Lecture 2

```
1\2
```

```
ans = 2
```

```
-5^2
```

```
ans = -25
```

```
(-5)^2
```

```
ans = 25
```

```
10-6/2
```

```
ans = 7
```

```
5*4/2*3
```

```
ans = 30
```

```
abs(-6)
```

```
ans = 6
```

```
abs = 11
```

```
abs = 11
```

```
clear abs
abs(-6)
```

```
ans = 6
```

```
plus(2,5)
```

```
ans = 7
```

```
3 == 5 + 2
```

```
ans = logical
   0
```

```
'b' < 'a'+1
```

```
ans = logical
   0
```

```
10 > 5 + 2
```

```
ans = logical
   1
```

```

(10 > 5) + 2
```

```
ans = 3
```

```
'c' == 'd' - 1 && 2 < 4
```

```
ans = logical
   1
```

```
'c' == 'd' - 1 || 2 > 4
```

```
ans = logical
   1
```

```
xor ( 'c' == 'd' - 1, 2> 4)
```

```
ans = logical
   1
```

```
xor ('c' == 'd' - 1, 2 <4)
```

```
ans = logical
   0
```

```
10 > 5> 2
```

```
ans = logical
   0
```

Lecture 3

```
v = [1 2 3 4]
```

```
v = 1×4
     1     2     3     4
```

```
v = [1,2,3,4]
```

```
v = 1×4
     1     2     3     4
```

```
vec = 2:6
```

```
vec = 1×5
     2     3     4     5     6
```

```
nv = 1:2:9
```

```
nv = 1×5
     1     3     5     7     9
```

```
ls = linspace(3,15,5)
```

```
ls = 1×5
     3     6     9    12    15
```

```
newvec = [nv ls]
```

```
newvec = 1×10
     1     3     5     7     9     3     6     9    12    15
```

```
logspace(1,4,4)
```

```
ans = 1×4
          10         100        1000       10000
```

Lecture 4

```
load('durer')
load('durer','X','map')
format short
x= (1:10)'
```

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

```
logs = [x log10(x)]
```

```
logs = 10×2
    1.0000         0
    2.0000    0.3010
    3.0000    0.4771
    4.0000    0.6021
    5.0000    0.6990
    6.0000    0.7782
    7.0000    0.8451
    8.0000    0.9031
    9.0000    0.9542
   10.0000    1.0000
```

```
headers = ['Author Last Name, Author First Name,' ...
'Author Middle Initial']
```

```
headers = 
'Author Last Name, Author First Name,Author Middle Initial'
```

Lecture 5

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

```
matrix = 3×3
     1     2     3
     4     5     6
     7     8     9
```

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

```
A = 2×3
     2    -3     5
    -1     4     5
```

```
x = [1 4 7]
```

```
x = 1×3
     1     4     7
```

```
x = [1;4;7]
```

```
x = 3×1
     1
     4
     7
```

```
cd=6; e=3; h=4;
Mat=[e cd*h cos(pi/3) ; h^2 sqrt(h*h/cd) 14]
```

```
Mat = 2×3
    3.0000   24.0000    0.5000
   16.0000    1.6330   14.0000
```

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

```
a = 2×2
     1     2
     3     4
```

```
b = [4 6;8 9]
```

```
b = 2×2
     4     6
     8     9
```

```
A = [a,b]
```

```
A = 2×4
     1     2     4     6
     3     4     8     9
```

```
B = [a;b]
```

```
B = 4×2
     1     2
     3     4
     4     6
     8     9
```

```
x = [1:2:10]
```

```
x = 1×5
     1     3     5     7     9
```

```
y = 1:5
```

```
y = 1×5
     1     2     3     4     5
```

```
x = 1:5:50
```

```
x = 1×10
     1     6    11    16    21    26    31    36    41    46
```

```
1:5:50
```

```
ans = 1×10
     1     6    11    16    21    26    31    36    41    46
```

```
x = 9:-2:1
```

```
x = 1×5
     9     7     5     3     1
```

```
a = zeros(4,3)
```

```
a = 4×3
     0     0     0
     0     0     0
     0     0     0
     0     0     0
```

```
b = ones(4,3)
```

```
b = 4×3
     1     1     1
     1     1     1
     1     1     1
     1     1     1
```

```
c = rand(4,3)
```

```
c = 4×3
    0.1966    0.3517    0.9172
    0.2511    0.8308    0.2858
    0.6160    0.5853    0.7572
    0.4733    0.5497    0.7537
```

```
d = eye(4)
```

```
d = 4×4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
e = magic(4)
```

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

```
mat = [4 3 1;2 5 6]
```

```
mat = 2×3
     4     3     1
     2     5     6
```

```
mat = [2:4;3:5]
```

```
mat = 2×3
     2     3     4
     3     4     5
```

```
intmat = [100 77;28 14]
```

```
intmat = 2×2
   100    77
    28    14
```

```
intmat(1)
```

```
ans = 100
```

```
intmat(2)
```

```
ans = 28
```

```
intmat(3)
```

```
ans = 77
```

```
intmat(4)
```

```
ans = 14
```

```
vec = -2:1
```

```
vec = 1×4
    -2    -1     0     1
```

```
length(vec)
```

```
ans = 4
```

```
size(vec)
```

```
ans = 1×2
     1     4
```

```
mat = [1:3;5:7]'
```

```
mat = 3×2
     1     5
     2     6
     3     7
```

```
[r,c] = size(mat)
```

```
r = 3
c = 2
```

```
size(mat)
```

```
ans = 1×2
     3     2
```

```
zeros(size(mat))
```

```
ans = 3×2
     0     0
     0     0
     0     0
```

```
v = 9:-2:1
```

```
v = 1×5
     9     7     5     3     1
```

```
numel(v)
```

```
ans = 5
```

```
mat = [1:3;44 9 2;5:-1:3]
```

```
mat = 3×3
     1     2     3
    44     9     2
     5     4     3
```

```
mat(3,2)
```

```
ans = 4
```

```
mat(2,:)
```

```
ans = 1×3
    44     9     2
```

```
size(mat)
```

```
ans = 1×2
     3     3
```

```
mat(:,4) = [8;11;33]
```

```
mat = 3×4
     1     2     3     8
    44     9     2    11
     5     4     3    33
```

```
numel(mat)
```

```
ans = 12
```

```
v = mat(3,:)
```

```
v = 1×4
     5     4     3    33
```

```
v(v(2))
```

```
ans = 33
```

```
v(1) = []
```

```
v = 1×3
     4     3    33
```

```
reshape(mat,2,6)
```

```
ans = 2×6
     1     5     9     3     3    11
    44     2     4     2     8    33
```

Lecture 6

```
mat = [1 :3; 44 9 2; 5:-1:3]
```

```
mat = 3×3
     1     2     3
    44     9     2
     5     4     3
```

```
max(mat)
```

```
ans = 1×3
    44     9     3
```

```
max(max(mat))
```

```
ans = 44
```

```
cumsum(mat)
```

```
ans = 3×3
     1     2     3
    45    11     5
    50    15     8
```

```
cummin(mat)
```

```
ans = 3×3
     1     2     3
     1     2     2
     1     2     2
```

```
cumprod(mat)
```

```
ans = 3×3
     1     2     3
    44    18     6
   220    72    18
```

```
mat = randi(20,2,3)
```

```
mat = 2×3
     8     9     5
    11     2     3
```

```
diff(mat)
```

```
ans = 1×3
     3    -7    -2
```

```
zeros(5)*10
```

```
ans = 5×5
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

```
vec = [2:12]
```

```
vec = 1×11
     2     3     4     5     6     7     8     9    10    11    12
```

```
vec = vec - 3
```

```
vec = 1×11
    -1     0     1     2     3     4     5     6     7     8     9
```

```
mat = [1 :3; 44 9 2; 5:-1:3]
```

```
mat = 3×3
     1     2     3
    44     9     2
     5     4     3
```

```
mat/3
```

```
ans = 3×3
    0.3333    0.6667    1.0000
   14.6667    3.0000    0.6667
    1.6667    1.3333    1.0000
```

```
mat.^2
```

```
ans = 3×3
           1           4           9
        1936          81           4
          25          16           9
```

```
vec=[5 9 3 4 6 11]
```

```
vec = 1×6
     5     9     3     4     6    11
```

```
v = [0 1 0 0 1 1]
```

```
v = 1×6
     0     1     0     0     1     1
```

```
v = logical(v)
```

```
v = 1×6 logical array
   0   1   0   0   1   1
```

```
vec(v)
```

```
ans = 1×3
     9     6    11
```

```
find(vec>9)
```

```
ans = 6
```

```
find(vec<9)
```

```
ans = 1×4
     1     3     4     5
```

```
vec(vec<0) = []
```

```
vec = 1×6
     5     9     3     4     6    11
```

```
neg = find(vec<0)
```

```
neg =

  1×0 empty double row vector
```

```
vec(neg) = []
```

```
vec = 1×6
     5     9     3     4     6    11
```

```
A = [1 4;3 3]
```

```
A = 2×2
     1     4
     3     3
```

```
B = [1 2;-1 0]
```

```
B = 2×2
     1     2
    -1     0
```

```
A.*B
```

```
ans = 2×2
     1     8
    -3     0
```

```
A*B
```

```
ans = 2×2
    -3     2
     0     6
```

```
B*A
```

```
ans = 2×2
     7    10
    -1    -4
```

Lecture 7

```
mat = randi(100,3,4)
```

```
mat = 3×4
    89    17    51    69
     3    98    48     5
    49    72     6     8
```

```
rot90(mat)
```

```
ans = 4×3
    69     5     8
    51    48     6
    17    98    72
    89     3    49
```

```
rot90(mat,2)
```

```
ans = 3×4
     8     6    72    49
     5    48    98     3
    69    51    17    89
```

```
rot90(mat,-1)
```

```
ans = 4×3
    49     3    89
    72    98    17
     6    48    51
     8     5    69
```

```
fliplr(mat)
```

```
ans = 3×4
    69    51    17    89
     5    48    98     3
     8     6    72    49
```

```
flipud(mat)
```

```
ans = 3×4
    49    72     6     8
     3    98    48     5
    89    17    51    69
```

```
vec = []
```

```
vec =

     []
```

```
length(vec)
```

```
ans = 0
```

```
vec = 1:10
```

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

```
vec(4) = []
```

```
vec = 1×9
     1     2     3     5     6     7     8     9    10
```

```
vec = 3:15
```

```
vec = 1×13
```

```
     3     4     5     6     7     8     9    10    11    12    13    14    15
```

```
vec(9:12) = []
```

```
vec = 1×9
     3     4     5     6     7     8     9    10    15
```

```
mat = randi(100,3,4)
```

```
mat = 3×4
    53    82    66    65
    10    73    52    81
    82    15    98    46
```

```
mat(:,2) = []
```

```
mat = 3×3
    53    66    65
    10    52    81
    82    98    46
```

```
vec = -5:1
```

```
vec = 1×7
    -5    -4    -3    -2    -1     0     1
```

```
abs(vec)
```

```
ans = 1×7
     5     4     3     2     1     0     1
```

```
mat = [-4 2 8;0 -10 -42;-9 15 0]
```

```
mat = 3×3
    -4     2     8
     0   -10   -42
    -9    15     0
```

```
sign(mat)
```

```
ans = 3×3
    -1     1     1
     0    -1    -1
    -1     1     0
```

```
str1 = "hello"
```

```
str1 = 
"hello"
```

```
str2 = "howdy"
```

```
str2 = 
"howdy"
```

```
str1 == str2
```

```
ans = logical
   0
```

Lecture 8

```
A = [3 2 -1;2 -2 1;-1 -0.5 -1]
```

```
A = 3×3
    3.0000    2.0000   -1.0000
    2.0000   -2.0000    1.0000
   -1.0000   -0.5000   -1.0000
```

```
B = [1;-2;0]
```

```
B = 3×1
     1
    -2
     0
```

```
X = A\B
```

```
X = 3×1
   -0.2000
    0.7200
   -0.1600
```

```
C = A*X
```

```
C = 3×1
    1.0000
   -2.0000
    0.0000
```

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

```
A = 3×3
     1     2     3
     3     3     4
     2     3     3
```

```
B = [1;1;2]
```

```
B = 3×1
     1
     1
     2
```

```
C = B\A
```

```
C = 1×3
    1.3333    1.8333    2.1667
```

```
linsolve(A,B)
```

```
ans = 3×1
   -0.5000
    1.5000
   -0.5000
```

```
transpose(C)
```

```
ans = 3×1
    1.3333
    1.8333
    2.1667
```

```
A = [3 -3 4;2 -3 4;0 -1 1]
```

```
A = 3×3
     3    -3     4
     2    -3     4
     0    -1     1
```

```
[v,d] = eig(A)
```

```
v = 3×3 complex
   0.8944 + 0.0000i   0.3536 - 0.3536i   0.3536 + 0.3536i
  -0.0000 + 0.0000i   0.7071 + 0.0000i   0.7071 + 0.0000i
  -0.4472 + 0.0000i   0.3536 + 0.3536i   0.3536 - 0.3536i
d = 3×3 complex
   1.0000 + 0.0000i   0.0000 + 0.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i  -0.0000 + 1.0000i   0.0000 + 0.0000i
   0.0000 + 0.0000i   0.0000 + 0.0000i  -0.0000 - 1.0000i
```

```
A = A.^2
```

```
A = 3×3
     9     9    16
     4     9    16
     0     1     1
```

```
[v,d] = eig(A)
```

```
v = 3×3
   -0.8242    0.9351   -0.0899
   -0.5651   -0.3322   -0.8440
   -0.0379   -0.1235    0.5287
d = 3×3
   15.9070         0         0
         0    3.6894         0
         0         0   -0.5964
```

```
round(rand*[20,35])
```

```
ans = 1×2
     3     4
```

```
x = 4
```

```
x = 4
```

```
3<x<5
```

```
ans = logical
   1
```

```
3 == 5+2
```

```
ans = logical
   0
```

```
'b' < 'a' + 1
```

```
ans = logical
   0
```

```
10 > 5 + 2
```

```
ans = logical
   1
```

```
(10>5) + 2
```

```
ans = 3
```

```
'c' == 'd' -1 && 2 < 4
```

```
ans = logical
   1
```

```
'c' == 'd' -1 | 2>4
```

```
ans = logical
   1
```

```
xor('c' == 'd' - 1, 2>4)
```

```
ans = logical
   1
```

```
xor('c' == 'd' - 1, 2<4)
```

```
ans = logical
   0
```

```
10>5> 2
```

```
ans = logical
   0
```

```
'b' >= 'c'- 1
```

```
ans = logical
   1
```

```
rand*(50-20)+20
```

```
ans = 47.4013
```

```
mat = [1 :3; 44 9 2; 5:-1:3]
```

```
mat = 3×3
     1     2     3
    44     9     2
     5     4     3
```

```
mat (3 , 2)
```

```
ans = 4
```

```
mat (2, : )
```

```
ans = 1×3
    44     9     2
```

```
size (mat)
```

```
ans = 1×2
     3     3
```

```
mat ( : , 4) = [8 ; 11;33]
```

```
mat = 3×4
     1     2     3     8
    44     9     2    11
     5     4     3    33
```

