An Introduction to MATLAB

1 What is MATLAB?

MATLAB is:

- a calculator;
- a powerful graphics tool; and
- a high–level programming language including a large number of mathematical tools.

2 Preliminaries

Set up a working directory for this course by opening a shell under Linux and typing

```
mkdir matlab_course
```

followed by

```
cd matlab_course
```

to switch to this directory. To list all files within the directory type

```
ls
```

Nothing will be listed since the directory is still empty. You may wish to create some appropriately named subdirectories to store separately the files from each of the practical sessions and the assignment.

3 Launching MATLAB

From within the course directory you just created type

```
matlab &
```

in a command window. A MATLAB window should open. Any files created and saved will now be saved within the current directory.

4 Using MATLAB as a calculator

MATLAB uses the standard computer symbols for addition ("+"), subtraction ("-"), multiplication ("*") and division ("/"). Brackets are also used (if required) to indicate which operations are performed first. For example to evaluate $3 + 4(1 + \frac{6}{3})$ type, in the Command Window,

```
3+4*(1+6/3)
```

and you get the answer 15.
Suppose we wish to set the variable x equal to $3+4(1+\frac{6}{3})$. Then we would type

```
x=3+4*(1+6/3)
```

and you get the answer $x = 15$. In long calculations it is often unneccesary for the value of x to be output to the screen. If we wanted to calculate x but not output the answer to the screen we would end the line with a semicolon

```
x=3+4*(1+6/3);
```

The value of x may now, if desired, be printed out by typing

```
x
```

Finally, to evaluate a power we use the "ˆ" symbol. For example, to evaluate 3^4 type

```
3^4
```

5 Vectors and matrices

5.1 Vectors

Suppose the vector $\mathbf{u}$ is given by $\mathbf{u} = (1, 4, 9)$. This may be entered into MATLAB in either of the following ways.

1. By entering each entry of the matrix individually.

   ```
   u(1)=1;
   u(2)=4;
   u(3)=9;
   ```

   Note that `u(1)` corresponds to the first entry of $\mathbf{u}$ etc.

2. By typing the entries inside square brackets.

   ```
   u=[1 4 9];
   ```

The transpose of $\mathbf{u}$ may be printed by typing

```
u'
```

Suppose the vector $\mathbf{v}$ is given by $\mathbf{v} = \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix}$. This may be entered by typing

```
v=[4; 5; 6];
```

Inside the square brackets the semicolon denotes a new line. Note that vectors are also called arrays.

5.2 Matrices

Suppose the matrix A is given by $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$. This may be entered into MATLAB in either of the following ways.

1. By entering each entry of the matrix individually.

   ```
   A(1,1)=1;
   A(1,2)=2;
   A(2,1)=3;
   A(2,2)=4;
   ```

   Note that `A(i,j)` corresponds to the entry in the ith row and jth column of the matrix A.

2. By typing

   ```
   A=[1 2; 3 4];
   ```

   Note the use of ";" to denote a new row in the matrix.

5.3 Editing vectors and matrices

Can be done by specifying the individual element: e.g. to change the value of A_{12} in the matrix A above to 5 we type

```
a(1,2)=5;
```

An alternative way is to double click on "a" in the Workspace window. This opens a window containing the contents of the matrix A. Click on the entry in the first row, second column, and change it to "5".

5.4 Looking at parts of matrices

The matrix $C = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$ may be entered in MATLAB as

```
C=[1 2 3 4; 5 6 7 8; 9 10 11 12];
```

To view the whole matrix we would type

```
C
```

To view the second column we would type

```
C(:,2)
```

To view the third row we would type

```
C(3,:)
```

To view the first two rows we would type

```
C(1:2,:)
```

To view the elements in both the last two rows and the last three columns we would type

```
C(2:3,2:4)
```

6 Addition, subtraction and multiplication of vectors and matrices

If two vectors or matrices are of the correct size, then they may be added, subtracted or multiplied using the standard operators "+", "-" and "*". For example, enter the following vectors and matrices

```
u=[1 2];
v=[3; 4];
A=[1 2; 3 4];
B=[5 6; 7 8];
```

then the following are valid operations

```
uv
vu
A+B
A-B
A*B
Av
uA
(A+B)*v
```

but

```
Au
```

is not. Why?

