# Programming in MATLAB/Octave

Lecture 2 – Sunday October 16, 2016

# Outline

- MATLAB Environment
- Identifiers
- Constants
- Variables
- Vectors and Matrices
- Plotting with MATLAB
- MATLAB Functions

# Outline

- **MATLAB Environment**
- Identifiers
- Constants
- Variables
- Vectors and Matrices
- Plotting with MATLAB
- MATLAB Functions

# MATLAB Environment

- MATLAB is a program for doing numerical computation. It was originally designed for solving linear algebra type problems using matrices.
- It's name is derived from **MAT**rix **LAB**oratory.
- MATLAB is a software environment for interactive numerical computations.
- MATLAB is a high-level language and interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming languages such as C, C++, and Fortran.

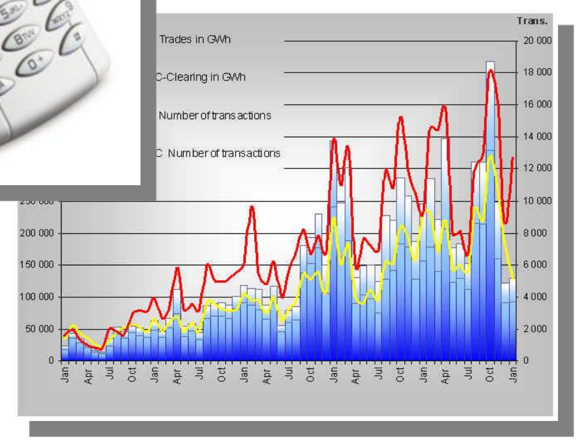
# MATLAB Environment

- **Tasks**
  - ◇ Matrix computations and linear algebra
  - ◇ Solving nonlinear equations
  - ◇ Numerical solution of differential equations
  - ◇ Mathematical optimization
  - ◇ Statistics and data analysis
  - ◇ Signal processing
  - ◇ Modelling of dynamical systems
  - ◇ Solving partial differential equations
  - ◇ Simulation of engineering systems

# MATLAB Environment

- **Usage**

Matlab used (on a daily basis) in many engineering companies

# MATLAB Environment

- **Background**

**Matlab** = **Mat**rix **Lab**oratory

- Originally a user interface for numerical linear algebra routines (Lapak/Linpak)

- Commercialized 1984 by The Mathworks

- Since then heavily extended (defacto-standard)

◇ **Alternatives**

Matrix-X

Octave (free; GNU)

Lyme (free; Palm)

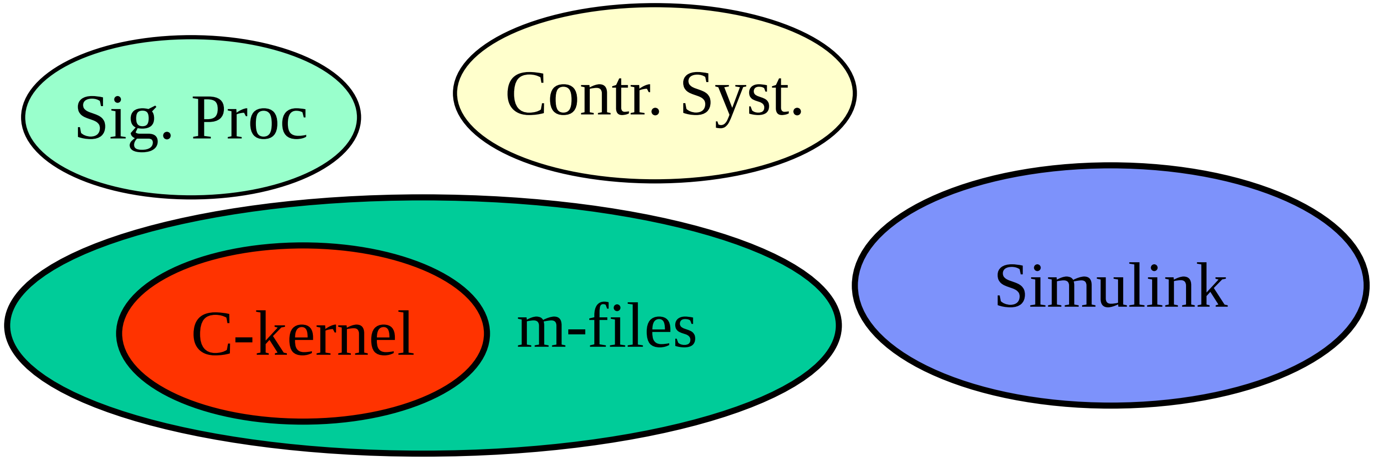
◇ **Complements**

Maple (symbolic)

Mathematica (symbolic)