```
numel (mat )
```

```
ans = 12
```

```
v = mat (3, : )
```

```
v = 1×4
     5     4     3    33
```

```
v (v (2) )
```

```
ans = 33
```

```
v (1) = []
```

```
v = 1×3
     4     3    33
```

```
reshape (mat , 2 , 6)
```

```
ans = 2×6
     1     5     9     3     3    11
    44     2     4     2     8    33
```

```
ones(2)*10
```

```
ans = 2×2
    10    10
    10    10
```

```
A = [1,4;3,3]
```

```
A = 2×2
     1     4
     3     3
```

```
B = [1 2;-1 0]
```

```
B = 2×2
     1     2
    -1     0
```

```
A.*B
```

```
ans = 2×2
     1     8
    -3     0
```

```
A*B
```

```
ans = 2×2
    -3     2
     0     6
```

```
B*A
```

```
ans = 2×2
     7    10
    -1    -4
```

```
vec = 3:15
```

```
vec = 1×13
     3     4     5     6     7     8     9    10    11    12    13    14    15
```

```
vec(8:12) = []
```

```
vec = 1×8
     3     4     5     6     7     8     9    15
```

```
mat = [1:3;44 9 2; 5:-1:3]
```

```
mat = 3×3
     1     2     3
    44     9     2
     5     4     3
```

```
mat(:,2) = []
```

```
mat = 3×2
     1     3
    44     2
     5     3
```

```
vec = -5:1
```

```
vec = 1×7
    -5    -4    -3    -2    -1     0     1
```

```
abs(vec)
```

```
ans = 1×7
     5     4     3     2     1     0     1
```

```
mat = [-4 2 8;0 -10 -42;-9 15 0]
```

```
mat = 3×3
    -4     2     8
     0   -10   -42
    -9    15     0
```

```
sign(mat)
```

```
ans = 3×3
```

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

```
str1 = "hello"
```

```
str1 = 
"hello"
```

```
str2 = "howdy"
```

```
str2 = 
"howdy"
```

```
str1 == str2
```

```
ans = logical
   0
```

Basic Plotting

1. Overview
2. Axis labels, and annotations
3. Adding Titles
4. Specifying line styles and colours
5. creating simple plots
6. multiple data sets in one plot

Overview

plot(***xvalues,yvalues,'style-options'***)

examples

```
plot(x,y)          plots y vs x with a solid line

plot(x,y,'--')     plots y vs x with a dashed line

plot(x)            plots the elements of x afainst the rows
```

Style Options

Color Style options

```
y = yellow

m = magneta

c = cyan

r = red

g = green

b = blue

w = white

k = black
```

Line Style options

```
-  = solid

-- = dashed

: = dotted

-. = dashed-dot

none = no line
```

Marker Style options

```
+ = plus sign

o = circle

* = asterick

x = x-mark

. = point

^ = up triangle

s = square

d = diamond
```

Lable and Title

xlabel('Pipe Length') labels x-axis with Pipe Length

ylabel('Fluid Pressure') labels y-axis with Fluid Pressure

title('Pressure Variation') titles the plot with Pressure Variation

legend(string1,string2,....) produces legend using the text in string1 string2 etc as labels

legend(LineStyle1,sting1,....) specifies the line style of each label writes the legend outside the plot frame

legend(....,pos) if pos = -1 and inside if pos = 0,

there are other options for pos too, and

legned off deletes the legend from the plot

Axis Control

axis([xmin xmax ymin ymax])

Examples

```
axis([-5 10 2 22]);                       %set the x-axis from 05 to 10 and y axis from 2 to
axy = [-5 10 2 22]; axis(axy)             %same as above,and
ax = [-5 10]; ay = [2 22]; axis([ax ay]); %same as above
```

axis (' equal') **sets equal scale on both axes,**

axis (' square') **sets the default rectangular frame to a square,**

axis ('normal') **resets the axis to default values,**

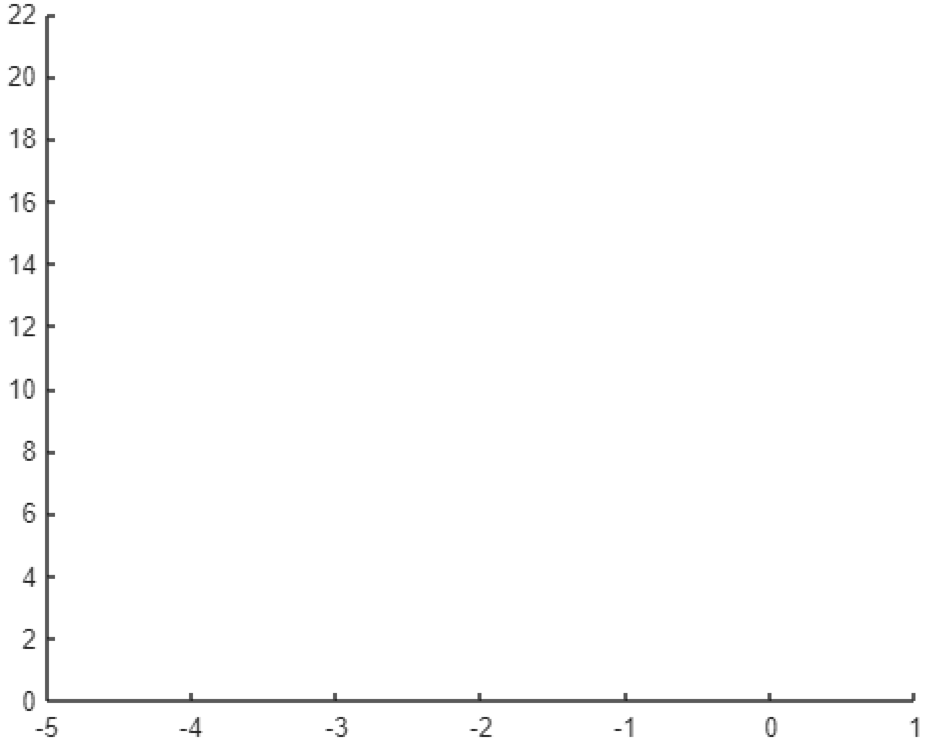
axis ('axis') **freezes the current axes limits, and**

axis (' off') **removes the surrounding frame and the tick marks.**

Semi Axis Control

```
axis ( [-5 10 -inf inf] )      %sets the x-axis limits at -5 and 10 and lets
```

```
                                %the y-axis limits be set automatically, and
axis ( [-5 inf -inf 22])        %sets the lower limit of the x-axis and the
```



```
                                % upper limit of the y-axis, and leaves the
                                % other two limits to be set automatically.
```

Overlay Plot

Method - 1 Using the plot command to generate overlay plots

plot(x1,y1,x2,y2,':',x3,y3,'o')

Method - 2 Using the hold command to generate overlay plots

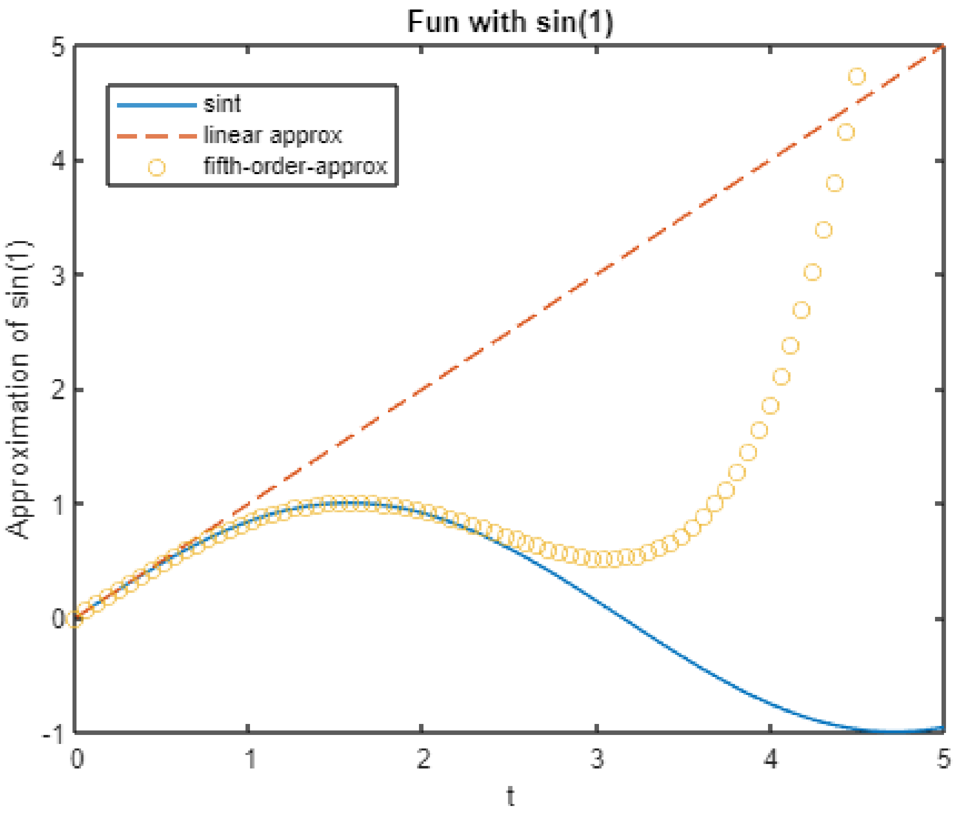
```
x = linspace(0,2*pi,100)
```

```
x = 1×100
         0    0.0635    0.1269    0.1904    0.2539    0.3173    0.3808    0.4443 ···
```

```
y1 = sin(x)
```

```
y1 = 1×100
         0    0.0634    0.1266    0.1893    0.2511    0.3120    0.3717    0.4298 ···
```

```
plot(x,y1)
hold on
y2 = x;
plot(x,y2,'--')
y3 = x - (x.^3)/6 + (x.^5)/120;
plot(x,y3,'o')
axis([0 5 -1 5])
xlabel('t')
ylabel('Approximation of sin(1)')
title('Fun with sin(1)')
legend('sint','linear approx','fifth-order-approx')
hold off

legend("Position",[0.16091,0.76058,0.25714,0.11905])
```



Method - 3 Using the line command to generate overlay plots

Specailized 2D plot

```
area - creates a filled area plot
bar - creates a bar graph
barh - creates horizontal bar graph
comet - makes animated 2D plot
compass - creates arrow graph for complex numbers
contour - makes contour plots
contourf - makes filled contour plots
errorbar - plots a graph and puts error bars
feather - makes feather plot
fill - draws a filled ploygon
fplot - plots a function of a single variable
hist - makes histograms,
loglog - creates plot with log scale on both the r-axis and the y-axis,
pareto - makes pareto plots,
pcolor - makes pseudocolor plot of a matrix,
pie - creates a pie chart,
plotyy - makes a double y-axis plot,
plotmatrix - makes a scatter plot of a matrix,
polar - plots curves in polar coordinates,
quiver - plots vector fields,
rose - makes angled histograms,
scatter - creates a scatter plot,
semilogx - makes semilog plot with log scale on the x-axis,
semilogy - makes semilog plot with log scale on the y-axis,
stairs - plots a stair graph, and
stem - plots a stem graph.
```

Let's say that you want to plot these two equations in the same window:

$y_1 = \cos(x)$

$y_2 = x^2 - 1$

Steps for 2D Plots

1. Define your interval of interest, think of highest and lowest values, and a step.
2. Define your function y = f(x). Take into account that you're working with arrays, not with scalars, use dot operators.
3. Use appropriate 2D built-in functions

1. Define your Interval

Think:

- What values for x do I want to take intoaccount?
- What steps in the array should I consider?

2. Define your Function(s)

Think of lower and upper values, and steps

