ over the region $S = \{(x,y) \mid -2 \le x \le 2, -1 \le y \le 1\}$.

b) Based on the contour plot you found in a) estimate the coordinates of two saddle points of the function in the region S defined in a). Mark the points using the Data Cursor.

Exercise 5.3 Suppose $f(x,y) = \sin(3x + y) - 2\cos(x - y)$.

a) Write an M-file that will produce 15 **labeled** contour curves for $f(x,y)$ over the square $S = \{(x,y) \mid 0 \le x \le 2, 0 \le y \le 2\}$.

b) Based on the contour plot you found in a) determine whether the function has any critical points in the square S defined in a). If there are any such points, provide **estimates** from the graph for their x and y coordinates and provide a **justification** from the graph as to whether these are **relative maxima**, **minima** or **saddle points**. Indicate your reasons as comments on the M-file and publish the contour plot and the M-file.

Exercise 5.4 Write an M-file that uses `ezplot` to display the graph of $2x^2 + xy + y^4 = 1$ and the circle $x^2 + y^2 = 1$, on the same figure. Your graph should show

a) A graphing window where x and y vary from -1.2 to $+1.2$.

b) Display of grid lines

c) The circle displayed in a dotted line style.

d) A legend displaying the equation of each curve and a title with your name

e) The curves meet at the points $(0, \pm 1)$ and at two other points. Using the Data Cursor, (if available) display labels at the approximate locations of these two other points. (To display multiple data points, right click the data marker box and choose the "Create New Datatip" option.) Otherwise, estimate the coordinates of these additional intersection points using the grid lines. Write your estimate as a comment in your M-file.

Publish your M-file. Be sure to include your name.

Exercise 5.5 Run the M-file below (you can download the file as `contourplot.m` from the course site). What does the file do? Explain what is done by each of the shaded lines.

```
% 3D Contour plotting
a=-2;b=2;
c=-2;d=2;
x=linspace(a,b,25);
y=linspace(c,d,25);
[x y] =meshgrid(x,y);
z=-x.*y.*exp(-x.^2-y.^2);
clf;
hold on
[C h]=contour3(x,y,z,'k');
set(h,'LineWidth',1.5);
surf(x,y,z);
view([1 1 1]);
grid on
shading interp
xlabel('x');ylabel('y');zlabel('z');
hold off
```

6: Symbolic and Numerical Calculations

We have touched on symbolic objects earlier in our discussion of the ez-graphing techniques, which serve as a parallel plotting world for symbolic users.

Now we want to consider using Matlab to do symbolic calculations. We will also consider how we do numerical calculations with symbolic expressions. For example, how do we compute a derivative symbolically and then find numerical values of the derivative? We begin first with a discussion of Matlab functions.

6.1 Matlab Functions

From the point of view of computing, a function is a structure *f* that accepts one or more inputs and produces one or more outputs. Matlab has two mechanisms for defining functions – the M-file function and the anonymous function. An M-file function, as the name suggests, is a function that is created as a special type of M-file. The built-in Matlab functions are M-functions. Such functions are saved by the system and can be accessed at a later date. Typically, one uses an M-function when constructing a function requires multiple lines of code. For example, in addition to carrying out its principal calculation, one might want the function to check that the user has entered a legal input, and if not, issue an error message and not proceed. In this course, we will not be interested in creating such carefully crafted functions so we will have little need to write M-file functions. Instead, we will focus on the more useful **anonymous function** construct.

6.1.1 Anonymous Functions

Although, as the name suggests, an anonymous function may indeed be used without giving it a name, more commonly we will attach names to these functions. Here is a very simple illustration of the construction.

Example 6.1: Write a Matlab anonymous function to define $f(x) = x^2$. The function should be vectorized so it accepts vector inputs and computes the result for each component of the vector.

Solution:

```
f=@(x)x.^2
f =
    @(x)x.^2
```

The symbol @ initiates the definition. This is followed by a list of the input variables, in parentheses, separated by commas if there is more than one variable. Finally, we state the formula defining the function. If we wish to have the function act component-wise on arrays, the dot operator should be included in this formula. The function has been named

f. Note that it is not necessary to declare x as symbolic in order to use it as a placeholder for the variable within an anonymous function ■

Such a function can be used in much the same way you use functions in ordinary mathematics, with the added feature of being vectorized. Thus, if

```
x=[1 2 3]
x =
     1     2     3
```

then entering

```
f(x)
```

produces the square of each input value.

```
ans =
     1     4     9
```

However, if x has not been declared to be symbolic then a request such as

```
f(x)
??? Undefined function or variable 'x'.
```

produces an error message.

We can define functions of more than one variable in a similar fashion.

Example 6.2: Define an anonymous function $g(x,y)=x^2+y^2$. Vectorize the function to accept input arrays.

