

Lab Assignment Week 4

Ques 1

```
U = [9; 1], V = [2 -1;1 0], W = [5 -2;1 4;8 2]
```

```
U = 2x1
```

```
9
```

```
1
```

```
V = 2x2
```

```
2 -1
```

```
1 0
```

```
W = 3x2
```

```
5 -2
```

```
1 4
```

```
8 2
```

```
transpose(W)
```

```
ans = 2x3
```

```
5 1 8
```

```
-2 4 2
```

```
V * U
```

```
ans = 2x1
```

```
17
```

```
9
```

```
W * V
```

```
ans = 3x2
```

```
8 -5
```

```
6 -1
```

```
18 -8
```

```
inv(V)
```

```
ans = 2x2
```

```
0 1
```

```
-1 2
```

```
inv(V) * U
```

```
ans = 2x1
```

```
1
```

```
-7
```

Ques 2

```
V = [4 3 2 1],W = [5 -2 0 0],X = [2;1;0;2],A = [1,0;0,4],B = [4 1 -2;3 7 0],C = [5;8],D = [8]
```

```

V = 1x4
    4    3    2    1
W = 1x4
    5   -2    0    0
X = 4x1
    2
    1
    0
    2
A = 2x2
    1    0
    0    4
B = 2x3
    4    1   -2
    3    7    0
C = 2x1
    5
    8
D = 8

```

A*B

```

ans = 2x3
    4    1   -2
   12   28    0

```

A.*B

Arrays have incompatible sizes for this operation.

[Related documentation](#)

A*A

```

ans = 2x2
    1    0
    0   16

```

B*C

Error using `*`
 Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on each element of the matrix individually, use `TIMES (.*)` for elementwise multiplication.

[Related documentation](#)

C*B

Error using `*`
 Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on each element of the matrix individually, use `TIMES (.*)` for

elementwise
multiplication.

[Related documentation](#)

B*D

```
ans = 2x3
    32     8   -16
    24    56     0
```

A^2

```
ans = 2x2
     1     0
     0    16
```

A.^2

```
ans = 2x2
     1     0
     0    16
```

2*V

```
ans = 1x4
     8     6     4     2
```

V*W

Error using *
Incorrect dimensions for matrix multiplication. Check that the number of columns in the first matrix matches the number of rows in the second matrix. To operate on each element of the matrix individually, use TIMES (.*) for elementwise multiplication.

[Related documentation](#)

V/5

```
ans = 1x4
    0.8000    0.6000    0.4000    0.2000
```

V./W

```
ans = 1x4
    0.8000   -1.5000         Inf         Inf
```

W*X

ans = 8

W.*X

```
ans = 4x4
    10    -4     0     0
     5    -2     0     0
     0     0     0     0
    10    -4     0     0
```

B+C

```
ans = 2x3
     9     6     3
    11    15     8
```

W-D

```
ans = 1x4
    -3   -10    -8    -8
```

Ques 3

I4 = eye(4)

```
I4 = 4x4
     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

I4(2,:) = ones

```
I4 = 2x4
     0     0     0    -2
     1     1     1     1
```

D = I4

```
D = 2x4
     0     0     0    -2
     1     1     1     1
```

I4(:,4) = -2

```
I4 = 2x4
     0     0     0    -2
     1     1     1    -2
```

D = I4

```
D = 2x4
     0     0     0    -2
     1     1     1    -2
```

Ques 4

Using Matrix Inverse Method

```
syms x y z
eqn1 = 5.7*x - 2.3*y + 0.9*z == 30.1;
eqn2 = -2.5*x + 1.3*y + 0*z == -12.6;
eqn3 = 8.0*x + 4.2*y - 7.5*z == 1.1;
[A,B] = equationsToMatrix([eqn1,eqn2,eqn3],[x,y,z])
```

A =

$$\begin{pmatrix} \frac{57}{10} & -\frac{23}{10} & \frac{9}{10} \\ -\frac{5}{2} & \frac{13}{10} & 0 \\ 8 & \frac{21}{5} & -\frac{15}{2} \end{pmatrix}$$

B =

$$\begin{pmatrix} \frac{301}{10} \\ -\frac{63}{5} \\ \frac{11}{10} \end{pmatrix}$$

```
x = linsolve(A,B)
```

x =

$$\begin{pmatrix} 4 \\ -2 \\ 3 \end{pmatrix}$$

```
D = [A B]
```

D =

$$\begin{pmatrix} \frac{57}{10} & -\frac{23}{10} & \frac{9}{10} & \frac{301}{10} \\ -\frac{5}{2} & \frac{13}{10} & 0 & -\frac{63}{5} \\ 8 & \frac{21}{5} & -\frac{15}{2} & \frac{11}{10} \end{pmatrix}$$

Using Gaussian Elimination

```
[m,n]=size(D);
for j=1:m-1
    for z=2:m
        if D(j,j)==0
            t=D(j,:);D(j,:)=D(z,:);
```

```

        D(z,:)=t;
    end
end
for i=j+1:m
    D(i,:)=D(i,:)-D(j,:)*(D(i,j)/D(j,j));
end
end
x=zeros(1,m);
for s=m:-1:1
    c=0;
    for k=2:m
        c=c+D(s,k)*x(k);
    end
    x(s)=(D(s,n)-c)/D(s,s);
end
D

```

D =

$$\begin{pmatrix} \frac{57}{10} & -\frac{23}{10} & \frac{9}{10} & \frac{301}{10} \\ 0 & \frac{83}{285} & \frac{15}{38} & \frac{343}{570} \\ 0 & 0 & -\frac{1563}{83} & -\frac{4689}{83} \end{pmatrix}$$

x'

```

ans = 3x1
     4
    -2
     3

```

Ques 5

```

B = [1 2 3;3 4 5];
[m n]=size(B);
myTranspose=zeros(n,m);
for i=1:n
    for j=1:m
        myTranspose(i,j)=B(j, i);
    end
end
B

```

B = 2x3

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 3 | 4 | 5 |

myTranspose

```

myTranspose = 3x2
     1     3
     2     4

```


Question 1

$$2^5/(2^5-1) \text{ and compare with } [1-(1/2^5)]-1$$

$$2^5/(2^5-1)$$

$$\text{ans} = 1.0323$$

$$[1-(1/2^5)]-1$$

$$\text{ans} = -0.0313$$

Question 2

$$[3(\sqrt{5}-1)/(\sqrt{5}+1)*2] - 1$$

$$[3*(\text{sqrt}(5)-1)/(\text{sqrt}(5)+1)*2]-1$$

$$\text{ans} = 1.2918$$

Question 3

$$\text{Area} = \pi r^2 \text{ with } r=\pi^{1/3}-1$$

$$r = \pi^{(1/3)}-1$$

$$r = 0.4646$$

$$\text{Area} = \pi * r^2$$

$$\text{Area} = 0.6781$$

Question 4

$$e^3, \ln(e^3), \log_{10}(e^3), \log_{10}(10^5)$$

$$\exp(3)$$

$$\text{ans} = 20.0855$$

$$\log(\exp(3))$$

$$\text{ans} = 3$$

$$\log_{10}(\exp(3))$$

$$\text{ans} = 1.3029$$


```
log10(10^5)
```

```
ans = 5
```

Question 5

$$e^{\pi\sqrt{163}}, \sin(2\pi/6) + \cos(2\pi/6)$$

```
exp(pi*sqrt(163))
```

```
ans = 2.6254e+17
```

```
sin((2*pi)/6)+cos((2*pi)/6)
```

```
ans = 1.3660
```

Question 6

$$y = \cosh 2x - \sinh 2x; \text{ where } x = 32\pi$$

```
x = 32*pi
```

```
x = 100.5310
```

```
y = cosh(2*x) - sinh(2*x)
```

```
y = 0
```

Question 7

$$\text{Solve } 3^x = 17 \text{ for } x$$

```
x = log(17)/log(3)
```

```
x = 2.5789
```

Question 8

$$1 + (3i/(1-3i)), e^{i\pi/4}$$

```
1+(3*i/(1-(3*i)))
```

```
ans = 0.1000 + 0.3000i
```

```
exp((i*pi)/4)
```

```
ans = 0.7071 + 0.7071i
```

Question 9

Execute the commands $\exp(\pi/2*i)$ and $\exp(\pi/2i)$.

```
exp(pi/2*i)
```

```
ans = 0.0000 + 1.0000i
```

```
exp(pi/2i)
```

```
ans = 0.0000 - 1.0000i
```

Question 10

$\cot(0)$, $\tan^{-1}(\infty)$

```
cot(0)
```

```
ans = Inf
```

```
(tan(inf))^-1
```

```
ans = NaN
```

| | |
|--------------------------------|-----------------------|
| Roll. No. A016 | Name: Varun Khadayate |
| Class B.Tech CSBs | Batch: 1 |
| Date of Experiment: 25-07-2021 | Subject: IT/WS |

1. The sum of a geometric series

$$1+r+r^2+r^3+\dots+r^n = (1-r^{n+1})/(1-r);$$

N=number of terms in a series.

Accept the value of r and n as input from keyboard. Verify the above equation

```

r = 2

r =

    2

>> N = 20

N =

    20

>> x = (1-r^N)/(1-r)

x =

    1048575

>> y = sum(r.^(0:N-1))

y =

    1048575

>> logical(x == y)

ans =

    logical

    1

```

2. Accept a square matrix A of any size from keyboard.

```
>> A = [9,9,3;7,2,7;2,8,5]
```

```
A =
```

```
9 9 3
7 2 7
2 8 5
```

Find:

- a. Size of A matrix

```
>> size(A)
```

```
ans =
```

```
3 3
```

- b. Determinant of A matrix

```
>> det(A)
```

```
ans =
```

```
-447.0000
```

- c. Display whether matrix A is singular or not

```
>> cond(A)
```

```
ans =
```

```
4.0913
```

- d. Transpose of A matrix

```
>> B = transpose(A)
```

```
B =
```

```
9 7 2
9 2 8
3 7 5
```

e. Perform $A+A'$, $A-A'$

```
>> C = A + B
```

```
C =
```

```
18 16 5
16 4 15
5 15 10
```

```
>> D = A - B
```

```
D =
```

```
0 2 1
-2 0 -1
-1 1 0
```

f. Find inverse of A matrix

```
>> inv(A)
```

```
ans =
```

```
0.1029 0.0470 -0.1275
0.0470 -0.0872 0.0940
-0.1163 0.1208 0.1007
```

g. Perform $A*A'$ and $A.*A'$

```
>> A*B
```

```
ans =
```

```
171 102 105
102 102 65
105 65 93
```

```
>> A.*B
```

```
ans =
```

```
81 63 6
63 4 56
6 56 25
```

h. Find square of A matrix

```
>> F = A*A
```

```
F =
```

```
150 123 105
91 123 70
84 74 87
```

i. Find rank of A matrix

```
>> rank(A)
```

```
ans =
```

```
3
```

j. Find eigenvalues and eigenvectors of A matrix

```
>> eig(A)
```

```
ans =
```

```
17.5118
```

```
4.3526
```

```
-5.8644
```

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

```
V =
```

```
-0.7183 -0.6835 0.3698
```

```
-0.5284 0.1125 -0.7794
```

```
-0.4527 0.7213 0.5058
```

```
D =
```

```
17.5118 0 0
```

```
0 4.3526 0
```

```
0 0 -5.8644
```

3. Create a vector and a matrix with the following commands: `v=0:0.2:12` and `M=[sin(v); cos(v)]`. Find the sizes of `v` and `M` and extract the first 10 elements of each row of the matrix and display them as column vectors.

```
>> v=0:0.2:12
```

```
v =
```

```
Columns 1 through 12
```

```
    0    0.2000    0.4000    0.6000    0.8000    1.0000    1.2000    1.4000    1.6000    1.8000
 2.0000    2.2000
```

```
Columns 13 through 24
```

```
    2.4000    2.6000    2.8000    3.0000    3.2000    3.4000    3.6000    3.8000    4.0000
 4.2000    4.4000    4.6000
```

```
Columns 25 through 36
```

```
    4.8000    5.0000    5.2000    5.4000    5.6000    5.8000    6.0000    6.2000    6.4000
 6.6000    6.8000    7.0000
```

```
Columns 37 through 48
```

```
    7.2000    7.4000    7.6000    7.8000    8.0000    8.2000    8.4000    8.6000    8.8000
 9.0000    9.2000    9.4000
```

```
Columns 49 through 60
```

```
    9.6000    9.8000   10.0000   10.2000   10.4000   10.6000   10.8000   11.0000   11.2000
 11.4000   11.6000   11.8000
```

```
Column 61
```

```
 12.0000
```

```
>> M=[sin(v); cos(v)]
```

```
M =
```

```
Columns 1 through 12
```

```
    0    0.1987    0.3894    0.5646    0.7174    0.8415    0.9320    0.9854    0.9996    0.9738
 0.9093    0.8085
    1.0000    0.9801    0.9211    0.8253    0.6967    0.5403    0.3624    0.1700   -0.0292   -
 0.2272   -0.4161   -0.5885
```

```
Columns 13 through 24
```

```
    0.6755  0.5155  0.3350  0.1411 -0.0584 -0.2555 -0.4425 -0.6119 -0.7568 -  
0.8716 -0.9516 -0.9937  
    -0.7374 -0.8569 -0.9422 -0.9900 -0.9983 -0.9668 -0.8968 -0.7910 -0.6536 -  
0.4903 -0.3073 -0.1122
```

Columns 25 through 36

```
    -0.9962 -0.9589 -0.8835 -0.7728 -0.6313 -0.4646 -0.2794 -0.0831  0.1165  
0.3115  0.4941  0.6570  
    0.0875  0.2837  0.4685  0.6347  0.7756  0.8855  0.9602  0.9965  0.9932  
0.9502  0.8694  0.7539
```

Columns 37 through 48

```
    0.7937  0.8987  0.9679  0.9985  0.9894  0.9407  0.8546  0.7344  0.5849  
0.4121  0.2229  0.0248  
    0.6084  0.4385  0.2513  0.0540 -0.1455 -0.3392 -0.5193 -0.6787 -0.8111 -  
0.9111 -0.9748 -0.9997
```

Columns 49 through 60

```
    -0.1743 -0.3665 -0.5440 -0.6999 -0.8278 -0.9228 -0.9809 -1.0000 -0.9792 -  
0.9193 -0.8228 -0.6935  
    -0.9847 -0.9304 -0.8391 -0.7143 -0.5610 -0.3853 -0.1943  0.0044  0.2030  
0.3935  0.5683  0.7204
```

Column 61

```
-0.5366  
0.8439
```

```
>> size(v)
```

```
ans =
```

```
1  61
```

```
>> size(M)
```

```
ans =
```

```
2  61
```

```
>> M(:,1:10)
```

```
ans =
```

```
    0  0.1987  0.3894  0.5646  0.7174  0.8415  0.9320  0.9854  0.9996  0.9738  
1.0000  0.9801  0.9211  0.8253  0.6967  0.5403  0.3624  0.1700 -0.0292 -  
0.2272
```


4. The polar equation of a circle is given by $x=r \cos\theta$, $y=r \sin\theta$. Take $\theta= 0$ to 2π with step size of $\pi/16$ and plot the circle on x-y axis for given value of radius r . Give labels to axis and title to the figure. Make use of new figure and redraw the circle with distinct points shown by 'o' rather than a continuous plot. Now combine the two plots in new figure to show the line through the data points as well as the distinct data points.

```
>> theta = 0:pi/16:2*pi

theta =

Columns 1 through 12

    0    0.1963    0.3927    0.5890    0.7854    0.9817    1.1781    1.3744    1.5708
1.7671    1.9635    2.1598

Columns 13 through 24

    2.3562    2.5525    2.7489    2.9452    3.1416    3.3379    3.5343    3.7306    3.9270
4.1233    4.3197    4.5160

Columns 25 through 33

    4.7124    4.9087    5.1051    5.3014    5.4978    5.6941    5.8905    6.0868    6.2832

>> x = r*cos(theta)

x =

Columns 1 through 12

    2.0000    1.9616    1.8478    1.6629    1.4142    1.1111    0.7654    0.3902    0.0000    -
0.3902   -0.7654   -1.1111

Columns 13 through 24

   -1.4142   -1.6629   -1.8478   -1.9616   -2.0000   -1.9616   -1.8478   -1.6629   -1.4142   -
1.1111   -0.7654   -0.3902

Columns 25 through 33

   -0.0000    0.3902    0.7654    1.1111    1.4142    1.6629    1.8478    1.9616    2.0000

>> y = r*sin(theta)

y =

Columns 1 through 12

    0    0.3902    0.7654    1.1111    1.4142    1.6629    1.8478    1.9616    2.0000
1.9616    1.8478    1.6629
```

Columns 13 through 24

```
1.4142  1.1111  0.7654  0.3902  0.0000 -0.3902 -0.7654 -1.1111 -1.4142 -  
1.6629 -1.8478 -1.9616
```

Columns 25 through 33

```
-2.0000 -1.9616 -1.8478 -1.6629 -1.4142 -1.1111 -0.7654 -0.3902 -0.0000
```

```
>> r = 10
```

```
r =
```

```
10
```

```
>> x = r*cos(theta)
```

```
x =
```

Columns 1 through 12

```
10.0000  9.8079  9.2388  8.3147  7.0711  5.5557  3.8268  1.9509  0.0000 -  
1.9509 -3.8268 -5.5557
```

Columns 13 through 24

```
-7.0711 -8.3147 -9.2388 -9.8079 -10.0000 -9.8079 -9.2388 -8.3147 -7.0711 -  
5.5557 -3.8268 -1.9509
```

Columns 25 through 33

```
-0.0000  1.9509  3.8268  5.5557  7.0711  8.3147  9.2388  9.8079 10.0000
```

```
>> y = r*sin(theta)
```

```
y =
```

Columns 1 through 12

```
0  1.9509  3.8268  5.5557  7.0711  8.3147  9.2388  9.8079 10.0000  
9.8079  9.2388  8.3147
```

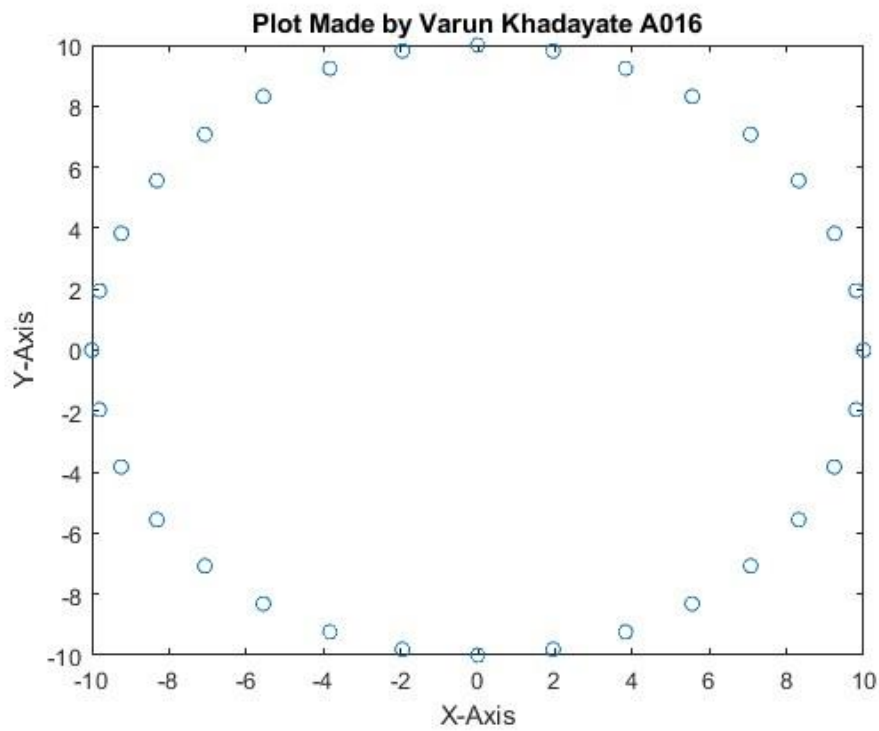
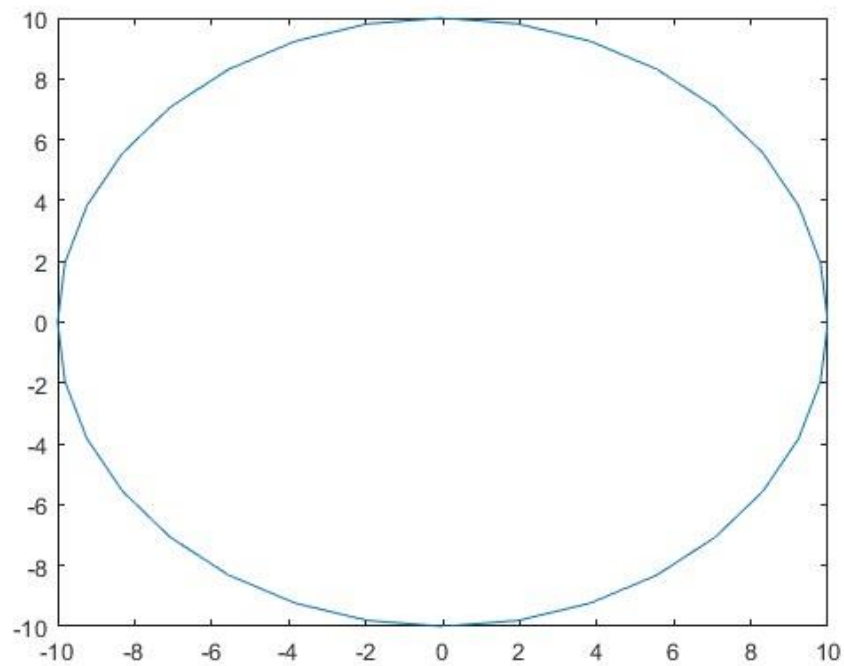
Columns 13 through 24

```
7.0711  5.5557  3.8268  1.9509  0.0000 -1.9509 -3.8268 -5.5557 -7.0711 -  
8.3147 -9.2388 -9.8079
```

Columns 25 through 33

```
-10.0000 -9.8079 -9.2388 -8.3147 -7.0711 -5.5557 -3.8268 -1.9509 -0.0000
```

```
>> plot(x,y)
>> plot(x,y)
>> plot(x,y,'O')
>> plot(x,y,'O')
>> xlabel('X-Axis')
>> ylabel('Y-Axis')
>> title('Plot Made by Varun Khadayate A016')
```



| | |
|--------------------------------|-----------------------|
| Roll. No. A016 | Name: Varun Khadayate |
| Class B.Tech CSBs | Batch: 1 |
| Date of Experiment: 29-07-2021 | Subject: IT/WS |

1. Create a variable myage and store your age in it. Subtract 2 from the value of the variable. Add 1 to the value of the variable. Observe the Workspace Window and Command History Window as you do this.

Command Window

```
>> myage = 21;
```

Workspace



Command Window

```
>> myage = myage - 2;
```

Workspace



Command Window

```
>> myage = myage + 1;
```

Workspace



2. Explain the difference between these two statements:

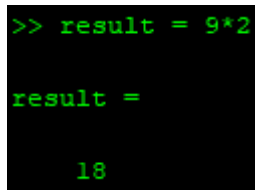
```
result = 9*2
```

```
result = 9*2;
```

Both will store 18 in the variable result. In the first, MATLAB will display this in the Command Window; in the second, it will not.

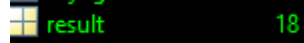
```
result = 9*2
```

Result shown in Command Window and in Worspace



```
result = 9*2;
```

Result shown in Workspace and not in Command Window

A screenshot of the MATLAB Workspace window. It shows a single variable named 'result' with a value of 18. The variable is represented by a small icon with a plus sign and the text 'result' and '18'.

3. Use the built-in function `namelengthmax` to find out the maximum number of characters that you can have in an identifier name under your version of MATLAB.

```
>> namelengthmax
```

```
ans =
```

```
63
```

4. Create two variables to store a weight in pounds and ounces. Use `who` and `whos` to see the variables. Use `class` to see the types of the variables. Clear one of them and then use `who` and `whos` again.

```
>> pounds = 4;  
>> ounces = 3.3;  
>> who
```

Your variables are:

```
ans  myage  ounces  pounds  result
```

```
>> whos
```

| Name | Size | Bytes | Class | Attributes |
|--------|------|-------|--------|------------|
| ans | 1x1 | 8 | double | |
| myage | 1x1 | 8 | double | |
| ounces | 1x1 | 8 | double | |
| pounds | 1x1 | 8 | double | |
| result | 1x1 | 8 | double | |

```
>> clear pounds  
>> who
```

Your variables are:

```
ans  myage  ounces  result
```

```
>> whos
```

| Name | Size | Bytes | Class | Attributes |
|--------|------|-------|--------|------------|
| ans | 1x1 | 8 | double | |
| myage | 1x1 | 8 | double | |
| ounces | 1x1 | 8 | double | |
| result | 1x1 | 8 | double | |

5. Explore the format command in more detail. Use help format to find options. Experiment with format bank to display dollar values.