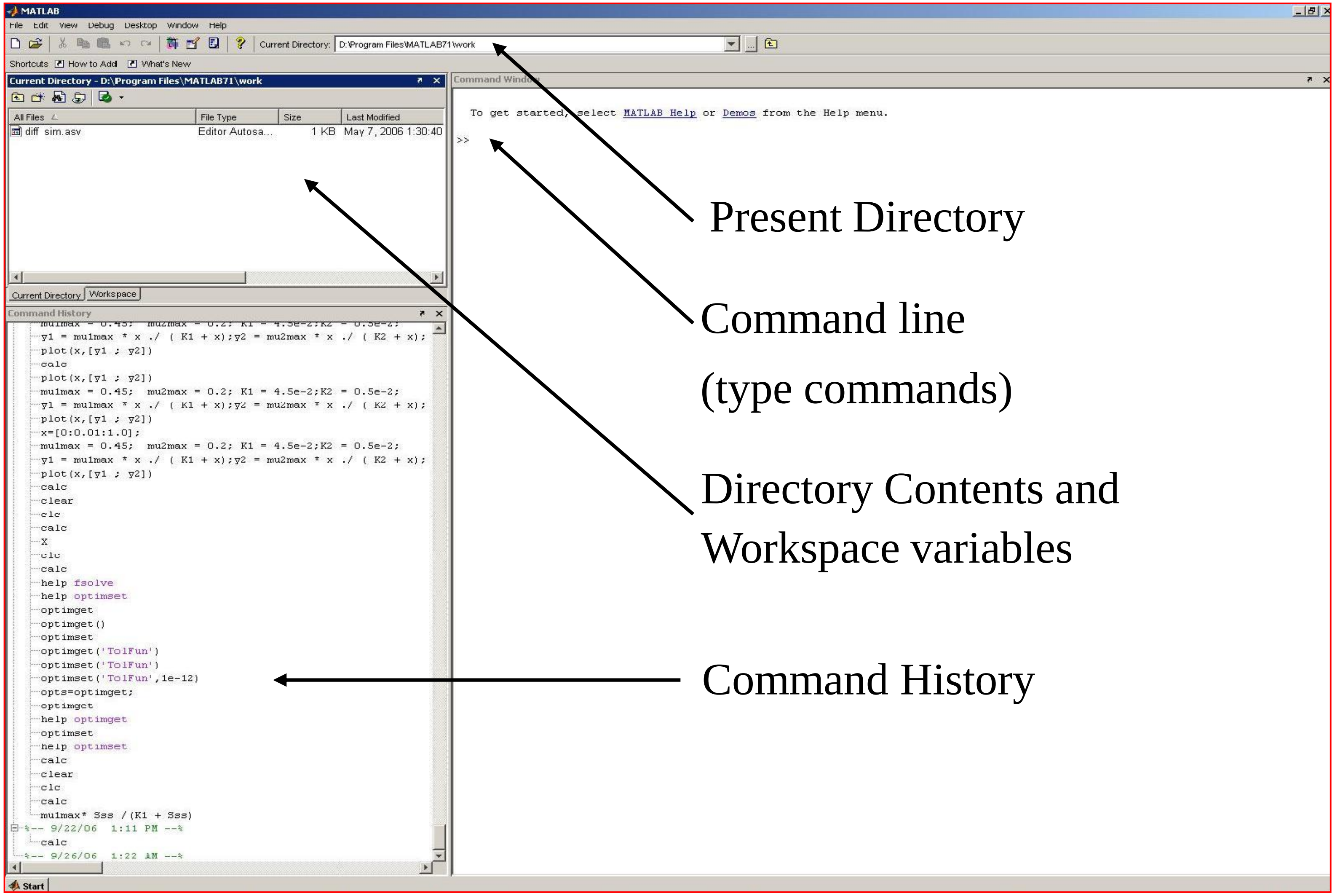
# MATLAB Environment

- **Functionality**
  - ◇ Core functionality: compiled C-routines
  - ◇ Most functionality is given as ***m-files***, grouped into toolboxes
    - – m-files contain source code, can be copied and altered
    - – m-files are platform independent (PC, Unix/Linux, MAC)
  - ◇ Simulation of dynamical systems is performed in Simulink.



# MATLAB Environment

- **Windows**



# MATLAB Environment

- **MATLAB Special Variables**

| | |
|---|---|
| ans | Default variable name for results |
| pi | Value of $\pi$ |
| eps | Smallest incremental number |
| inf | Infinity |
| NaN | Not a number e.g. 0/0 |
| i and j | i = j = square root of -1 |
| realmin | The smallest usable positive real number |
| realmax | The largest usable positive real number |

# MATLAB Environment

- **MATLAB Math & Assignment Operators**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Power | ^ | or | .^ | a^b | or | a.^b |
| Multiplication | * | or | .* | a*b | or | a.*b |
| Division | / | or | ./ | a/b | or | a./b |
| or | \ | or | .\ | b\a | or | b.\a |
| NOTE: | 56/8 = 8\56 | | | | | |

- (unary) + (unary)

| | | | |
|---|---|---|---|
| Addition | + | a + b | |
| Subtraction | - | a - b | |
| Assignment | = | a = b | (assign b to a) |

# MATLAB Environment

- **Other MATLAB Symbols**

| | |
|---|---|
| >> | prompt |
| . . . | continue statement on next line |
| , | separate statements and data |
| % | start comment which ends at end of line |
| ; | (1) suppress output |
| | (2) used as a row separator in a matrix |
| : | specify range |

# MATLAB Environment

- **Interactive Calculations**

Matlab is interactive, no need to declare variables

```
>> 2+3*4/2
>> a=5e-3; b=1; a+b
```

Most elementary functions and constants are already defined

```
>> cos(pi)
>> abs(1+i)
>> sin(pi)
```

Last call gives answer 1.2246e-016 !?

# MATLAB Environment

- **Variable and Memory Management**

Matlab uses double precision (approx. 16 significant digits)

```
>> format long
>> format compact
```

All variables are shown with

```
>> who
>> whos
```

Variables can be stored on file

```
>> save filename
>> clear
>> load filename
```

# MATLAB Environment

- **Some Useful MATLAB commands**
  - ◇ who — List known variables
  - ◇ whos — List known variables plus their size
  - ◇ help — >> help sqrt — Help on using sqrt
  - ◇ lookfor — >> lookfor sqrt — Search for keyword sqrt in m-files
  - ◇ what — >> what a: — List MATLAB files in a:
  - ◇ clear — Clear all variables from work space
  - ◇ clear x y — Clear variables x and y from work space
  - ◇ clc — Clear the command window

# MATLAB Environment

- **Some Useful MATLAB commands**

| | |
|---|---|
| ◇ what | List all m-files in current directory |
| ◇ dir | List all files in current directory |
| ◇ ls | Same as dir |
| ◇ type test | Display test.m in command window |
| ◇ delete test | Delete test.m |
| ◇ cd a: | Change directory to a: |
| ◇ chdir a: | Same as cd |
| ◇ pwd | Show current directory |
| ◇ which test | Display directory path to ‘closest’ test.m |