Solution:

```
g=@(x,y)x.^2+y.^2
g =
    @(x,y)x.^2+y.^2
```

Then `g(1,2)` is

```
ans =
     5
```

and if `x=[1 2 -1]` and `y=[3 2 1]` then

```
x =
     1     2    -1
y =
     3     2     1
```

and `g(x,y)`

```
ans =
    10     8     2
```

consists of the values $g(1,3)$, $g(2,2)$ and $g(-1,1)$. ■

6.2 Symbolic Functions

So far, when using anonmymous functions we did not require that the function variables represent symbolic quantities. When we add that option we obtain symbolic functions, which closely resemble ordinary mathematical functions. Let us first declare a pair of symbolic variables.

```
syms x y  .
```

We can now create symbolic functions of these variables.

Example 6.3: Define the function $h(x,y) = x^2 + y^2$ as a symbolic, vectorized function of x and y.

Solution:

```
h=@(x,y)x.^2+y.^2
h =
    @(x,y)x.^2+y.^2
```

Then not only can we compute `h(2,3)`

```
ans =
    13
```

but we can also ask for symbolic quantities, such as `h(x,x^2)`

```
ans =
x^2+x^4
```

An important point to observe, however, is that we must never assign any numeric value to the symbolic variables themselves. Doing so will destroy their symbolic character. Thus, if we want to compute the value of h at each pair (x,y) from the arrays [1, 2, -1] and [3, 2, 1] we can do so by assigning suitable upper-case letters as names for the input arrays. Remember, Matlab is case-sensitive, so that x and X represent different quantities.

`X=[1,2,-1]` and `Y=[3,2,1]`

```
X =
     1     2    -1
Y =
     3     2     1
```

Then `h(X,Y)`

yields the desired result.

```
ans =
    10     8     2
```

■

6.2.1 Calculus: - Differentiation

One purpose in using the Symbolic Toolbox is to have the system perform complicated manipulations using its built-in symbolic processing. A particularly useful feature is symbolic differentiation. Later we will also take up integration. The `diff` command, which we have briefly encountered in Chapter 1, computes derivatives of a symbolic expression.

Example 6.4: Find the derivative of $y = x\sin(2x)$.

Solution: Assuming that x has been declared symbolic we proceed as follows:

```
y=x.*sin(2*x); diff(y)

ans =

sin(2*x)+2*x*cos(2*x)
```

Notice that although we started with a vectorized expression, the formula that is returned for the derivative is not vectorized. We can get Matlab to supply the appropriate dot operators using the `vectorize` command.

```
vectorize(ans)

ans =

sin(2.*x)+2.*x.*cos(2.*x)
```

As in this case, the `vectorize` command usually supplies more dot operators than are actually needed. ■

We can compute higher order derivatives by including an extra parameter in the `diff` function.

Example 6.5: Compute the 3rd derivative of $y = x\sin(2x)$.

Solution: Add the order of differentiation as a second argument in the `diff` function.

`diff(y,3)` gives the 3rd derivative.

```
ans =

-12*sin(2*x)-8*x*cos(2*x)
```

■

Notice that the input to `diff` is an expression, and the command returns an expression as output. Sometimes it is useful to have a derivative function. This is particulary so when we want to make numerical evaluations of the derivative. We can get Matlab to convert the expression for the first derivative generated above, $\sin(2x) + 2x\cos(2x)$, into a function, as follows:

Example 6.6: Construct a derivative function from the symbolic derivative formula.

Solution:

Suppose we start with the symbolic function $f(x) = x\sin(2x)$. We want to construct the function $f'(x)$. Moreover, both functions should be vectorized. Here's how to do it.

First let's define $f(x)$.

```
f=@(x)x.*sin(2*x)
```

```
f =
    @(x)x.*sin(2*x)x
```

Then we construct the derivative expression via

```
dy=diff(f(x))
dy =
sin(2*x)+2*x*cos(2*x)
```

Finally, we can create the appropriate derivative function using the `subs` command.

```
df=@(x)subs(dy)
df =
    @(x)subs(dy)
```

This is a perfectly general method for converting an expression defined through some other procedure, in this case as a result of applying the `diff` function, into an anonymous function. [1]

When we evaluate the function *df*, the `subs` command passes any input values to the expression *dy*. If we tried to do this construction without the `subs` command, it would not produce the desired result. (Try it!) In our case, everything works fine. For example,

```
df(x)
ans =
sin(2*x)+2*x*cos(2*x)
```

returns the expression for the derivative, *dy*. Although it does not appear from the latter formula that the derivative function *df* is vectorized, in fact it is.

```
df([pi,pi/2])
ans =
    6.2832   -3.1416
```

returns the numerical value of the derivative at π and $\pi/2$. ■

Everything we have done for functions of a single variable works equally well for functions of several variables. The only extra bookkeeping is that the `diff` command requires that we specify the variable of differentiation.

