CUFE, M. Sc., 2015-2016

Computers & Numerical Analysis (STR 681)

Lecture 4
Introduction to Programming

Dr. Maha Moddather
Structural Engineering Department
Faculty of Engineering – Cairo University
mahamoddather@eng.cu.edu.eg

Spring 2016

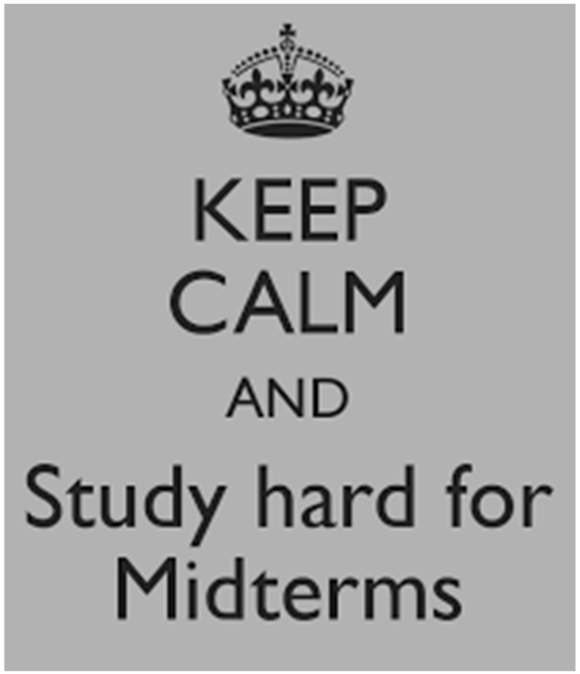
Assignment Policy

Copying the assignment (PARTS OR WHOLE) will result in losing the whole Grade.

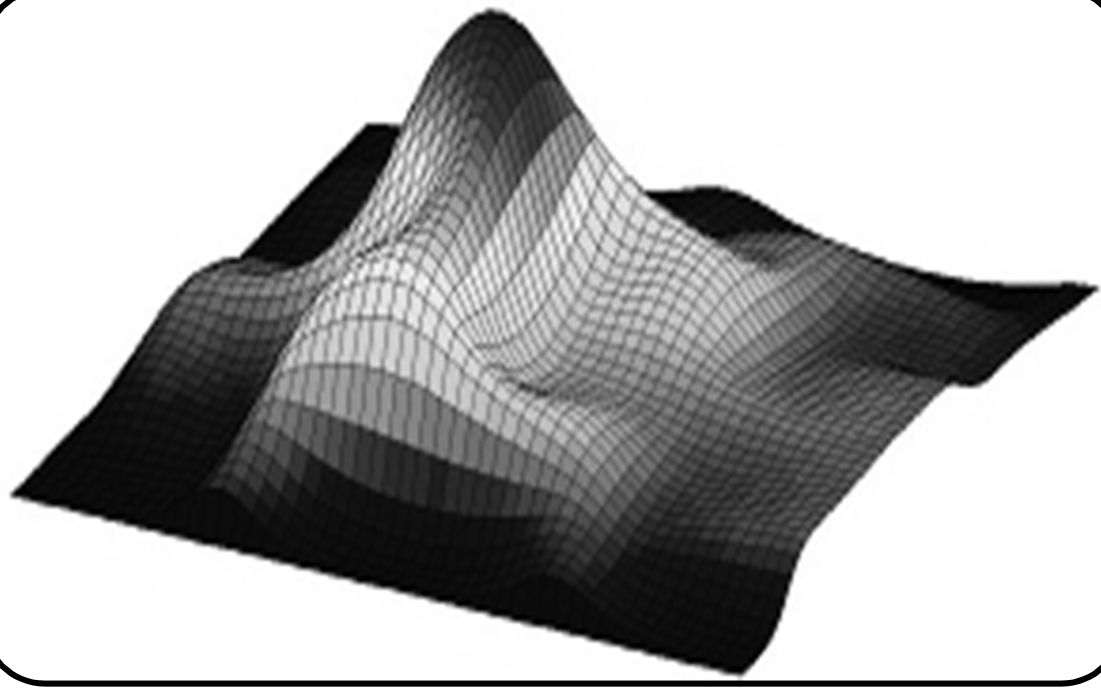
Midterm Exam

Tuesday 5-4-2016

7:30 - 8:30 p.m.



MatLAB Program



Cramer's Rule

$$[A]\{X\} = \{B\}$$

where $[A]$ is the coefficient matrix:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

Cramer's Rule

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D}$$

Cramer's Rule

$$0.3x_1 + 0.52x_2 + x_3 = -0.01$$
$$0.5x_1 + x_2 + 1.9x_3 = 0.67$$
$$0.1x_1 + 0.3x_2 + 0.5x_3 = -0.44$$

$$D = \begin{vmatrix} 0.3 & 0.52 & 1 \\ 0.5 & 1 & 1.9 \\ 0.1 & 0.3 & 0.5 \end{vmatrix} = 0.3(-0.07) - 0.52(0.06) + 1(0.05) = -0.0022$$

Cramer's Rule

$$x_1 = \frac{\begin{vmatrix} -0.01 & 0.52 & 1 \\ 0.67 & 1 & 1.9 \\ -0.44 & 0.3 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.03278}{-0.0022} = -14.9$$

$$x_2 = \frac{\begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.0649}{-0.0022} = -29.5$$

$$x_3 = \frac{\begin{vmatrix} 0.3 & 0.52 & -0.01 \\ 0.5 & 1 & 0.67 \\ 0.1 & 0.3 & -0.44 \end{vmatrix}}{-0.0022} = \frac{-0.04356}{-0.0022} = 19.8$$

Cramer's Rule

```
A=[0.3,0.52,1;0.5,1,1.9;0.1,0.3,0.5];
B=[-0.01;0.67;-0.44];


D=det(A);


Column_1=[A(1,1);A(2,1);A(3,1)];
Column_2=[A(1,2);A(2,2);A(3,2)];
Column_3=[A(1,3);A(2,3);A(3,3)];
```

Cramer’s Rule

```
M_1=[B,Column_2,Column_3];
X1=det(M_1)/D;


M_2=[Column_1,B,Column_3];
X2=det(M_2)/D;


M_3=[Column_1,Column_2,B];
X3=det(M_3)/D;
```

Cramer’s Rule

```
disp(['X1=',num2str(X1)]);
disp(['X2=',num2str(X2)]);
disp(['X3=',num2str(X3)]);
```

Display Output

Command: disp

Display value of variable

```
>> x=2;
>> disp(x)
     2
```

Defines variable x

Displays the value of variable x

Drawbacks of the Program

- Only for one problem with single values of matrix A and vector B.
- Only for one problem size (3 equations and 3 unknowns).
- Does test if the equations does not have a solution.

