MATLAB Notes

Course over view: (What am I going to learn)

1. What MATLAB is?
2. Learn how to create and manipulate Matrices.
3. Save mathematical problems in easy and efficient way.
4. 2D/3D graphs and 2D animation.
5. MATLAB Programming to create algorithms.
6. Import and export Data from excel and txt files.
7. Symbolic Math Toolbox.

Get the most out of this course

1. Finish the tutorial
2. Take notes
3. Apply on MATLAB
4. Take the quiz

Array-Defination

- An array is a collection of systematic arrangement of objects of the same data type.
- MATLAB considers any variable is an array even if that variable is a single number, it is 1x1 array

```
a1  a2  a3 . . .  an
```

Types-of-Array

1. 1D (Vector)

```
a1 = 1 2 3 4 5


         1

a2 =     2

         3
```

2. 2D (Matrix)

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

3. Multidimensional array

Matrix-Opration

```
U = [ 1, 2, 3 ]

>> U =

         1    2    3

U = [ 1, 2, 3 ]'  or U = [ 1; 2; 3 ]

>> U =

            1

            2

            3
--------------------------------

A = [1, 2, 3];
B = [4, 5, 6];
C = [7, 8, 9];

X = [ A; B; C;]

>> X =

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

>> X(1:4) (to select 1 to 4 elements)

```
= 1 4 7 2


>> X =
        1 2 3
        4 5 6
        7 8 9


>> X(5) (to select 5th element)

= 5

>> X =

        1 2 3
        4 5 6
        7 8 9

>> X(1,:) (to select first row)

= 1 2 3

>> X =

        1 2 3
        4 5 6
        7 8 9

>> X(:,2) (to select Second Column)

    2
=   5
    8

>> X =

        1 2 3
        4 5 6
        7 8 9

>> X(n1:n2,:) (to select one row to another row)
>> X(:,n1:n2) (to select one Column to another Column)

--------------------------------

>>linspace(x1,x2,n)
```

- Gives line space of lower limit “x1” upper limit “x2” with number of point is equal to “n”

- It gives linearly spaced row vector

Ex:

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

ans =

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

```
>>logspace(a,b,n)
```

- Gives log space of lower limit "10^a" upper limit "10^b" with number of point is equal to "n"

Ex:

```
>> logspace(1,2,5)

ans =

   10.0000   17.7828   31.6228   56.2341  100.0000
```

Special-Matrices-in-MATLAB

1. Zeros Function

- Gives matrix with all the elements is equal to zeros

```
>> a=zeros(4) (gives 4x4 matrix)

a =

     0     0     0     0
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> a=zeros(2,3) (gives 2x3 matrix)

a =
     0     0     0
     0     0     0
--------------------------------
```

If I have an existing matrix of lets say

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

a =

     1     2     3
     4     5     6
```

and want to have zero matrix of that matrix then

```
>> b=zeros(size(a))

b =

     0     0     0
     0     0     0
--------------------------------
```

2. Ones Function

- Gives matrix with all the elements is equal to one

```
>> a=ones(4)

a =

     1     1     1     1
     1     1     1     1
     1     1     1     1
     1     1     1     1
--------------------------------
```

3. Eye Function

- Gives identity matrix

```
>> a=eye(3)

a =

     1     0     0
     0     1     0
     0     0     1
--------------------------------
```

4. Rand Function

– Gives matrix with random number from 0 to 1

```
>> a=rand(2,3)

a =

    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
--------------------------------
```

5. Randi Function

– Gives matrix with random integers

```
>> a=randi(10,3,2) (gives 3x2 matrix with random numbers
ranging from 1 to 10)

a =

     7     2
     7     5
     2    10
--------------------------------
```

Size-Sort-Find-Max-Min

1. Size Function

– Returns two values Row & Column

```
>> a=randi(10,3)

a =

     4     8     7
     6     3     9
     3     6    10

>> b=size(a)

b =
     3     3
--------------------------------
```

2. Sort Function

- Sorts all the column of a matrix in ascending order (less—>more)

```
>> a=randi(10,4,2)

a =

     6     9
     2     3
     2     9
     3     3

>> sort(a)

ans =

     2     3
     2     3
     3     9
     6     9
```

Note: More about sort() function in upcoming topics.

```
--------------------------------
```

3. Find Function

- Outputs the values as the place for non-zeros elements. (Find indices and values of nonzero elements)

```
>> a=[0,2,3;1,0,9;8,9,0]

a =

     0     2     3
     1     0     9
     8     9     0

>> b=find(a) (returns indices of nonzero elements)

b =
     2  (1)
     3  (8)
     4  (2)
     6  (9)
     7  (3)
     8  (9)

>> a=[1,2,3;4,5,6;7,8,0]

a =

     1 (1)     2 (2)     3 (3)
     4 (1)     5 (2)     6 (3)
     7 (1)     8 (2)     0


>> [row,col,k]=find(a) (returns row and column
subscripts also the nonzero elements of a)

row =

     1
     2
     3
     1
     2
     3
     1
     2


col =

     1
     1
     1
     2
     2
     2
     3
     3


k =

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

4. Maximum AND Minimum of “a”

- Returns the value and indices of largest element of each column.

```
>> a=randi(10,3,3)

a =

    10 (1)     3        4
     4         7 (2)    9 (2)
     2         5        6

>> [x,y]=max(a)

x =

    10     7     9


y =

     1     2     2

>> a=randi(10,3,3)

a =

    10        3 (1)    4 (1)
     4        7        9
     2 (3)    5        6

>> [x,y]=min(a)

x =

     2     3     4
```

```
y =

     3     1     1
```

Basic-Operations-in-MATLAB

Addition and Substractions

```
a+b
a-b
```

Condition: size(a)=size(b)

——————————————————————————————

Multiplication

1. Matrix Multiplication

```
a*b
```

a*n ; n ∊ R

Condition: col of "a" = row of "b"
Result: matrix with row of "a" and column of "b"

2. Element by element multiplication

```
a.*b
```

Condition: size(a)=size(b)

——————————————————————————————

Division (element by element)

```
a./b
```

Condition: size(a)=size(b)

——————————————————————————————

Dot product of two Vectors

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

>> dot(a,b)

ans =

    32

>> sum(a.*b) or >> a*b' or >> b*a' (Alternate mathod)

ans =

    32
--------------------------------
```

Cross product of two Vectors

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

>> cross(a,b)

ans =

    -3      6     -3
```

Shifting-and-Sorting-of-Matrices

Sort (1D)

```
 a= >> a=[1,4,3,6,2];

>> sort(a)

ans =

     1      2      3      4      6

            low → high


>> sort(a,'descend')
```

```
ans =

     6     4     3     2     1

           high → low
```

Sort (Matrices)

1. Dimension 1 sort (along the Column) (top to bottom)

```
>> a=randi(10,3)

a =

     8     3     9
     5    10     6
     1     2    10

>> sort(a)

ans =

     1     2     6     top
     5     3     9      ↓
     8    10    10     bottom
```

2. Dimension 2 sort (along the Row) (left to right)

```
>> a=randi(10,3)

a =

     8     3     9
     5    10     6
     1     2    10

>> sort(a,2)

ans =

     3     8     9
     5     6    10
     1     2    10

      left → right
```

3. With indices

```
>> [x,y]=sort(a)

x =

     1     2     6
     5     3     9
     8    10    10


y =

     3     3     2
     2     1     1
     1     2     3
```

"sortrows" Function

- it sorts the matrix rows

```
>> a=randi(10,3)

a =

     1    10     9  (1st row)
     5     1     9  (2nd row)
     2     8     1  (3rd row)

>> sortrows(a)

ans =

     1    10     9  (1st row) top
     2     8     1  (3st row)   ↓
     5     1     9  (2st row) bottom
```

- It sorts the row with respect to the elements of the first column

- If we want to sort the row with respect to second row then..

```
>> sortrows(a,2)

ans =
     5     1     9
     2     8     1
     1    10     9
```

“issorted” Function

- Returns boolean value (true = 1 , false = 0)

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

>> issorted(a)

ans =

  logical

   1

>> a=[1,2,3;6,5,4;9,0,1]

a =

     1     2     3
     6     5     4
     9     0     1

>> issorted(a)

ans =

  logical

   0

>> issorted(a(:,1))

ans =

  logical

   1
```

Circular shift function

- It functions takes the first row and move it “n” in curricular way.

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

a =

     1     2     3
     4     5     6
     7     8     9

>> circshift(a,2)

ans =

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

- In this case n = 2, that means it took the first row and move not two steps in circle.

How to choose “n” random numbers from a Matrix ?

Step-1: find the number of element of a Matrix we want to choose numbers from using “ k=numel(a) ” function.

Step-2: generate “n” random number from 1 to “k” with “ i = randperm(k,n) ” function.

Step-3: 1st to “n”th element will be “n” random numbers from the matrix. Get those elements with N1= a(i (1)), N1= a(i (2)), N1= a(i (3))…… N1= a(i (n))

Ex: get two random numbers from the following matrix

```
>> a=[1,2,3,4;5,6,7,8;9,10,11,12;13,14,15,16]

a =

     1     2     3     4
     5     6     7     8
     9    10    11    12
    13    14    15    16

>> k=numel(a) (gives the number of elemnts of a matrix)

k =
    16

>> i=randperm(16,2) (randperm(k,n)= n random intigers
from 1 to k)

i =

     7     4

>> N=a(i)

N =

    10    13

10 and 13 are two random numbers
```

How to solve linear equation using matrix in MATLAB

In there following section we gonna see...

1. Determinants of matrices
2. Cramer's rule
3. Example - 1 solving equations using Cramer's rule
4. Inverse of a matrix
5. Example - 2 solving equations using inverse mathod

Determinants of matrices

- In MATLAB use det(a) function to find determinant of a matrix "a"

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

ans =

    -2

[4*1-3*2]
```

Cramer's rule

$a_{11}X + a_{12}Y + a_{13}Z = A$
$a_{21}X + a_{22}Y + a_{23}Z = A$
$a_{31}X + a_{32}Y + a_{33}Z = A$

$$x_0 = \frac{|D_1|}{\Delta}$$

$$y_0 = \frac{|D_2|}{\Delta}$$

$$z_0 = \frac{|D_3|}{\Delta}$$

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

$$D_1 = \begin{matrix} A & a_{12} & a_{13} \\ B & a_{22} & a_{23} \\ C & a_{32} & a_{33} \end{matrix}$$

$$D_2 = \begin{matrix} a_{11} & A & a_{13} \\ a_{21} & B & a_{23} \\ a_{31} & C & a_{33} \end{matrix}$$

$$D_3 = \begin{matrix} a_{11} & a_{12} & A \\ a_{21} & a_{22} & B \\ a_{31} & a_{32} & C \end{matrix}$$

Example -1 using Cramers's rule

1. Define Δ matrix in MATLAB
2. Then follow this…

```
>> %2X+3Y+Z=9
>> %X+2Y+3Z=6
>> %3X+Y+2Z=8
>> delta=[2,3,1;1,2,3;3,1,2]

delta =

     2     3     1
     1     2     3
     3     1     2

>> x1=det([9,3,1;6,2,3;8,1,2])/det(delta) (x0 = D1 / Δ)

x1 =

    1.9444

>> y1=det([2,9,1;1,6,3;3,8,2])/det(delta) (y0 = D2 / Δ)

y1 =

    1.6111

>> z1=det([2,3,9;1,2,6;3,1,8])/det(delta) (z0 = D3 / Δ)

z1 =

    0.2778
```

Inverse of Matrix

- In MATLAB use inv(a) function to find inverse of a matrix “a”

```
>> delta=[2,3,1;1,2,3;3,1,2]

delta =

     2     3     1
     1     2     3
     3     1     2

>> inv(delta)

ans =

    0.0556   -0.2778    0.3889
    0.3889    0.0556   -0.2778
   -0.2778    0.3889    0.0556
```

Example -2 inverse method

$a_{11}X + a_{12}Y + a_{13}Z = A$
$a_{21}X + a_{22}Y + a_{23}Z = A$
$a_{31}X + a_{32}Y + a_{33}Z = A$

$\Delta X = B$

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

$$X = \begin{matrix} Z \\ Y \\ Z \end{matrix}$$

$$B = \begin{matrix} A \\ B \\ C \end{matrix}$$

$$\Delta \cdot X = B$$

$$X = \Delta^{-1}B \quad \ldots(1)$$

- According to (1) equation I can say inv(delta)*B is solution.

Ex:

```
>> %2X+3Y+Z=9
>> %X+2Y+3Z=6
>> %3X+Y+2Z=8
>> delta=[2,3,1;1,2,3;3,1,2]

delta =

     2     3     1
     1     2     3
     3     1     2

>> B=[9;6;8]

B =

     9
     6
     8

>> X=inv(delta)*B

X =
    1.9444
    1.6111
    0.2778
```

operator precedence

Precedence	Operator(s)
Highest	Parentheses ()
	Exponentiation ^
	Multiplication * and Division /
	Addition + and Subtraction -
	Relational Operators <, >, <=, >=, ==, ~=
	Logical Operators & (AND), `
Lowest	Assignment Operator =

Trigonometry in MATLAB

unction	Description
sin(x)	Sine of angle *x* in radians.
cos(x)	Cosine of angle *x* in radians.
tan(x)	Tangent of angle *x* in radians.
asin(x) or asin(x, k)	Arcsine (inverse sine) of *x* in radians. Optional *k* specifies the branch (usually not needed).
acos(x) or acos(x, k)	Arccosine (inverse cosine) of *x* in radians. Optional *k* specifies the branch (usually not needed).
atan(x) or atan2(y, x)	Arctangent (inverse tangent) of *x* in radians. atan2(y, x) considers the signs of *x* and *y*.
sind(x)	Sine of angle *x* in degrees.
cosd(x)	Cosine of angle *x* in degrees.
tand(x)	Tangent of angle *x* in degrees.
asind(x) or asind(x, k)	Arcsine (inverse sine) of *x* in degrees. Optional *k* specifies the branch (usually not needed).

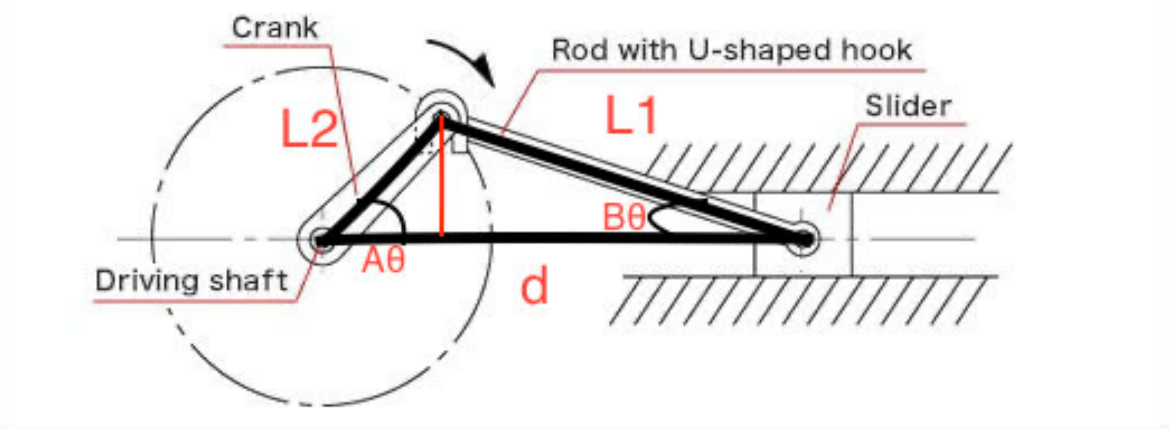
acosd(x) or acosd(x, k)	Arccosine (inverse cosine) of *x* in degrees. Optional *k* specifies the branch (usually not needed).
atand(x) or atand(y, x)	Arctangent (inverse tangent) of *x* in degrees. atand(y, x) considers the signs of *x* and *y*.

Some extra function:

```
>> deg2rad(30) (converts degree to radian)

ans =

    0.5236

>> hypot(3,4) (gives hypotenius)

ans =

     5
```

Mathematical Modelling of Slider in MATLAB

[Fig] Example of a slider-crank mechanism with exchangeable rod/slider parts



d = L1 * cos(B) + L2 * cos(A)

sin(B) = X / L1

X = L1 * sin(B)

X = L2 * sin(A)

$B = (L2 / L1) * \sin(A)$

$B = \sin^{-1}((L2 / L1) * \sin(A))$

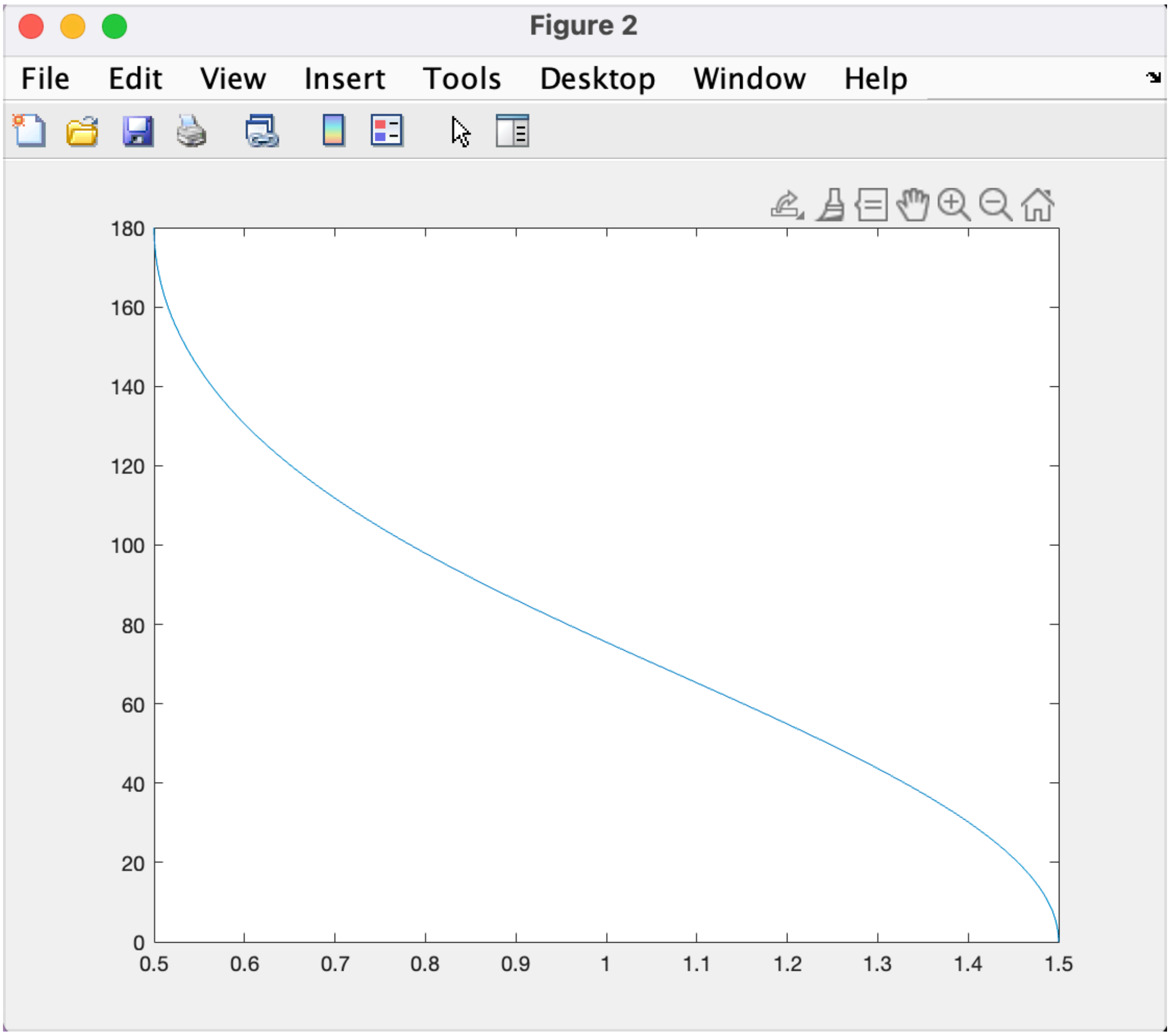
That means

$d(A) = L1 * \cos(\sin^{-1}((L2 / L1) * \sin(A))) + L2 * \cos(A)$

To model this in MATLAB lets us first define L1,L2 and angle

```
L1 = 1 , L2 = 0.5 and A starts from 0 to 180
```

```
>> L1=1; L2=0.5;
>> A=0:1:180;

>> d = L1 * cosd(asind((L2 / L1) * sind(A))) + L2 *
cosd(A);
>> plot(d,A)
```



Random Function

1. Ceil function
- It round offs to nearest integers in → direction (on the number line)

```
>> ceil(0.7)

ans =

     1
>> ceil(-2.3)

ans =

    -2
```

2. Floor function
- Opposite to ceil function I.e. rounds off in ← direction

```
>> floor(0.7)

ans =

     0

>> floor(-2.3)

ans =

    -3
```

3. Fix function
- Round offs to the nearest integer in the direction of “0”

```
>> fix(0.7)

ans =

     0

>> fix(-2.3)

ans =

    -2
```

4. Round function
- Round offs to the nearest integer

```
>> round(1.6)

ans =

     2

>> round(1.4)

ans =

     1

>> round(1.5)

ans =
     2
```

5. Sign function

- Returns three values -1, 0 and 1 according to the sign of the number