```
>> format +
>> 1234

ans =

+

>> -1234

ans =

-

>> format bank
>> 33.4

ans =

    33.40

>> 69.435

ans =

    69.44
```

6. Find a format option that would result in the following output format:

```
>> 5/16 + 2/7
ans =
    67/112
```

```
>> format rat
>> 5/16 + 2/7

ans =

    67/112
```

7. Think about what the results would be for the following expressions, and then type them in to verify your answers.

```
25 / 5 * 5
4 + 3 ^ 2
(4 + 3) ^ 2
3 \ 12 + 5
4 - 2 * 3
```

```
>> 25/5*5
```

```
ans =
```

```
25
```

```
>> 4+3^2
```

```
ans =
```

```
13
```

```
>> (4+3)^2
```

```
ans =
```

```
49
```

```
>> 3/12+5
```

```
ans =
```

```
21/4
```

```
>> 4-2*3
```

```
ans =
```

```
-2
```

8. Create a variable pounds to store a weight in pounds. Convert this to kilograms and assign the result to a variable kilos. The conversion factor is

1 kilogram = 2.2 lb.

```
>> pounds = 69
```

```
pounds =
```

```
69
```

```
>> kilos = pounds / 2.2
```

```
kilos =
```

```
31.3636
```

9. Create a variable ftemp to store a temperature in degrees Fahrenheit (F). Convert this to degrees Celsius (C) and store the result in a variable ctemp. The conversion factor is $C = (F - 32) * 5/9$.

```
>> f = 75

f =

    75

>> c = (f - 32) * 5/9

c =

    23.8889
```

10. The following assignment statements either contain at least one error, or could be improved in some way. Assume that radius is a variable that has been initialized. First, identify the problem, and then fix and/or improve them:
- ```
33 = number
my variable = 11.11;
area = 3.14 * radius^2;
x = 2 * 3.14 * radius;
```

`33 = number`

The variable is always on the left ***number = 33***

`my variable = 11.11;`

Spaces are not allowed in variable names ***my\_variable = 11.11;***

`area = 3.14 * radius^2;`

Using pi is more accurate than 3.14 ***area = pi \* radius^2;***

`x = 2 * 3.14 * radius;`

x is not a descriptive variable name and Using pi is more accurate than 3.14

***circumference = 2 \* pi \* radius;***

11. Experiment with the functional form of some operators such as plus, minus, and times.

```
>> plus(6,9)

ans =

 15

>> plus(6,-9)

ans =
```



```
-3
>> minus(1,7)
ans =
-6
>> minus(4,-6)
ans =
10
>> times(4,6)
ans =
24
>> times(6,-7)
ans =
-42
```

12. Generate a random

a. real number in the range (0, 20)

```
>> rand * 20
ans =
16.2945
```

b. real number in the range (20, 50)

```
>> rand*(50-20)+20
ans =
47.1738
```

c. integer in the inclusive range from 1 to 10

```
>> randi(10)

ans =

 2
```

d. integer in the inclusive range from 0 to 10

```
>> randi([0, 10])

ans =

 10
```

e. integer in the inclusive range from 50 to 100

```
>> randi([50, 100])

ans =

 82
```

13. Get into a new Command Window, and type `rand` to get a random real number. Make a note of the number. Then, exit MATLAB and repeat this, again making a note of the random number; it should be the same as before. Finally, exit MATLAB and again get into a new Command Window. This time, change the seed before generating a random number; it should be different.

```
>> rand

ans =

 0.0975

>> rng('shuffle')
>> rand

ans =

 0.3191
```

14. What is the difference between `x` and `'x'`?

In an expression, the first would be interpreted as the name of a variable, whereas `'x'` is the character `x`.

15. Explain the difference between constants and variables.

Constants store values that are known and do not change. Variables are used when the value will change, or when the value is not known to begin with (e.g., the user will provide the value).

16. What would be the result of the following expressions?

'b' >= 'c' - 1  
3 == 2 + 1  
(3 == 2) + 1  
xor(5 < 6, 8 > 4)  
10 > 5 > 2  
0 <= result <= 10

```
>> 'b' >= 'c' - 1
'b' >= 'c' - 1
 ↑
Error: Invalid text character. Check for unsupported symbol, invisible character, or pasting of non-ASCII characters.

>> 3 == 2 + 1

ans =

logical

1

>> xor(5 < 6, 8 > 4)

ans =

logical

0

>> 10 > 5 > 2

ans =

logical

0

>> result = 3^2 - 20;
>> result = 3^2 - 20

result =
```

```
-11
```

```
>> 0 <= result <= 10
```

```
ans =
```

```
logical
```

```
1
```

17. Create two variables x and y and store numbers in them. Write an expression that would be true if the value of x is greater than five or if the value of y is less than ten, but not if both of those are true.

```
>> x = 6
```

```
x =
```

```
6
```

```
>> y = 9
```

```
y =
```

```
9
```

```
>> xor(x > 5, y < 10)
```

```
ans =
```

```
logical
```

```
0
```

18. Use the equality operator to verify that  $3 \cdot 10^5$  is equal to  $3e5$ .

```
>> 3*10^5 == 3e5
```

```
ans =
```

```
logical
```

```
1
```

19. In the ASCII character encoding, the letters of the alphabet are in order: 'a' comes before 'b' and also 'A' comes before 'B'. However, which comes first - lower or uppercase letters?

```
>> int32('a')
```

```
ans =
```

```
int32
```

```
97
```

```
>> int32('A')
```

```
ans =
```

```
int32
```

```
65
```

20. Are there equivalents to `intmin` and `intmax` for real number types? Use help to find out.

```
>> help intmax
```

`intmax` Largest positive integer value.

`X = intmax` is the largest positive value representable in an `int32`.

Any value that is larger than `intmax` will saturate to `intmax` when cast to `int32`.

`intmax('int32')` is the same as `intmax` with no arguments.

`intmax(CLASSNAME)` is the largest positive value in the integer class `CLASSNAME`. Valid values of `CLASSNAME` are `'int8'`, `'uint8'`, `'int16'`, `'uint16'`, `'int32'`, `'uint32'`, `'int64'` and `'uint64'`.

`intmax('like', Y)` returns the largest positive value in the integer class with the same data type and complexity (real or complex) as the numeric variable `Y`.

See also `intmin`, `realmax`.

Documentation for `intmax`

```
>> help intmin
```

`intmin` Smallest integer value.

`X = intmin` is the smallest value representable in an `int32`.

Any value that is smaller than `intmin` will saturate to `intmin` when cast to `int32`.

`intmin('int32')` is the same as `intmin` with no arguments.

`intmin(CLASSNAME)` is the smallest value in the integer class `CLASSNAME`. Valid values of `CLASSNAME` are `'int8'`, `'uint8'`, `'int16'`, `'uint16'`,

'int32', 'uint32', 'int64' and 'uint64'.

intmin('like', Y) returns the smallest value in the integer class with the same data type and complexity (real or complex) as the numeric variable Y.

See also intmax, realmin.

Documentation for intmin

```
>> realmin
```

```
ans =
```

```
2.2251e-308
```

```
>> realmin('double')
```

```
ans =
```

```
2.2251e-308
```

```
>> realmin('single')
```

```
ans =
```

```
single
```

```
1.1755e-38
```

```
>> realmax
```

```
ans =
```

```
1.7977e+308
```

21. Use intmin and intmax to determine the range of values that can be stored in the types uint32 and uint64.

```
>> intmin('uint32')
```

```
ans =
```

```
uint32
```

```
0
```

```
>> intmax('uint32')
```

```
ans =
```

```
uint32
4294967295
>> intmin('uint64')
ans =
uint64
0
>> intmax('uint64')
ans =
uint64
18446744073709551615
```

22. Use the cast function to cast a variable to be the same type as another variable.

```
>> vara = uint16(6 + 9)
vara =
uint16
15
>> varb = 4*7
varb =
28
>> class(varb)
ans =
'double'
>> varb = cast(varb, 'like', vara)
varb =
uint16
28
>> class(varb)
```

```
ans =
```

```
'uint16'
```

23. Use help elfun or experiment to answer the following questions:

a. Is `fix(3.5)` the same as `floor(3.5)`?

```
>> fix(3.5)
```

```
ans =
```

```
3
```

```
>> floor(3.5)
```

```
ans =
```

```
3
```

b. Is `fix(3.4)` the same as `fix(-3.4)`?

```
>> fix(3.4)
```

```
ans =
```

```
3
```

```
>> fix(-3.4)
```

```
ans =
```

```
-3
```

c. Is `fix(3.2)` the same as `floor(3.2)`?

```
>> fix(3.2)
```

```
ans =
```

```
3
```

```
>> floor(3.2)
```

```
ans =
```

```
3
```



d. Is `fix(-3.2)` the same as `floor(-3.2)`?

```
>> fix(-3.2)

ans =

 -3

>> floor(-3.2)

ans =

 -4
```

e. Is `fix(-3.2)` the same as `ceil(-3.2)`?

```
>> fix(-3.2)

ans =

 -3

>> ceil(-3.2)

ans =

 -3
```

24.

a. For what range of values is the function `round` equivalent to the function `floor`?

**For positive numbers:** when the decimal part is less than .5  
**For negative numbers:** when the decimal part is greater than or equal to .5

b. For what range of values is the function `round` equivalent to the function `ceil`?

**For positive numbers:** when the decimal part is greater than or equal to .5  
**For negative numbers:** when the decimal part is less than .5

25. Use `help` to determine the difference between the `rem` and `mod` functions.

```
>> help rem
rem Remainder after division.
 rem(x,y) returns x - fix(x./y).*y if y ~= 0, carefully computed to
 avoid rounding error. If y is not an integer and the quotient x./y is
 within roundoff error of an integer, then n is that integer. The inputs
 x and y must be real and have compatible sizes. In the simplest cases,
 they can be the same size or one can be a scalar. Two inputs have
 compatible sizes if, for every dimension, the dimension sizes of the
```

inputs are either the same or one of them is 1.

By convention:

`rem(x,0)` is NaN.

`rem(x,x)`, for  $x \neq 0$ , is 0.

`rem(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $x$ .

Note: `MOD(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $y$ .

`rem(x,y)` and `MOD(x,y)` are equal if  $x$  and  $y$  have the same sign, but differ by  $y$  if  $x$  and  $y$  have different signs.

See also `mod`.

Documentation for `rem`

Other functions named `rem`

`>> help mod`

`mod` Modulus after division.

`mod(x,y)` returns  $x - \text{floor}(x./y) \cdot y$  if  $y \neq 0$ , carefully computed to avoid rounding error. If  $y$  is not an integer and the quotient  $x./y$  is within roundoff error of an integer, then  $n$  is that integer. The inputs  $x$  and  $y$  must be real and have compatible sizes. In the simplest cases, they can be the same size or one can be a scalar. Two inputs have compatible sizes if, for every dimension, the dimension sizes of the inputs are either the same or one of them is 1.

The statement " $x$  and  $y$  are congruent mod  $m$ " means `mod(x,m) == mod(y,m)`.

By convention:

`mod(x,0)` is  $x$ .

`mod(x,x)` is 0.

`mod(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $y$ .

Note: `REM(x,y)`, for  $x \neq y$  and  $y \neq 0$ , has the same sign as  $x$ .

`mod(x,y)` and `REM(x,y)` are equal if  $x$  and  $y$  have the same sign, but differ by  $y$  if  $x$  and  $y$  have different signs.

See also `rem`.

Documentation for `mod`

Other functions named `mod`

26. Find MATLAB expressions for the following

$$\sqrt{19}$$

`>> sqrt(19)`

`ans =`

```
4.3589
```

$3^{1.2}$

```
>> 3^1.2
```

```
ans =
```

```
3.7372
```

$\tan(\pi)$

```
>> tan(pi)
```

```
ans =
```

```
-1.2246e-16
```

27. Using only the integers 2 and 3, write as many expressions as you can that result in 9. Try to come up with at least 10 different expressions (Note: don't just change the order). Be creative! Make sure that you write them as MATLAB expressions. Use operators and/or built-in functions.

```
>> 3^2
```

```
ans =
```

```
9
```

```
>> 2^3+(3-2)
```

```
ans =
```

```
9
```

```
>> 3*3
```

```
ans =
```

```
9
```

```
>> 3^3-3*3*2
```

```
ans =
```

```
9
```

```
>> 2^3+abs(2-3)
```

```
ans =
```

```
9
>> 2^3+sign(3)
ans =
9
>> 3/2*2*3
ans =
9
>> 2\3*2*3
ans =
9
>> sqrt(3^(2+2))
ans =
9
```

28. A vector can be represented by its rectangular coordinates  $x$  and  $y$  or by its polar coordinates  $r$  and  $\theta$ .  $\theta$  is measured in radians. The relationship between them is given by the equations:

$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

```
>> r = 46
r =
46
>> theta = 0.7
theta =
0.7000
>> x = r*cos(theta)
x =
```

```
35.1827
```

```
>> y = r*sin(theta)
```

```
y =
```

```
29.6340
```

29. In special relativity, the Lorentz factor is a number that describes the effect of speed on various physical properties when the speed is significant relative to the speed of light. Mathematically, the Lorentz factor is given as:

$$\gamma = \frac{1}{\sqrt{1 - \frac{v^2}{c^2}}}$$

Use  $3 \times 10^8$  m/s for the speed of light,  $c$ . Create variables for  $c$  and the speed  $v$  and from them a variable *lorentz* for the Lorentz factor.

```
>> c = 3e8
```

```
c =
```

```
300000000
```

```
>> v = 2.9e8
```

```
v =
```

```
290000000
```

```
>> lore = 1 / sqrt(1 - v^2/c^2)
```

```
lore =
```

```
3.9057
```

30. A company manufactures a part for which there is a desired weight. There is a tolerance of  $N$  percent, meaning that the range between minus and plus  $N\%$  of the desired weight is acceptable. Create a variable that stores a weight, and another variable for  $N$  (for example, set it to two). Create variables that store the minimum and maximum values in the acceptable range of weights for this part.

```
>> weight = 46.6917
```

```
weight =
```

```
46.6917
```

```
>> N = 7

N =

 7

>> min = weight - weight*0.01*N

min =

 43.4233

>> max = weight + weight*0.01*N

max =

 49.9601
```

31. An environmental engineer has determined that the cost  $C$  of a containment tank will be based on the radius  $r$  of the tank:

$$C = \frac{32430 + 428\pi}{r}$$

Create a variable for the radius, and then for the cost

```
>> radius = 46;
>> c = 32430/radius + 428*pi*radius

c =

 62556.68
```

32. A chemical plant releases an amount  $A$  of pollutant into a stream. The maximum concentration  $C$  of the pollutant at a point which is a distance  $x$  from the plant is:

$$C = \frac{A}{x} \sqrt{\frac{2}{\pi e}}$$

Create variables for the values of  $A$  and  $x$ , and then for  $C$ . Assume that the distance  $x$  is in meters. Experiment with different values for  $x$ .

```
>> A = 80000

A =

 80000.00
```

```

>> x = 100

x =

 100.00

>> C = A/x * sqrt(2/(pi*exp(1)))

C =

 387.15

>> x = 1000;

>> C = A/x * sqrt(2/(pi*exp(1)))

C =

 38.72

>> x = 20000;
>> C = A/x * sqrt(2/(pi*exp(1)))

C =

 1.94

```

33. The geometric mean  $g$  of  $n$  numbers  $x_i$  is defined as the  $n^{\text{th}}$  root of the product of  $x_i$ :

$$g = \sqrt[n]{x_1 x_2 x_3 \dots x_n}$$

(This is useful, for example, in finding the average rate of return for an investment which is something you'd do in engineering economics). If an investment returns 15% the first year, 50% the second, and 30% the third year, the average rate of return would be  $(1.15 * 1.50 * 1.30)^{1/3}$ . ) Compute this.

```

>> x1 = 1.15

x1 =

 1.15

>> x2 = 1.5

x2 =

 1.50

```

```
>> x3 = 1.3
```

```
x3 =
```

```
1.30
```

```
>> g = nthroot(x1*x2*x3, 3)
```

```
g =
```

```
1.31
```

34. Use the **deg2rad** function to convert 180 degrees to radians.

```
>> deg2rad(180)
```

```
ans =
```

```
3.14
```



|                                |                       |
|--------------------------------|-----------------------|
| Roll. No. A016                 | Name: Varun Khadayate |
| Class B.Tech CsBs              | Batch: 1              |
| Date of Experiment: 29-07-2021 | Subject: IT/WS        |

1. If a variable has the dimensions 3 x 4, could it be considered to be (Check all that apply):

- a. a matrix
- b. a row vector
- c. a column vector
- d. a scalar

2. If a variable has the dimensions 1 x 5, could it be considered to be (Check all that apply):

- a. a matrix
- b. a row vector
- c. a column vector
- d. a scalar

3. If a variable has the dimensions 5 x 1, could it be considered to be (Check all that apply):

- a. a matrix
- b. a row vector
- c. a column vector
- d. a scalar

4. If a variable has the dimensions 1 x 1, could it be considered to be (Check all that apply):

- a. a matrix
- b. a row vector
- c. a column vector
- d. a scalar

5. Using the colon operator, create the following row vectors

|        |   |        |   |        |        |
|--------|---|--------|---|--------|--------|
| 2      | 3 | 4      | 5 | 6      | 7      |
| 1.1000 |   | 1.3000 |   | 1.5000 | 1.7000 |
| 8      | 6 | 4      | 2 |        |        |

```
>> 2:7
```

```
ans =
```

```
2.00 3.00 4.00 5.00 6.00 7.00
```

```
>> 1.1000:0.2000:1.7000
```

```
ans =
```

```
1.1000 1.3000 1.5000 1.7000
```

```
>> 8:-2:2
```

```
ans =
```

```
8 6 4 2
```

6. Using a built-in function, create a vector `vec` which consists of 20 equally spaced points in the range from  $-\pi$  to  $+\pi$ .

```
>> vec = linspace(0,2*pi,20)

vec =

Columns 1 through 12

 0 0.3307 0.6614 0.9921 1.3228 1.6535 1.9842 2.3149 2.6456 2.9762 3.3069
3.6376

Columns 13 through 20

 3.9683 4.2990 4.6297 4.9604 5.2911 5.6218 5.9525 6.2832
```

7. Write an expression using `linspace` that will result in the same as `2: 0.2: 3`

```
>> linspace(2,3,6)

ans =

 2.0000 2.2000 2.4000 2.6000 2.8000 3.0000
```

8. Using the colon operator and also the `linspace` function, create the following row vectors:

|    |    |    |    |    |
|----|----|----|----|----|
| -5 | -4 | -3 | -2 | -1 |
| 5  | 7  | 9  |    |    |
| 8  | 6  | 4  |    |    |

```
>> -5:-1

ans =

 -5 -4 -3 -2 -1

>> linspace(-5,-1,5)

ans =

 -5 -4 -3 -2 -1

>> 5:2:9

ans =

 5 7 9

>> linspace(5,9,3)

ans =
```

```

5 7 9

>> 8:-2:4

ans =

 8 6 4

>> linspace(8,4,3)

ans =

 8 6 4

```

9. How many elements would be in the vectors created by the following expressions?

a. `linspace(3,2000)`

100 (always, by default)

b. `logspace(3,2000)`

50 (always, by default – although these numbers would get very large quickly; most would be represented as Inf)

10. Create a variable `myend` which stores a random integer in the inclusive range from 5 to 9.

Using the colon operator, create a vector that iterates from 1 to `myend` in steps of 3.

```

>> myend = randi([5, 9])

myend =

 9

>> vec = 1:3:myend

vec =

 1 4 7

```

11. Using the colon operator and the transpose operator, create a column vector `myvec` that has the values -1 to 1 in steps of 0.5.

```

>> rowVec = -1: 0.5: 1

rowVec =

-1.0000 -0.5000 0 0.5000 1.0000

>> rowVec'

ans =

 -1 -0.5 0 0.5 1

```

```
-1.0000
-0.5000
0
0.5000
1.0000
```

12. Write an expression that refers to only the elements that have odd- numbered subscripts in a vector, regardless of the length of the vector. Test your expression on vectors that have both an odd and even number of elements.

```
>> vec = 1:8

vec =

1 2 3 4 5 6 7 8

>> vec(1:2:end)

ans =

1 3 5 7

>> vec = 4:12

vec =

4 5 6 7 8 9 10 11 12

>> vec(1:2:end)

ans =

4 6 8 10 12
```

13. Generate a 2 x 4 matrix variable mat. Replace the first row with 1:4. Replace the third column (you decide with which values).

```
>> mat = [2:5; 1 4 11 3]

mat =

2 3 4 5
1 4 11 3

>> mat(1,:) = 1:4

mat =

1 2 3 4
1 4 11 3

>> mat(:,3) = [4;3]
```

```
mat =
```

```
1 2 4 4
1 4 3 3
```

14. Generate a 2 x 4 matrix variable *mat*. Verify that the number of elements is the product of the number of rows and columns

```
>> mat = randi(20,2,4)
```

```
mat =
```

```
6 5 14 4
15 1 9 6
```

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

```
r =
```

```
2
```

```
c =
```

```
4
```

```
>> numel(mat) == r * c
```

```
ans =
```

```
logical
```

```
1
```

15. Which would you normally use for a matrix: length or size? Why?

Size, because it tells you both the number of rows and columns.

16. When would you use length vs. size for a vector?

If you want to know the number of elements, you'd use length. If you want to figure out whether it's a row or column vector, you'd use size.

17. Generate a 2 x 3 matrix of random

a. real numbers, each in the range (0, 1)

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

```
ans =
```

```
0.0334 0.0782 0.7775
0.0746 0.4518 0.1108
```

b. real numbers, each in the range (0, 10)

```
>> rand(2,3)*10
```

```
ans =
```

```
0.0534 6.6795 6.0724
9.5356 6.9165 9.2982
```

c. integers, each in the inclusive range from 5 to 20

```
>> randi([5, 20],2,3)
```

```
ans =
```

```
17 16 13
14 11 14
```

18. Create a variable rows that is a random integer in the inclusive range from 1 to 5. Create a variable cols that is a random integer in the inclusive range from 1 to 5. Create a matrix of all zeros with the dimensions given by the values of rows and cols.

```
>> rows = randi([1,5])
```

```
rows =
```

```
3
```

```
>> cols = randi([1,5])
```

```
cols =
```

```
2
```

```
>> zeros(rows,cols)
```

```
ans =
```

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

19. Create a matrix variable mat. Find as many expressions as you can that would refer to the last element in the matrix, without assuming that you know how many elements or rows or columns it has (i.e., make your expressions general).

```
>> mat = [12:15; 6:-1:3]
```

```
mat =
```

```
12 13 14 15
6 5 4 3
```

```
>> mat(end,end)
```

```
ans =

 3

>> mat(end)

ans =

 3

>> [r c] = size(mat)

r =

 2

c =

 4

>> mat(r,c)

ans =

 3
```

20. Create a vector variable `vec`. Find as many expressions as you can that would refer to the last element in the vector, without assuming that you know how many elements it has (i.e., make your expressions general).

```
>> vec = 1:2:9

vec =

 1 3 5 7 9

>> vec(end)

ans =

 9

>> vec(numel(vec))

ans =

 9

>> vec(length(vec))
```

```
ans =

 9

>> v = fliplr(vec)

v =

 9 7 5 3 1

>> v(1)

ans =

 9
```

21. Create a 2 x 3 matrix variable `mat`. Pass this matrix variable to each of the following functions and make sure you understand the result: `flip`, `fliplr`, `flipud`, and `rot90`. In how many different ways can you reshape it?

```
>> mat = randi([1,20], 2,3)

mat =

 19 20 3
 8 19 10

>> flip(mat)

ans =

 8 19 10
 19 20 3

>> fliplr(mat)

ans =

 3 20 19
 10 19 8

>> flipud(mat)

ans =

 8 19 10
 19 20 3

>> rot90(mat)

ans =
```



```

 3 10
 20 19
 19 8

>> rot90(rot90(mat))

ans =

 10 19 8
 3 20 19

>> reshape(mat,3,2)

ans =

 19 19
 8 3
 20 10

>> reshape(mat,1,6)

ans =

 19 8 20 19 3 10

>> reshape(mat,6,1)

ans =

 19
 8
 20
 19
 3
 10

```

22. What is the difference between `fliplr(mat)` and `mat = fliplr(mat)`?

`fliplr(mat)` stores the result in `ans` so `mat` is not changed.  
`mat = fliplr(mat)` changes `mat`.

23. Use `reshape` to reshape the row vector `1:4` into a 2x2 matrix; store this in a variable named `mat`. Next, make 2x3 copies of `mat` using both `repelem` and `repmat`.

```

>> mat = reshape(1:4,2,2)

mat =

 1 3
 2 4

>> repelem(mat,2,3)

```

```
ans =
```

```
1 1 1 3 3 3
1 1 1 3 3 3
2 2 2 4 4 4
2 2 2 4 4 4
```

```
>> repmat(mat,2,3)
```

```
ans =
```

```
1 3 1 3 1 3
2 4 2 4 2 4
1 3 1 3 1 3
2 4 2 4 2 4
```

|                                |                       |
|--------------------------------|-----------------------|
| Roll. No. A016                 | Name: Varun Khadayate |
| Class B.Tech CsBs              | Batch: 1              |
| Date of Experiment: 29-07-2021 | Subject: IT/WS        |

1. Create a 3 x 5 matrix of random real numbers. Delete the third row.

