



# CPSC-440 Computer System Architecture

## MATLAB Review

# Matlab

- Is a numerical computing environment and 4<sup>th</sup> generation programming language
- Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran

# Free Matlab for Students

- Available at CSUF IT website:
  - <http://www.fullerton.edu/it/students/software/matlab/>

# Matlab Default View

The screenshot displays the MATLAB R2013a interface. The top menu bar includes HOME, PLOTS, and APPS. Below it is a ribbon with various toolboxes and functions. The main workspace is divided into three panes:

- Current Folder:** Shows a list of files and folders in the current directory, including AOS\_Level\_Seg, boostingDemo, drtoolbox, Fuzzy Hough Transform, Install Files, LabelMeToolbox, prtools\_ac, rtmc9s12\_CW\_R2009b\_002\_006, Shared Funcs, sltoolbox\_r101, spider, stprtool, ViolaJones\_version0b, and prtools\_ac.zip.
- Command Window:** Contains the following code:

```
>> a = [1 2 3 4 6 4 3 4 5]

a =

     1     2     3     4     6     4     3     4     5

fx >>
```
- Workspace:** Shows a table with the following data:

| Name | Value               | Min |
|------|---------------------|-----|
| a    | [1,2,3,4,6,4,3,4,5] | 1   |
- Command History:** Shows a list of commands executed, including:

```
ind = reshape(ind,9,2);
ind = find(h > 0);
ind = ind - 1;
ind = reshape(ind,9,2)';
ReverbLowest
ReverbLow
clc
ReverbMiddle
ReverbHigh
ReverbHighest
5/2/2013 1:11 PM
5/6/2013 1:27 PM
example_17_14
C2 = A*B;
8/29/2013 11:31 AM
a = [1 2 3 4 6 4 3 4 5]
```

# Command Window

The image displays the MATLAB R2013a software interface. The Command Window is highlighted with a red border and contains the following text:

```
>> a = [1 2 3 4 6 4 3 4 5]

a =

     1     2     3     4     6     4     3     4     5

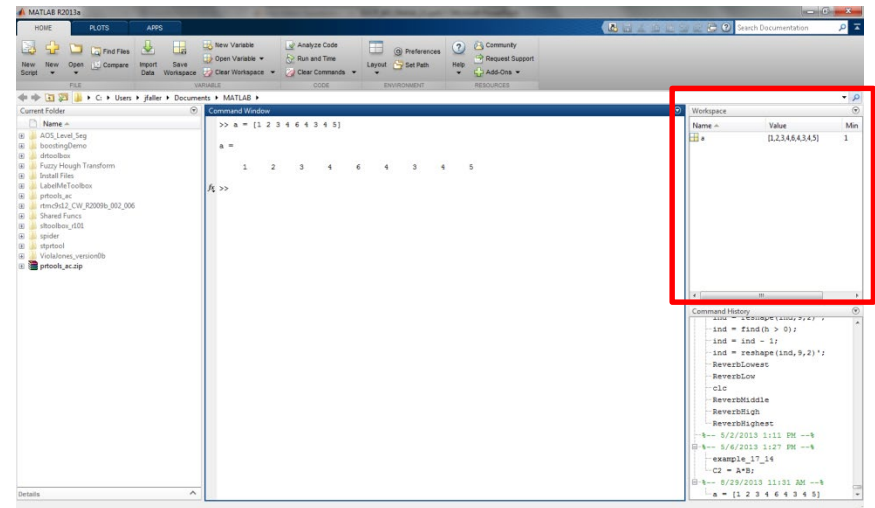
fx >>
```

The Workspace window on the right shows the variable 'a' with the value [1,2,3,4,6,4,3,4,5] and a memory size of 1. The Command History window at the bottom right shows the following commands:

```
ind = reshape(ind,9,2) ;
ind = find(h > 0);
ind = ind - 1;
ind = reshape(ind,9,2)';
ReverbLowest
ReverbLow
clc
ReverbMiddle
ReverbHigh
ReverbHighest
5/2/2013 1:11 PM --
5/6/2013 1:27 PM --
example_17_14
C2 = A*B;
8/29/2013 11:31 AM --
a = [1 2 3 4 6 4 3 4 5]
```