```
>> sign(1212)

ans =

     1

>> sign(-1324)

ans =

    -1

>> sign(0)

ans =

     0
```

Polynomials in MATLAB

1. How to define polynomials in MATLAB
2. Polynomial multiplication and division
3. Polynomial roots
4. Plotting polynomials
5. Residue function
6. Polynomial Integration (polying function)
7. Polynomial differentiation (polymer function)

How to define polynomials in MATLAB

- To define polynomial $9x^3 - 5x^2 + 3x + 7$

```
>> a=[9,-5,3,7]

a =

     9    -5     3     7
```

For the polynomial of $6x^2 - x + 2$

```
>> b=[6,-1,2]

b =

     6    -1     2
```

Polynomial multiplication and division

1. Multiplication of polynomials a and b

```
>> a=[9,-5,3,7]

a =

     9    -5     3     7

>> b=[6,-1,2]

b =

     6    -1     2

>> conv(a,b)

ans =

    54   -39    41    29    -1    14
```

- That means multiplication of $9x^3 - 5x^2 + 3x + 7$ and $6x^2 - x + 2$ is $54x^5 - 39x^4 + 41x^3 + 29x^2 - x + 14$

2. Division of polynomials

```
>> a=[3,1,0,-5];
>> b=[1,4];
>> deconv(a,b)

ans =

     3   -11    44

>> [q,r]=deconv(a,b)    (q saves result and r saves
reminder)

q =
     3   -11    44


r =

     0     0     0  -181
```

Polynomial roots

- To compute the roots of polynomial array “a” use function roots(a)

x^2+7^x+10 = (x+2)(x+5)

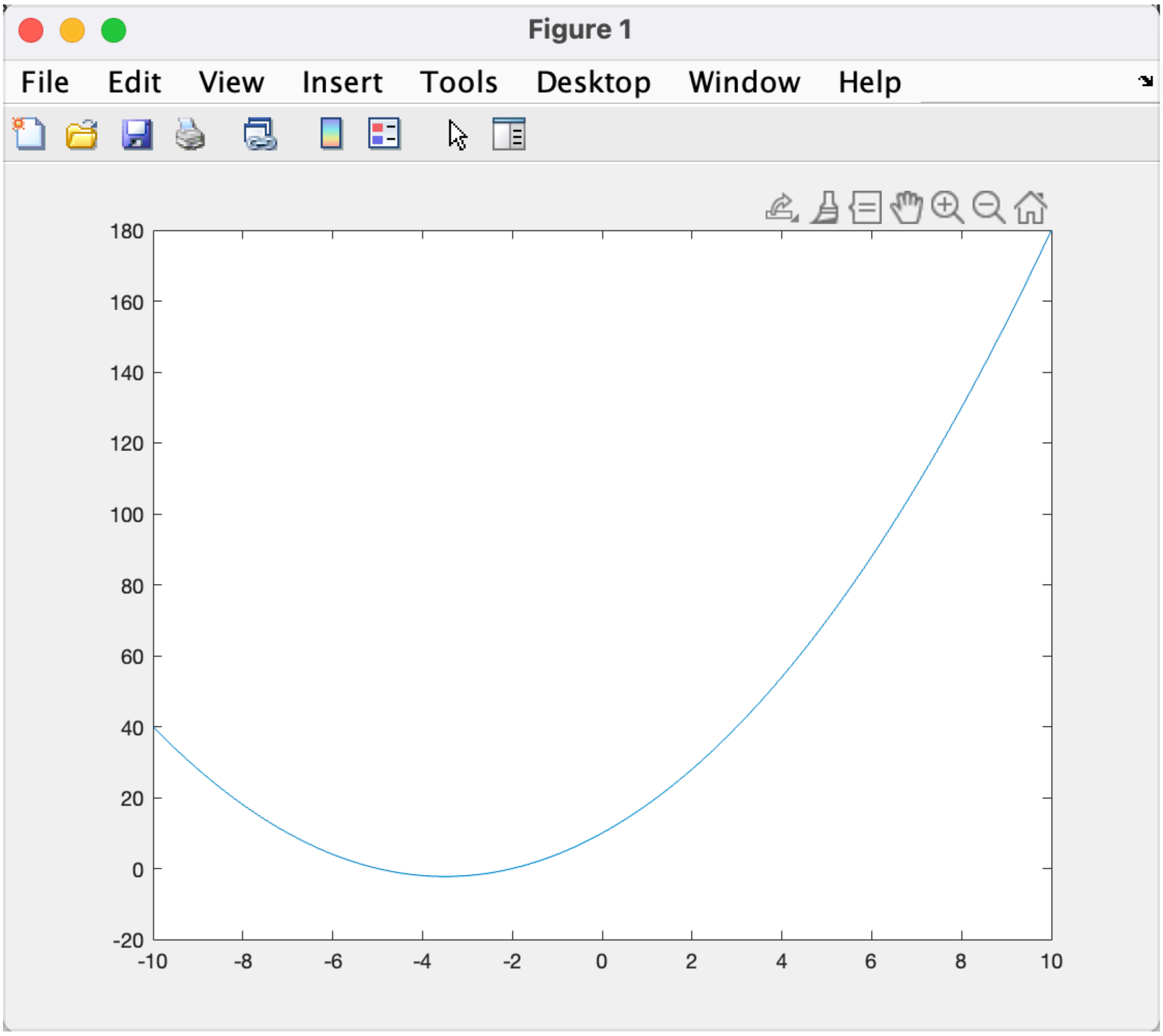
```
>> a=[1,7,10];
>> roots(a)

ans =

    -5
    -2
```

- Poly function (Opposite of root function)

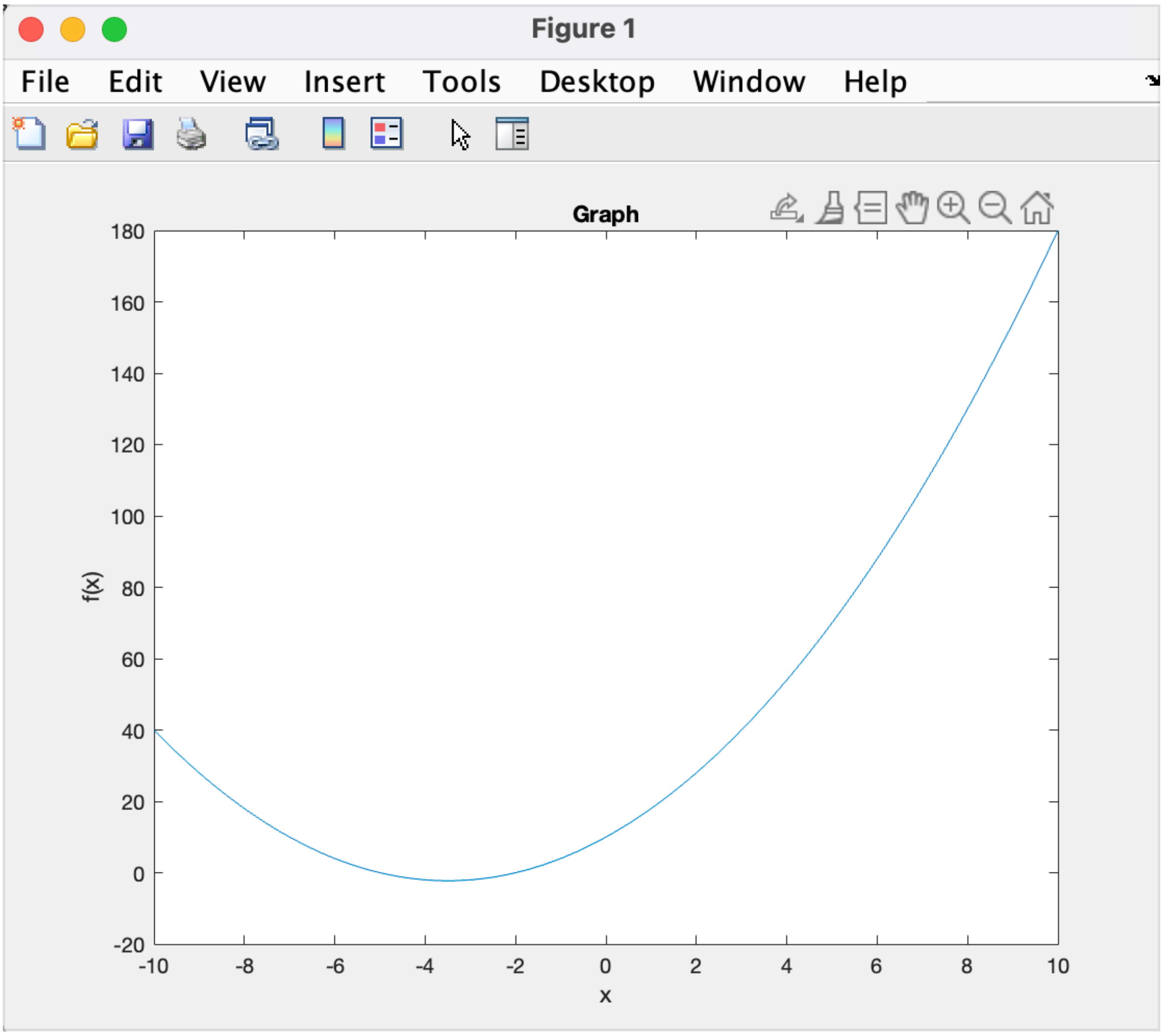
```
>> poly([-5,-2])

ans =

     1     7    10
```

Plotting polynomials

```
>> a=[1,7,10];
>> x=[-10:0.01:10];
>> f=polyval(a,x);
>> plot(x,f)
```



```
>> plot(x,f), xlabel ('x'), ylabel ('f(x)'), title
('Graph')
```



Residue function

- Finds the residue, poles and direct term of partial function expression pf the ratio of 2 polynomials

```
f(x) =        −4x + 8
           ‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑‑
            x^2 + 6x + 8

         −12          8
f(x) = ‑‑‑‑‑‑‑‑   +  ‑‑‑‑‑‑‑     + 0
        (x + 4)      (x + 2)
```

- Syntax of a function

```
[ r, p, k ] = residue(b,a)
```

- b = numinator poly
- a = denominator poly

```
>> a=[1,6,8];
>> b=[-4,8];
>> [r,p,k]=residue(b,a)

r =

   -12
     8


p =

    -4
    -2


k =

     []

>> [B,A]=residue(r,p,k) (viseversa is true)

B =

    -4     8


A =

     1     6     8
```

```
 f(x) =      -12            8
        --------   +  -------    + 0
          (x + 4)       (x + 2)
```

Polynomial Integration (polyint function)

```
>> polyint(a)
```

```
ans =

    0.3333    3.0000    8.0000         0

>> polyint(a,4)

ans =

    0.3333    3.0000    8.0000    4.0000
```

Polynomial differentiation (polymer function)

```
>> polyder(a)

ans =

     2     6



>> polyder(a,b)
```

- Gives differentiation of multiplication of polynomial a and b

Complex numbers in MATLAB

1. How to define complex numbers in MATLAB
2. Absolute function to calculate absolute value of complex number
3. Angle function
4. Conjugate of complex number
5. Real and imaginary function
6. isreal function

How to define complex numbers in MATLAB

```
>> z=3+4i

z =

   3.0000 + 4.0000i

or

>> z=3+4j
```

```
z =

   3.0000 + 4.0000i
```

Another way

```
>> z=complex(3,4)

z =

   3.0000 + 4.0000i
```

Absolute function to calculate absolute value of complex number

```
>> z=complex(3,4)

z =

   3.0000 + 4.0000i

>> abs(z)

ans =

     5
```

Angle function

- It gives angle of complex number with real axis

```
>> angle(z)

ans =

    0.9273
```

Conjugate of complex number

- Gives conjugate of complex number
- Conjugate is mirror image of complex number wrt to real axis
- For example conjugate of 3+4i is 3-4i

```
>> conj(z)

ans =
   3.0000 - 4.0000i
```

Real and imaginary function

- Real function gives real part of the number

```
z =

   3.0000 + 4.0000i

>> real(z)

ans =

     3
```

- imag function gives the imaginary part of the function

```
>> imag(z)

ans =

     4
```

isreal function

- Checks weather number or array z is real or not , and outputs boolean values accordingly

```
>> z=[1,2,3,4];
>> isreal(z)

ans =

  logical

   1

>> z=[1,2,3,i];
>> isreal(z)

ans =

  logical

   0
```

Log and exp function

Log Functions	Description	Example Usage
log(x)	Natural logarithm (base e) of x.	natural_log = log(x);
log10(x)	Base-10 logarithm of x.	base10_log = log10(x);
log2(x)	Base-2 logarithm of x.	base2_log = log2(x);
log1p(x)	Natural logarithm of (1 + x).	log_plus_1 = log1p(x);
Exp Functions	**Description**	**Example Usage**
exp(x)	Exponential function, e^x.	exponential = exp(x);
expm1(x)	e^x - 1, with higher precision for small x.	exp_minus_1 = expm1(x);

Cartesian to Polar

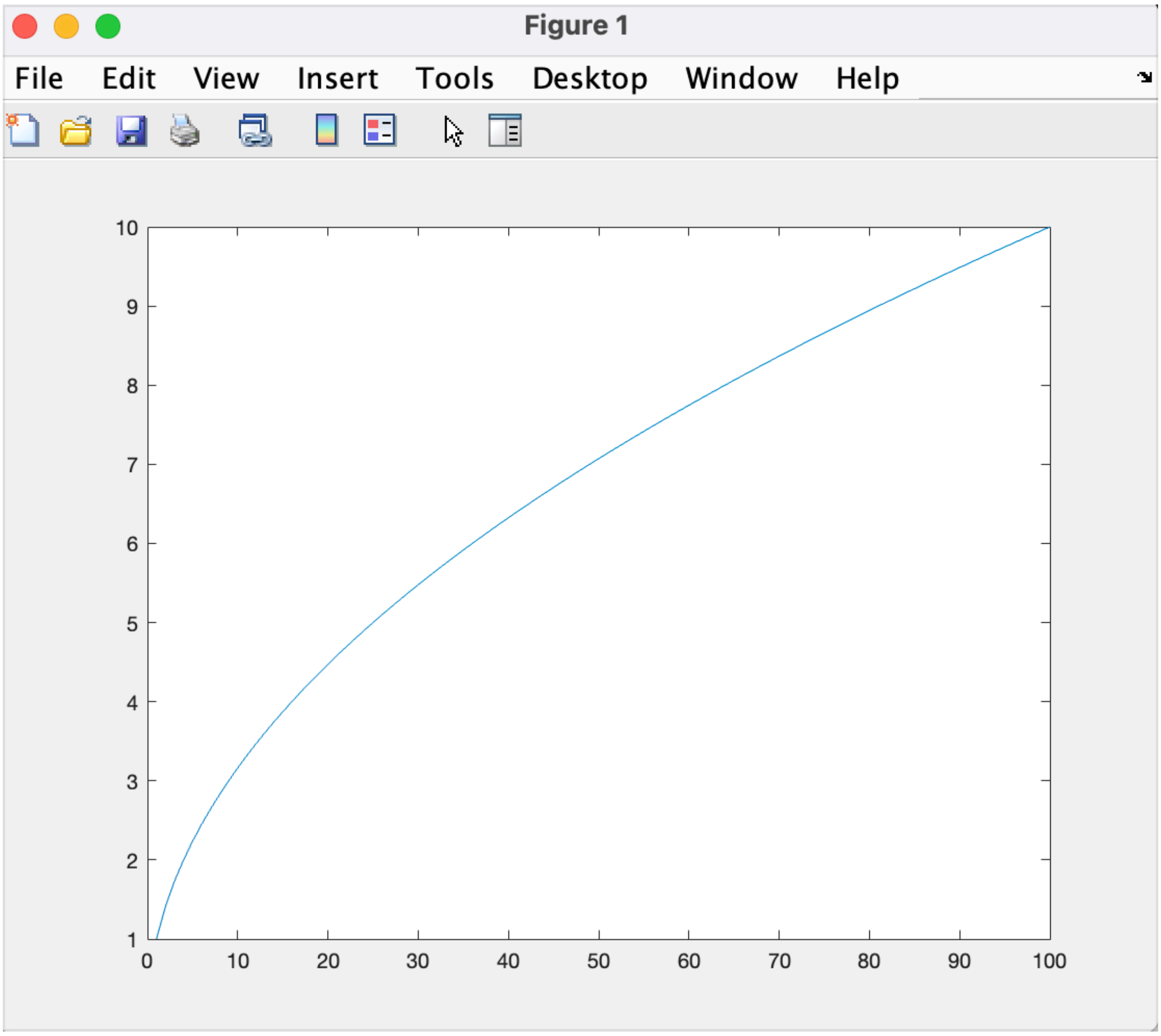
1. What is Polar coordinate
2. cart2pol function

Cartesian to Spherical

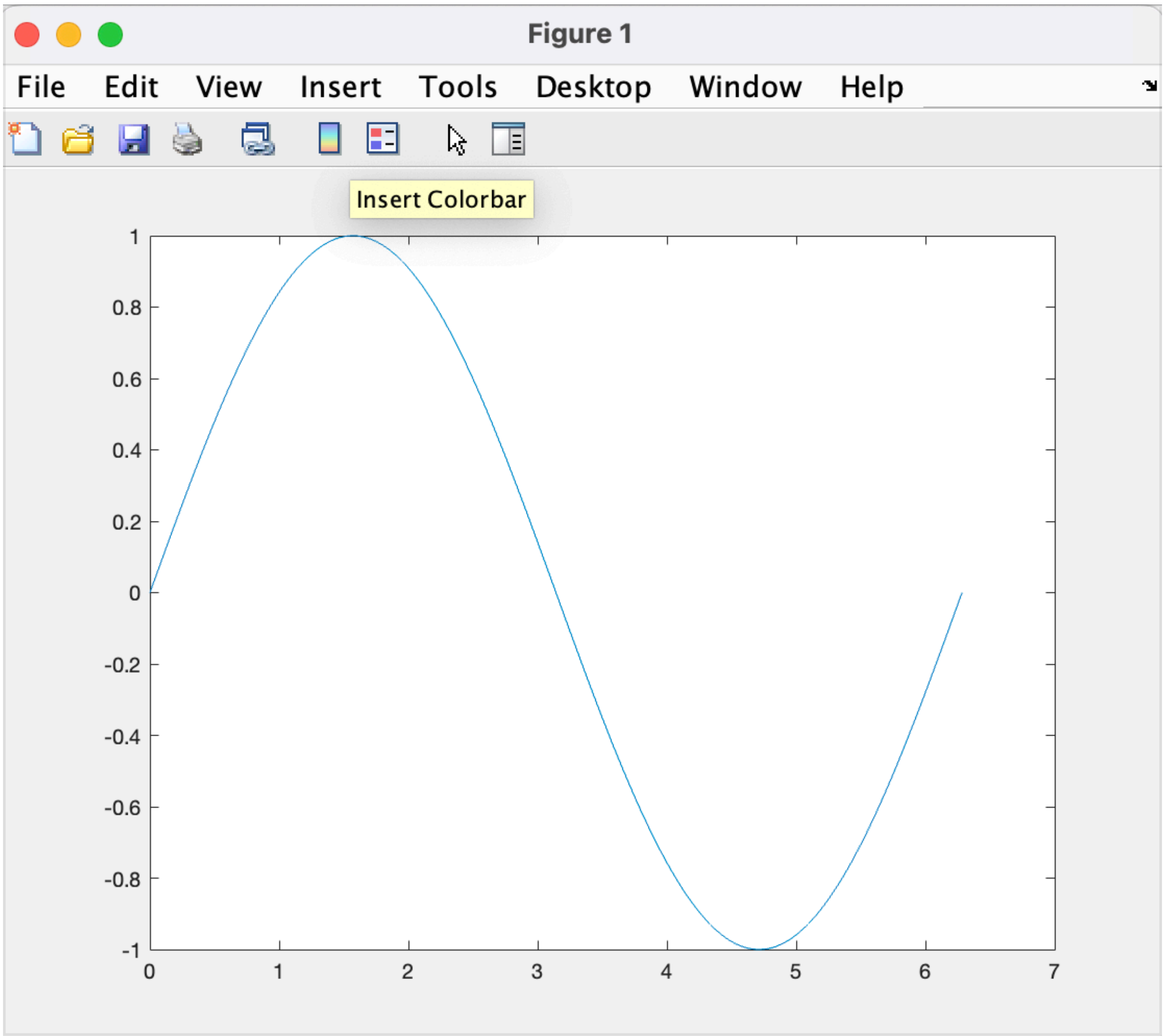
1. What us Spherical co-ordinate
2. cart2sph function

Plotting 2-D graph

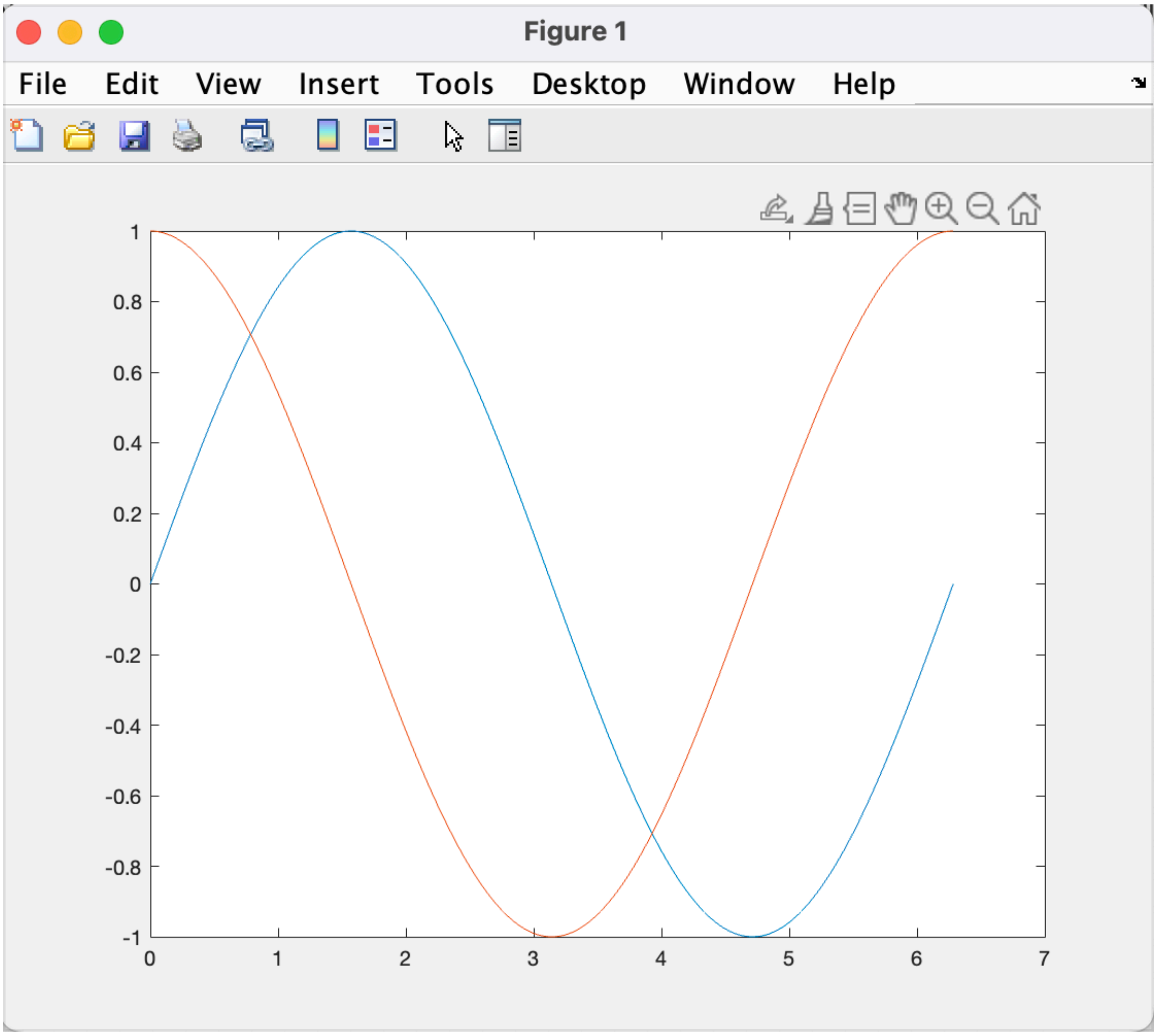
```
>> x=linspace(1,100);
>> y=sqrt(x);
>> plot(x,y)
```



