To start type ’matlab’ under unix (or click icon under windows). You will get the matlab logo and then the prompt:

>>

In version 6 a new command window will pop-out, in which you can type commands. You can avoid this [matlab v-6 is still a little buggy] by starting matlab with matlab -nojvm instead of matlab.

Basic Operations in Matlab

The following demo is on the basics of matlab. it starts with some basic operations and the help command. The file below is obtained by typing

>>diary filename.

This is equivalent to a typescript. Everything that starts with >> is what I typed into the matlab prompt.

Simple operations

>> 4+6+3
ans =
   13
>> 4*20+3*57 + exp(-0.1)
ans =
  251.9048

Getting Help

Most of the help for matlab is online. You can get help by typing

’>> help’ by itself will list the help topics

To start a help window you can click on the help icon or type

under matlab version 5.3 (or under -nojvm).
exp(x) is the exponential of the elements of x, e to the x.
for complex z=x+i*y, exp(z) = exp(x)*(cos(y)+i*sin(y)).
see also log, log10, expm, expint.
overloaded methods
help sym/exp.m

>> help

help topics:

saad/matlab       - (no table of contents file)
matlab/general    - general purpose commands.
matlab/ops        - operators and special characters.
matlab/lang       - programming language constructs.
matlab/elm        - elementary matrices and matrix manipulation.
matlab/elfun      - elementary math functions.
matlab/specfun    - specialized math functions.

>> more on

'more on' allows you to scroll page by page

>> help elfun

elementary math functions.
trigonometric.
sin - sine.
sinh - hyperbolic sine.
asin - inverse sine.
asinh - inverse hyperbolic sine.
cos - cosine.
cosh - hyperbolic cosine.
acos - inverse cosine.
acosh - inverse hyperbolic cosine.
tan - tangent.
.
.
Example:

>> help mod
mod   modulus (signed remainder after division).
    mod(x,y) is x - y.*floor(x./y) if y ~= 0. by convention,
    mod(x,0) is x. the input x and y must be real arrays of the
    same size, or real scalars. the statement "x and y are
    congruent mod m" means mod(x,m) == mod(y,m).

MOD(x,y) has the same sign as y while REM(x,y) has the same
sign as x. MOD(x,y) and REM(x,y) are equal if x and y have
the same sign, but differ by y if x and y have different signs.

See also REM.

>> a = 25; b = 3;
>> mod(a,b)
ans =
     1
>> mod(a,5)
ans =
     0
>> mod(25.2,2)
ans =
     1.2000

Note: A colon after a command will prevent displaying the result.

>> a
ans =
     25
>> a;

'who' lists the variable currently stored

>> who

your variables are:

a  ans   b

>>
Complex Numbers

```matlab
>> c = 1 - 2i
c = 1.0000 - 2.0000i
>> conj(c)
ans = 1.0000 + 2.0000i
>> c*conj(c)
ans = 5
```

Note: \( \text{abs}(c) \) is the modulus of \( c \)

Matrices

- To define a matrix enter entries row by row, separated by a ";":

```matlab
>> a = [1 1 1; 2 2 2; 3 3 3]
an =
    1 1 1
    2 2 2
    3 3 3
```

- Could use commas for separating columns (not required):

```matlab
>> a = [1, 1, 1; 2, 2, 2; 3, 3, 3];
```

- Can use matrices as blocks [very convenient!]

```matlab
>> b = [a, a]
b =
    1 1 1 1 1 1
    2 2 2 2 2 2
    3 3 3 3 3 3
```

```matlab
>> c = [a, -a; a*a, 2*a]
c =
    1 1 1 1 1 1
    2 2 2 2 2 2
    3 3 3 3 3 3
```
c =

1 1 1 -1 -1 -1
2 2 2 -2 -2 -2
3 3 3 -3 -3 -3
6 6 6 2 2 2
12 12 12 4 4 4
18 18 18 6 6 6

The transpose is defined by the prime sign

>> cp = c'

cp =

1 2 3 6 12 18
1 2 3 6 12 18
1 2 3 6 12 18
-1 -2 -3 2 4 6
-1 -2 -3 2 4 6
-1 -2 -3 2 4 6

Two important special matrix functions

>> a = eye(5,5)
a =

1 0 0 0 0
0 1 0 0 0
0 0 1 0 0
0 0 0 1 0
0 0 0 0 1

'eye' is defined for rectangular matrices too

>> a = zeros(3,4)
a =

0 0 0 0
0 0 0 0
0 0 0 0

To get out use exit or quit

>> quit
Defining a vector through loop constructs

```
>> start=0; inc=2; last=12;
>> start:inc:last
ans =
     0     2     4     6     8    10    12
>> 0:2:12
ans =
     0     2     4     6     8    10    12
```

Can also use real numbers

```
start = 0.0; inc = 0.15; last = 1.0;
>> start:inc:last
ans =
     0.0000    0.1500    0.3000    0.4500    0.6000    0.7500    0.9000
>> x = 0:0.15:1
x =
     0.0000    0.1500    0.3000    0.4500    0.6000    0.7500    0.9000
```

Because you cannot square a rectangular matrix.

Pointwise (array) product:

```
>> a = [2, 3, 4] ; b = [ 0 5 6] ;
>> c = a .* b
  c =
      0    15    24

Let us go back to \( z = x^2 \). To square the components of \( x \), do:

```
>> y = x.^2
y =
     0    0.0225    0.0900    0.2025    0.3600    0.5625    0.8100
or
>> y = x.*x
y =
     0    0.0225    0.0900    0.2025    0.3600    0.5625    0.8100
```

```
The function `size` gives the dimensions of the object \( x \)

```
>> size(x)
ans =
     1    7

```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```

```
```
Basic operators

- **Standard arithmetic operators:**
  
  +, -, *, /

- **Unary operations** (for example $-A$).

- **Back-slash operator:**
  
  $x = A \backslash b$

  where $A$ is a matrix and $b$ a vector (or matrix) then $x = A^{-1}b$.

**Example:**

```matlab
A = [1 2 4; 2 1 3; 0 1 0];
>> b = [1 3 4]';
>> x = A \ b
ans =
   3.4000
   4.0000
  -2.6000
>> A*x
ans =
   25
```

- **Relational operators.**

  - Equal $==$
  - Not equal $~=$
  - Less than $<$
  - Greater than $>$
  - Less than or equal $<=$
  - Greater than or equal $>=$

**Example:**

```matlab
>> a = 1; b = 0; c = 2;
>> a+b+c == c+a+b
ans =
   1
>> a = 1:2:20
26
a =
   1  3  5  7  9 11 13 15 17 19
>> b = 2:2:21
b =
   2  4  6  8 10 12 14 16 18 20
>> a == b
ans =
   0  0  0  0  0  0  0  0  0  0
>> a+1 == b
ans =
   1  1  1  1  1  1  1  1  1  1

Note: 1 means “true”, 0 means “false”
```