# MATLAB Environment

- **The Help System**

Search for appropriate function

>> lookfor *keyword*

Rapid help with syntax and function definition

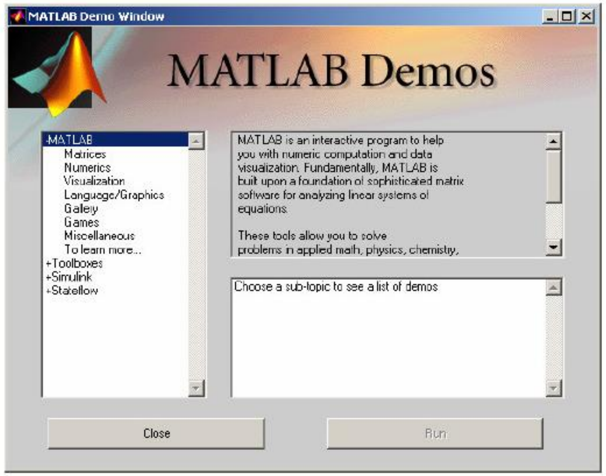
>> help *function*

An advanced hyperlinked help system is launched by

>> helpdesk

Demo launched by

>> demo



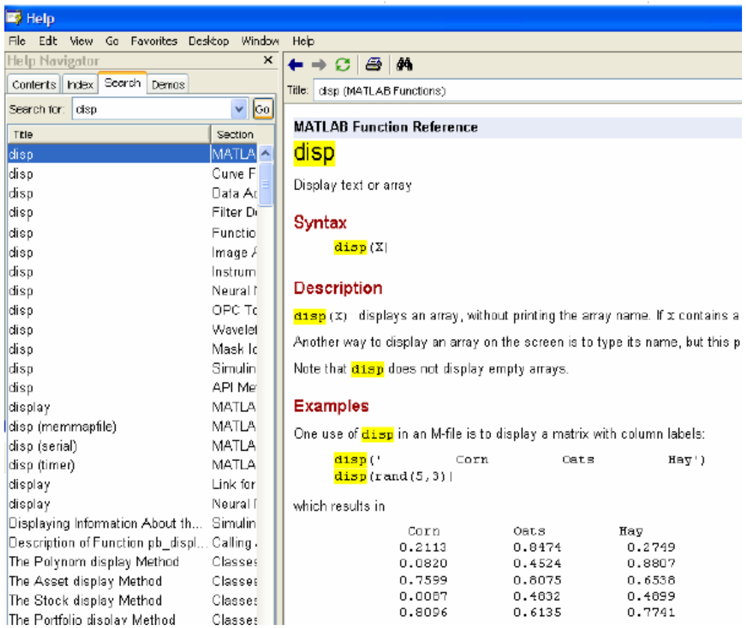
# MATLAB Environment

- **Technical Documentations**

To get a nicer version of help with examples and easy-to-read descriptions

>> doc *function*

>> doc *disp*




To search for a function by specifying keywords:

»doc + *Search tab*

# Outline

- MATLAB Environment
- **Identifiers**
- Constants
- Variables
- Vectors and Matrices
- Plotting with MATLAB
- MATLAB Functions

# Identifiers

- **MATLAB Variable Names**
  - ◇ Variable names ARE case sensitive
  - ◇ Variable names can contain up to 63 characters (as of MATLAB 6.5 and newer)
  - ◇ Variable names must start with a letter followed by letters, digits, and underscores.

# Identifiers

- Identifiers are all the words that build up the program
- An identifier is a sequence of letters, digits and underscores "_"
- Maximal length of identifiers is 63 characters
- Can't start with a digit
- Can't be a reserved word

Legal identifiers

- time
- day_of_the_week
- bond007
- findWord

Illegal identifiers

- 007bond
- #time
- ba-baluba
- if
- while

# Identifiers

- **Reserved words**

There are 17 reserved words:

for
function
otherwise
try
break
end
return
switch
catch
if
elseif
continue
global
while
case
else
persistent

# Outline

- MATLAB Environment
- Identifiers
- **Constants**
- Variables
- Vectors and Matrices
- Plotting with MATLAB
- MATLAB Functions

# Constants

The value of a constant is fixed and does not change throughout the program.

Numbers

100 0.3

Chars

'c'

Strings

'I like to eat sushi'

'1 + 2'

Arrays

[ 1 2 3 4 5 ]

Matrices

[5 3
 4 2]

# Outline

- MATLAB Environment
- Identifiers
- Constants
- **Variables**
- Vectors and Matrices
- Plotting with MATLAB
- MATLAB Functions

# Variables

Variable

Constant

```
>> salary = 9000;
>> new_salary = salary * 3;
>> disp(new_salary);
27000
```

Library function

Computer memory

salary

9000

new_salary

27000

If we update salary, new_salary will NOT be updated automatically

# Outline

- MATLAB Environment
- Identifiers
- Constants
- Variables
- **Vectors and Matrices**
- Plotting with MATLAB
- MATLAB Functions

# Vectors and Matrices

Vectors (arrays) are defined as

```
>> v = [1, 2, 4, 5]
>> w = [1; 2; 4; 5]
```

$$v = \begin{bmatrix} 1 & 2 & 4 & 5 \end{bmatrix}$$

$$w = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 5 \end{bmatrix}$$

Matrices (2D arrays) defined similarly

```
>> A = [1,2,3;4,-5,6;5,-6,7]
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ 5 & -6 & 7 \end{bmatrix}$$

# Vectors and Matrices

- **Matrix Operators**

All common operators are overloaded

```
>> v + 2
```

Common operators are available

```
>> B = A'
>> A*B
>> A+B
```

Note:

Matlab is case-sensitive

A and a are two different variables

# Vectors and Matrices

- **Indexing Matrices**

Indexing using parentheses

>> A(2,3)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

Index submatrices using vectors of row and column indices

>> A([2 3],[1 2])

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

Ordering of indices is important!

>> B=A([3 2],[2 1])

>> B=[A(3,2),A(3,1);A(2,2);A(2,1)]

$$B = \begin{bmatrix} 6 & 5 \\ -5 & 4 \end{bmatrix}$$

# Vectors and Matrices

- **Indexing Matrices**

Index complete row or column using the colon operator

>> A(1,:)

$$A = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{3} \\ 4 & -5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

Can also add limit index range

>> A(1:2,:)

>> A([1 2],:)

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & -5 & 6 \\ 5 & 6 & 7 \end{bmatrix}$$

General notation for colon operator

>> v=1:5

>> w=1:2:5

$$v = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

$$w = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}$$

# Vectors and Matrices

- **Numerical Linear Algebra**

Basic numerical linear algebra

```
>> z=[1;2;3]; x=inv(A)*z
>> x=A\z
```

$$Ax = \begin{bmatrix}1\\2\\3\end{bmatrix}$$

$$x = A^{-1}\begin{bmatrix}1\\2\\3\end{bmatrix}$$

Many standard functions predefined

```
>> det(A)
>> rank(A)
>> eig(A)
```

The number of input/output arguments can often be varied

```
>> [V,D]=eig(A)
```

# Outline

- MATLAB Environment
- Identifiers
- Constants
- Variables
- Vectors and Matrices
- **Plotting with MATLAB**
- MATLAB Functions

# Plotting with MATLAB

- MATLAB will plot one vector vs. another. The first one will be treated as the abscissa (or x) vector and the second as the ordinate (or y) vector. The vectors have to be the same length.

- MATLAB will also plot a vector vs. its own index. The index will be treated as the abscissa vector. Given a vector "time" and a vector "dist" we could say:

```
>> plot (time, dist)     % plotting versus time
>> plot (dist)           % plotting versus index
```

# Plotting with MATLAB

- There are commands in MATLAB to "annotate" a plot to put on axis labels, titles, and legends. For example:

```
>> % To put a label on the axes we would use:
>> xlabel ('X-axis label')
>> ylabel ('Y-axis label')

