AMATH 352 Lecture 3
MATLAB Tutorial

MATLAB (short for MATrix LABoratory) is a very useful piece of software for numerical analysis. It provides an environment for computation and the visualization. Learning MATLAB is not the goal of this course, but a working knowledge of MATLAB will allow you to implement and test the algorithms that form the heart of this course. Seeing these algorithms at work will hopefully enable a deeper understanding of the mechanics, strengths and pitfalls of each algorithm.

Starting MATLAB

To start MATLAB, click on the MATLAB icon or type `matlab` at the command line prompt. To exit, type `quit`.

Entering Variables

There are three types of variables that you can use in matlab: scalars, vectors and matrices. To assign a value to a scalar variable, type

```
>> x = 0.2
```

After hitting return, MATLAB will echo the value of the variable back to you:

```
x =

    0.2000
```

If you don't want to see the value of the variable, add a semicolon at the end of the line:

```
>> x = 0.2;
```

To enter a vector or matrix, use square brackets to indicate the start and end of the vector/matrix. For example a row vector may be entered:

```
>> y = [ 0 1 2 3 4]

y =

     0     1     2     3     4
```

To enter a variable with more than one row, the rows of the vector or matrix are separated by semicolons or by carriage returns. For example a column vector may be entered:

```
>> z = [ 0; 1; 2; 3; 4]

z =

     0
     1
     2
     3
     4
```

A matrix may be entered similarly, with the rows of the matrix separated in this case by carriage returns and the whole expression enclosed in square brackets:

```
>> A = [ 0 2
         3 7
        12 8]

A =

     0     2
     3     7
    12     8
```

The (complex conjugate) transpose of a vector or matrix may be obtained by placing an apostrophe after the expression:

```
>> w = z′

w =

     0     1     2     3     4

>> B = A′

B =

     0     3    12
     2     7     8
```

To give a variable a set of evenly-spaced values, use the colon operator:

```
>> t = 0:6

t =

     0     1     2     3     4     5     6

>> u = 0:0.3:1.8

u =

         0    0.3000    0.6000    0.9000    1.2000    1.5000    1.8000
```

The first and last numbers are the starting and ending points for the series. The middle number is the spacing between the members of the series. If no spacing is given, MATLAB assumes a spacing of 1.

Accessing elements of a vector/matrix

The individual elements of a vector or matrix may be accessed and/or changed individually by specifying the row and/or column number of the element. In a row or column vector, only a single number is required. In a matrix, both the row and column number must be specified with `A(i,j)` choosing the ith row and jth column.

```
>> y(3)

ans =
     2

>> A(3,2)

ans =
     8
```

However, you must specify a position within the matrix or vector. Otherwise, MATLAB will complain:

```
>> A(2,3)
???  Index exceeds matrix dimensions.
```

You may change individual elements of a matrix or vector in this way:

```
>> y(3) = y(3) + 2

y =
     0     1     4     3     4

>> A(3,2) = A(3,2) - 3

A =
     0     2
     3     7
    12     5
```

You may also select parts of a matrix or vector, using `B(2,1:2)` to specify the first two elements of the second row of B or `B(2,:)` for the entire second row of B.

```
>> B(2,:)

ans =

     2     7     8
```

Often you need to access elements at or near the end of a vector. You can do this by typing `y(end)` or `y(2:end)` or `y(2:end-1)`. This is very handy, especially when you want to plot part of a vector.

Getting information about variables

To see the size of a variable or its length (number of rows), type `size(A)` or `length(A)`. To see all of the variables that are currently in the MATLAB workspace, type `whos`:

```
>> whos
  Name      Size         Bytes  Class

  A         3x2             48  double array
  B         2x3             48  double array
  ans       1x3             24  double array
  t         1x7             56  double array
  u         1x7             56  double array
  w         1x5             40  double array
  x         1x1              8  double array
  y         1x5             40  double array
  z         5x1             40  double array

Grand total is 45 elements using 360 bytes
```

Elementary computations

Variables may be added, subtracted, multiplied and divided as long as the rules of arithmetic and linear algebra are obeyed, i.e. you can't divide by zero or multiply a 1x2 matrix by a 3x4 matrix. Multiplying a vector or matrix by a scalar will scale each element of the vector or matrix by the value of the scalar.

```
>> C = 2*[1 2; 3 4]

C =

     2     4
     6     8

>> v = 2*[1 2 3 4]

v =

     2     4     6     8
```

Adding a scalar to a vector or matrix will add the value of the scalar to **each** element of the matrix:

```
>> D = 2+[1 2; 3 4]

D =

     3     4
     5     6
```

```
>> s = 2 + [0:5]

s =

     2     3     4     5     6     7
```