```
>> x=linspace(0,2.*pi,10000);
>> y=sin(x);
>> plot(x,y)
```



```
>> y2=cos(x);
>> plot(x,y,x,y2)
```



```
>> plot(x,y), hold on,plot(x,y2)
```

- Hold on command hold on to the previous graph

```
>> plot(x,y), figure ,plot(x,y2)
```

- Figure command plots both graph in individual tab

```
>> x=linspace(0,2.*pi,10000);
>> y=sin(x);
>> plot(x,y,'--')
```

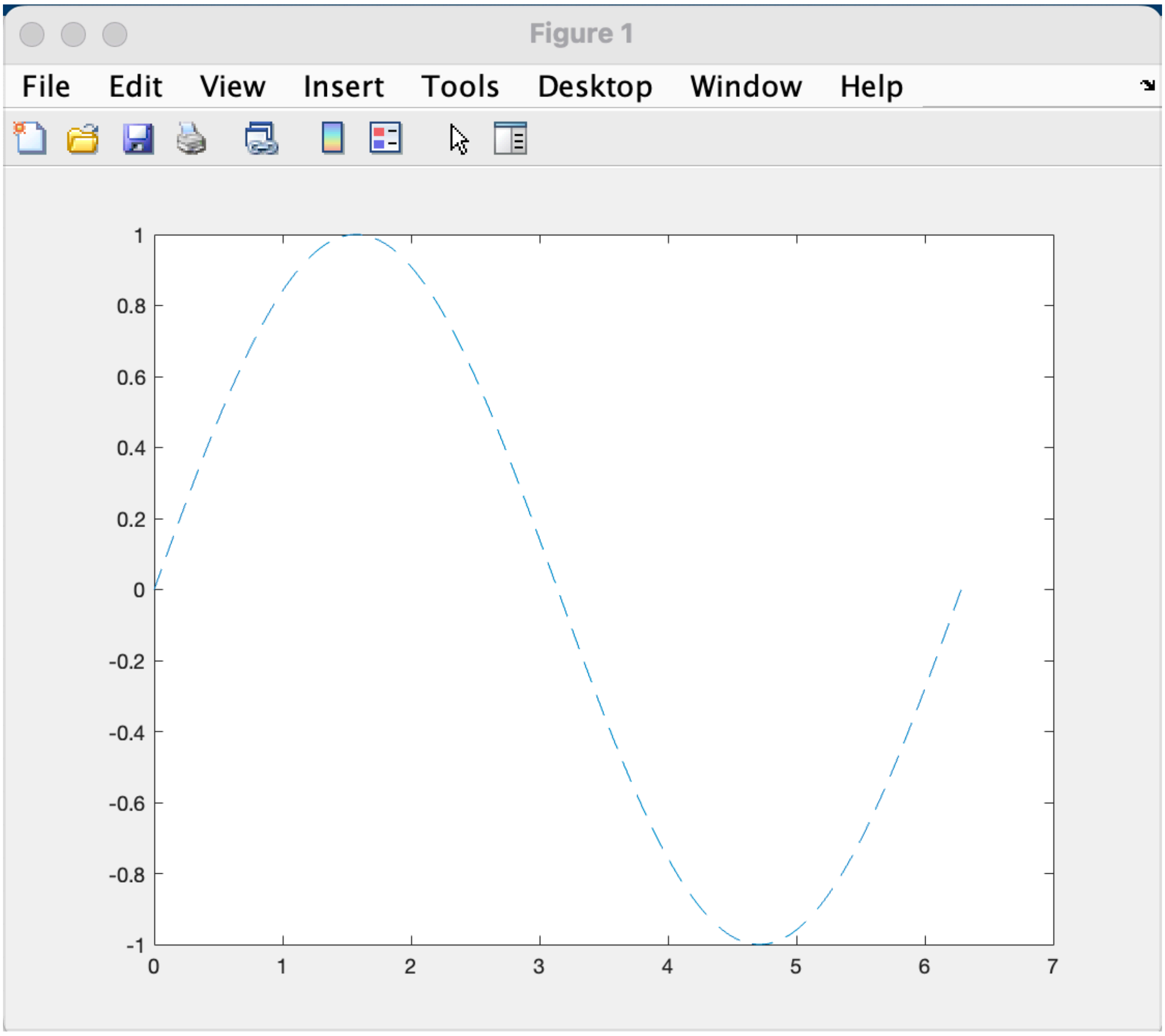
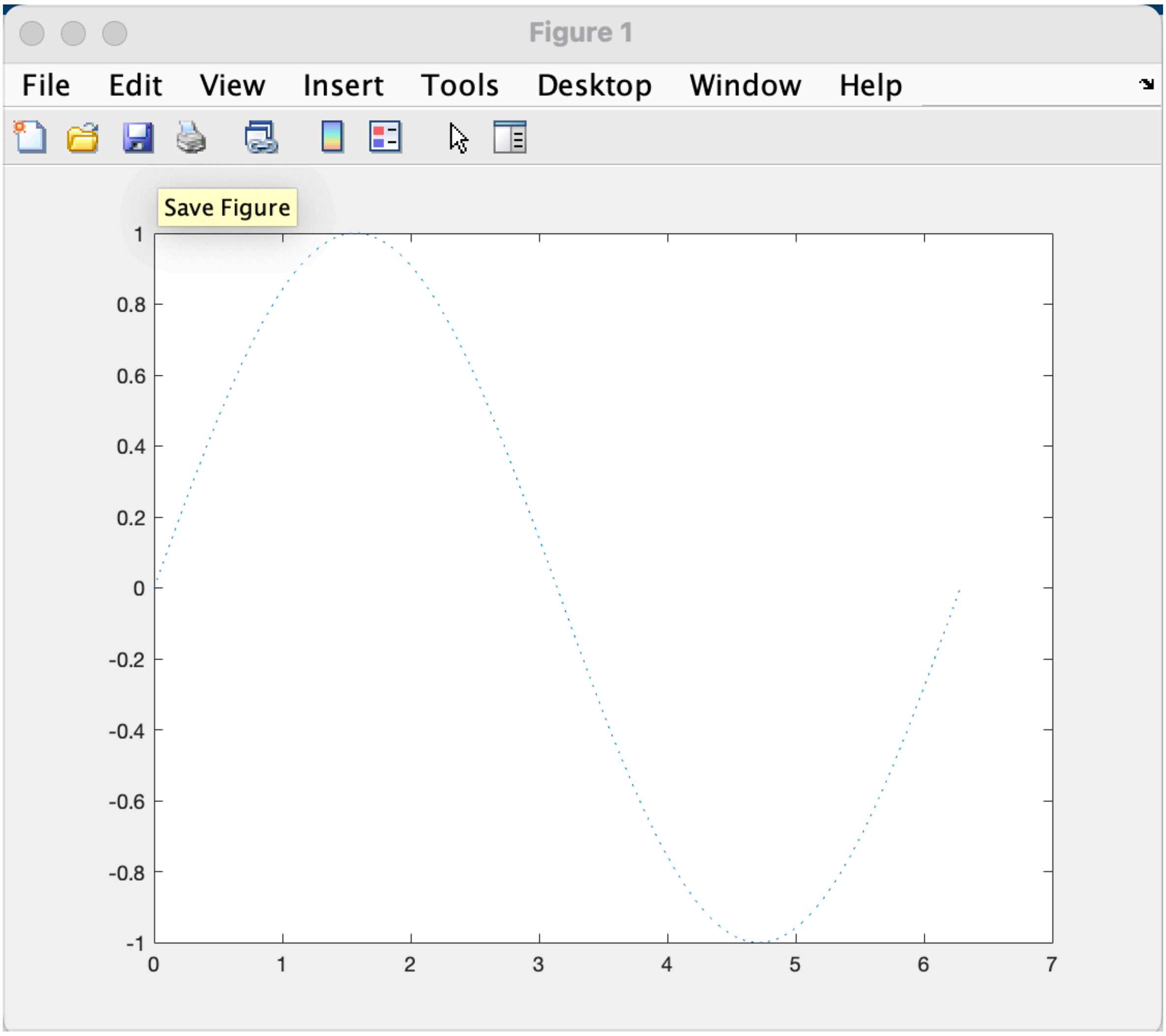


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7

```
>> plot(x,y,':')
```

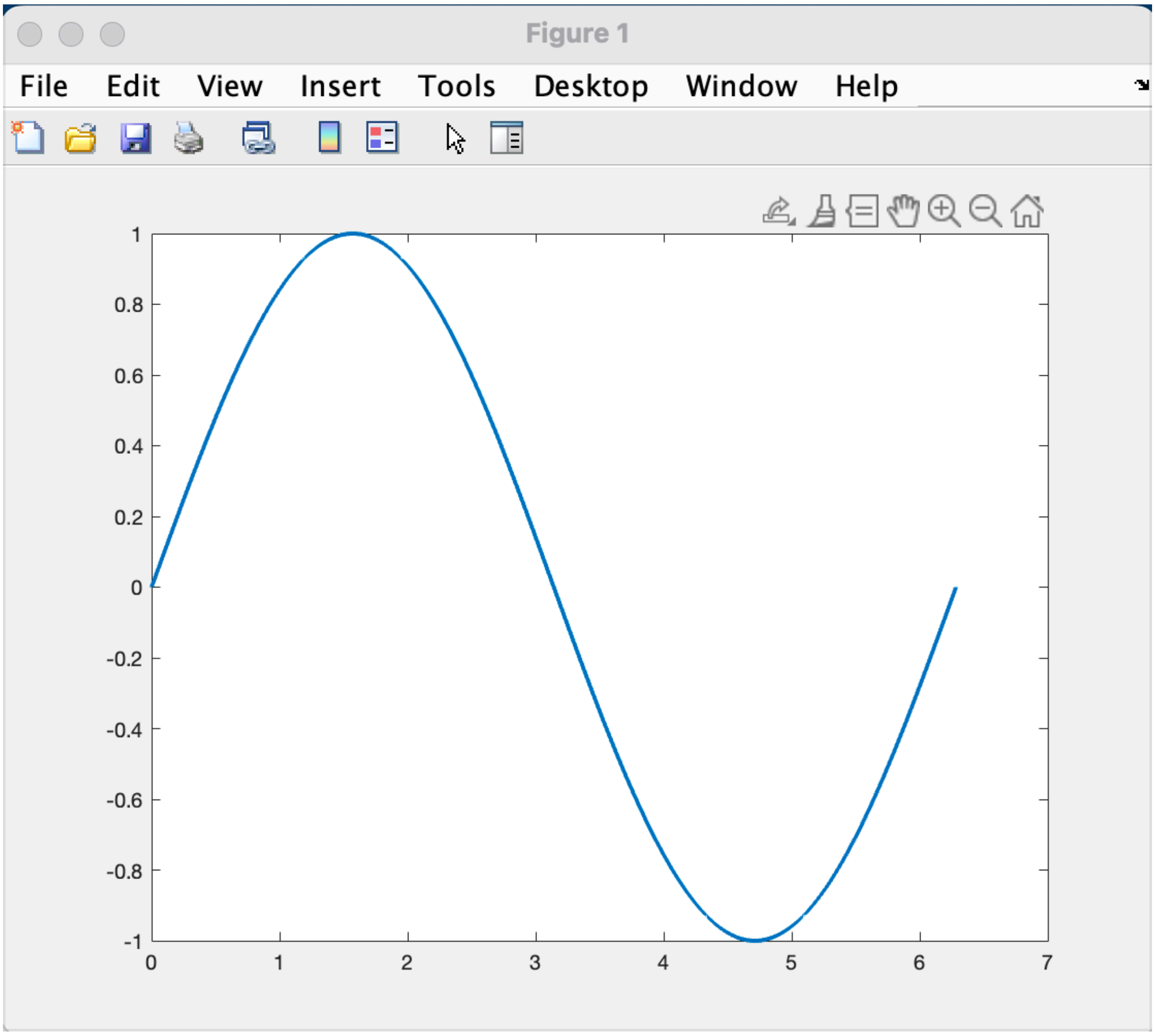


Line/Marker Style Specifiers Table:

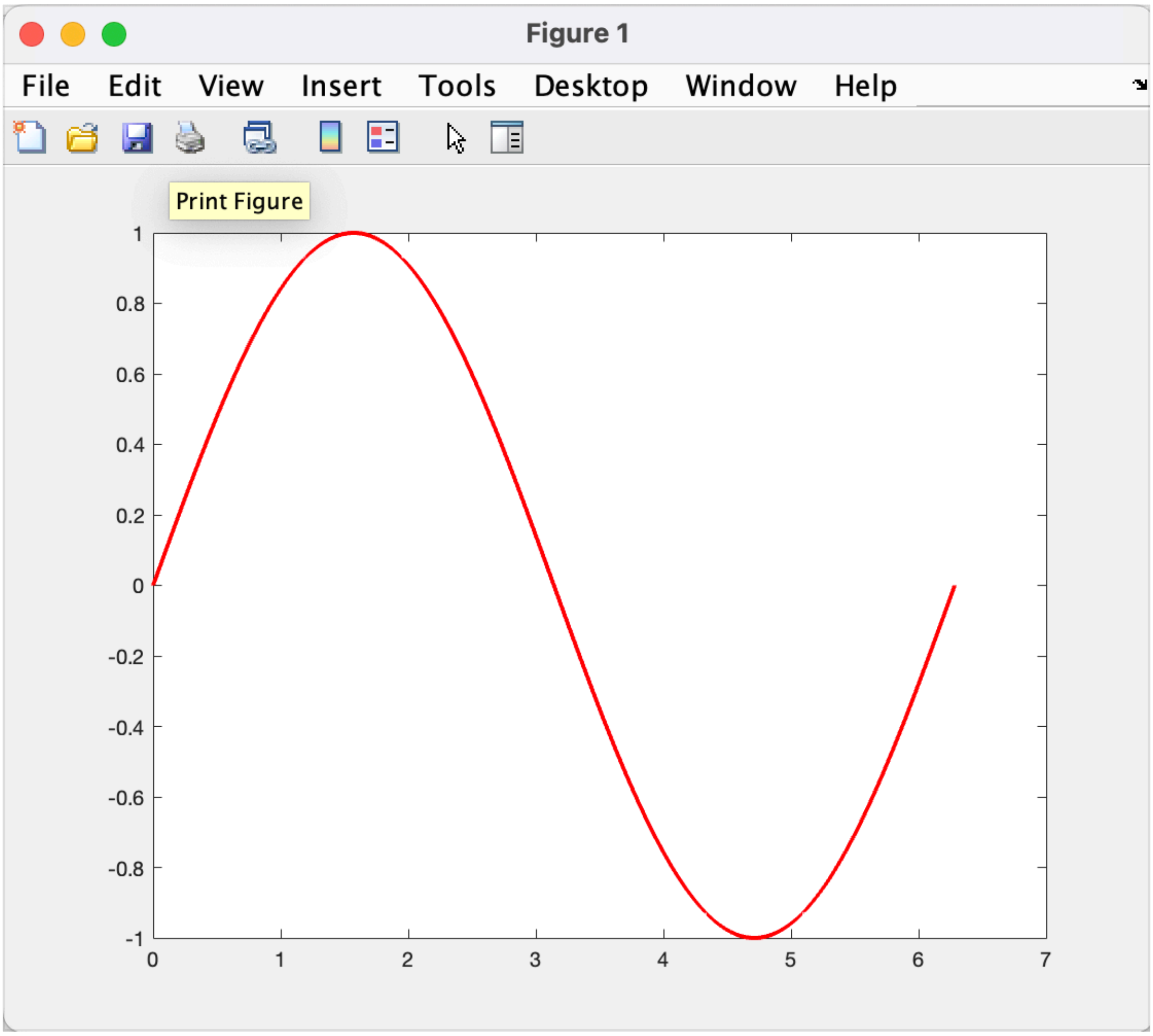
Style Specifier	Style Description
'-'	Solid line
'--'	Dashed line
':'	Dotted line
'-.'	Dash-dot line
'.'	Point marker
'o'	Circle marker
'x'	Cross marker
'+'	Plus marker
'*'	Asterisk marker
's'	Square marker
'd'	Diamond marker
'^'	Upward-pointing triangle marker
'v'	Downward-pointing triangle marker
'<'	Left-pointing triangle marker

'>'	Right-pointing triangle marker
'p'	Pentagon marker
'h'	Hexagram marker