# Workspace Window

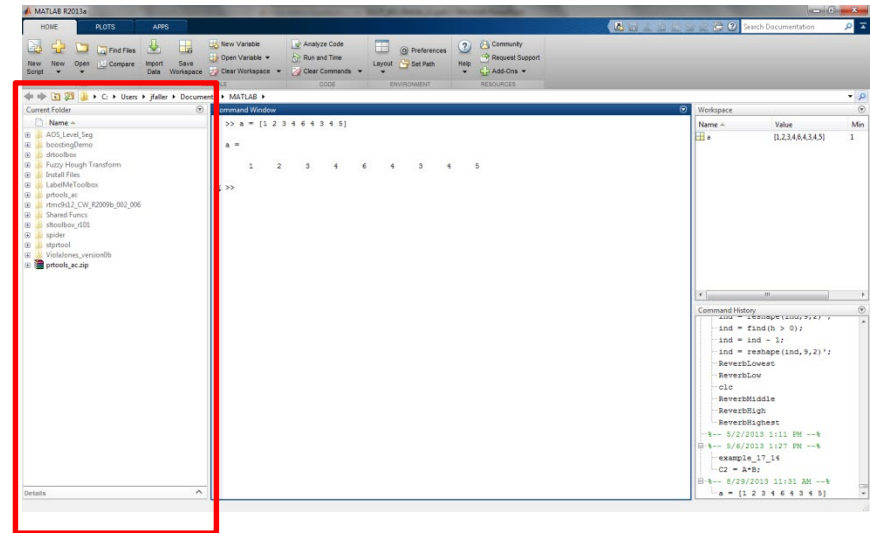
- Shows the variables currently available to you





# Current Folder Window

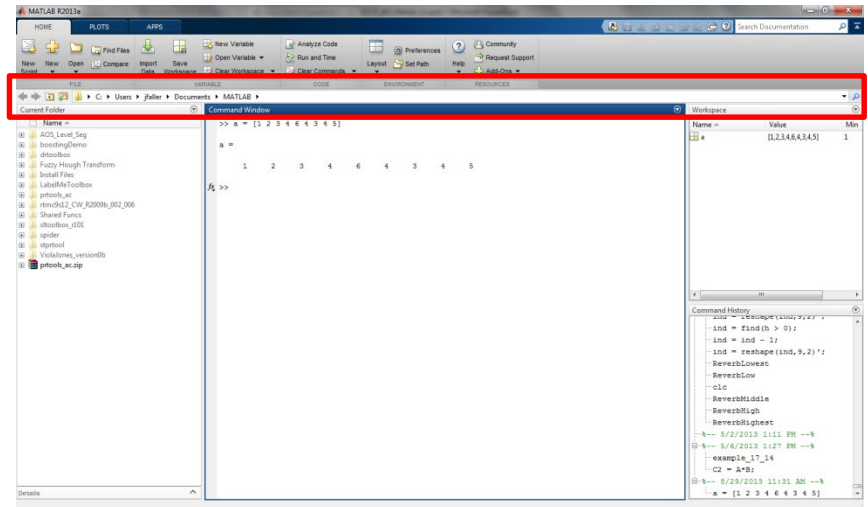
- Shows the folders for the present working directory