Example 6.7: Find $\frac{\partial f}{\partial x}$, $\frac{\partial f}{\partial y}$, and $\frac{\partial^4 f}{\partial y^2 \partial x^2}$ for the function $f(x,y) = \sin(x^2 + y^2)$.

Solution: Define a symbolic function *f* of two variables.

```
f=@(x,y)sin(x.^2+y.^2);
```

Then we have

```
zx=diff(f(x,y),x)
```

[1] I am grateful to Prof. Peter Brinkmann for this construction.

```
zx =
2*cos(x^2+y^2)*x
```

for z_x and

```
zy=diff(f(x,y),y)
zy =
2*cos(x^2+y^2)*y
```

for z_y.

If we wish to generate (vectorized) partial derivative functions we can do so through

```
fx=@(x,y)subs(zx);
```

and `fy=@(x,y)subs(zy);` .

Then, for example

using the arrays `X=[1, 2 ,-1]` and `Y=[3, 2, 1]` , we get

```
X =
     1     2    -1
Y =
     3     2     1
```

and we have `fx(X,Y)`

```
ans =
   -1.6781   -0.5820    0.8323
```

The required mixed partial cannot be computed in a single application of the `diff` function. The latter can only compute derivatives with respect to one variable at a time. To compute mixed partials, we use the theorem that says the order of differentiation is not important. Thus, to find $\frac{\partial^4 f}{\partial y^2 \partial x^2}$, we first find $z_{xx} = \frac{\partial^2 f}{\partial x^2}$ and then take the second partial of this result with respect to y.

```
zx2=diff(f(x,y),x,2);
```

This is the partial z_{xx}.

Then take the second partial of this with respect to y.

```
zx2y2=diff(zx2,y,2)
zx2y2 =
16*sin(x^2+y^2)*y^2*x^2-8*cos(x^2+y^2)*x^2-8*cos(x^2+y^2)*y^2-
4*sin(x^2+y^2)
```

■

The `pretty` command can be used to write this last expression in a somewhat more legible form, though if mathematically typeset output is desired one must use the `latex` command to convert the expression to the mathematical markup language latex. The latter result can then be printed through a program that generates mathematical expressions from such input. Unfortunately, it is not (yet) possible to include computer generated latex output in a published version of an M-file, (you can add explicit latex code), but undoubtedly this will be an added feature in a future release.

```
pretty(zx2y2)
```

```
          2    2   2  2          2    2   2          2    2   2
  16 sin(x  + y ) y  x  - 8 cos(x  + y ) x  - 8 cos(x  + y ) y

                  2    2
        - 4 sin(x  + y )
```

6.3 Tangent Planes

The following M-file uses some of the ideas discussed above to draw the graph of a function and its tangent plane at a specified point. We highlight the lines that use material discussed in this section. The M-file is available as *tanplane.m* on the course website. One may have to rotate the figure to get a view that shows the tangency to the best effect.

```
% Surface and tangent plane plot
% User Input
a=0; b=1.5; % x interval for plotting
c=0; d=1.5;  % y interval for plotting
% grid size parameters
m=15;
n=15;
% x, y coordinates of point of tangency
X0=1;
Y0=1;
%function to plot
syms x y
f=@(x,y)sin(x.^2+y.^2);  % anonymous function
% End user input
% Begin Matlab computations
% Construct 1st derivative functions
zx=diff(f(x,y),x);
zy=diff(f(x,y),y);
fx=@(x,y)subs(zx);
fy=@(x,y)subs(zy);
% Plot Surface
clf
hold on
X=linspace(a,b,m); % use X, Y, Z for numeric values
Y=linspace(c,d,n);
[X Y] = meshgrid(X,Y);
Z=f(X,Y);
surf(X,Y,Z); view([1 1 1]);
% Construct symbolic function for z value on tangent plane
Z0=f(X0,Y0);
A=fx(X0,Y0);
B=fy(X0,Y0);
z=@(x,y)Z0+A*(x-X0)+B*(y-Y0);
% Now we plot the tangent plane, but use only the four
% boundary points of the plotting region to get an
%uncluttered picture
X=linspace(a,b,2);
Y=linspace(c,d,2);
```

```
[X Y] = meshgrid(X,Y);
Z=z(X,Y);
surf(X,Y,Z,'FaceColor','cyan') % adjust the color of the
%patch
% The next line marks the point of tangency
plot3(X0,Y0,Z0, 'or','MarkerSize',2,'MarkerFaceColor','r')
grid on
xlabel('x');ylabel('y');zlabel('z');hold off
```