```
>> plot(x,y,'linewidth',2)
```



```
>> plot(x,y,'r','linewidth',2)
```

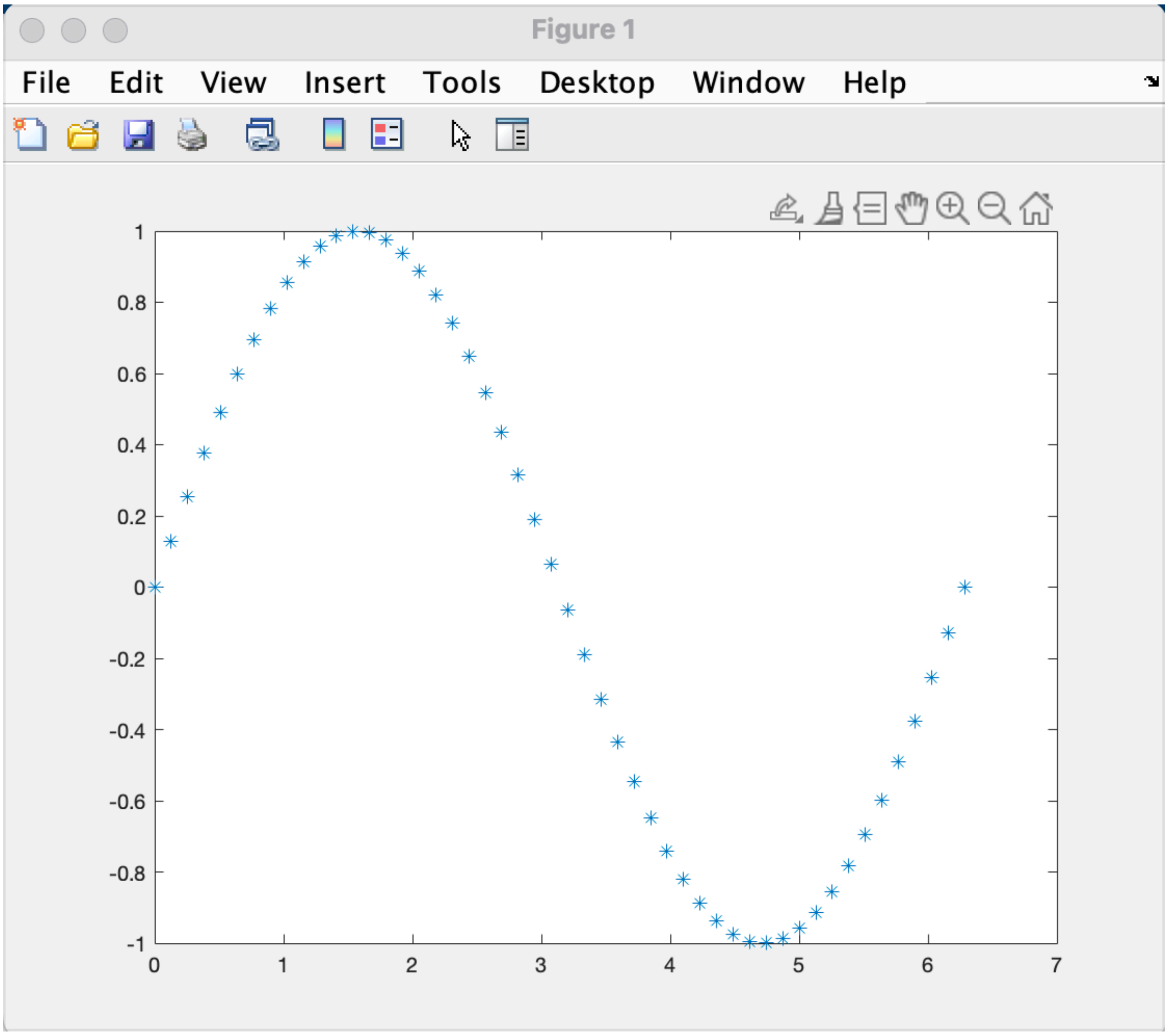


Color Specifiers Table:

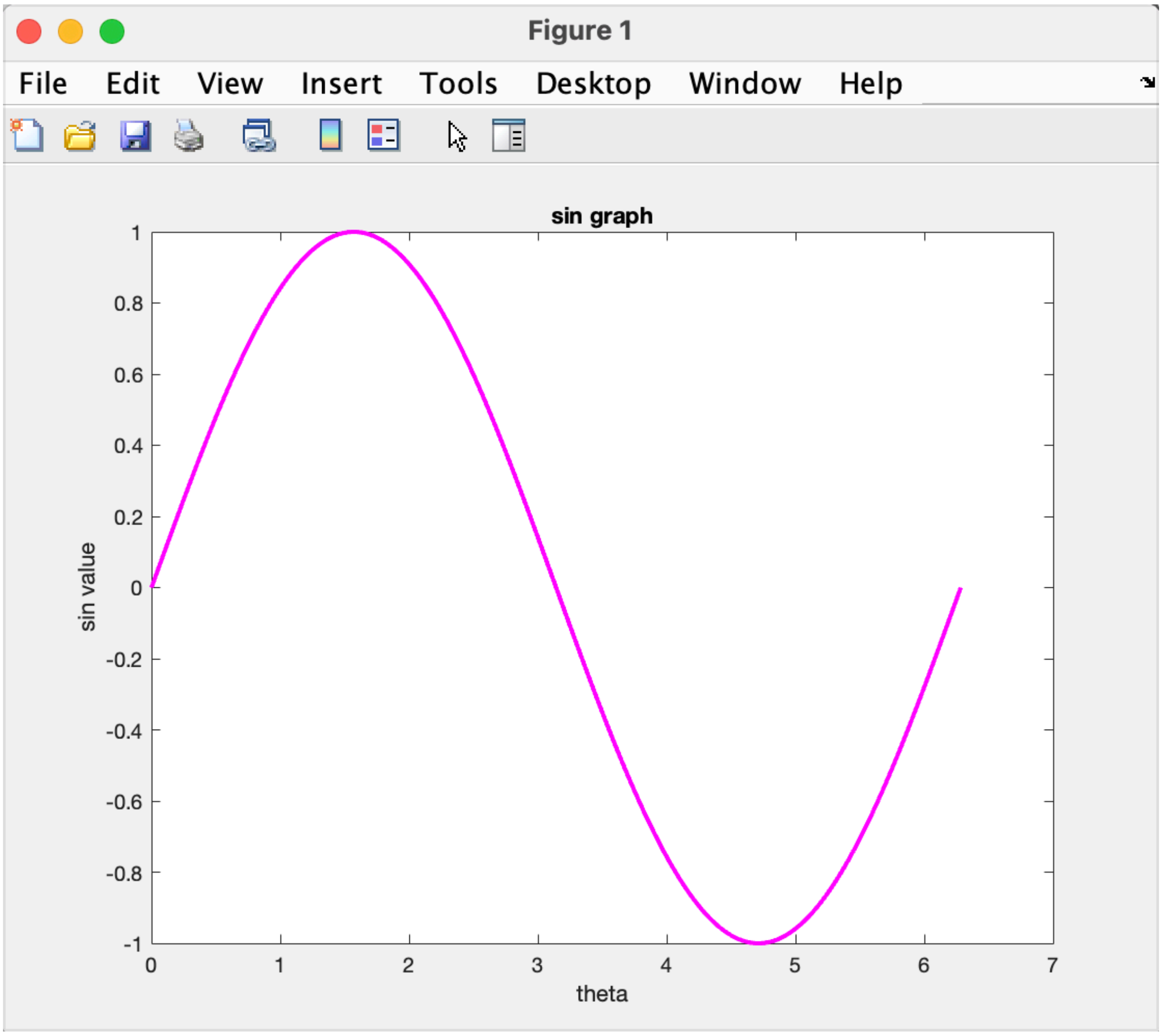
Color Specifier	Color Name
'b'	Blue
'g'	Green
'r'	Red
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

```
>> x=linspace(0,2.*pi,50);
>> y=sin(x);
>> plot(x,y,'*')
```

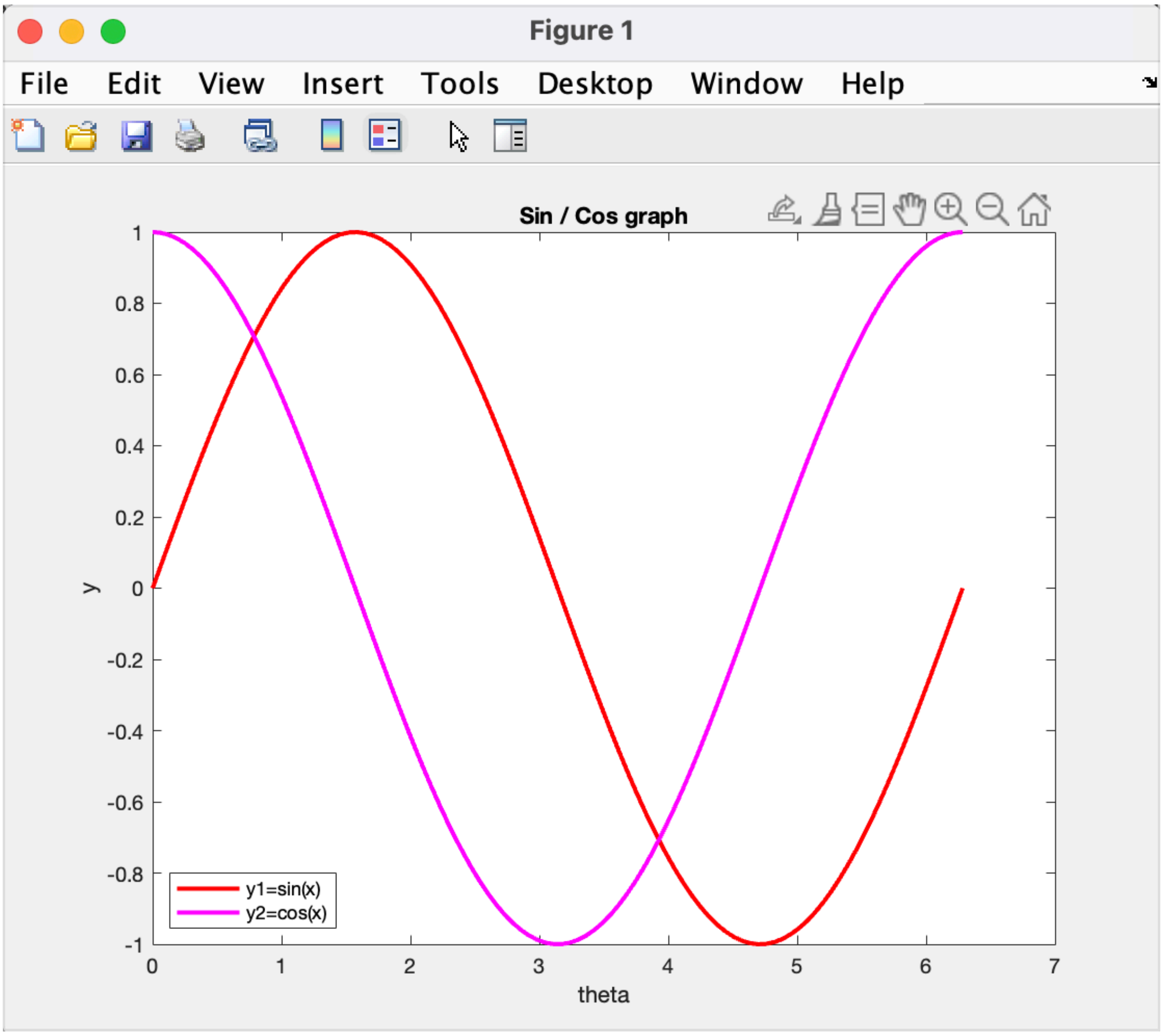
– '*' = to plot only data points



```
>> plot(x,y,'m','linewidth',2),title('sin
graph'),xlabel('theta'),ylabel('sin value')
```

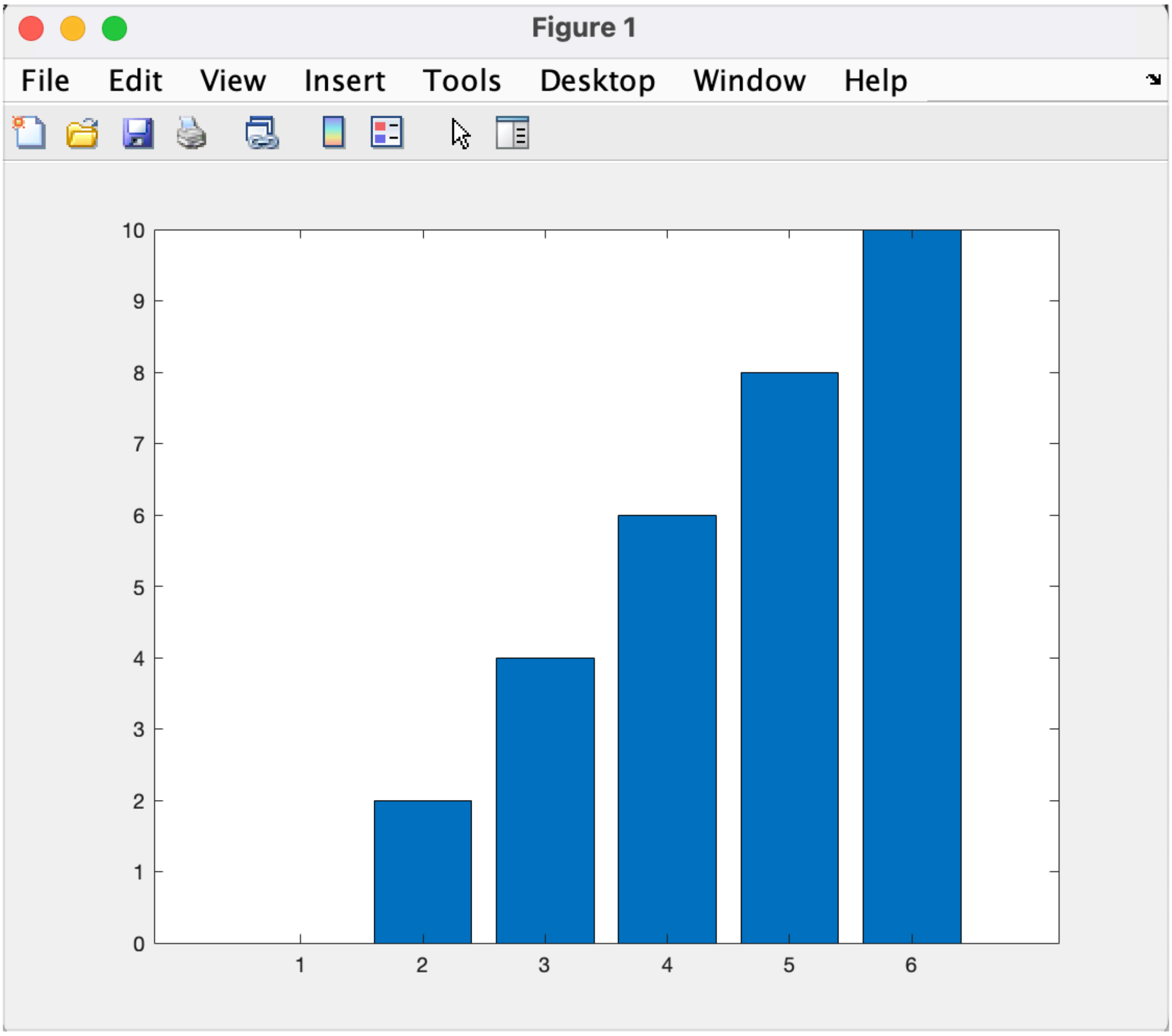


```
>> plot(x,y,'r',x,z,'m','linewidth',2),title('Sin / Cos
graph'),xlabel('theta'),ylabel('y'),legend('y1=sin(x)','
y2=cos(x)','location','southwest')
```



Plotting Bar graphs

```
 >> y=[0:2:10]

y =

     0     2     4     6     8    10

>> bar(y)
```



```
>> x=[1900:10:2000];
>> y=[90:15:250];
>> bar(x,y)
```

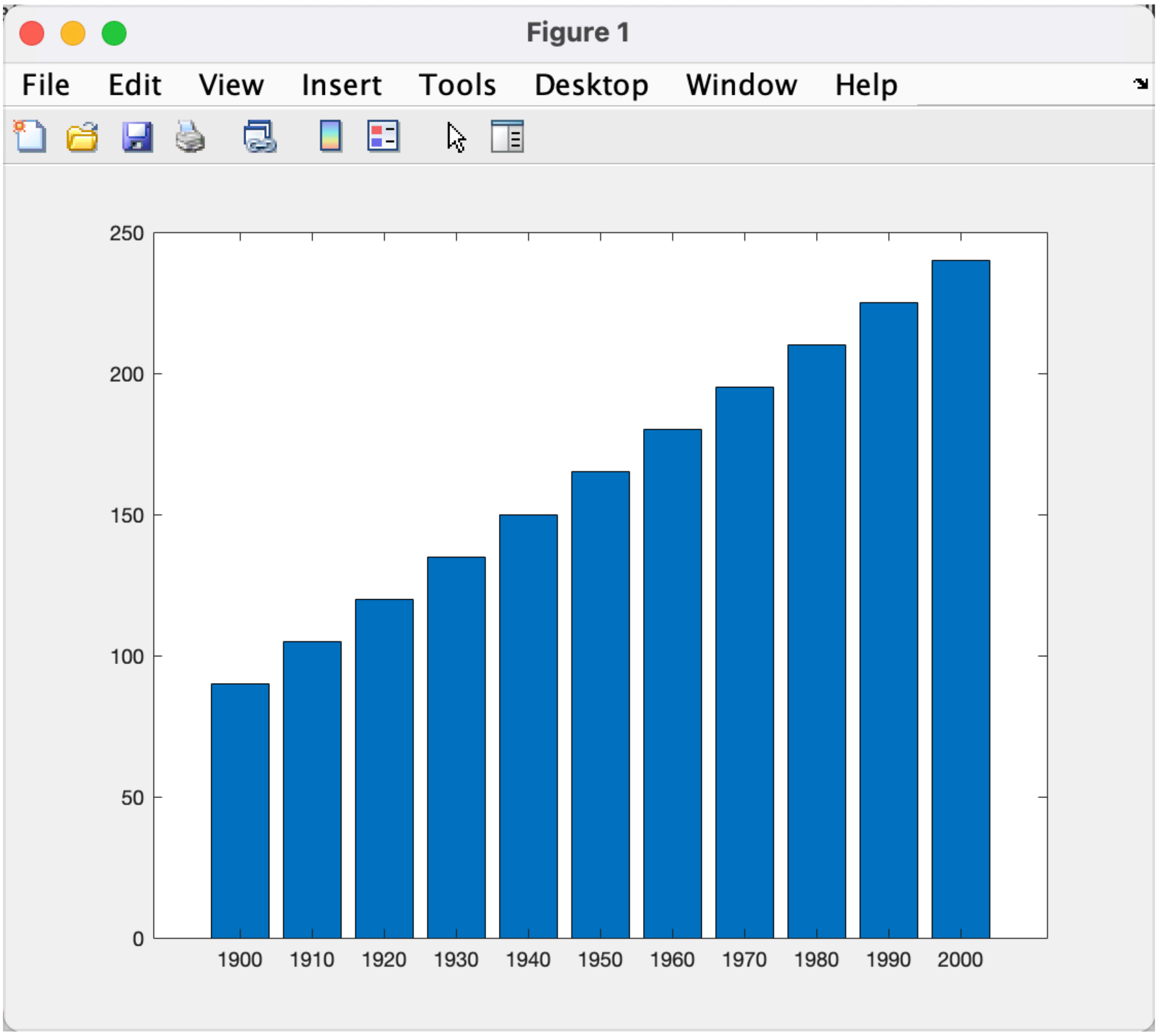
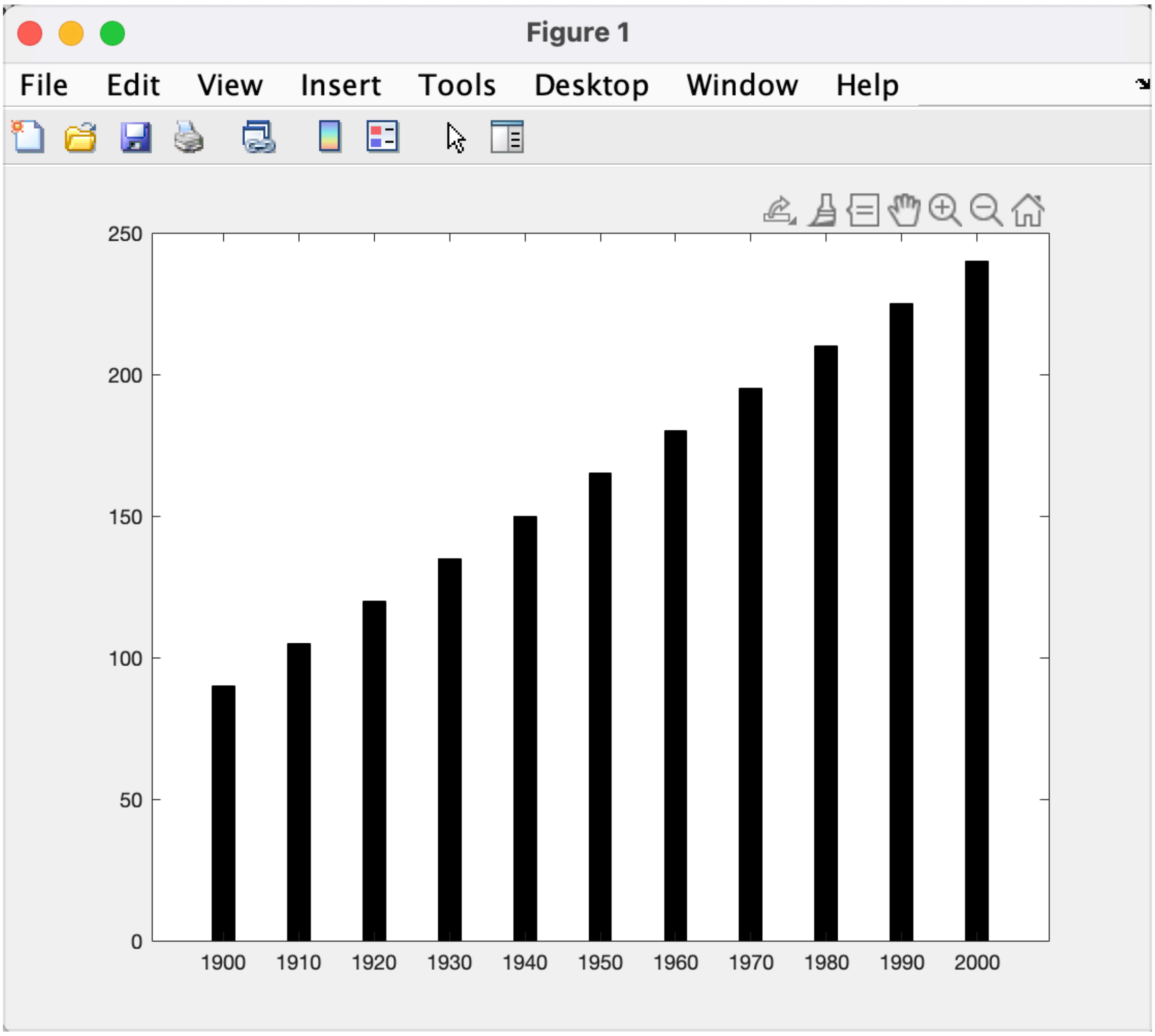


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
250
200
150
100
50
0
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000

