Introduction to Matlab

By:Mohammad Sadeghi

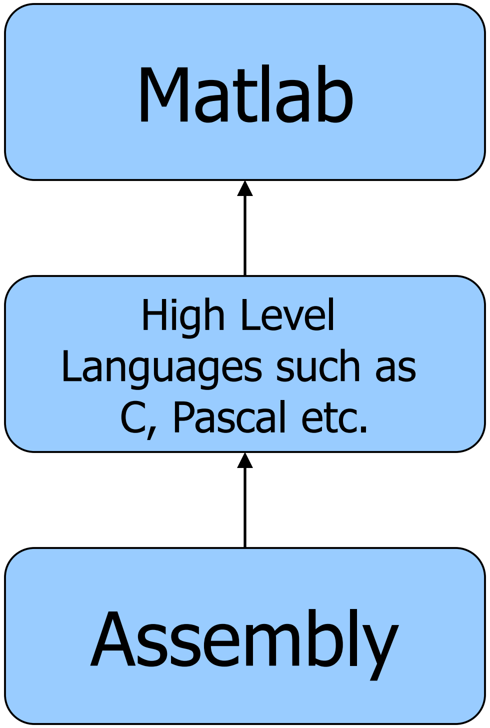
*Dr. Sajid Gul Khawaja Slides has been used partially to prepare this presentation

Outline:

- What is Matlab?
- Matlab Screen
- Basic functions
- Variables, matrix, indexing
- Operators (Arithmetic, logical)
- Basic Plotting

What is Matlab?

- Matlab is basically a high level language which has many specialized toolboxes for making things easier for us
- How high?

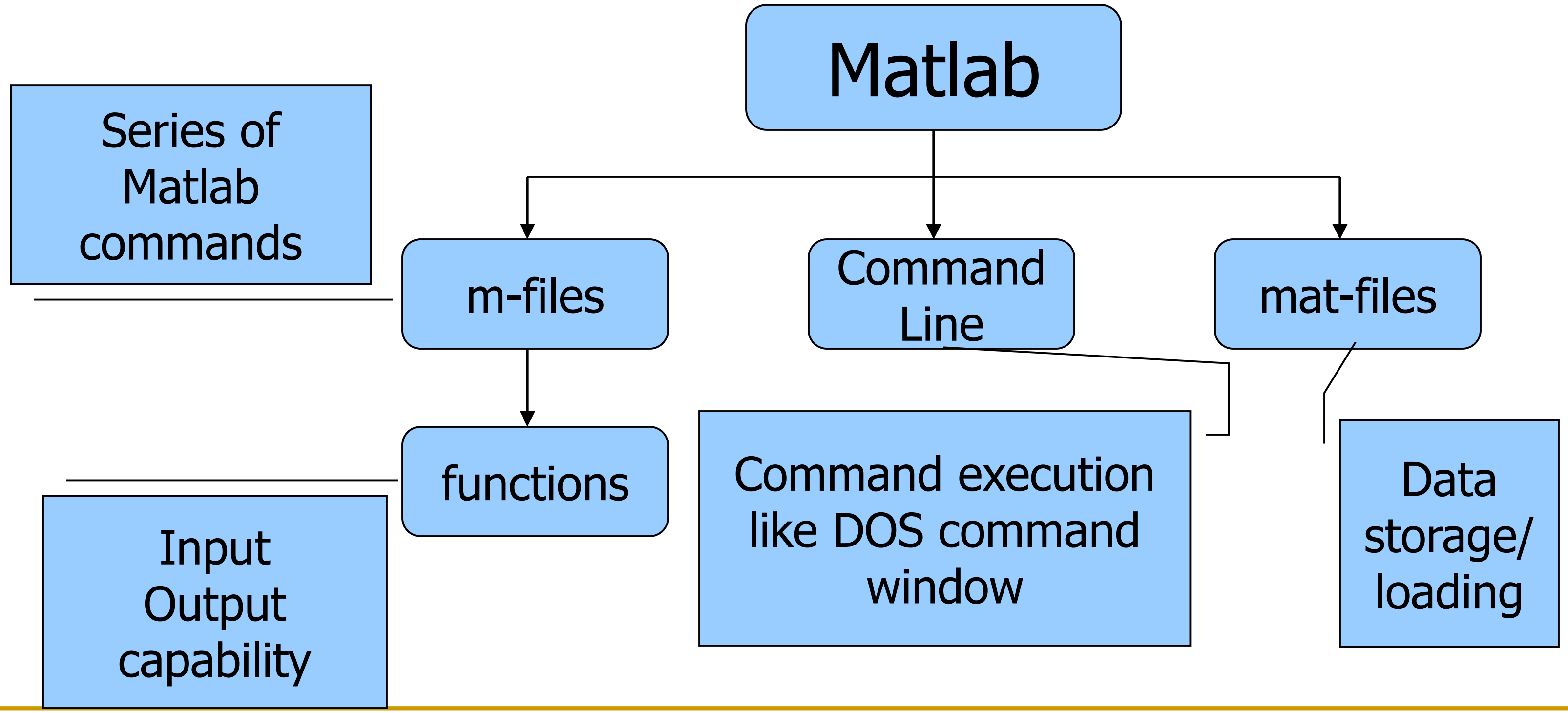


What is Matlab?