>> % To put a title on the plot, we would use:
>> title ('Title of my plot')
```

# Plotting with MATLAB

- Vectors may be extracted from matrices. Normally, we wish to plot one column vs. another. If we have a matrix "mydata" with two columns, we can obtain the columns as a vectors with the assignments as follows:

```
>> first_vector = mydata ( : , 1) ;         % First column
>> second_vector = mydata ( : , 2) ;    % Second one
>>        % and we can plot the data
>> plot ( first_vector , second_vector )
```

# Plotting with MATLAB

Visualization of vector data is available

```
>> x=-pi:0.1:pi; y=sin(x);
>> plot(x,y)
>> plot(x,y,'s-')
>> xlabel('x'); ylabel('y=sin(x)');
```

Can change plot properties in Figure menu, or via "handle"

```
>> h=plot(x,y); set(h, 'LineWidth', 4);
```

Many other plot functions available `>> v=1:4; pie(v)`

# Plotting with MATLAB

Three-dimensional graphics

```
>> A = zeros(32);
>> A(14:16,14:16) = ones(3);
>> F=abs(fft2(A));
>> mesh(F)
>> rotate3d on
```

Several other plot functions available

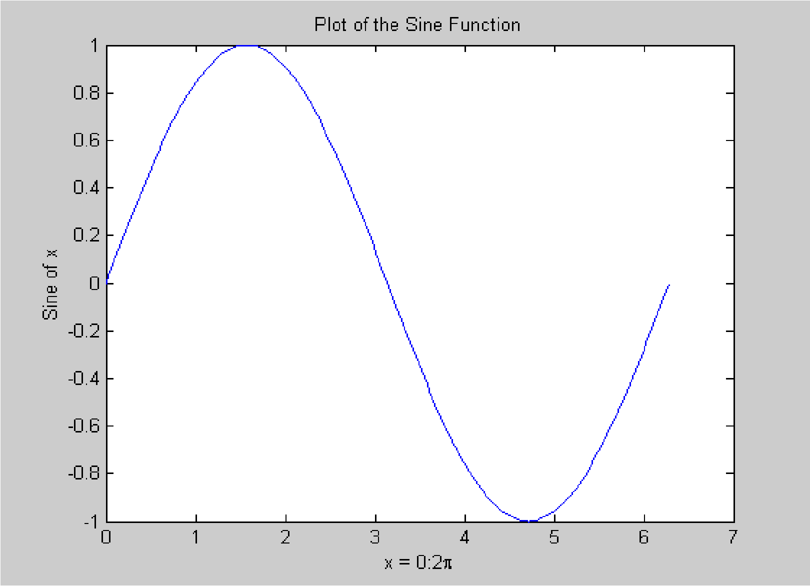
```
>> surfl(F)
```

Can change lightning and material properties

```
>> cameramenu
>> material metal
```

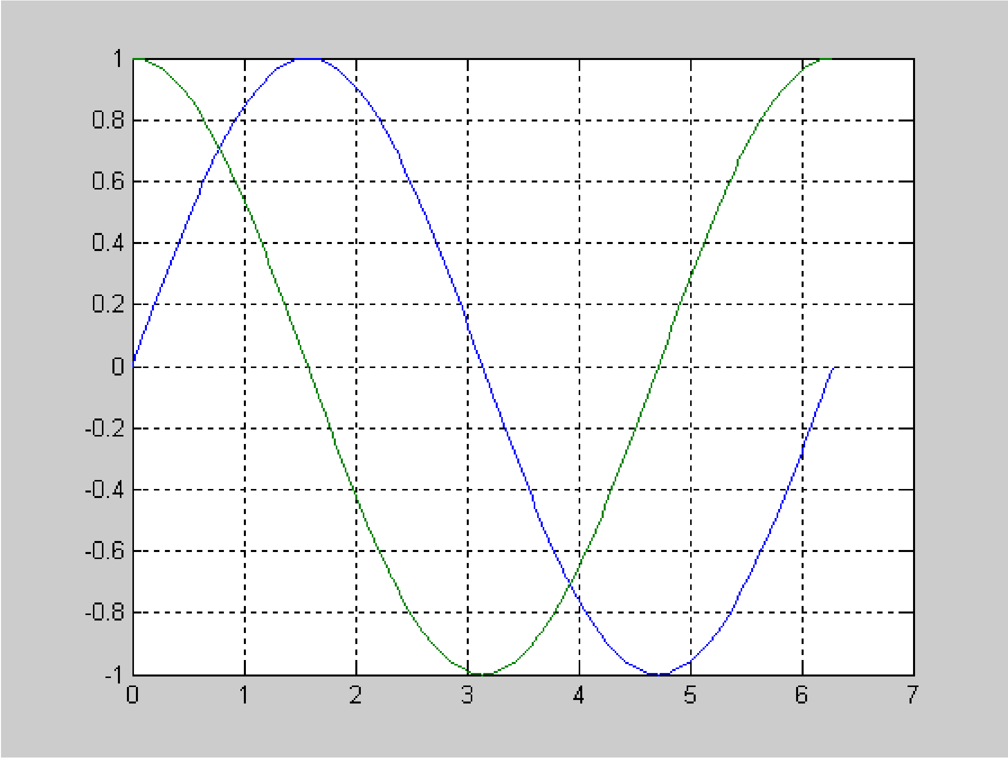
# Plotting with MATLAB

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
xlabel('x = 0:2\pi')
ylabel('Sine of x')
title('Plot of the Sine Function')
```