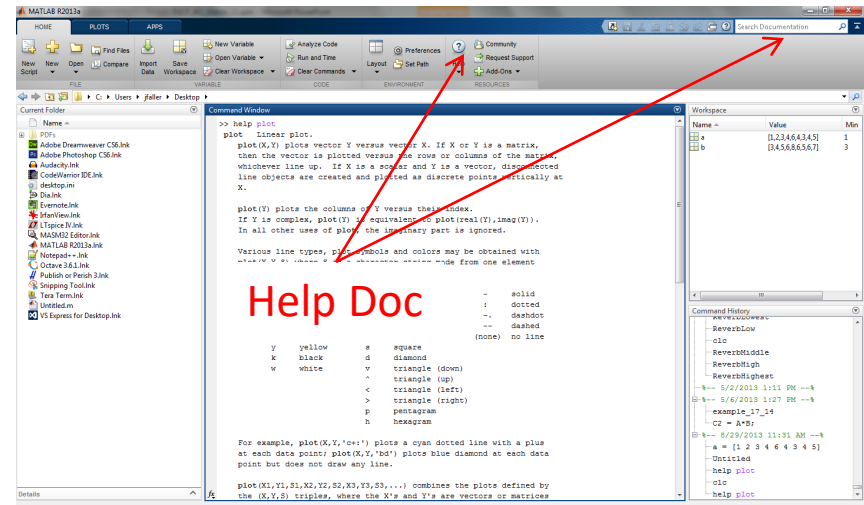
# Present Working Directory

- Shows the current folder you are working in
- You can also use the command “pwd”



# Help Docs

- Searchable help doc
- You can also use the “help” command
- Example:  
`help plot`



# Creating Scripts

The image shows the MATLAB R2013a interface. The 'New Script' button in the HOME tab is highlighted with a red box and an arrow. The Command Window displays the help text for the 'plot' function.

**Launches the script editor**

```
>> help plot
plot Linear plot.

plot(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
then the vector is plotted versus the rows or columns of the matrix,
whichever line up. If X is a scalar and Y is a vector, disconnected
line objects are created and plotted as discrete points vertically at
X.

plot(Y) plots the columns of Y versus their index.

plot(real(Y), imag(Y)).
art is ignored.

various line types, plot symbols and colors may be obtained with
plot(X,Y,S) where S is a character string made from one element
from any or all the following 3 columns:

      b  blue      .   point      -   solid
      g  green     o   circle     :   dotted
      r  red       x   x-mark    -.  dashdot
      c  cyan      +   plus       --  dashed
      m  magenta   *   star       (none) no line
      y  yellow    s   square
      k  black     d   diamond
      w  white     v   triangle (down)
      ^   triangle (up)
      <   triangle (left)
      >   triangle (right)
      p   pentagram
      h   hexagram

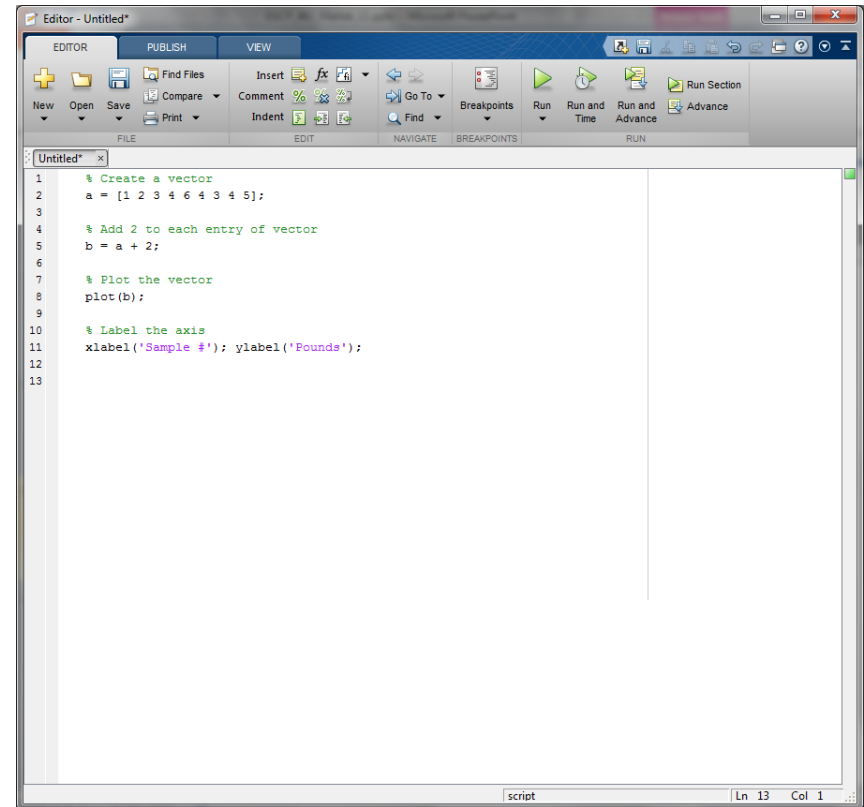
For example, plot(X,Y,'c+!') plots a cyan dotted line with a plus
at each data point; plot(X,Y,'bd') plots blue diamond at each data
point but does not draw any line.

plot(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...) combines the plots defined by
the (X,Y,S) triples, where the X's and Y's are vectors or matrices
```

The Workspace window shows variables 'a' and 'b' with their values and dimensions. The Command History window shows the execution of 'help plot' and 'clc'.