```
>> mat = rand(3,5)

mat =

 0.3238 0.7131 0.2204 0.7464 0.6632
 0.2118 0.2151 0.2079 0.4527 0.3126
 0.4927 0.4089 0.6518 0.8642 0.8317

>> mat(3,:) = []

mat =

 0.3238 0.7131 0.2204 0.7464 0.6632
 0.2118 0.2151 0.2079 0.4527 0.3126
```

2. Given the matrix:

```
>> mat = randi([1 20], 3, 5) mat =
6 17 7 13 17
17 5 4 10 12
6 19 6 8 11
```

Why wouldn't this work:

```
mat(2:3, 1:3) = ones(2)
```

Because the left and right sides are not the same dimensions.

3. Create a three-dimensional matrix with dimensions 2 x 4 x 3 in which the first "layer" is all 0s, the second is all 1s and the third is all 5s. Use size to verify the dimensions.

```
>> mat3d = zeros(2,4,3)

mat3d(:,:,1) =

 0 0 0 0
 0 0 0 0

mat3d(:,:,2) =

 0 0 0 0
 0 0 0 0
```

```
mat3d(:,:,3) =
```

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

```
>> mat3d(:,:,2) = 1
```

```
mat3d(:,:,1) =
```

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

```
mat3d(:,:,2) =
```

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

```
mat3d(:,:,3) =
```

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

```
>> mat3d(:,:,3) = 5
```

```
mat3d(:,:,1) =
```

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

```
mat3d(:,:,2) =
```

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

```
mat3d(:,:,3) =
```

```
 5 5 5 5
 5 5 5 5
```

```
>> mat3d
```

```
mat3d(:,:,1) =
```

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

```
mat3d(:,:,2) =
```

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

```
mat3d(:,:,3) =
```

```
5 5 5 5
5 5 5 5
```

4. Create a vector  $x$  which consists of 20 equally spaced points in the range from  $-\pi$  to  $+\pi$ . Create a  $y$  vector which is  $\sin(x)$ .

```
>> x = linspace(-pi,pi,20)
x =
Columns 1 through 12
-3.1416 -2.8109 -2.4802 -2.1495 -1.8188 -1.4881 -
1.1574 -0.8267 -0.4960 -0.1653 0.1653 0.4960
Columns 13 through 20
0.8267 1.1574 1.4881 1.8188 2.1495 2.4802
2.8109 3.1416
>> y = sin(x)
y =
Columns 1 through 12
-0.0000 -0.3247 -0.6142 -0.8372 -0.9694 -0.9966 -
0.9158 -0.7357 -0.4759 -0.1646 0.1646 0.4759
Columns 13 through 20
0.7357 0.9158 0.9966 0.9694 0.8372 0.6142
0.3247 0.0000
```

5. Create a 3 x 5 matrix of random integers, each in the inclusive range from -5 to 5. Get the sign of every element.

```
>> mat = randi([-5,5], 3,5)
mat =
```

|    |    |   |    |    |
|----|----|---|----|----|
| -2 | -2 | 4 | 5  | -2 |
| 5  | -3 | 2 | -2 | 3  |
| -3 | 4  | 4 | 5  | -4 |

```
>> sign(mat)

ans =

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

6. Find the sum 3+5+7+9+11.

```
>> sum(3:2:11)

ans =

 35
```

7. Find the sum of the first n terms of the harmonic series where n is an integer variable greater than one.

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots$$

```
>> n = 4

n =

 4

>> sum(1./(1:n))

ans =

 2.0833
```

8. Find the following sum by first creating vectors for the numerators and denominators:

$$\frac{3}{1} + \frac{5}{2} + \frac{7}{3} + \frac{9}{4}$$

```
>> num = 3:2:9

num =

 3 5 7 9

>> denom = 1:4
```

```
denom =
```

```
1 2 3 4
```

```
>> fracs = num ./ denom
```

```
fracs =
```

```
3.0000 2.5000 2.3333 2.2500
```

```
>> sum(fracs)
```

```
ans =
```

```
10.0833
```

9. Create a matrix and find the product of each row and column using prod.

```
>> mat = randi([1, 30], 2,3)
```

```
mat =
```

```
13 9 9
28 23 2
```

```
>> prod(mat)
```

```
ans =
```

```
364 207 18
```

```
>> prod(mat,2)
```

```
ans =
```

```
1053
```

```
1288
```

10. Write a relational expression for a vector variable that will verify that the last value in a vector created by cumsum is the same as the result returned by sum.

```
>> vec = 2:3:17

vec =

 2 5 8 11 14 17

>> cv = cumsum(vec)

cv =

 2 7 15 26 40 57

>> sum(vec) == cv(end)

ans =

logical

 1
```

11. Create a vector of five random integers, each in the inclusive range from -10 to 10. Perform each of the following:

```
>> vec = randi([-10, 10], 1,5)

vec =

 -3 10 -1 3 2
```

a. subtract 3 from each element

```
>> vec-3

ans =

 -6 7 -4 0 -1
```

b. count how many are positive

```
>> sum(vec > 0)

ans =

 3
```



c. get the cumulative minimum

```
>> cummin(vec)
```

```
ans =
```

```
-3 -3 -3 -3 -3
```

12. Create a 3 x 5 matrix. Perform each of the following:

```
>> mat = randi([-10 10], 3, 5)
```

```
mat =
```

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

a. Find the maximum value in each column.

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

```
ans =
```

```
 1 7 10 10 10
```

b. Find the maximum value in each row.

```
>> max(mat, [], 2)
```

```
ans =
```

```
 10
 10
 9
```

c. Find the maximum value in the entire matrix.

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

```
ans =
```

```
 10
```

d. Find the cumulative maxima.

```
>> cummax(mat)
```

```
ans =
```

```
 1 7 10 10 -7
 1 7 10 10 10
 1 7 10 10 10
```

13. Find two ways to create a 3 x 5 matrix of all 100s (Hint: use ones and zeros).

```
>> ones(3,5)*100

ans =

 100 100 100 100 100
 100 100 100 100 100
 100 100 100 100 100

>> zeros(3,5)+100

ans =

 100 100 100 100 100
 100 100 100 100 100
 100 100 100 100 100
```

14. Given the two matrices:

|   | A  |   | B |   |   |
|---|----|---|---|---|---|
| 1 | 2  | 3 | 2 | 4 | 1 |
| 4 | -1 | 6 | 1 | 3 | 0 |

Perform the following operations:

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

A =

 1 2 3
 4 -1 6

>> B = [2,4,1;1,3,0]

B =

 2 4 1
 1 3 0
```

a.  $A + B$

```
>> A + B

ans =

 3 6 4
 5 2 6
```

b.  $A - B$

```
>> A - B

ans =

 -1 -2 2
 3 -4 6
```

c.  $A * B$

```
>> A .* B
```

```
ans =
```

```
 2 8 3
 4 -3 0
```

15. The built-in function `clock` returns a vector that contains 6 elements: the first three are the current date (year, month, day) and the last three represent the current time in hours, minutes, and seconds. The seconds is a real number, but all others are integers. Store the result from `clock` in a variable called `myc`. Then, store the first three elements from this variable in a variable `today` and the last three elements in a variable `now`. Use the `fix` function on the vector variable `now` to get just the integer part of the current time.

```
>> myc = clock
```

```
myc =
```

```
1.0e+03 *
 2.0220 0.0070 0.0290 0.0160 0.0030 0.0567
```

```
>> today = myc(1:3)
```

```
today =
```

```
 2022 7 29
```

```
>> now = myc(4:end)
```

```
now =
```

```
 16.0000 3.0000 56.7430
```

```
>> fix(now)
```

```
ans =
```

```
 16 3 56
```

16. A vector `v` stores for several employees of the Green Fuel Cells Corporation their hours worked one week followed for each by the hourly pay rate. For example, if the variable stores

```
>> v
```

```
v =
```

```
 33.0000 10.5000 40.0000 18.0000 20.0000 7.5000
```

that means the first employee worked 33 hours at \$10.50 per hour, the second worked 40 hours at \$18 an hour, and so on. Write code that will separate this into two vectors, one that stores the hours worked and another that stores the hourly rates. Then, use the array multiplication operator to create a vector, storing in the new vector the total pay for every employee.

```
>> v = [33.0000 10.5000 40.0000 18.0000 20.0000 7.5000]
```

```
v =
```

```
33.0000 10.5000 40.0000 18.0000 20.0000 7.5000
```

```
>> hours = v(1:2:length(v))
```

```
hours =
```

```
33 40 20
```

```
>> payrate = v(2:2:length(v))
```

```
payrate =
```

```
10.5000 18.0000 7.5000
```

```
>> totpay = hours .* payrate
```

```
totpay =
```

```
346.5000 720.0000 150.0000
```

17. A company is calibrating some measuring instrumentation and has measured the radius and height of one cylinder 10 separate times; they are in vector variables r and h. Find the volume from each trial, which is given by  $\pi r^2 h$ . Also use logical indexing first to make sure that all measurements were valid ( $> 0$ ).

```
>> r = [5.501 5.5 5.499 5.498 5.5 5.5 5.52 5.51 5.5 5.48];
```

```
>> h = [11.11 11.1 11.1 11.12 11.09 11.11 11.11 11.1 11.08 11.11];
```

```
>> r = [5.501 5.5 5.499 5.498 5.5 5.5 5.52 5.51 5.5 5.48]
```

```
r =
```

```
5.5010 5.5000 5.4990 5.4980 5.5000 5.5000
5.5200 5.5100 5.5000 5.4800
```

```
>> h = [11.11 11.1 11.1 11.12 11.09 11.11 11.11 11.1 11.08 11.11]
```

```
h =
```

```
11.1100 11.1000 11.1000 11.1200 11.0900 11.1100
11.1100 11.1000 11.0800 11.1100
```

```
>> vol = pi * r.^2 .* h
```

```
vol =
```

```
1.0e+03 *
1.0562 1.0549 1.0545 1.0560 1.0539 1.0558
1.0635 1.0587 1.0530 1.0482
```

18. For the following matrices A, B, and C:

$$A = \begin{bmatrix} 1 & 4 \\ 3 & 2 \end{bmatrix} \quad B = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 5 & 6 \\ 3 & 6 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 3 & 2 & 5 \\ 4 & 1 & 2 \end{bmatrix}$$

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

```
A =
```

```
 1 4
 3 2
```

```
>> B = [2 1 3;1 5 6;3 6 0]
```

```
B =
```

```
 2 1 3
 1 5 6
 3 6 0
```

```
>> C = [3 2 5;4 1 2]
```

```
C =
```

```
 3 2 5
 4 1 2
```

a. Give the result of  $3 \cdot A$ .

```
>> 3 * A
```

```
ans =
```

```
 3 12
 9 6
```

b. Give the result of  $A \cdot C$ .

```
>> A * C
```

```
ans =
```

```
 19 6 13
 17 8 19
```

c. Are there any other matrix multiplications that can be performed? If so, list them.

```
>> C * B
```

```
ans =
```

```
 23 43 21
 15 21 18
```

19. For the following vectors and matrices, A, B, and C:

$$A = \begin{bmatrix} 4 & 1 & -1 \\ 2 & 3 & 0 \end{bmatrix} \quad B = [1 \ 4] \quad C = \begin{bmatrix} 2 \\ 3 \end{bmatrix}$$

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

```
A =
```

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

```
>> B = [1 4]
```

```
B =
```

```
 1 4
```

```
>> C = [2; 3]
```

```
C =
```

```
 2
 3
```

**a. A \* B**

```
>> A * B
```

```
Error using *
```

```
Incorrect dimensions for matrix multiplication. Check that the
number of columns in the first matrix matches the number
of rows in the second matrix. To operate on each element of the
matrix individually, use TIMES (.*) for elementwise
multiplication.
```

**b. B \* C**

```
>> B * C
```

```
ans =
```

```
 14
```

**c. C \* B**

```
>> C * B
```

```
ans =
```

```
 2 8
 3 12
```

20. The matrix variable rainmat stores the total rainfall in inches for some districts for the years 2010-2013. Each row has the rainfall amounts for a given district. For example, if rainmat has the value:

```
>> rainmat
```

```
ans =
```

```
 25 33 29 42
```

53      44      40      56  
etc.

district 1 had 25 inches in 2010, 33 in 2011, etc. Write expression(s) that will find the number of the district that had the highest total rainfall for the entire four year period.

```
>> rainmat = [25 33 29 42; 53 44 40 56]
```

```
rainmat =
```

```
25 33 29 42
53 44 40 56
```

```
>> large = max(max(rainmat))
```

```
large =
```

```
56
```

```
>> linind = find(rainmat== large)
```

```
linind =
```

```
8
```

```
>> floor(linind/4)
```

```
ans =
```

```
2
```

21. Generate a vector of 20 random integers, each in the range from 50 to 100. Create a variable `evens` that stores all of the even numbers from the vector, and a variable `odds` that stores the odd numbers.

```
>> nums = randi([50, 100], 1, 20)
```

```
nums =
```

```
99 95 92 90 70 87 74 72 93 80 59
55 61 58 71 52 77 65 86 56
```

```
>> evens = nums(rem(nums,2)==0)
```

```
evens =
```

```
92 90 70 74 72 80 58 52 86 56
```

```
>> odds = nums(rem(nums,2)~=0)
```

```
odds =
```

```
99 95 87 93 59 55 61 71 77 65
```

22. Assume that the function `diff` does not exist. Write your own expression(s) to accomplish the same thing for a vector.

```
>> vec = [5 11 2 33 -4]

vec =

 5 11 2 33 -4

>> v1 = vec(2:end)

v1 =

 11 2 33 -4

>> v2 = vec(1:end-1)

v2 =

 5 11 2 33

>> v1-v2

ans =

 6 -9 31 -37
```

23. Create a vector variable `vec`; it can have any length. Then, write assignment statements that would store the first half of the vector in one variable and the second half in another. Make sure that your assignment statements are general, and work whether `vec` has an even or odd number of elements (Hint: use a rounding function such as `fix`).

```
>> vec = 1:9

vec =

 1 2 3 4 5 6 7 8 9

>> fhalf = vec(1:fix(length(vec)/2))

fhalf =

 1 2 3 4

>> shalf = vec(fix(length(vec)/2)+1:end)

shalf =

 5 6 7 8 9
```



## Lab 2

### Question 1

```
A = 30000
```

```
A = 30000
```

```
x = 100
```

```
x = 100
```

```
C = A/x * sqrt(2/(pi*exp(1)))
```

```
C = 145.1824
```

```
x = 1000
```

```
x = 1000
```

```
C = A/x * sqrt(2/(pi*exp(1)))
```

```
C = 14.5182
```

```
x = 2000
```

```
x = 2000
```

```
C = A/x * sqrt(2/(pi*exp(1)))
```

```
C = 7.2591
```

### Question 2

```
r = 2
```

```
r = 2
```

```
N = 20
```

```
N = 20
```

```
x = (1-r^N)/(1-r)
```

```
x = 1048575
```

```
y = sum(r.^(0:N-1))
```

```
y = 1048575
```

### Question 3

```
theta = 0:pi/16:2*pi
```

```
theta = 1×33
```

|   |        |        |        |        |        |        |            |
|---|--------|--------|--------|--------|--------|--------|------------|
| 0 | 0.1963 | 0.3927 | 0.5890 | 0.7854 | 0.9817 | 1.1781 | 1.3744 ... |
|---|--------|--------|--------|--------|--------|--------|------------|

```
r = 10
```

```
r = 10
```

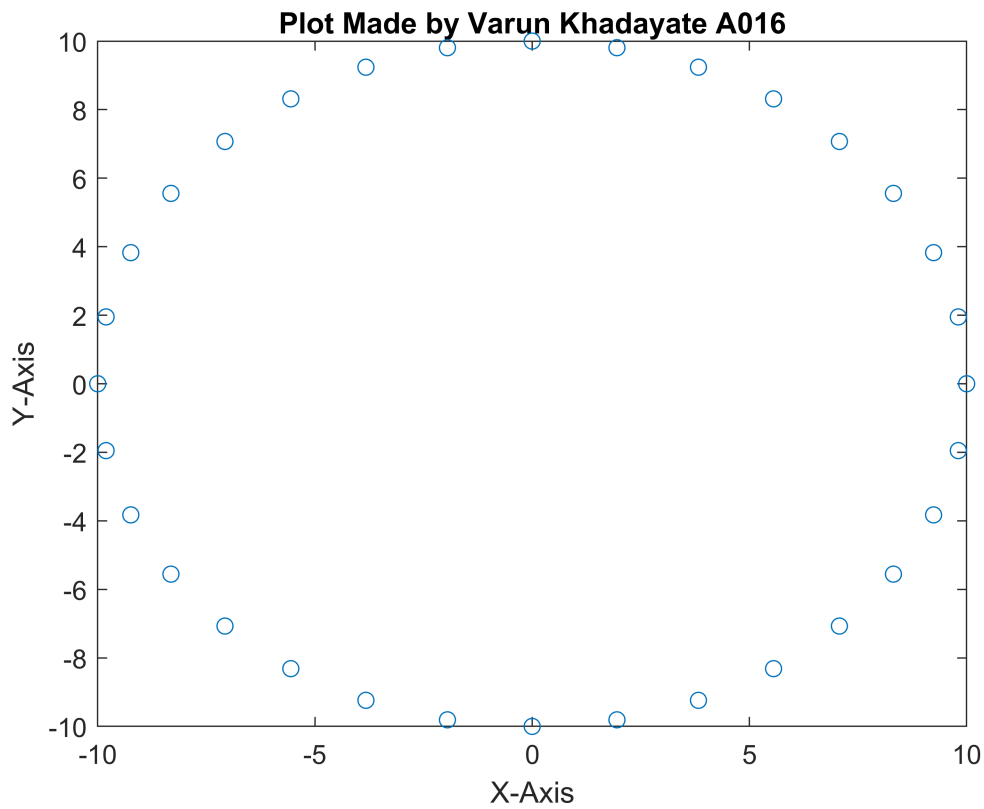
```
x = r*cos(theta)
```

|          |         |        |        |        |        |        |        |            |
|----------|---------|--------|--------|--------|--------|--------|--------|------------|
| x = 1×33 | 10.0000 | 9.8079 | 9.2388 | 8.3147 | 7.0711 | 5.5557 | 3.8268 | 1.9509 ... |
|----------|---------|--------|--------|--------|--------|--------|--------|------------|

```
y = r*sin(theta)
```

|          |   |        |        |        |        |        |        |            |
|----------|---|--------|--------|--------|--------|--------|--------|------------|
| y = 1×33 | 0 | 1.9509 | 3.8268 | 5.5557 | 7.0711 | 8.3147 | 9.2388 | 9.8079 ... |
|----------|---|--------|--------|--------|--------|--------|--------|------------|

```
plot(x,y,'o')
xlabel('X-Axis')
ylabel('Y-Axis')
title('Plot Made by Varun Khadayate A016')
```



|                                |                       |
|--------------------------------|-----------------------|
| Roll. No. A016                 | Name: Varun Khadayate |
| Class B.Tech CsBs              | Batch: 1              |
| Date of Experiment: 19-07-2021 | Subject: IT/WS        |

1. Launch MATLAB, create list of following variables in command window:

a)  $m=10$ ,  $n=25$ ,  $p=43$

Ans:

```
>> m = 10

m =

 10

>> n = 25

n =

 25

>> p = 43

p =

 43
```

b)  $A=m^2$ ,  $B=n^3$ ,  $C=(A+B)*p$

Ans:

```
>> A = m^2

A =

 100

>> B = n^3

B =

 15625

>> C = (A + B)*p

C =

 676175
```

c)  $t=0.1$ ,  $f=0.5$ ,  $a=5$ ,  $x=a*\sin(2\pi ft)$

Ans:

```
>> t = 0.1

t =

 0.1000

>> f = 0.5

f =

 0.5000

>> a = 5

a =

 5

>> x = a*sin(2*pi*f*t)

x =

 1.5451
```

d)  $y=mx+C$

Ans:

```
>> y = m*x + C

y =

 6.7619e+05
```

e)  $k=(t^2+1)(t^2-1)$

Ans:

```
>> k = (t^2 + 1)*(t^2 - 1)

k =

 -0.9999
```

2.

- (a) From Question1 make a new variable 'v', overwriting part (c), i.e.,  
 $x=a*\sin(2\pi ft)$   
by adding cosh(t)

Ans:

```
>> v = x + cosh(t)

v =

2.5501
```

- (b) Create variable 'r', store value to it to find the area of circle :

$$A=\pi r^2$$

where 'r' is the radius of circle. Further, using the built in function **namelengthmax** , find the maximum number of character in "A".

[Hint: store value of 'r' as 10]

Ans:

```
>> r = 10

r =

10

>> A = pi*r^2

A =

314.1593

>> A = namelengthmax

A =

63
```

3. Explore the solve command using MATLAB help and find the solution for the problem given :  
 $X + 1 = 2$ , find X

Ans:

```
>> syms X
>> solve(X + 1 == 2, X)

ans =

1
```

4. Complete the table using **help** command:

Ans:

| Command name | Purpose                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |      |                  |      |                  |       |                                            |       |                      |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------|------------------|------|------------------|-------|--------------------------------------------|-------|----------------------|
| whos         | <p>&gt;&gt; help whos</p> <p>whos List current variables, long form.</p> <p>whos is a long form of WHO. It lists all the variables in the current workspace, together with information about their size, bytes, class, etc.</p> <p>In a nested function, variables are grouped into those in the nested function and those in each of the containing functions, each group separated by a line of dashes.</p> <p>whos GLOBAL lists the variables in the global workspace.</p> <p>whos -FILE FILENAME lists the variables in the specified .MAT file.</p> <p>whos ... VAR1 VAR2 restricts the display to the variables specified.</p> <p>The wildcard character '*' can be used to display variables that match a pattern. For instance, whos A* finds all variables in the current workspace that start with A.</p> <p>whos -REGEXP PAT1 PAT2 can be used to display all variables matching the specified patterns using regular expressions. For more information on using regular expressions, type "doc regexp" at the command prompt.</p> <p>Use the functional form of whos, such as whos('-file',FILE,V1,V2), when the filename or variable names are stored as a character vector or string scalar.</p> <p>S = whos(...) returns a structure with the fields:</p> <table><tr><td>name</td><td>-- variable name</td></tr><tr><td>size</td><td>-- variable size</td></tr><tr><td>bytes</td><td>-- number of bytes allocated for the array</td></tr><tr><td>class</td><td>-- class of variable</td></tr></table> | name | -- variable name | size | -- variable size | bytes | -- number of bytes allocated for the array | class | -- class of variable |
| name         | -- variable name                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |                  |      |                  |       |                                            |       |                      |
| size         | -- variable size                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |      |                  |      |                  |       |                                            |       |                      |
| bytes        | -- number of bytes allocated for the array                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |      |                  |      |                  |       |                                            |       |                      |
| class        | -- class of variable                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |      |                  |      |                  |       |                                            |       |                      |



|  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>For more information, see the <a href="#">clear Reference page</a>.</p> <p><code>clear IMPORT</code> clears the base import list. It can only be issued at the command prompt. It cannot be used in a function or a script.</p> <p><code>clear CLASSES</code> is the same as <code>clear ALL</code> except that class definitions are also cleared. If any objects exist outside the workspace (say in userdata or persistent in a locked program file) a warning will be issued and the class definition will not be cleared.</p> <p>Calling <code>clear CLASSES</code> decreases code performance and is usually unnecessary.</p> <p>If you modify a class definition, MATLAB automatically updates it.</p> <p>For more information, see the <a href="#">clear Reference page</a>.</p> <p><code>clear JAVA</code> is the same as <code>clear ALL</code> except that java classes on the dynamic java path (defined using <code>JAVACLASSPATH</code>) are also cleared.</p> <p><code>clear VAR1 VAR2 ...</code> clears the variables specified. The wildcard character '*' can be used to clear variables that match a pattern. For instance, <code>clear X*</code> clears all the variables in the current workspace that start with X.</p> <p><code>clear -REGEXP PAT1 PAT2</code> can be used to match all patterns using regular expressions. This option only clears variables. For more information on using regular expressions, type "doc regexp" at the command prompt.</p> <p>If X is global, <code>clear X</code> removes X from the current workspace, but leaves it accessible to any functions declaring it global.</p> <p><code>clear GLOBAL -REGEXP PAT</code> removes global variables that match regular expression patterns.</p> |
|--|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|       | <p>Note that to clear specific global variables, the GLOBAL option must come first. Otherwise, all global variables will be cleared.</p> <p>clear FUN clears the function specified. If FUN has been locked by MLOCK it will remain in memory. If FUN is a script or function that is currently executing, then it is not cleared. Use a partial path (see PARTIALPATH) to distinguish between different overloaded versions of FUN. For instance, 'clear inline/display' clears only the INLINE method for DISPLAY, leaving any other implementations in memory.</p> <p>Examples for pattern matching:</p> <pre>clear a*           % Clear variables starting with "a" clear -regexp ^b\d{3}\$ % Clear variables starting with "b" and                         % followed by 3 digits clear -regexp \d    % Clear variables containing any digits</pre> <p>See also clearvars, who, whos, mlock, munlock, persistent, import.</p> <p>Documentation for clear<br/>Other functions named clear</p> |
| pwd   | <pre>&gt;&gt; help pwd pwd Show (print) current working directory. pwd displays the current working directory.</pre> <p>S = pwd returns the current directory in the string S.</p> <p>See also cd.</p> <p>Documentation for pwd</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| diary | <pre>&gt;&gt; help diary diary Save text of MATLAB session. diary FILENAME causes a copy of all subsequent command window input and most of the resulting command window output to be appended to the named file. If no file is specified, the file 'diary' is used.</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

|  |                                                                                                                                                                                                                                                                                        |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | <p>diary OFF suspends it.<br/> diary ON turns it back on.<br/> diary, by itself, toggles the diary state.</p> <p>Use the functional form of diary, such as<br/> diary('file'),<br/> when the file name is stored in a string.</p> <p>See also save.</p> <p>Documentation for diary</p> |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

5. Given  $\theta = 145$  degrees. A vector can be represented by its rectangular coordinates  $x$  and  $y$  or by its polar coordinates  $r$  and  $\theta$ .  $\theta$  is measured in radians. The relationship between them is given by the equations:

$$x = r * \cos(\theta)$$

$$y = r * \sin(\theta)$$

Assign values for the polar coordinates to variables  $r$  and  $\theta$ . Then, using these values, assign the corresponding rectangular coordinates to variables  $x$  and  $y$ .

Ans:

```
>> r = 10

r =

 10

>> theta = 145

theta =

 145

>> x = r * cos(theta)

x =

 8.8386

>> y = r * sin(theta)

y =

 4.6775
```

6. The combined resistance  $R_r$  of three resistors  $R_1$ ,  $R_2$ , and  $R_3$  in parallel is given by

$$R_r = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3}}$$

Create variables for the three resistors and store values in each, and then calculate the combined resistance. (Hint: consider  $R_1=50\Omega$ ,  $R_2=25\Omega$  and  $R_3=60\Omega$ )

Ans:

```
>> R1 = 50

R1 =

50

>> R2 = 25

R2 =

25

>> R3 = 60

R3 =

60

>> R = (1/((1/R1)+(1/R2)+(1/R3)))

R =

13.0435
```

## Practical 6

Name : Varun Khadayate

Roll No. : A016

### Basic Plotting-2D and 3D

#### Part A: Theory

`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,

`surf1` - 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,

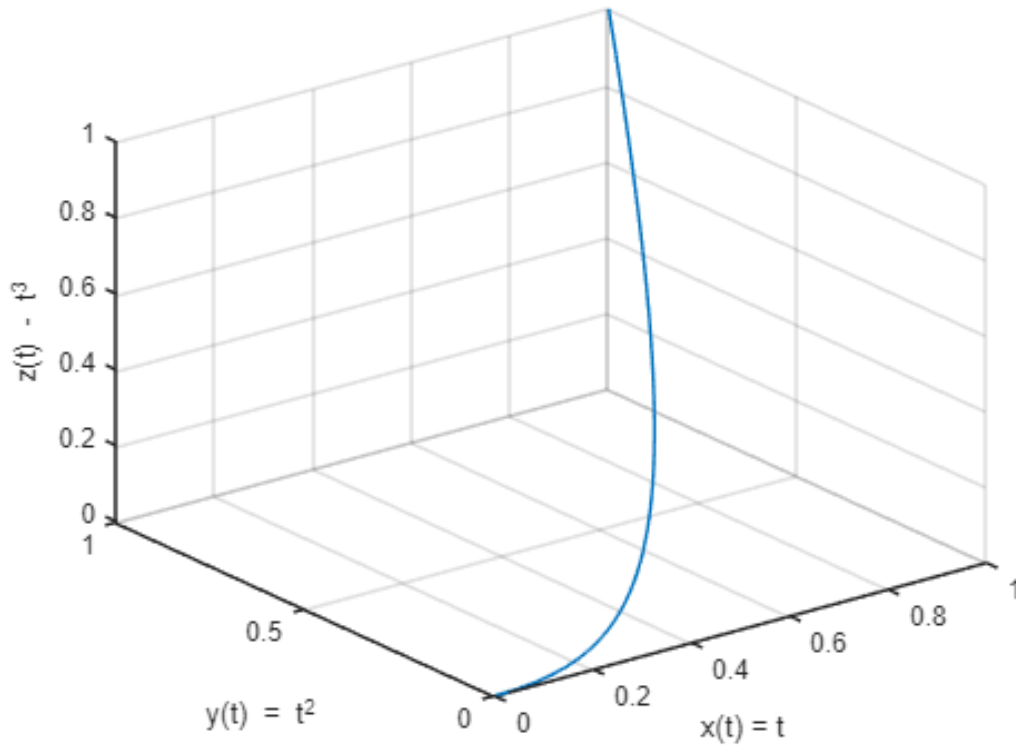
`cylinder` - generates a cylinder,

`ellipsoid` - generates an ellipsoid, and

`sphere` - generates a sphere.

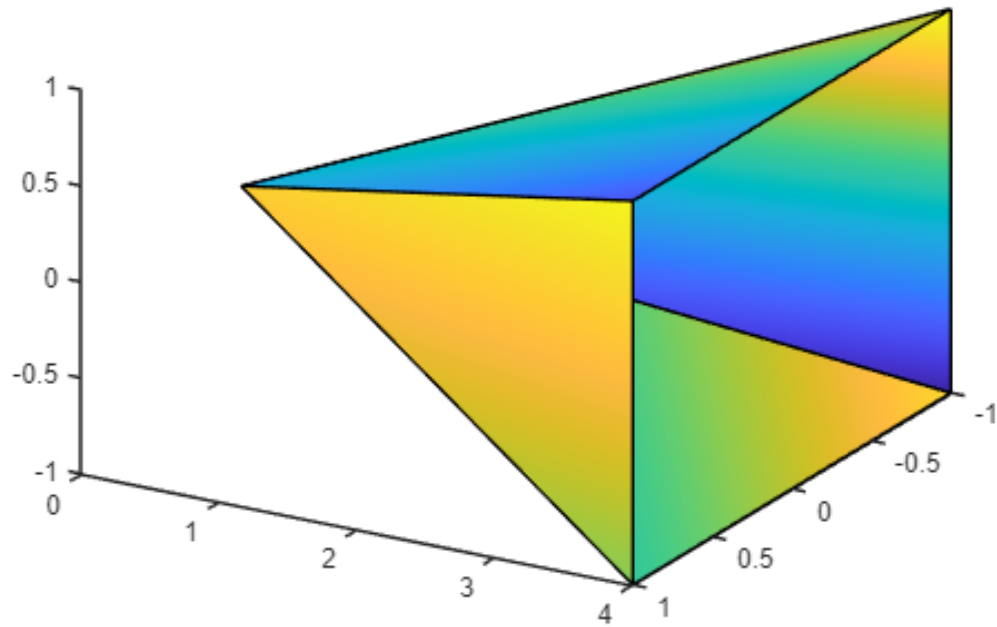
### plot3

```
t = linspace (0, 1, 100);
x = t; y = t.^ 2; z = t.^ 3;
plot3(x, y,z), grid
xlabel ('x(t) = t')
ylabel ('y(t) = t^2')
zlabel ('z(t) = t^3')
```



### fill3

```
X = [0 0 0 0; 1 1 -1 1;1 -1 -1 -1] ;
Y = [0 0 0 0; 4 4 4 4;4 4 4 4] ;
Z = [0 0 0 0; 1 1 -1 -1;-1 1 1 -1];
fillcolor=rand(3,4);
fill3(X,Y,Z,fillcolor)
view(120,30)
```

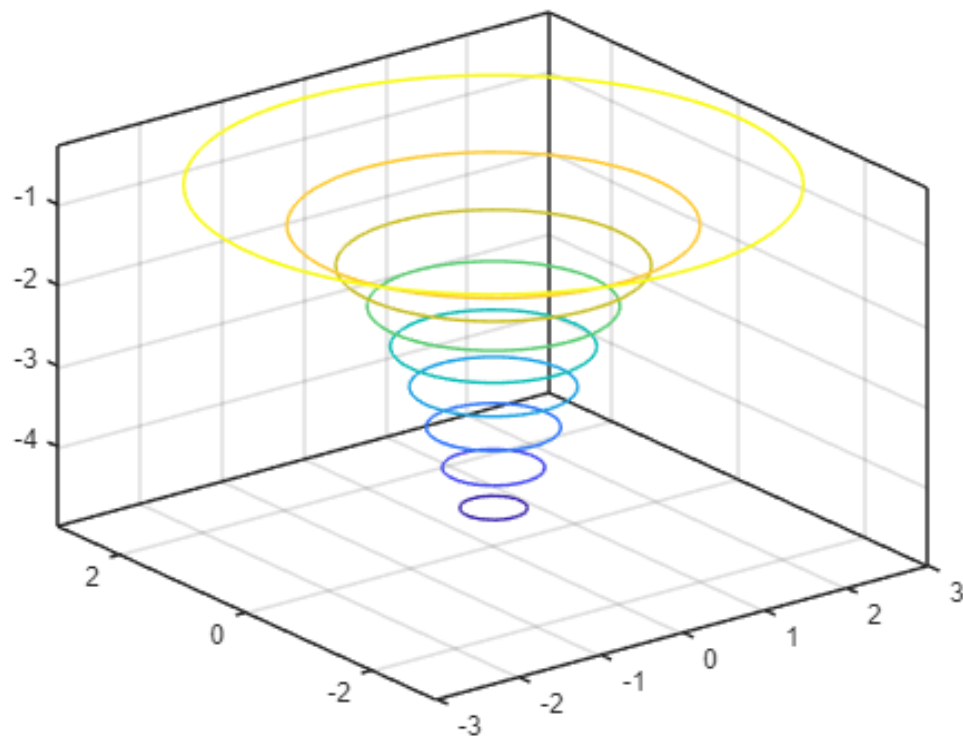


### contour3

```
r = linspace(-3,3,50);
[x,y] = meshgrid(r,r);
z = -5./(1+ x.^2 + y.^2)
```

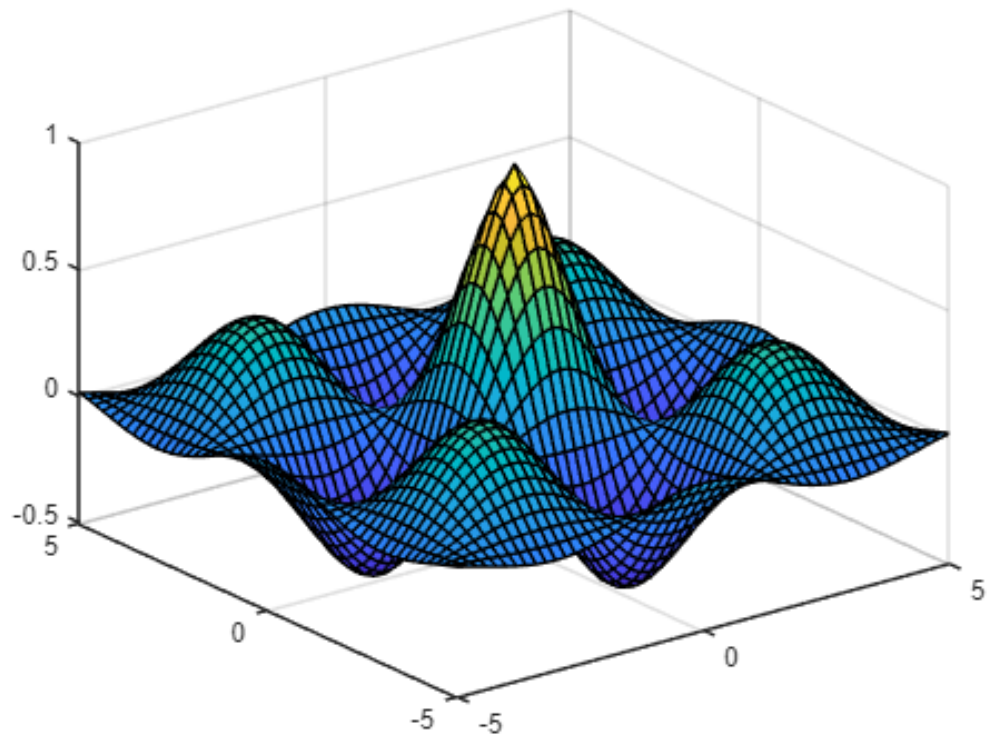
```
z = 50x50
-0.2632 -0.2735 -0.2842 -0.2953 -0.3067 -0.3184 -0.3304 -0.3427 ...
-0.2735 -0.2847 -0.2964 -0.3084 -0.3209 -0.3337 -0.3469 -0.3604
-0.2842 -0.2964 -0.3090 -0.3221 -0.3358 -0.3498 -0.3644 -0.3793
-0.2953 -0.3084 -0.3221 -0.3364 -0.3513 -0.3668 -0.3828 -0.3993
-0.3067 -0.3209 -0.3358 -0.3513 -0.3676 -0.3845 -0.4022 -0.4204
-0.3184 -0.3337 -0.3498 -0.3668 -0.3845 -0.4031 -0.4225 -0.4427
-0.3304 -0.3469 -0.3644 -0.3828 -0.4022 -0.4225 -0.4439 -0.4663
-0.3427 -0.3604 -0.3793 -0.3993 -0.4204 -0.4427 -0.4663 -0.4910
-0.3551 -0.3742 -0.3946 -0.4162 -0.4392 -0.4637 -0.4895 -0.5169
-0.3676 -0.3881 -0.4101 -0.4335 -0.4586 -0.4853 -0.5137 -0.5438
⋮
```

```
contour3(x,y,z)
```



## surf

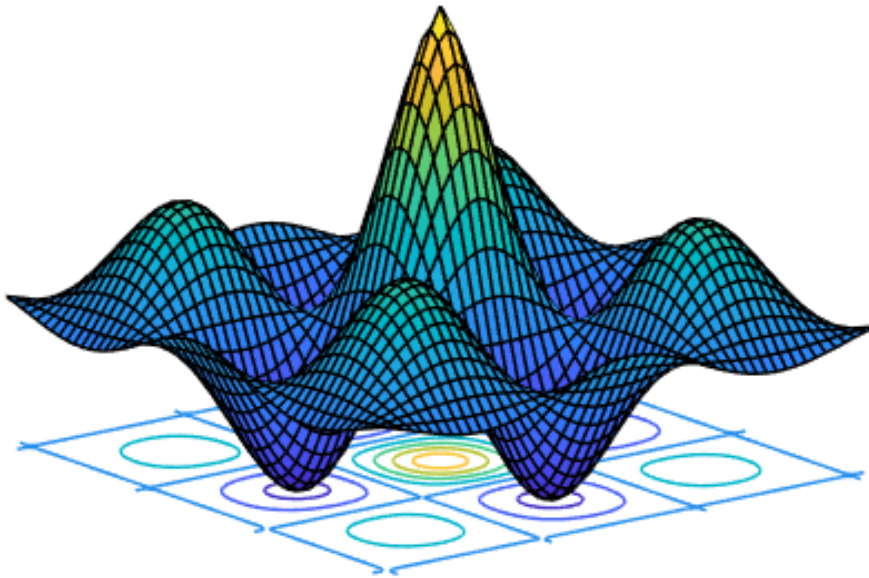
```
u = -5:.2:5;
[X,Y] = meshgrid(u,u);
Z = cos(X) .* cos(Y) .* exp(-sqrt(X.^2+ Y.^2)/4);
surf(X,Y,Z)
```



## surfz

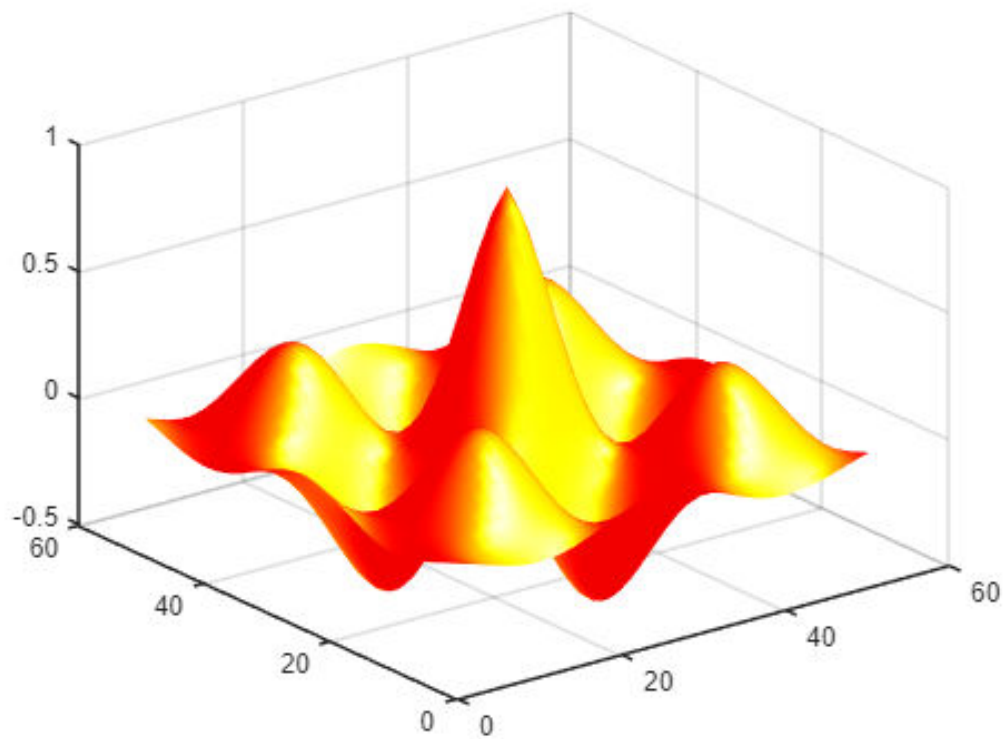
```
u = -5:.2:5;
[X,Y] = meshgrid(u,u);
Z = cos(X) .* cos(Y) .* exp(-sqrt(X.^2+ Y.^2)/4);
surfz(Z)
view(-37.5,20)
axis('off')
```





## surf1

```
u = -5:.2:5;
[X,Y] = meshgrid(u,u);
z = cos(X) .* cos(Y) .* exp(-sqrt(X.^2+ Y.^2)/4);
surf1(Z)
shading interp
colormap hot
```

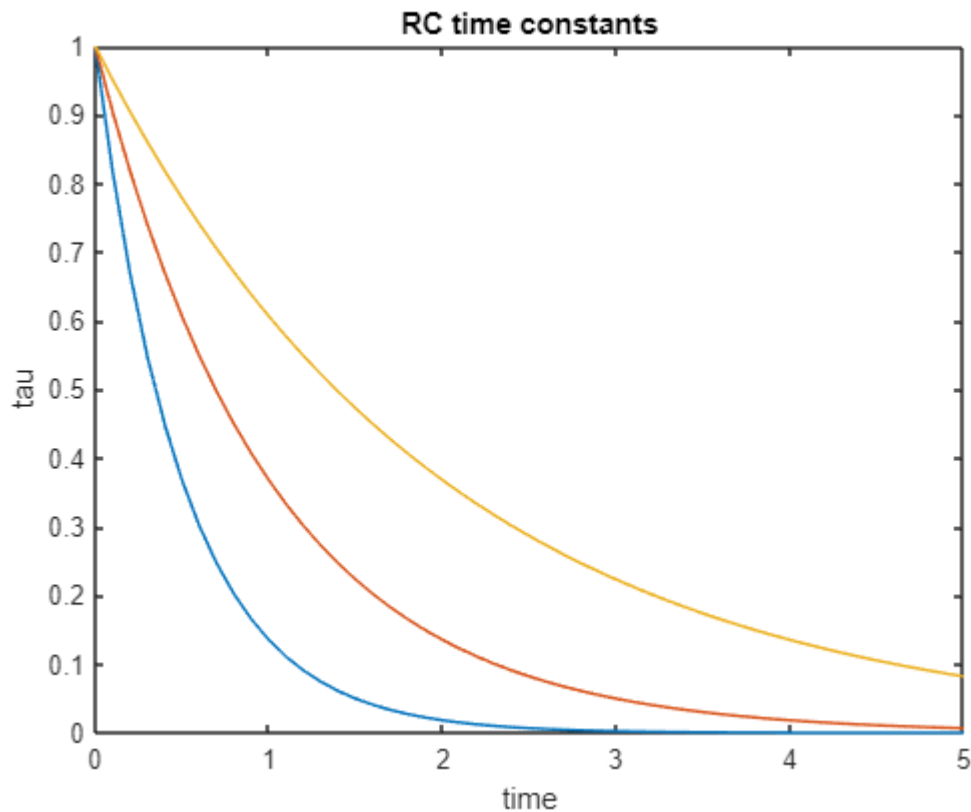


## Part B: Practical

1. Plot voltage vs time for various RC time constants

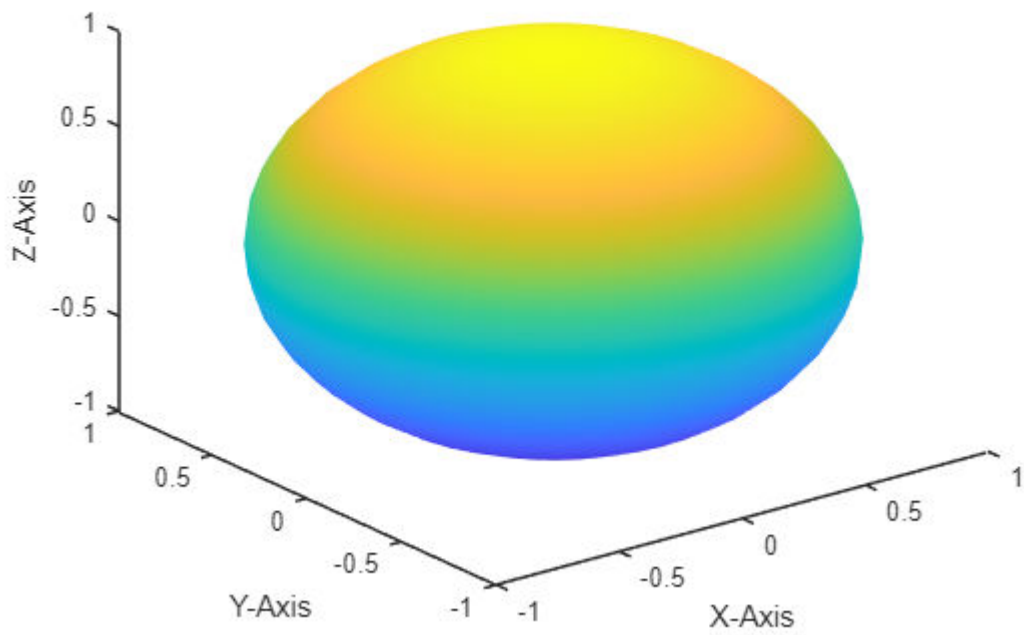
$$\frac{v}{V} = e^{-\frac{t}{\tau}}$$

```
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')
```



2. Plot a sphere, which is defined as  $[x(t, s), y(t, s), z(t, s)] = [\cos(t) \cos(s), \cos(t) \sin(s), \sin(t)]$  (use 'surf'). for  $t, s = [0, 2\pi]$ . Make first equal axes, then remove them. Use 'shading interp' to remove black lines.

