General To start make a new directory named, say, matlab, enter that directory in, at least, two different windows. You will use one window to run `Matlab` , and the other in order to edit your .m files (see below). In the first window type

`matlab`

You are running your main program (which you quit with `quit`) . `Matlab` is interactive: whenever you hit the `return` key, `Matlab` will display all variables that either have been defined, or have been altered, since the last `return`. Furthermore, `Matlab` will show you all intermediate values of these variables, if several changes had occured in their values.

Commands are separated by either a comma, or a `return` (or both). Using comma allows you to put several commands on the same line, and also to postpone the printing of output. Semi-colon is the same as comma, but supresses the printing. Try to type

`A=1, B=2`

and then

`A=3; B=2;`

You do not need, and in most cases, you do not want, to type your entire program on-line: `Matlab` allows you to use subroutines. All subroutines are called `.m files` . These are files that end with the suffix `.m` . They must be located either in your working directory, or be a part of the package library. For example, in the other window, open a file named `shalom1.m` and type there

`A=[1 2 3] ; B=2*A, C=A.^2`

(and write the file, of course). Then, in the main program window, type

`shalom1`

`Matlab` will try first to find a *variable* named `shalom1` . If it finds such, it will print its value; otherwise, it will look for an .m-file named `shalom1.m` and will execute all its lines. If you are not pleased with the output, you may edit the .m-file in the other window, and re-execute it by retyping its name in the main window. So, big chunks of your program can be stored is such .m-files, which are called *script files*. The variables in the script-files are identical to those of the main program. Thus, if you type now

`A`

you will get the vector A printed on the screen. You can call one .m-file from another file. script files are particularly useful for input data.

There is another type of .m-files called *function files* They will be explained later on. Had I wanted `shalom1.m` to be a function .m-file, I would have had to have its first line look like, for example,

`function val=shalom1(x,y,z)`

Thus, all .m files that do not start with the word `function`, are treated as script files.

Input

Each variable in `Matlab` is either a string of characters, or a matrix. Thus,

`A=11`

declares A to a 1×1 matrix whose single entry equals 11. It is the same as

`A=[11]`

A row vector is entered as follows

`B=[7 9.2 -36 78]`

and a column vector can be either entered as

`C=[7; 9.2; -36; 78]`

or

`C=[7 9.2 -36 78]'`

(since `M'` is the transpose of `M`), or

```
C=[7
    9.2
    -36
    78]
```

Here is an example where I input a 3×3 matrix D:

```
D=[12.1 0 7;12 1+2i 7; 12 13 14]
```

Variables in `Matlab` need not be declared. Thus, the fact that D is a
3×2
matrix was understood by `Matlab` when you entered that matrix. If you want now to multiply the matrix D by its transpose D', type

```
E=D*D'
```

However, if you want the product to be computed *pointwise* and not as matrix multiplication, type

```
F=D.*D'
```

if you want to generate a large random matrix R for experimentations, simply type

```
R=rand(m,n)
```

with m, n the number of rows, columns that you desire. Entries in matrices are referred to as in $R(4,3)$. If you type

```
R(10,3)=13
```

and there are less than 10 rows and/or less than 3 columns in R, the matrix R will be expanded, with all new entries are 0. You may, then, define a new, 4×3, zero matrix N, by typing

```
N(4,3)=0
```

(there are other ways to generate a zero matrix; see the srcipt).
There are easy ways to generate vectors:

```
T=[5:12]
```

or

```
TT=[12:-.5:  -1]
```

and then, if you want to evaluate some function at all points of TT, type

```
Cos=cos(TT)
```

Plotting

Suppose that we want to plot

$$x^3 - 2x^2 - 2x + 6$$

on the interval $[-2, 4]$. First,

```
X=[-2:.1:4];
```

generates the points on the X-axis. Then,

```
Y=X.^3-2*X.^2-2*X+6;
```

generates the Y values. Then

```
plot(X,Y)
```

generates the graph.