Bonus Grades

2 Grades are bonus for the first two students writing a MatLAB code that can solve the 3 problems in Cramer's Rule Code.

Matrix Operations

INDEXING:

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

a =

     1     2
     3     4
```

```
>> a(1,1)

ans =

     1
```

Row No.

```
>> a(2,1)

ans =

     3
```

Column No.

Matrix Operations

INDEXING:

```
>> A=[2,3,5;6,7,8]

A =

     2     3     5
     6     7     8

>> x=A(:,2)

x =

     3
     7
```

All elements in second Column

Indexing for a certain range

```
>> A(1:2,2:3)

ans =

     3     5
     7     8
```

Introduction

Command: ones

Create an array of ones

```
>> ones(3)

ans =

     1     1     1
     1     1     1
     1     1     1
```

Square Matrix 3*3

No. of Rows

```
>> ones(3,2)

ans =

     1     1
     1     1
     1     1
```

No. of Columns

Introduction

Command: ones

Create an array of ones

```
A =

     1     2
     3     4

>> ones(size(A))

ans =

     1     1
     1     1
```

Return an Array of ones same size of array (A)

Introduction

Command: zeros

Create an array of zeros

```
>> zeros(3)

ans =

     0     0     0
     0     0     0
     0     0     0
```

Square Matrix 3*3

No. of Rows

```
>> zeros(3,2)

ans =

     0     0
     0     0
     0     0
```

No. of Columns

Introduction

Command: zeros

Create an array of zeros

```
A =

     1     2
     3     4

>> zeros(size(A))

ans =

     0     0
     0     0
```

Return an Array of ones same size of array (A)

Introduction

```
>> F=zeros(2)

F =

     0     0
     0     0

>> F(2,1)=5

F =

     0     0
     5     0
```

Assigns value to a certain location in an array

Introduction

Command: linspace

Generate linearly spaced vector

```
>> linspace(1,2)
```

Return a row vector of 100 evenly spaced points between 1 and 2

```
ans =

  Columns 1 through 6

    1.0000    1.0101    1.0202    1.0303    1.0404    1.0505

  Columns 7 through 12

    1.0606    1.0707    1.0808    1.0909    1.1010    1.1111
```

Introduction

Command: linspace

Generate linearly spaced vector

```
>>  linspace(1,2,10)

ans =

  Columns 1 through 6

    1.0000    1.1111    1.2222    1.3333    1.4444    1.5556

  Columns 7 through 10

    1.6667    1.7778    1.8889    2.0000
```

Return a row vector of n evenly spaced points between 1 and 2, where n=10

Introduction

Command: norm

Returns norm of matrix or vector

```
>> A=[2,3;4,5]

A =

    2     3
    4     5

>> norm(A)

ans =

   7.3434
```

Second Norm

Measure of the magnitude of the matrix

Matrix Norm

The norm of a square matrix A is a non-negative real number denoted $\|A\|$. There are several different ways of defining a matrix norm, but they all share the following properties:

1. $\|A\| \geq 0$ for any square matrix A.

2. $\|A\| = 0$ *if and only if* the matrix $A = 0$.

3. $\|kA\| = |k| \; \|A\|$, for any scalar k.

4. $\|A + B\| \leq \|A\| + \|B\|$.

5. $\|AB\| \leq \|A\| \; \|B\|$.

The norm of a matrix is a measure of how large its elements are. It is a way of determining the "size" of a matrix that is not necessarily related to how many rows or columns the matrix has.

Matrix Norm

1st Norm:

Calculate the 1-norm of A= $\begin{bmatrix} 1 & -7 \\ -2 & -3 \end{bmatrix}$.

$$\|A\|_1 = \max_{1 \le j \le n} \left(\sum_{i=1}^{n} |a_{ij}| \right)$$

First Column = 1+|-2| = 3
Second Column = |-7|+|-3| =10

Matrix First Norm = 10

Matrix Norm

2nd Norm:

Calculate the 2-norm of $A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{Ax}\|_2 = \sqrt{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest number λ such that $\mathbf{A}^*\mathbf{A} - \lambda\mathbf{I}$ is singular.

Matrix Norm

2nd Norm:

Calculate the 2-norm of $A = \begin{bmatrix} 1 & 1 \\ 2 & 1 \end{bmatrix}$

$$A^*A = A^TA = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}.$$

The Eigen values of the matrix are 6.8541 and 0.1459.

The second norm is the root of the largest Eigen value = $\sqrt{6.8541}$ = 2.618

LU Decomposition using MatLAB

$$[A] = \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.1 & 7 & -0.3 \\ 0.3 & -0.2 & 10 \end{bmatrix}$$

$$[A] = [L][U]$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0.0333333 & 1 & 0 \\ 0.100000 & -0.0271300 & 1 \end{bmatrix} \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0 & 7.00333 & -0.293333 \\ 0 & 0 & 10.0120 \end{bmatrix}$$

$$= \begin{bmatrix} 3 & -0.1 & -0.2 \\ 0.0999999 & 7 & -0.3 \\ 0.3 & -0.2 & 9.99996 \end{bmatrix}$$

LU Decomposition using MatLAB

```
>> A=[3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10]

A =

    3.0000   -0.1000   -0.2000
    0.1000    7.0000   -0.3000
    0.3000   -0.2000   10.0000
```

LU Decomposition using MatLAB

```
>> A=[3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10]

A =

    3.0000   -0.1000   -0.2000
    0.1000    7.0000   -0.3000
    0.3000   -0.2000   10.0000
```

```
>> [L,U]=lu(A)

L =

    1.0000         0         0
    0.0333    1.0000         0
    0.1000   -0.0271    1.0000