```
x = -1 : 0.1 : 1.5;
y1 = cos(x);
y2 = x.^2 - 1;
```

Now, x, y1 and y2 are vectors with appropriate values.

3. Use 2D built-in Functions

You can use functions such as:

plot

stem

polar, compass, rose

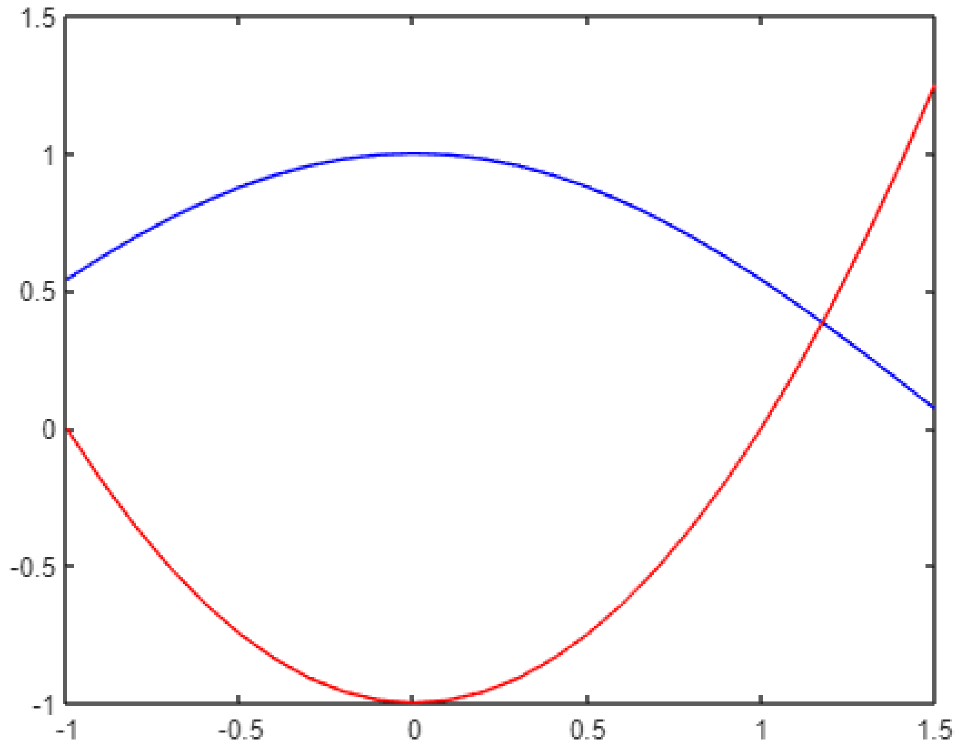
loglog, semilogx, semilogy

area,fill

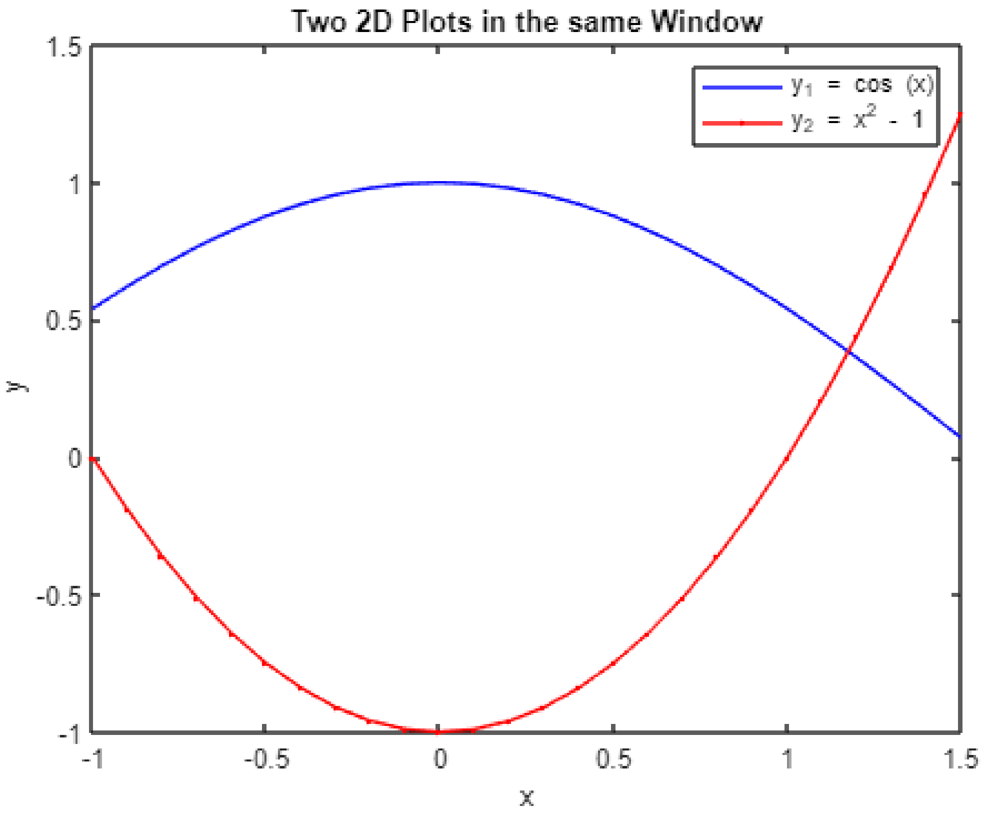
pie

hist, stairs

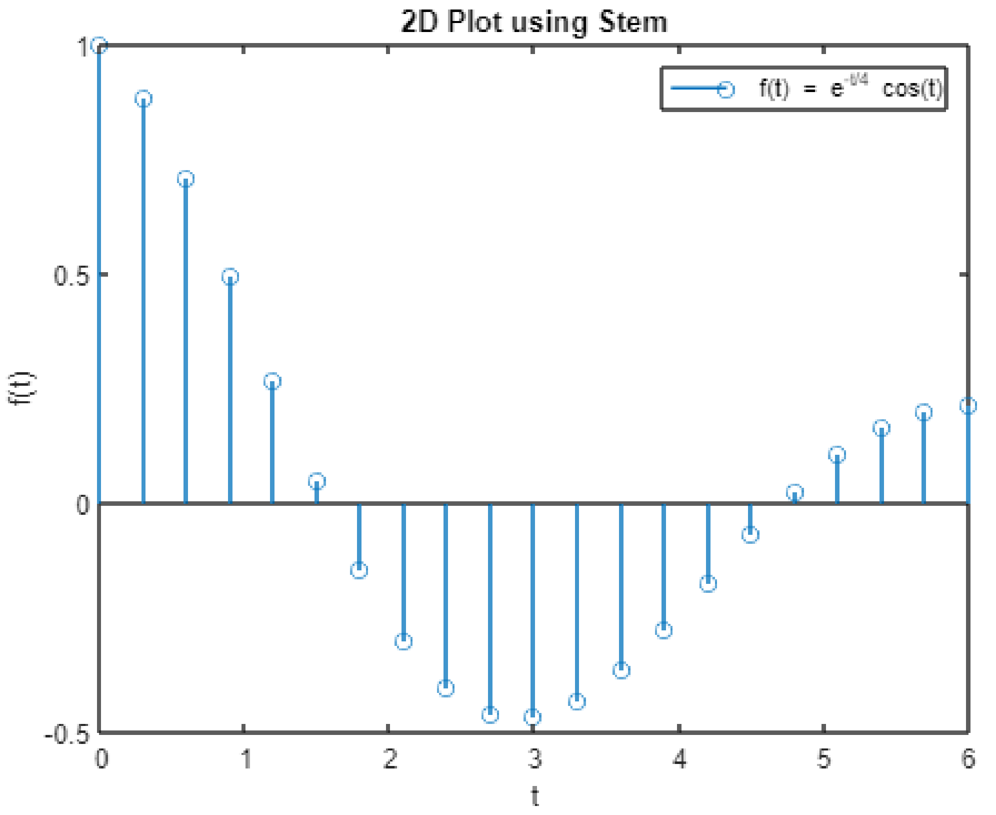
```
plot(x,y1,'b',x,y2,'r')
```



```
plot(x, y1, 'b' , x, y2, 'r.-' )
title('Two 2D Plots in the same Window' )
legend('y_1 = cos (x)','y_2 = x^2 - 1')
xlabel('x')
ylabel('y')
```



```
t = 0 : .3 : 2*pi;
f = exp(-t/4) .* cos(t);
stem(t,f)
title('2D Plot using Stem' )
legend( 'f(t) = e^{-t/4} cos(t) ')
xlabel('t' )
ylabel('f(t)')
```



```
rows = randi(1,5)
```

```
rows = 5×5
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
col = randi(1,5)
```

```
col = 5×5
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
     1     1     1     1     1
```

```
x = length(rows)
```

```
x = 5
```

```
y = length(col)
```

```
y = 5
```

```
mat = zeros(x,y)
```

```
mat = 5×5
     0     0     0     0     0
```

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

fplot

- `fplot(@fun,lims)` - plots the function fun between the x-axis limits
- `lims = [xmin xmax ymin ymax]` - axis limits
- The function `fun(x)` must return a row vector for each element of vector x.

AXIS Control

- axis scaling and appearance
- `axis([xmin xmax ymin ymax])`
- Sets scaling for the x- and y-axes on the current plot
- **`axis auto`** - returns the axis scaling to its default, automatic mode.
- **`axis off`** - turns off all axis labelling,tick marks and background.
- **`axis on`** - turms axis labeling, tick marks and background.
- **`axis equal`** - makes both axes equal.

3D Plot

the general syntax for plot3 command is

plot3(x,y,z,'*style-option*')

`plot3` - plots curves in space,

`stem3` - creates discrete data plot with stems in 3-D,

`bar3` - plots 3-D bar graph,

`bar3h` - plots 3-D horizontal bar graph,

`pie3` - makes 3-D pie chart,

`comet3` - makes animated 3-D line plot,

`fill3` - draws filled 3-D polygons,

`contour3` - makes 3-D contour plots,

`quivers` - draws vector fields in 3-D,

`scatter3` - makes scatter plots in 3-D,

`mesh` - draws 3-D mesh surfaces (wire-frame),

`meshc` - draws 3-D mesh surfaces along with contours,

`meshz` - draws 3-D mesh surfaces with reference plane curtains,

`surf` - creates 3-D surface plots,

`surfc` - creates 3-D surface plots along with contours, `surfl` - creates 3-D surface plots with specified light source,

`trimesh` - mesh plot with triangles,

`trisurf` - surface plot with triangles,

`slice` - draws a volumetric surface with slices,

`waterfall` - creates a waterfall plot of 3-D data,

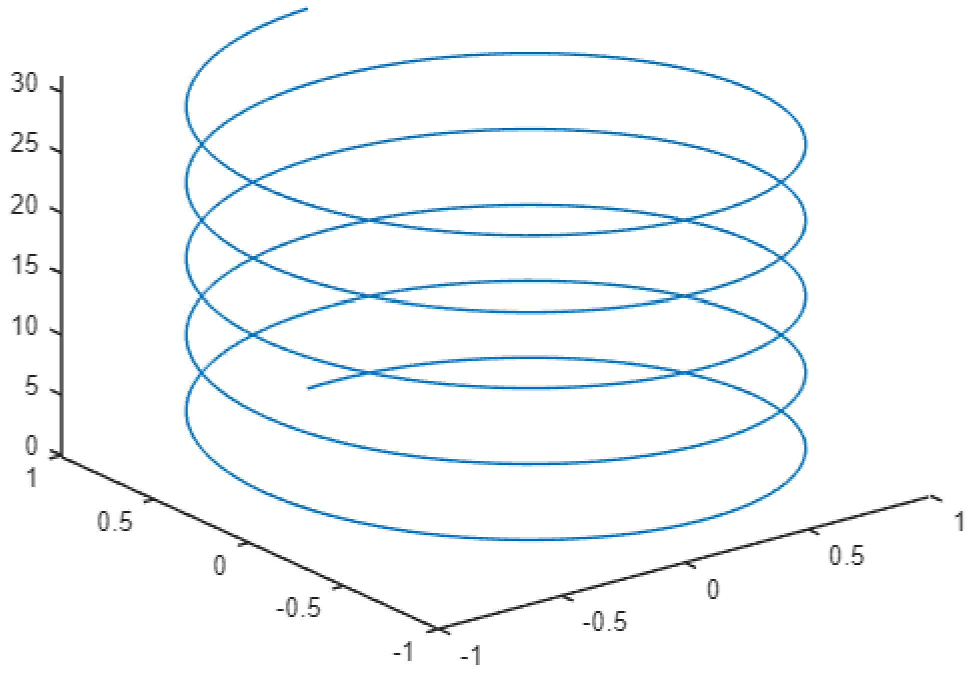
`cylinder` - generates a cylinder,

`ellipsoid` - generates an ellipsoid, and

`sphere` - generates a sphere.

3D Plot: Question 1

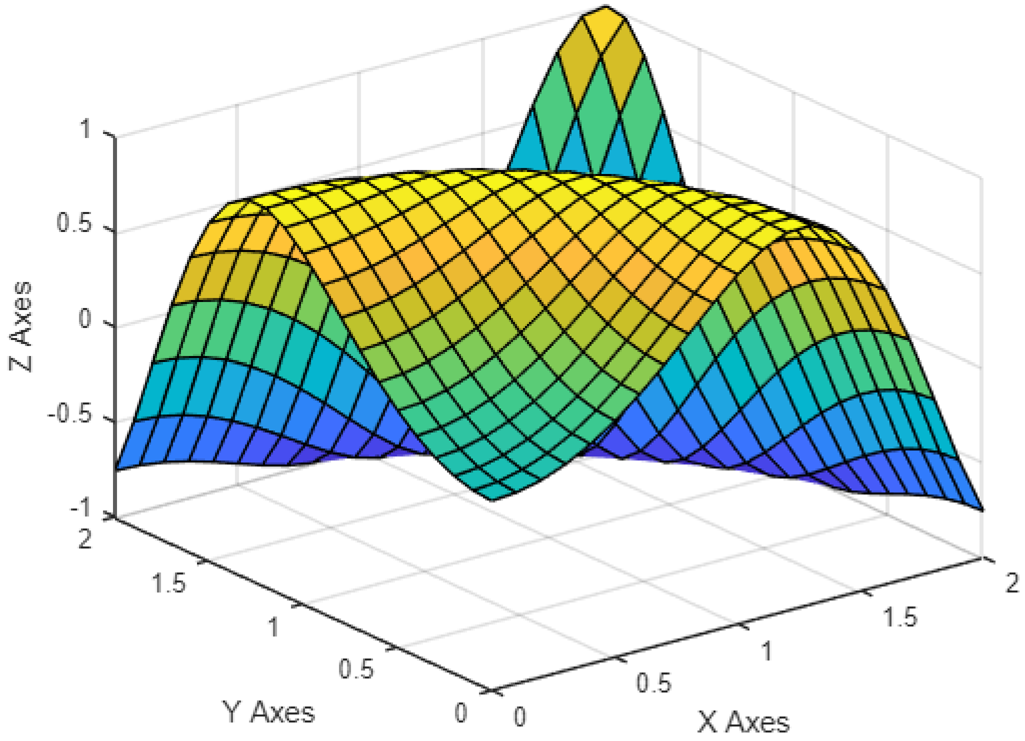
```
t = 0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
```



Surface Plot: Question 2

```
x = 0:0.1:2;
y = 0:0.1:2;
[xx,yy] = meshgrid(x,y);
zz = sin(xx.^2+yy.^2);
surf(xx,yy,zz)
```

```
xlabel('X Axes')
ylabel('Y Axes')
zlabel('Z Axes')
```

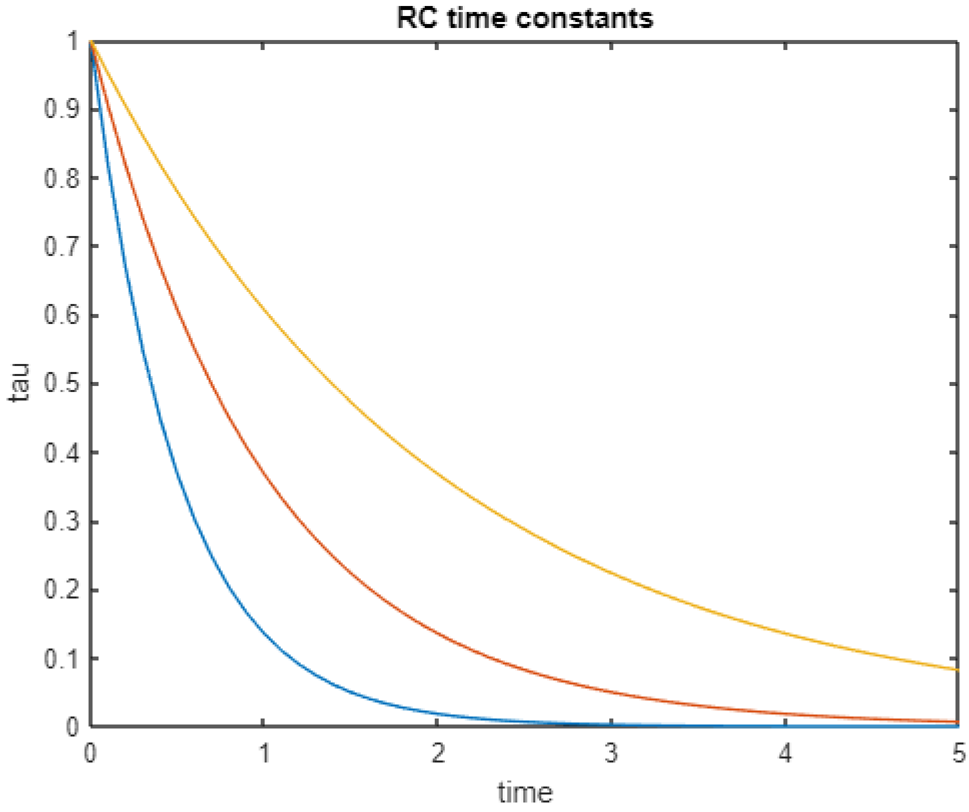


Question 3

Plot voltage vs time for various RC time constants

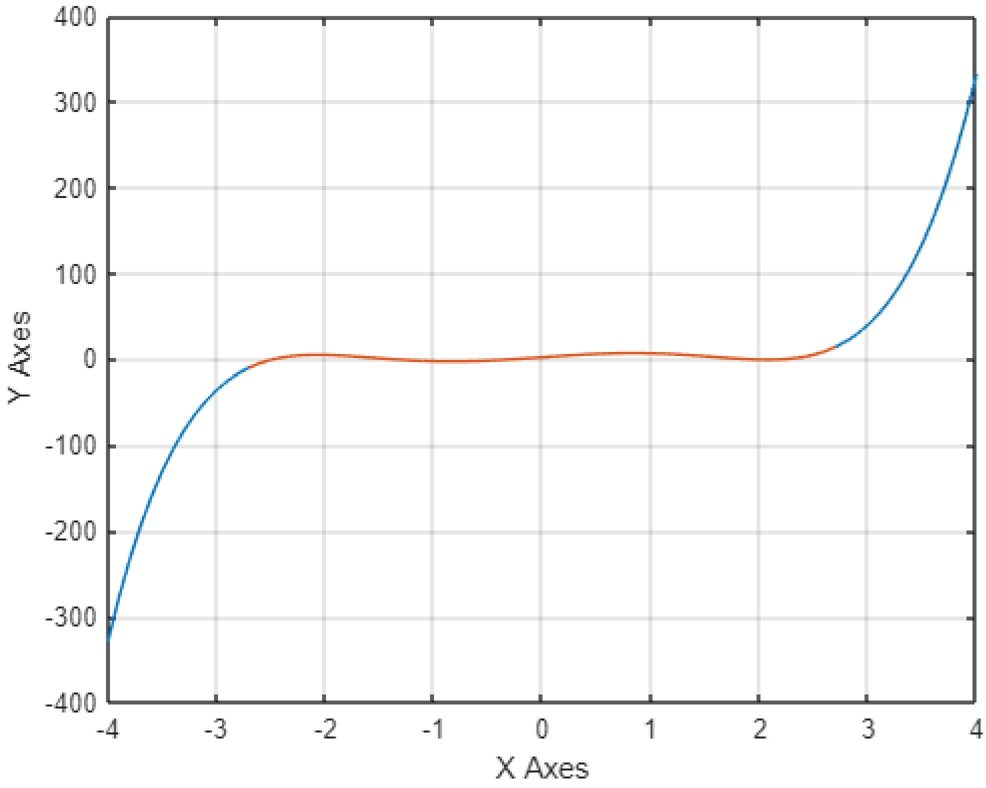
$$\frac{v}{V} = e^{-\frac{t}{\tau}}$$