```
[s,t] = meshgrid(0:0.05*pi:2*pi,0:0.05*pi:2*pi);
x = cos(t) .* cos(s);
y = cos(t) .* sin(s);
z = sin(t);
surf(x,y,z)
xlabel('X-Axis')
ylabel('Y-Axis')
zlabel('Z-Axis')
grid off
shading interp
```



**1) Using the top-down design approach, write an algorithm for 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 remove 2 slices

etc.

**2) Write a simple script that will calculate the volume of a hollow sphere,**

$$\frac{4\pi}{3} (r_o^3 - r_i^3)$$

where  $r_i$  is the inner radius and  $r_o$  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** to view the comment in 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 decimal places.**

```
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 of 10 with 4 decimal places

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

```
The number is 12345.6789
```

- in a field width of 10 with 2 decimal places

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

```
The number is 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 prompt the 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  $\alpha$  in degrees, the third side a is calculated as follows:**

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

**Write a script *thirdside* that will prompt the user and read in values for b, c, and  $\alpha$  (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 exactly like 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/meters<sup>2</sup>. Here is an example of executing the script:**

```
>> lumin
```

This script will calculate the luminosity of a star.

When prompted, enter the star's distance 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 number is 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

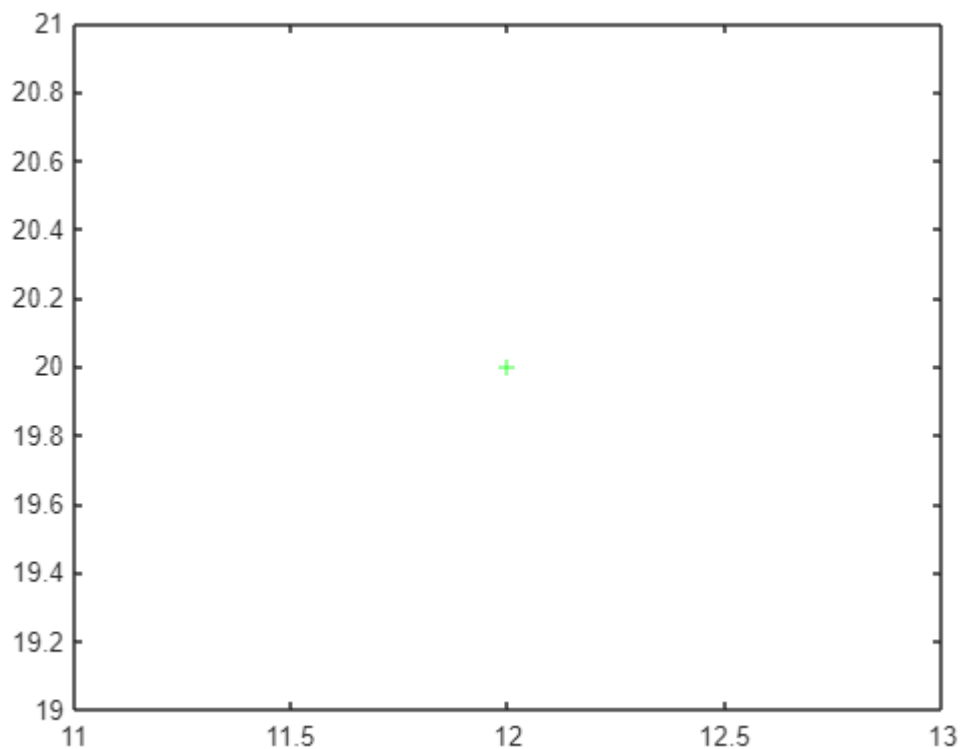
```
fprintf('Your char is %6c!\n', mychar)
```

```
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 coordinate and 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+')
```

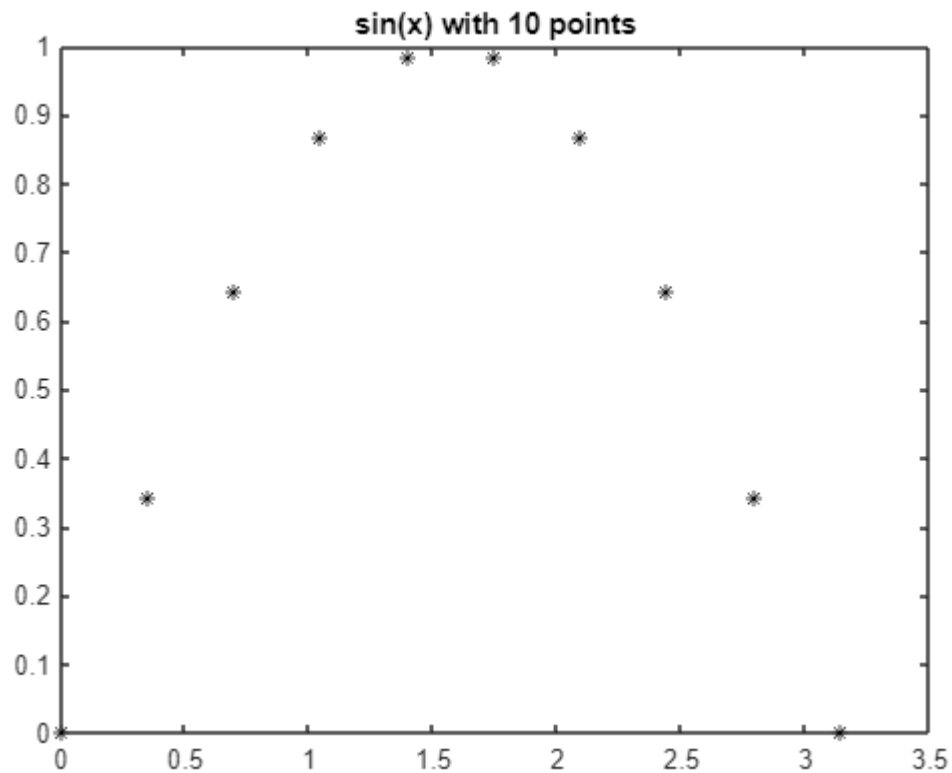


**15) Plot  $\sin(x)$  for x values ranging from 0 to  $\pi$  (in separate Figure Windows):**

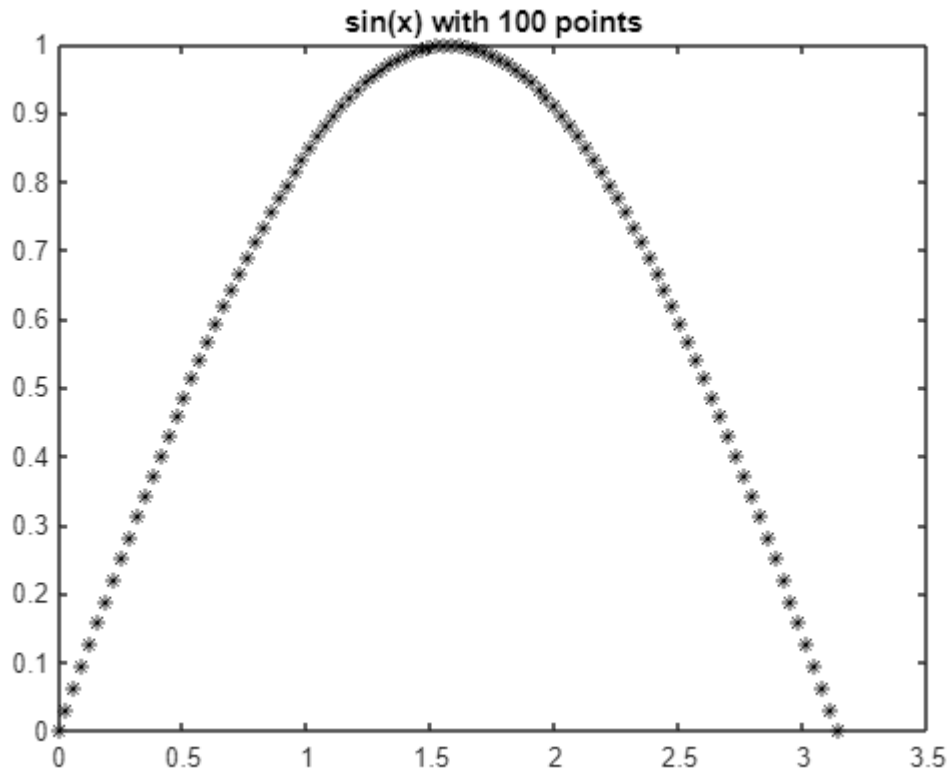
- using 10 points in this range
- using 100 points in this range

```
% 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 various altitudes. The altitudes are in the first column and the temperatures in the second. For example, it may look like this:**

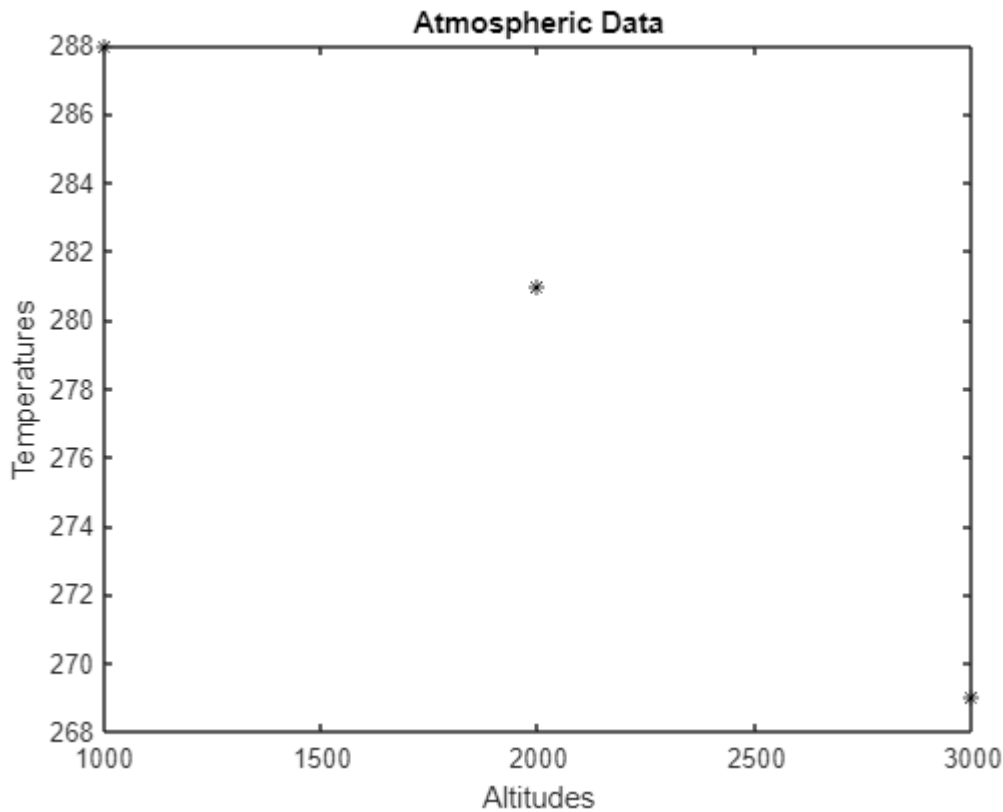
|      |     |
|------|-----|
| 1000 | 288 |
| 2000 | 281 |
| 3000 | 269 |

```
% 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, and plot the squares.**

```

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

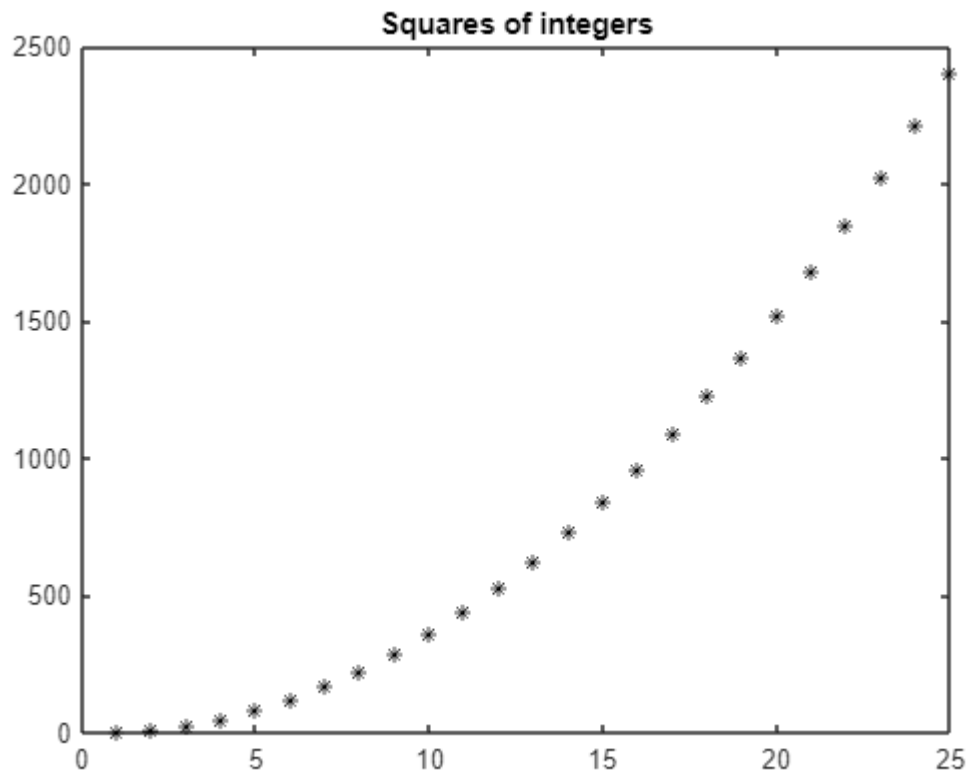
```

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 matrix of 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 = 3x6
```

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 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 = 2x6
```

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 69 | 53 | 77 | 97 | 79 | 50 |
| 78 | 52 | 89 | 56 | 73 | 67 |

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

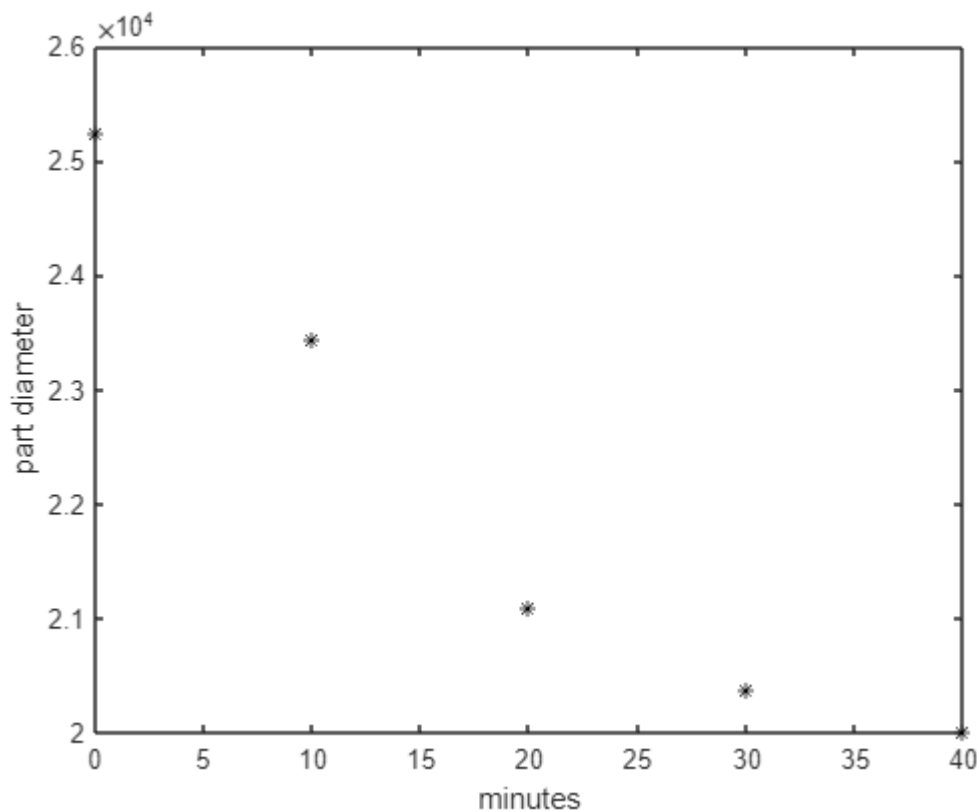
21) A particular part is being turned on a lathe. The diameter of the part is supposed to be 20,000 mm. The diameter is measured every 10 minutes and the results are stored in a file called *partdiam.dat*. Create a data file to

**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 = 5x2
 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
```

*Renewable energy sources such as biomass are 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 meter of 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** of the file.
- A function is called whereas a script is executed.
- Arguments are passed to functions but not to scripts.
- Functions can return arguments whereas scripts cannot.
- The block comment is typically in the beginning of a script but under the function header.
- The scope of variables is different: scripts use the baseworkspace, whereas functions have their own workspaces.

## 26) In quantum mechanics, the angular wavelength for a wavelength $\lambda$ 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 argument is 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 argument is 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 might expect, as shown here.

[illegible]

```

% 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

```

```
% 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 amount of snow
rain = input('How much rain in inches: ');
snow = rainToSnow(rain);
fprintf('%0.1f inches of rain would be ', rain)
```

25.0 inches of rain would be

```
fprintf('%0.1f 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 temperature of reservoir is %d and %d respectively\n',Tc,Th)
```

The Cold and Hot temperature 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  $\tau$  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?

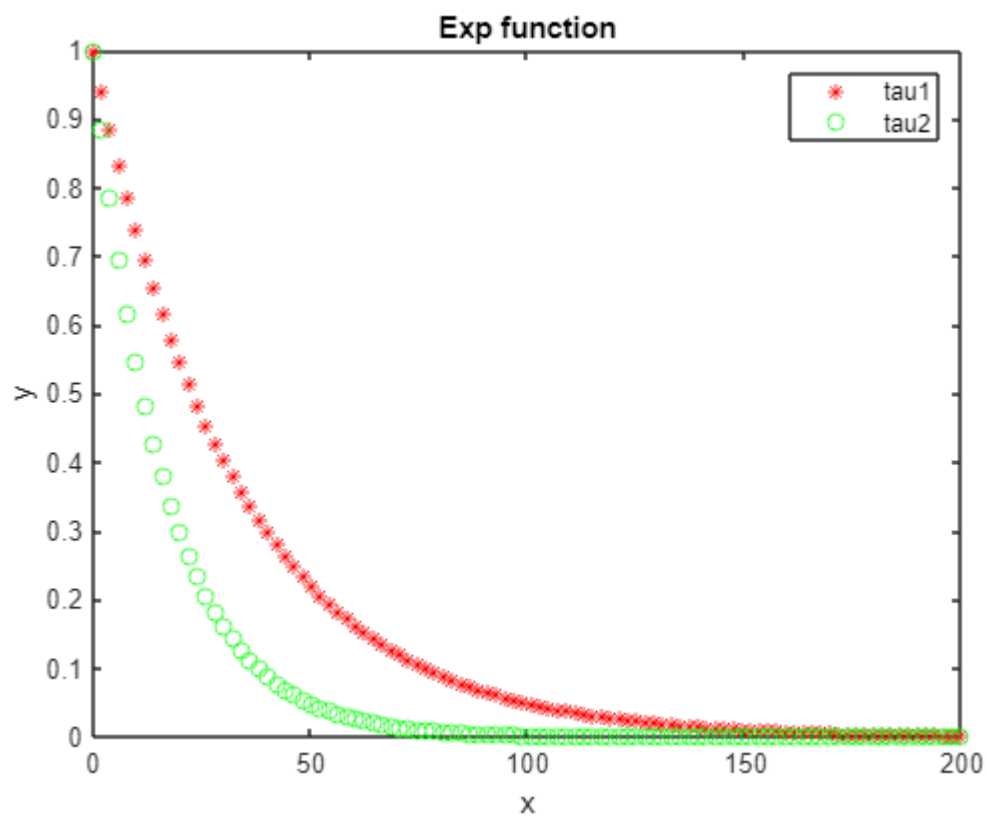
```
A = 1;
tau1 = input('Enter a time constant: ');
tau2 = input('Enter another time constant: ');
tstart = input('Enter the beginning t: ');
tend = input('Enter the end of t: ');
fprintf("The time constant 1 and 2 are %f and %f respectively",tau1,tau2)
```

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')
```

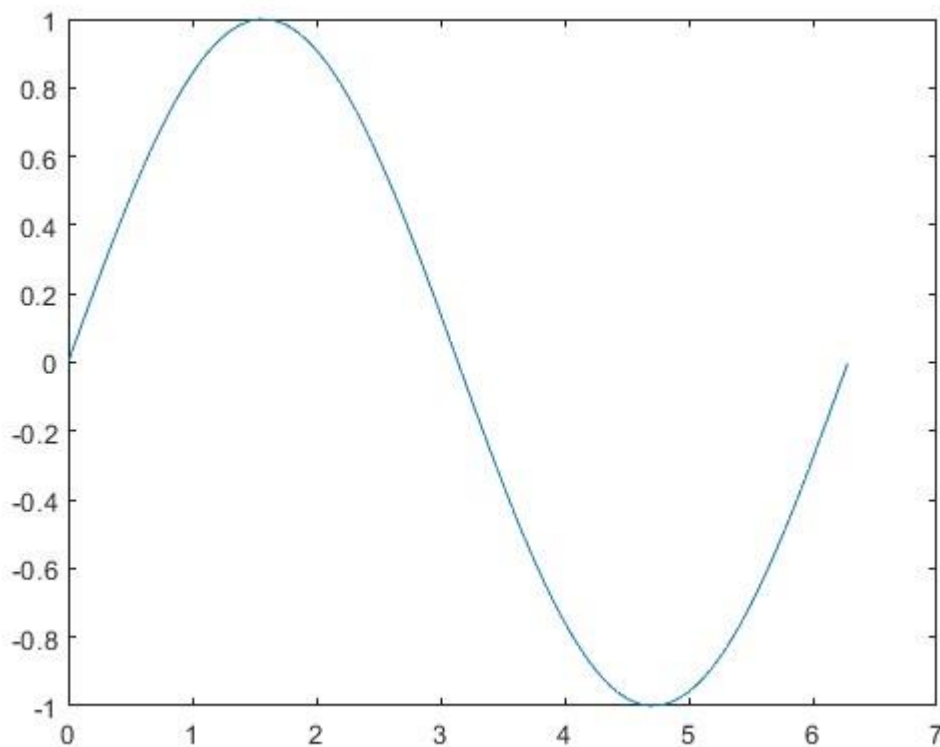


|                                |                       |
|--------------------------------|-----------------------|
| Roll No.: A016                 | Name: Varun Khadayate |
| Class: B. Tech CsBs            | Batch: 1              |
| Date of Experiment: 18-09-2022 | Subject: ITWS         |

## Lab 5: Basic Plotting

- Creates a 2-D line plot of the data in y versus the corresponding values in x. Create x as a vector of linearly spaced values between 0 and  $2\pi$ . Use an increment of  $\pi/100$  between the values. Create y as sine values of x. Calling the plot function will display the x versus y data on a scaled viewing domain and range.

```
x = 0:pi/100:2*pi;
y = sin(x);
plot(x,y)
```



- Explore various specifications of plot command and put it in a tabular format.

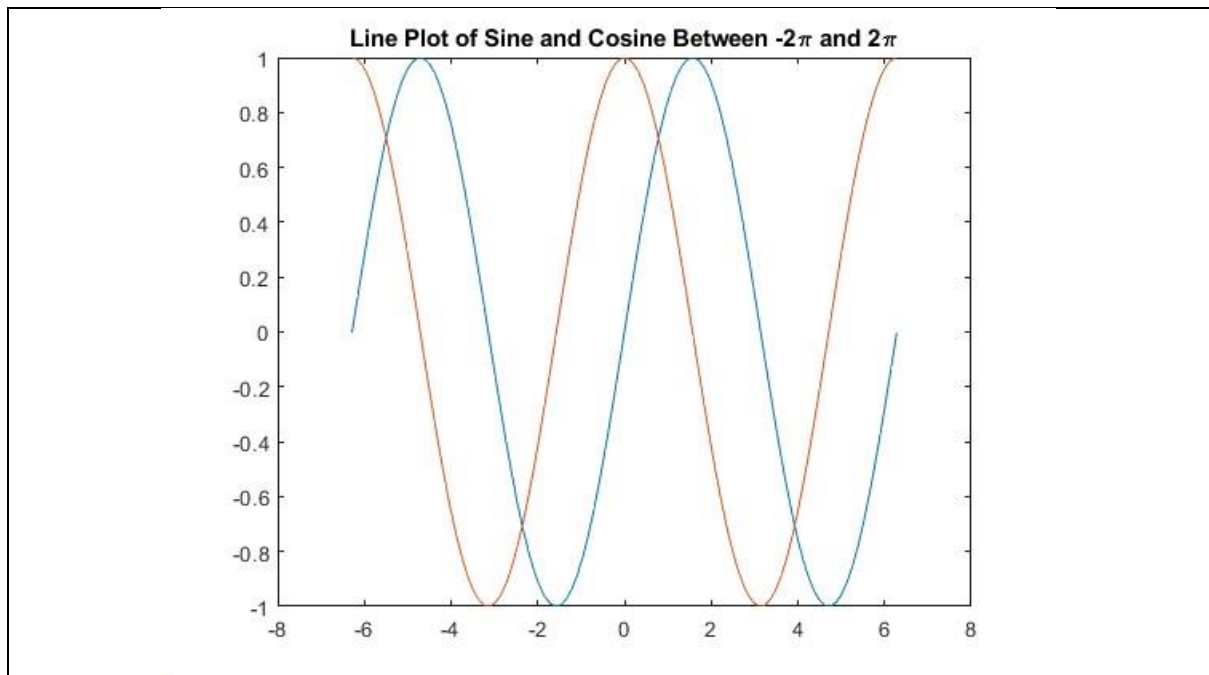
| <b>Specification</b> | <b>Purpose</b>                                        | <b>Syntax</b>                        | <b>Example</b>                           |
|----------------------|-------------------------------------------------------|--------------------------------------|------------------------------------------|
| xlabel, ylabel       | xlabel( <b>txt</b> ) labels the x-axis of the current | xlabel('string')<br>ylabel('string') | xlabel('X-Axis')<br><br>ylabel('Y-Axis') |



|        |                                                                                                                                                                             |                 |                                         |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|-----------------------------------------|
|        | axes or standalone visualization.<br><br>ylabel( <b>txt</b> ) labels the y-axis of the current axes or standalone visualization.                                            |                 |                                         |
| title  | title( <b>titletext</b> ) adds the specified title to the current axes or standalone visualization.                                                                         | title('string') | title('Plot between X-axis and Y-axis') |
| legend | The legend automatically updates when you add or delete data series from the axes. This command creates a legend in the current axes, which is returned by the gca command. | legend          | legend('x','y')                         |

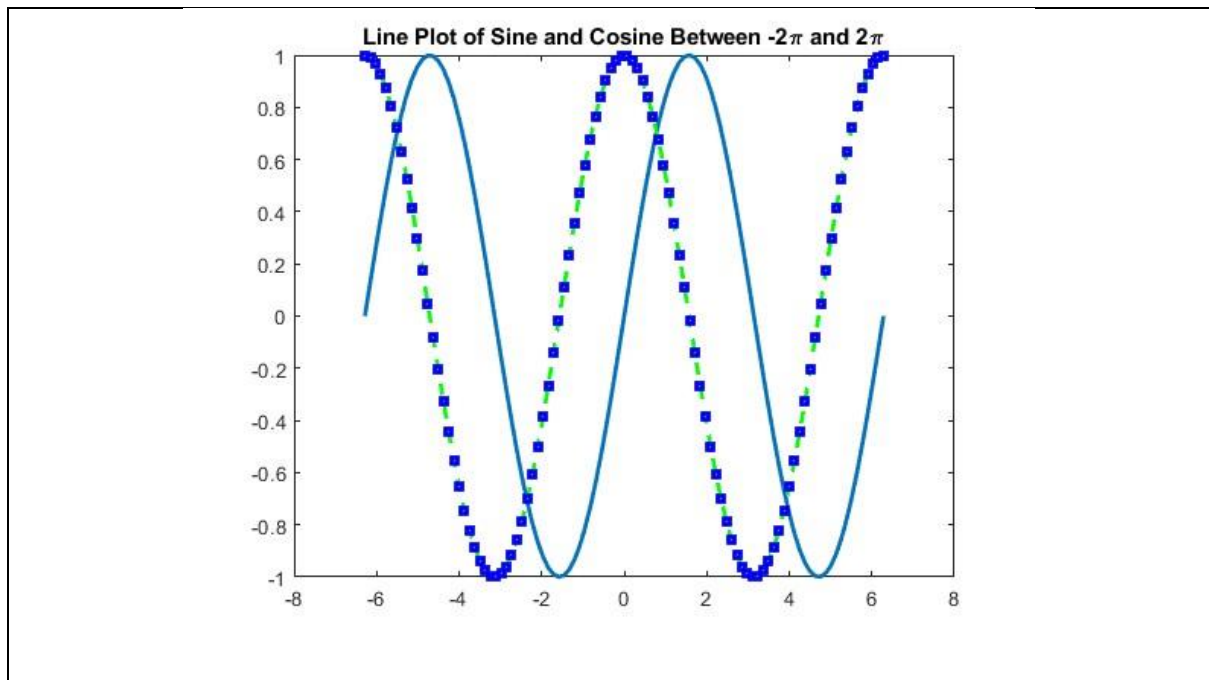
3. Create x as 100 linearly spaced values between  $-2\pi$  and  $2\pi$ . Create y1 and y2 as sine and cosec values of x. Plot both sets of data. Add a title to the chart by using the title function. To display the Greek symbol  $\pi$ , use pi.