```
>> bar(x,y,0.3,'k')
```



```
y =

     2     3     4
     3     2     4
     2     7     3

>> bar(y)
```

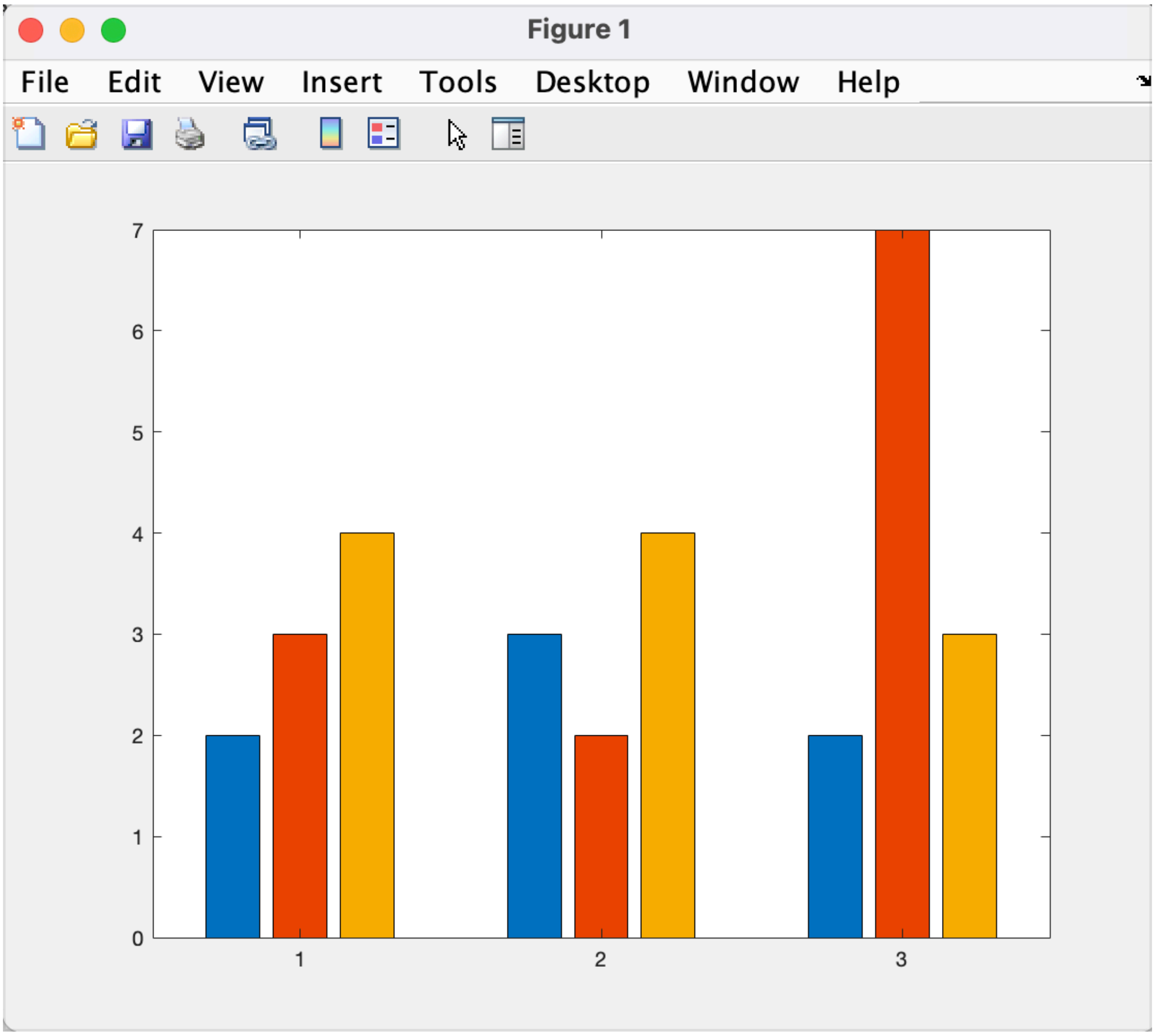
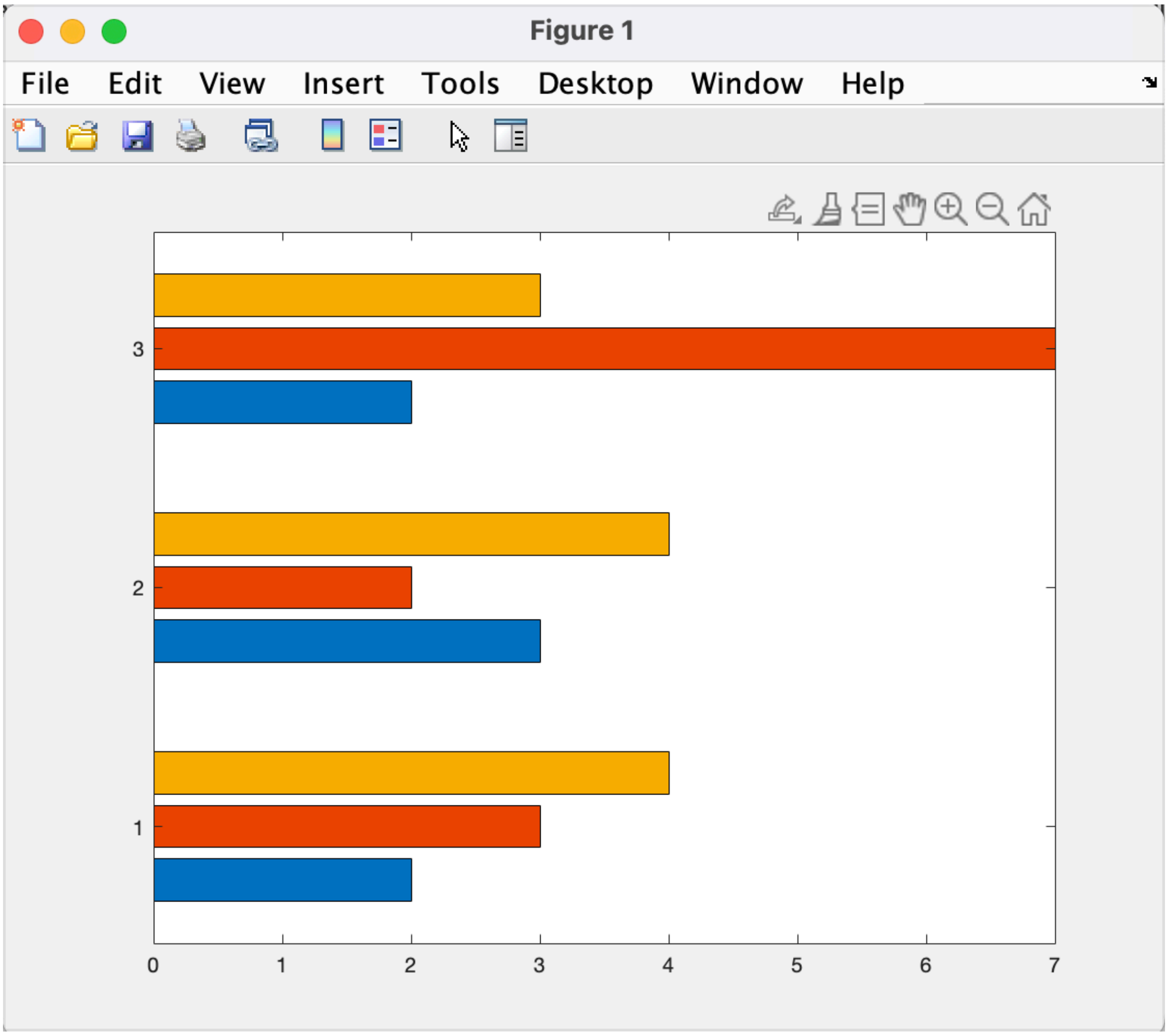


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
7
6
5
4
3
2
1
0
1
2
3

```
>> barh(y)
```



Plotting Pi graphs

```
>> pie(y)
>> y=[1,3,1,5];
>> pie(y)
```

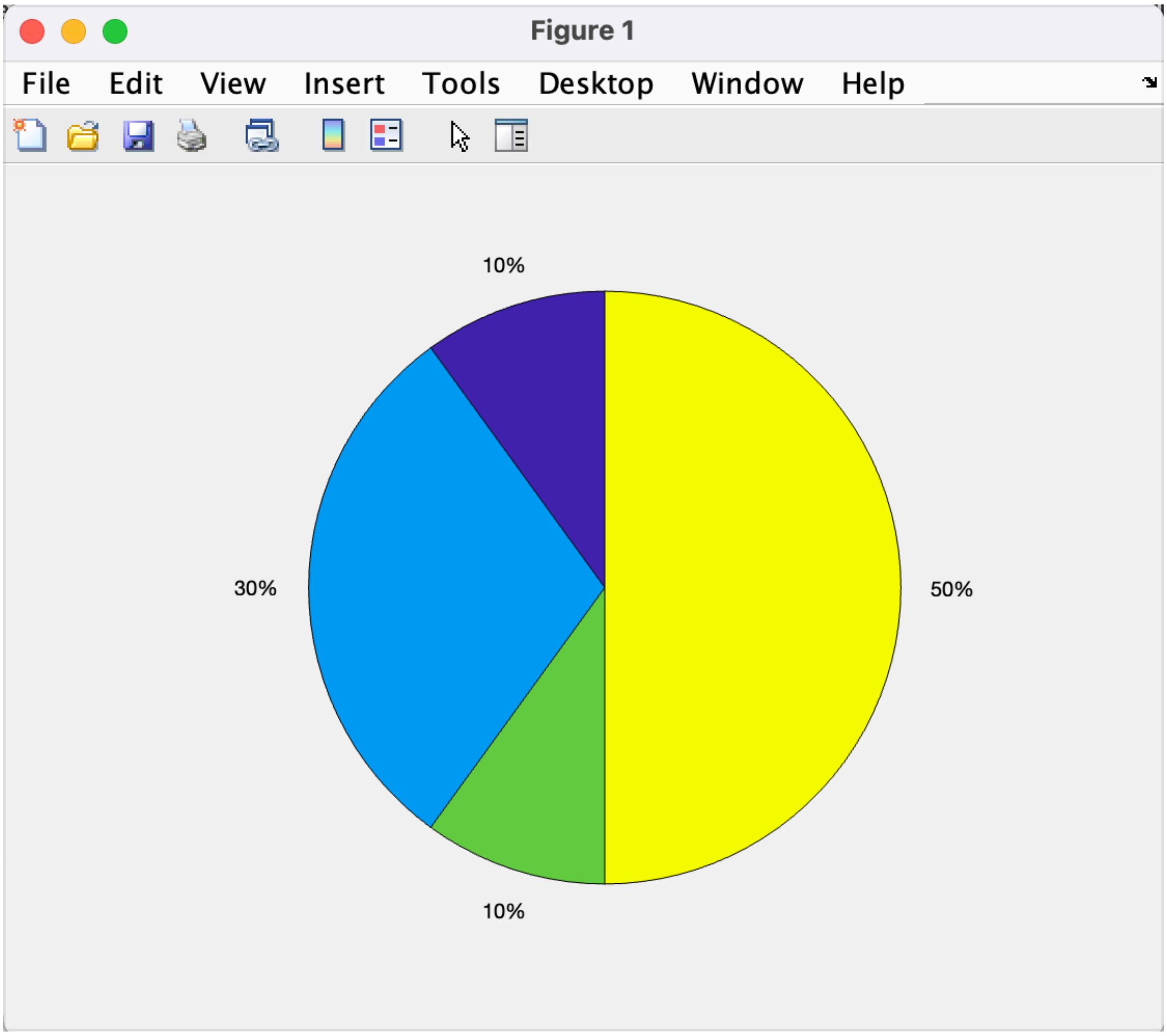
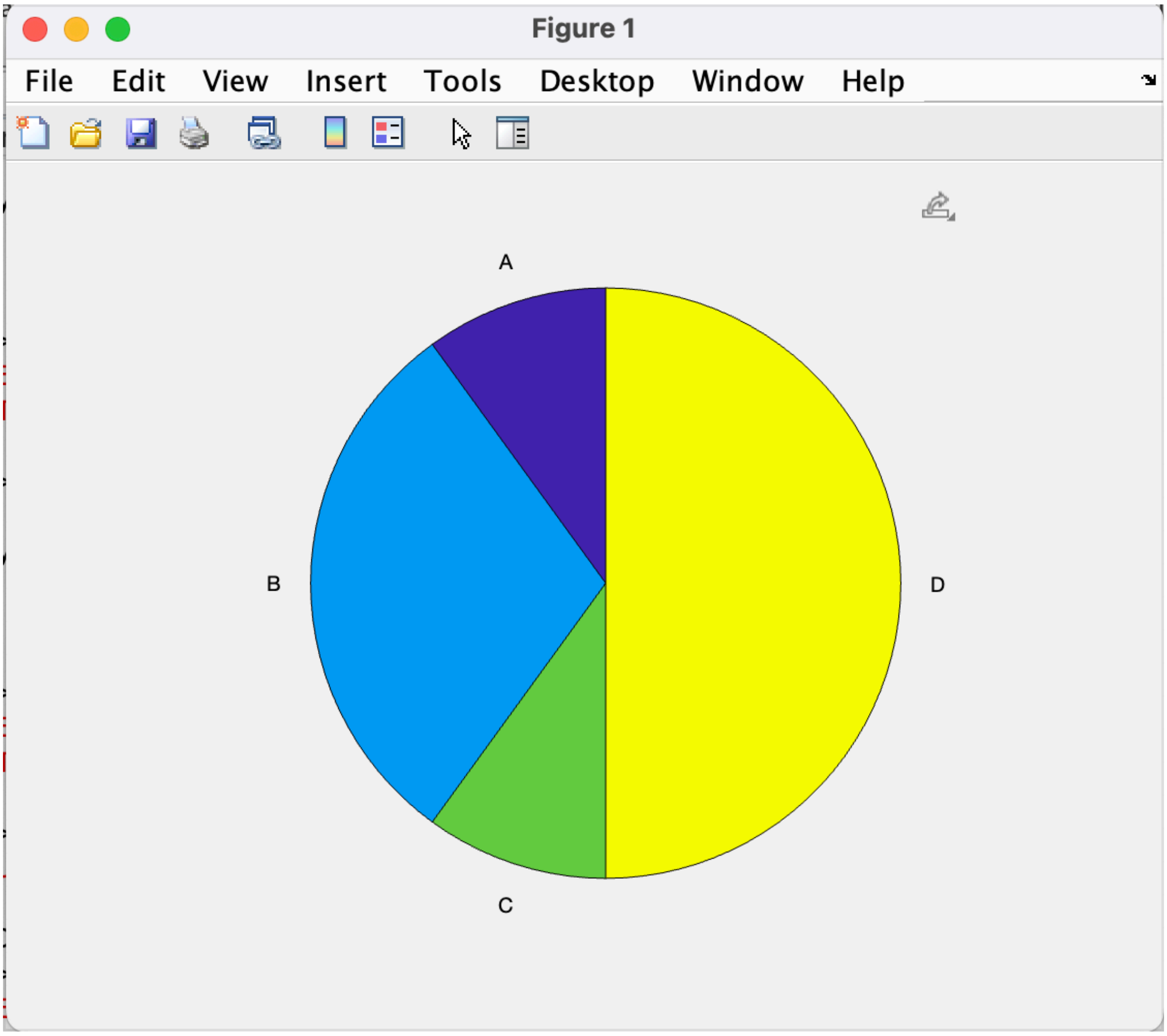
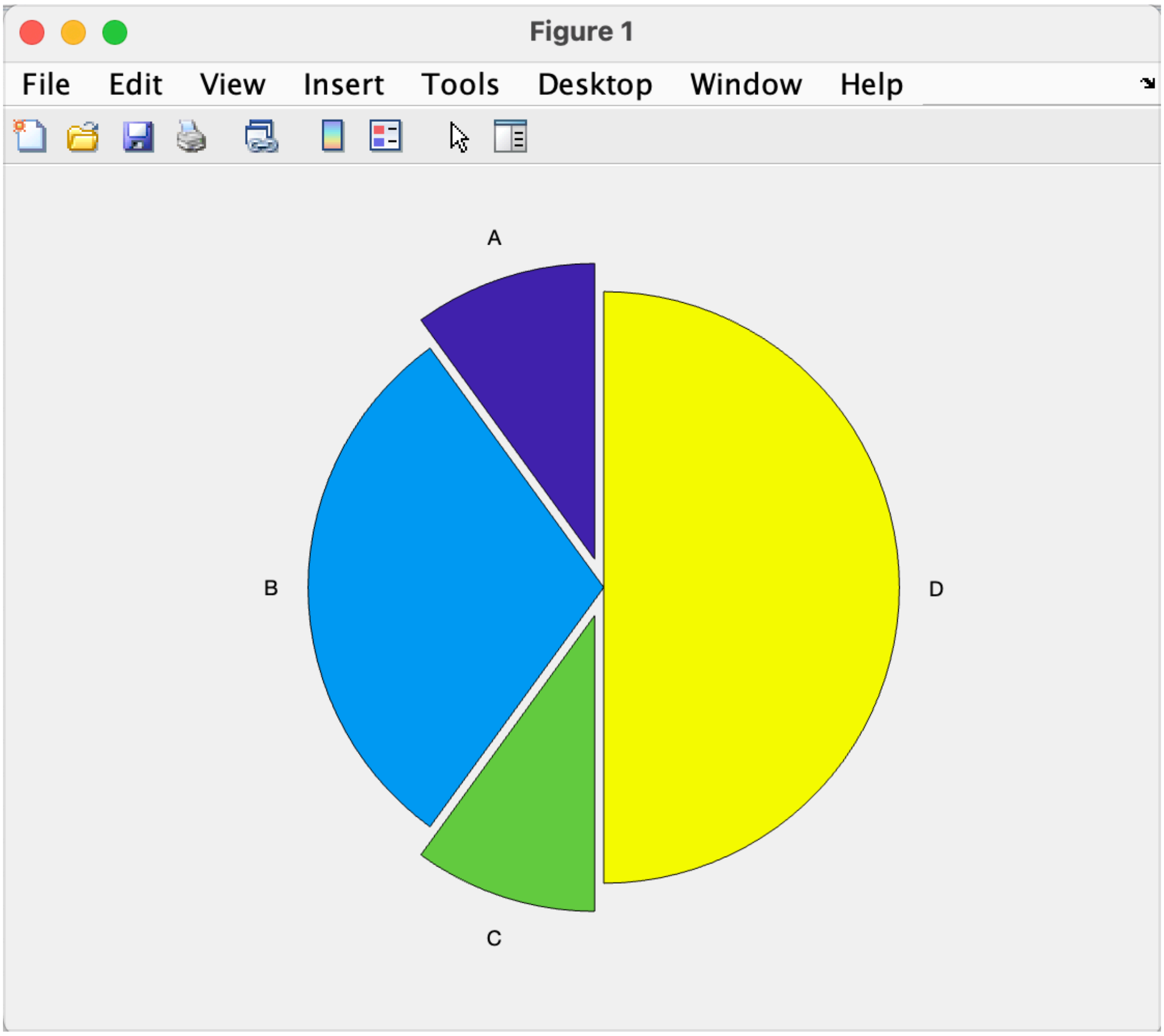


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
10%
30%
50%
10%

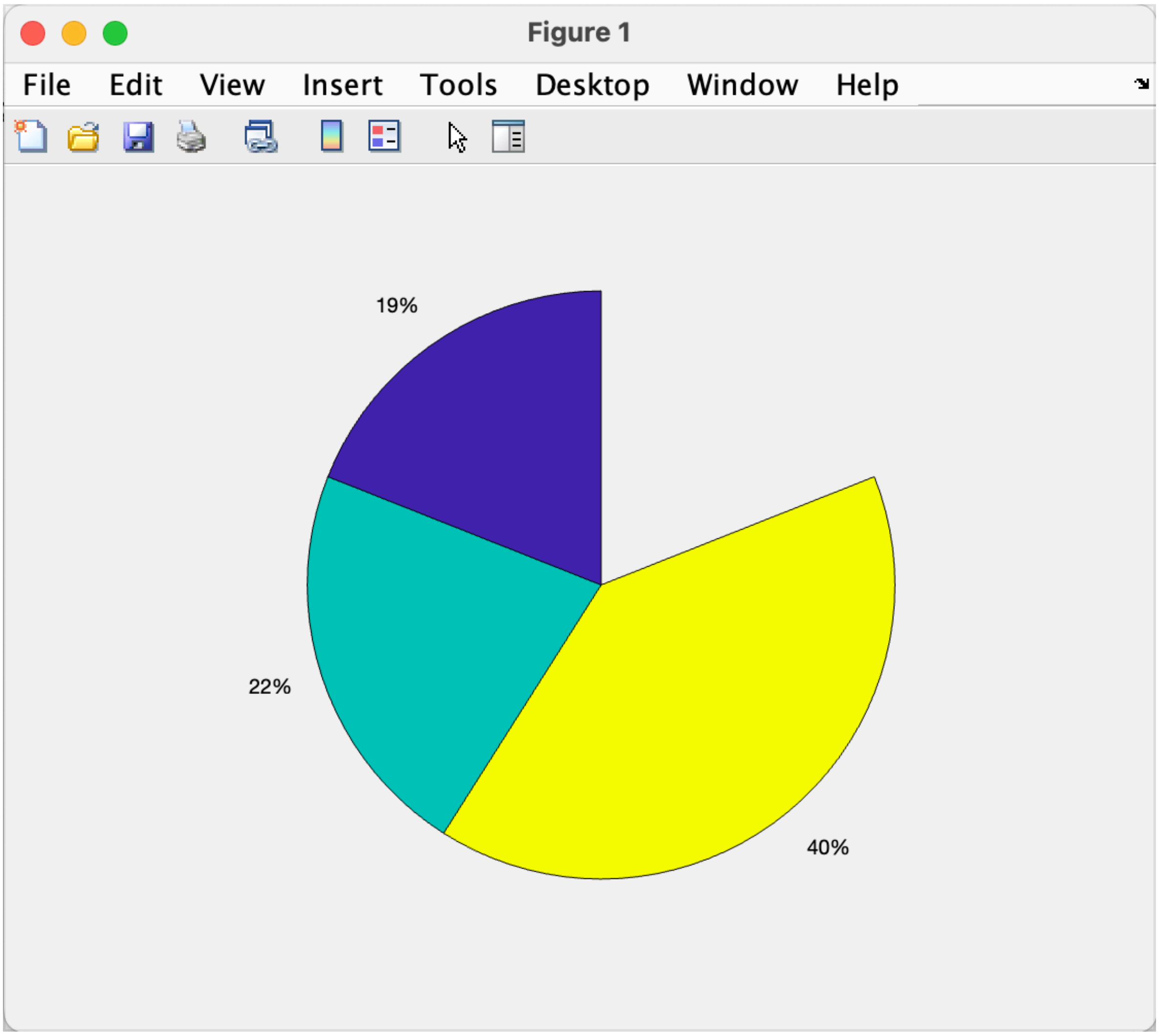
```
>> pie(y,{'A','B','C','D'})
```



