Tutorial 1: Introduction to Matlab

Matlab, *MAT*rix *LAB*oratory, is a powerful, high-level language and a technical computing environment, which provides core mathematics and advanced graphical tools for data analysis, visualisation, and algorithm and application development. It is intuitive and easy to use. Many common functions have already been programmed in the main program or on one of the many toolboxes.

In MATLAB everything is a matrix, (this is the mantra: *everything is a matrix*) and therefore it is a bit different from other programming languages, such as C or JAVA. Matrix operations are programmed so that element-wise operations or linear combinations are more efficient than loops over the array elements. The *for* instruction is recommended only as a last resource.

Once you are in MATLAB, many UNIX commands can be used: *pwd, cd, ls,* etc. To get help over any command you can type:

```
>> help command
```

For example try:

```
>> help for
>> help sum
>> help fft
>> help help
```

To create a matrix you can type its values directly:

```
>> x = [ 1 2 3 4 5 6 7 8 9 10 ];
```

Which is equivalent to:

```
 >> x = 1:10;
```

where only initial and final values are specified. It is possible to define the initial and final value and increment (lower limit: increment: upper limit) using the colon operator in the following way:

```
>> z = 0 : 0.1 :20;
```

Both are 1 x 10 matrices. Note that this would be different from:

```
>> y = [1;2;3;4;5;6;7;8;9;10];
```

or

```
>> y=[1:10]';
```

Both are 10 x 1 matrices. The product x*y would yield the inner product of the vectors, a single value, y*x would yield the outer product, a 10 x 10 matrix, while the products x*x and y*y are not valid because the matrix dimensions do not agree. If element-to-element operations are desired then a dot "." before the operator can be used, e. g. x.*x would multiply the elements of the vectors:

```
>> x*y
ans =   385

>> x*x
??? Error using ==> *
Inner matrix dimensions must agree.

>> y*x
ans =
1       2      3      4      5      6      7      8      9      10
2       4      6      8      10     12     14     16     18     20
3       6      9      12     15     18     21     24     27     30
4       8      12     16     20     24     28     32     36     40
5       10     15     20     25     30     35     40     45     50
6       12     18     24     30     36     42     48     54     60
7       14     21     28     35     42     49     56     63     70
8       16     24     32     40     48     56     64     72     80
9       18     27     36     45     54     63     72     81     90
10      20     30     40     50     60     70     80     90     100
>> x.*x
ans =    1      4      9     16     25     36     49     64     81    100
```

The Matrix:

$$mat = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix}$$

can be obtained by typing:

```
>> mat = [1 2 3 4;2 3 4 5;3 4 5 6;4 5 6 7];
```

The final semicolon (;) inhibits the echo to the screen. Any individual value of the matrix can be read by typing (without the semicolon):

```
>> mat (2,2)
ans = 3
```

Mathematical functions can be used over the defined matrices, for example:

```
>> s1 = sin (z);
```

A column or line of a matrix can be obtained from another one:

```
>> s2(1,:) = -s1/2;
```

```
    >> s2(2,:) = s1;
    >> s2(3,:) = s1 * 4;
```

To display a 1D matrix you can use *plot*, and for 2D you can use *mesh*, Figure 1 shows the result of typing:

```
    >> plot(s1);
    >> mesh(s2);
```

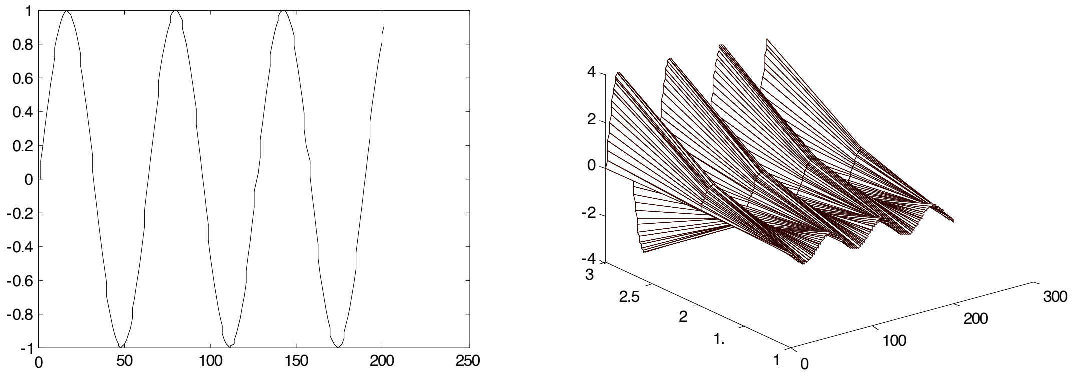

Figure 1

1.1 Use *help* for the following functions.

sign	subplot	abs	imshow	surf	colormap
sum	cumsum	fix	round	subplot	whos
title	for	who	sqrt	conv	floor
det	fft	abs	semilogx	axes	axis
zeros	ones	rand	randn	pi	real

1.2 There are many toolboxes with specialised functions, try:

```
    >> help images      Image processing toolbox.
    >> help signal      Signal processing toolbox.
    >> help stats       Statistics toolbox
    >> help nnet        Neural networks toolbox
```

1.3 To find out which toolboxes you have installed type ver:

```
    >> ver
```