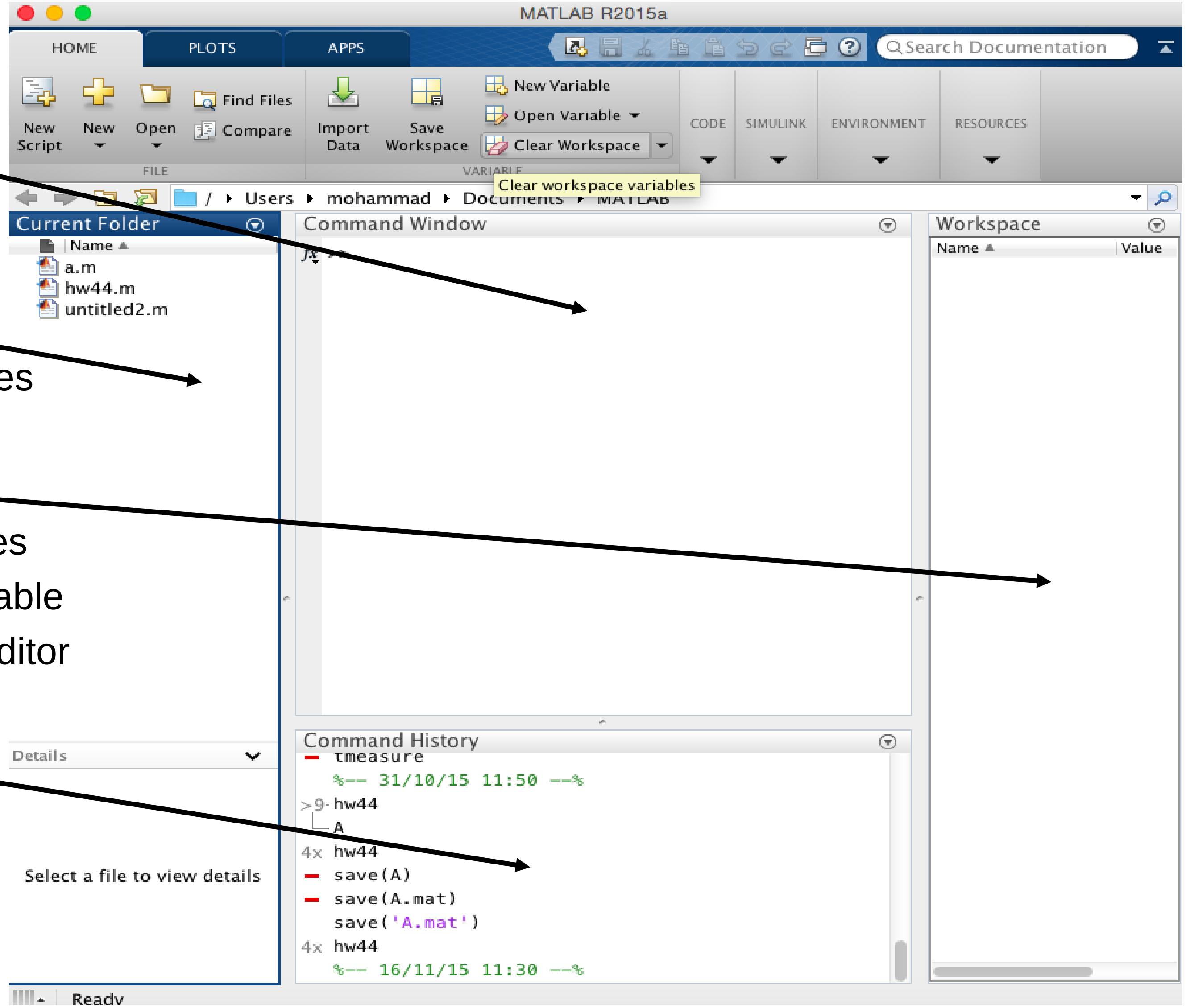
- MatLab : **Mat**rix **Lab**oratory
- Numerical Computations with matrices
  - *Every number can be represented as matrix*
- Why Matlab?
  - User Friendly (GUI)
  - Easy to work with
  - Powerful tools for complex mathematics
- Matlab has extensive demo and tutorials to learn by yourself
  - Use help command

What are we interested in?

- Matlab is too broad for our purposes in this course.
- The features we are going to require is



Matlab Screen



- Command Window
  - type commands
- Current Directory
  - View folders and m-files
- Workspace
  - View program variables
  - Double click on a variable to see it in the Array Editor
- Command History
  - view past commands
  - save a whole session using diary

Variables

- No need for types. i.e.,

```
int a;
double b;
float c;
```

- All variables are created with double precision unless specified and they are matrices.

```
Example:
>>x=5;
>>x1=2;
```

- After these statements, the variables are 1x1 matrices with double precision

Variables (con't…)

- Special variables:
  - ans : default variable name for the result
  - pi: π = 3.1415926…………
  - eps: $\in$ = 2.2204e-016, smallest amount by which 2 numbers can differ.
  - Inf or inf : ∞, infinity
  - NaN or nan: not-a-number

Elementary Math Function

- Abs(), sign()
  - Sign(A) = A./abs(A)
- Sin(), cos(), asin(), acos()
- Exp(), log(), log10()
- Ceil(), floor()
- Sqrt()
- Real(), imag()
- ^

Array, Matrix

- a vector `x = [1 2 5 1]`

```
x =
     1    2    5    1
```

- a matrix `x = [1 2 3; 5 1 4; 3 2 -1]`

```
x =
     1      2      3
     5      1      4
     3      2     -1
```

- transpose `y = x'`

```
y =
     1
     2
     5
     1
```

Long Array, Matrix

- `t =1:10`

```
t =
     1    2    3    4   5   6    7   8    9    10
```

- `k =2:-0.5:-1`

```
k =
     2   1.5   1   0.5   0   -0.5   -1
```

- `B = [1:4; 5:8]`

```
x =
     1      2      3     4
     5      6      7     8
```

Vectors (con't…)

Some useful commands:

x = start:end	create row vector x starting with start, counting by one, ending at end
x = start:increment:end	create row vector x starting with start, counting by increment, ending at or before end
linspace(start,end,number)	create row vector x starting with start, ending at end, having number elements
length(x)	returns the length of vector x
y = x'	transpose of vector x
dot (x, y)	returns the scalar dot product of the vector x and y.