```
time = 0:0.1:5;
tau = [0.5 1.0 2.0];
[TIME TAU] = meshgrid(time,tau);
V = exp(-TIME./TAU);
plot(time,V)
xlabel('time')
ylabel('tau')
title('RC time constants')
```



Question 4

```
x1 = -4:0.1:4;
x2 = -2.7:0.1:2.7;
f1 = 0.6*x1.^5 - 5*x1.^3+9*x1+2;
f2 = 0.6*x2.^5 - 5*x2.^3+9*x2+2;
plot(x1,f1,x2,f2)
grid on
xlabel('X Axes')
ylabel('Y Axes')
```

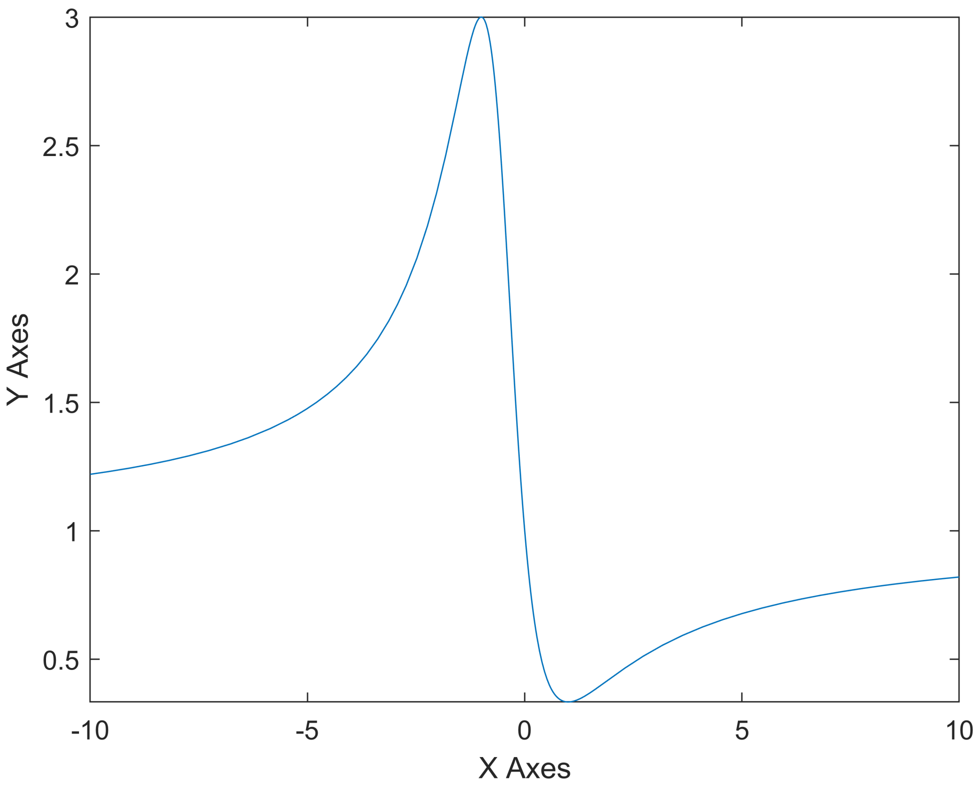


Question 5

```
f = @(x)(x^2 - x + 1)/(x^2 + x + 1);
l = [-10,10];
fplot(f,l)
```

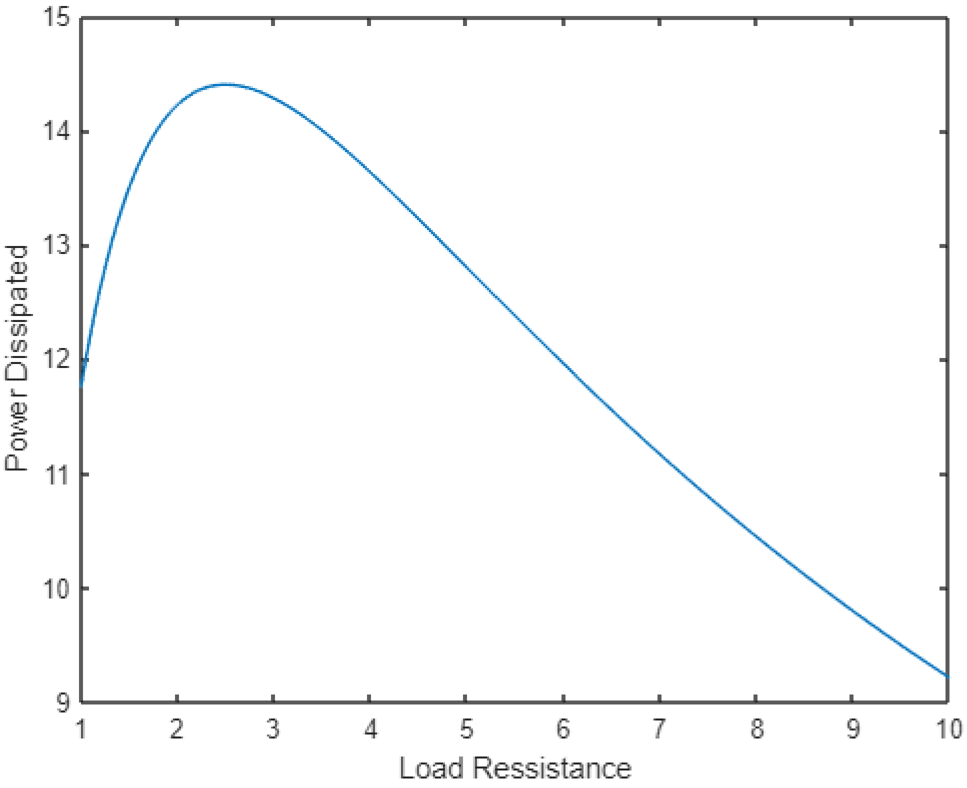
```
Warning: Function behaves unexpectedly on array inputs. To improve performance, properly vectorize your
function to return an output with the same size and shape as the input arguments.
```

```
xlabel('X Axes')
ylabel('Y Axes')
```



Question 6

```
RL = 1:0.01:10;
Vs = 12;
Rs = 2.5;
P = (Vs^2*RL)./(RL+Rs).^2;
plot(RL,P)
xlabel('Load Ressistance')
ylabel('Power Dissipated')
```


15
14
13
12
11
10
9
1
2
3
4
5
6
7
8
9
10
Power Dissipated
Load Ressistance

1) Using the top-down design approach, write an algorithmfor making a sandwich.

- Get the ingredients
- Get the utensils
- Assemble the sandwich

Get the ingredients:

- Get the bread
- Get the cheese
- Get the condiments

Get the bread:

- Open the bread box
- Select desired bread
- Open bag and remove2 slices

etc.

2) Write a simple scriptthat will calculatethe volume of a hollowsphere,

$$\frac{4\pi}{3}\left(r_o^3 - r_1^3\right)$$

where ri is the inner radius and ro is the outer radius. Assign a value to a variable for the inner radius, and also assign a value to another variable for the outer radius. Then, using these variables, assign the volume to a third variable. Include comments in the script. Use **help** toview the commentin your script.

```
% This script calculates the volume of a hollow sphere

% Assign values for the inner and outer radii
ri = 5.1;
ro = 6.8;
% Calculate the volume
vol = (4*pi)/3*(ro^3-ri^3)
```

```
vol = 761.4425
```

3) Write a statement that prompts the user for his/her favorite number.

```
favnum = input('What is your favorite number: ')
```

```
favnum = 810
```

4) Write a statement that prompts the user for his/her name.

```
uname = input('What is your name: ', 's')
```

```
uname = 
'Varun'
```

5) Write an input statement that will prompt the user for a real number,and store it in a variable. Then, use the fprintf function to print the value of this variable using 2 decimalplaces.

```
realnum = input('Enter a real number: ');
fprintf('The number is %.2f\n', realnum)
```

```
The number is 810.00
```

6) Experiment, in the Command Window, with using the fprintf function for real numbers. Make a note of what happens for each. Use fprintf to print the real number 12345.6789.

```
realnum = 12345.6789;
```

- without specifying any field width

```
fprintf('The number is %f\n', realnum)
```

```
The number is 12345.678900
```

- in a field width of10 with 4 decimalplaces

```
fprintf('The number is %10.4f\n', realnum)
```

```
The number is 12345.6789
```

- in a field width of10 with 2 decimalplaces

```
fprintf('The numberis %10.2f\n', realnum)
```

```
The numberis   12345.68
```

- in a field width of 6 with 4 decimal places

```
fprintf('The number is %6.4f\n', realnum)
```

```
The number is 12345.6789
```

- in a field width of 2 with 4 decimal places

```
fprintf('The number is %2.4f\n', realnum)
```

```
The number is 12345.6789
```

7) Experiment, in the CommandWindow, with using the fprintf function for integers. Make a note of what happens for each. Use fprintf to print the integer 12345.

```
intnum = 12345;
```

- without specifying any field width

```
fprintf('The number is %d\n', intnum)
```

```
The number is 12345
```

- in a field width of 5

```
fprintf('The number is %5d\n', intnum)
```

```
The number is 12345
```

- in a field width of 8

```
fprintf('The number is %8d\n', intnum)
```

```
The number is    12345
```

- in a field width of 3

```
fprintf('The number is %3d\n', intnum)
```

```
The number is 12345
```

8) When would you use disp instead of fprintf? When would you use fprintf instead of disp? The disp function is used when no formatting is required. It is also easier to print vectors and matrices using disp. The fprintf function is used for formatted output.

9) Write a script called*echostring* that will promptthe user for a string,and will echo print the string in quotes:

```
>> echostring

Enter your string: hi there

Your string was: 'hi there'
```

```
% Prompt the user and print a string in quotes
str = input('Enter your string: ', 's');
fprintf('Your string was: ''%s''\n',str)
```

```
Your string was: 'Varun Khadayate'
```

10) If the lengths of two sides of a triangle and the angle between them are known, the length of the third side can be calculated. Given the lengths of two sides (b and c) of a triangle, and the angle between them α in degrees, the third side a is calculated as follows:

$$a^2 = b^2 + c^2 - 2\text{bccos}(\alpha)$$

Write a script *thirdside* thatwill prompt the user and read in values for b, c, and α (in degrees), and then calculate and print the value of a with 3 decimal places. The format of the output from the script should look exactlylike this:

```
>> thirdside

Enter the first side: 2.2

Enter the second side: 4.4

Enter the angle between them: 50

The third side is 3.429
```

```
% Calculates the third side of a triangle, given
% the lengths of two sides and the angle between them
b = input('Enter the first side: ');
c = input('Enter the second side: ');
alpha = input('Enter the angle between them: ');
a = sqrt(b^2 + c^2 -2*b*c*cosd(alpha));
fprintf('\nThe third side is %.3f\n', a)
```

```
The third side is 3.429
```

11) Write a script that will prompt the user for a character, and will print it twice; once left-justified in a field width of 5, and again right-justified in a field width of 3.

```
mych = input('Enter a character: ', 's');
fprintf('Here it is: %-5c and again: %3c\n',mych,mych)
```

```
Here it is: v     and again:   a
Here it is: r     and again:   u
Here it is: n     and again:
Here it is: k     and again:   h
Here it is: a     and again:   d
Here it is: a     and again:   y
Here it is: a     and again:   t
Here it is: e     and again:   v
Here it is: a     and again:   r
Here it is: u     and again:   n
Here it is:       and again:   k
Here it is: h     and again:   a
Here it is: d     and again:   a
Here it is: y     and again:   a
Here it is: t     and again:   e
```

12) Write a script *lumin* that will calculate and print the luminosity L of a star in Watts. The luminosity L is given by $L = 4\pi d^2 b$ where d is the distance from the sun in meters and b is the brightness in Watts/meters2. Here is an example of executing the script:

```
>> lumin

This scriptwill calculate the luminosity of a star.

When prompted, enter the star'sdistance from the sun in meters, and its brightness in
W/meters squared.

Enter the distance: 1.26e12 Enter the brightness: 2e-17

The luminosity of this star is 399007399.75 watts
```

```
% Calculates the luminosity of a star
disp('This script will calculate the luminosity of a star.')
```

```
This script will calculate the luminosity of a star.
```

```
disp('When prompted, enter the star''s distance from the sun')
```

```
When prompted, enter the star's distance from the sun
```

```
fprintf(' in meters, and its brightness in W/meters squared.\n\n')
```

```
 in meters, and its brightness in W/meters squared.
```

```
d = input('Enter the distance: ');
b = input('Enter the brightness: ');
L = 4*pi*d^2*b;
fprintf('The luminosity of this star is %.2f watts\n', L)
```

```
The luminosity of this star is 399007399.75 watts
```

13) A script *iotrace* has been written. Here's what the desired output looks like:

```
>>iotrace

Please enter a number: 33 Please enter a character: x Your numberis 33.00

Your char is      x!
```

Fix this script so that it works as shown above:

```
mynum = input('Please enter a number:\n ');
mychar = input('Please enter a character: ', 's');
fprintf('Your number is %.2f\n', mynum)
```

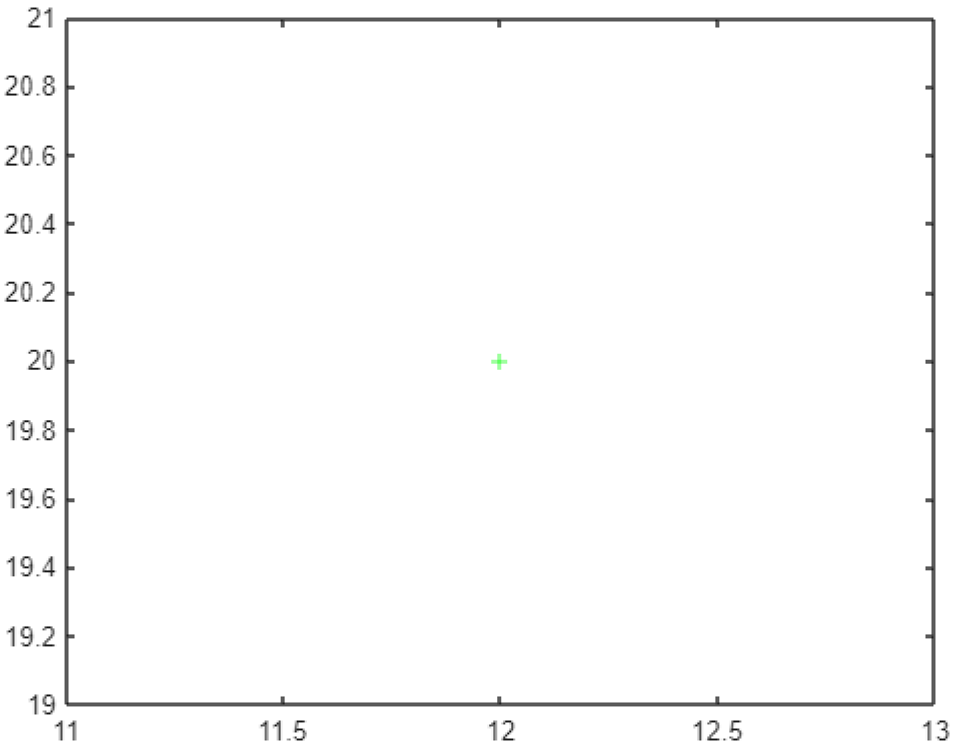
```
Your number is 12.00
```

```
fprintf('Your char is %6c!\n', mychar)
```

```
Your char is      V!
Your char is      a!
Your char is      r!
Your char is      u!
Your char is      n!
```

14) Write a script that assigns values for the x coordinateand then y coordinate of a point, and then plots this using a green +.