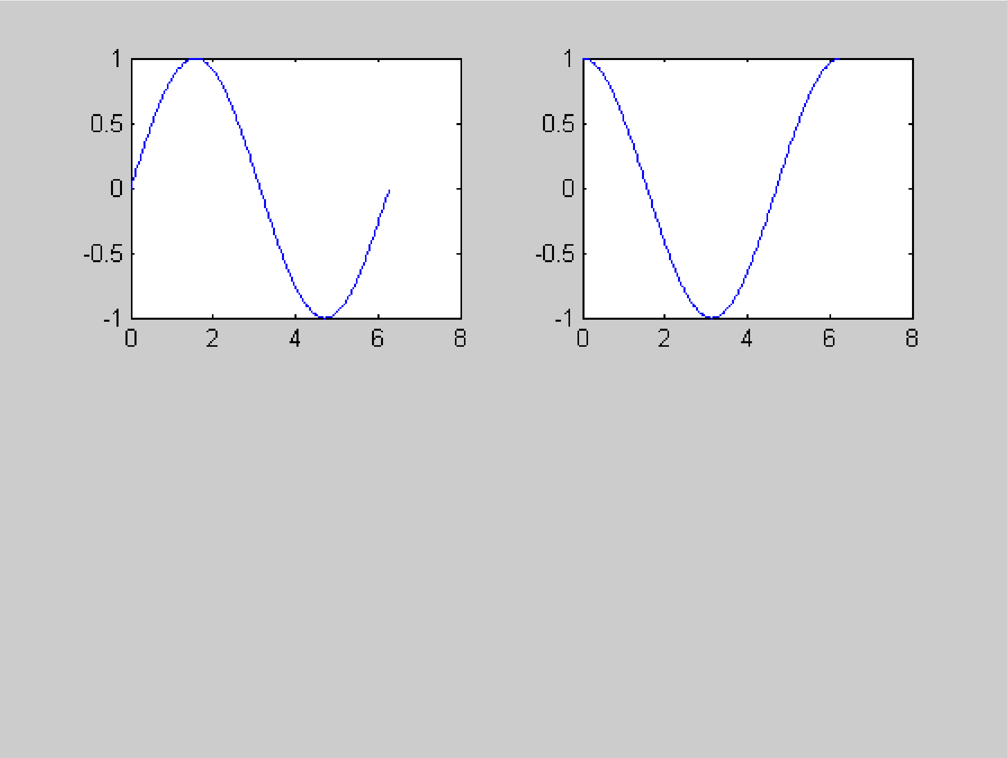
# Plotting with MATLAB

```
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
plot(t,y1,t,y2)
grid on
```



# Plotting with MATLAB

```
t = 0:pi/100:2*pi;
y1=sin(t);
y2=sin(t+pi/2);
subplot(2,2,1)
plot(t,y1)
subplot(2,2,2)
plot(t,y2)
```



# Plotting with MATLAB

**3-D Plots:** Try these

**contour(x,y,z)**

Generate a contour plot of the surface defined by the matrix z

**contour(x,y,z,v)**

Generate a contour plot. The vector v defines the vales to use for the contour lines.

**meshc(x_pts,y_pts,z)**

Generate an open mesh plot of the surface defined by the matrix z. The arguments x_pts and y_pts can be vectors defining the ranges of values of the x- and y coordinates.

# Plotting with MATLAB

| | |
|---|---|
| • plot | linear plot |
| • stem | discrete plot |
| • grid | add grid lines |
| • xlabel | add X-axis label |
| • ylabel | add Y-axis label |
| • title | add graph title |
| • subplot | divide figure window |
| • figure | create new figure window |
| • pause | wait for user response |

# Outline

- MATLAB Environment
- Identifiers
- Constants
- Variables
- Vectors and Matrices
- Plotting with MATLAB
- **MATLAB Functions**

# MATLAB Functions

- **Mathematical Functions**

***Example:*** If you want to compute the sine of an angle and tore the result in b:

```
b=sin(angle);
```

```
b=sin(angle*pi/180);
```

```
angle_radians = angle*pi/180;
b=sin(angle_radians);
```

# MATLAB Functions

- **Mathematical Functions**

***Example:*** If you want to compute logarithm of the absolute vale of x:

```
log_x=log(abs(x));
```

Common Math Functions:

| | |
|---|---|
| **abs(x)** | Computes the absolute value of **x**. |
| **sqrt(x)** | Computes the square root of **x**. |
| **round(x)** | Rounds **x** to the nearest integer. |
| **fix(x)** | Rounds (or truncates) **x** to the nearest integer toward 0. |
| **floor(x)** | Rounds **x** to the nearest integer toward $-\infty$. |
| **ceil(x)** | Rounds **x** to the nearest integer toward $\infty$. |
| **sign(x)** | Returns a value of −1 if **x** is less than 0, a value of 0 if **x** equals 0, and a value of 1 otherwise. |
| **rem(x,y)** | Returns the remainder of *x*/*y*. For example, **rem(25,4)** is 1, and **rem(100,21)** is 16. This function is also called a **modulus** function. |
| **exp(x)** | Computes $e^x$, where *e* is the base for natural logarithms, or approximately 2.718282. |
| **log(x)** | Computes 1n **x**, the natural logarithm of **x** to the base *e*. |
| **log10(x)** | Computes $\log_{10}$ **x**, the common logarithm of **x** to the base 10. |

# MATLAB Functions

- **Trigonometric Functions**

Trigonometric functions assume that angles in radians.

```
angle_degrees=angle_radians*(180/pi);
angle_radians=angle_degrees*(pi/180);
```

Trigonometric Functions:

| | |
|---|---|
| `sin(x)` | Computes the sine of **x**, where **x** is in radians. |
| `cos(x)` | Computes the cosine of **x**, where **x** is in radians. |
| `tan(x)` | Computes the tangent of **x**, where **x** is in radians. |
| `asin(x)` | Computes the arcsine or inverse sine of **x**, where **x** must be between $-1$ and 1. The function returns an angle in radians between $-\pi/2$ and $\pi/2$. |
| `acos(x)` | Computes the arccosine or inverse cosine of **x**, where **x** must be between $-1$ and 1. The function returns an angle in radians between 0 and $\pi$. |
| `atan(x)` | Computes the arctangent or inverse tangent of **x**. The function returns an angle in radians between $-\pi/2$ and $\pi/2$. |
| `atan2(y,x)` | Computes the arctangent or inverse tangent of the value $y/x$. The function returns an angle in radians that will be between $-\pi$ and $\pi$, depending on the signs of **x** and **y**. |

# MATLAB Functions

- **Trigonometric Functions**

The other trigonometric functions can be computed using these equations:

$$\sec(x) = \frac{1}{\cos(x)} \qquad \csc(x) = \frac{1}{\sin(x)} \qquad \cot(x) = \frac{1}{\tan(x)}$$

$$\operatorname{arcsec}(x) = \operatorname{arcos}\left(\frac{1}{x}\right) \text{ for } |x| \geq 1$$

$$\operatorname{arccsc}(x) = \arcsin\left(\frac{1}{x}\right) \text{ for } |x| \geq 1$$

$$\operatorname{arccot}(x) = \arccos\left(\frac{x}{\sqrt{1+x^2}}\right)$$

# MATLAB Functions

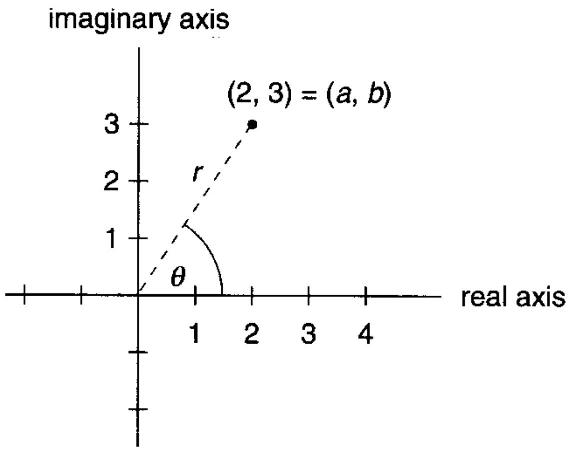
- **Complex Number Functions**

Complex numbers are needed to solve many problems in science and engineering.

