Variables, Vectors, Arithmetic

At a basic level, Octave is a fancy calculator. Here's a snippet of a session in Octave:

```
octave-3.2.3:1> x=3;
octave-3.2.3:2> y=5;
octave-3.2.3:3> z=x+y
z =  8
```

In this session, we made three variables: x, y, and z. We assigned the values of x and y directly. The variable z is new and contains the sum of the values of x and y.

Octave knows about vectors as well as plain numbers. Here's how you make vectors.

```
octave-3.2.3:1> x=[ 3, 1, 7]
x =

   3   1   7
octave-3.2.3:2> y=[2 ; 9]
y =

   2
   9
```

Notice that x is a row vector; its entries are listed horizontally. On the other hand, y is a column vector. Its entries are listed vertically. The only difference when you enter them is the comma or semicolon used to separate the entries.

If two vectors have the same number of entries and are both column or both row vectors, you can add them together.

```
octave-3.2.3:3> x=[3;1;7]
x =

   3
   1
   7
octave-3.2.3:4> y=[2;9;8]
y =

   2
   9
   8

octave-3.2.3:5> z=x+y
z =
    5
   10
   15
```

The first entry of z is 5 because $3 + 2 = 5$. The second entry of z is 10 because $1 + 9 = 10$. And the third entry of z is 15 because $7 + 8 = 15$.

We can access the entries of z as follows

```
octave-3.2.3:6> z(1)
ans =  5
octave-3.2.3:7> z(2)
ans =  10
octave-3.2.3:8> z(3)
ans =  15
```

You can things with the new vector z just like you did with z and y. For example

```
octave-3.2.3:7> z+z
ans =

    10
    20
    30
```

The result of this computation is stored in a temporary variable called ans. You could also have assigned z+z to a new variable.

```
octave-3.2.3:8> w=z+z
w =

    10
    20
    30
```

You can also multiply a vector by a number.

```
octave-3.2.3:31> x=[3,1,7]
x =

   3   1   7

octave-3.2.3:32> z=4*x
z =

   12    4   28
```

Multiplying two vectors is a little more tricky, and we need to start by discussing matrices. Matrices are like vectors, except they are two-dimensional. Here's how you enter a matrix.

```
octave-3.2.3:6> A = [ 4, 9, 8; 1, -3, 5]
A =

   4   9   8
   1  -3   5
```

We say that `A` is a 2×3 matrix because it has two rows and three columns. You separate entries of a row with commas, and you separate rows with semicolons. Vectors are special cases of matrices: a row vector with 5 entries is a 1×5 matrix, and a column vector with 5 entries is a 5×1 matrix.

There is a notion of matrix multiplication, but it might be different than you might have guessed at first. You can multiply a $n \times m$ matrix by a $m \times k$ matrix, and the result will be a $n \times k$ matrix.

Consider the following session:

```
octave-3.2.3:10> x = [1, 5, 2]
x =

   1   5   2

octave-3.2.3:12> y = [3, 4, 9]
y =

   3   4   9

octave-3.2.3:13> z = [3; 4; 9]
z =

   3
   4
   9

octave-3.2.3:14> x*y
error: operator *: nonconformant arguments (op1 is 1x3, op2 is 1x3)
octave-3.2.3:14> x*z
ans =  41
```

The first multiplication gives an error because you can't multiply a 1×3 matrix by a 1×3 matrix. The second multiplication works because `x` is a 1×3 matrix and `y` is a 3×1 matrix. The result is a 1×1 matrix, which in Octave is the same thing as a plain old number. Notice that the answer, 41, is the dot product of `x` and `y`. Is it legal to multiply `z*x`? What size of a matrix will you get? We'll talk more about matrix multiplication later in the class.

Often you want to multiply the entries of two vectors together term by term. Octave has a special command for this: `.*` (i.e. dot-times). For example:

```
octave-3.2.3:16> x = [2, 5, 2]
x =

   2   5   2

octave-3.2.3:17> y = [3, 4, 9]
y =

   3   4   9

octave-3.2.3:18> x.*y
ans =

    6   20   18
```

The first entry of `ans` is 6 since the product of 2 and 3 is 6.

You can also divide the entries of two vectors term by term with `./` (i.e dot-divide).

```
octave-3.2.3:22> x./y
ans =

   0.66667   1.25000   0.22222
```

And you can raise each entry of a vector to a power with `.^` (i.e. dot-power).

```
octave-3.2.3:23> x.^2
ans =

    4   25    4
```

Basic Plotting

Here's a handy way to make a large vector with equally spaced entries:

```
octave-3.2.3:24> x=[0:0.1:1]
x =

 Columns 1 through 8:

   0.00000    0.10000    0.20000    0.30000    0.40000    0.50000    0.60000    0.70000
 Columns 9 through 11:

   0.80000   0.90000   1.00000
```

The command `[0:0.1:1]` makes a row vector that starts at 0 and increases each entry by 0.1, until it reaches 1.