U =

    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0         0   10.0120
```

LU Decomposition using MatLAB

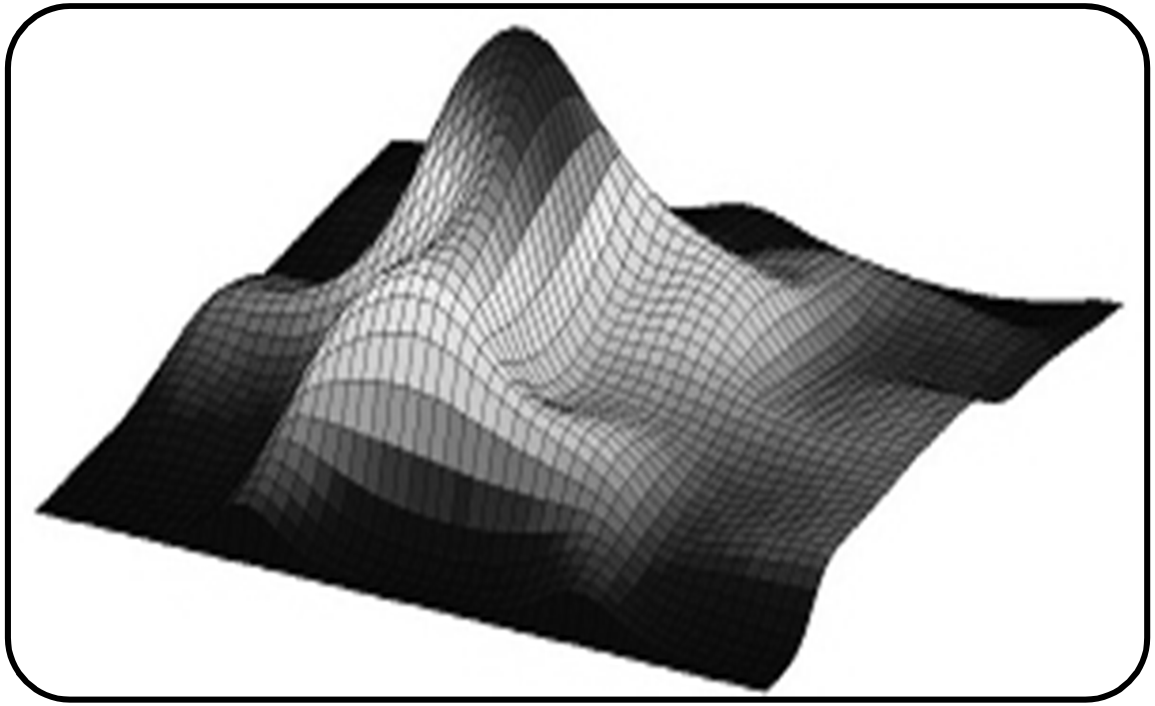
```
>> A=[3,-0.1,-0.2;0.1,7,-0.3;0.3,-0.2,10]

A =

    3.0000   -0.1000   -0.2000
    0.1000    7.0000   -0.3000
    0.3000   -0.2000   10.0000
```

```
>> [L,U]=lu(A)

L =

    1.0000         0         0
    0.0333    1.0000         0
    0.1000   -0.0271    1.0000