**Arithmetic Operations with Complex Numbers**

| Operation | Result |
|---|---|
| $c_1 + c_2$ | $(a_1 + a_2) + i(b_1 + b_2)$ |
| $c_1 - c_2$ | $(a_1 - a_2) + i(b_1 - b_2)$ |
| $c_1 \cdot c_2$ | $(a_1a_2 - b_1b_2) + i(a_1b_2 + a_2b_1)$ |
| $\frac{c_1}{c_2}$ | $\left(\frac{a_1a_2 + b_1b_2}{a_2^2 + b_2^2}\right) + i\left(\frac{a_2b_1 - b_2a_1}{a_2^2 + b_2^2}\right)$ |
| $\lvert c_1 \rvert$ | $\sqrt{a_1^2 + b_1^2}$ (magnitude or absolute value of $c_1$) |
| $c_1^*$ | $a_1 - ib_1$ (conjugate of $c_1$) |

(Assume that $c_1 = a_1 + ib_1$ and $c_2 = a_2 + ib_2$.)




*Complex plane*

A complex variable can be defined in MATLAB as follows:

```
x = 1 - i*0.5;
```

# MATLAB Functions

- **Complex Number Functions**

MATLAB includes several functions that are specific for complex numbers and their conversion.

| | |
|---|---|
| `conj(x)` | Computes the complex **conjugate** of the complex number `x`. Thus, if `x` is equal to $a + i\,b$, then `conj(x)` will be equal to $a - i\,b$. |
| `real(x)` | Computes the real portion of the complex number `x`. |
| `imag(x)` | Computes the imaginary portion of the complex number `x`. |
| `abs(x)` | Computes the absolute value or **magnitude** of the complex number `x`. |
| `angle(x)` | Computes the angle using the value of `atan2(imag(x), real(x))`; thus, the angle value is between $-\pi$ and $\pi$. |

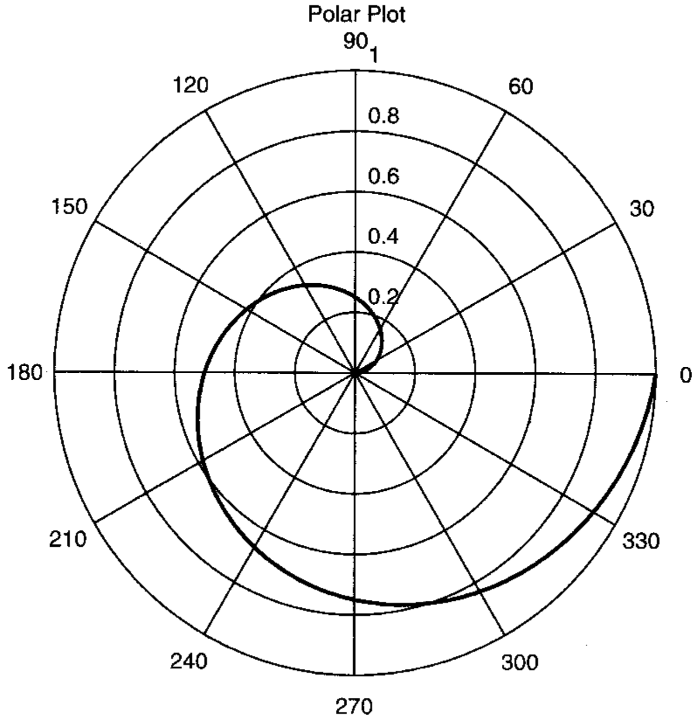
# MATLAB Functions

- **Complex Number Functions**

Polar Plots

| | |
|---|---|
| `polar(theta,r)` | Generates a polar plot of the angles `theta` (in radians) versus the magnitudes `r`. |

```
theta = 0:2*pi/100:2*pi;
r = theta/(2*pi);
polar(theta,r),title('Polar Plot')
```




*Polar plot with increasing radius.*

# MATLAB Functions

- **Polynomial Functions**

A polynomial function is a function of a single variable that can be expressed in the general form:

$$f(x) = a_0 x^N + a_1 x^{N-1} + a_2 x^{N-2} + \cdots + a_{N-2} x^2 + a_{N-1} x + a_N$$

Where the variable is x and the polynomial coefficient are represented by the values $a_o$, $a_1$....$a_n$.

The degree of a polynomial is equal to the largest value used as an exponent.

$g(x) = a_0 x^3 + a_1 x^2 + a_2 x + a_3$ ⇨ Cubic (degree 3) polynomial

$h(x) = x^3 - 2x^2 + 0.5x - 6.5.$ ⇨ Example for Cubic polynomial

# MATLAB Functions

- **Polynomial Functions**

There are several ways to evaluate a polynomial using MATLAB

***Example:*** $f(x) = 3x^4 - 0.5x^3 + x - 5.2$

```
f = 3*x^4 - 0.5*x^3 + x - 5.2;
```

```
f = 3*x.^4 - 0.5*x.^3 + x - 5.2;
```

→ If x is a vector or a matrix

| | |
|---|---|
| `polyval(a,x)` | Evaluates a polynomial with coefficients **a** for the values in **x**. The result is a matrix the same size as **x**. |

```
a = [3,-0.5,0,1,-5.2];
f = polyval(a,x);
```

or

```
f = polyval([3,-0.5,0,1,-5.2],x);
```

# MATLAB Functions

- **Polynomial Functions**

This code will generate 201 points of the polynomial over the desired interval.

```
x = 0:5/200:5;
a = [-1,0,3,-2.5,0,-2.5];
g = polyval(a,x);
plot(x,g),title('Polynomial Function')
```

# MATLAB Functions

- **Polynomial Functions**

Polynomial Operations:

$$g(x) = x^4 - 3x^2 - x + 2.4$$

$$h(x) = 4x^3 - 2x^2 + 5x - 16$$

$$s(x) = g(x) + h(x)$$

MATLAB statements to perform this polynomial addition are

```
g = [1,0,-3,-1,2.4];
h = [0,4,-2,5,-16];
s = g + h;
```

