MATLAB INSTRUCTIONS

ASE 211, Fall 2006

1. Matlab Access

For this class you will need access to a working version of Matlab. Matlab is available on all the PC's in the Aerospace LRC. To use these machines you will need to obtain an account at the LRC. More information about obtaining an account can be found at the LRC website (http://www.ae.utexas.edu/lrc) under "User Accounts". If you have any questions or problems please contact any of the proctors for help. If you want you may buy the student version of Matlab to use on you own computer. It should be available at the Campus Computer Store, if not you can get it from the mathworks website (http://www.mathworks.com).

2. Matlab Hints

Programming in Matlab will make doing the assignments much easier, once you get used to it. There are only a few things we need to be able to do in Matlab. First, you have to know how to start a Matlab session. Look for a Matlab icon in the window of the machine you are on. If you can't find one, then click on 'start' followed by 'programs'. Matlab should be in the list of programs. If not, contact one of the proctors for help.

Once in Matlab, you should get a prompt that looks like

>>

Now you're ready to start entering commands. Some basic facts:

- Matlab is case sensitive.
- Every command given to Matlab is followed by output from Matlab. Type a semicolon after the command to suppress output.
- The standard arithmetic experessions are used,
  +, -, *, /, ^.
- The standard functions are used sin, cos, tan, log, exp, etc.
- Matlab can be used in two different modes, interactive or script.

  Interactive mode is when commands are entered in the command window. Matlab processes these commands immediately and displays the results. An interactive session can be saved using the "diary" command. Type "diary" at the prompt

before your session and all commands and output will be saved in a file called "diary" in the Matlab folder.

Script mode is like writing a computer program. A Matlab program is called an "m-file". The commands are programmed using the Edit window, and can be saved for later use as an "m-file". An example of such a file is given below.

- Help with Matlab can be obtained by typing "help" at the prompt.

Example of interactive commands. This session was saved by typing "diary" at the >> prompt.

```
>> a=3

a =

     3

>> b=6;
>> c=a*b

c =

    18

>> d=a/b

d =

    0.5000

>> e=c^d

e =

    4.2426

>> f=log(c)

f =

    2.8904

>> g=exp(f)
```

```
g =

    18.0000

>> A=g-a

A =

    15.0000

>> a

a =

     3

>> diary
```

Try typing these commands into Matlab.

Vectors and matrices. Suppose you want to enter a column vector b

$$b = \begin{bmatrix} -2 \\ 3 \\ 13 \end{bmatrix}.$$

Then you would type

```
>> b = [-2; 3; 13]
```

Note that semicolons are used to separate the rows in the vector.

Suppose you want to enter a matrix

$$A = \begin{bmatrix} -3 & 0 & 1 \\ 1 & -2 & 6 \\ 8 & 1 & 3 \end{bmatrix}$$

Then you would type

```
>> A = [-3 0 1; 1 -2 6; 8 1 3]
```

Notice that the rows in the matrix are separated by a semicolon.

Matlab will now have stored a vector *b* and a matrix *A*. To display the entries of *A*, for example, just type

```
>> A
```

If you want to see a particular entry in *A*, say A_{23}, type

```
>> A(2,3)
```

Here is an example:

```
>> b = [-2; 3; 13]

b =

    -2
     3
    13

>> A = [-3 0 1; 1 -2 6; 8 1 3]

A =

    -3     0     1
     1    -2     6
     8     1     3

>> c = A\b

c =

     1
     2
     1

>> A*c

ans =

    -2
     3
    13

>> diary
```

Notice that the command c = A\b actually solved the equation A*c = b. Matlab has a number of built-in commands for manipulating and solving systems of equations. We will discuss this more later in the course.

Plotting functions. Matlab has a number of built-in plotting functions. For example, suppose you want to plot $f(x) = \sin(\pi x)$ for x between 0 and 1. Then we want to create a vector X containing a bunch of points between 0 and 1, and create another vector Y containing the value of f at each of these points. We can then plot the vectors. Here's how we can do this:

```
>> X = (0:.01:1);
```

```
>> Y = sin(pi*X);
>> plot(X,Y)
>> xlabel('X')
>> ylabel('SIN(PI*X)')
>> title('Plot of x vs. sin(pi*x)')
>> print plot1.ps
>> diary
```

The first command above is actually an implicit loop. It creates a vector X which goes from 0 to 1 by .01. Then we create a vector *Y* whose entries are $\sin(\pi * X(i))$. Then we plot *X* vs. *Y*. The "xlabel, ylabel" and "title" commands create labels on the *x* and *y* axes of the plot, and a title above the plot. The "print" command saves the plot in a postscript file called plot1.ps. Instead of using these commands you can use the menus in the plot window to create the labels and title and also to print out the plot.

The output of the plot is seen in Figure 1.