7 Generating vectors using ":"

- A vector x consisting of integers between 1 and 10 inclusive may be generated by typing

  ```
  x=1:10
  ```

- A vector y consisting of 100 points equally spaced between 5 and 15 may be generated by typing

  ```
  y=linspace(5,15,100)
  ```

8 Some special matrices and vectors

MATLAB has commands to create some simple matrices

- The 3×4 matrix with all entries equal to 0 may be produced using the command

  ```
  zeros(3,4)
  ```

- The 5×2 matrix with all entries equal to 1 may be produced using the command

  ```
  ones(5,2)
  ```

- The 4×6 matrix with diagonal entries equal to 1 and all other entries equal to 0 is given by

  ```
  eye(4,6)
  ```

- The 2×1 matrix (i.e. column vector of length 2) with entries randomly distributed with a uniform distribution on the interval [0,1] is given by

  ```
  rand(2,1)
  ```

- The 1×3 matrix (i.e. row vector of length 3) with entries uniformly distributed with mean 0 and standard deviation 1 is given by

  ```
  randn(1,3)
  ```

- If the matrix required with any of the commands in this section is square, then the second index is not required. E.g.

  ```
  ones(4)
  rand(2)
  ```

9 Complex numbers

Complex numbers may be entered as, for example,

```
z=3+4i
```

The real part, imaginary part, magnitude, argument and complex conjugate of z may be evaluated using the following commands

```
real(z)
imag(z)
abs(z)
angle(z)
conj(z)
```

10 Simple M-files

Instead of typing the commands into the Command Window, they may be typed into a file with a `.m` extension. This allows the build up of commands into a high–level computer program.

10.1 Opening a new M-file

Click on “File” on the toolbar on the Matlab window. Then click on “New”, then on “M-file”. This will open a new window that is a text–editor.

10.2 Writing an M-file

In the text editor type the following lines of code (using the “Return” or “Enter” key to start a new line)

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

Click on the “save” symbol on the toolbar (the symbol that looks like a floppy disc) and save the file as `mymfile.m`

10.3 Running the M-file

Return to the Command Window and type

```
mymfile
```

This has the same output as if you had entered the three lines of the M-file in at the Command Window.

11 Loops

11.1 `For` loops

If a collection of statements need to be executed a fixed number of times, a `for` loop may be used. For example the following commands print out the integers between 1 and 10 inclusive:

```
for i=1:10
i
end
```

11.2 `While` loops

If a collection of statements need to be executed until a stopping criteria is reached a `while` loop may be used. For example the following commands print out the integers between 1 and 10 inclusive:

```
i=1;
while ( i <= 10 )
i
i=i+1;
end
```

Operators that may be useful when using `while` loops are

`==`	equal to
`∼=`	not equal to
`<`	less than
`>`	greater than
`<=`	less than or equal to
`>=`	greater than or equal to

12 Simple Graphics

To plot the function sin(x) for x between 0 and 10 type

```
fplot(’sin(x)’,[0 10])
```

To label the axes and add a title type

```
xlabel(’x’)
ylabel(’y’)
title(’y=sin(x)’)
```

To plot two data sets y_1 and y_2 against x on the same diagram use the method shown below.

```
x=[1 2 3 4 5 6];
y1=[1 4 9 16 25 36];
y2=[6 5 4 3 2 1];
plot(x,y1,’-’,x,y2,’--’)
xlabel(’x’)
ylabel(’y’)
legend(’y1’,’y2’)
```

The statement `plot(x,y1,’-’,x,y2,’--’)` the tells Matlab to plot y_1 against x as a solid line, and to plot y_2 against x as a broken line. Note the use of `legend` to label the lines. This is only a minute subset of the graphics tools available in Matlab - refer to a manual for others.

13 The Help Command

The help facility within matlab is very extensive. The only real drwaback is that you need to know not only that something is there but also what it is called. Fortunately, most of the Matlab commands have sensible names. Try typing the following:

```
help plot
help +
help odefile
```