Vectors (con't…)

- Vector operation:
  - Max(), min(): max/min element of a vector
  - Mean(), median()
  - Std(), var(): standard deviation and variance
  - Sum(), prod(): sum/product of elements
  - Sort(): sort in ascending order

Generating Vectors from functions

- zeros(M,N) MxN matrix of zeros

```
x = zeros(1,3)
x =
   0     0     0
```

- ones(M,N) MxN matrix of ones

```
x = ones(1,3)
x =
   1     1     1
```

- rand(M,N) MxN matrix of uniformly distributed random numbers on (0,1)

```
x = rand(1,3)
x =
 0.9501  0.2311 0.6068
```

Matrix Index

- The matrix indices begin from 1 (not 0 (as in C))
- The matrix indices must be positive integer

Given:

```
A =

     3     5     3
     6     8     2
     2     7     3
```

```
>> A(6)

ans =

     7
```

```
>> A(3,2)

ans =

     7
```

```
>> A(2,:)

ans =

     6     8     2
```

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

ans =

     5
     8
```

A(-2), A(0)

Error: ??? Subscript indices must either be real positive integers or logicals.

A(4,2)

Error: ??? Index exceeds matrix dimensions.

A(:, 2)=[]

Delete second column

Concatenation of Matrices

- `x = [1 2], y = [4 5], z=[ 0 0]`

```
A = [ x y]

    1    2    4    5

B = [x ; y]

    1 2
    4 5
```

```
C = [x y ;z]
```

Error:

??? Error using ==> vertcat CAT arguments dimensions are not consistent.

Operators (arithmetic)

+ addition

- subtraction

* multiplication

/ division

^ power

‘ complex conjugate transpose

Matrices Operations

Given A and B:

```
>> A = [1 2 3;4 5 6;7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9
```

```
>> B = [3 5 2; 5 2 8; 3 6 9]

B =

     3     5     2
     5     2     8
     3     6     9
```

Addition

```
>> X = A + B

X =

     4     7     5
     9     7    14
    10    14    18
```

Subtraction

```
>> Y = A - B

Y =

    -2    -3     1
    -1     3    -2
     4     2     0
```

Product

```
>> Z = A * B

Z =

    22    27    45
    55    66   102
    88   105   159
```

Transpose

```
>> T = A'

T =

     1     4     7
     2     5     8
     3     6     9
```

Matrices (con't…)

more commands

Transpose	B = A′
Identity Matrix	eye(n) ➔ returns an n x n identity matrix eye(m,n) ➔ returns an m x n matrix with ones on the main diagonal and zeros elsewhere.
Addition and subtraction	C = A + B C = A – B
Scalar Multiplication	B = αA, where α is a scalar.
Matrix Multiplication	C = A*B
Matrix Inverse	B = inv(A), A must be a square matrix in this case. rank (A) ➔ returns the rank of the matrix A.
Matrix Powers	B = A.^2 ➔ squares each element in the matrix C = A * A ➔ computes A*A, and A must be a square matrix.
Determinant	det (A), and A must be a square matrix.

A, B, C are matrices, and m, n, α are scalars.

Operators (Element by Element)

.* element-by-element multiplication

./ element-by-element division

.^ element-by-element power

The use of "." – "Element" Operation

```
A = [1 2 3; 5 1 4; 3 2 1]
    A =
            1     2     3
            5     1     4
            3     2    -1
```

```
x = A(1,:)

x=
     1   2   3
```

```
y = A(3 ,:)

y=
    3  4  -1
```

```
b = x .* y

b=
     3  8 -3
```

```
c = x . / y

c=
   0.33   0.5   -3
```

```
d = x .^2

d=
     1    4    9
```

```
K= x^2
Erorr:
 ??? Error using ==> mpower  Matrix must be square.
B=x*y
Erorr:
??? Error using ==> mtimes Inner matrix dimensions must agree.
```

Solutions to Systems of Linear Equations

- Example: a system of 3 linear equations with 3 unknowns (x_1, x_2, x_3):

$$\begin{aligned} 3x_1 + 2x_2 - x_3 &= 10 \\ -x_1 + 3x_2 + 2x_3 &= 5 \\ x_1 - x_2 - x_3 &= -1 \end{aligned}$$

Let :

$$A = \begin{bmatrix} 3 & 2 & 1 \\ -1 & 3 & 2 \\ 1 & -1 & -1 \end{bmatrix} \qquad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad b = \begin{bmatrix} 10 \\ 5 \\ -1 \end{bmatrix}$$

Then, the system can be described as:

$$Ax = b$$

Integral and derivative

- int(-2*x/(1 + x^2)^2,x)
- int(-2*x/(1 + x^2)^2,x,2,4)

- quad(@(x)x.^5.*exp(-x).*sin(x),2,4)

- Diff(-2*x/(1 + x^2)^2,x)
- Diff(-2*x/(1 + x^2)^2,x,2,4)

Solve equations

- solve(@(x)sin(x)==1,x)

- syms u v
- [solv, solu] = solve([2*u^2 + v^2 == 0, u - v == 1], [v, u])

Solutions to Systems of Linear Equations (con't…)

- Solution by Matrix Inverse:
  $Ax = b$
  $A^{-1}Ax = A^{-1}b$
  $x = A^{-1}b$
- MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = inv(A)*b
x =
   -2.0000
    5.0000
   -6.0000
```

Answer:
$x_1 = -2$, $x_2 = 5$, $x_3 = -6$

- Solution by Matrix Division:
  The solution to the equation
  $$Ax = b$$
  can be computed using left division.
- MATLAB:

```
>> A = [ 3 2 -1; -1 3 2; 1 -1 -1];
>> b = [ 10; 5; -1];
>> x = A\b
x =
   -2.0000
    5.0000
   -6.0000
```