U =

    3.0000   -0.1000   -0.2000
         0    7.0033   -0.2933
         0         0   10.0120
```

Compare with manually calculated results

MatLAB Program

“Plotting”

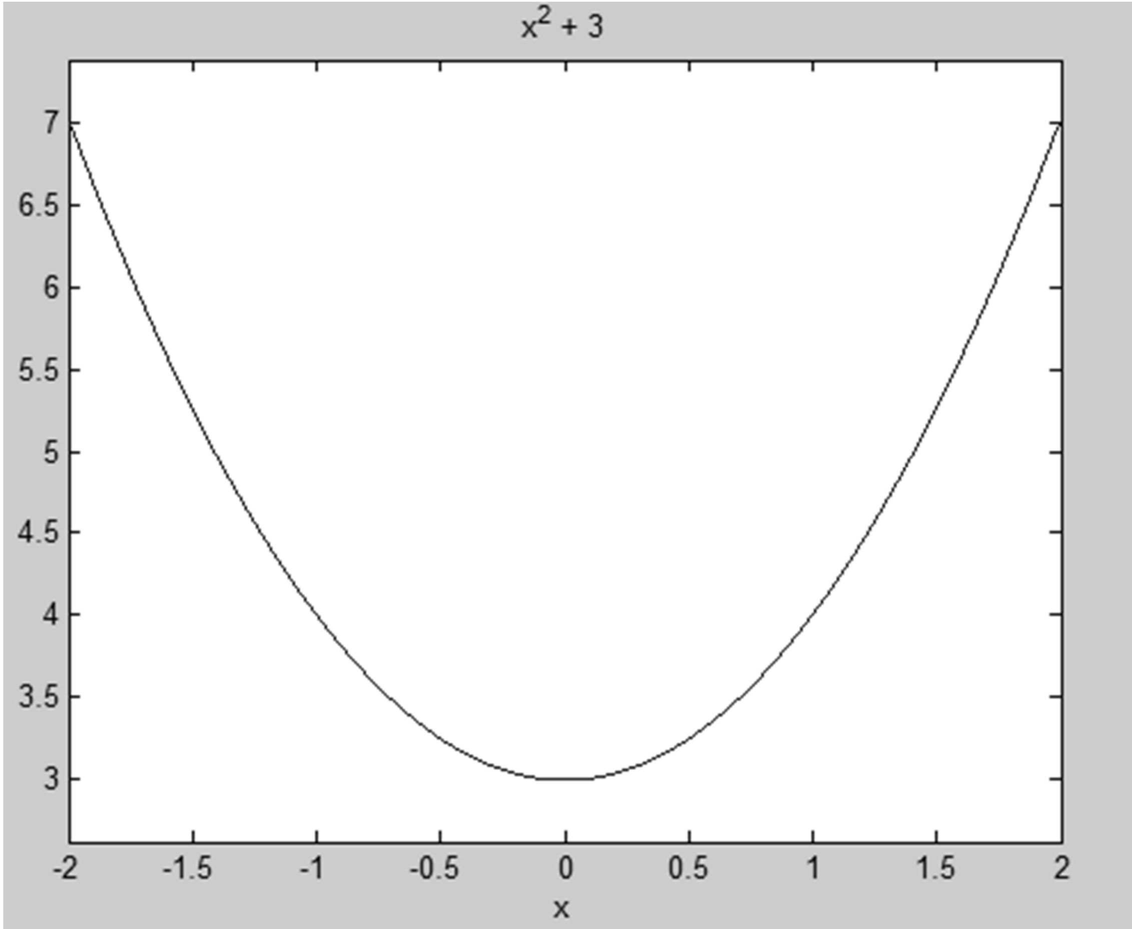


Plotting

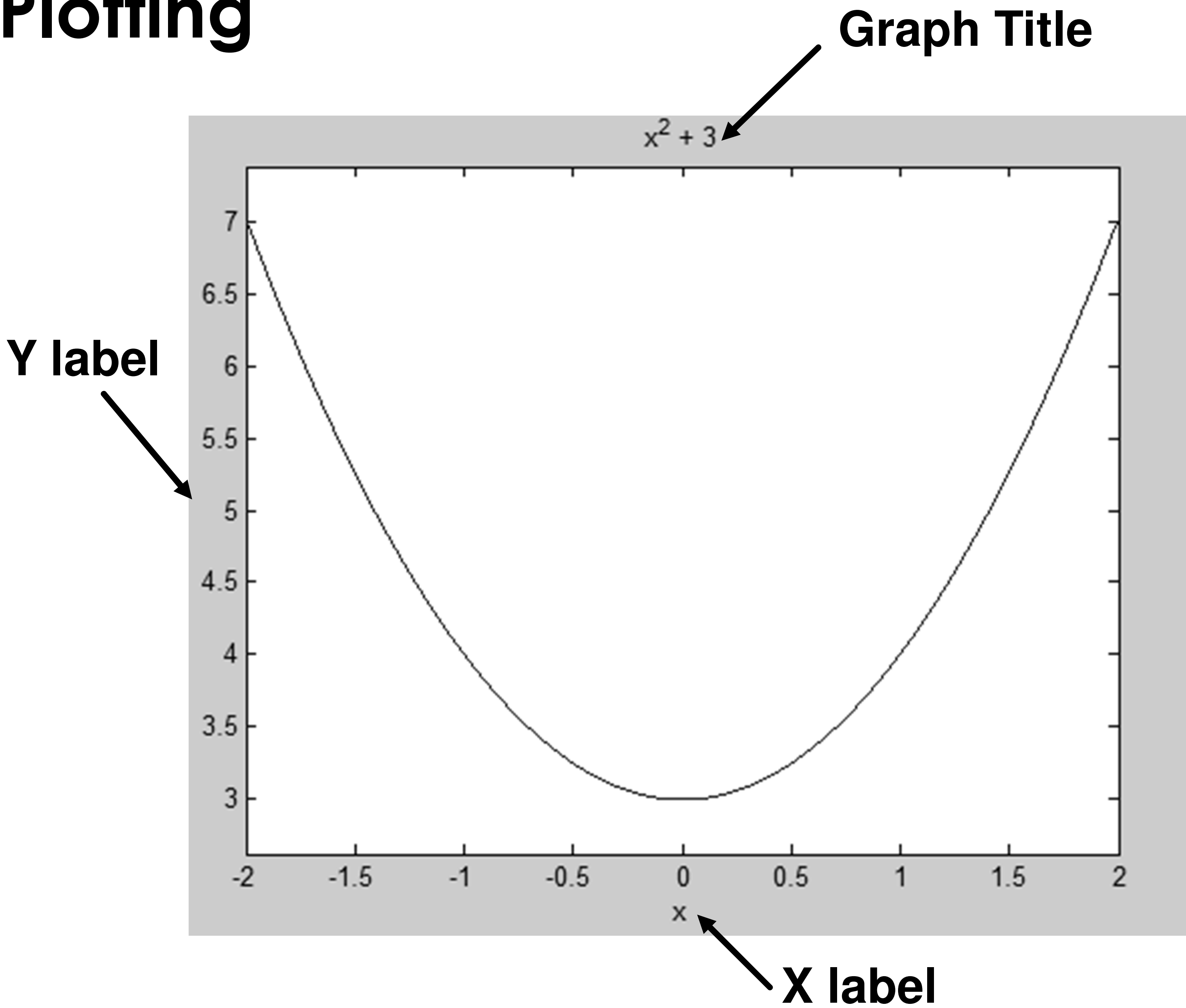
Command: ezplot

Plots expression or function