```
x = linspace(-2*pi,2*pi,100);
y1 = sin(x);
y2 = cos(x);
figure
plot(x,y1,x,y2)
title('Line Plot of Sine and Cosine Between -2\pi and 2\pi')
```



4. In the above plot, use the LineSpec option to specify a dashed green line with square markers. Use Name,Value pairs to specify the line width, point marker size, and point marker colors. Set the marker edge color to blue and set the marker face color using an RGB color value.

```
x = linspace(-2*pi,2*pi,100);
y1 = sin(x);
y2 = cos(x);
figure
plot(x,y1,x,y2,'--gs',...
 'LineWidth',2,...
 'MarkerSize',5,...
 'MarkerEdgeColor','b',...
 'MarkerFaceColor',[0.5,0.5,0.5])
title('Line Plot of Sine and Cosine Between -2\pi and 2\pi')
```



5. Display Multiple Plots in a Figure Window : Call the tiledlayout function to create a 2-by-1 tiled chart layout. Call the nexttile function to create an axes object and return the object as ax1. Create the top plot by passing ax1 to the plot function. Add a title and y-axis label to the plot by passing the axes to the title and ylabel functions. Repeat the process to create the bottom plot.

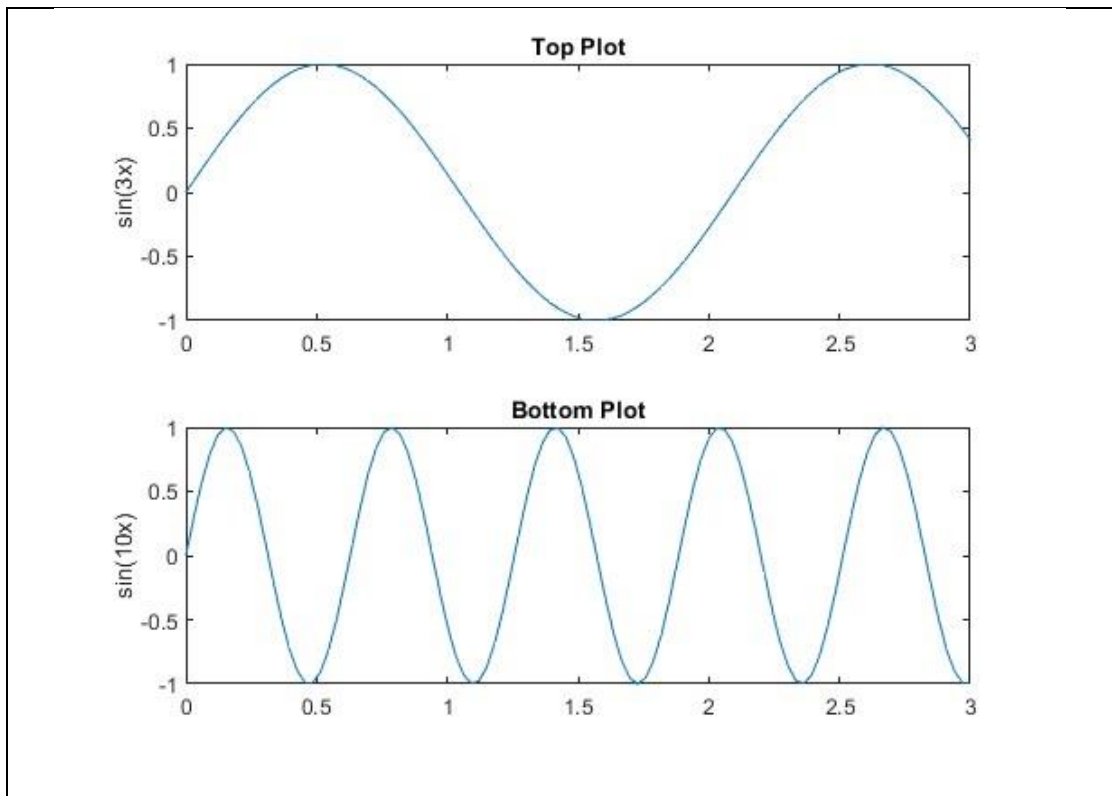
(y = sin(3x) %for 1st plot)

(y = sin(10x) %for 2nd plot)

```
x = linspace(0,3);
y1 = sin(3*x);
y2 = sin(10*x);
tiledlayout(2,1)

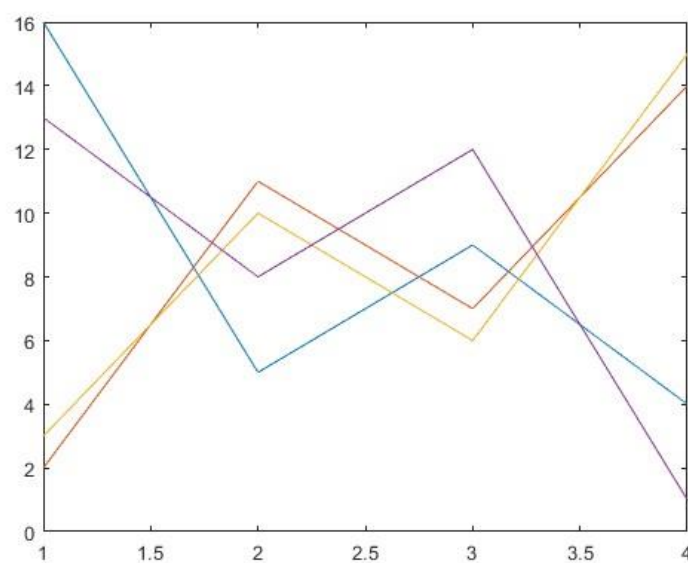
% Top plot
ax1 = nexttile;
plot(ax1,x,y1)
title(ax1,'Top Plot')
ylabel(ax1,'sin(3x)')

% Bottom plot
ax2 = nexttile;
plot(ax2,x,y2)
title(ax2,'Bottom Plot')
ylabel(ax2,'sin(10x)')
```



6. Define Y as the 4-by-4 matrix returned by the magic function. Create a 2-D line plot of Y. MATLAB® plots each matrix column as a separate line.

```
Y = magic(4);
figure
plot(Y)
```



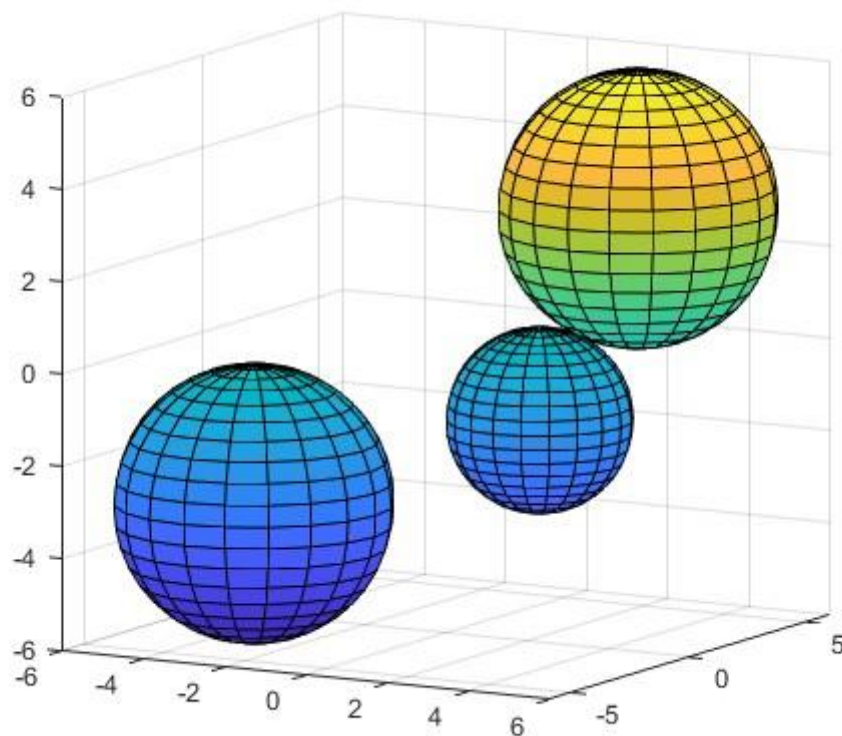
|                                |                       |
|--------------------------------|-----------------------|
| Roll No.: A016                 | Name: Varun Khadayate |
| Class: B. Tech CsBs            | Batch: 1              |
| Date of Experiment: 18-09-2022 | Subject: ITWS         |

## Lab 6: M-script file

1.
  - a. Create a file called mysphere.mat
  - b. Create and plot a sphere using sphere function. Sphere function creates a unit circle which can be scaled using surf function (hint: try different values of 'r' using surf function).
  - c. Run and observe the output
  - d. What is your observation?

While running the code with different value of r there are different sphere, and we can see that the dimension also changes. Below is the screenshot of 3 sphere in one graph

```
[x y z] = sphere;
a=[3 3 3 3;-3 -3 -3 3;0 4 -2 2];
s1=surf(x*a(1,4)+a(1,1),y*a(1,4)+a(1,2),z*a(1,4)+a(1,3));
hold on
s2=surf(x*a(2,4)+a(2,1),y*a(2,4)+a(2,2),z*a(2,4)+a(2,3));
s3=surf(x*a(3,4)+a(3,1),y*a(3,4)+a(3,2),z*a(3,4)+a(3,3));
daspect([1 1 1])
view(30,10)
```



2.

a. Create an excel sheet consisting the following data :

| Temperatures of Cities<br>as on 20.6.2006 |      |      |
|-------------------------------------------|------|------|
| City                                      | Max. | Min. |
| Ahmedabad                                 | 38°C | 29°C |
| Amritsar                                  | 37°C | 26°C |
| Bangalore                                 | 28°C | 21°C |
| Chennai                                   | 36°C | 27°C |
| Delhi                                     | 38°C | 28°C |
| Jaipur                                    | 39°C | 29°C |
| Jammu                                     | 41°C | 26°C |
| Mumbai                                    | 32°C | 27°C |

b. Explore xlsread function using help

c. Read the excel file created in step a) using xlsread function

d. d) Create an M-File and write a script which finds the following:

i. Compare and identify the city that recorded the max temperature

ii. Plot the given data using bar graph

help **xlsread**

**xlsread** Read Microsoft Excel spreadsheet file.

\*\*\* **xlsread** is not recommended \*\*\*

See readtable, readmatrix, or readcell.

-----  
[NUM,TXT,RAW]=**xlsread**(FILE) reads data from the first worksheet in the Microsoft Excel spreadsheet file named FILE and returns the numeric data in array NUM. Optionally, returns the text fields in cell array TXT, and the unprocessed data (numbers and text) in cell array RAW.

[NUM,TXT,RAW]=**xlsread**(FILE,SHEET) reads the specified worksheet.

[NUM,TXT,RAW]=**xlsread**(FILE,SHEET,RANGE) reads from the specified SHEET and RANGE. Specify RANGE using the syntax 'C1:C2', where C1 and C2 are opposing corners of the region. Not supported for XLS files in BASIC mode.

[NUM,TXT,RAW]=**xlsread**(FILE,SHEET,RANGE,'basic') reads from the spreadsheet in BASIC mode, the default on systems without Excel for Windows. RANGE is supported for XLSX files only.

[NUM,TXT,RAW]=**xlsread**(FILE,RANGE) reads data from the specified RANGE of the first worksheet in the file. Not supported for XLS files in BASIC mode.

The following syntaxes are supported only on Windows systems with Excel software:

[NUM,TXT,RAW]=**xlsread**(FILE,-1) opens an Excel window to select data interactively.

[NUM,TXT,RAW,CUSTOM]=**xlsread**(FILE,SHEET,RANGE,'',FUNCTIONHANDLE) reads from the spreadsheet, executes the function associated with FUNCTIONHANDLE on the data, and returns the final results. Optionally, returns additional CUSTOM output, which is the second output from the function. **xlsread** does not change the data stored in the spreadsheet.

#### Input Arguments:

**FILE** Name of the file to read. **SHEET** Worksheet to read. One of the following:

- \* The worksheet name.
- \* Positive, integer-valued scalar indicating the worksheet index.

**RANGE** Character vector or string that specifies a rectangular portion of the worksheet to read. Not case sensitive. Use Excel A1 reference style. If you do not specify a **SHEET**, **RANGE** must include both corners and a colon character (:), even for a single cell (such as 'D2:D2').

'basic' Flag to request reading in BASIC mode, which is the default for systems without Excel for Windows. In BASIC mode, **xlsread**:

- \* Reads XLS, XLSX, XLSM, XLTX, and XLTM files only.
- \* Does not support an xlRange input when reading XLS files. In this case, use '' in place of xlRange.
- \* For XLS files, requires a name to specify the **SHEET**, and the name is case sensitive.
- \* Does not support function handle inputs.
- \* Imports all dates as Excel serial date numbers. Excel serial date numbers use a different reference date than MATLAB date numbers.

-1 Flag to open an interactive Excel window for selecting data. Select the worksheet, drag and drop the mouse over the range you want, and click OK. Supported only on Windows systems with Excel software.

#### FUNCTIONHANDLE

Handle to your custom function. When **xlsread** calls your function, it passes a range interface from Excel to provide access to the data. Your function must include this interface (of type 'Interface.Microsoft\_Excel\_5.0\_Object\_Library.Range', for example) both as an input and output argument.

#### Notes:

- \* On Windows systems with Excel software, **xlsread** reads any file format recognized by your version of Excel, including XLS, XLSX, XLSB, XLSM, and HTML-based formats.
- \* If your system does not have Excel for Windows, **xlsread** operates in BASIC mode (see Input Arguments).
- \* **xlsread** imports formatted dates as character vectors (such as '10/31/96'), except in BASIC mode. In BASIC mode, **xlsread** imports all dates as serial date

numbers. Serial date numbers in Excel use different reference dates than date numbers in MATLAB. For information on converting dates, see the documentation on importing spreadsheets.

```

*** xlsread is not recommended ***

```

#### Compatibility Considerations:

If the spreadsheet columns are mixed types, but has homogenous types along the columns, use READTABLE.

For reading numeric data, replace the following:

```
NUM = xlsread(FILE);
```

Instead Use:

```
NUM = readmatrix(FILE);
```

For reading text data, replace the following:

```
[~,TXT] = xlsread(FILE);
```

Instead Use:

```
TXT = readmatrix(FILE,"OutputType","char");
```

When reading RAW data, replace the following:

```
[~,~,RAW] = xlsread(FILE);
```

Instead Use:

```
RAW = readcell(FILE);
```

For specifying sheet and range, replace the following:

```
... = xlsread(FILE,SHEET,RANGE);
```

Instead use:

```
... = read<type>(FILE,"Sheet",SHEET,"RANGE",RANGE);
```

See also readtable, readmatrix, readcell, detectImportOptions

Documentation for xlsread

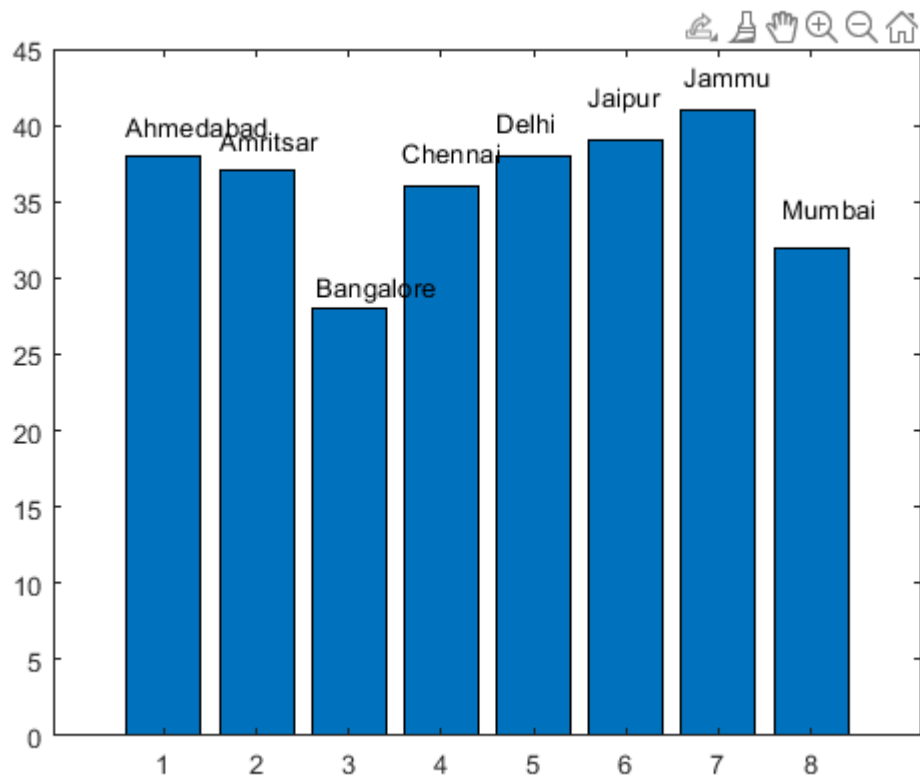
```
data = readtable("Lab_6_1.xlsx");
bar(data.Max)

cont =
char('Ahmedabad','Amritsar','Bangalore','Chennai','Delhi','Jaipur','Jammu',
 'Mumbai');

for i = 1:8,
 gtext(cont(i,:));

end
```





**Hence Jammu has recorded the Max Temperature of 41 Degrees**

3.

a. Create an excel file with the following data:

| Roll No | Name     | Accountancy | English | Maths | Economics | Business Studies |
|---------|----------|-------------|---------|-------|-----------|------------------|
| 1       | Akhilesh | 97          | 36      | 47    | 13        | 34               |
| 2       | Ruchi    | 69          | 85      | 86    | 51        | 53               |
| 3       | Bhawna   | 19          | 72      | 41    | 53        | 40               |
| 4       | Isha     | 76          | 68      | 46    | 11        | 22               |
| 5       | Chetan   | 55          | 31      | 56    | 99        | 93               |
| 6       | Neeti    | 84          | 57      | 68    | 30        | 31               |
| 7       | Chanchal | 18          | 46      | 51    | 63        | 22               |
| 8       | Preeti   | 93          | 93      | 31    | 93        | 20               |
| 9       | Richa    | 33          | 89      | 55    | 46        | 69               |
| 10      | Manish   | 21          | 27      | 84    | 82        | 96               |
| 11      | Karun    | 13          | 48      | 27    | 26        | 38               |
| 12      | Madhur   | 85          | 74      | 26    | 53        | 84               |
| 13      | Nitesh   | 28          | 31      | 27    | 77        | 17               |

- Read the file in matlab using xlsread
- Create a function tot\_marks to calculate total marks of each Student.
- Create function avg to calculate average marks of each student.
- Plot the marks obtained in step c. using

```
data_1 = readtable("Lab_6_2.xlsx")
```

Warning: Column headers from the file were modified to make them valid MATLAB identifiers before creating variable names for the table. The original column headers are saved in the VariableDescriptions property.

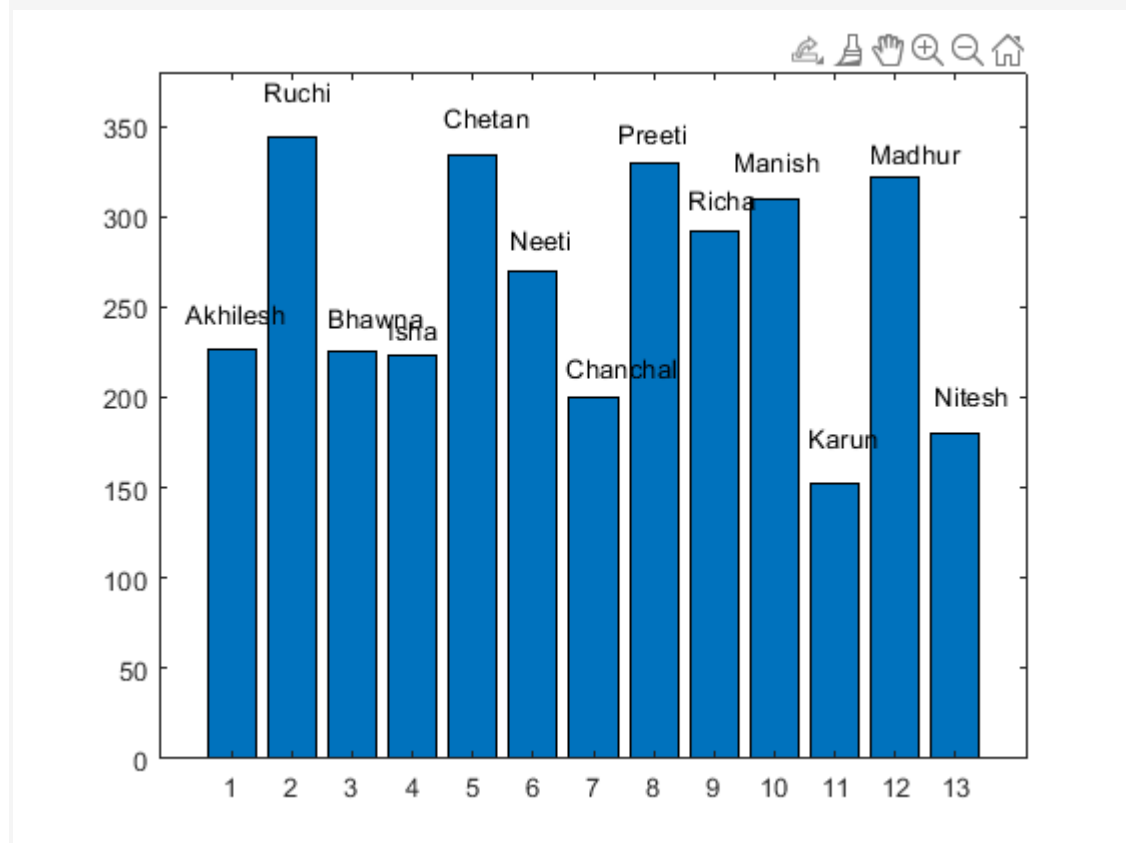
Set 'VariableNamingRule' to 'preserve' to use the original column headers as table variable names.