```
>> y=[1,3,1,5];
>> label={'A','B','C','D'};
>> exp=[1,0,1,0];
>> pie(y,exp,label)
```



```
>> y=[0.19,0.22,0.4]

y =

    0.1900    0.2200    0.4000

>> pie(y)
```



Plotting Circle in MATLAB

```
>> theta=0:0.001:2*pi;
>> r=5;
>> x=r.*cos(theta);
>> y=r.*sin(theta);
>> plot(x,y)
```

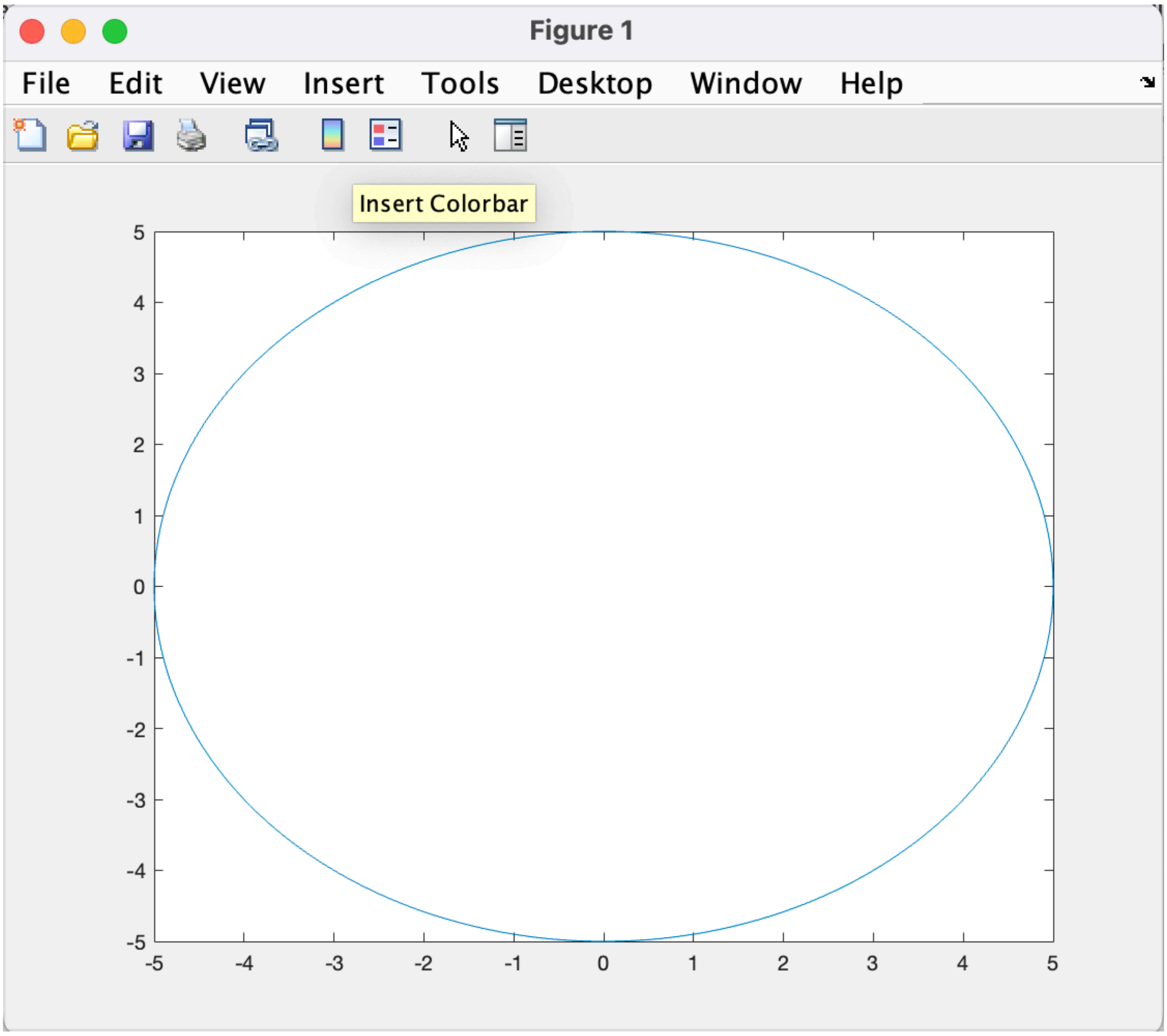
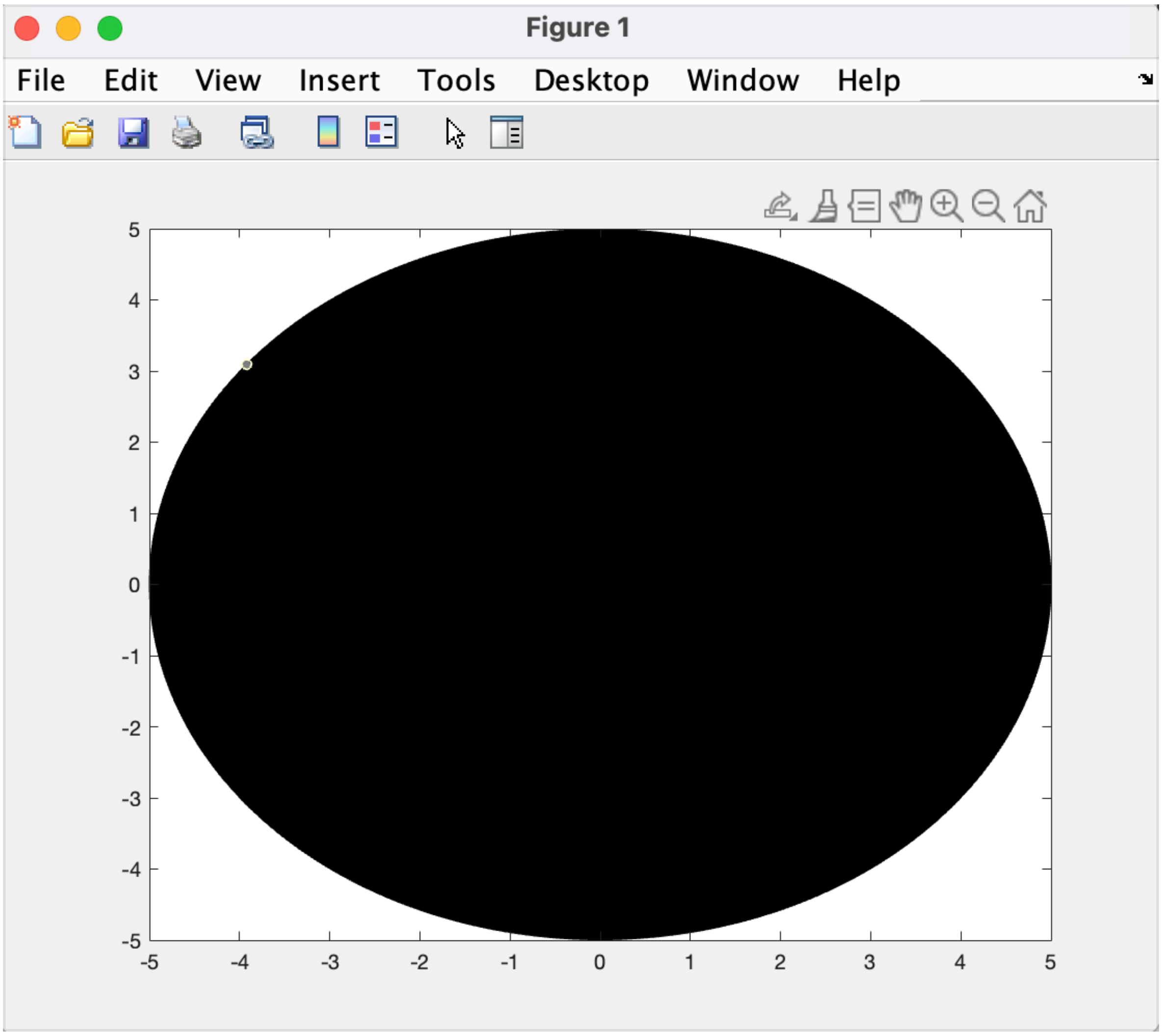


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Insert Colorbar
5
4
3
2
1
0
-1
-2
-3
-4
-5
-5
-4
-3
-2
-1
0
1
2
3
4
5

```
>> area(x,y)
```



How to creat Log plots

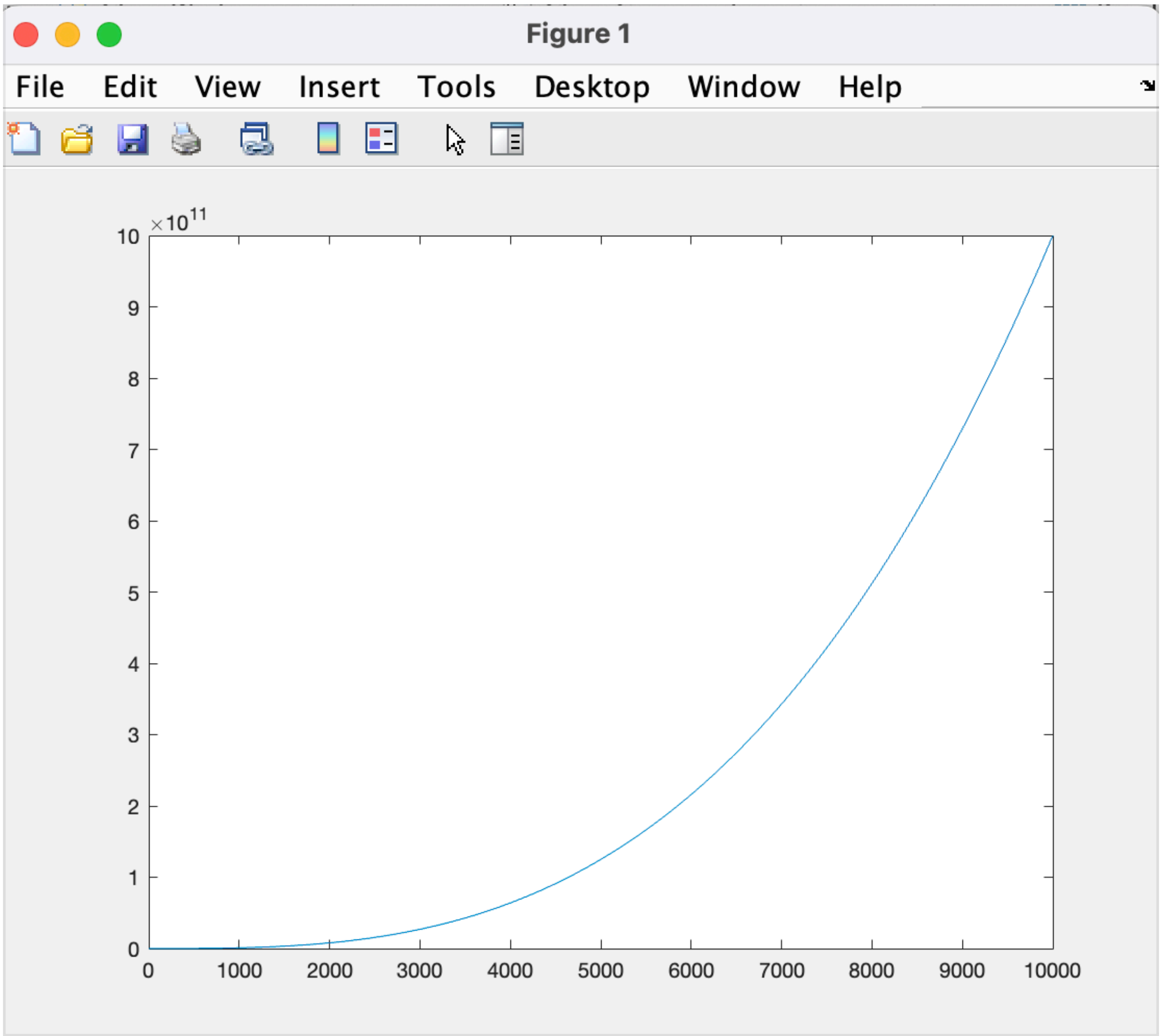
- Generate 3 plots

1. y=log(x) plot
2. semilog X & Y
3. semilog Y & X
4. log(x) vs log(y)

y=log(x) plot

- If your plotting graph of lets say y=10³ then use limit till 10,000 in linespace for better resolution

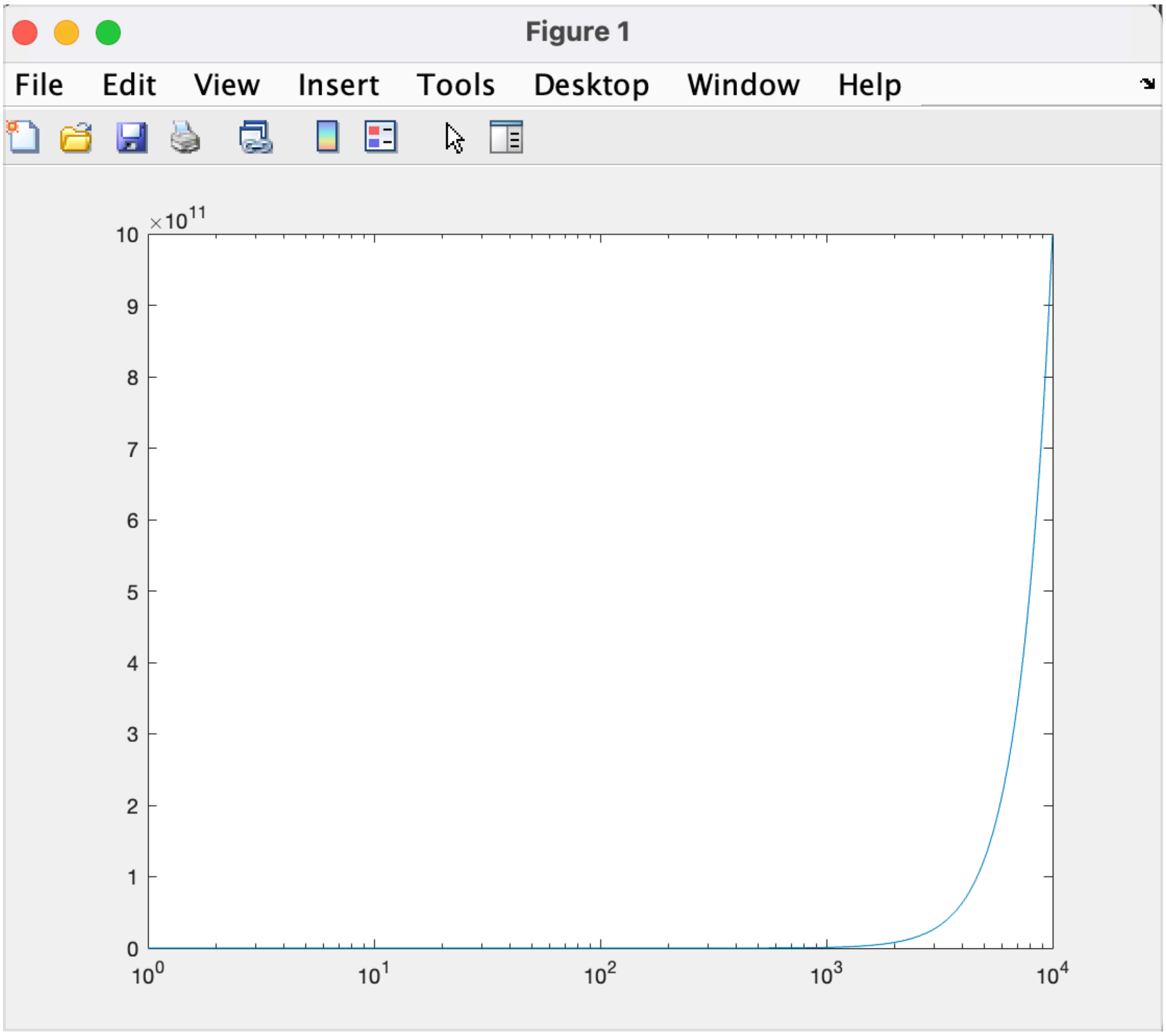
```
>> x=linspace(1,10000,100);
>> y=x.^3;
>> plot(x,y);
```



semilog X & Y

- semilogx gives the graph of Y vs log(x)

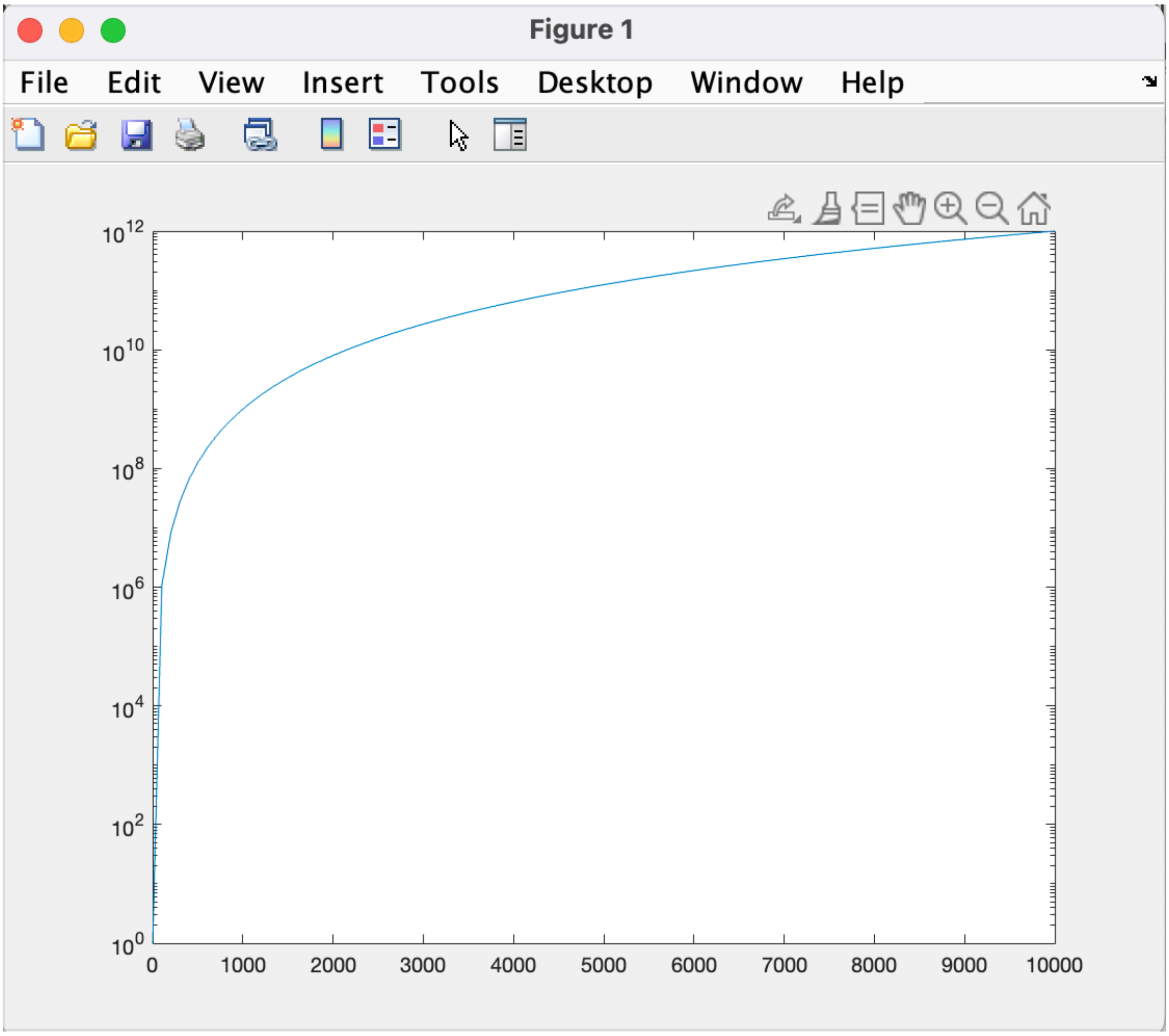
```
>> x=linspace(1,10000,100);
>> y=x.^3;
>> plot(x,y);
>> semilogx(x,y)
```



semiology Y & X

- semilogx gives the graph of log(y) vs X

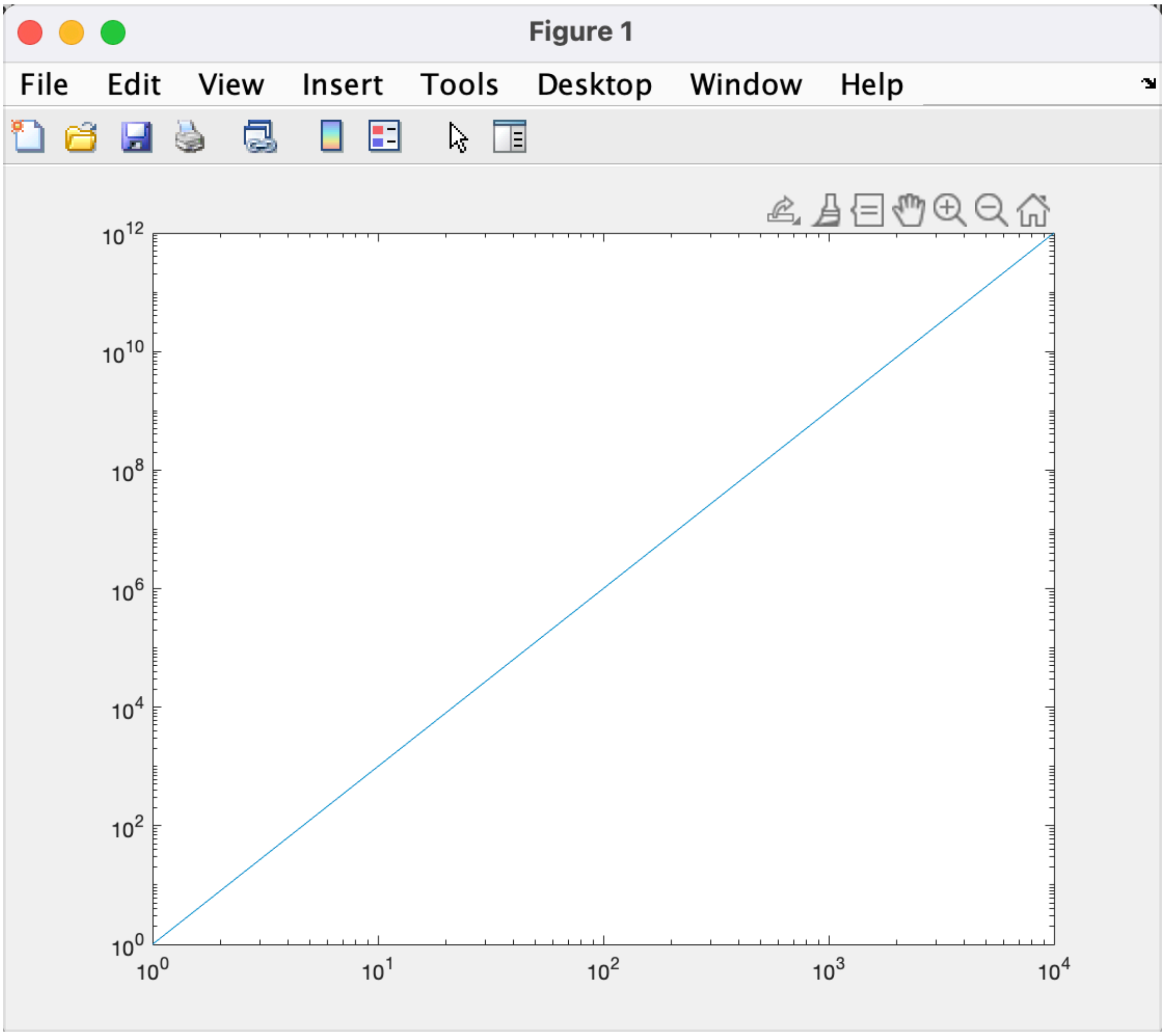
```
>> x=linspace(1,10000,100);
>> y=x.^3;
>> plot(x,y);
>> semilogy(x,y)
```



logX & logY

- loglog(x,y) gives the graph of log(y) vs log(x)