Answer:
$x_1 = -2$, $x_2 = 5$, $x_3 = -6$

NOTE:
left division: A\b ➔ $b \div A$

right division: x/y ➔ $x \div y$

Save/Load Data

- Save fname
  - Save all workspace data into fname.mat
  - Save fname x y z
  - Save(fname): when fname is a variable
- Load fname
  - Load(fname)

Operators (relational, logical)

- == Equal to
- ~= Not equal to
- < Strictly smaller
- > Strictly greater
- <= Smaller than or equal to
- >= Greater than equal to
- & And operator
- | Or operator

Basic Task: Plot the function sin(x) between $0 \le x \le 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

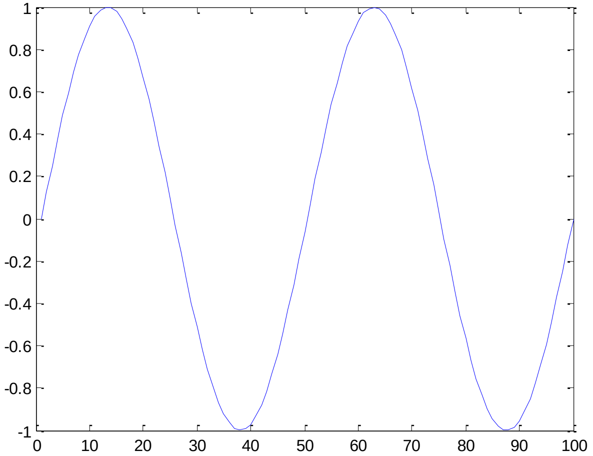
```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Plot the y-array

```
>>plot(y)
```



Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

- Create an x-array of 100 samples between 0 and 4π.

```
>>x=linspace(0,4*pi,100);
```

- Calculate sin(.) of the x-array

```
>>y=sin(x);
```

- Calculate $e^{-x/3}$ of the x-array

```
>>y1=exp(-x/3);
```

- Multiply the arrays y and y1

```
>>y2=y*y1;
```

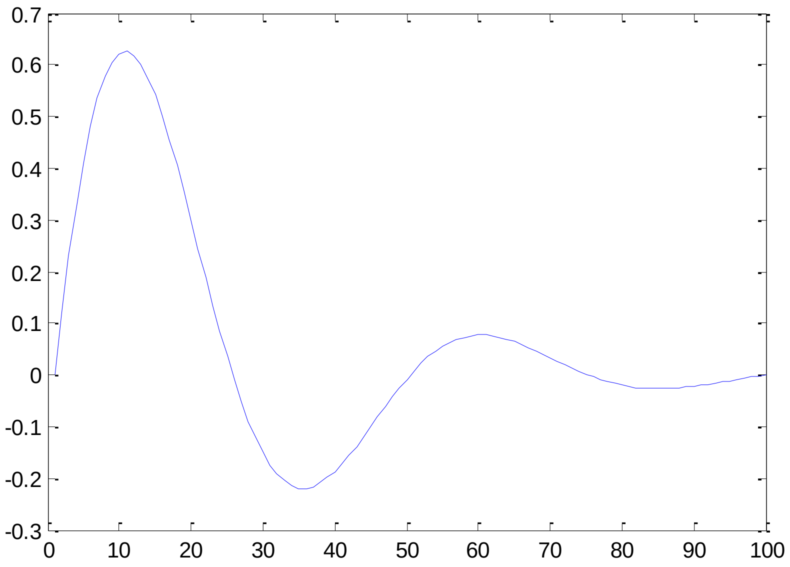
Plot the function $e^{-x/3}\sin(x)$ between $0 \le x \le 4\pi$

- Multiply the arrays y and y1 correctly

```
>>y2=y.*y1;
```

- Plot the y2-array

```
>>plot(y2)
```



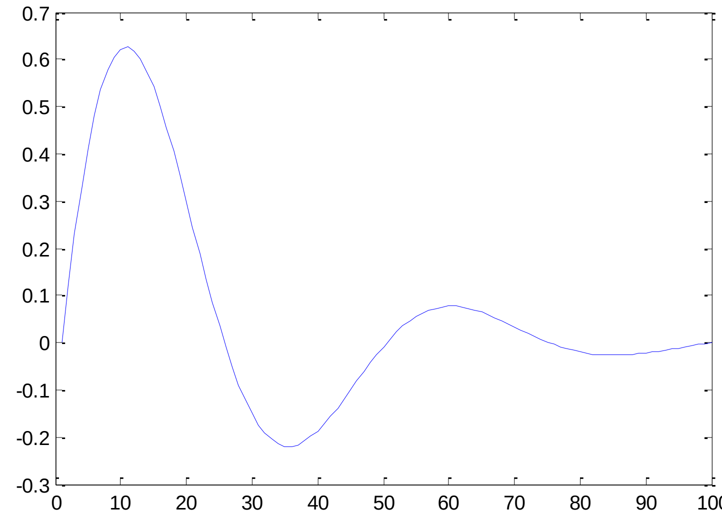
Display Facilities

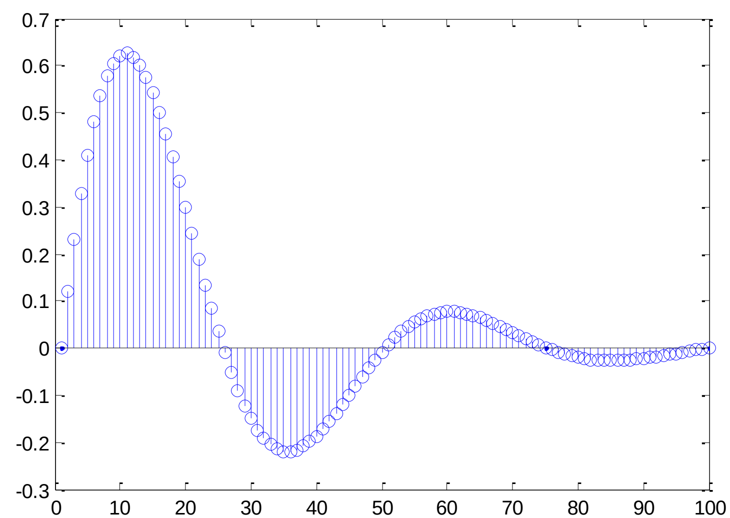
- plot(.)