```
>> syms x y
>> y=x^2+3

y =

x^2 + 3

>> ezplot(y,[-2 2])
```



Plotting



Graph Title
$x^2 + 3$
Y label
X label
x
7
6.5
6
5.5
5
4.5
4
3.5
3
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Plotting

To change Graph title:

>>title ‘Y-Function’

To add X-Label and Y-Label:

>>xlabel ‘X’

>>ylabel ‘Y’

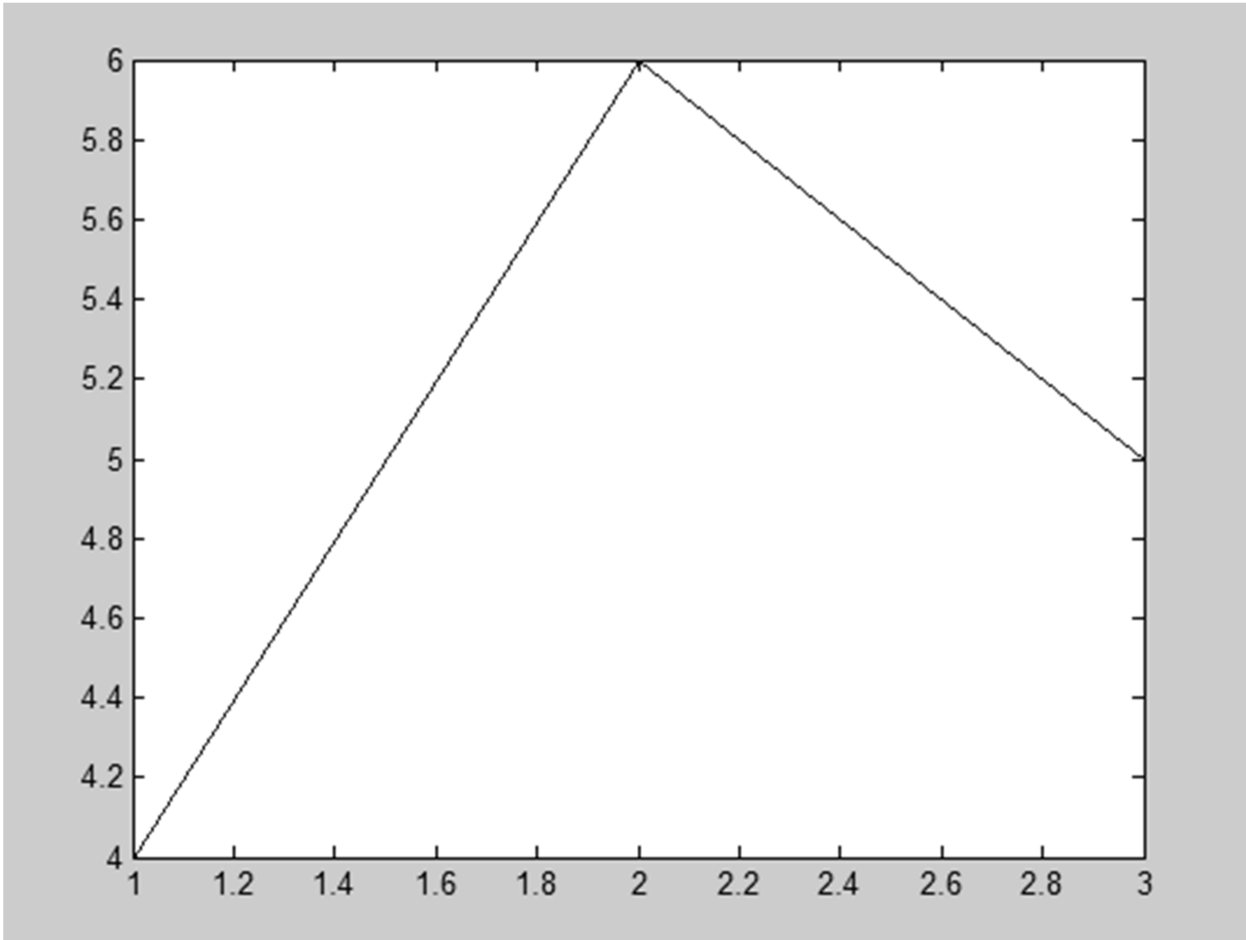
To change the range of coordinates:

$$\texttt{>> axis ([}\underbrace{-2\ 2}_{\text{Range of Hz. axis}}\ \ \underbrace{1\ 4}_{\text{Range of Vl axis}}\texttt{]}$$

Plotting

Command: plot : 2-D line plot

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

x =

     1     2     3

>> y=[4 6 5]

y =

     4     6     5

>> plot(x,y)
```



Plotting

Command: plot : 2-D line plot

```
>> x=-2:0.1:2

x =

  Columns 1 through 6

   -2.0000   -1.9000   -1.8000   -1.7000   -1.6000   -1.5000

 Columns 37 through 41

    1.6000    1.7000    1.8000    1.9000    2.0000
```

Plotting

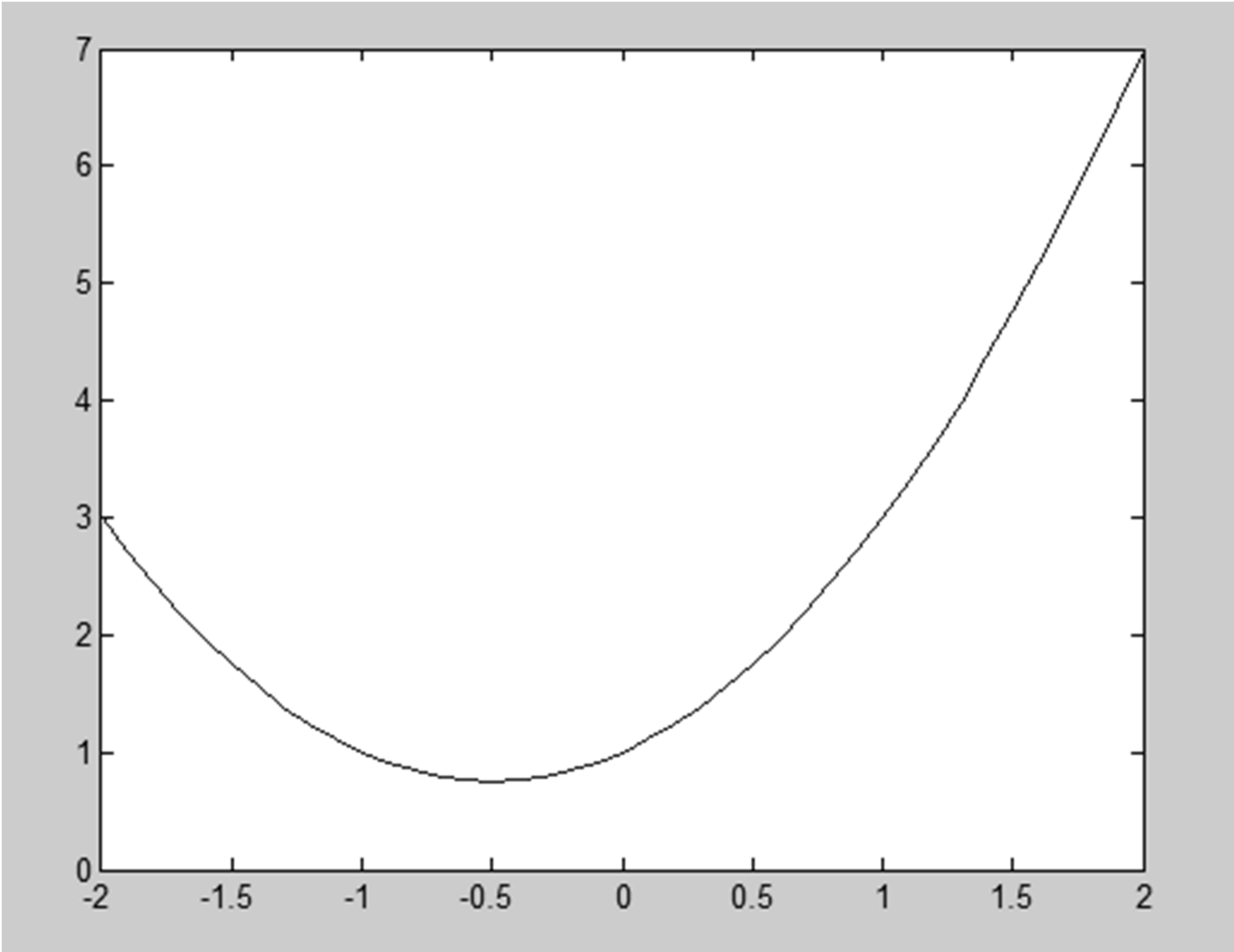
Command: plot : 2-D line plot

```
>> x=-2:0.1:2

x =

  Columns 1 through 6

   -2.0000   -1.9000   -1.8000   -1.7000   -1.6000   -1.5000


  Columns 37 through 41

    1.6000    1.7000    1.8000    1.9000    2.0000
```

```
>> plot(x,x.^2+x+1)
```