```
>> x=linspace(1,10000,100);
>> y=x.^3;
>> plot(x,y);
>> loglog(x,y)
```

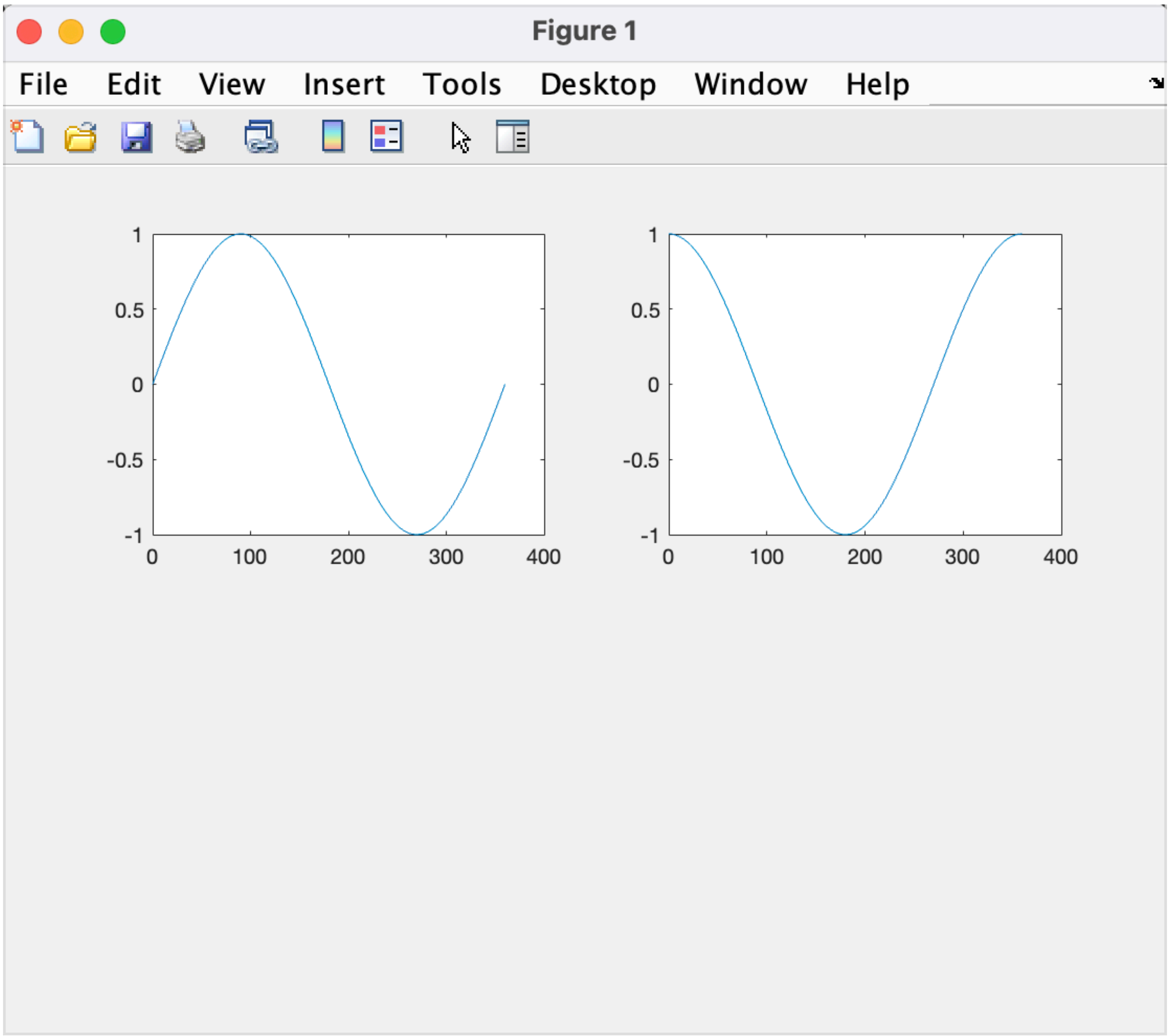


How to use Sub-Plot

- Sub-plots is used to display multiple plots in different sub-region.

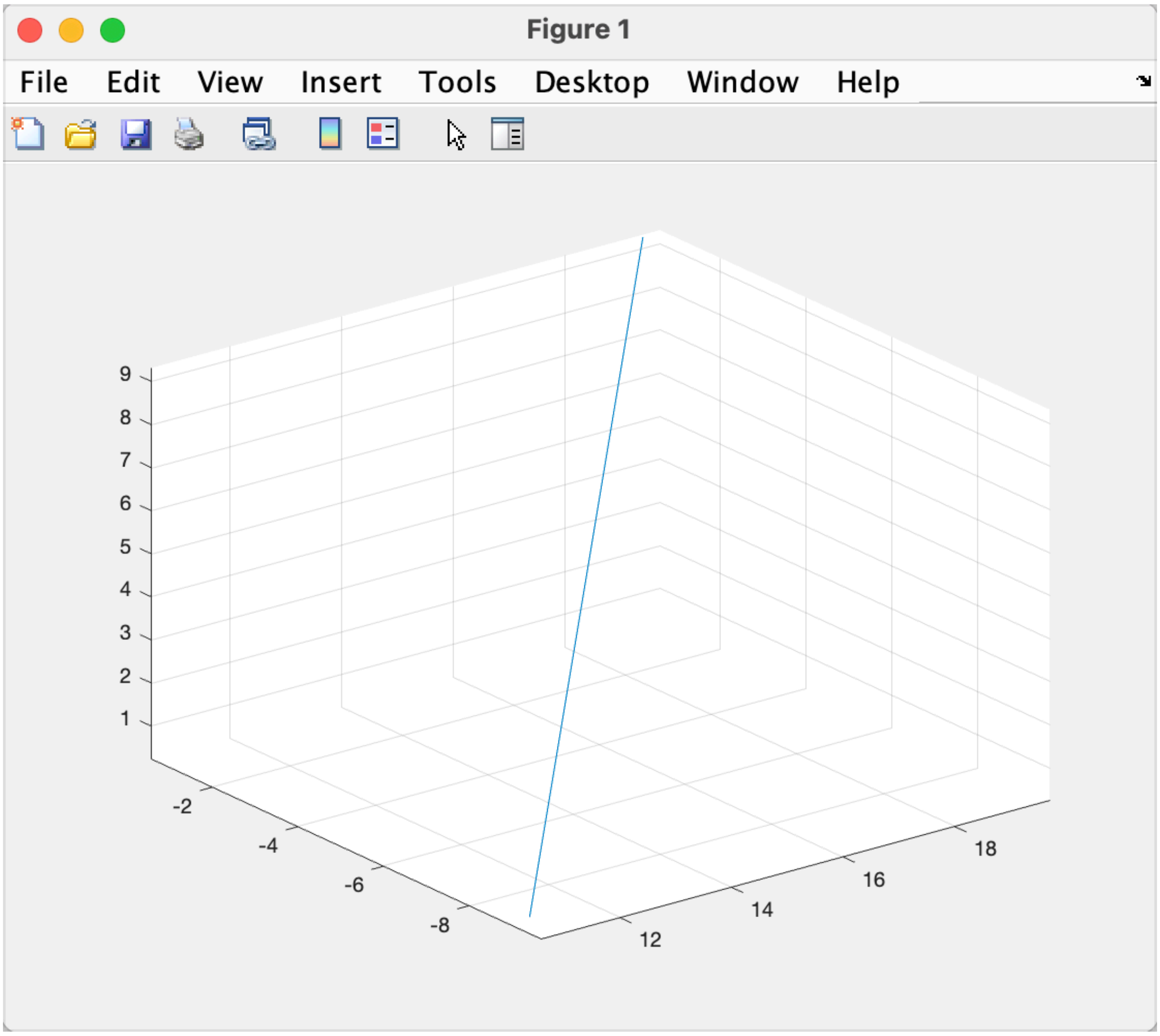
```
>> t=0:0.001:360;
>> y1=sind(t);
>> y2=cosd(t);
>> subplot(2,2,1),plot(t,y1),subplot(2,2,2),plot(t,y2);
```

- If we want four sub plates then use (2,2)
- To locate the position of graph in the plot use third value corresponds to the position of graph you want
- For example (2,2,1) mean top left square , (2,2,2) means top right square, (2,2,4) means bottom right square.



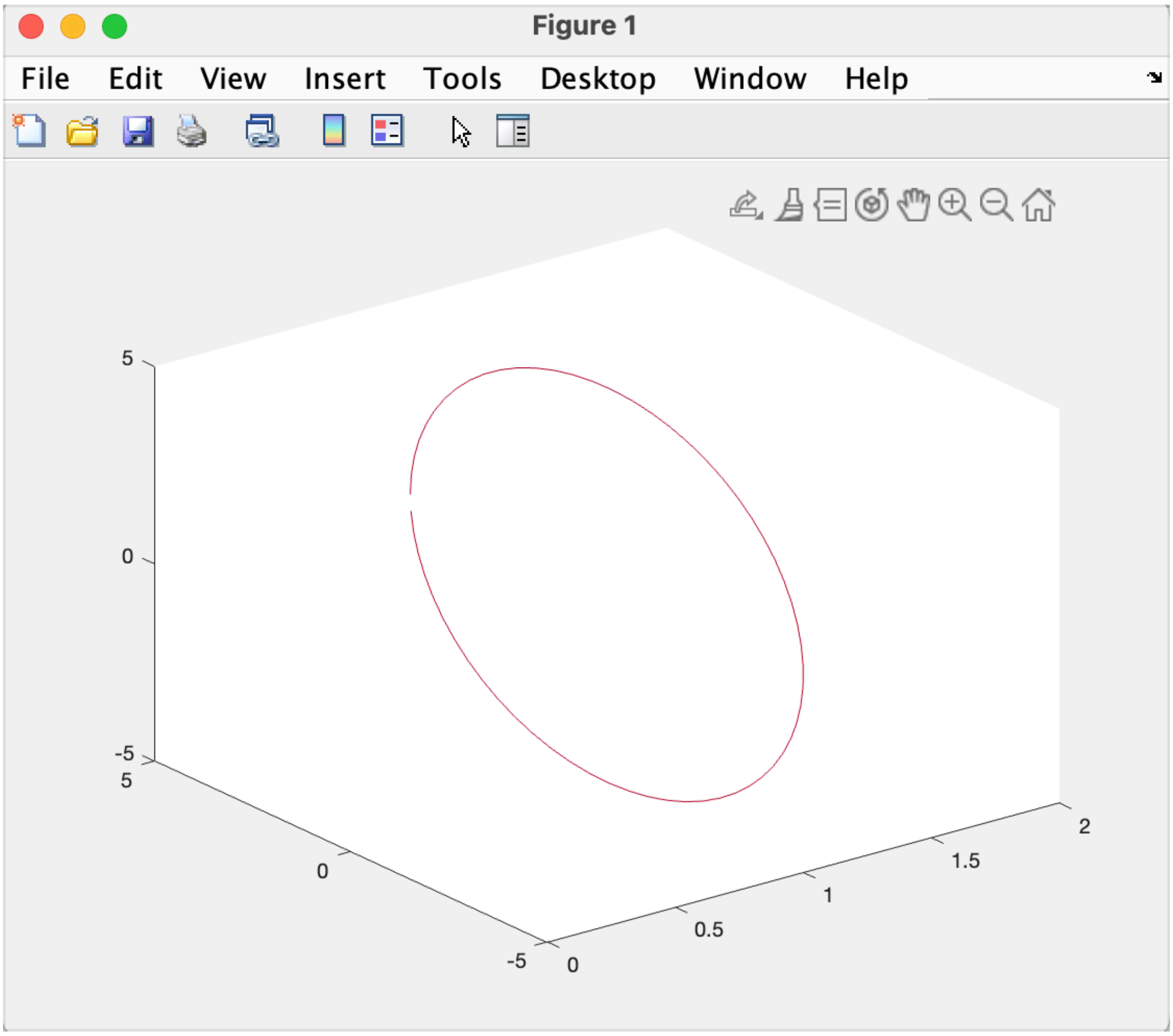
How to plot 3-D graph in MATLAB

```
>> t=0:0.01:10;
>> x=t+10;
>> y=t−10;
>> z=t;
>> plot3(x,y,z),grid on;
```



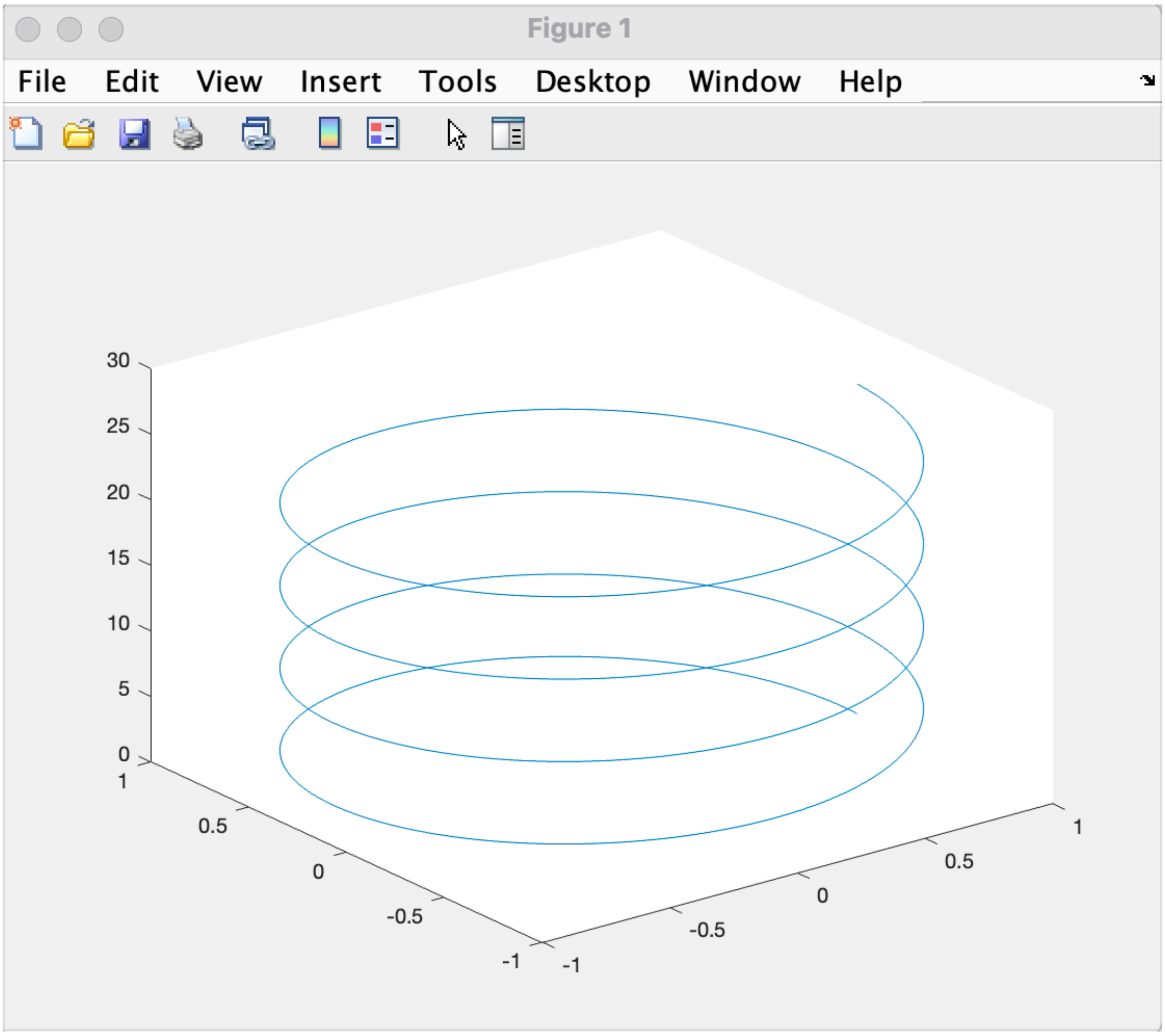
How to plot 3-D circle in MATLAB

```
>> t=0:0.1:2*pi;
>> x=ones(length(t));
>> y=5.*cos(t);
>> z=5.*sin(t);
>> plot3(x,y,z)
```



How to create Helix in MATLAB