```
Example:
>>x=linspace(0,4*pi,100);
>>y=sin(x);
>>plot(y)
>>plot(x,y)
```

- stem(.)

```
Example:
>>stem(y)
>>stem(x,y)
```





Plotting function

- Plot(X, Y):
  - Plots vector Y versus vector X
- Hold: next plot action on the same figure
- Title('title text here')
- Xlabel('…'), ylabel('…')
- Axis([XMIN XMAX YMIN YMAX])
- Legend('…')
- Grid

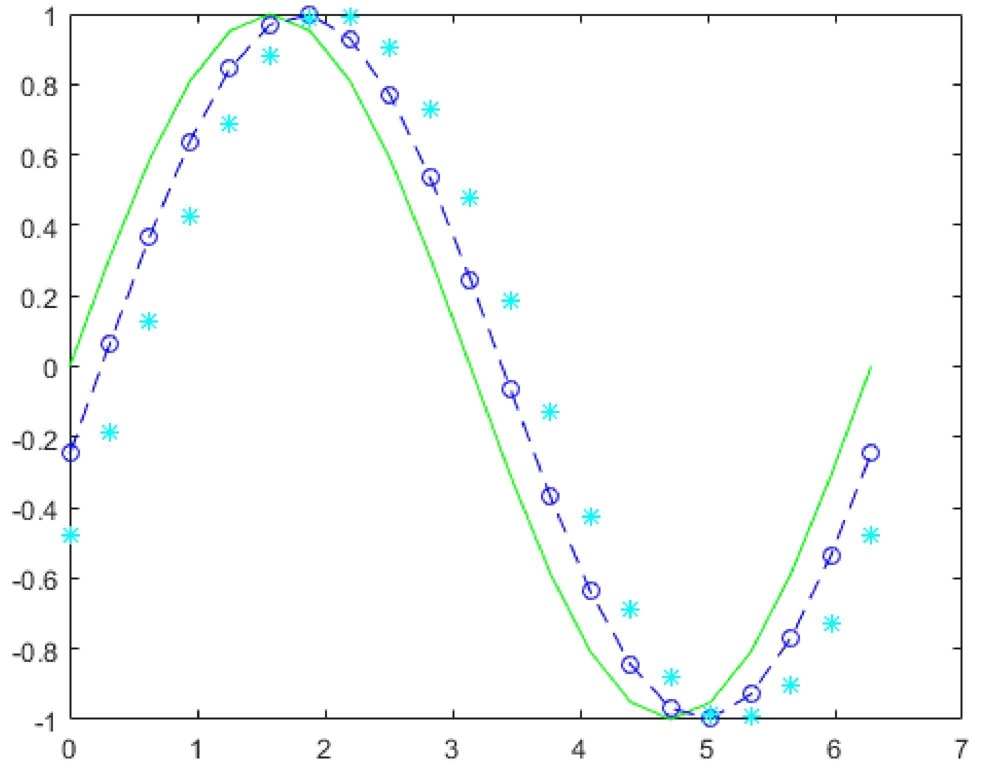
Plotting example

x = 0:pi/10:2*pi;

y1 = sin(x);

y2 = sin(x-0.25);

y3 = sin(x-0.5);