For very large vectors, you might not want to see the output. For example, if you enter `x = [0: 0.01 :1]` then x will have 101 entries. If you end a line with a semi-colon, you won't see the output. For example:

```
octave-3.2.3:37> x=[0:0.05:1];
octave-3.2.3:38> x
x =

 Columns 1 through 8:

   0.00000   0.05000   0.10000   0.15000   0.20000   0.25000   0.30000   0.35000

 Columns 9 through 16:

   0.40000   0.45000   0.50000   0.55000   0.60000   0.65000   0.70000   0.75000

 Columns 17 through 21:

   0.80000   0.85000   0.90000   0.95000   1.00000
```

We can also use the `linspace` command:

```
octave-3.4.0:1> x=linspace(0,pi,10)
x =

 Columns 1 through 8:

   0.00000   0.34907   0.69813   1.04720   1.39626   1.74533   2.09440   2.44346

 Columns 9 and 10:

   2.79253   3.14159
```

This gives us a row vector of length 10, starting at 0, ending at π, with entries that are equally spaced.

Suppose we want to plot the polynomial $1 + 2 * x^2 - 3x^3$ over the interval $[-1, 1]$. As a first step, we make a vector for x values:

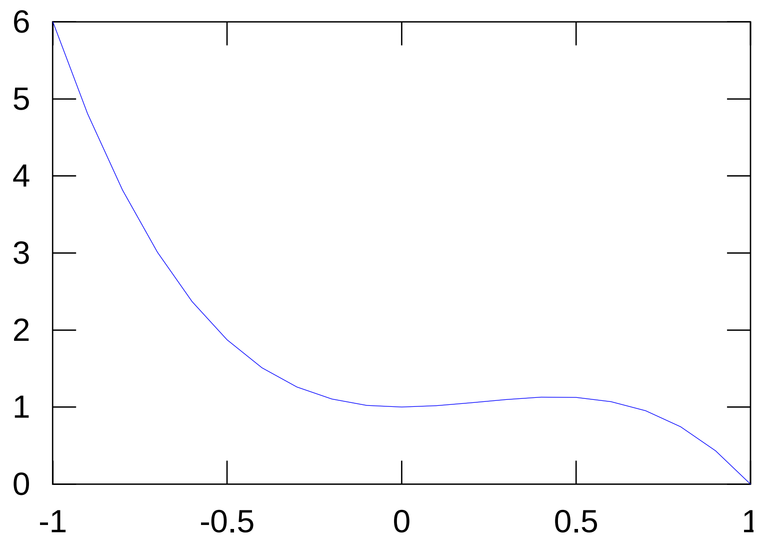
```
octave-3.2.3:39> x=[-1:0.1:1];
```

Then we make a vector for the y values:

```
octave-3.2.3:40> y = 1 + 2*x.^2 - 3*x.^3;
```

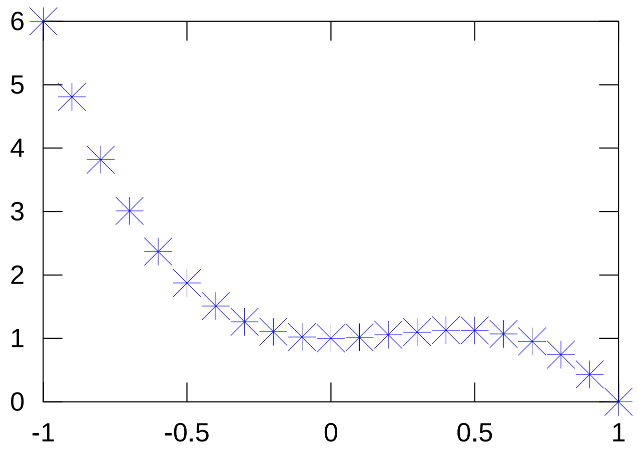
Finally, we plot these x and y coordinate points with

```
octave-3.2.3:41> plot(x,y)
```



This may look like a continuous curve, but Octave has plotted each x and y coordinate and connected them with straight lines. You can see the individual points that were plotted using

```
octave-3.2.3:41> plot(x,y,'*')
```



Exercises

In the subsequent exercises, $x = [7, 1, 8, 2]$ and $y = [4, 9, 8, -11]$. These exercises are for your benefit. **DO NOT HAND THESE IN!**

Exercise 1:

Enter all of the commands in this tutorial into Octave. Really!

Exercise 2:

Use Octave to compute $x + y$.

Exercise 3:

Use Octave to compute $-3 * x$.

Exercise 4:

Use Octave to compute a vector that contains the square root of all the entries of x. Hint: square root is the $1/2$ power.

Exercise 5:

Let $p(t) = -1 + 3t - 2t^3$ – that is, p is a polynomial. Use Octave to compute the value of p at each of the entries of x. The first entry of this matrix should be $p(7)$ since the first entry of x is 7. The last entry should be $p(2)$ since 2 is the last entry of x.

Exercise 6:

Use Octave to plot $p(t) = -1 + 3t - 2t^3$ over the interval $0 \leq t \leq 2$. Hand in a printout of your graph.