Plotting

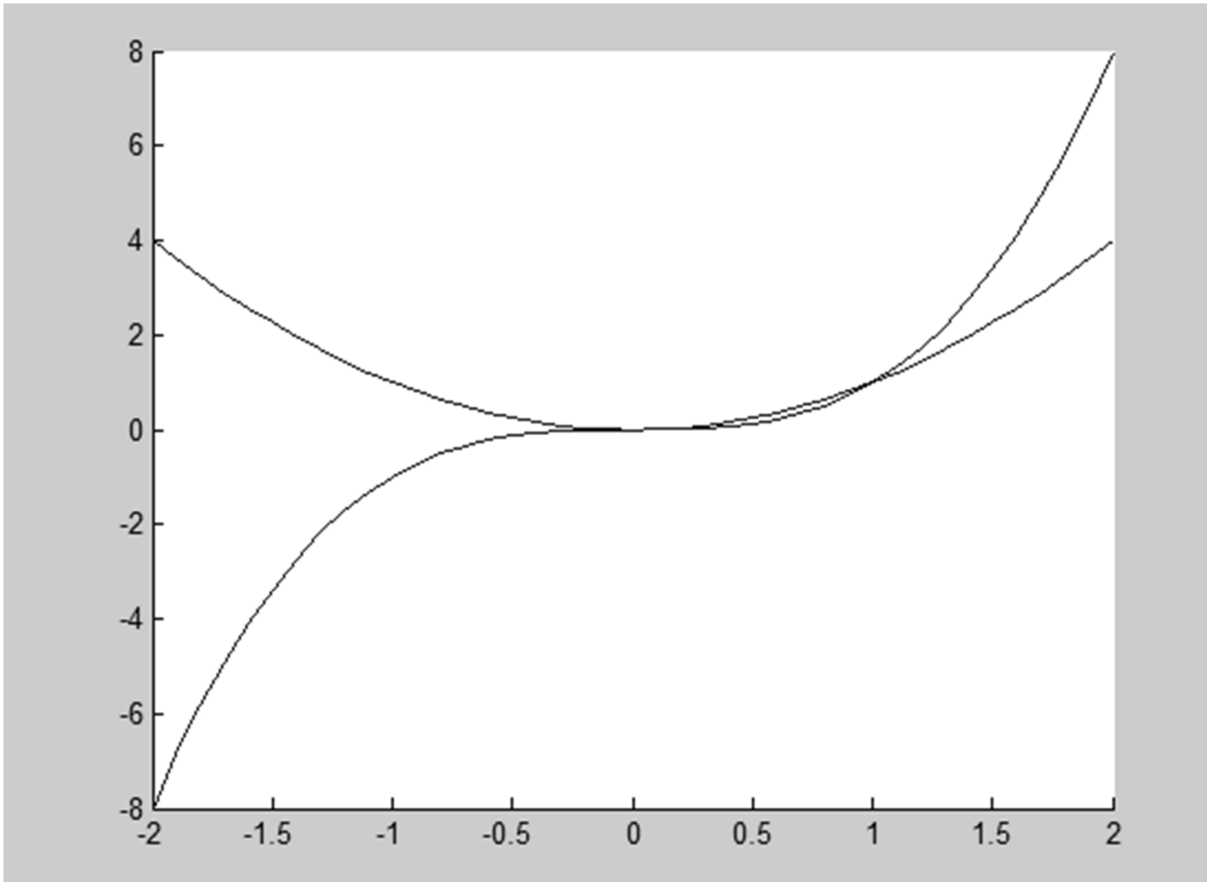
Command: plot : 2-D line plot



Plotting

To plot several curves on the same figure use "hold on"

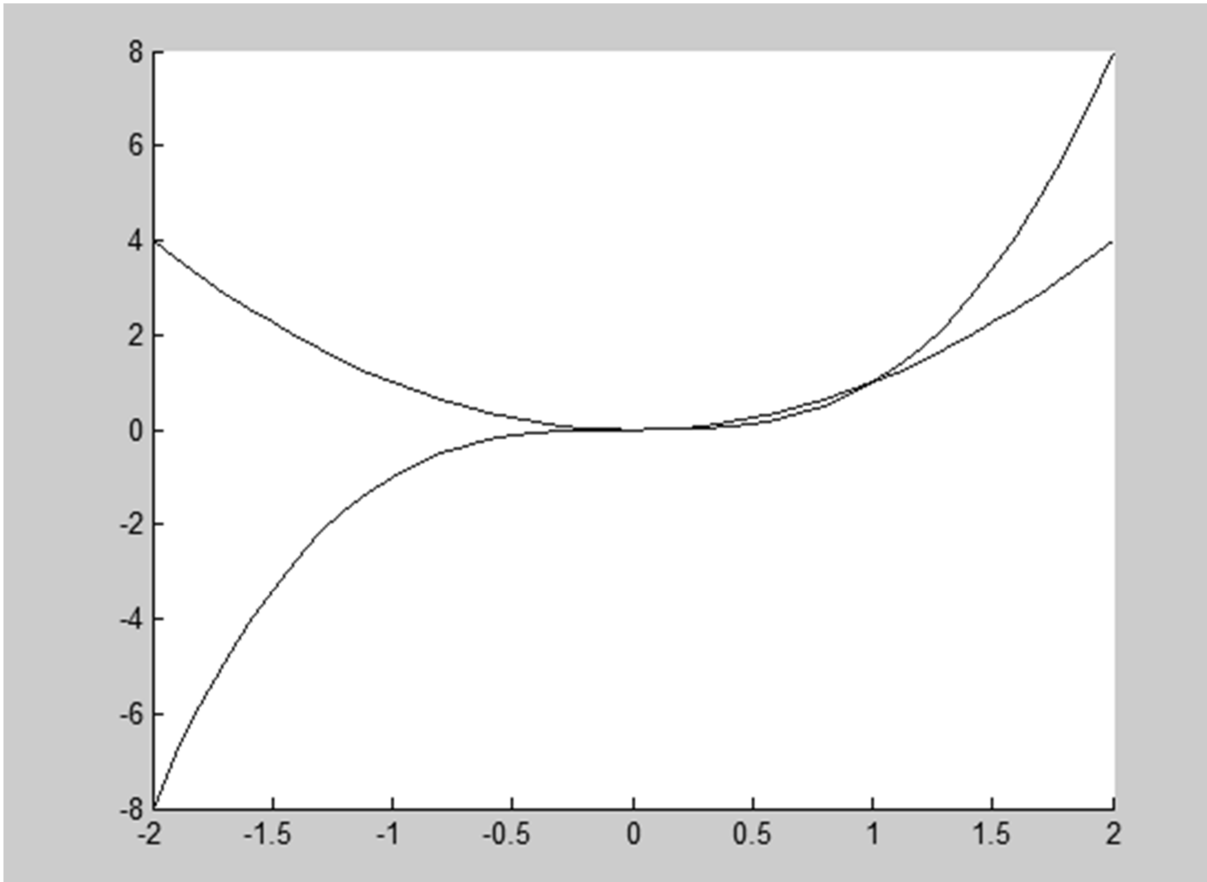
```
>> hold on
>> plot(x,x.^2)
>> plot(x,x.^3)
```