# MATLAB Functions

- **Polynomial Functions**

MATLAB contains functions to perform polynomial multiplication and division:

| | |
|---|---|
| `conv(a,b)` | Computes a coefficient vector that contains the coefficients of the product of polynomials represented by the coefficients in `a` and `b`. The vectors `a` and `b` do not have to be the same size. |
| `[q,r] = deconv(n,d)` | Returns two vectors. The first vector contains the coefficients of the quotient and the second vector contains the coefficients of the remainder polynomial. |

# MATLAB Functions

- **Polynomial Functions**

***Example:***

$$g(x) = (3x^3 - 5x^2 + 6x - 2)(x^5 + 3x^4 - x^2 + 2.5)$$

```
a = [3,-5,6,-2];
b = [1,3,0,-1,0,2.5];
g = conv(a,b);
```

# MATLAB Functions

- **Polynomial Functions**

***Example:***

$$g(x) = 3x^8 + 4x^7 - 9x^6 + 13x^5 - x^4 + 1.5x^3 - 10.5x^2 + 15x - 5$$

$$h(x) = \frac{3x^8 + 4x^7 - 9x^6 + 13x^5 - x^4 + 1.5x^3 - 10.5x^2 + 15x - 5}{x^5 + 3x^4 - x^2 + 2.5}$$

```
g = [3,4,-9,13,-1,1.5,-10.5,15,-5];
b = [1,3,0,-1,0,2.5];
[q,r] = deconv(g,b);
```

As expected, the quotient coefficient vector is [3,-5,6,-2], which represents a quotient polynomial of $3x^3-5x^2+6x-2$, the remainder vector contains zeros.

# MATLAB Functions

- **Polynomial Functions**

**Roots of Polynomial:** The solution of many engineering problems involve finding the roots of an equation of the form

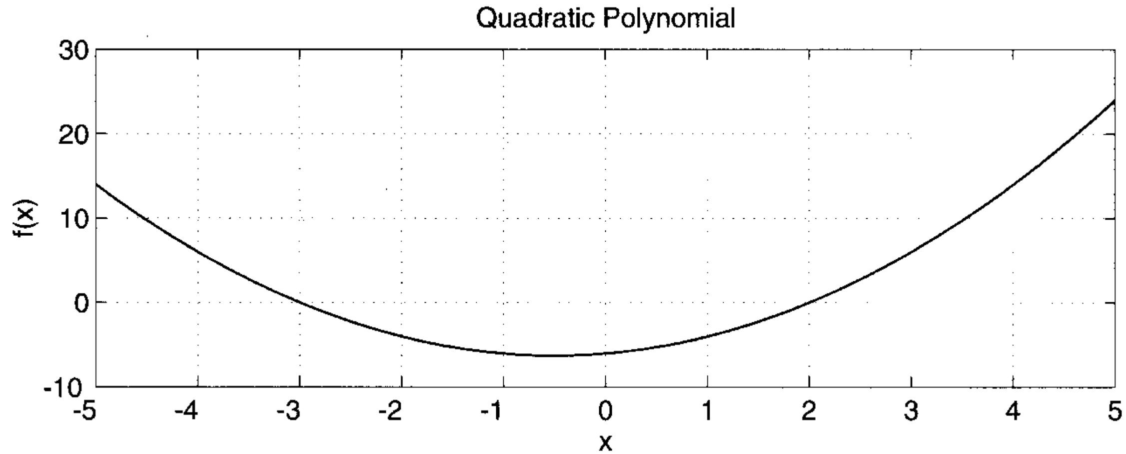
$$y = f(x)$$

Where the roots are the values of x for which y is equal to 0.

$$f(x) = x^2 + x - 6$$

$$= (x - 2)(x + 3)$$

Roots of this polynomial are 2 and -3




*Polynomial with two real roots*

# MATLAB Functions

- **Polynomial Functions**

**Cubic polynomial:**

$$f(x) = a_0x^3 + a_1x^2 + a_2x + a_3$$

3 real distinct roots

3 real multiple roots

1 distinct real root and 2 multiple real roots

1 real root and a complex conjugate pair of roots

Examples of functions:

$$f_1(x) = (x - 3)(x + 1)(x - 1) = x^3 - 3x^2 - x + 3$$

$$f_2(x) = (x - 2)^3 = x^3 - 6x^2 + 12x - 8$$

$$f_3(x) = (x + 4)(x - 2)^2 = x^3 - 12x + 16$$

$$f_4(x) = (x + 2)(x - (2+i))(x - (2-i)) = x^3 - 2x^2 - 3x + 10$$

# MATLAB Functions

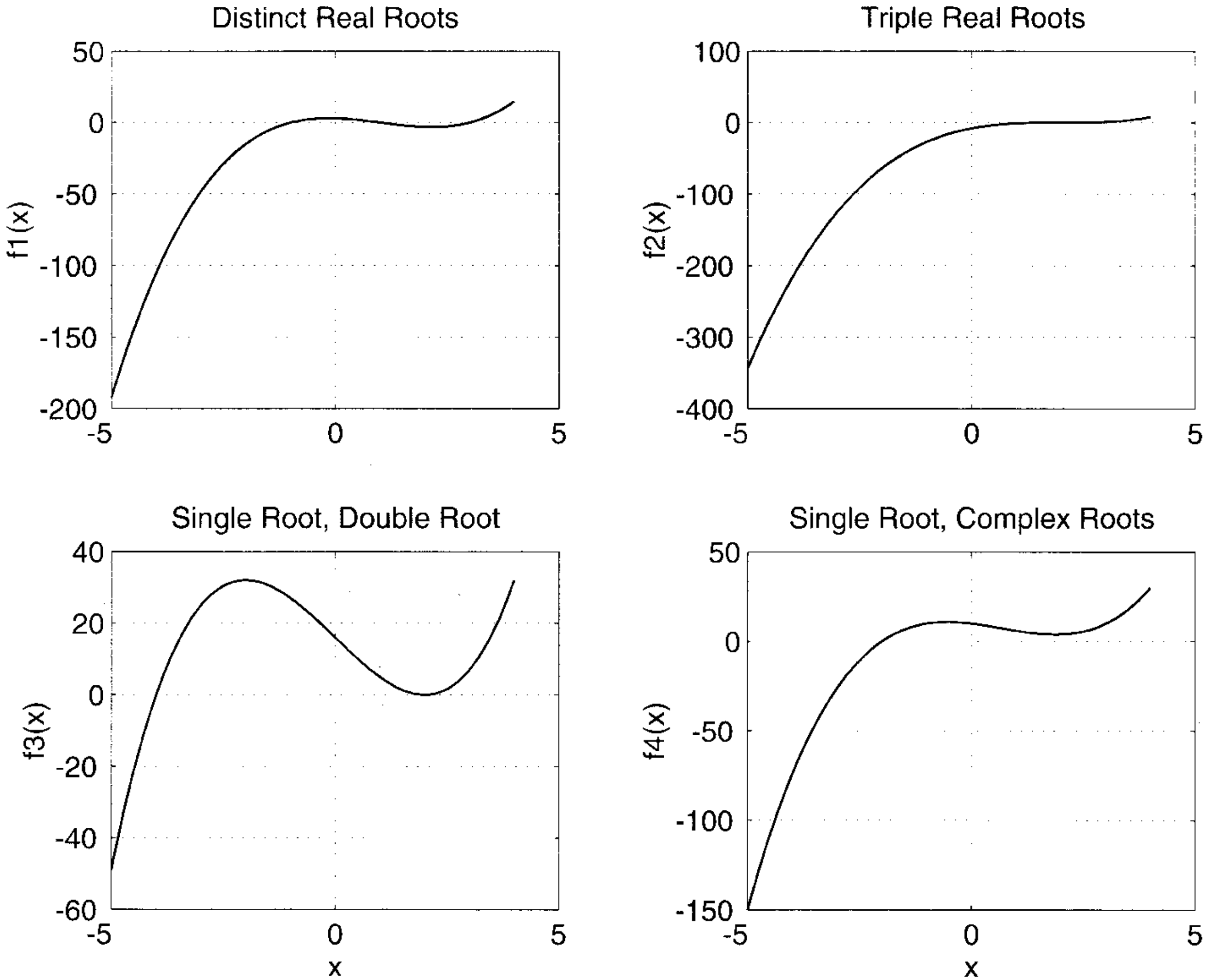
- **Polynomial Functions**

$$f_1(x) = (x-3)(x+1)(x-1) = x^3 - 3x^2 - x + 3$$

$$f_2(x) = (x-2)^3 = x^3 - 6x^2 + 12x - 8$$

$$f_3(x) = (x+4)(x-2)^2 = x^3 - 12x + 16$$

$$f_4(x) = (x+2)(x-(2+i))(x-(2-i)) = x^3 - 2x^2 - 3x + 10$$




*Cubic polynomials*

# MATLAB Functions

- **Polynomial Functions**

## Roots of Polynomial

| | |
|---|---|
| `roots(a)` | Determines the roots of the polynomial represented by the coefficient vector `a`. |

***Example:*** $f(x) = x^3 - 2x^2 - 3x + 10$

```
p = [1,-2,-3,10];
r = roots(p)
```

```
r = roots([1,-2,-3,10])
```

```
polyval([1,-2,-3,10],r)
```

→ Value of the polynomial at the roots

| | |
|---|---|
| `poly(r)` | Determines the coefficients of the polynomial whose roots are contained in the vector `r`. |

```
a = poly([-1,1,3]);
```

→ The result

# MATLAB Functions

- **Data Analysis Functions**

MATALB contains a number of functions to make it easy to evaluate and analyze data.

Simple Analysis:

**max(x)**

The largest value in x.

**max(x,y)**

Determine a matrix the same size as x,y. each element in the matrix contains the maximum value from the corresponding positions in x and y.

**min(x)**

The smallest value in x.

**min(x,y)**

# MATLAB Functions

- **Data Analysis Functions**

MATALB contains a number of functions to make it easy to evaluate and analyze data.

Sums and Products: **sum(x)**

The sum of the elements in x.

**prod(x)**

The product of the elements in x.

**cumsum(x)**

The cumulative sums.

**cumprod(x)**

The cumulative products.

# MATLAB Functions

- **Data Analysis Functions**

MATALB contains a number of functions to make it easy to evaluate and analyze data.

Median:

**mean(x)**

The mean value of x.

**median(x)**

The median value of x.

Sorting:

**sort(x)**

Returns vector with the values of x in ascending order. If x is a matrix, this function returns a matrix with each column in ascending order.

# MATLAB Functions

- **Data Analysis Functions**

MATALB contains a number of functions to make it easy to evaluate and analyze data.

Variance and Standard Deviation:

$$\sigma^2 = \frac{\sum_{k=1}^{N}(x_k - \mu)^2}{N-1}$$

The standard deviation is the square root of the variance.

**std(x)**

Computes the standard deviation of the values in x.

# MATLAB Functions

- **Data Analysis Functions**

MATALB contains a number of functions to make it easy to evaluate and analyze data.

Histograms: A histogram is a special type of graph that is particularly relevant to the statistical measurements and shows the distribution of a set of values.

**hist(x)**

Generates a histogram of the values in x using 10 bins.

**hist(x,n)**

Generates a histogram of the values in x using n bins.

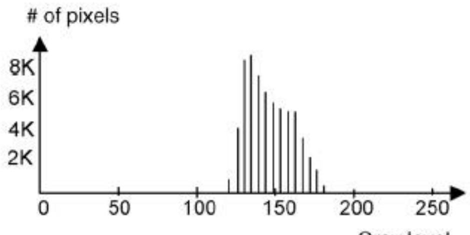
# MATLAB Functions

- **Data Analysis Functions**

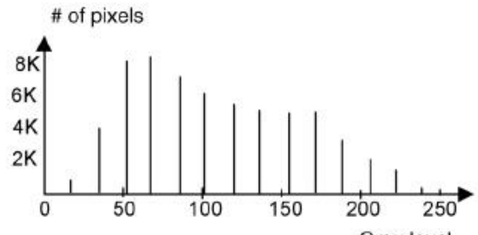
A histogram is a representation of the total number of pixels of an image at each gray level.

(a)

(b)




(c)




(d)

**hist(pixels,16)**

The Actual Grayness Values and # of Pixels for Images in Figure (a) and (b).

| Levels | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # of Pixels | 0 | 750 | 5223 | 8147 | 8584 | 7769 | 6419 | 5839 | 5392 | 5179 | 5185 | 3451 | 2078 | 1692 | 341 | 0 |
| For (b) | 0 | 17 | 34 | 51 | 68 | 85 | 102 | 119 | 136 | 153 | 170 | 187 | 204 | 221 | 238 | 256 |
| For (a) | 120 | 124 | 128 | 132 | 136 | 140 | 144 | 148 | 152 | 156 | 160 | 164 | 168 | 172 | 176 | 180 |

- **For more info about MATLAB programming**

http://www.alaakhamis.org/teaching/BSE122/resources.html