```
% Prompt the user for the coordinates of a point and plot
% the point using a green +

x = input('Enter the x coordinate: ');
y = input('Enter the y coordinate: ');
plot(x,y, 'g+')
```

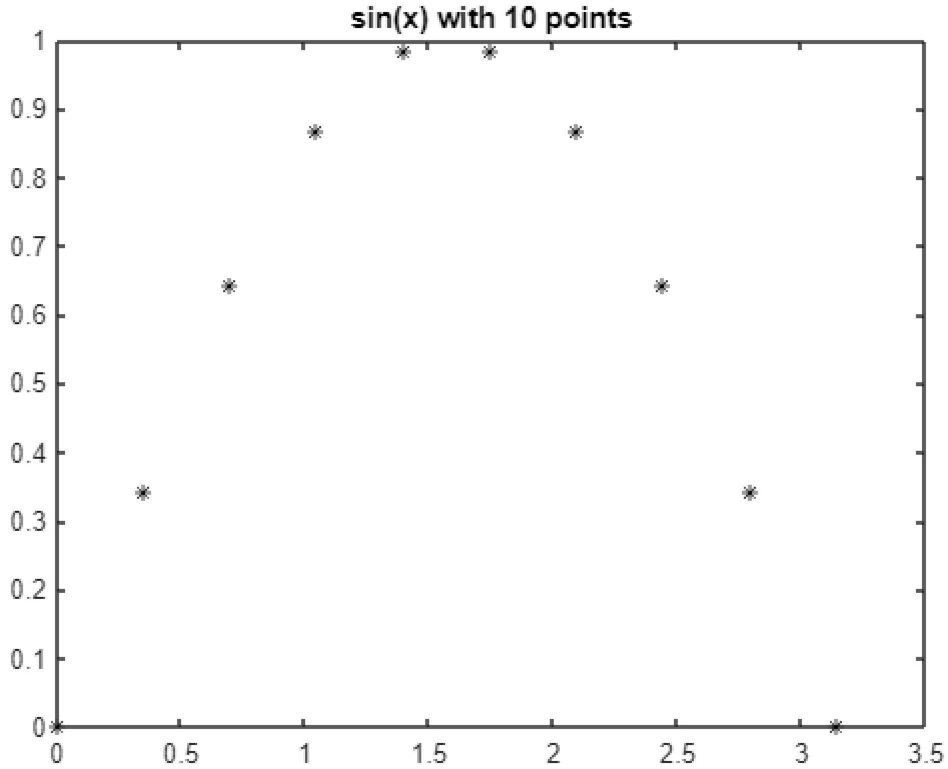


15) Plot sin(x) for x values ranging from 0 to π (in separate Figure Windows):

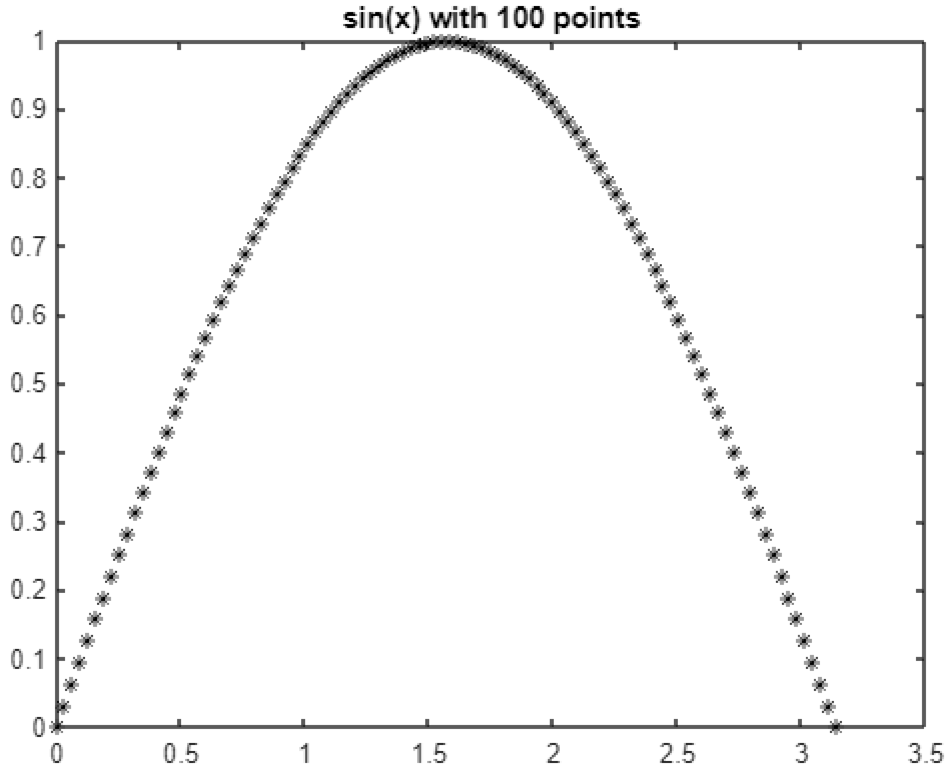
- using 10 points in this range
- using 100 points in this range

```
% Plots sin(x) with 10 points and 100 points in range 0 to pi
x = linspace(0,pi,10);
y = sin(x);
```

```
clf
figure(1)
plot(x,y,'k*')
title('sin(x) with 10 points')
```



```
figure(2)
x = linspace(0,pi); y = sin(x);
plot(x,y,'k*')
title('sin(x) with 100 points')
```



16) When would it be important to use legend in a plot?

```
When you have more than one plot in a single Figure Window.
```

17) Why do we always suppress all assignment statements in scripts?

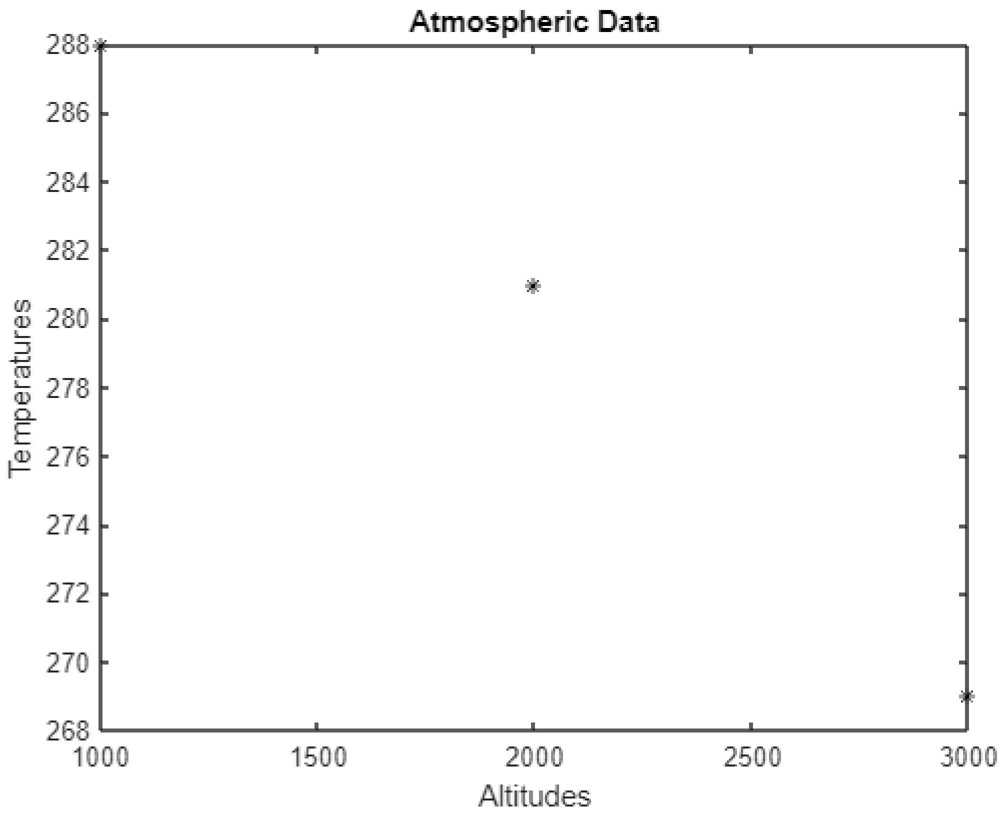
- So we don’t just see the variable = and then the value.

18) Atmospheric properties such as temperature, air density, and air pressure are important in aviation. Create a file that stores temperatures in degrees Kelvin at variousaltitudes. The altitudes are in the first column and the temperatures in the second. For example, it may look like this:

```
1000    288

2000    281

3000    269
```

```
% Read altitudes and temperatures from a file and plot
row1 = [1000 288];
row2 = [2000 281];
row3 = [3000 269];
mat = [row1;row2;row3];
% Save the matrix as an ASCII file.
```

```
save alttemps.dat mat -ascii
load alttemps.dat
altitudes = alttemps(:,1);
temps = alttemps(:,2);
plot(altitudes,temps,'k*')
xlabel('Altitudes')
ylabel('Temperatures')
title('Atmospheric Data')
```

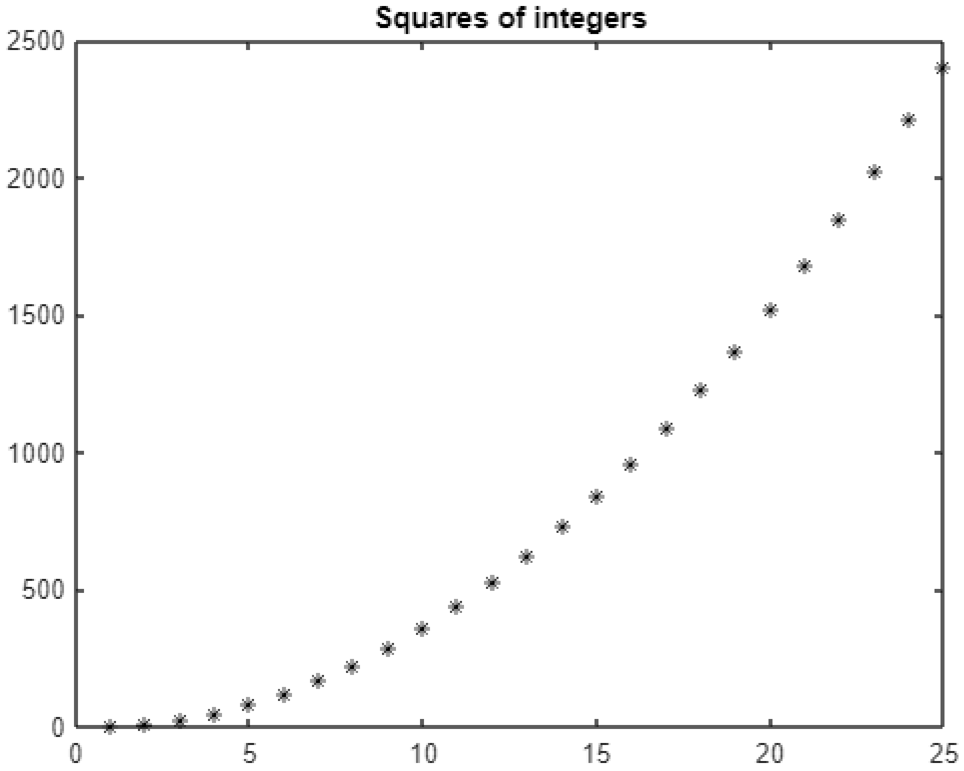


19) Generate a random integer*n*, create a vector of the integers 1 through *n* in steps of 2, square them, andplot the squares.

```
% Create a vector of integers 1:2:n where n is random
% square them and plot the squares
n = randi([1,50])
```

```
n = 49
```

```
vec = 1:2:n;
vecsq = vec .^ 2;
plot(vecsq,'k*')
title('Squares of integers')
```



20) Create a *3 x 6* matrix of random integers,each in the range of 50 - 100. Write this to a file called*randfile.dat*. Then, create a new matrixof random integers,but this time make it a *2 x 6* matrix of random integers,each in the range of 50 - 100. Append this matrix to the original file. Then, read the file in (which will be to a variable called *randfile*) just to make sure that worked!

```
mat = randi([50,100], 3,6)
```

```
mat = 3×6
    92    62    60    74    79    64
    62    97    62    67    78    88
    91    67    81    92    96    88
```

```
save randfile.dat mat -ascii
newmat = randi([50,100], 2,6)
```

```
newmat = 2×6
    69    53    77    97    79    50
    78    52    89    56    73    67
```

```
save randfile.dat newmat -ascii -append
load randfile.dat
```

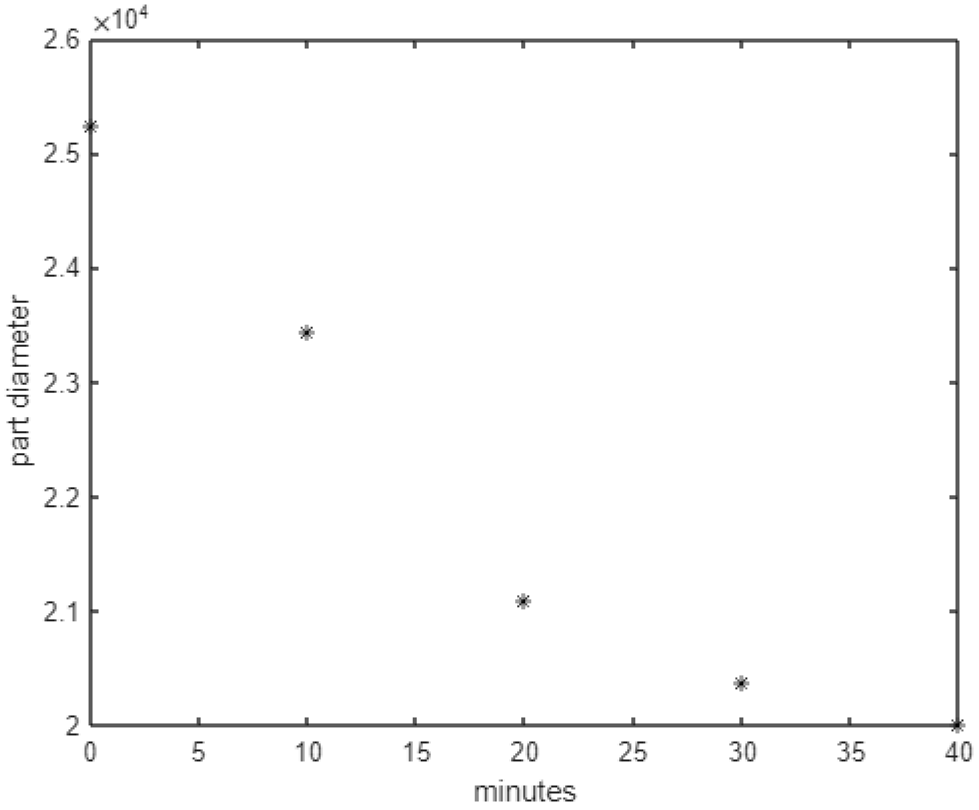
21) A particular part is being turned on a lathe. The diameter of the part is supposed to be 20,000 mm. The diameter is measured every 10 minutes and the results are stored in a file called partdiam.dat. Create a data file to

simulate this. The file will store the time in minutes and the diameter at each time. Plot the data.

