Notes on using MATLAB

MATLAB is an interactive program for numerical methods, with graphing capability. These notes describe some useful functions and syntax. The following sites have more extensive tutorials:

http://www.engin.umich.edu/group/ctm/basic/basic.html

http://web.cecs.pdx.edu/ gerry/MATLAB/

http://www.cyclismo.org/tutorial/matlab/

The command for starting MATLAB depends of your system configuration (you can often start MATLAB on Unix by typing **matlab**). To obtain help from within MATLAB, type **help**; this provides a list of available functions. For demonstration of a few commands, type **demo**. To terminate a MATLAB session, type **quit**.

Formats for printing numbers.

format short	3.1416
format short e	3.1416e+000
format long	3.14159265358979
format long e	3.141592653589793e+000

There is only one data type in MATLAB, complex matrices. Vectors and scalars are special cases. Matrices can be created as follows, **A=[1,1,1,1;1,2,3,4]**. This creates a 2×4 matrix A whose first row is $(1, 1, 1, 1)$ and whose second row is $(1, 2, 3, 4)$. The dimensions of a matrix A can be found by typing **size(A)**.

To create a vector, type **x=[1,2,3,4]**. The system responds with:

```
x=
     1 2 3 4
```

The commas are optional, **x=[1 2 3 4]** gives the same result. If an assignment statement ends with a semicolon, then the result is not displayed. Thus if your type **x=[1 2 3 4];**, nothing will be displayed. You can the type **x** to display the vector. The length of a vector x is obtained from **length(x)**. Indices for vectors and matrices must be positive integers. Thus, $A(1.5, 2)$ and $x(0)$ are not allowed. There is a special syntax for creating a vector whose components differ by a fixed increment. Thus, **x=[0 .2 .4 .6 .8 1]** can be created by typing **x=0:.2:1**.

Built-in functions.

pi	3.1415...
zeros(3,3)	3×3 matrix of zeros
eye(5)	5×5 identity matrix
ones(10)	vector of length 10 with all entries =1
abs(x)	absolute value
sqrt(x)	square root, e.g. **i=sqrt(-1)**
real(z), imag(z)	real, imaginary parts of a complex number

conj(z)	complex conjugate
atan2(y,x)	polar angle of the complex number x+i y
sin(x), cos(x)	trig functions
sinh(x), cosh(x)	hyperbolic functions
exp(x)	exponential function
log(x)	natural logarithm
gamma(n)	Gamma function = (n-1)!
bessel(a,x)	Bessel function of order a at x

Example of a loop.

```
for i=1:4
    x(i)=i;
end
```

Example of a conditional statement.

```
  if a==0;
    x=a+1;
elseif a<0;
    x=a-1;
else;
    x=a+1;
end;
```

Plotting.

plot(x,y)	linear plot, uses defaults limits, **x** and **y** are vectors
grid	draw grid lines on graphics screen
title('text')	prints a title for the plot
xlabel('text')	prints a label for the x-axis
ylabel('text')	prints a label for the y-axis
axis([0,1,-2,2])	overrides default limits for plotting
hold on	superimpose all subsequent plots
hold off	turns off a previous hold on
clg	clear graphics screen
mesh	3-d plot; type **help mesh** for details
contour	contour plot; type **help contour** for details
subplot	several plots in a window; type **help subplot** for details

Example. To plot a Gaussian function, type the following lines:

```
x=-3:.01:3;
y=exp(-x.*x);
plot(x,y)
```

Matrix functions.

x=A\b	gives the solution of Ax=b
[l,u]=lu(A)	LU decomposition of A
[v,d]=eig(A)	eigenvalues in d, eigenvectors in v
[u,s,v]=svd(A)	singular value decomposition

chol(A)	Cholesky factorization
inv(A)	inverse of a square matrix
rank(A)	matrix rank
cond(A)	condition number
*, +	matrix product and sum
.*, .+	element by element product and sum (componentwise)
'	transpose, e.g. **A'**
^	power, e.g. **A^2**
.^	element by element power, e.g. **A.^2**

m-files.

An m-file is a file that contains a sequence of MATLAB commands. Some m-files are built into MATLAB. A user can create a new m-file using an editor. For example, an m-file called fourier.m could be created containing the lines:

```
%
% Plot a trigonometric function
%
x=0:.01:1;
y=sin(2*pi*x);
plot(x,y)
```

In this case, typing **fourier** would produce a plot of a sine curve. (Note: % in an m-file denotes a comment line). In order to pass arguments to and from an m-file, the world "function" must be on the first line. For example,

```
function [x,y]=fourier(n,xmax)
%
% Plot a trigonometric function
%
x=0:.01:1;
y=sin(2*pi*x);
plot(x,y)
```

Typing **[x,y]=fourier(2,7);** plots a sine curve. After execution, the vectors **x** and **y** are available for further calculations.