plot(x,y1,'g',x,y2,'b--o',x,y3,'c*')

Plotting example

x = 0:pi/10:2*pi;

y1 = sin(x);

plot(x,y1,'g’)

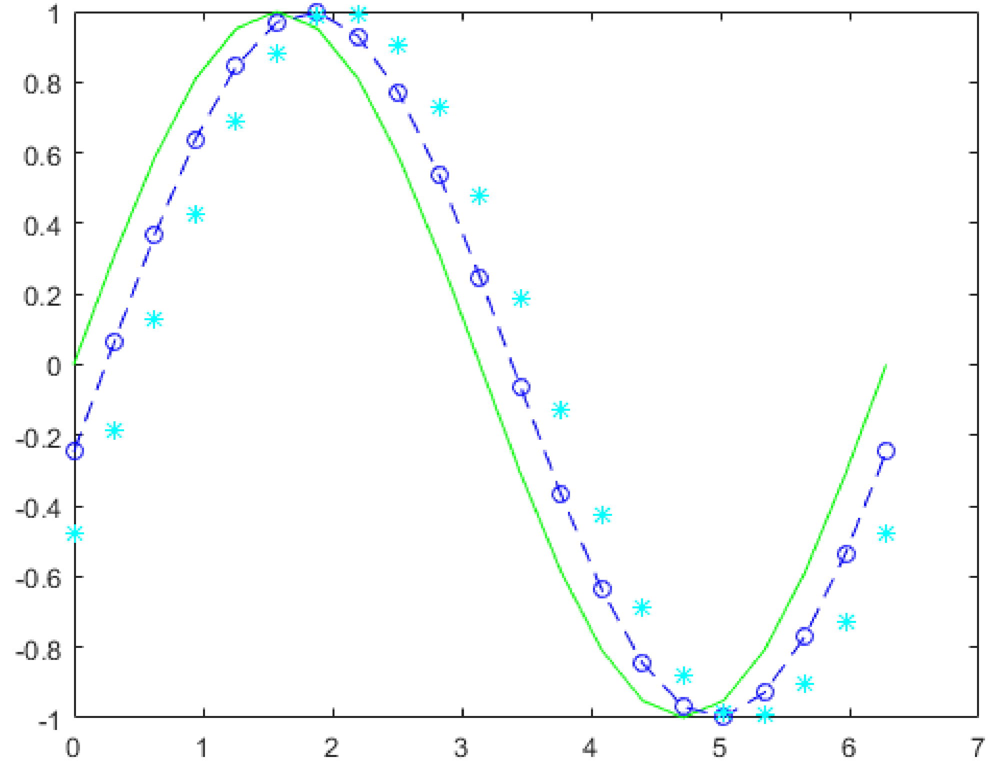
hold on

y2 = sin(x-0.25);

Plot(x,y2,'b--o’)

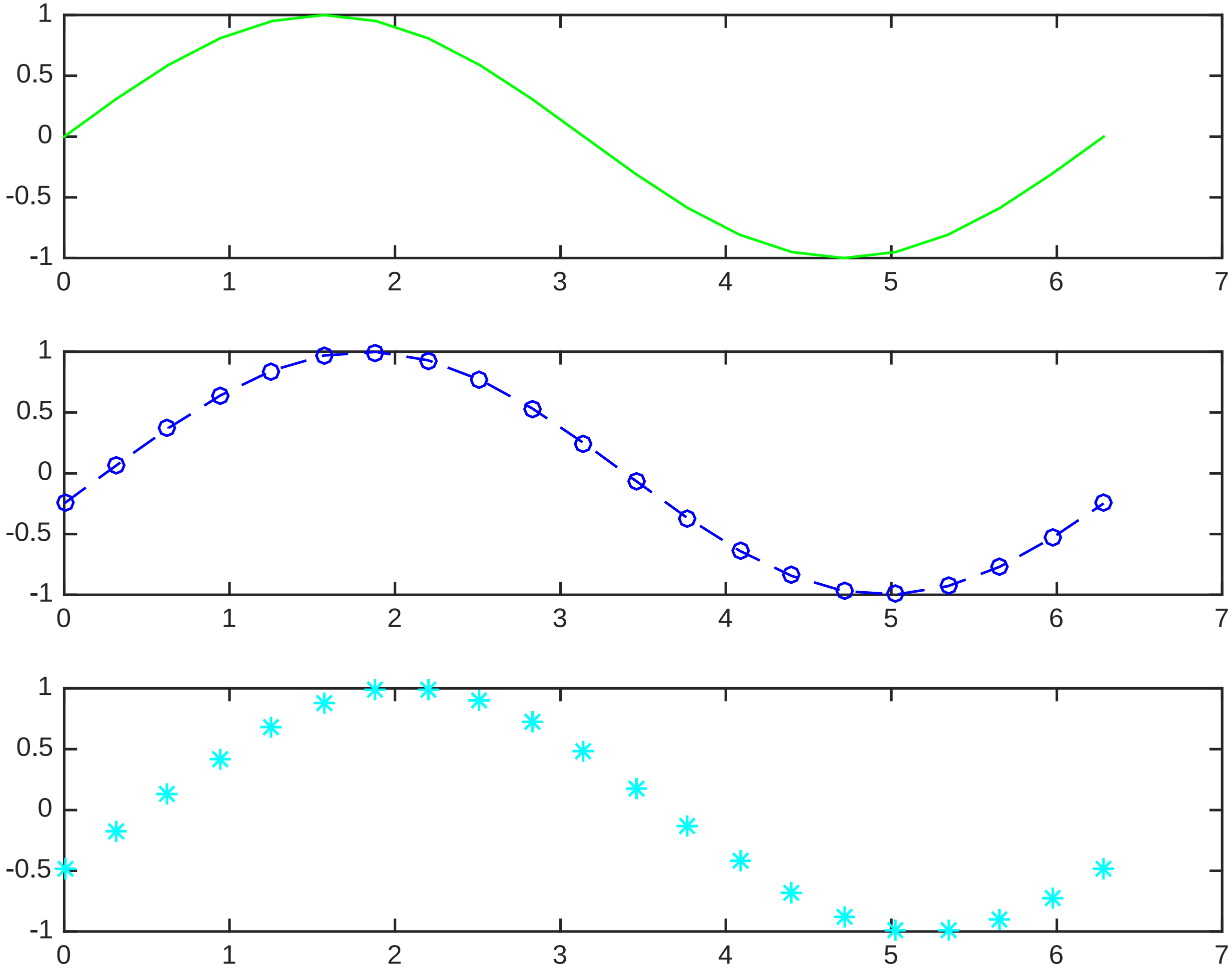
y3 = sin(x-0.5);

Plot(x,y3,'c*')



subplot

subplot(3,1,1);
plot(x,y1,'g')
subplot(3,1,2);
plot(x,y2,'b--o')
subplot(3,1,3);
plot(x,y3,'c*')



Display Facilities

- title(.)

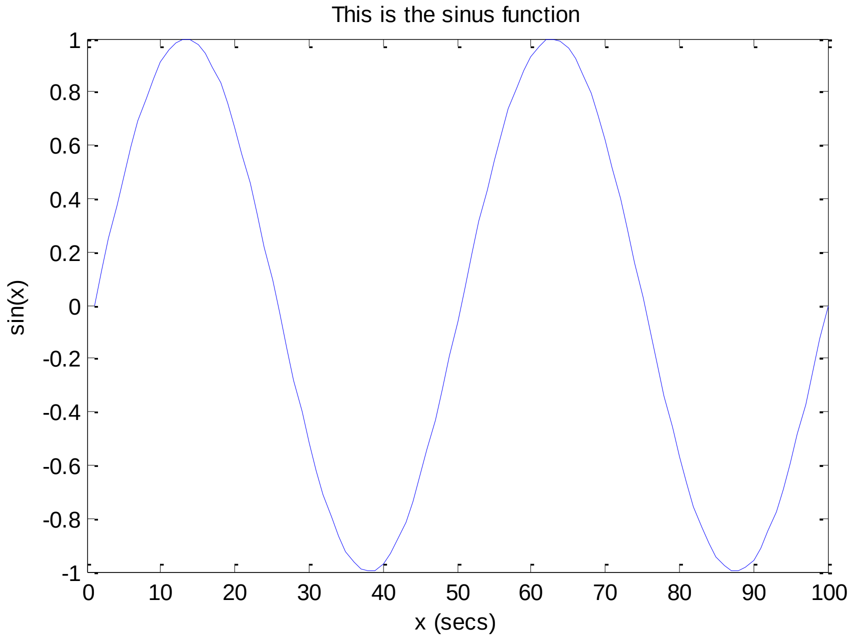
```
>>title('This is the sinus function')
```

- xlabel(.)

