The Matlab Environment:

Matlab has three display windows that we will be using:
- a command window for entering commands and data
- an edit window for creating and modifying programs (m-files)
- a figure window for displaying linear plots and surfaces as well as other graphical representations of data.

Entering Data into Matlab:

There are three methods that we will use to initialize data in Matlab: manually listing the data, using the colon operator, and using built-in Matlab functions.

Manually Initializing Variables

To manually enter data variables, at the Matlab command prompt, you type a variable name, an equal sign, and then some value or string of values enclosed in square brackets, depending on structure of data you are attempting to enter. A variable name may contain both letters and numbers, but it must begin with a letter and be no more than 16 characters long. To enter a scalar variable, simply type the value after the equal sign. To initialize an array or vector, type the elements of the array separated by either a comma or space and enclosed in square brackets. To enter a matrix, type the elements of each row separated by either a comma or space using a semicolon to designate the beginning of each new row all, again, enclosed in square brackets. Examples of initializing these three different types of data structures are shown below:

```
EDU» A=25
A =
    25
EDU» B=[1 2 3 4]
B =
    1    2    3    4
EDU» C=[1 2 3 4; 5 6 7 8; 9 10 11 12]
C =
    1    2    3    4
    5    6    7    8
    9   10   11   12

You can also use variables that you have already defined to initialize new variables. Consider the examples below using the array "B" that we defined above:

EDU» D=[-1 0 B]
D =
    -1     0    1    2    3    4
```
You can also modify or add data to existing variables. The examples below use the matrix "E" that was previously defined to illustrate how to do this:

EDU» E(1,1)=10
E =
    10     2     3     4
    5     6     7     8
EDU» E(2,3)=10
E =
    10     2     3     4
    5    10     8
EDU» E(3,4)=10
E =
    10     2     3     4
    5    10     8
    0    0    0    10

Notice that if you define a value that is outside of the range of the original variable, as in the third example, all values not defined are set to zero.

The Colon Operator

The colon operator is a very powerful tool for creating new vectors and matrices. The colon operator can be used to create a vector from a previously initialized matrix.

When a colon is used in the place of a numeric subscript in a matrix reference, the colon represents all elements of the particular row or column. Consider the following examples:

EDU» F=[1 2 3; 4 5 6; 7 8 9]
F =
    1     2     3
    4     5     6
    7     8     9
EDU» column1=F(:,1)
column1 =
    1
    4
    7
EDU» row2=F(2,:) 
row2 =
    4     5     6

Notice that if the colon operator is used to replace the row index in a matrix reference, then the colon represents every row for the given column and the result is a column vector. Conversely, if
the colon operator is used to replace the column index in a matrix reference, then the colon
represents every column for the given row and the result is a row vector.

When two numbers separated by a colon is used in the place of a subscript reference to a
matrix, the index represents all rows or columns beginning with the first number and ending with
the second number. The following examples best illustrate this use of the colon operator:

EDU» G=[1 2 3 4; 5 6 7 8; 9 10 11 12]
G =
1   2   3   4
5   6   7   8
9  10  11  12

EDU» H=G(2:3,2)
H =
6 
10

EDU» H=G(3,1:3)
H =
  9  10  11

EDU» H=G(2:3,3:4)
H =
   7   8
  11  12

A second, and more valuable, use of the colon operator is in the generation of new
vectors. If a colon is used to separate two integers in a variable initialization, the result is a vector
that contains all of the integers between the two specified integers, including those specified. If a
colon is used to separate three numbers (not necessarily integers), the result is a vector that begins
with the first number and ends with the third number and progresses in increments that are equal
to the second number. The following examples illustrate these uses of the colon operator:

EDU» I=1:6
I =
1   2   3   4   5   6

EDU» J=1:0.5:3
J =
1.0000  1.5000  2.0000  2.5000  3.0000

EDU» K=10:-2:0
K =
 10   8   6   4   2   0

As you can see, you could easily create very long vectors by typing just a few numbers. All you
have to provide is the number to start with, an increment, and the number to end with.
Built-in Matlab Functions

There are three built-in functions in Matlab that are used to create matrices. These functions are zeros(), ones(), and eye(). The arguments for these functions are the same, the difference is the type of matrix that each one creates. The function zeros() creates a matrix of all zeros, ones() creates a matrix of all ones, and eye() creates an identity matrix (a matrix with ones on the main diagonal and zeros elsewhere). Each of these three functions takes two arguments. The first argument is the number of rows the created matrix will have and the second argument is the number of columns the created matrix will have. If only one argument is given, the matrix will be a square matrix with the same number of rows and columns as the argument's value. Consider the following examples of these three functions.

```
EDU» L=zeros(1,5)
L =
      0   0   0   0   0
EDU» M=ones(2,3)
M =
      1   1   1
      1   1   1
EDU» N=eye(4,5)
N =
      1   0   0   0   0
      0   1   0   0   0
      0   0   1   0   0
      0   0   0   1   0
EDU» O=ones(3)
O =
      1   1   1
      1   1   1
      1   1   1
```

Array Operations:

Suppose that you have created two arrays, P and Q, each with 3 elements and you would like to create a third vector, R, whose values are the products of the values in P and Q. Or, in other words, the first element in R equals the first element in P multiplied by the first element in Q, the second element in R equals the second element in P multiplied by the second element in Q, and so on. This is what is called an element-by-element or array operation. Matlab conveniently does element-by-element operations for you. To use Matlab's built-in array operations you place a "." before the normal scalar operator that separates the two vectors. The following examples demonstrate this:

```
EDU» P=1:3
P =
      1   2   3
EDU» Q=4:6
Q =
      4   5   6
```
EDU» R = P.*Q

R =
4  10  18

EDU» R = P./Q

R =
0.2500  0.4000  0.5000

EDU» R = P.^Q

R =
1  32  729

EDU» R = P+Q

R =
5  7  9

EDU» R = P-Q

R =
-3  -3  -3

Note that addition or subtraction of two arrays does not have to be done element-by-element and the "." is not used before the operator.