Plotting

To plot several curves on the same figure use "hold on"

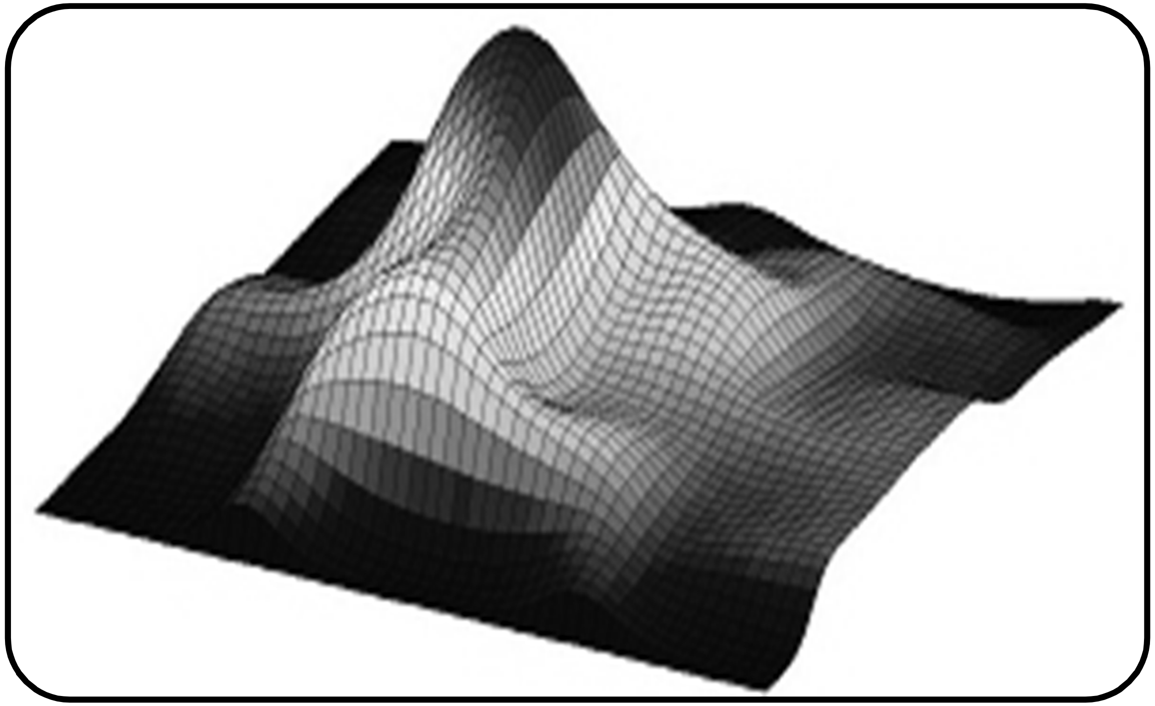
```
>> hold on
>> plot(x,x.^2)
>> plot(x,x.^3)
```



Use "hold off" to redraw a new group.

MatLAB Program

“Loops & Flow Control”



Relational Operators

Standard Relational Operators:

Symbol	Meaning
==	Equal
~=	Not Equal
>	Greater than
<	Less than
>=	Greater or Equal
<=	Less or Equal

Relational Operators

Logical Operations:

Symbol	Meaning
&	AND
|	OR
~	NOT
all	All True
any	Any True

If Statement

if cond

Commands

end

```
>> x=3;
>> y=5;
>> if x<y
x=y
end

x =

     5
```

If Statement

if cond

Command1

else

Command2

end

```
>> x=3;
>> y=5;
>> if x<y
x=y
else
y=x
end

x =

     5
```

If Statement

```
if cond 1
Command1
elseif cond 2
Command2
else
Command3
end
```

For Statement

Looping for a known number of iterations:

for n=1:100

Command

end

```
>> x = ones(1,10);
for n = 2:6
    x(n) = 2 * x(n - 1);
end
>> x

x =

     1     2     4     8    16    32     1     1     1     1
```

While Statement

Looping till a condition is not satisfied:

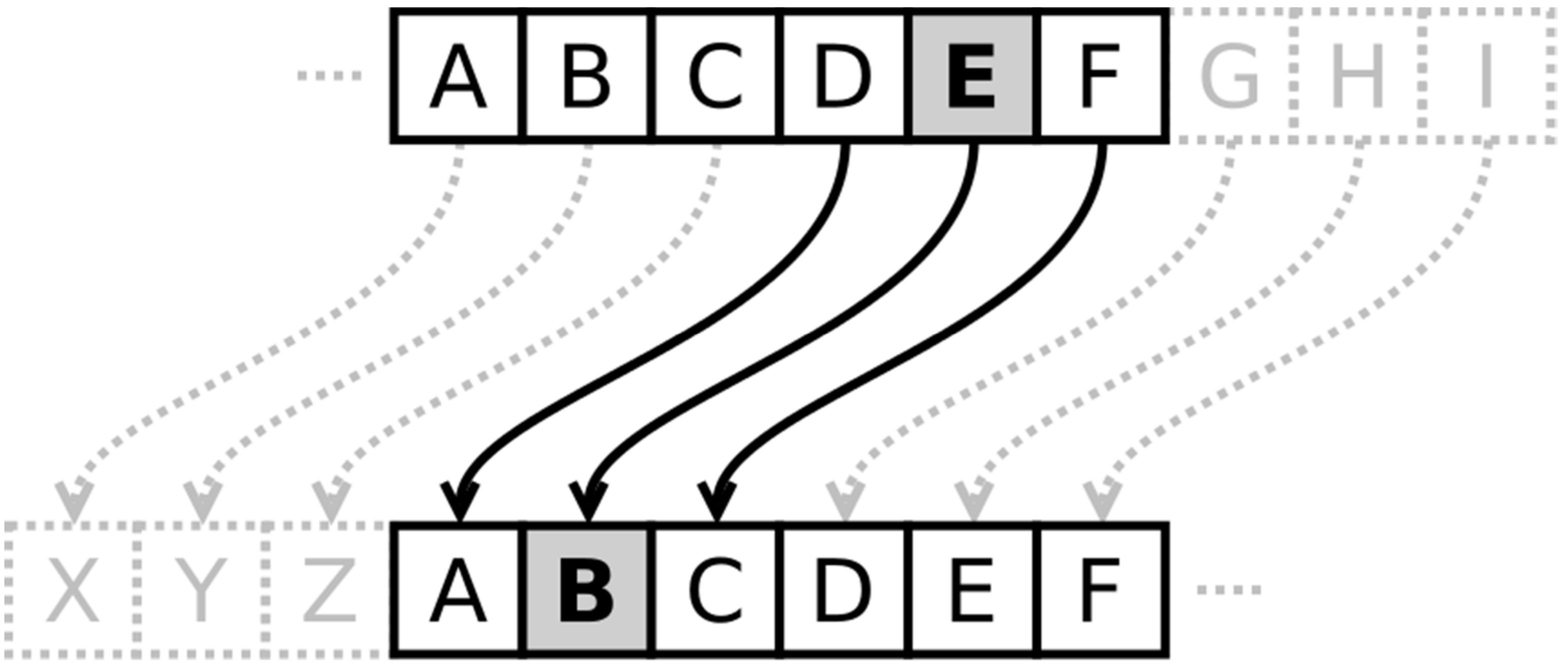
while cond

Command

end

```
>> i = 1;
s = 0;
while i < 3
  s = s + i;
  i = i + 1;
end
>> s

s =

     3
```

Caesar Cipher

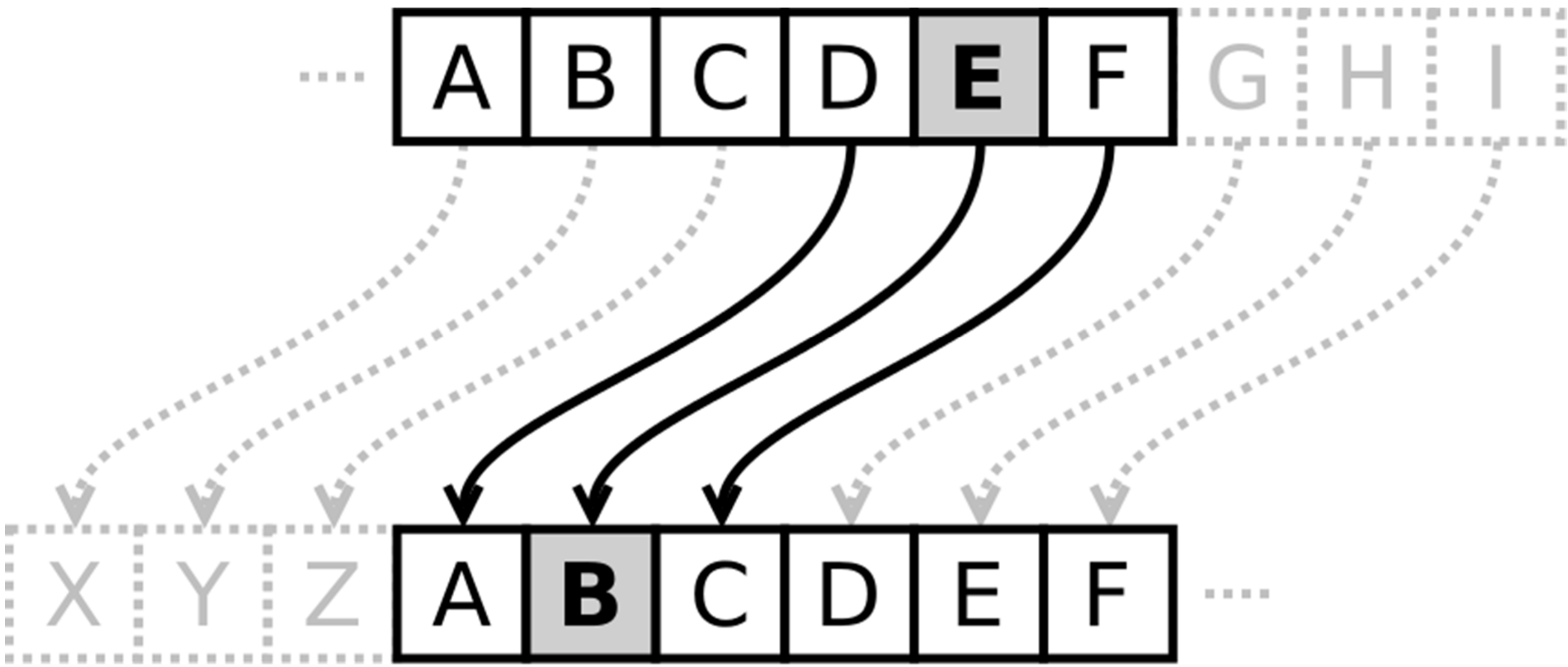


Caesar Cipher

The Caesar cipher is one of the earliest known and simplest ciphers. It is a type of substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet.

For example, with a shift of 1, A would be replaced by B, B would become C, and so on.

Caesar Cipher



Caesar Cipher

To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. For the caesar cipher, the key is the number of characters to shift the cipher alphabet.

Caesar Cipher

Shift (Key) of 1:

Plain Text:

Defend the east wall of the castle

Cipher Text:

Efgfoe uif fbtu xbmm pg uif dbtumf