```
row1 = [0 25233];
row2 = [10 23432];
row3 = [20 21085];
row4 = [30 20374];
row5 = [40 20002];
mat = [row1;row2;row3;row4;row5]
```

```
mat = 5×2
         0       25233
        10       23432
        20       21085
        30       20374
        40       20002
```

```
save partdiam.dat mat -ascii
load partdiam.dat
mins = partdiam(:,1);
diams = partdiam(:,2);
plot(mins,diams,'k*')
xlabel('minutes')
ylabel('part diameter')
```



22) Create a file called "testtan.dat" comprised of two lines with three real numbers on each line (some negative, some positive, in the -1 to 3

range). The file can be created from the Editor, or saved from a matrix. Then, load the file into a matrix and calculate the tangent of every element in the resulting matrix.

```
mat = rand(2,3)*4-1
```

```
mat = 2×3
   -0.3513    0.2449   -0.3374
    2.1771    1.1141    1.4079
```

```
save testtan.dat mat -ascii
load testtan.dat
tan(testtan)
```

```
ans = 2×3
   -0.3665    0.2499   -0.3508
   -1.4420    2.0354    6.0855
```

23) Write a function calcrectarea that will calculate and return the area of a rectangle. Pass the length and width to the function as input arguments.

```
% function area = MIDTERM_2_PREP(length, width)
% % This function calculates the area of a rectangle
% % Format of call: calcrectarea(length, width)
% % Returns the area
% area = length * width;
% end
% calcrectarea(12,5)
%
% ans =
%
%     60
```

Renewableenergy sources such as biomassare gaining increasing attention. Biomass energy units include megawatt hours (MWh) and gigajoules(GJ). One MWh is equivalent to 3.6 GJ. For example,one cubic meterof wood chips produces 1 MWh.

24) Write a function *mwh_to_gj* that will convert from MWh to GJ.

```
% function out = mwh_to_gj(mwh)
% % Converts from MWh to GJ
%
% % Format of call: mwh_to_gj(mwh)
% % Returns gigajoules
% out = mwh * 3.6;
% end
% mwh_to_gj(34)
%
% ans =
%
%   122.4000
```

25) List some differences between a script and a function.

- A function has a header whereas a script does not.
- A function typically has end at the **end** ofthe file.
- A function is called whereasa script is executed.
- Arguments are passed to functions but not to scripts.
- Functions can return argumentswhereas scripts cannot.
- The block comment is typically in the beginningof a script but under the functionheader.
- The scope of variables is different: scripts use the baseworkspace, whereas functions have their own workspaces.

26) In quantum mechanics, the angular wavelengthfor a wavelength λ is defined as $\frac{\lambda}{2\pi}$. Write a function named *makeitangular* that will receive the wavelength as an input argument, and will return the angular wavelength.

```
% function angwave = makeitangular(wavelength)
% angwave = wavelength/(2*pi);
% end
% makeitangular(250)
%
% ans =
%
%    39.7887
```

27) Write a fives function that will receive two arguments for the number of rows and columns, and will return a matrix with that size of all fives.

```
% function five = fives(r,c)
% % Returns a matrix of fives of specified size
% % Format of call: fives(rows, cols)
% % Returns a rows by cols matrix of all fives
%
% % Initialization
% five = zeros(r,c) + 5;
% end
% ans =
%
%      5     5     5     5
%      5     5     5     5
%      5     5     5     5
```

28) Write a function *isdivby4* that will receive an integer input argument, and will return logical 1 for true if the input argumentis divisible by 4, or logical false if it is not. **Write a function *isdivby4* that will receive an integer input argument, and will return logical 1 for true if the input argumentis divisible by 4, or logical false if it is not.**

```
% function out = isdivby4(inarg)
% % Returns 1 for true if the input argument is
% % divisible by 4 or 0 for false if not
% % Format of call: isdivby4(input arg)
% % Returns whether divisible by 4 or not
% out = rem(inarg,4) == 0;
% end
% isdivby4(12)
%
% ans =
%
%   logical
%
%    1
%
% isdivby4(1)
%
% ans =
%
%   logical
%
%    0
```

29) Write a function *isint* that will receive a number input argument *innum*, and will return 1 for true if this number is an integer, or 0 for false if not. Use the fact that *innum* should be equal to int32(innum) if it is an integer. Unfortunately, due to round-off errors, it should be noted that it is possible to get logical 1 for true if the input argument is close to an integer. Therefore the output may not be what you mightexpect, as shown here.

```
>> isint(4)

ans =

1

>> isint(4.9999)

ans =

0

>> isint(4.9999999999999999999999999999)

ans =
1
```

```
% function out = isint(innum)
% % Returns 1 for true if the argument is an integer
% % Format of call: isint(number)
% % Returns logical 1 iff number is an integer
% out = innum == int32(innum);
% end
% isint(0810)
%
% ans =
%
%   logical
%
%    1
%
% isint(0810.1172)
%
% ans =
%
%   logical
%
%    0
```

30) A Pythagorean triple is a set of positive integers (a,b,c) such that $a^2 + b^2 = c^2$. Write a function ispythag that will receive three positive integers (a, b, c in that order) and will return logical 1 for true if they form a Pythagorean triple, or 0 for false if not.

```
% function out = ispythag(a,b,c)
% % Determines whether a, b, c are a Pythagorean triple or not
% % Format of call: ispythag(a,b,c)
% % Returns logical 1 if a Pythagorean triple
% out = a^2 + b^2 == c^2;
% end
% ispythag(12,24,2)
%
% ans =
%
%   logical
%
%    0
%
% ispythag(3,4,5)
%
% ans =
%
%   logical
%
```

```
%    1
```

31) A function can return a vector as a result. Write a function vecout that will receive one integer argument and will return a vector that increments from the value of the input argument to its value plus 5, using the colon operator. For example,

>> vecout(4)

ans =

4 5 6 7 8 9

```
% function outvec = vecout(innum)
% % Create a vector from innum to innum + 5
% % Format of call: vecout(input number)
% % Returns a vector input num : input num+5
% outvec = innum:innum+5;
% end
% vecout(20)
%
% ans =
%
%     20    21    22    23    24    25
```

32) Write a function called pickone, which will receive one input argument x, which is a vector, and will return one random element from the vector. For example,

>> pickone(4:7)

ans =

5

>> disp(pickone(-2:0))

-1

>> help pickone

pickone(x) returns a random element from vector x

```
% function elem = pickone(invec)
% % pickone(x) returns a random element from vector x
% % Format of call: pickone(vector)
% % Returns random element from the vector
% len = length(invec);
% ran = randi([1, len]);
% elem = invec(ran);
% end
% pickone(4:7)
%
% ans =
%
%      7
```

33) The conversion depends on the temperature and other factors, but an approximation is that 1 inch of rain is equivalent to 6.5 inches of snow. Write a script that prompts the user for the number of inches of rain, calls a function to return the equivalent amount of snow, and prints this result. Write the function, as well!

```
% Prompt the user for a number of inches of rain
% and call a function to calculate the
% equivalent amountof snow
rain = input('How much rain in inches: ');
snow = rainToSnow(rain);
fprintf('%.1f inches of rain would be ', rain)
```

```
25.0 inches of rain would be
```

```
fprintf('%.1f inches of snow\n', snow)
```

```
162.5 inches of snow
```

34) In thermodynamics, the Carnot efficiency is the maximum possible efficiency of a heat engine operating between two reservoirs at different temperatures. The Carnot efficiency is given as

$$\eta = 1 - \frac{T_C}{T_H}$$

where T_C and T_H are the absolute temperatures at the cold and hot reservoirs, respectively. Write a script “carnot” that will prompt the user for the two reservoir temperatures in Kelvin, call a function to calculate the Carnot efficiency, and then print the corresponding Carnot efficiency to 3 decimal places. Also write the function.

```
% Calculates the Carnot efficiency, given the temps
% of cold and hot reservoirs, error-checking both
Tc = input('Enter the cold reservoir temperature: ');
Th = input('Enter the hot reservoir temperature: ');
fprintf('The Cold and Hot tempreature of reservoir is %d and %d respectively\n',Tc,Th)
```

```
The Cold and Hot tempreature of reservoir is 35 and 180 respectively
```

```
carnotEff = calcCarnot(Tc, Th);
```

```
fprintf('The Carnot efficiency is %.3f\n',carnotEff)
```

```
The Carnot efficiency is 0.806
```

35) Many mathematical models in engineering use the exponential function. The general form of the exponential decay function is:

$$y(t) = Ae^{-\tau t}$$

where A is the initial value at t=0, and τ is the time constant for the function. Write a script to study the effect of the time constant. To simplify the equation, set A equal to 1. Prompt the user for two different values for the time constant, and for beginning and ending values for the range of a t vector. Then, calculate two different y vectors using the above equation and the two time constants, and graph both exponential functions on the same graph within the range the user specified. Use a function to calculate y. Make one plot red. Be sure to label the graph and both axes. What happens to the decay rate as the time constant gets larger?

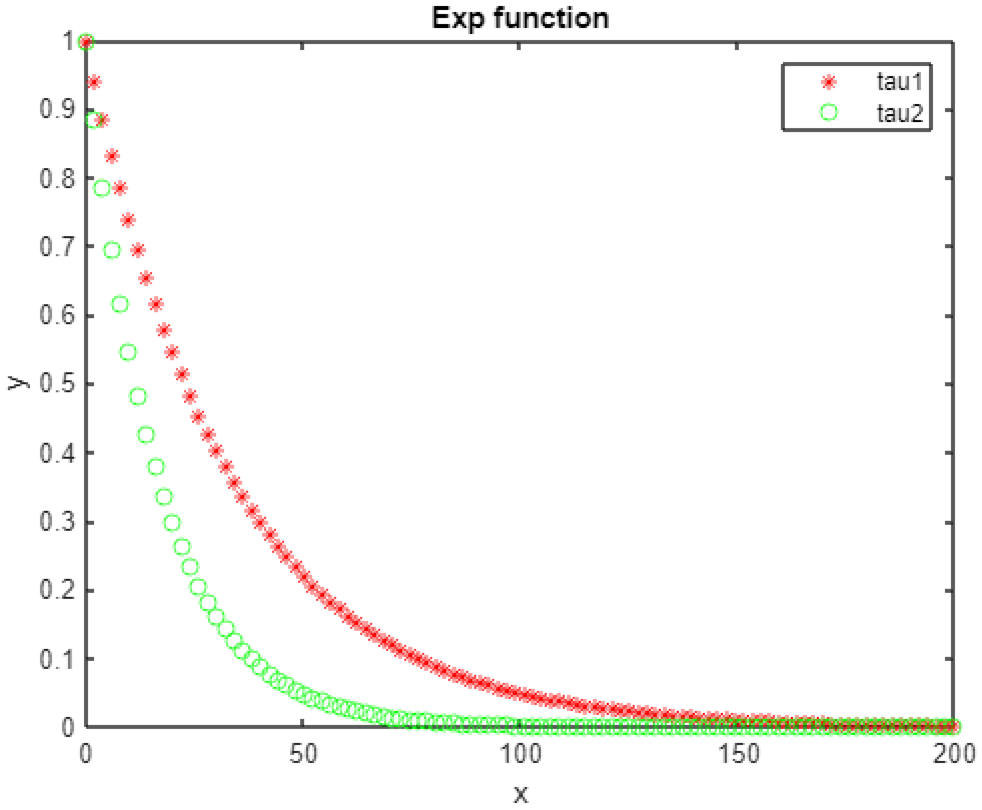
```
A = 1;
tau1 = input('Enter a time constant: ');
tau2 = input('Enter another time constant: ');
tstart = input('Enter the beginning t: ');
tend = input('Enter the end of t: ');
fprintf("The time constant 1 and 2 are %f and %f respectively",tau1,tau2)
```

```
The time constant 1 and 2 are 0.030000 and 0.060000 respectively
```

```
fprintf("The beginning and end of time t are %d and %d respectively",tstart,tend)
```

```
The beginning and end of time t are 0 and 200 respectively
```

```
t = linspace(tstart,tend);
y1 = expfn(A, t, tau1);
y2 = expfn(A, t, tau2);
plot(t,y1,'r*',t,y2,'go')
xlabel('x')
ylabel('y')
title('Exp function')
legend('tau1','tau2')
```


Exp function
tau1
tau2
y
x
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
50
100
150
200

“if ... end” structure

The “IF” statement

The if statement chooses whether another statement, or group of statements, is executed or not. The general form of the if statement is:

```
if condition

action

end
```

For example, the following if statement checks to see whether the value of a variable is negative. If it is, the value is changed to a zero; otherwise, nothing is changed.