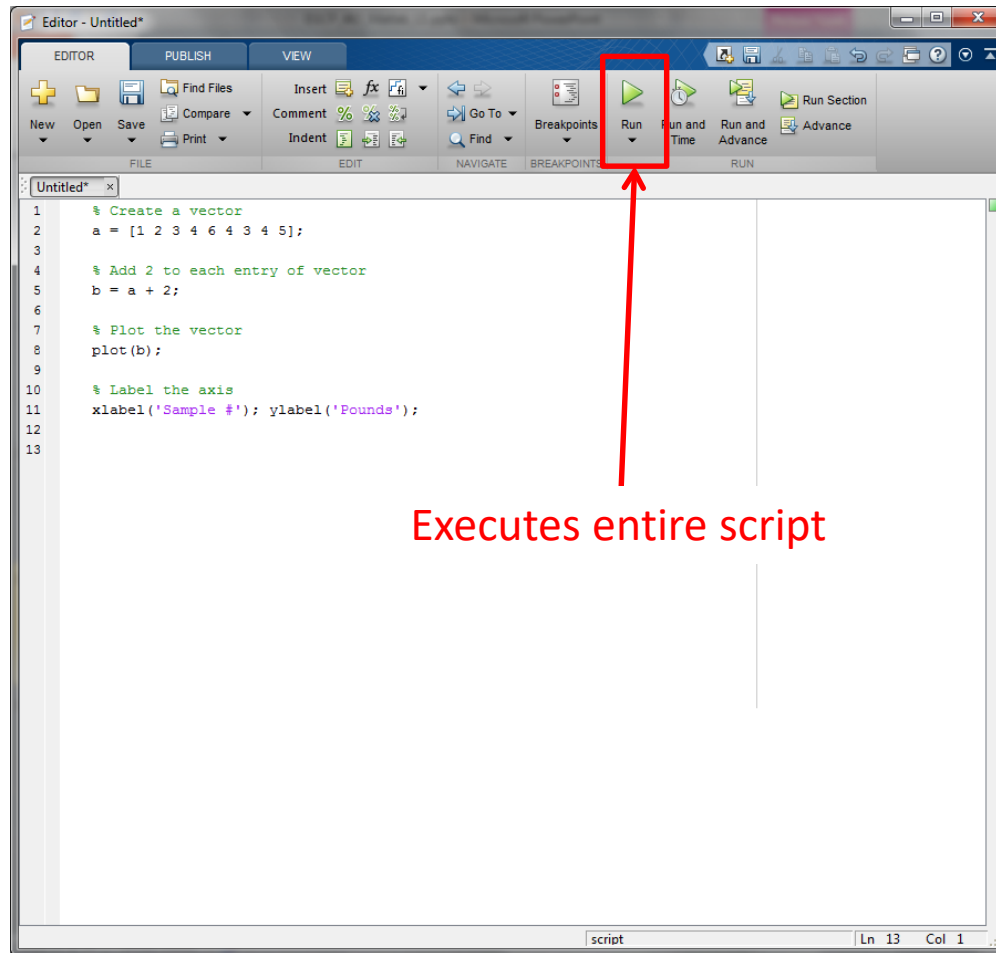
# Script Editor

- Instead of entering in the command window directly, you can also enter commands in the script editor and save as a m-file

