Tutorial 0: Some Matlab basics

In this tutorial you will learn some basic Matlab commands, such as how to enter matrices and vectors.

Entering and multiplying matrices and vectors

You enter a matrix by using square brackets and then just type the rows:

```
A = [1 2 3
     4 5 6];
```

Adding the semicolon (;) at the end of the line supresses the output.

```
A = [1 2 3
     4 5 6]
```

Vectors can also be entered in this way:

```
x = [1
     2
     3];
```

You can multiply matrices and vectors by using the *

```
 A*x
```

If you want to multiply two matrices that are incompatible, then Matlab will give you an error.

```
A = [1 2 3
     4 5 6];
x = [1
     2];
A*x
```

Accessing elements of vectors or matrices

You can access an element of a matrix as follows. If you want the 3rd element in the 2nd row of **A** you write:

```
A(2,3)
```

If you want the 1st element in the 1st row of **A** you write:

```
A(1,1)
```

If you want to access a row of **A**, you use the ":" To get the first or second row of **A**, write:

```
A(1,:) % 1st row
A(2,:) % 2nd row
```

Here I used "comments," which start with the % sign. Comments can contain words, explanations or code, but the comments are not executed as part of the code.

If you want to access a column of **A**, you also use the ":" To get the first, second or or column of **A**, write:

```
A(:,1)
A(:,2)
```

Plotting graphs

Matlab can also plot functions. For example we can plot the function f(x) = sin(x) as follows. First we define a vector x where we want to evaluate the sin function.

```
x = [0 pi/2 pi 3*pi/2 2*pi];
```

Now evaluate the sin function at these x:

```
f = sin(x);
```

Plot the result:

```
plot(x,f)
```

This does not look much like the sin function you know because we only evaluated sin(x) at 5 different locations. We can evaluate sin(x) at 0, 0.1, 0.2, 0.3, ... 2 pi, by using the following commands:

```
x = 0:0.1:2*pi;
```

This generates a vector x that has elements that are between 0 and 2pi in steps of 0.1. We can evaluate the sin function at these x:

```
f = sin(x);
```

And plot the result:

```
plot(x,f)
```

We can also plot the function g(x) = cos(x):

```
g = cos(x);
hold on, plot(x,g)
```

We can also plot just a bunch of points.

```
x = 1:100;
y = randn(100,1);
figure
plot(x,y,'.','MarkerSize',20)
```

Here, x are the numbers 1 through 100 in increments of 1 (x = [1 2 3 4 5 .... 100]) and y are 100 randomly generated numbers.

For loops

You can also implement for loops in Matlab. For example, you can have Matlab count to ten in a for loop.

```
for kk=1:10
    fprintf('%g\n',kk)
end
fprintf('Done.')
```

Here, 1:10 is a vector whose elements are the integers from 1 to 10.

When you specify a for loop, you can tell Matlab to perform a sequence of commands many times. For example, you can multiply a given number by 3:

```
n = 2;
for kk=1:10
    n = n*3; % this replaces the number n by 3*n
end
n
```

This is of course the same as multiplying n by raising 3 to the power 10:

```
2*3^10
```

For loops are useful to implement recursions, which you will learn about in subsequent tutorials.