```
if num < 0

num = 0

end
```

sqrtifexmp.m

```
% Prompt the user for a number and print its sqrt
num = input ( ' Please enter a number: ' ) ;
```

```
The number entered is : -4.5
```

```
fprintf("The number entered is : %.1f",num)
% If the user entered a negative number, change it
if num < 0
num = 0 ;
end
fprintf ( 'The sqrt of : %.1f is : %.1f\n' , num, sqrt (num))
```

```
The sqrt of : 0.0 is : 0.0
```

sqrtifexampii .m

```
% Prompt the user for a number and print its sqrt
num = input ( ' Please enter a number: ' ) ;
fprintf("The number entered is : %.1f",num)
```

```
The number entered is : -25.0
```

```
% If the user entered a negative number, tell the user and change it
if num < 0
    disp('OK, we''ll use the absolute value' )
    num = abs (num);
end
```

```
OK, we'll use the absolute value
```

```
fprintf ( 'The sqrt of %.1f is %.1f\n' , num, sqrt(num) )
```

```
The sqrt of 25.0 is 5.0
```

createvec.m

```
function outvec = createvec (mymin, mymax)
% createvec creates a vector that iterates from a
% specified minimum to a maximum
% Format of call: createvec (minimum, maximum)
% Returns a vector If the "minimum" isn't smaller than the "maximum",
% exchange the values using a temporary variable
if mymin > mymax
    temp = mymin;
    mymin = mymax;
    mymax = temp ;
end
% Use the colon operator to create the vector
outvec = mymin : mymax ;
end
```

Output

```
createvec(4,10)

ans =

4     5     6     7     8     9    10
```

The “if-else” Statement

The if statement chooses whether or not an action is executed. Choosing

between two actions, or choosing from among several actions, is accomplished

using if-else, nested if-else, and switch statements.

The if-else statement is used to choose between two statements, or sets of statements. The general form is:

if condition

action1

else

action2

end

For example, to determine and print whether or not a random number in the

range from 0 to 1 is less than 0.5, an if-else statement could be used:

```
if rand < 0.5
    disp ( 'It was less than . 5!' )
else
    disp ( ' It was not less than . 5! ')
end
```

```
It was not less than . 5!
```

One application of an if-else statement is to check for errors in the inputs to a script (this is called error-checking). For example, an earlier script prompted the user for a radius, and then used that to calculate the area of a circle. However, it did not check to make sure that the radius was valid (e.g., a positive number). Here is a modified script that checks the radius:

`checkradius.m`

```
% This script calculates the area of a circle
% It error-checks the user's radius
radius = input ( ' Please enter the radius: ') ;
if radius <= 0
    fprintf ( 'Sorry; %.2f is not a valid radius\n' , radius)
else
    area = pi*radius*radius ;
    fprintf ( ' For a circle with a radius of %.2f, ' , radius)
    fprintf ( ' the area is : %.2f\n' , area)
end
```

```
For a circle with a radius of 12.00,
the area is : 452.39
```

Nested “if-else” Statement

`y =1 if x < -1`

`y = x^2 if - 1 <= x <=2`

`y = 4 if x > 2`

The value of y is based on the value of x, which could be in one of three possible

ranges. Choosing which range could be accomplished with three separate if

statements, is as follows:

```
x = 2;
if x < -1
    y= 1;
end
if x >= -1 && x <=2

    y = x^2;
end
if x > 2
```

```
    y = 4;
end
fprintf("y = %d",y)
```

```
y = 4
```

“elseif” Statement

For example, the previous example could be written using the elseif clause

rather than nesting if-else statements:

```
if x < -1

y = 1;

elseif x <= 2

y = x^2;

else

y = 4;

end
```

calcy.m

```
function y = calcy (x)
%  calcy calculates y as a function of x
%  Format of call: calcy (x)
if x <-1
    y = 1;
elseif x <= 2
    y = x^2;
else
    y = 4;
end
end
```

Output

```
x = 1.1

x =

1.1000

y = calcy(x)

y =

1.2100
```

findargtype .m

```
function outtype = findargtype(inputarg)
% findargtype determines whether the input
% argument is a scalar, vector, or matrix
% Format of call: findargtype (inputArgument)
% Returns a string

[r c] = size (inputarg) ;
if r==1 && c ==1
    outtype = 'scalar' ;
elseif r==1 || c==1
    outtype = 'vector' ;
else
    outtype = 'matrix';
end
end
```

Output

```
findargtype([2,4,5])


ans =


'vector'


findargtype(2)


ans =


'scalar'
```

letgrade.m

```
% function grade = letgrade (quiz)
% % letgrade returns the letter grade corresponding
% % to the integer quiz grade argument
% % Format of call: letgrade (integerQuiz)
% % Returns a character
% % First, error-check
% if quiz < 0 || quiz > 10
%     grade = 'x' ;
% % If here, it is valid so figure out the
% % corresponding letter grade
% elseif quiz == 9 || quiz == 10
%     grade = 'A' ;
% elseif quiz == 8
%     grade = 'B' ;
% elseif quiz == 7
%     grade = 'c' ;
% elseif quiz == 6
%     grade = 'D' ;
% else
%     grade = 'F' ;
% end
% end
```

Output

```
quiz = 8


quiz =


8


lettergrade = letgrade(quiz)


lettergrade =


'B'


quiz = 4


quiz =


4


lettergrade = letgrade(quiz)


lettergrade =
'F'
```

Practice 4.1

Write an if statement that would print "Hey, you get overtime!" if the value of a variable hours is greater than 40. Test the if statement for values of hours less than, equal to, and greater than 40. Will it be easier to do this in the Command Window or in a script?

```
% Enter variable hours
variableHours = input('Enter Variable Hours : ');
% Variable Hours greater than 40
if variableHours > 40
    fprintf("The hours worked are : %d",variableHours)
    fprintf('Hey! you get overtime! \n');
end
```

```
The hours worked are : 50
Hey! you get overtime!
```

Practice 4.3

Modify the function findargtype to return ‘scalar’, ‘row vector’, ‘column vector’, or ‘matrix’ depending on the input argument

```
% function type = findargtype2(X)
%
% findargtype2 finds if the input is a scalar, row vector,
%
% column vector, or matrix.
%
% Format of call: findargtype2(X)
%
% Returns a string
%
% [r c] = size(X);
%
% if r == 1 && c == 1
%
% type = 'scalar';
%
% elseif r == 1
%
% type = 'row vector';
%
% elseif c == 1
%
% type = 'column vector';
%
```

```
% else
%
% type = 'matrix';
%
% end
%
% end
```

Output

```
findargtype2([2;4;5])


ans =


'column vector'


findargtype2([2,4,5])


ans =


'row vector'
```

Practice 4.4

Modify the original function findargtype to use three separate if statements instead of a nested if-else statement.

```
% function outtype = findargtype_prac(inputarg)
% findargtype determines whether the input argument is
%a scalar, vector or matrix and returns a string
[r, c] = size(inputarg);
if r == 1 && c == 1
    outtype = 'scalar';
end
if r == 1 && c > 1
    outtype = 'vector';
end
if r > 1 && c > 1
    outtype = 'matrix';
end
end
```

Output

```
findargtype_prac(2)

ans =

'scalar'
```

Practice 4.2

Write a script printsindegorrad that will:

1. Prompt the user for an angle. :

2. Prompt the user for (r)adians or (d)egrees, with radians as the default. :

3. If the user enters ‘d’, the sind function will be used to get the sine of the angle in degrees; otherwise, the sin function will be used. Which sine function to use will be based solely on whether the user enters a ‘d’ or not. A ‘d’ means degrees, so sind is used; otherwise, for any other character the default of radians is assumed so sin is used. :

4. Print the result.

Here are examples of running the script:

>> printsindegorrad

Enter the angle: 45

(r)adians (the default) or (d)egrees: d

The sin is 0.71

>> printsindegorrad

Enter the angle: pi

(r)adians (the default) or (d)egrees: r

The sin is 0.00

```
% This script shows how to use an if-else statement

% Ask for the angle

ang = input('Enter the angle: ');

% Ask for the units

u = input('Enter d for degrees or r for radians: ','s');

% Test to select which sin function to use, sind or sin
```

```
if u == 'd'

res = sind(ang);

else

res = sin(ang);

end

% Display the answer

fprintf('sin(%.2f) = %.2f\n',ang,res)
```

Output

```
printsindegorrad

Enter the angle: 45

Enter d for degrees or r for radians: r

sin(45.00) = 0.85

printsindegorrad

Enter the angle: pi

Enter d for degrees or r for radians: r

sin(3.14) = 0.00
```

“for ... end” structure

Question

Loop through the matrix and assign each element a new value. Assign 2 on the main diagonal, -1 on the adjescent diagonals and 0 everywhere else.

Given: nrows=4; nclos=6; A=ones(nrows,nclos);

```
nrows=4;
nclos=6;
A=ones(nrows,nclos)
```

```
A = 4×6
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
     1     1     1     1     1     1
```

```
for c = 1:nclos
    for r = 1:nrows
        if r == c
            A(r, c) = 2;
        elseif abs(r-c) == 1
            A(r, c) = -1;
        else
            A(r, c) = 0;
        end
    end
end
A
```

```
A = 4×6
     2    -1     0     0     0     0
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
```

example using “any"

```
limit = 0.75;
A = rand(10,1)
```

```
A = 10×1
    0.1576
    0.9706
    0.9572
    0.4854
    0.8003
    0.1419
    0.4218
    0.9157
    0.7922
    0.9595
```

```
if any(A > limit)
```

```
    disp( 'There is at least one value above the limit. ')
else
    disp( 'All values are below the limit. ')
end
```

```
There is at least one value above the limit.
```

example using “isequal"

Test Arrays for Equality

Compare arrays using i sequal rather than the == operator to test for equality, because ==

results in an error when the arrays are different sizes.

Create two arrays.

```
A = ones (3,4)
```

```
A = 3×4
    1     1     1     1
    1     1     1     1
    1     1     1     1
```

```
B = rand(3,4)
```

```
B = 3×4
   0.6557    0.9340    0.7431    0.1712
   0.0357    0.6787    0.3922    0.7060
   0.8491    0.7577    0.6555    0.0318
```

If size(A) and size(B) are the same, concatenate the arrays; otherwise, display a warning

and return an empty array.

```
if isequal(size(A),size(B))
    C = [A;B] % append
else
    disp( 'A and B are not the same size. ')
    C = [];
end
```

```
C = 6×4
   1.0000    1.0000    1.0000    1.0000
   1.0000    1.0000    1.0000    1.0000
   1.0000    1.0000    1.0000    1.0000
   0.6557    0.9340    0.7431    0.1712
   0.0357    0.6787    0.3922    0.7060
   0.8491    0.7577    0.6555    0.0318
```

Evaluate multiple condition in an expression

Q: Determine if a value falls within a specified range.

```
x = 10;
minVal = 2;
```

```
maxVal = 6;
if (x >= minVal) && (x <= maxVal)

    disp('Value within specified range. ')
elseif (x > maxVal)
    disp( 'Value exceeds maximum value. ')
else
    disp( 'Value is below minimum value. ')
end
```

```
Value exceeds maximum value.
```

Compare Vectors Containing NaN Values

Create three vectors containing NaN values.

```
A1 = [1 NaN NaN]
```

```
A1 = 1×3
     1   NaN   NaN
```

```
A2 = [1 NaN NaN]
```

```
A2 = 1×3
     1   NaN   NaN
```

```
A3 = [1 NaN NaN]
```

```
A3 = 1×3
     1   NaN   NaN
```

Compare the vectors for equality.

```
tf = isequaln(A1, A2, A3)
```

```
tf = logical
   1
```

The result is logical 1 (true) because isequaln treats the NaN values as equal to each other.

The “for” loop

The general form of the for loop is:

for loopvar = range

action

end

where loopvar is the loop variable, "range" is the range of values through which the loop variable is to iterate, and the action of the loop consists of all state- ments up to the end. Just like with if statements, the action is indented to make it easier to see. The range can be specified using any vector, but normally the easiest way to specify the range of values is to use the colon operator.

The “for” loop: Question

How could you print this column of integers [using the programming method]:

0

50

100

150

200

```
for i = 0:50:200
    fprintf("%3d\n",i)
end
```

```
  0
 50
100
150
200
```

The “for” loop that don’t use iterator

```
for i = 1:3
    fprintf("I will not chew gum\n")
end
```

```
I will not chew gum
I will not chew gum
I will not chew gum
```

`forecho.m`

```
% This script loops to repeat the action of prompting the user for a number and echo-printing i
for iv = 1:3
    inputnum = input ( ' Enter a number: ') ;
    fprintf ('You entered : %.1f\n' , inputnum)
end
```

```
You entered : 33.0
You entered : 1.1
You entered : 50.0
```

Find sum and product

`sumnnums.m`

```
% sumnnums calculates the sum of the n numbers entered by the user
n = randi ( [3 10]) ;
runsum = 0 ;
for i = 1:n
```

```
    inputnum = input ( 'Enter a number: ') ;
    runsum = runsum + inputnum;
end
fprintf('The sum is %.2f \n' , runsum)
```

```
The sum is 890.30
```

Preallocating vector

forgenvec.m

```
% forgenvec creates a vector of length n It prompts the user and puts n numbers into a vector
n = randi ( [4 8]) ;
numvec = zeros (1, n) ;
for iv = 1:n
    inputnum = input ( ' Enter a number: ' ) ;
    numvec (iv) = inputnum;
end
fprintf ('The vector is: \n')
```

```
The vector is:
```

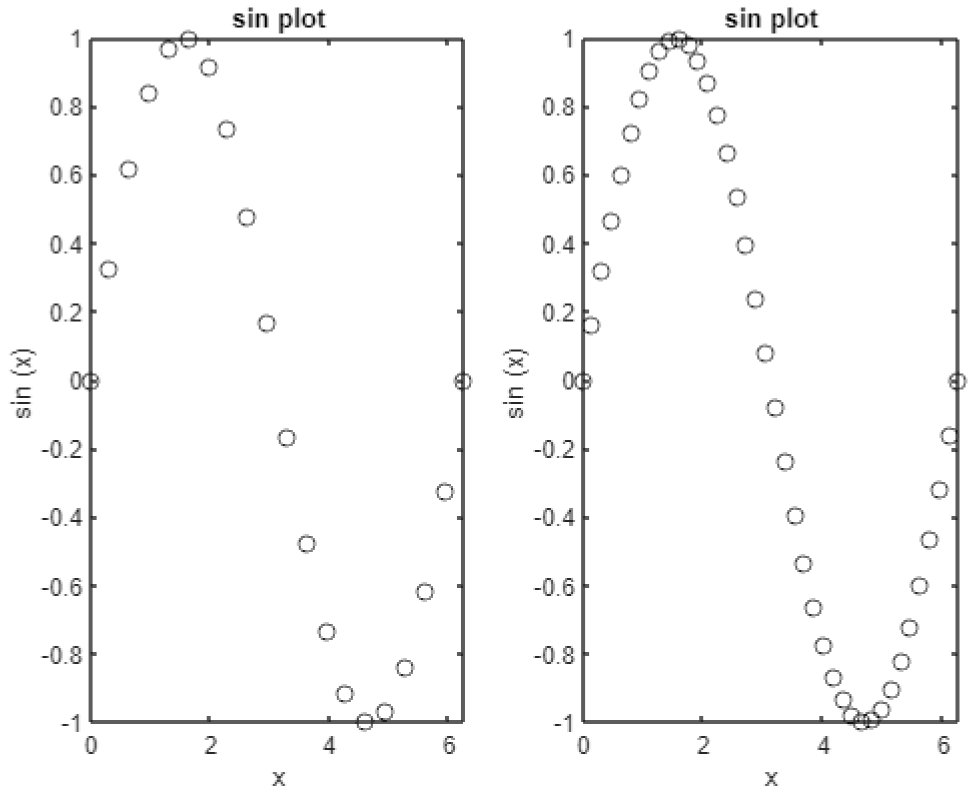
```
disp (numvec)
```

```
   44.0000    2.3000    1.1000    3.2000
```

“for” loop-subplot

subplotex.m

```
% Demonstrates subplot using a for loop
for i = 1:2
    x = linspace (0, 2*pi , 20*i) ;
    y = sin (x) ;
    subplot (1, 2, i)
    plot (x, y, 'ko' )
    xlabel ( 'x' )
    ylabel ( 'sin (x) ')
    title ('sin plot')
end
```



Nested “for” loop

- For every row of output:
- Print the required number of stars
- Move the cursor down to the next line (print \n')

printstars.m