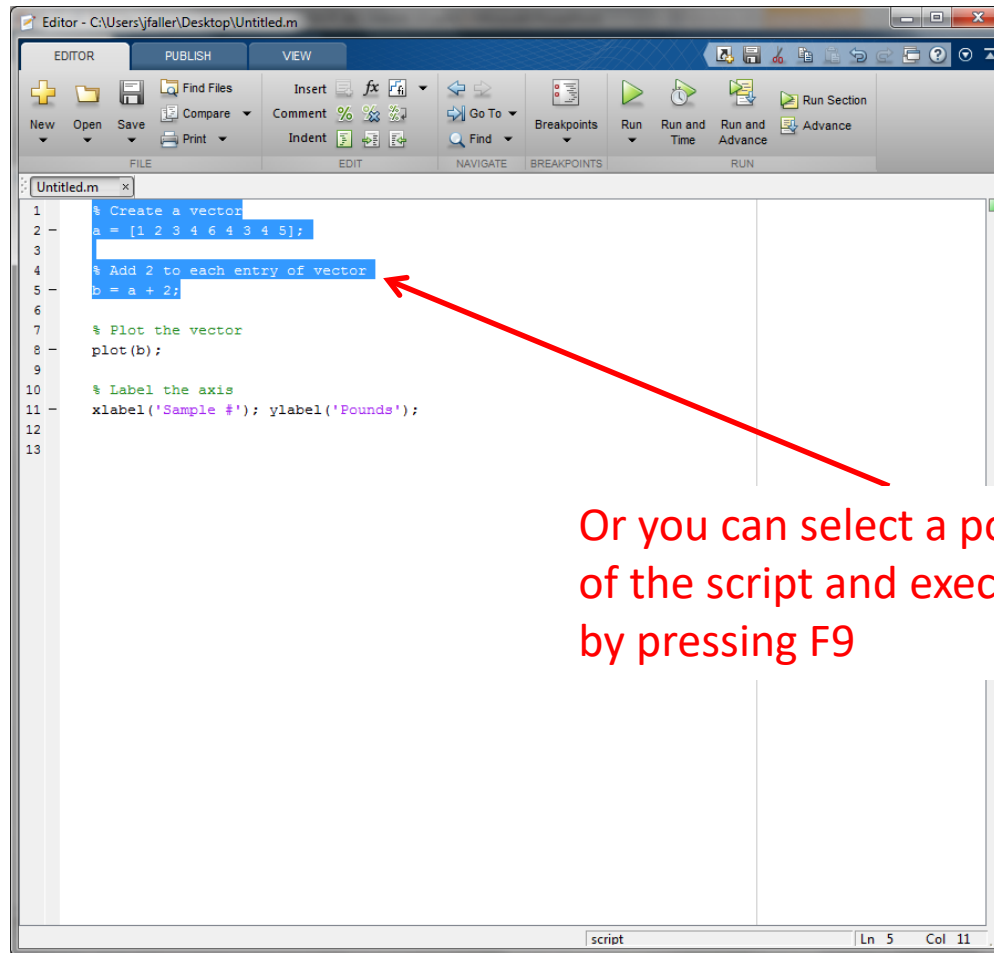


```
1 % Create a vector
2 a = [1 2 3 4 6 4 3 4 5];
3
4 % Add 2 to each entry of vector
5 b = a + 2;
6
7 % Plot the vector
8 plot(b);
9
10 % Label the axis
11 xlabel('Sample #'); ylabel('Pounds');
12
13
```

# Script Editor



# Script Editor



Or you can select a portion of the script and execute it only by pressing F9

# Getting Started

```
>> a = [ 1 2; 2 1 ]
```

```
a =
```

```
1 2  
2 1
```

```
>> a*a
```

```
ans =
```

```
5 4  
4 5
```

- Example
  - Define a matrix “a” and computed its square
  - “a times a”
- Text in bold is what you type in the command window
- Ordinary text is what Matlab outputs

# Matrices

- To enter the matrix:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- and store it in a variable “a”, do this:

```
>> a = [1 2; 3 4];
```

- To redisplay the matrix, just type its name:

```
>> a
```

- Once you know how to enter and display matrices, it is easy to compute with them. First we will square the matrix “a”:

```
>> a * a
```



# Matrices

- Now we'll try something a little harder. First we define a matrix  $b$ :

```
>> b = [1 2; 0 1];
```

- Then we compute the product  $ab$ :

```
>> a*b
```

- Finally, we compute the product in the other order:

```
>> b*a
```

# Matrices

- Notice that the two products are different
  - Matrix multiplication is non-commutative
- Of course, we can also add matrices:  
**>> a + b**
- Now let's store the result of this addition so that we can use it later:  
**>> s = a + b**