Matrices and vectors may be added or subtracted as long as they are the same size. They may be multiplied as long as there are the same number of columns in the first as there are rows in the second.

```
>> s + 2*[0:5]

ans =

     2     5     8    11    14    17

>> A*B

ans =

     4    14    16
    14    58    92
    10    71   184

>> B*A

ans =

   153    81
   117    93
```

Putting a period in front of the multiplication, division or power operator performs that operation componentwise, i.e. $(\texttt{x.*y})_i = x_i \cdot y_i$ or $(\texttt{x.}\wedge\texttt{2})_i = x_i^2$.

```
>> [1 2 3 4].^2

ans =

     1     4     9    16

>> [1 2 3 4].*[5 0 5 0]

ans =

     5     0    15     0
```

For loops

To run a command more than once as an index varies, you may use a for loop. In the following example, a for loop is used to compute $n!$ for $n = 1, \ldots, 10$:

```
>> j = 1;
>> for i = 1:10
       j = j*i
end
```

You may also do the loop with the values of `i` in the matrix $[0, 2, 3, 6]$:

```
>> for i = [ 0 2 3 6]
       DO SOMETHING
end
```

Note that MATLAB will not run the for loop until you have hit return after typing `end` to indicate the end of the for loop. I find it useful to use the tab key to indent all commands within the loop. This can make the code much easier to read and understand.

If statements

MATLAB uses a similar structure for if statements:

```
>> if i == 1
      DO SOMETHING
elseif i == 2
      DO SOMETHING ELSE
elseif i == 3
      DO SOMETHING ELSE
else
      DO SOMETHING (IF i >= 4 or i <= 0)
end
```

Plotting

To make a line plot of t versus $\sqrt{t}$, simply type:

```
>> plot(t,sqrt(t))
```

To make a plot to t versus $\sqrt{t}$ and t versus t on the same plot, type

```
>> plot(t,sqrt(t),t,t)
```

putting each x-y pair together. You can add symbols or use symbols instead of lines by adding commands to each pair:

```
>> plot(t,sqrt(t),'*-',t,t,'o-')
```

See `help plot` for more information on plotting and a catalog of the available symbols and line types. Use `legend('sqrt(t)','t')` to label the different lines. Title your plot by typing `title('A plot of t versus sqrt(t)')`. You can add labels to the axes similarly: `xlabel('t')` or `ylabel('sqrt(t)')`. Other useful commands: axis, plot3 and (for a bit of fun) comet.

Running `.m` files

You can write scripts of MATLAB functions which may be run as a whole. Save the files with a `.m` extension so that MATLAB will recognize it as a script. Type

```
>> junk
```

to run the script in the file `junk.m`. DO NOT USE THE NAME OF AN INTRINSIC MATLAB FUNCTION AS THE TITLE OF YOUR SCRIPT.

Creating your own functions

You can create your own functions to use in MATLAB. There are two options for this. If your function is simple (i.e. $f(x) = x^2 + 2x$), then you may enter it using the command `inline`:

```
>> f = inline('x.^2 + 2*x')

f =

     Inline function:
     f(x) = x.^2 + 2*x
```

We can then apply this function to a scalar or vector since we used the `.∧2` command for computing the square of x:

```
>> f([0:5])

ans =

     0     3     8    15    24    35
```

If the function is more complicated, you may create a file whose name ends in `.m` which tells MATLAB how to compute the function. Type `help function` to see the format of the function file. Our function here could be computed using the following file (called `f.m`):

```
function [output] = f(x),
output = x.^2 + 2*x
```

This function is called from MATLAB in the same way as above, i.e. `f(x)`, where `x` can be a scalar or vector.

Importing data

Use the `load` command to bring data into matlab from an external file:

```
>> load filename
```

Clearing variables

You may clear a particular variable by typing

```
>> clear x
```

or all variables with

```
>> clear all
```

Formatting MATLAB output

By default, MATLAB outputs numbers with four digits after the decimal point. If one of the numbers is very large or all of them are very small, MATLAB uses scientific notation. However, the exponent is written only once at the beginning of the output, so be careful. For example:

```
>> [1 2 6 24 120  factorial(20)]

ans =

   1.0e+18 *

    0.0000    0.0000    0.0000    0.0000    0.0000    2.4329
```

To control the format of the output, type

```
>> format long e
```

for scientific notation with 15 digits. Other options include `format short e`, `format short`, `format long`, `format short g`. The default is `format short`.

Getting help

Type `help` followed by the name of the function. For example:

```
>> help plot
```

To get more help, try `helpwin`, `helpdesk`, `demo` or `tour`. Also, there is help available on the Math Works website at `http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml`.