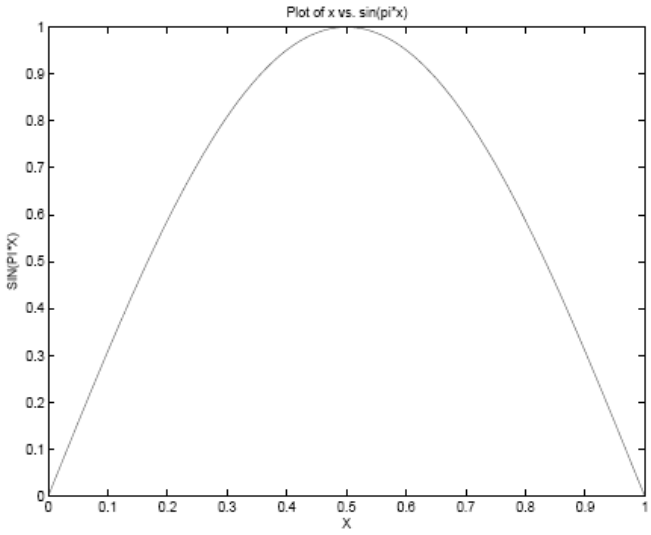

Figure 1

M-Files. An m-file is simply a text file containing a sequence of commands, similar to a Fortran or C++ program. Suppose that you want to create an m-file that takes as input a matrix and a vector, multiplies them, and prints the answer. Call the file *matmul.m*. You should create this file using a text editor and save it in your Matlab folder.

```
function matmul(A, b)
% This function multiplies a matrix A times a vector b and prints the result
c = A*b;
disp('A*b=')
c
```

Here the *disp* command displays on the screen whatever is in quotes.

Create such a file and enter the matrix *A* and vector *b* as described above. In Matlab, type

```
>> matmul(a, b)
```

Matlab will find the file *matmul.m* and invoke the commands. Assuming there are no syntax errors in your m-file the solution should print.

Matlab allows loops and if-statements just as C++ does. The only difference is in the syntax.

An example of an if-statement:

```
if (c = = 0 && b > 1)
    d = b;
else
    disp('error')
    break
end
```

The statement says that 'if c equals zero and b is greater than 1, then d equals b, otherwise an error has occurred and the program should stop.' The *break* statement stops the program.

An example of a for-loop:

```
for i = 1:n
    a(i) = b(i) + c(i);
end
```

This loop increments *i* from 1 to *n* by 1. Suppose you want *i* to go from *n* to 1 by -1.

```
for i = n:-1:1
    a(i) = b(i) + c(i);
end
```

This reads 'for *i* going from *n*, decreasing by -1, to 1 …'

An example of a while loop:

```
while (k < maxk)
    k = k+1;
end
```

Here's a more complicated m-file *matmanip.m* which takes an input two *n* x *n* matrices *a* and *c* and does some things with them

```
function matmanip(a, c, n)
%
```

```
% Multiply a and c the hard way
%
for j = 1:n
      for i = 1:n
            b(i, j) = 0.0;
            for k = 1:n
                  b(i, j) = b(i, j) + a(i, k)*c(k, j);
            end
%
% Modify b(i, j) if it is negative
%
            if (b(i, j) < 0)
                b(i, j) = 0;
            end
      end
end
b
%
% Multiply a and c the easy way
%
a*c
k = 0;
while (k < 3)
      k = k+1;
      for i = 1:n
            for j = 1:n
                  d(i, j) = b(i, j)^(1/k);
            end
      end
      d
end
```

Here is a sample Matlab session using this m-file.

```
>> a=[3 4; 1 6]

a =

      3      4
      1      6

>> c = [2 7; 22 1]

c =
```

```
     2     7
    22     1

>> matmanip(a, c, 2)

b =

    94    25
   134    13


ans =

    94    25
   134    13


d =

    94    25
   134    13


d =

    9.6954    5.0000
   11.5758    3.6056


d =

    4.5468    2.9240
    5.1172    2.3513

>> diary
```

Matlab functions can also return variables. For example, suppose you wanted to return the matrices *b* and *d* generated in *matmanip.m*. Then the first line in the file *matmanip.m* should be changed to

function [b,d] = matmanip(a, c, n)

The Matlab session above now looks like this:

```
>> a=[3 4; 1 6]
```

```
a =

     3     4
     1     6

>> c = [2 7; 22 1]

c =

     2     7
    22     1

>> [b, d] = matmanip_f(a, c, 2)

b =

    94    25
   134    13


ans =

    94    25
   134    13


d =

    94    25
   134    13


d =

    9.6954    5.0000
   11.5758    3.6056


d =

    4.5468    2.9240
    5.1172    2.3513
```

```
b =

    94    25
   134    13


d =

    4.5468    2.9240
    5.1172    2.3513

>> diary off
```

Notice that now *b* and *d* are printed within *matmanip* and after the return from *matmanip*.

The only real way to get comfortable with Matlab is to play around with it. So get cracking!