6.4 Summary

We summarize the commands used in this section:

@: - symbol to initiate an **anonymous** function

diff: - compute a symbolic derivative or partial derivative (note that `diff` can also apply to arrays and produces the so-called first difference array. This is used in computing numerical approximations for derivatives.)

pretty: - display a symbolic expression in a style that is closer to standard math output.

subs: - substitutes input values for variables in symbolic expressions. Useful for converting symbolic expressions to (vectorized) anonymous functions.

vectorize: - inserts dot operands into an expression

6.5 Exercises 

Exercise 6.1 Write an M-file in which you define each of the following functions and have Matlab compute the indicated derivative, and the value of **the derivative** for the specified inputs. Publish the M-file, making sure that you divide the published document into cells, each of which produces the answer for a single part. Do not forget to include your name at the top of the file.

a) $f(x)=x^7-3x^5+5x^4+3$: $f'''(x)$: evaluate at $x=1:5$.

b) $f(x)=e^{x\sin x}+2\ln(x^2+1)$: $f''(x)$: evaluate at $x=-pi:pi/4:pi$ (Note: use exp to enter the exponential function and log for the natural logarithm.)

c) $f(x,y)=\dfrac{x^2+y^2}{(x+y)^2}$: $f_{xy}(x,y)$: evaluate at all pairs $(1,1),(2,2),(3,3),(4,4),(5,5)$

d) $g(x,y,z)=x^3y^2z+\sqrt{x^2+y^2+z^2}$: $g_{xyz}(x,y,z)$: evaluate at the eight vertices of the cube with coordinates at $x=\pm1$, $y=\pm1$, $z=\pm1$

Exercise 6.2 Write an M-file in which you define the function $f(x)=\sin(x^2)\cos x$ and have Matlab compute the first two derivatives as functions *df* and *df2*. Using `plot`,

create a graph of the function f and the first two derivatives over the interval $[-\pi, \pi]$. Use different line styles for each plot and a label to distinguish which is which. You may want to truncate the y-axis display to the interval $[-10,10]$. This can be done using the command `ylim`. Publish your result, after dividing the M-file into appropriate cells so that the output is closely related to the input statements.

Exercise 6.3 Using the tangent plane M-file above as a model, write an M-file that plots the graph of the function $f(x) = e^{-x}\cos(2x)$ over the interval $[0, \pi]$ and draws the tangent line to the graph at the point with x coordinate $\pi/4$. Matlab should be used to compute the necessary formulas and values for derivatives.

7: Parametric Surfaces

So far we have examined surfaces defined as the graph of an equation $z = f(x, y)$. However, there are many interesting surfaces that are not graphs of functions. For example, the sphere given as the set of points satisfying $x^2 + y^2 + z^2 = 1$. This surface can be pieced together from its upper and lower hemispheres defined by $z = \sqrt{1 - x^2 - y^2}$ and $z = -\sqrt{1 - x^2 - y^2}$, respectively. For the purposes of graphing, however, this produces a poor picture because the domain of these functions (the inside of the unit circle) is not so easily approximated by a rectangular grid.

We can generalize the method of representing a surface by introducing the idea of a parametic representation. This is analogous to the parametric description of curves. Namely, we say that a surface is given parametrically by the parameters *s* and *t*, if there are functions *f*, *g*, and *h* of the variables *s* and *t* such that any point (x, y, z) on the surface satisfies

$$x = f(s,t) \quad y = g(s,t) \quad z = h(s,t).$$

Here *s* and *t* vary over some region in the *st*-plane, usually a rectangle. We can use this representation to generate points (x, y, z) on the surface, from which Matlab can construct a 3D plot. Let's consider some examples.

7.1 Basic Examples

Example 7.1: Plot the **circular cylinder** $x^2 + y^2 = 1$.

Solution: We use the θ and *z* variables of a cylindrical coordinate system for the parametric representation. Namely, on the surface of the cylinder we can take