# Matrices

- Matrices can sometimes be inverted:

```
>> inv(s)
```

- To check that this is correct, we compute the product of  $s$  and its inverse:

```
>> s * inv(s)
```

- The result is the unit, or identity matrix. We can also write the computation as

```
>> s/s
```

- We can also write

```
>> s\s
```

- which is the same as

```
>> inv(s) * s
```

# Matrices

- To see that these operations, left and right division, are really different, we do the following:

```
>> a/b
```

```
>> a\b
```

- Not all matrices can be inverted, or used as the denominator in matrix division:

```
>> c = [ 1 1; 1 1 ];
```

```
>> inv(c);
```

- A matrix can be inverted if and only if its determinant is nonzero:

```
>> det(a)
```

```
>> det(c)
```

# Systems of Equations

- Now consider a linear equation

$$ax + by = p$$

$$cx + dy = q$$

- We can write this more compactly as

$$AX = B$$

- where the coefficient matrix A is

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- the vector of unknowns is

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

- and the vector on the right-hand side is

$$\begin{bmatrix} p \\ q \end{bmatrix}$$

- If A is invertible,  $X = (1/A)B$ , or, using Matlab notation,  $X = A \setminus B$ . Let's try this out by solving  $ax = b$  with a as before and  $b = [1; 0]$ . Note that b is a column vector.

```
>> b = [ 1; 0 ]
```

```
>> a\b
```

# Loops

- Loop Example
  - We regard  $x$  as representing (for example) the population state of an island
  - The first entry (1) gives the fraction of the population in the west half of the island, the second entry (0) give the fraction in the east half
  - The state of the population  $T$  units of time later is given by the rule  $y = ax$
  - This expresses the fact that an individual in the west half stays put with probability 0.8 and moves east with probability 0.2 (note  $0.8 + 0.2 = 1$ ), and the fact that in individual in the east stays put with probability 0.9 and moves west with probability 0.1
  - Thus, successive population states can be predicted/computed by repeated matrix multiplication

```
>> a = [ 0.8 0.1; 0.2 0.9 ]
```

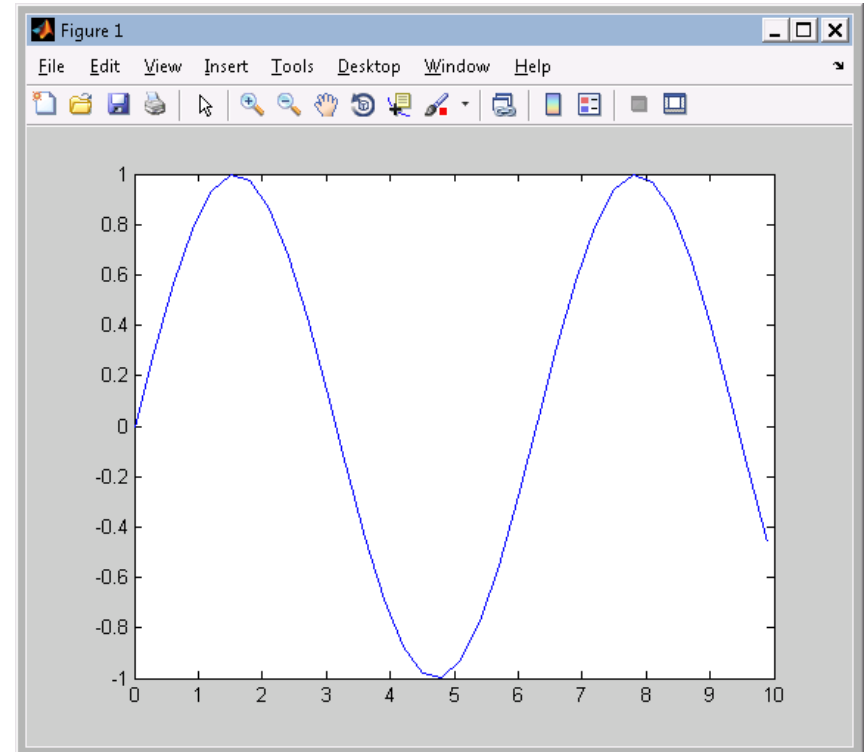
```
>> x = [ 1; 0 ]
```

```
>> for i = 1:20, x = a*x, end
```

