INTRODUCTION TO MATLAB

KARL VOSS

1. Introduction

MATLAB if a software package for scientific computations. MATLAB stands for MATrix LABoratory. The power of MATLAB is its simplicity and that it is a matrix/vector-oriented programming language which can take advantage of the "vectorized" calculations which occur repeatedly in engineering, science, and mathematics.

To start MATLAB on one of the PCs, log in, click START and click PROGRAMS and find MATLAB and click on it. All commands in MATLAB are entered at the prompt `>>`. In these notes, all MATLAB commands are written in `typewriter font` to distinguish them from the other text. In the displayed examples, both the input and output are shown in `typewriter font`. To end a MATLAB session type `>>quit` or `>>bye`.

In MATLAB you can get help in several ways. To find out about a particular command type `>>help` followed by the command name. For example, try typing `>>help +`. MATLAB is case-sensitive so upper and lower case letters are distinguished. The command `>>helpwin` will provide you with a table of contents. The command `>>helpdesk` will open a web browser with access to on-line MATLAB help. The on-line help is quite good and covers much more than what is in these notes.

Another important aspect of MATLAB is managing you variables. To list all variables which MATLAB is currently using type `>>who`. If you use `>>whos` instead you will also get the size of each of the variables. To erase all of the variables in use, type `>>clear`. To erase only specific variables type`>>clear` followed by the names of the variables you want erased.

2. Basics

Unsurprisingly, `+`, `-`, `*`, `/`, and `^` stand for addition subtraction, multiplication, division and exponentiation respectively. For example,

```
>> 2+2

ans =

     4
```

In MATLAB, π is written as `pi`.

```
>> pi^3

ans =

   31.0063
```

A partial list of common mathematical functions is presented in the table below.

Date: 8/23/04.

`sin`	Sine
`asin`	Inverse Sine
`cos`	Cosine
`acos`	Inverse Cosine
`tan`	Tangent
`atan`	Inverse Tangent
`sinh`	Hyperbolic Sine
`exp`	Exponential
`log`	Natural Logarithm
`log10`	Logarithm base 10
`sqrt`	Square Root
`fix`	Round Toward Zero
`floor`	Round toward $-\infty$
`ceil`	Round toward ∞
`round`	Round to Nearest Integer

For example,

```
>> sin(pi/2)


ans =


     1
```

There are many other functions and we will introduce them as needed. All MATLAB calculations are done in double precision (we will be discussing what this mean shortly). The format of MATLAB output is controlled by the user. To change the format, type `>>format` followed by `short`, `short e`, `long`, `long e`, `hex`, `rat`, or `bank`. `short` indicates only five digits will be shown. `long` shows fourteen. Scientific notation is indicated by `e`. `hex` is for hexadecimal, `rat` is the closest ration approximation MATLAB can find, and `bank` presents the result with exactly two decimal places.

3. Vectors

The row vector $x = [1, 2, 3]$ is entered as

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


x =


     1     2     3
```

If you want a column vector, use semicolons between the rows.

```
>> y=[1 ; 2 ; 3]


y =


     1
     2
     3
```

Many operations can be done term by term on vectors (and as we shall see matrices too). For example,

```
>> z=sin(x)


z =


    0.8415    0.9093    0.1411
```

or

```
>> a=z+x


a =


    1.8415    2.9093    3.1411
```

To multiply two vectors together term by term you need to use `.*`.

```
>> x.*x


ans =


     1     4     9

```

Question Experiment to find out what `>>x*x` does.

To suppress the output, append a semicolon to the end of the command line. Both commands below set y to $x + [1\ 1\ 0]$ but only one displays the value of y.

```
>> b=x+[1 1 0];
>> b=x+[1 1 0]


b =


     2     3     3

```

Column and row vectors can be transposed using an apostrophe, `’`.

```
>> b’


ans =


     2
     3
     3

```

The length of the vector, x, can be found by typing `>>length(x)`. There is an alternative way to enter certain special vectors. The command `>>v=i:j:k` sets v to a vector whose first entry is i, whose last entry is k, and the difference between successive entries is j. If only two values are given, j is assumed to be 1. For example try, `>>v=0:10:100` or `>>v=2:10`.

A similar command is `>>linspace(a,b,n)`. This produces a vector of length n, whose first entry is a and last entry is b. The intermediate entries are evenly spaced.

4. Matrices

Much of what we have seen about vectors applies to matrices. In MATLAB there is only one data type, the complex matrix. In the previous calculations single numbers were treated as 1×1 matrices and the vectors as $1 \times n$ or $n \times 1$ matrices. The matrix

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 0 \end{pmatrix}$$

is entered as

```
>> A=[1 2 3;4 5 0]


A =


     1     2     3
     4     5     0

```

This matrix can also be entered as

```
>> A=[1 2 3
      4 5 0];
```

or

```
>> A=[1 2 3; 4 ...
       5 0];
```

Question What happens if you enter `>>A=[1 0;1]`? Why?

The command `>>size(A)` gives the dimensions of A as number of rows and number of columns.