$$x = \cos\theta \quad y = \sin\theta \quad z = z$$

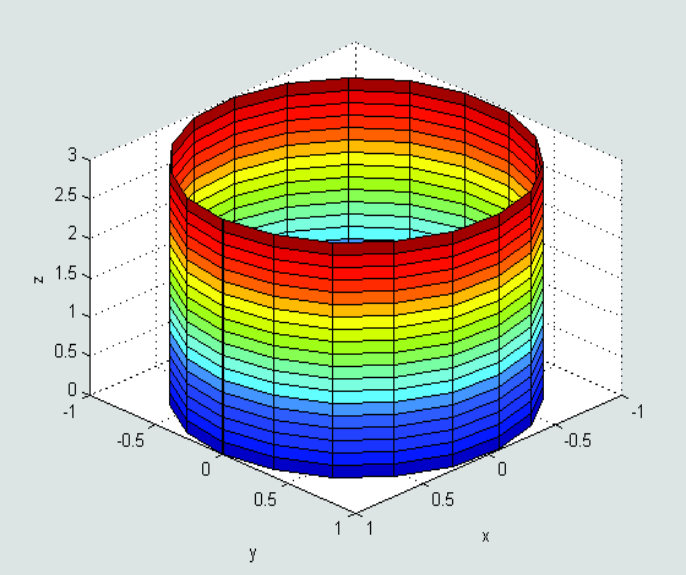
where $0 \le \theta \le 2\pi$ and $0 \le z \le 3$, (for example). To plot, we set up a meshgrid in the θ, z variables and determine the *x*, *y*, and *z* values from the above equations. Let's see how it is done. For simplicity, we will use *t* instead of θ,

```
t=linspace(0,2*pi,20);
z=linspace(0,3,20);
[t,z]=meshgrid(t,z);
```

Now we compute the values of *x*, *y*, and *z* corresponding to the values of *t* and *z* in the grid. Since $z = z$, we do not have to do any additional computation to compute these values. Once we have the data values for the points on the surface, we are in business and can use the `surf` command as usual.

```
x=cos(t); y=sin(t);
surf(x,y,z)
grid on; view([1 1 1]);
```

```
xlabel('x');ylabel('y'),zlabel('z');
```



■

In generic terms the curves we see on a parametric surface plot are the images on the surface of the grid lines in the *s*, *t*-plane where *s* or *t* is constant. For example, in the plot above, the circles are curves on which *z* is constant, while the vertical lines are curves on which θ (i.e. *t* in our notation) is constant.

We can carry out similar constructions for cylinders parallel to the other axes. For example, to plot the cylinder $x^2 + z^2 = 1$, use the parametric equations $x = \cos t$, $z = \sin t$ and $y = y$, with $0 \le t \le 2\pi$ and $a \le y \le b$, for some interval $[a,b]$.

Plotting cylinders evidently depends on being able to parametrize the trace curve that defines the cylindrical cross-section. Here are some additional commonly occurring examples:

Parabolic cylinder: $y = x^2$: Use *x* and *z* as parameters. The equations are $x = x$, $y = x^2$, $z = z$, with $a \le x \le b$ and $c \le z \le d$. We have actually used this construction earlier in Exercise 5.5

Elliptic Cylinder: $\frac{x^2}{9} + \frac{y^2}{16} = 1$. Here we can let $x = 3\cos t$ and $y = 4\sin t$, while $z = z$ is the other parameter. The range of the variables is $0 \le t \le 2\pi$ and $c \le z \le d$.

Example 7.2: **Surface of Revolution**. Describe parametric equations for the surface of revolution obtained by revolving a curve in the *x*, *z* plane around the *z*-axis. Apply the method to plotting a cone.

Solution: The cylinder in Example 7.1 is obtained by rotating around the *z*-axis the straight line $x = 1$ in the *xz*-plane. More generally, we can rotate around the *z*-axis any curve with equation $x = f(z)$ in the *xz*-plane. The resulting surface can be described

using the angle of rotation t and the z coordinate as parameters. In fact, the general equations are

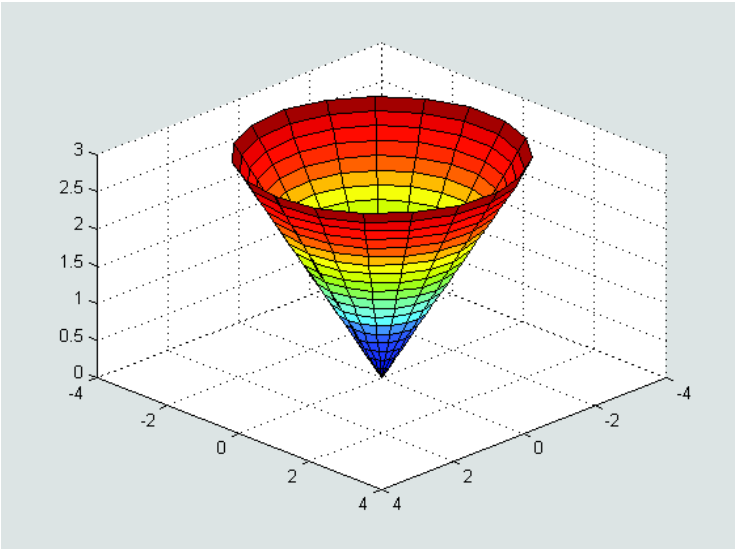
$$x = f(z)\cos t \quad y = f(z)\sin t \quad z = z\,.$$

For example, if we take the line $x = z$ and rotate it around the z-axis we obtain a cone. This can be described by the parametric equations

$$x = z\cos t \quad y = z\sin t \quad z = z\,.$$

This description is also clear from the rectangular equation of the cone, namely $z^2 = x^2 + y^2$. A Matlab plot using a meshgrid in the t, z-plane gives:

```
t=linspace(0,2*pi,20);z=linspace(0,3,20);
[t z]=meshgrid(t,z);
x=z.*cos(t);y=z.*sin(t);
surf(x,y,z); grid on; view([1 1 1]);
```



■

Example 7.3: Parametrize a **sphere** and use the parametrization to plot the sphere.