# Graphing

## Functions of One Variable

- To make a graph of  $y = \sin(t)$  on the interval  $t = 0$  to  $t = 10$  we do the following:
  - >> **`t = 0:.3:10;`**
  - >> **`y = sin(t);`**
  - >> **`plot(t,y)`**
- The command `t = 0:.3:10;` defines a vector with components ranging from 0 to 10 in steps of 0.3
- The `y = sin(t);` defines a vector whose components are  $\sin(0)$ ,  $\sin(0.3)$ ,  $\sin(0.6)$ , etc.
- Finally, `plot(t,y)` use the vector of  $t$  and  $y$  values to construct the graph

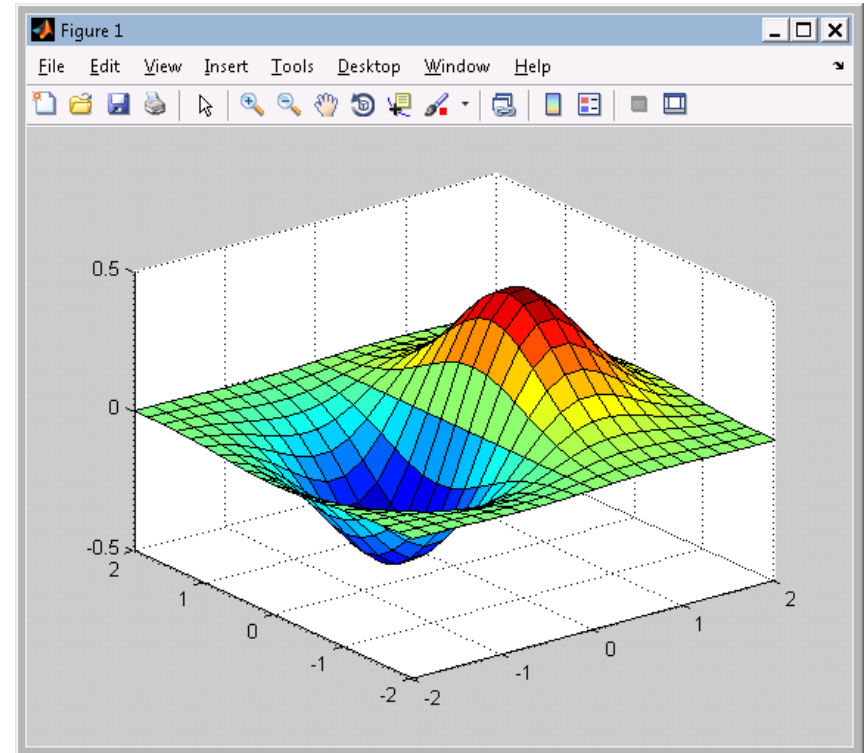


# Graphing

## Functions of Two Variable

- Here is how we graph the function
$$z(x, y) = xe^{-x^2 - y^2}$$

```
>> [x,y] = meshgrid(-2:.2:2, -2:.2:2);  
>> z = x .* exp(-x.^2 - y.^2);  
>> surf(x,y,z)
```
- The first command creates a matrix whose entries are the points of a grid in the square  $-2 \leq x \leq 2$ ,  $-2 \leq y \leq 2$
- The small squares which make up the grid are 0.2 units wide and 0.2 unit tall
- The second command creates a matrix whose entries are the values of the function  $z(x,y)$  at the grid points
- The third command uses this information to construct the graph





# Common Commands and Operators

- <http://www.hkn.umn.edu/resources/files/matlab/MatlabCommands.pdf>

# Useful Tutorials

- Download MATLAB and do the following tutorials:
  - [Basic Matrix Operations](#)
  - [Getting Started with MATLAB](#)
  - [Matlab Overview Video](#)
  - [Analyzing and Visualizing Data with MATLAB](#)
  - [Programming and Developing Algorithms with MATLAB](#)
  - [Signal Related Videos](#)