```
>> size(A)


ans =


     2     3
```

This indicates that A has 2 rows and 3 columns. You can take the transpose of a matrix in the same way as a vector. What does the command `>>size(A')` yield?

The command `>>A(i,j)` is the entry at the ith row and jth column of A.

```
>> A(2,3)


ans =


     0
```

Question What does the command `>>A(2,3)=6` do?

You can extract any part or sub-matrix of a larger matrix. For example, `>>A(m:n,k:l)` is the sub-matrix of A made up of the mth to the nth rows of A and the kth to the lth columns.

```
>> A(1:2,1:2)


ans =


     1     2
     4     5
```

A lone colon indicates all rows or columns.

```
>> A(1:1,:)


ans =


     1     2     3
```

For convenience, MATLAB has several built in matrices.

1. `eye(m,n)` is an m by n matrix with ones on the main diagonal and zeros elsewhere.
2. `zeros(m,n)` is an m by n matrix with zeros everywhere.
3. `ones(m,n)` is an m by n matrix with ones everywhere.
4. `rand(m,n)` is an m by n matrix of random numbers between zero and one.
5. `diag(v)` is an `length(v)` by `length(v)` matrix which has the vector v on the main diagonal.
6. `diag(A)` is a vector whose entries come from the main diagonal of A.
7. `diag(A,1)` is the same as before but the vector is created with the entries from the diagonal one above the main diagonal.
8. `[]` is the null vector.

This is only a very partial list.

5. Manipulating Matrices

Let

```
>> A=rand(3,3)


A =


    0.9331    0.1010    0.4051
    0.3660    0.4310    0.5128
    0.5834    0.2137    0.9114
```

```
>> B=ones(3,1)


B =


     1
     1
     1


>> C=zeros(1,3)


C =


     0     0     0
```

To add a row to A use

```
>> D=[A;C]


D =


    0.9331    0.1010    0.4051
    0.3660    0.4310    0.5128
    0.5834    0.2137    0.9114
         0         0         0
```

You can add a column to A, by typing

```
>> D=[A B]

D =


    0.9331    0.1010    0.4051    1.0000
    0.3660    0.4310    0.5128    1.0000
    0.5834    0.2137    0.9114    1.0000
```

To erase a row or column use the null vector

```
>> A(2,:)=[]


A =


    0.9331    0.1010    0.4051
    0.5834    0.2137    0.9114


>> A(:,1:2)=[]


A =


    0.4051
    0.9114
```

Question 3 Execute the following commands and describe the relationship between A in the beginning, B, C, and A in the end. Perform these command without the semicolon if it helps.

```
>>A=rand(10,10);
>>B=rand(10,10);
>>C=rand(1,20);
>>A([1 3 6 7],:)=[B(1:3,:);C(5:14)]
```

Several of the arithmetic operations which work with vectors also work with matrices. For example,

```
>> A=[1 2;3 4];
>> B=[5 6;7 8];
>> A+B


ans =


     6     8
    10    12


>> A-B


ans =


    -4    -4
    -4    -4


>> A*B


ans =


    19    22
    43    50


>> A.*B



ans =


     5    12
    21    32


>> A./B


ans =


    0.2000    0.3333
    0.4286    0.5000


>> A.^B


ans =


           1          64
        2187       65536
```

Note the difference between `A*B` and `A.*B`. The former is matrix multiplication and the latter is component by component multiplication. As with vectors, `.*`, `./`, and `.^` tell MATLAB to perform the operations component by component. One can also divide matrices as in

```
 >> A/B


ans =


    3.0000   -2.0000
    2.0000   -1.0000
```

where A divided by B means A times the inverse of B. We will discuss what this means later. All of the above operations work if B is a scalar. For instance, `>>A+B` would be equivalent to `>>A+B*ones(size(A))` and so on for the other commands. The command `>>A^2` is equivalent to `>>A*A`.

The relations `<`, `>`, `<=`, `>=`, `==`(equal), `~=`(not equal) can be used between matrices of the same size. The command `>>A<B` is a matrix of `size(A)` where each entry is a one or a zero

depending on whether the relation is true or false component by component.

6. Plotting

There are several ways to plot using MATLAB. As an example consider plotting $y = \sin x$ from 0 to 2π. First we set up the independent variable, then define the function. After plotting the function, we add a few frills to the plot. When you type the plot command a graphics window will open.

```
>> x=linspace(0,2* pi,20);
>> y=sin(x);
>> plot(x,y,'ok')
>> title('sin(x)')
>> xlabel('x')
>> ylabel('y')
```

Question What type of mathematical object is x in this example? What type must y be? Use help to figure out what `'ok'` did in the plot command. Can you change the command so that $\sin x$ is plotted with a green solid curve?

Using a simple for loop, we can give another way of defining a function.