Solution: The parametrization can be obtained using spherical coordinates. For example, to plot $x^2 + y^2 + z^2 = 4$ use the equations

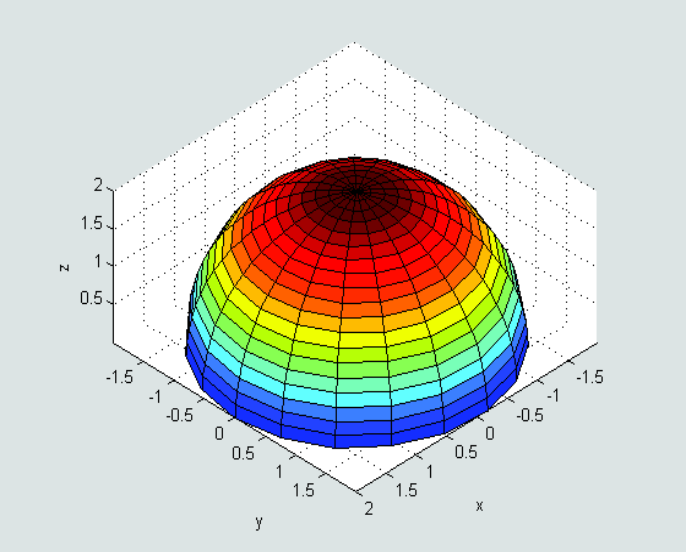
$$x = 2\cos t\sin s \quad y = 2\sin t\sin s \quad z = 2\cos s$$

For simplicity in typing, we have identified t with θ and s with ϕ in the usual notation. Thus, for the full sphere we would let $0 \le t \le 2\pi$ and $0 \le s \le \pi$. To plot the upper hemisphere we would restrict s to the interval $0 \le s \le \pi/2$. The Matlab plot is:

```
t=linspace(0,2*pi,20);s=linspace(0,pi/2,20);
[s t]=meshgrid(s,t);
```

```
x=2*cos(t).*sin(s);  y=2*sin(t).*sin(s);  z=2*cos(s);
surf(x,y,z); view([1 1 1]); grid on;
xlabel('x');ylabel('y');zlabel('z');
axis equal
```

The last command simply draws the sphere without distorting distances, so the appearance is actually spherical, instead of egg-shaped.



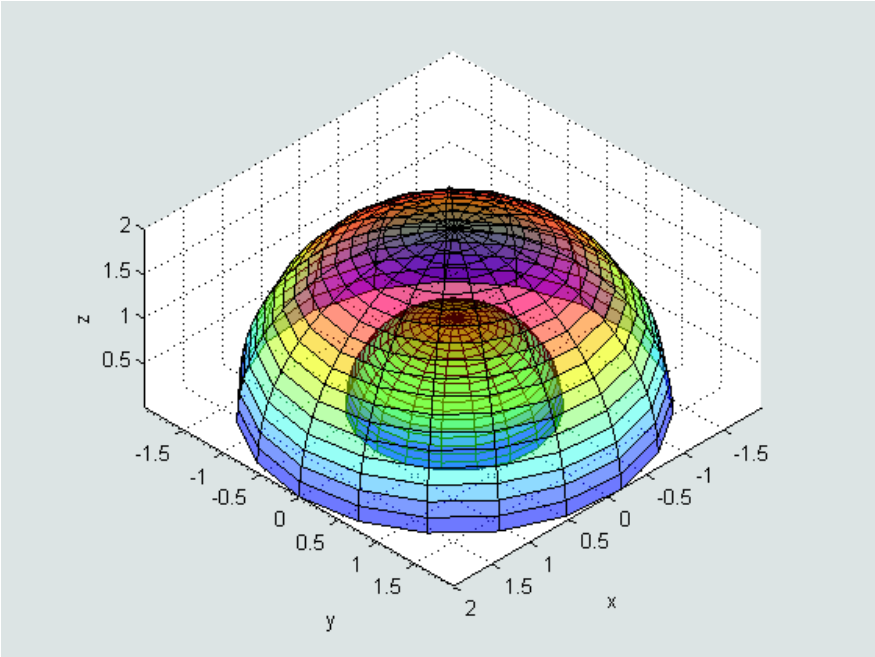
■

Example 7.4: Draw a plot of two concentric spheres with the outer sphere rendered as semi-transparent.