```
>> t=0:0.01:8.*pi;
>> x=cos(t);
>> y=sin(t);
>> z=t;
>> plot3(x,y,z);
```

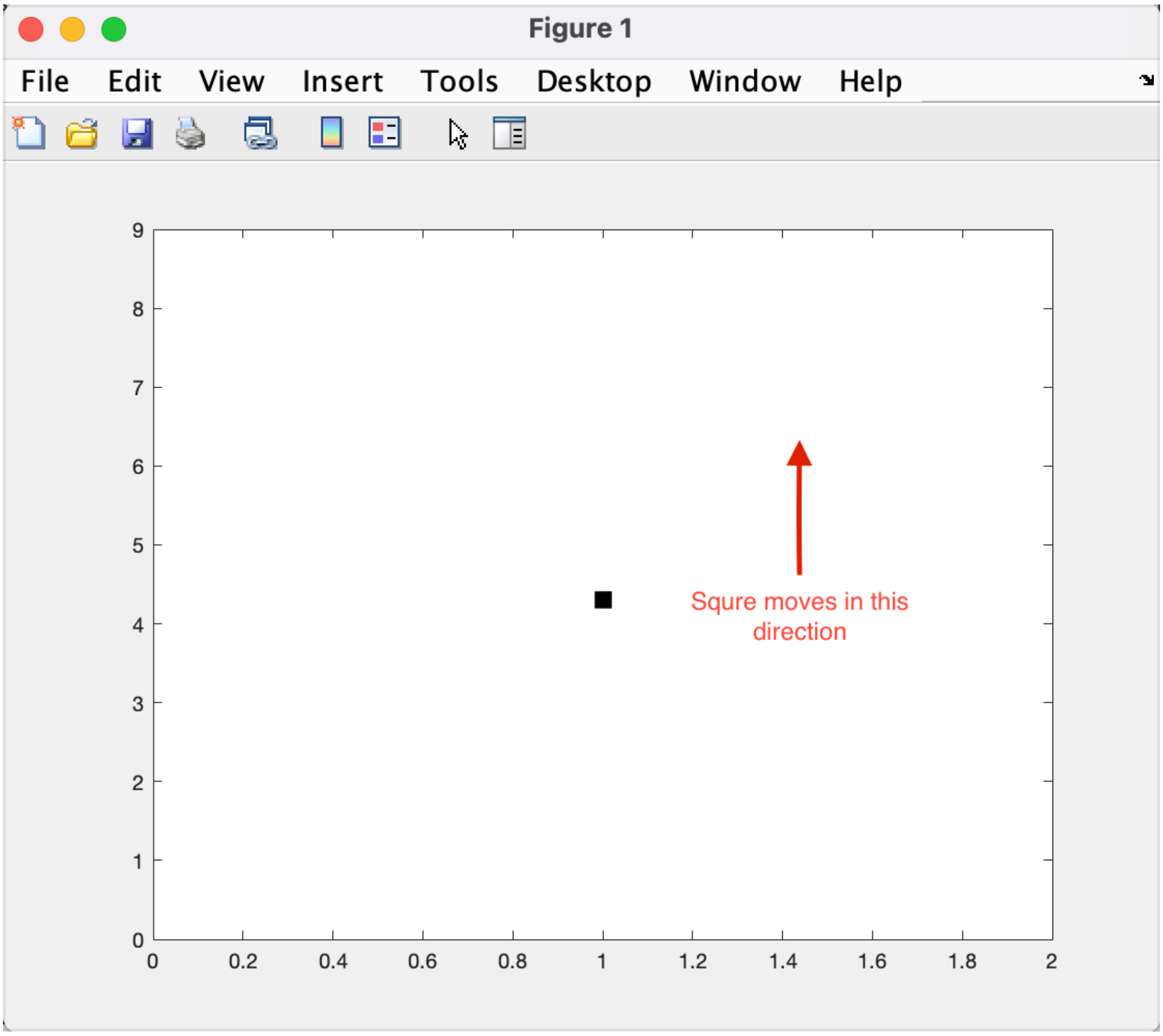


How to create animation in MATLAB

1. Point animation

- Having the animation function
- Animation function submerse with set of the points
- Plots each points
- Takes screenshot
- Display it

```
>> for t=0:0.1:9;
y=t;
plot(y,'s','MarkerSize',10,'MarkerFaceColor','k','Marker
EdgeColor','k');
ylim([0,9]);
anim(i)=getframe;
i=i+1;
end
```



Programming in MATLAB

Lecture overview

What is programming

- It is the process of creating software or website or application or games.
- Software is set of instruction(code) to perform a task.
- Instructions are not only written in natural language. Instead they are written in language that computer can understand (C, C++, C#, MATLAB)

Algorithms (different type of algorithms)

- An ordered sequence of precisely defined instructions that performs task in finite amount of time.

- Programming in MATLAB is more about building algorithms
- There are three categories of Algorithms operations:

1. Sequential operations: instructions executed in order
2. Conditional operation: first they ask us a question which we have to answer in true/ false and then select the next instruction.
3. Loop operation: that is structure that repeats the execution of block of instruction.

Structured programming

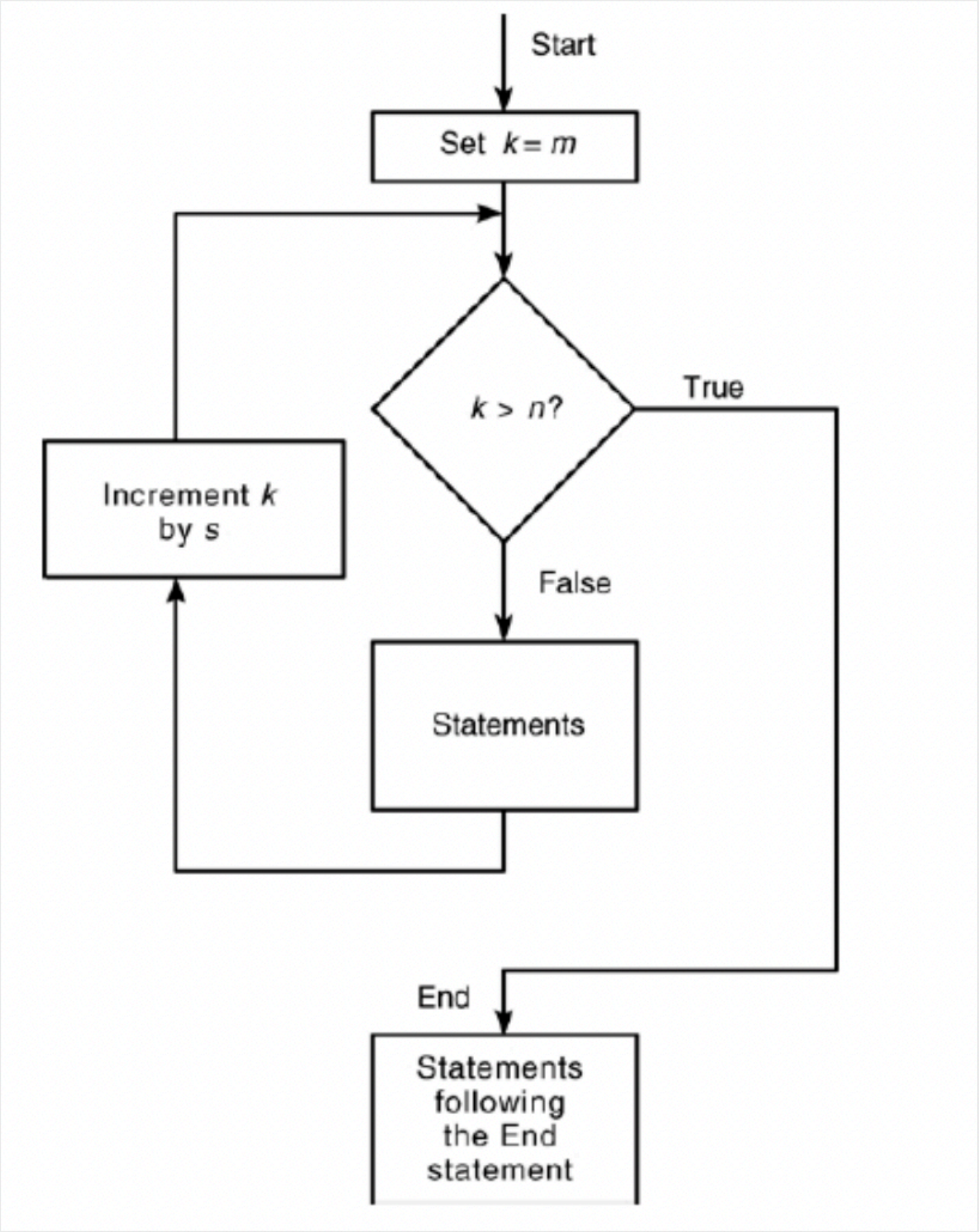
- A technique for defining programs in which a hierarchy of modules is used, each having a single term and single exit point.
- In MATLAB these module can be built-in or user-defined function.

Advantages of using structured programming

1. Easier to write: because the programmer can study the overall problem first and then deal with the details later
2. Reusable code: function written for a program can also use for other applications
3. Easier to debug: because each function (module) is made for separate task.
4. Effective in team work environment: because many people can work on common programm, each working on separate module.
5. Easier to understand and modify.

Flow charts

- Flow charts are type of diagrams that represent an algorithm, workflow or process showing the steps as boxes of various types.
- Flow charts for a FOR loop



Finding bugs

- Process of finding and fixing bugs.

1. Syntax error
2. Runtime errors: error due to incorrect mathematical procedure

Section overview

1. M-files
2. Input and output commands
3. fprintf Function in MATLAB

4. Functions (User and Built in)
5. The if, elseif and else statement
6. For & While loops
7. The switch structure
8. Logical Operators (and , or ,xor , not)

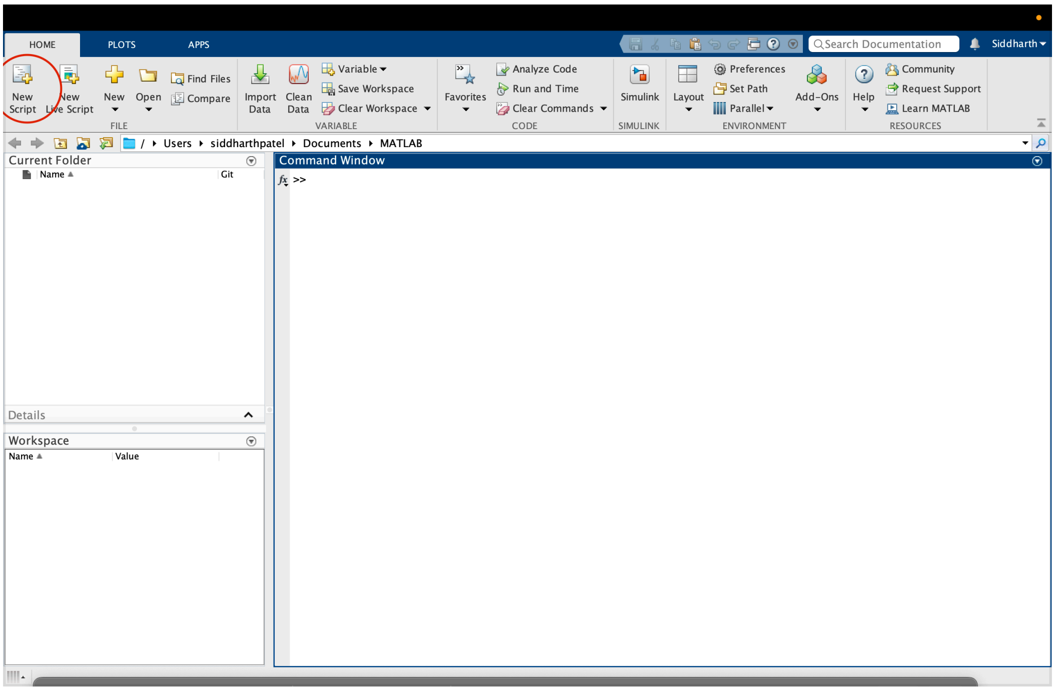
M- Files in MATLAB

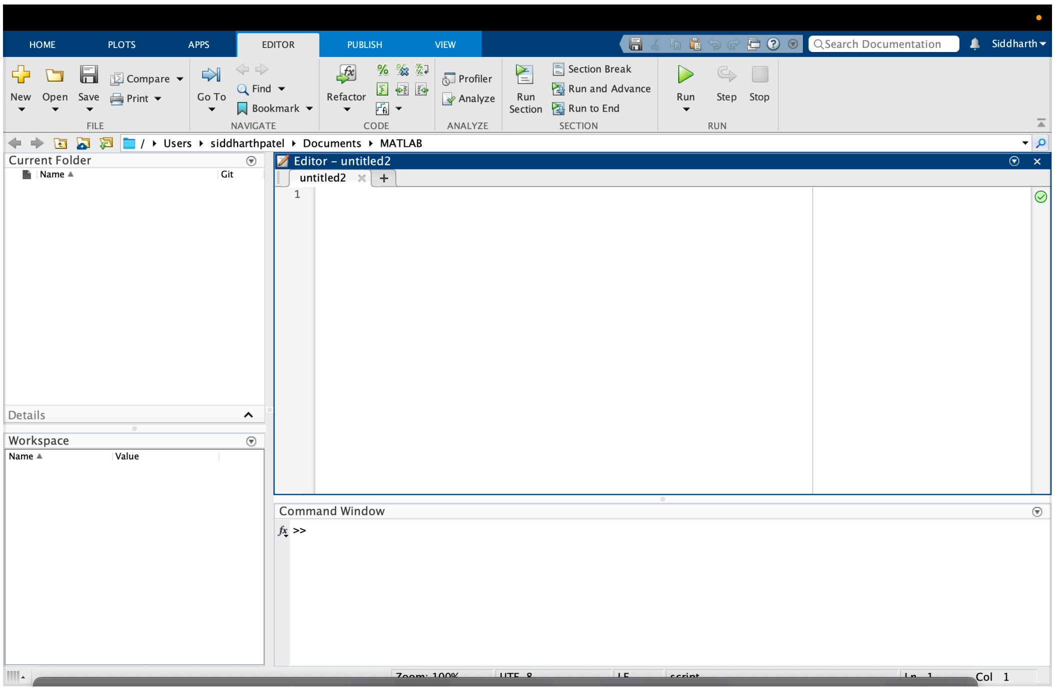
We can perform operations in MATLAB in two ways

1. In the interactive mode (writing command directly in command line) (what we were doing till now)
2. By running a MATLAB program stored in script M-files

What is script M-Files ?

1. Contains MATLAB commands
2. Where to write M- program





3. Things to keep in mind while creating M-files

- Do not give a script file name as a variable
- Do not give a script file name as MATLAB function or command

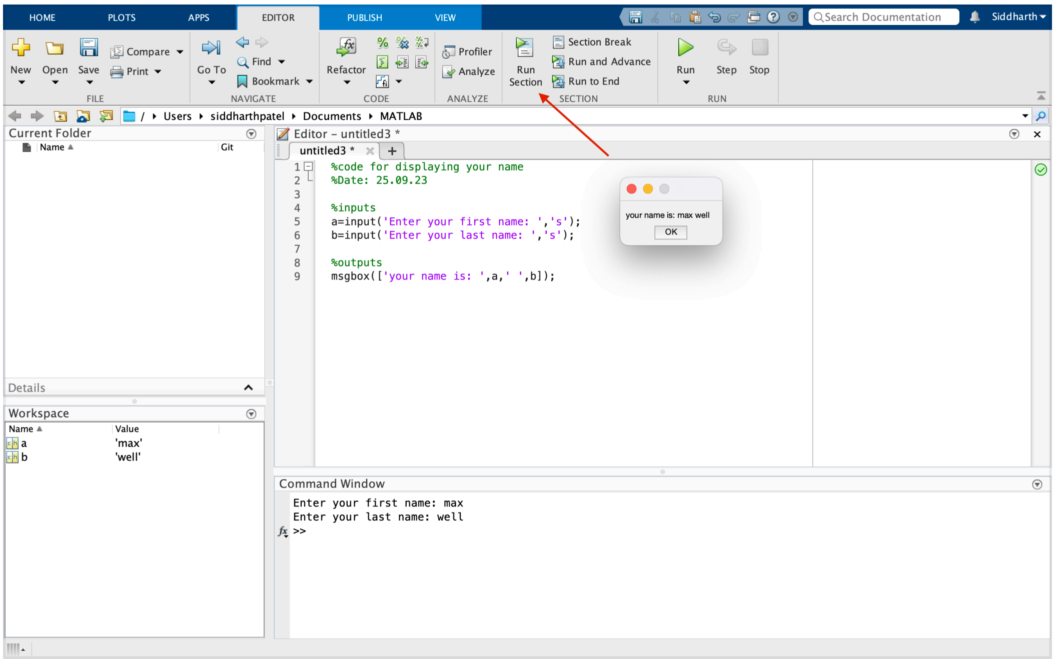
Programming Style

1. Comment section: use "%" sign
2. Input Section: (input data, input function, may be comments)
3. Calculation section:: formula used and the units of measurements
4. Output section: contains functions of displaying outputs

Ex: write and code that display your name

```
%code for displaying your name
%Date: 25.09.23

%inputs
a=input('Enter your first name: ','s');
b=input('Enter your last name: ','s');

%outputs
msgbox(['your name is: ',a,' ',b]);
```

- To run the code press on "Run Selection" button or type the file name in command window

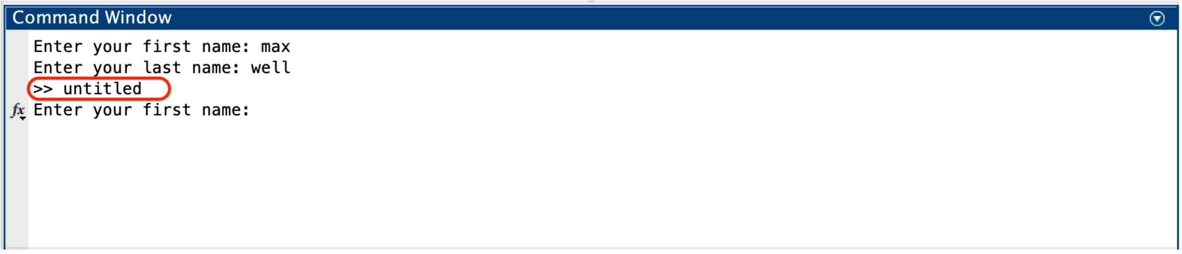


```
%code for displaying your name
%Date: 25.09.23

%inputs
a=input('Enter your first name: ','s');
b=input('Enter your last name: ','s');

%outputs
msgbox(['your name is: ',a,' ',b]);
```

your name is: max well

Name	Value
a	'max'
b	'well'

```
Enter your first name: max
Enter your last name: well
>>
```



Command Window

```
Enter your first name: max
Enter your last name: well
>> untitled
Enter your first name:
```

Inputs & Outputs Commands In MATLAB

Some of Inputs & Outputs Commands in MATLAB

1 - Input Commands

x = input ('Please Enter the Value of X')

Displays the text in quotes, waits for user input from the keyboard, and stores the value in x.

x = input ('Please Enter your name','s')

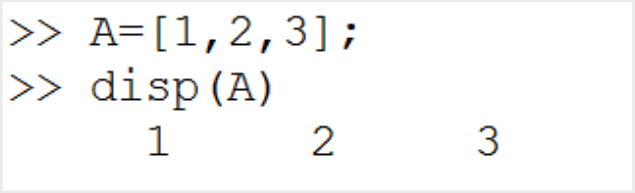
Displays the text in quotes, waits for user input from the keyboard, and stores the input as a string in x. So the s in the single quotation marks tells MATLAB that the input type is string.

2 - Output Commands

disp (A)

Displays the contents, but not the name, of the array A.

Example

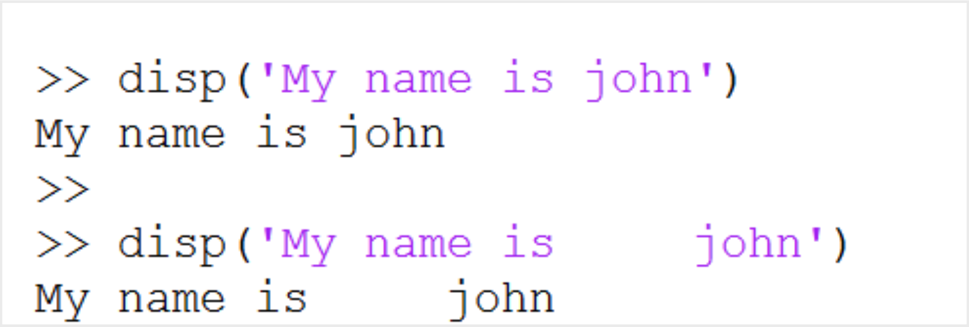


```
>> A=[1,2,3];
>> disp(A)
     1     2     3
```

disp ('text')

Displays the text string enclosed within quotes.

Example



```
>> disp('My name is john')
My name is john
>>
>> disp('My name is     john')
My name is     john
```

In The Switch Structure Lecture the input & output commands are used, so this lecture is explained again in details in The Switch Structure Lecture.

fprint Function in MATLAB

- One of the most important output function in MATLAB
- It can be used to print numbers , texts, variables tables and so on...