```
data_1 = 13x7 table
```

|    | Roll No | Name       | Accountancy | English | Maths | Economics | Business Studies |
|----|---------|------------|-------------|---------|-------|-----------|------------------|
| 1  | 1       | 'Akhilesh' | 97          | 36      | 47    | 13        | 34               |
| 2  | 2       | 'Ruchi'    | 69          | 85      | 86    | 51        | 53               |
| 3  | 3       | 'Bhawna'   | 19          | 72      | 41    | 53        | 40               |
| 4  | 4       | 'Isha'     | 76          | 68      | 46    | 11        | 22               |
| 5  | 5       | 'Chetan'   | 55          | 31      | 56    | 99        | 93               |
| 6  | 6       | 'Neeti'    | 84          | 57      | 68    | 30        | 31               |
| 7  | 7       | 'Chanchal' | 18          | 46      | 51    | 63        | 22               |
| 8  | 8       | 'Preeti'   | 93          | 93      | 31    | 93        | 20               |
| 9  | 9       | 'Richa'    | 33          | 89      | 55    | 46        | 69               |
| 10 | 10      | 'Manish'   | 21          | 27      | 84    | 82        | 96               |
| 11 | 11      | 'Karun'    | 13          | 48      | 27    | 26        | 38               |
| 12 | 12      | 'Madhur'   | 85          | 74      | 26    | 53        | 84               |
| 13 | 13      | 'Nitesh'   | 28          | 31      | 27    | 77        | 17               |

```
vars = ["Accountancy", "English", "Maths", "Economics", "BusinessStudies"];
data_1.TestMean = sum(data_1{:,vars},2);
bar(data_1.TestMean)
cont = char('Akhilesh', 'Ruchi', 'Bhawna', 'Isha', 'Chetan', ...
 'Neeti', 'Chanchal', 'Preeti', 'Richa', 'Manish', 'Karun', 'Madhur', 'Nitesh');
ylim([0 380])
for i = 1:13,
 gtext(cont(i,:));
```

end



```
data_1.Average = mean(data_1[:,vars],2)
```

```
data_1 = 13x9 table
```

|   | Roll No | Name       | Accountancy | English | Maths | Economics | Business Studies | Test Mean | Average |
|---|---------|------------|-------------|---------|-------|-----------|------------------|-----------|---------|
| 1 | 1       | 'Akhilesh' | 97          | 36      | 47    | 13        | 34               | 227       | 45.4000 |
| 2 | 2       | 'Ruchi'    | 69          | 85      | 86    | 51        | 53               | 344       | 68.8000 |
| 3 | 3       | 'Bhawna'   | 19          | 72      | 41    | 53        | 40               | 225       | 45      |
| 4 | 4       | 'Ishfa'    | 76          | 68      | 46    | 11        | 22               | 223       | 44.6000 |
| 5 | 5       | 'Chetan'   | 55          | 31      | 56    | 99        | 93               | 334       | 66.8000 |
| 6 | 6       | 'Neeti'    | 84          | 57      | 68    | 30        | 31               | 270       | 54      |
| 7 | 7       | 'Chanchal' | 18          | 46      | 51    | 63        | 22               | 200       | 40      |
| 8 | 8       | 'Preeti'   | 93          | 93      | 31    | 93        | 20               | 330       | 66      |

|        |    |              |    |    |    |    |    |     |             |
|--------|----|--------------|----|----|----|----|----|-----|-------------|
| 9      | 9  | 'Rich<br>a'  | 33 | 89 | 55 | 46 | 69 | 292 | 58.4<br>000 |
| 1<br>0 | 10 | 'Mani<br>sh' | 21 | 27 | 84 | 82 | 96 | 310 | 62          |
| 1<br>1 | 11 | 'Karu<br>n'  | 13 | 48 | 27 | 26 | 38 | 152 | 30.4<br>000 |
| 1<br>2 | 12 | 'Mad<br>hur' | 85 | 74 | 26 | 53 | 84 | 322 | 64.4<br>000 |
| 1<br>3 | 13 | 'Nites<br>h' | 28 | 31 | 27 | 77 | 17 | 180 | 36          |

|                                |                       |
|--------------------------------|-----------------------|
| Roll No.: A016                 | Name: Varun Khadayate |
| Class: B. Tech CsBs            | Batch: 1              |
| Date of Experiment: 18-09-2022 | Subject: ITWS         |

## Lab 7: M-script file-extended

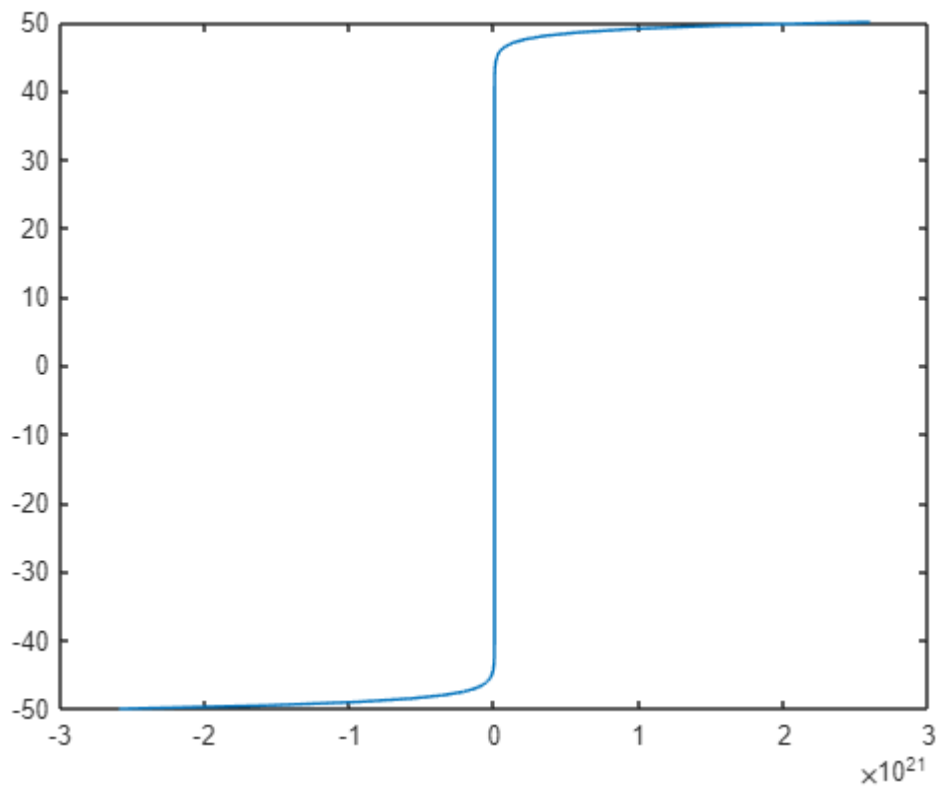
1.

Write three MATLAB functions to calculate the hyperbolic sine, cosine, and tangent functions:

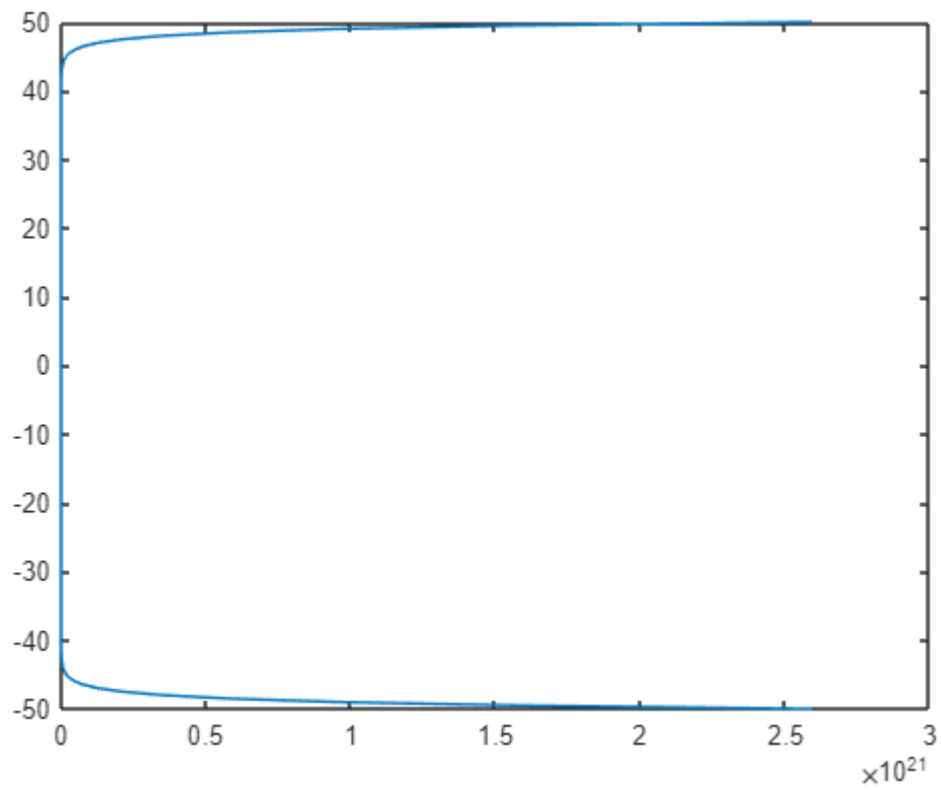
$$\sinh(x) = \frac{e^x - e^{-x}}{2} \quad \cosh(x) = \frac{e^x + e^{-x}}{2} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Use your functions to plot the shapes of the hyperbolic sine, cosine, and tangent functions.

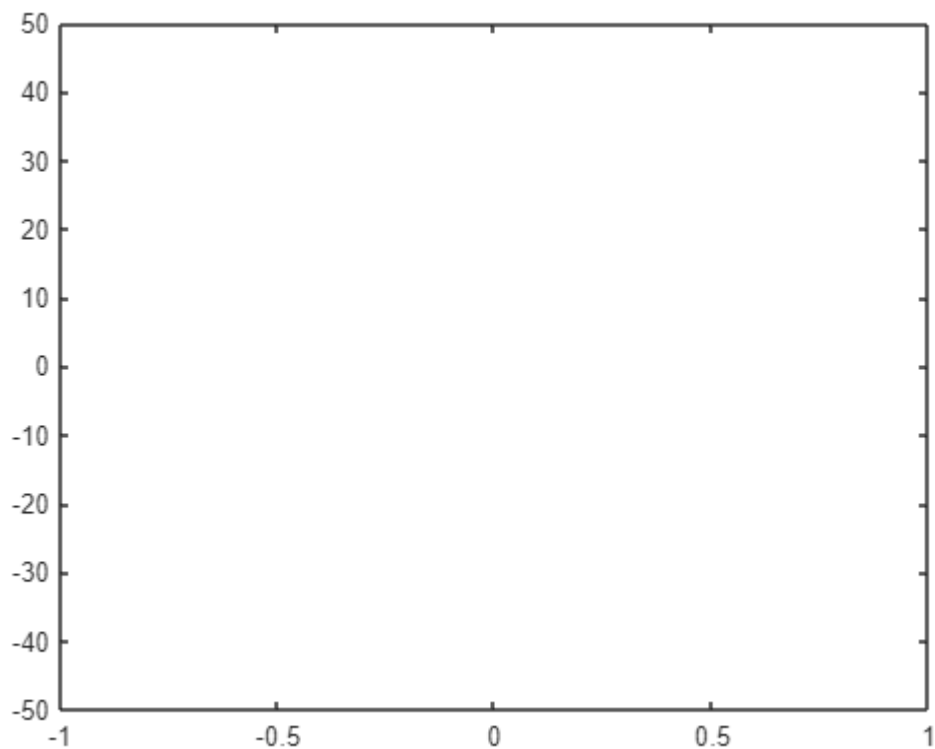
```
x = linspace(-50,50,200) ;
y_sinh = (exp(x) - exp(-x))/2;
figure;
plot(y_sinh,x);
```



```
x = linspace(-50,50,200) ;
y_cosh = (exp(x) + exp(-x))/2;
figure;
plot(y_cosh,x);
```



```
x = linspace(-50,50,200) ;
y_tanh = (exp(x) - exp(-x))/(exp(x) + exp(-x));
figure;
plot(y_tanh,x);
```



2.

The gravitational force  $F$  between two bodies of masses and is given by the equation

$$F = \frac{Gm_1m_2}{r^2}$$

where  $G$  is the gravitation constant ( $6.672 \times 10^{11} \text{ N m}^2 / \text{kg}^2$ ), and are the masses of the bodies in kilograms, and  $r$  is the distance between the two bodies. Write a function to calculate the gravitational force between two bodies given their masses and the distance between them. Test you function by determining the force on an 800 kg satellite in orbit 38,000 km above the Earth. (The mass of the Earth is  $5.98 \times 10^{24} \text{ kg}$ .)<sup>3</sup>

```
m1 = 800;
m2 = 5.98E24;
r = 38000;
G = 6.672E-11;
force = G * m1 * m2 / r^2
```

**Output**

```
force = 2.2104e+08
```

3.

- a) Write a program that will solve for the roots of a quadratic equation and display the answer on the screen. The inputs required by this program are the coefficients a,b and c (to be taken from the user ) of the quadratic equation ( $ax^2+bx+c = 0$ )

```
a = input ('Enter the coefficient A: ');
b = input ('Enter the coefficient B: ');
c = input ('Enter the coefficient C: ');
d = b^2 - 4 * a * c;
x1 = (-b + sqrt(d)) / (2 * a);
x2 = (-b - sqrt(d)) / (2 * a);
fprintf(['The coefficient A: %d \n' ...
 'The coefficient B: %d\n'...
 'the coefficient C: %d\n'...
 'The roots of this equation (%d)x^2 + (%d)x + (%d) are: %d and %d'],a,b,c,a,b,c,x1,x2);
```

- b) Test the program for :

i.  $x^2+5x+6 = 0$

```
The coefficient A: 1
The coefficient B: 5
the coefficient C: 6
The roots of this equation (1)x^2 + (5)x + (6) are: -2 and -3
```

ii.  $x^2+2x+5 = 0$

```
The coefficient A: 1
The coefficient B: 2
the coefficient C: 5
The roots of this equation (1)x^2 + (2)x + (5) are: -1 and -1
```



|                                |                       |
|--------------------------------|-----------------------|
| Roll No.: A016                 | Name: Varun Khadayate |
| Class: B. Tech CsBs            | Batch: 1              |
| Date of Experiment: 20-09-2022 | Subject: ITWS         |

## Lab 8

### CircleIO.m

```
% This script calculates the area of a circle
% It prompts the user for the radius

%Prompt the user for the radius and calculate the area based on that radius
fprintf('Note: the units will be inches. \n')
radius = input('Please enter the radius: ');
area = pi * (radius ^2) ;

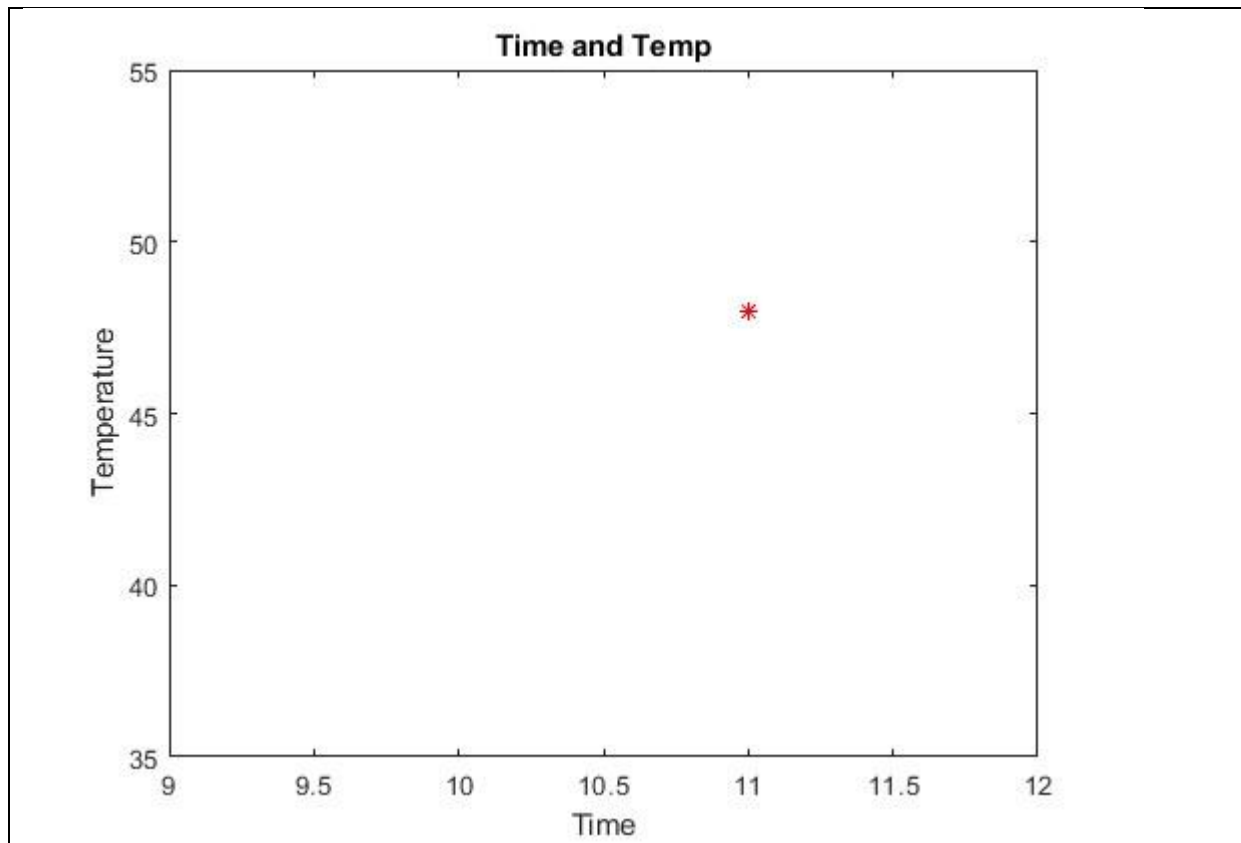
%Print all variables in a sentence format
fprintf('For a circle with a radius of %.2f inches, \n' ,radius)
fprintf('the area is %.2f inches squared \n' , area)
```

```
>> CircleIO
Note: the units will be inches.
Please enter the radius: 5
For a circle with a radius of 5.00 inches,
the area is 78.54 inches squared
```

### plotonepoint.m

```
% This is a really simple plot of just one point !
% Create coordinate variables and plot a red ' *
x = 11;
y = 48;

plot(x, y, 'r*')
% Change the axes and label them
axis([9 12 35 55])
xlabel('Time')
ylabel('Temperature')
% Put a title on the plot
title('Time and Temp')
```



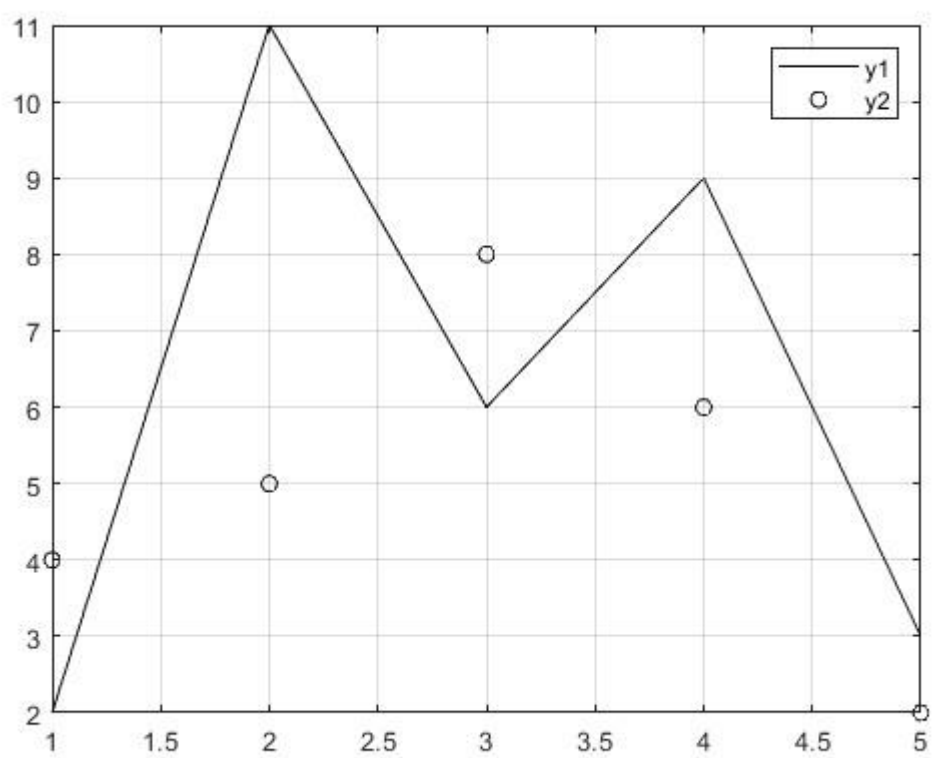
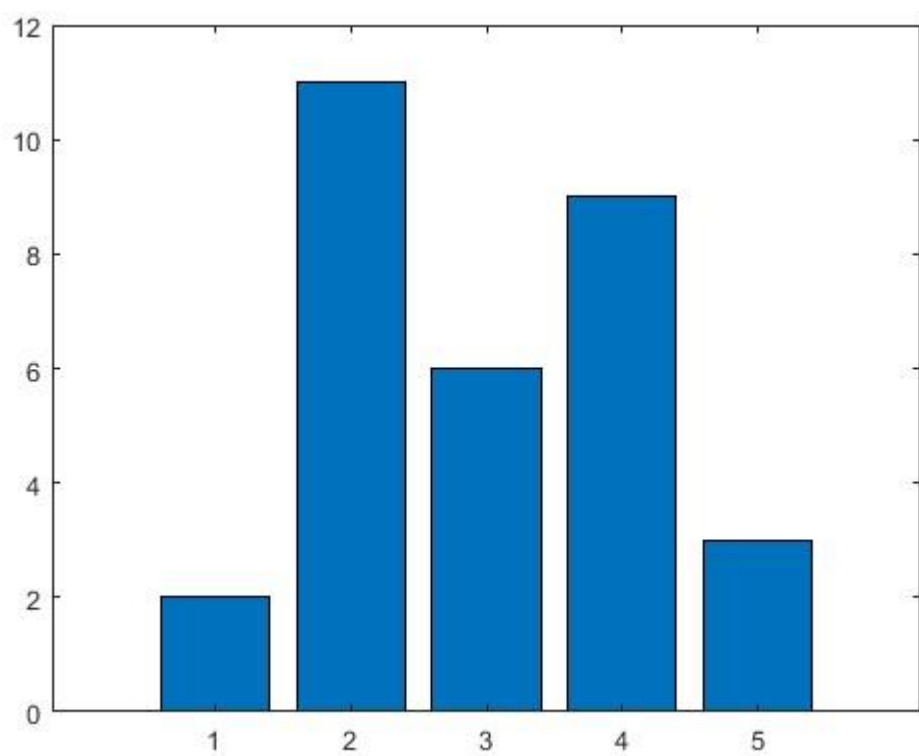
### plot2figs.m

```
% This creates 2 different plots, in 2 different
% Figure Windows, to demonstrate some plot features
clf

x = 1:5; % Not necessary
y1 = [2 11 6 9 3] ;
y2 = [4 5 8 6 2] ;
% Put a bar chart in Figure 1

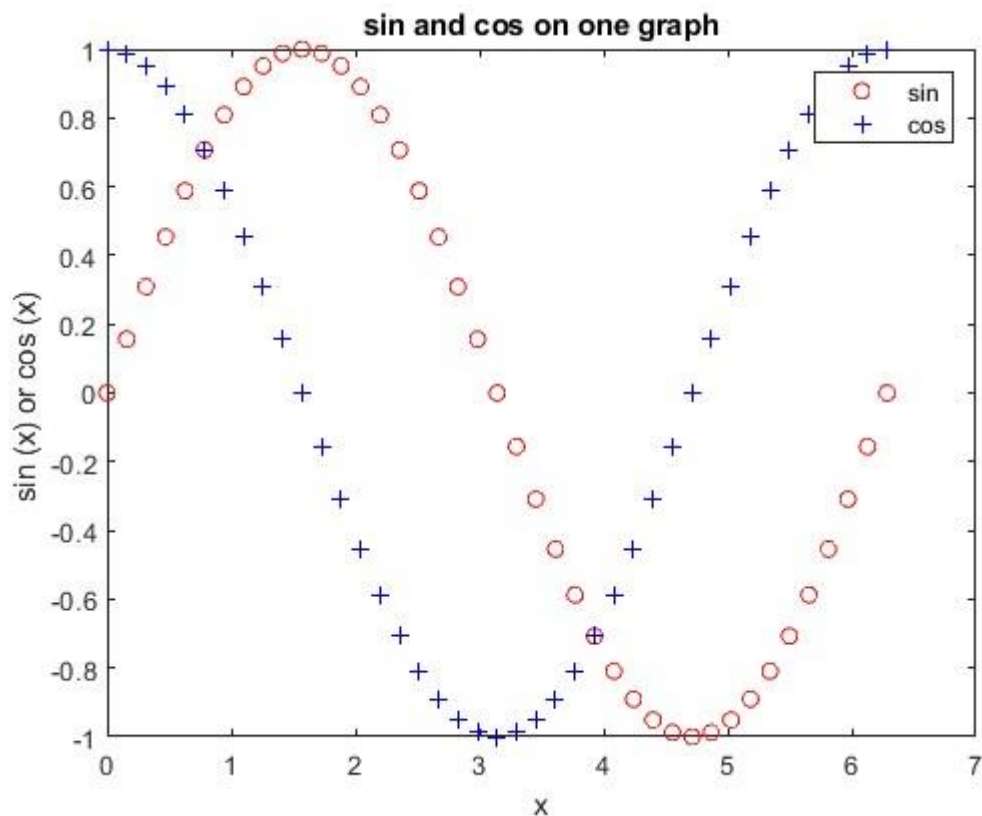
figure (1)
bar (x, y1)
% Put plots using different y values on one plot with a legend
figure (2)

plot (x, y1, 'k')
hold on
plot (x, y2, 'ko')
grid on
legend ('y1' , 'y2')
```



## sinncos.m

```
% takes the character and the number and prints them
a=input('Enter a character--->','s');
n=input('Enter a number--->');
% display the character in a field width of 3.
fprintf('%3c',a);
fprintf('\n')
% display the left-justified number in a field width of 8 with 3 decimal places.
fprintf('%-8.3f\n',n);
```



## Practice 3.1

Write a script to calculate the circumference of a circle ( $C = 2\pi r$ ). Comment the script.