Solution: Using `hold on` we can easily create plots showing multiple surfaces. To assist our visualization it is sometimes useful to adjust the transparency of one of the surfaces. This is done with the property ***FaceAlpha***. This takes a numeric value between 0 and 1, with a value of 1 signifying an opaque surface (the default) and a value of 0 denoting a completely clear surface. Observe the effect of using this command in the following figure, which shows the hemisphere of radius one inside the hemisphere of radius two. The latter has been drawn as a semi-transparent surface.

```
% Hemisphere of radius two
clf; hold on;
t=linspace(0,2*pi,20);s=linspace(0,pi/2,20);
[s t]=meshgrid(s,t);
x=2*cos(t).*sin(s);  y=2*sin(t).*sin(s);  z=2*cos(s);
surf(x,y,z, 'FaceAlpha',.5);
view([1 1 1]); grid on;
xlabel('x');ylabel('y');zlabel('z');
% Hemisphere of radius one
t=linspace(0,2*pi,20);s=linspace(0,pi/2,20);
```

```
[s t]=meshgrid(s,t);
x=cos(t).*sin(s);  y=sin(t).*sin(s);  z=cos(s);
surf(x,y,z);
axis equal;
hold off
```



■

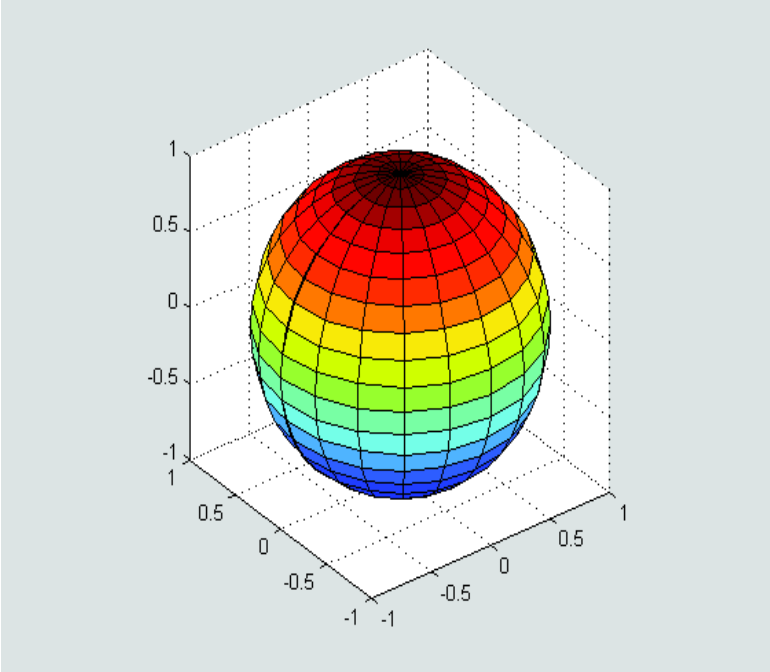
7.2 Built-in Parametric Plotting

By way of completeness, we mention that Matlab has commands for doing some of the parametric plots described above.

Example 7.5: Use the `sphere` command to draw spheres of radius one centered at the origin and at the point (1,0,0).

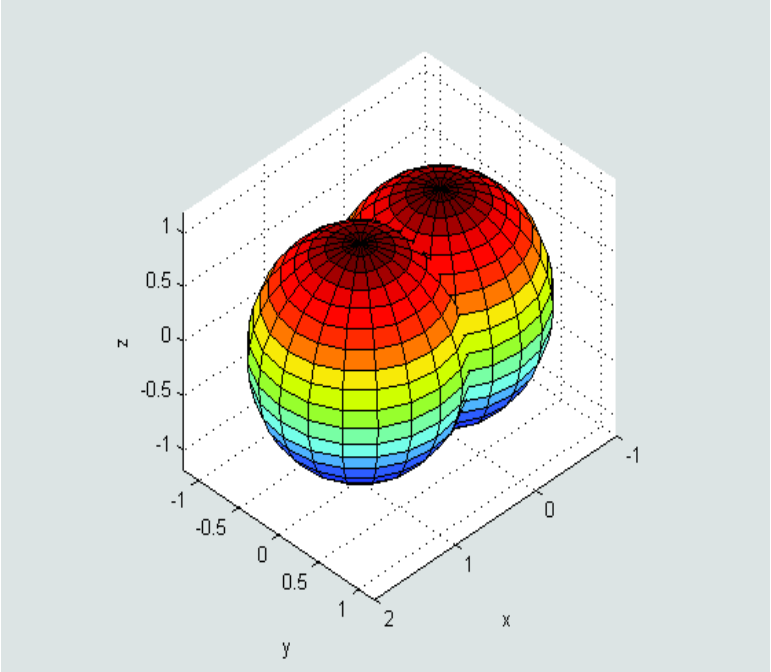
Solution: The command `sphere` generates a plot of a sphere of radius one centered at the origin. Thus

```
sphere; axis equal
```