```
>> for i=1:20
          z(i)=cos(x(i));
   end
>> plot(x,y,'ok',x,z,'+r')
>> title('sin(x) and cos(x)')
>> xlabel('x')
>> ylabel('y and z')
```

Question Which of these two examples is taking advantage of MATLAB's "vectorized" calculations?

At the end of this handout you can find a broad range of graphics options available to plot data in two or three dimensions on MATLAB.

7. Script and function files

A script file is a sequence of MATLAB commands stored in a file. A function file is like a script file but it also has a function definition which defines the input and output explicitly. Anything after `%` is a comment which MATLAB ignores.

To write a script or function file you must open an editor. The editor can be opened by clicking on the first icon on the left side of the toolbar in MATLAB. In the editor type the following script:

```
%Script File: circle
%
%This script will plot a circle whose
%center and radius are entered by the user.


r=input('Enter the radius of the circle:')
X0=input('Enter the x-coor. of the center:')
Y0=input('Enter the y-coor. of the center:')
theta=linspace(0,2*pi,100);
x=r*cos(theta)+X0;
y=r*sin(theta)+Y0;
plot(x,y);
axis('equal');
title('You chose this circle.');
```

All scripts and function files are stored in files with a `.m` extension. You should always include comments in m-files which give the name of the file, explain the purpose of the file, and list input and output where appropriate. If you type `>>help` *filename*, MATLAB will print the comments at the start of the file. Save the above file as `circle.m`. Returning to MATLAB you can run this script by typing `>>circle`. Try typing `>>help circle`. All variables in scripts are global so to avoid problems never call a script file with the same name as the variable it computes.

The first line of a function file should be of the form `function`[*output variables*] = *function_name* (*input_variables*). The following function file converts a pair of temperatures from Fahrenheit to Celsius. As before you will need to open an editor and save the file as `Temp.m`

```
function Ctemp = Temp(temp1, temp2)
%  Function file:  Temp
```

```
%
%  This function file will convert a
%  pair temperatures from Fahrenheit
%  to Celsius.  The input is a pair
%  of numbers and the output is a
%  pair of numbers.


F=[temp1,temp2];
Ctemp=(F-32)*5/9;
```

To use a function file you need to call the function by its name and give the input the function needs in parenthesis. For example, you can type `>>Temp(10,80)` to covert 10 degrees and 80 degrees Fahrenheit to Celsius. You should always have the name of the function and the name of the function file match. Be careful to save the file in the same directory you are running MATLAB in or MATLAB may not be able to find the file. To see the contents of an `.m` file without opening the file in an editor, type `type file.m`.

To find all of the script and function files available in the current directory one can type `>>what`.

8. Control-Flow Statements

MATLAB has syntax for looping and branching much like `C`. There are three basic control flow statements.

The first is the `for` loop. `For` loops repeat a set of code a specific number of times. In a `for` loop, the commands following `>>for n=a:b` and before `end` are repeated $b - a + 1$ times and n is incremented by one on each loop. The command `>>for n=a:c:b` starts n at a and increments n by c every loop until n reaches b. The following statements find an approximation of sin(0.2) using a Taylor approximation of degree 7 centered at 0 (we will be reviewing the meaning of this).

```
>> x=0.2;
>> approx=0;
>> for n=1:2:7
      approx=(x)^n/factorial(n) + approx;
   end
>> approx


approx =


    0.2013


>> sin(0.2)


ans =


    0.1987
```

The command `factorial(n)` calculates the factorial of n or in mathematical notation, $n! = 1\cdot 2\cdot 3\cdot 4\cdots n$. A `while` loop performs an indefinite number of calculations until the condition in the while statement is false. Try to run

```
>> v=[];
>> i=0;
>> while length(v)<100
      i=i+1;
      v=primes(i);
   end
>> i


i =


   541
```

The command `v=primes(i)` sets v to a row vector whose entries are all of the primes less than or equal to i. The `while` loop continues as long as the number of primes (`length(v)`) is less than 100. When the `while` loop fails, i will be the one hundredth prime.

The last of the major control-flow commands is `if-else`. This command provides logical branching within MATLAB. The following function gives the value of sgn(x). That is, it returns 1, 0 or -1 depending on whether x is positive, zero, or negative, respectively. In this example, there are two nested `if-else` commands, The `else` part of the command is optional.

```
function s=signof(x)
%  Function file: sign of x
%
%  This function will calculate
%  the signum of x.  x is the input
%  and the output is -1,0,1
%  depending on whether x is positive,
%  negative or zero.


if x==0
   s=0;
 else
   if x>0
     s=1;
   else
     s=-1;
     end
end
```

Other control commands include, `break` which terminates a `for` or `while` loop immediately. The command `error('`*message*`')` will print *message* and return control to the keyboard. The command `pause` will halt MATLAB's processing. MATLAB will start again as soon as any key is hit.

9. Diary

You can record all of the work you do in a MATLAB session and edit it later using the diary command. To do this type
`>>diary` *name_of_diary*, then everything which happens during your session will be recorded in *name_of_diary*. To stop the recording, use `>>diary off`. If you wish to start recording again in the same diary as before, type `>>diary on`. After recording your session, you can edit the diary for others to look at.

To save or print a picture use the pull-down menus on the graphics window to save or print your file.