Useful commands.

type dft	lists the content of the m-file dft.m
save A	stores a matrix in a file called A.mat
save	saves all variables in a file called matlab.mat
load temp	retrieves all the variables from file temp.mat
print	prints the current graphics window

3-d Plotting Commands

In order to plot in three dimensions one need to specify a grid of points in the x-y domain; this is much like specifing the ordered pairs in the plane where points will be plotted. The Matlab command **meshgrid** achieves this as follows.

```
>>x=a:dc:b;
>>y=c:dy:d;
[X,Y]=meshgrid(x,y);
```

The last command creates two matrices X and Y of size length(y) by length(x) (row by column). The element by element correspondence between these matrices is the collection of ordered pairs forming the grid points for plotting. An explicit example shows how this is done:

```
>> x=[1 2 3];
>> y=[4 5 6 7];
>> [X,Y]=meshgrid(x,y)
X =
1 2 3
1 2 3
1 2 3
1 2 3

Y =
4 4 4
5 5 5
6 6 6
7 7 7
```

Notice that X increases along the columns from left to right in the x variable; Y increases along the rows from top to bottom in the y variable. If a function is computed on this grid, say $z = f(X, Y)$, then the command

```
>plot3(X,Y,z)
```

forms a linear plot much like that of the two dimensional plot command.

The **plot3** command is very useful in plotting space curves. For example the helix is plotted with the commands

```
>> t=0:pi/30:6*pi;
>> plot3(cos(t),sin(t),t)
```

The commands used in 2d plots for titles, controling axis tick marks, axis labels, legend, etc. all work the same in 3 dimensions.

Surface Plots

Matlab provides two key commands for plotting surfaces: **mesh** and **surf**. **mesh** takes the 3-d data and creates a wire mesh through adjacent points. On the other hand, **surf** creates a mesh plot with the spaces between the lines, called patches, filled in according to a color scheme based on the z data. Here is an example

```
>> x=0:pi/20:pi;
>> y=x;
>> [X,Y]=meshgrid(x,y);
>> f=inline('sin(2*x).*cos(3/2*y)','x','y')
f =

Inline function:
f(x,y) = sin(2*x).*cos(3/2*y)

>> %Note the function is vectorized
>> subplot(1,2,1),mesh(X,Y,f(X,Y))
>> title('A Mesh Plot')
>> subplot(1,2,2),surf(X,Y,f(X,Y))
>> title('A Surf Plot')
```

NOTE: Either of the sub-figures can be viewed from different view points. On the tool bar select the button for 3-d rotation. This allows grabbing the axis with the mouse and rotating to a different view. This can also be achieved from the command line. Type: **help view** to find out how.

Colormap

The color scheme for suface plots can be easily changes. Some built in color maps are: **hot, cool, gray, copper, summer, winter, bone**. Here is the above plot with different color schemes. You can achieve this by clicking on a subfigure and typing at the prompt

```
>>colormap(whatever you choose)
>>colorbar
```

The color bar depicts how Matlab associates a particular shade of copper with the numerical values of the function.

Shading in Surf Plots

A shading effect in surf plots can be achieved; this controls how color is interpolated between the lines. The default is **faceted** (above figures) meaning a stained glass effect with constant color patches. Other possibilities are **flat** (lines removed but each piece has constant color) and **interpolated** (lines removed and color of each patch interpolated between the edges).

As an example, and using the same function as in previous examples.

```
>> subplot(1,2,1),surf(X,Y,f(X,Y))
>> colormap(bone)
>> shading flat
>> title('Flat Shading')
>> subplot(1,2,2),surf(X,Y,f(X,Y))
>> shading interp
>> title('Interpolated Shading')
```

Plotting in Cylindrical Coordinates

Given a function $z = f(r, theta)$, one may desire a surface plot on a circular **meshgrid**. Combining the **meshgrid** command with a command to convert polar coordinates to rectangular this is easily achieved. The following example will get you started.

```
>> [th,r]=meshgrid(0:pi/40:2*pi,0:0.05:1);
>> [X,Y]=pol2cart(th,r);
>> g=inline('r.^n.*sin(n*th)','r','th','n')
g =

Inline function:

g(r,th,n) = r.^n.*sin(n*th)
>> surf(X,Y,g(r,th,5))
>> hold on
>> mesh(X,Y,-ones(size(X)))
>> title('A Cylindrical Plot')
```

Contour Plots

Given mesh data, contour plots can be generated using the command **contour**. For example:

```
>> x=0:0.5:6;
>> t=0:0.5:20;
>> [X,T]=meshgrid(x,t);
>> g=inline('cos(x-0.4*y).*exp(-0.4*x)','x','y')
g =

Inline function:
g(x,y) = cos(x-0.4*y).*exp(-0.4*x)
>> contour(X,T,g(X,T))
>> colorbar
>> title('Damped Traveling Wave')
>> xlabel('x')
>> ylabel('t')
```

This generates the following figure, a spacially damped traveling wave.

Contours can be combined with **surf** and **mesh** plots, for example

```
>> surfc(X,Y,g(X,Y))
% surfc or meshc are the commands
>> xlabel('x')
>> ylabel('t')
>> colormap(bone)
```

Finally, the combination of patch shading and contour can be achieved with the command **pcolor**.

```
>> x=0:0.5:6;
>> t=0:0.5:40;
>> [X,T]=meshgrid(x,t);
>> pcolor(X,T,g(X,T))
>> shading interp
>> hold on
>> contour(X,T,g(X,T),'k')
% the 'k' forces the contour lines to be in black
>> colorbar
>> title('Traveling Wave with pcolor')
>> xlabel('x')
>> ylabel('t')
```