The second sphere, with its center shifted along the x-axis, can be obtained from the first by translating all points on the latter sphere by the vector $\mathbf{v} = \langle 1 \quad 0 \quad 0 \rangle$. In order to do this, we first need to generate those data points. We can obtain the points by assigning variables $[x, y, z]$ as the output of the `sphere` command. Then we add 1 to the x coordinates of these points, leaving the others fixed. The following code will implement this and draw the new graph, together with the sphere generated above.

```
clf; hold on;
[x y z]=sphere;
surf(x,y,z)   % plots sphere centered at the origin
x=x+1;  % translates the x-coordinates
surf(x,y,z);  % plots translated sphere
axis equal; hold off;
grid on; view([1 1 1]);
xlabel('x');ylabel('y');zlabel('z');
```



■

7.3 Summary

We showed how to construct surface plots based on the parametric representation of a surface. Matlab contains some built-in parametric plotters, of which we discussed in detail the `sphere` command. Other parametric plotters are the `ellipsoid` command and the `cylinder` command. The latter can be used to construct surfaces of revolution whose rotation axis is the *z* axis. Geometric transformations are needed if you wish to use this command to plot cylindrical surfaces with other axes of rotation.

7.4 Exercises

All exercises should be done by writing appropriate M-files and publishing the results.

Exercise 7.1 Plot the portion of the parabolic cylinder $z = 4 - x^2$ that lies in the first octant with $0 \le y \le 4$. (Use parameters *x* and *y*. Determine the range of values for *x* to only plot points in the first octant.)

Exercise 7.2 Rotate the curve $x = 1 + \cos z$, for $0 \le z \le 2\pi$, around the *z*-axis. Use Matlab to draw the surface of revolution. Use `axis equal` if necessary for a non-distorted view.

Exercise 7.3 Draw the portion of the cone $y^2 = x^2 + z^2$ for $0 \le y \le 3$. (This is obtained by rotating the line $z = y$ around the *y*-axis. Use the angle of rotation *t* and the *y* variable as parameters.)

Exercise 7.4 Draw the ellipsoid whose equation is $\frac{x^2}{4} + \frac{y^2}{16} + \frac{z^2}{2} = 1$ in two ways.

a) Using explicit parametric equations base on spherical coordinates. (Observe: The quantities $x' = x/2$, $y' = y/4$, and $z' = z/\sqrt{2}$ satisfy the equation $(x')^2 + (y')^2 + (z')^2 = 1$. Use spherical coordinates for $x', y',$ and z' to parametrize the latter surface.) Use `axis equal` to obtain a non-distorted view.

b) Using the built-in function `ellipsoid`.

Exercise 7.5 The circle of radius one with center $(2,0,0)$ in the x, z-plane has parametric equations $x = 2 + \cos t$, $y = 0$, $z = \sin t$, where $0 \le t \le 2\pi$. If we revolve that circle around the z-axis we obtain a donut-shaped figure called a **torus**. Letting s denote the angle of rotation about the z-axis show that the parametric equations of this surface are

$$x = (2 + \cos t)\cos s \quad y = (2 + \cos t)\sin s \quad z = \sin t,$$

where $0 \le t \le 2\pi$ and $0 \le s \le 2\pi$. Using Matlab, draw the surface. (N.B. Use `axis equal` to obtain a non-distorted view.)

Exercise 7.6 Create a single plot showing the cylinders $x^2 + z^2 = 1$ and $y^2 + z^2 = 1$ within the box where $-2 \le x \le 2$, $-2 \le y \le 2$, and $-2 \le z \le 2$. Later we will compute the volume enclosed by the two cylinders. Make one cylinder semi-transparent using the `FaceAlpha` property.

Exercise 7.7 Create a single plot showing the portion of the parabolic cylinder $z = 4 - x^2$ that lies in the first octant with $0 \le y \le 3$, (see Exercise 7.1) and the first octant portion of the vertical plane $y = x$, with $0 \le z \le 4$. Draw the plane with a single patch using a 'FaceColor' of 'cyan'. Make the cylinder semi-transparent so you can see the vertical plane inside the cylinder.

Exercise 7.8 Create a single plot that shows the sphere $x^2 + y^2 + z^2 = 4$ and the cylinder $(x-1)^2 + y^2 = 1$. Use `axis equal` to obtain a non-distorted view. Make the sphere semi-transparent so you can see the cylinder inside of it.