```
>>xlabel('x (secs)')
```

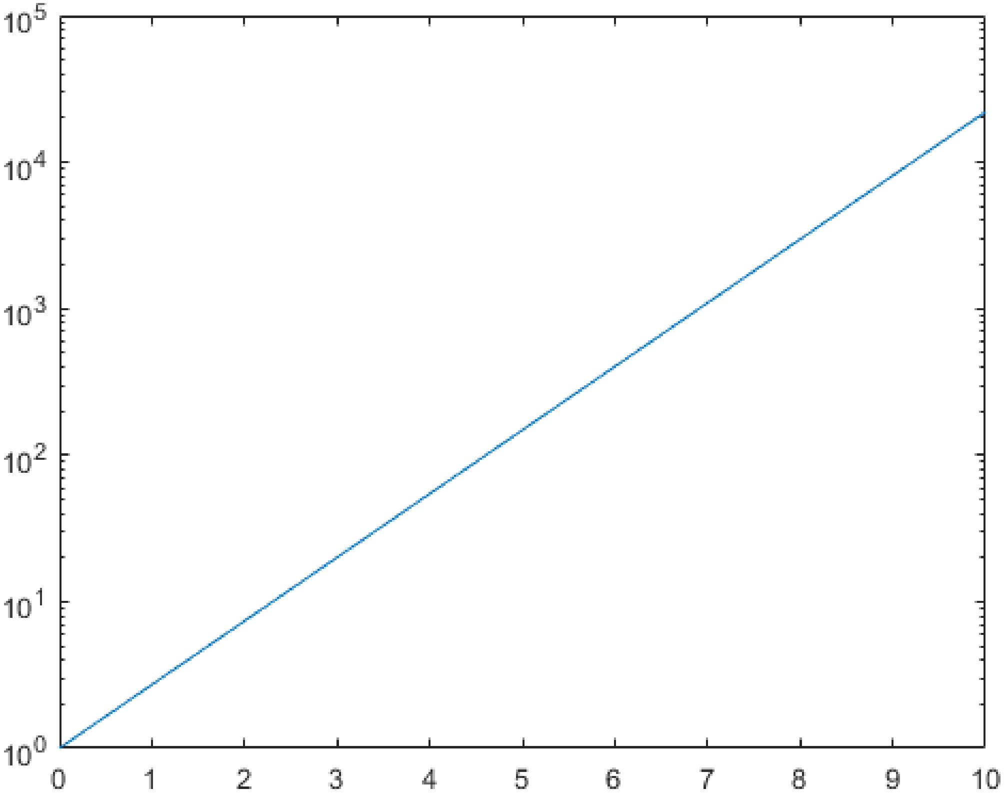
- ylabel(.)

```
>>ylabel('sin(x)')
```



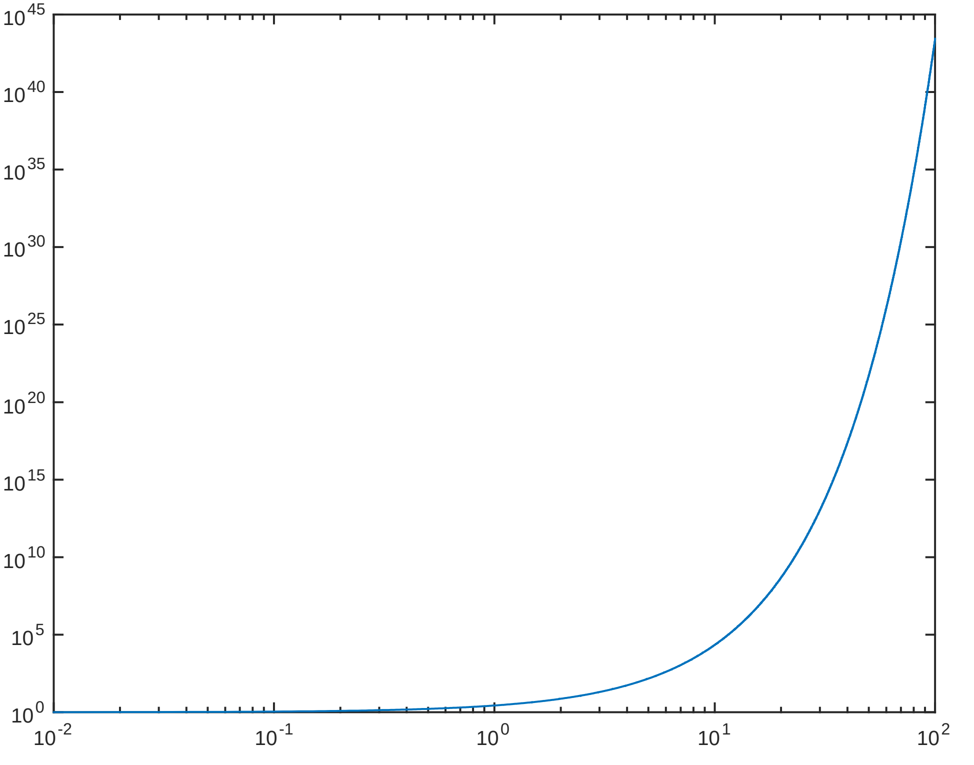
semilogy

- x = 0:0.1:10;
- y = exp(x);
- semilogy(x,y)

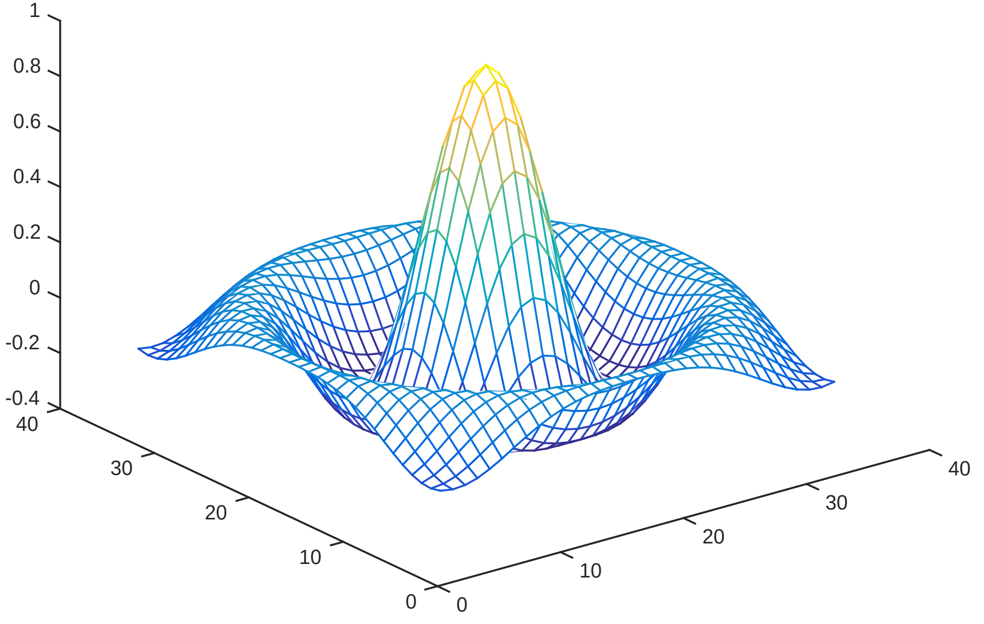


loglog

- x = 0.01: 0.01:100;
- y = exp(x);
- loglog(x,y)



- [X,Y] = meshgrid(-8:.5:8);
- R = sqrt(X.^2 + Y.^2) ;
- Z = sin(R)./R;
- mesh(Z)



The *for* Loop in MATLAB

- In MATLAB, a *for* loop begins with the statement indicating how many times the statements in the loop will be executed
- A counter is defined within this statement
- Examples:

```
for k = 1:100
```

(counter = *k*, the loop will be executed 100 times)

```
for i = 1:2:7
```

(counter = *i*, the counter will be incremented by a value of 2 each time until its value reaches 7. Therefore, the loop will be executed 4 times (*i* = 1,3,5, and 7)

for Loop Example

```
for j = 1:10
    x(j) = 5*j;
end
```

- The first time through the loop, $j = 1$
- Because of the single value in parentheses, x will be a one-dimensional array
- $x(1)$ will be set equal to $5*1 = 5$
- The second time through the loop, j = 2
- $x(2)$ will be set equal to $5*2 = 10$
- This will be repeated until $j = 10$ and $x(10) = 50$

For loop exercises

- Find n! using matlab
- Find the 1+2+3+…+100 using matlab
- Find the 3+6+9+99 using matlab
- Make matrix of form

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36

using for loop in matlab

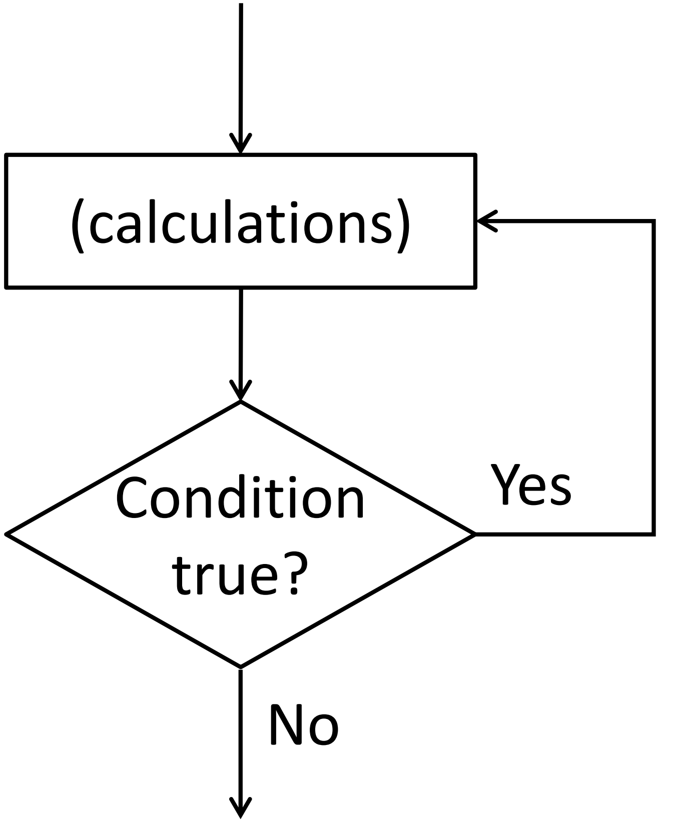
Flow Chart of *while* Loop

- The first line of this loop is:

```
while (condition)
```

- Last line is:

```
end
```



Example

- Consider this loop:

```
k = 0;
while k < 10
    k = k + 2
end
```

- How many times will the loop be executed?

Initially, k = 0, so the loop is entered
Pass #1: k = 2, so execution continues
Pass #2: k = 4, so execution continues
Pass #3: k = 6, so execution continues
Pass #4: k = 8, so execution continues
Pass #5, k = 10, so k is not less than 10 and execution ends

Useful Commands

- The two commands used most by Matlab users are

```
>>help functionname
```

```
>>lookfor keyword
```

Thank You…