```
% Prints a box of stars
% How many will be specified by two variables
% for the number of rows and columns
rows = 3;
columns = 5;
% loop over the rows
for i=1:rows
% for every row loop to print *'s and then one \n
    for j=1:columns
        fprintf ('*')
    end
    fprintf ('\n')
end
```

```
*****

*****
```

```
*****
```

printtristars.m

```
% Prints a triangle of stars
% How many will be specified by a variable
% for the number of rows
rows = 3 ;
for i=1:rows
%inner loop just iterates to the value of i
    for j=1:i
        fprintf ( '*' )
    end
    fprintf ( ' \n')
end
```

```
*

**

***
```

printloopvars . m

```
% Displays the loop variables
for i = 1:3
    for j = 1:2
        fprintf ('i=%d,j=%d\n', i, j)
    end
    fprintf (' \n' )
end
```

```
i=1,j=1
i=1,j=2

i=2,j=1
i=2,j=2

i=3,j=1
i=3,j=2
```

multtable. m

```
% function outmat = multtable (rows, columns)
% % multtable returns a matrix which is a
% % multiplication table
% % Format : multtable (nRows, nColumns)
% % Preallocate the matrix
% outmat = zeros (rows , columns) ;
% for i = 1 : rows
%     for j = 1: columns
%         outmat (i , j) = i*j ;
%     end
% end
% end
```

Output

```
multtable(3,5)


ans =


1    2    3    4    5

2    4    6    8    10

3    6    9    12    15
```

```
createmulttab. m
```

```
% Prompt the user for rows and columns and create a multiplication table to store in a file "my
num_rows = input ( 'Enter the number of rows : ' ) ;
num_cols = input ( 'Enter the number of columns: ' ) ;
multmatrix = multtable (num_rows, num_cols) ;
save mymulttable.dat multmatrix -ascii
```

Output

```
Enter the number of rows : 6

Enter the number of columns: 4

load mymulttable.dat

mymulttable


mymulttable =


1    2    3    4

2    4    6    8

3    6    9    12

4    8    12    16

5    10    15    20
```

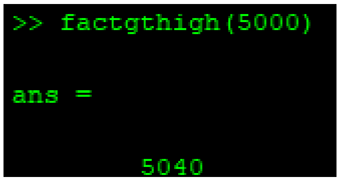
6 12 18 24

“while ... end" loop

factgthigh.m

```
% function facgt = factgthigh (high)
% % factgthigh returns the first factorial > input
% % Format : factgthigh (inputInteger)
%
% i=0 ;
% fac=1;
% while fac <= high
%     i=i+1;
%     fac = fac * i;
% end
% facgt = fac;
% end
```

Output



```
>> factgthigh(5000)

ans =

        5040
```

Input in “while” loop

whileposnum. m

```
% Prompts the user and echo prints the numbers entered
% until the user enters a negative number
inputnum=input ( ' Enter a positive number: ') ;
while inputnum >= 0
    fprintf ( 'You entered a %d. \n\n' , inputnum)
    inputnum = input ( ' Enter a positive number: ') ;
end
```

```
You entered a 6.
```

```
fprintf ( 'OK! \n' )
```

```
OK!
```

Counting in “while” loop

countposnum . m

```
% Prompts the user for positive numbers and echo prints as
% long as the user enters positive numbers
% Counts the positive numbers entered by the user
```

```
counter=0;
inputnum=input ( ' Enter a positive number: ') ;
while inputnum >= 0
    fprintf ( 'You entered a %d. \n\n' , inputnum)
    counter = counter+1;
    inputnum = input ( ' Enter a positive number: ') ;
end
```

```
You entered a 4.

You entered a 11.
```

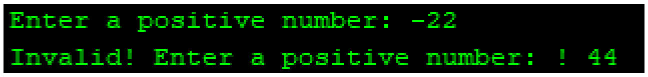
```
fprintf ( 'Thanks, you entered %d positive numbers. \n' , counter)
```

```
Thanks, you entered 2 positive numbers.
```

readonenum.m

```
% Loop until the user enters a positive number
inputnum=input ( 'Enter a positive number: ' ) ;

while inputnum < 0
    inputnum = input ( 'Invalid! Enter a positive number: ! ') ;
end
fprintf ( ' Thanks, you entered a %.1f \n' , inputnum)
```

```
 Thanks, you entered a 44.0
```



```
Enter a positive number: -22
Invalid! Enter a positive number: ! 44
```

Practise question

Practice 5.1

Write a for loop that will print a column of five *’s.

```
% Prints a column of five *'s

n=5;
for i=1:n
    fprintf('* \n');
end
```

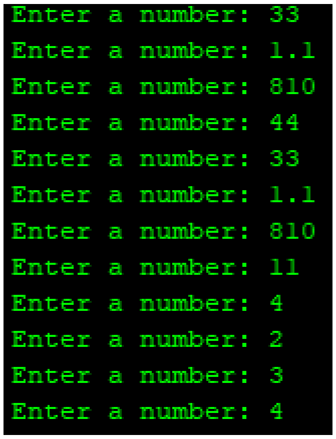
```
*
*
*
*
*
```

Practice 5.2

Write a function prodnnums that is similar to the sumnnums function, but will calculate the product of the numbers entered by the user.

```
% prodnnums calculates the sum of the n numbers entered by the user
n = randi ( [3 10]) ;
runprod = 1 ;
for i = 1:n
    inputnum = input ( 'Enter a number: ') ;
    runprod = runprod * inputnum;
end
fprintf('The sum is %.2f \n' , runprod)
```

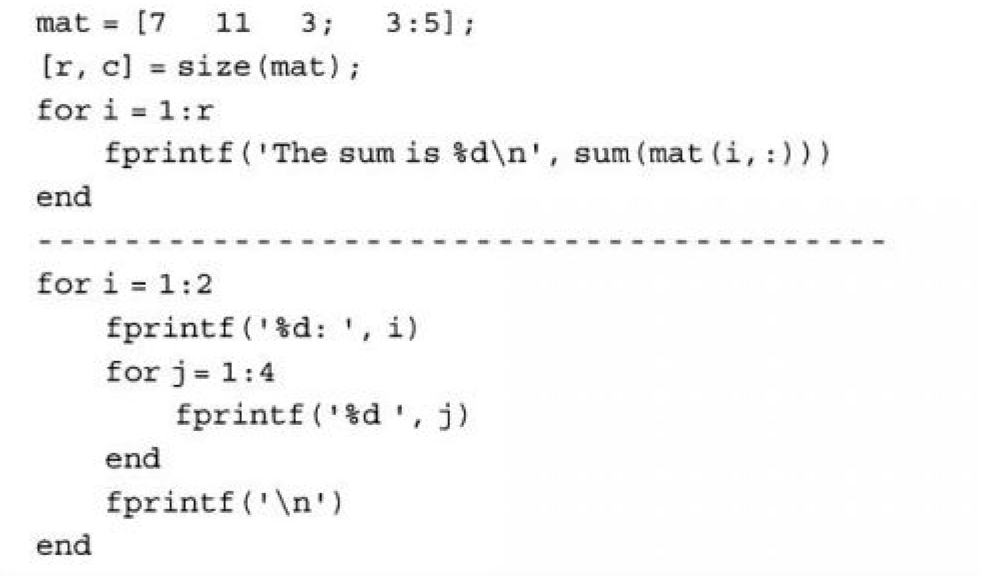
```
The sum is 1366180992.00
```



```
Enter a number: 33
Enter a number: 1.1
Enter a number: 810
Enter a number: 44
Enter a number: 33
Enter a number: 1.1
Enter a number: 810
Enter a number: 11
Enter a number: 4
Enter a number: 2
Enter a number: 3
Enter a number: 4
```

Practice 5.3

For each of the following (they are separate), determine what would be printed. Then, check your answers by trying them in MATLAB.



```
mat = [7   11   3;   3:5];
[r, c] = size(mat);
for i = 1:r
    fprintf('The sum is %d\n', sum(mat(i,:)))
end
----------------------------------------
for i = 1:2
    fprintf('%d: ', i)
    for j = 1:4
        fprintf('%d ', j)
    end
    fprintf('\n')
end
```

In the first script, variables *r* and *c* store the matrix dimensions 2 and 3. Then *for* loop in the script prints the sum of the rows of the matrix as

The sum is 21

The sum is 12

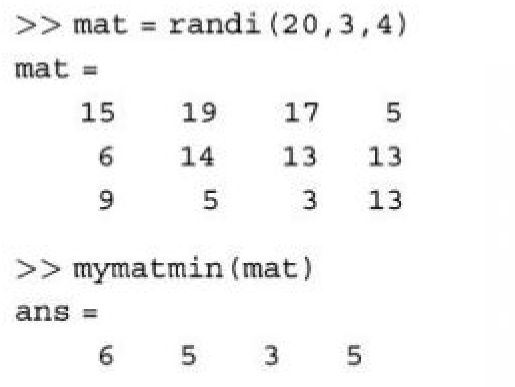
In the second script, since a *for* loop is called two times using a *for* loop the inner *for* loop is executed two times and prints the following.

1: 1,2,3,4

2: 1,2,3,4

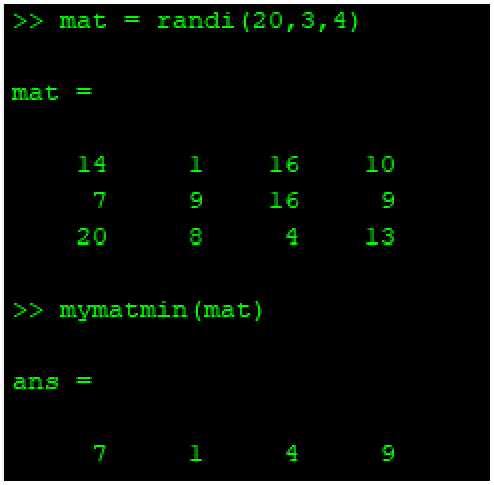
Practice 5.4

Write a function mymatmin that finds the minimum value in each column of a matrix argument and returns a vector of the column minimums. An example of calling the function follows:



```
>> mat = randi(20,3,4)
mat =
    15    19    17     5
     6    14    13    13
     9     5     3    13

>> mymatmin(mat)
ans =
     6     5     3     5
```

```
% displays the vector of minimums in columns
% function output=mymatmin(matrix)
% [r, c]=size(matrix);
% output=zeros(1,c);
% for j=1:c
%     output(1,j)=min(matrix(:,j));
% end
```



```
>> mat = randi(20,3,4)

mat =

    14     1    16    10
     7     9    16     9
    20     8     4    13

>> mymatmin(mat)

ans =

     7     1     4     9
```

Practice 5.5

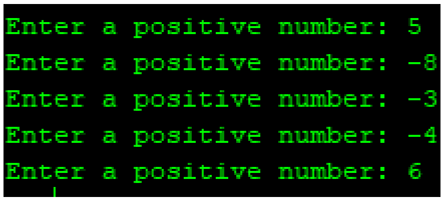
Write a script avenegnum that will repeat the process of prompting the user for negative numbers, until the user enters a zero or positive number, as just shown. Instead of echo printing them, however, the script will print the average (of just the negative numbers). If no negative numbers are entered, the script will print an error message instead of the average. Use the programming method. Examples of executing this script follow:

```
>>avenegnum

Enter a negative number: 5

No negative numbers to average.

>>avenegnum

Enter a negative number: -8

Enter a negative number: -3

Enter a negative number: -4

Enter a negative number: 6

The average was -5.00
```

```
% calculates the average of the negative numbers
innum=input ( 'Enter a positive number: ') ;
if innum>=0
    fprintf ('No negative numbers to average \n' )
end
n=0 ;
sum=0 ;
while innum<0
    sum=sum+innum;
    n=n+1;
    innum=input ( 'Enter a positive number: ') ;
end
average=sum/n;
```

```
if average<0
    fprintf ( 'The average is %6.2f\n', average) ;
end
```

```
The average is  -5.00
```



```
Enter a positive number: 5
Enter a positive number: -8
Enter a positive number: -3
Enter a positive number: -4
Enter a positive number: 6
```

Other flow structure

Repeat Statements Until Expression Is False

Use a while loop to calculate factorial (10)

```
n = 10;
f = n;
while n > 1
    n = n-1;

    f = f*n;
end
disp(['n! = ' num2str(f) ])
```

```
n! = 3628800
```

```
% num2str - converts numeric array to character array
```

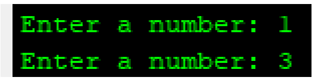
Switch,case, otherwise

Compare Single Values

Display different text conditionally, depending on a value entered at the command prompt

```
n = input( 'Enter a number: ' );

switch n

    case -1

        disp('negative one')
    case 0
        disp( 'zero')
    case 1
        disp('positive one')
    otherwise
        disp( 'other value' )
end
```

```
other value
```

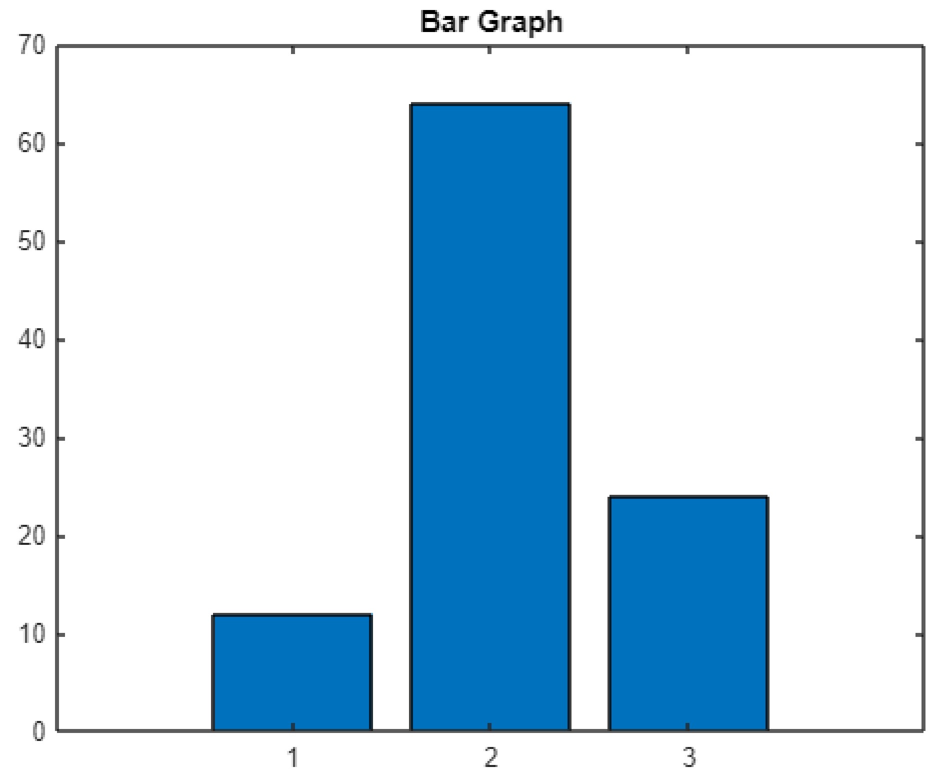


```
Enter a number: 1
Enter a number: 3
```

Compare against multiple value

```
x = [12 64 24];
plottype = "bar";
switch plottype
    case "bar"
```

```
        bar (x)
        title("Bar Graph")
    case {"pie", "pie3"}
        pie3(x)
        title('Pie Chart')
    otherwise
        warning( "Unexpected plot type. No plot created.")
end
```


Bar Graph