```
% This script calculates the circumference of a circle
% It prompts the user for the radius

%Prompt the user for the radius and calculate the area based on that radius
fprintf('Note: the units will be inches. \n')
radius = input('Please enter the radius: ');
area = 2 * pi * radius ;

%Print all variables in a sentence format
fprintf('For a circle with a radius of %.2f inches, \n' ,radius)
fprintf('the Circle is %.2f inches \n' , area)
```

```
>> practice3_1
Note: the units will be inches.
Please enter the radius: 5
For a circle with a radius of 5.00 inches,
the Circle is 31.42 inches
```

## Practice 3.2

Create a script that would prompt the user for a length, and then 'f' for feet or 'm' for meters, and store both inputs in variables. For example, when executed it would look like this (assuming the user enters 12.3 and then m):

```
Enter the length: 12.3
Is that f(eet) or m(eters)? : m
```

```
% This script records the length and the unit of that length
% It prompts the user for the radius

% Prompts the user for values and stores them
length=input ('Enter the value for length: ') ;
units=input ('Enter the units for length (f or m) : ', 's');
>> practice3_2
Enter the value for length: 12
Enter the units for length (f or m) : m
```

## Practice 3.3

Write a script to prompt the user separately for a character and a number, and print the character in a field width of 3 and the number left-justified in a field width of 8 with 3 decimal places. Test this by entering numbers with varying widths.

```
% takes the character and the number and prints them
a=input('Enter a character--->','s');
n=input('Enter a number--->');
% display the character in a field width of 3.
fprintf('%3c',a);
fprintf('\n')
% display the left-justified number in a field width of 8 with 3 decimal places.
fprintf('%-8.3f\n',n);

>> practice3_3
Enter a character--->varun
Enter a number--->123455678.123344556789
 v a r u n
123455678.123
```

## Practice 3.4

Modify the script *plotonepoint* to prompt the user for the time and temperature, and set the axes based on these values.

```
% This script plots one point (time, temp) from data
% acquired from the user.
% Ask the user for the data to plot and store in the
% variables time and temp
time = input('Enter the time in hours: ');

temp = input('Enter the temperature in degrees C: ');
plot(time,temp,"*")

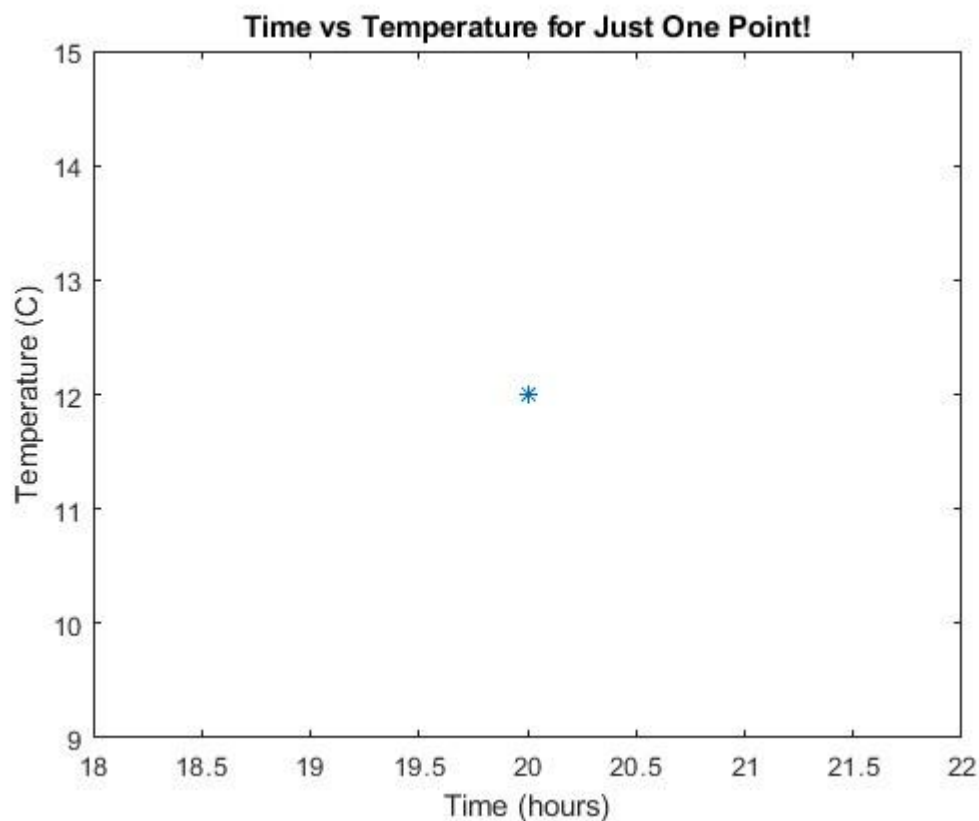
% Adjust the x and y axis limits

axis([time-2 time+2 temp-3 temp+3]);

% Label the axis
xlabel("Time (hours)")

ylabel("Temperature (C)")
% Add a title
title("Time vs Temperature for Just One Point!")
```

```
>> practice3_4
Enter the time in hours: 12
Enter the temperature in degrees C: 50
```

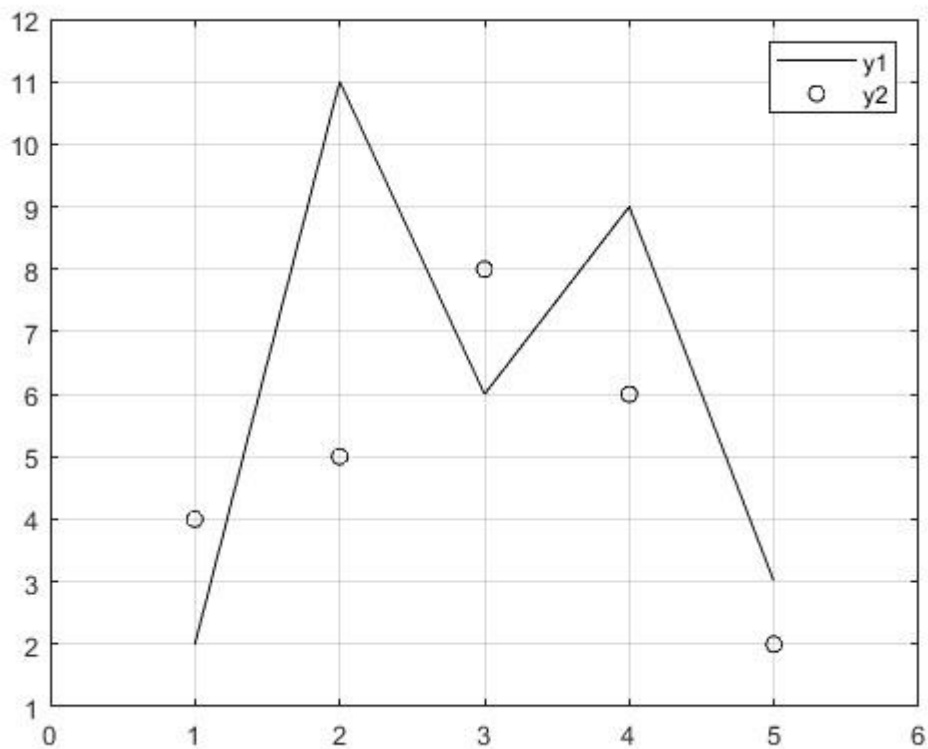


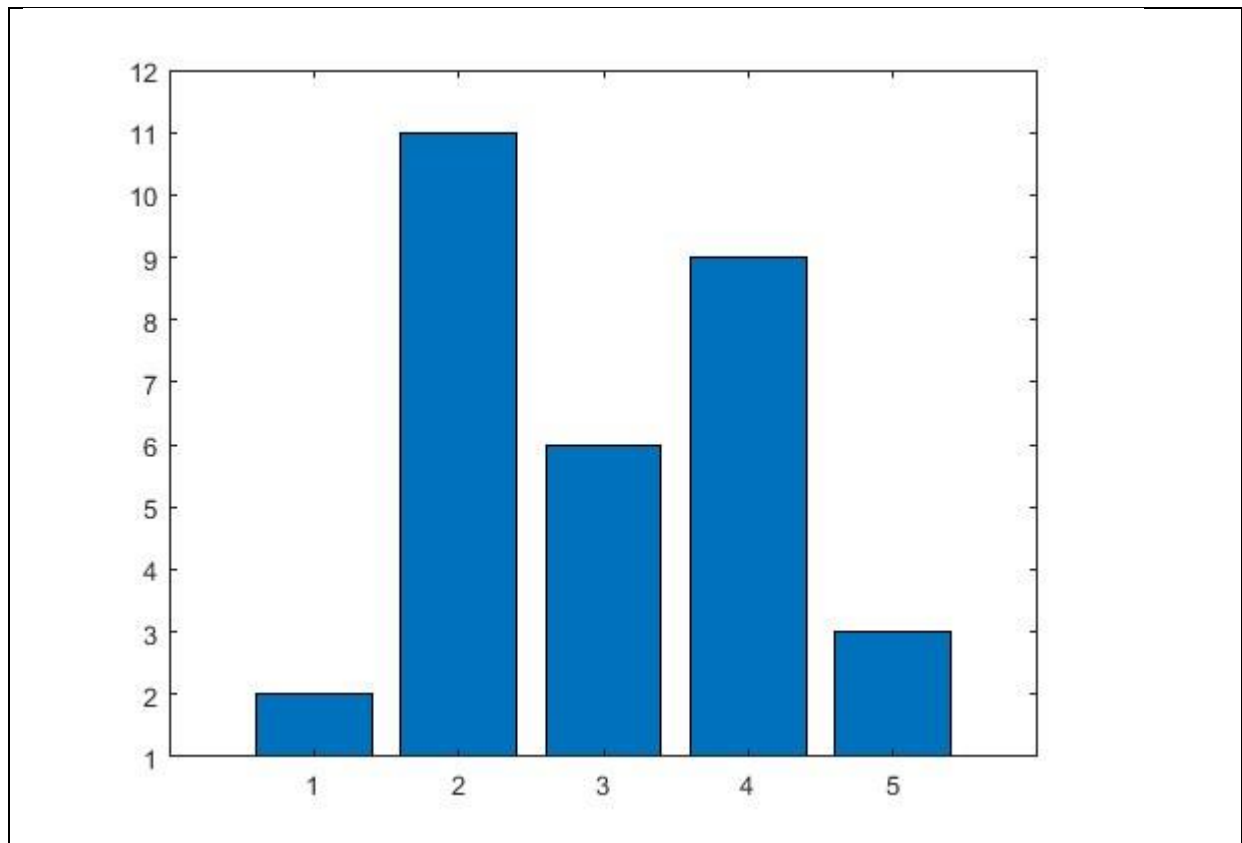
## Practice3.5

Modify the *plot2figs* script using the **axis** function so that all points are easily seen.

```
% creates two different plots
x=1:5;
y1=[2 11 6 9 3];

y2=[4 5 8 6 2];
%Put a bar chart in Figure 1
figure(1)
bar(x,y1)
%Change the axis settings
axis([0 6 1 12]);
%Put plots using different y values on one plot with a legend
figure(2)
plot(x,y1,"k")
hold on
plot(x,y2,'ko')
grid on
legend('y1', 'y2')
%Change the axis settings
axis([0 6 1 12]);
```



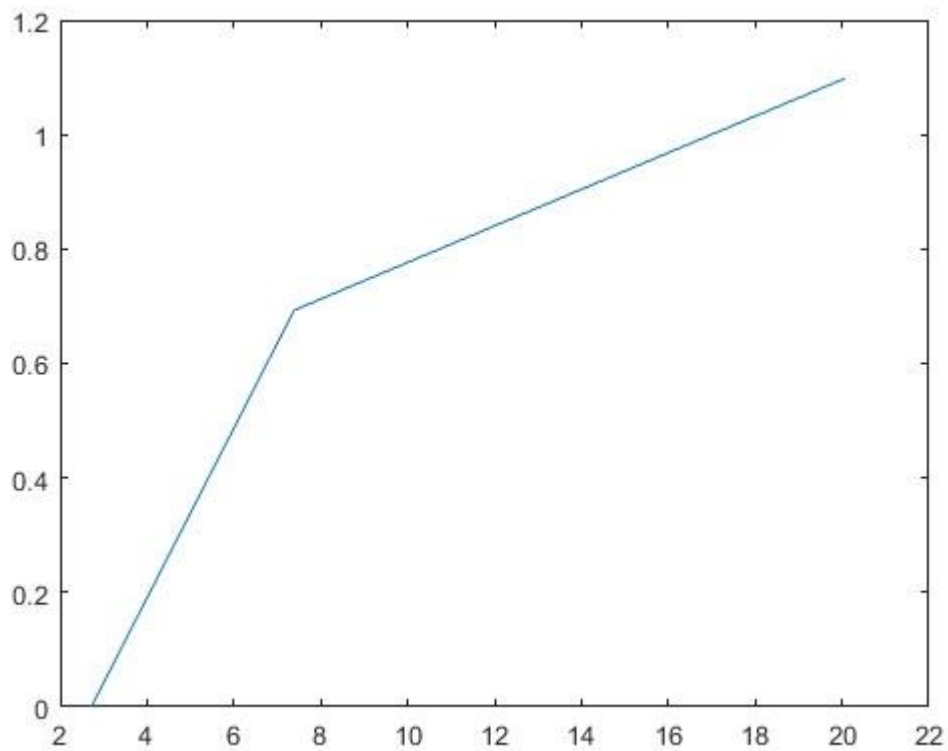


### Practice 3.6

Write a script that plots  $\exp(x)$  and  $\log(x)$  for values of  $x$  ranging from 0 to 3.5.

```
x = 0:3.5;
plot(exp(x),log(x))
```





### Practice 3.7

Prompt the user for the number of rows and columns of a matrix, create a matrix with that many rows and columns of random integers, and write it to a file.

```
% This script asks for user input for the number of rows
% and columns for a matrix. Then a matrix of that size is
% filled with random numbers and save in ASCII format to a
% file.
% Ask user for input
row = input('Enter the number of matrix rows: ');
col = input('Enter the number of matrix columns: ');

% Create the matrix of random numbers of specified size.
mat = randi(25, row,col)
% Save the matrix as an ASCII file.
save myMatrix.dat mat -ascii
```

```
>> practice3_7
Enter the number of matrix rows: 5
Enter the number of matrix columns: 5

mat =

 11 1 19 1 18
 23 22 10 7 8
 20 24 17 2 24
 24 17 5 3 1
 17 19 18 21 11
```

## Practice 3.8

The sales (in billions) for two separate divisions of the ABC Corporation for each of the four quarters of 2013 are stored in a file called "salesfigs.dat":

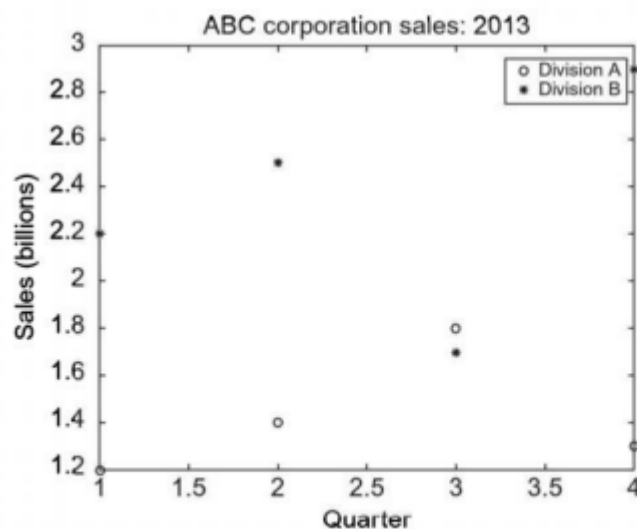
```
1.2 1.4 1.8 1.3
2.2 2.5 1.7 2.9
```

- First, create this file (just type the numbers in the Editor, and Save As "salesfigs.dat").
- Then, write a script that will

load the data from the file into a matrix

separate this matrix into 2 vectors.

create the plot seen in Fig. 3.7 (which uses black circles and stars as the plot symbols).



```
row = [1.2 1.4 1.8 1.3];
col = [2.2 2.5 1.7 2.9];
mat = [row;col]
% Save the matrix as an ASCII file.
save salesfigs.dat mat -ascii
% Load .dat file
load salesfigs.dat
x=salesfigs(1,:)
y=salesfigs(2,:)
```

```

plot(1:numel(x),x,'o')
hold on
plot(1:numel(y),y,',' , 'MarkerSize',20)
ylabel('Sales(billion)')
title('ABC corporation Sales:2013')
legend('Division A','Division B')
hold off

```

```

>> practice3_8

mat =

 1.2000 1.4000 1.8000 1.3000
 2.2000 2.5000 1.7000 2.9000

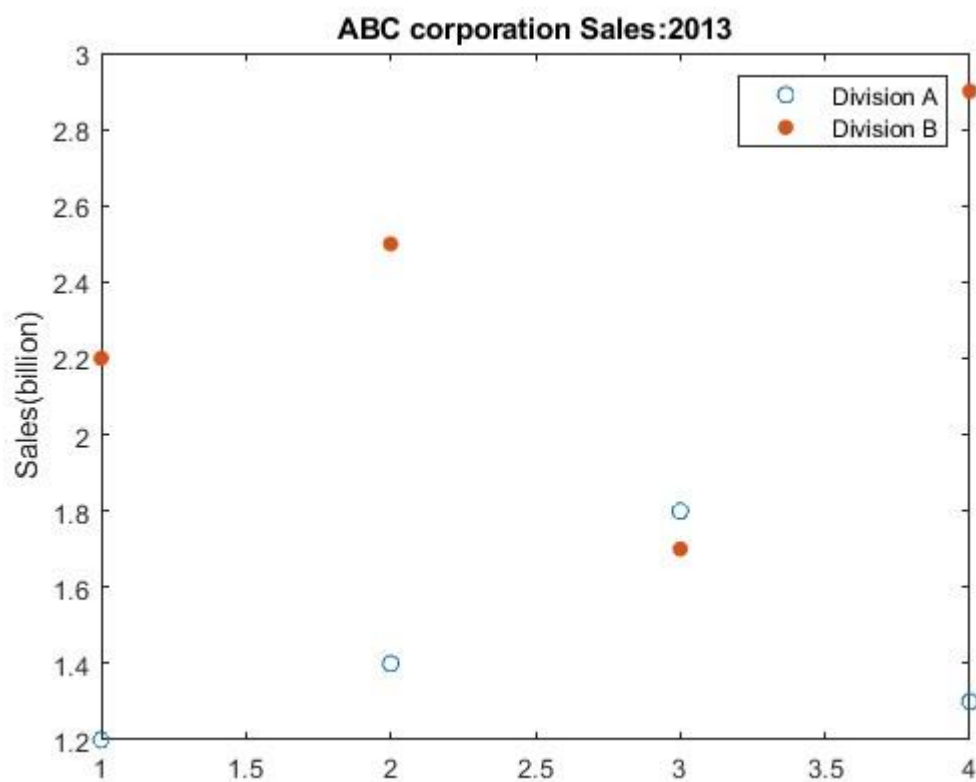
x =

 1.2000 1.4000 1.8000 1.3000

y =

 2.2000 2.5000 1.7000 2.9000

```



## Practice 3.9

Write a script that will prompt the user for the radius and height, call the function *conevol* to calculate the cone volume, and print the result in a nice sentence format. So, the program will consist of a script and the *conevol* function that it calls.

```
function [V] = conevol(r,h)
% CONEVOL finds the volume of a cone
% Format of call: conevol(r,h)

% Returns cone volume
V = pi * r^2 * h/3;
end
```

```
>> conevol(4,5)

ans =

 83.7758
```

## Practice 3.10

For a project, we need some material to form a rectangle. Write a function *calcrectarea* that will receive the length and width of a rectangle in inches as input arguments, and will return the area of the rectangle. For example, the function could be called as shown, in which the result is stored in a variable and then the amount of material required is printed, rounded up to the nearest square inch.

```
>> ra = calcrectarea(3.1, 4.4)
ra =
 13.6400

>> fprintf('We need %d sq in.\n', ceil(ra))
We need 14 sq in.
```

```
function [A] = calcrectarea(L,W)
% CALRECTAREA Finds the area of a rectangle rounded up to
% the nearest integer.
% Format of call: calRectArea(L,W)
% Returns rectangle area rounded up to the nearest integer
% Compute the area and round the result up to the nearest integer.
A = ceil(L * W);
end
```

```
>> calcrectarea(4,4)

ans =

 16
```

## Lab 9: If-Else statement

1. Whether a storm is a tropical depression, tropical storm, or hurricane is determined by the average sustained wind speed. In miles per hour, a storm is a tropical depression if the winds are less than 38 mph. It is a tropical storm if the winds are between 39 and 73 mph, and it is a hurricane if the wind speeds are  $\geq 74$  mph. Write a script that will prompt the user for the wind speed of the storm, and will print which type of storm it is.

```
% Prints whether a storm is a tropical depression,
% tropical storm, or hurricane based on wind speed

wind = input('Enter the wind speed of the storm: ');
if wind < 38
 disp('Tropical depression')
elseif wind >= 38 && wind < 73
 disp('Tropical storm')
else
 disp('Hurricane')
end
```

Tropical storm

2. In aerodynamics, the Mach number is a critical quantity. It is defined as the ratio of the speed of an object (e.g., an aircraft) to the speed of sound. If the Mach number is less than 1, the flow is subsonic; if the Mach number is equal to 1, the flow is transonic; if the Mach number is greater than 1, the flow is supersonic. Write a script that will prompt the user for the speed of an aircraft and the speed of sound at the aircraft's current altitude and will print whether the condition is subsonic, transonic, or supersonic.

```
% Prints whether the speed of an object is subsonic,
% transonic, or supersonic based on the Mach number

plane_speed = input('Enter the speed of the aircraft: ');
sound_speed = input('Enter the speed of sound: ');
mach = plane_speed/sound_speed;

if mach < 1
 disp('Subsonic')
elseif mach == 1
 disp('Transonic')
else
 disp('Supersonic')
end
```

Supersonic

3. In a script, the user is supposed to enter either a 'y' or 'n' in response to a prompt. The user's input is read into a character variable called "letter". The script will print "OK, continuing" if the user enters either a 'y' or 'Y' or it will print "OK, halting" if the user enters a 'n' or 'N' or "Error" if the user enters anything else. Put this statement in the script first:

```
letter = input('Enter your answer: ', 's');
```

Write the script using a single nested if-else statement (elseif clause is permitted).

```
% Prompts the user for a 'y' or 'n' answer and responds
% accordingly, using an if-else statement
letter = input('Enter your answer: ', 's');
if letter == 'y' || letter == 'Y'
 disp('OK, continuing')
elseif letter == 'n' || letter == 'N'
 disp('OK, halting')
else
 disp('Error')
end
```

OK, continuing

4. Write a function *flipvec* that will receive one input argument. If the input argument is a row vector, the function will reverse the order and return a new row vector. If the input argument is a column vector, the function will reverse the order and return a new column vector. If the input argument is a matrix or a scalar, the function will return the input argument unchanged.

```
% function out = flipvec(vec)
% % Flips it if it's a vector, otherwise
% % returns the input argument unchanged
% % Format of call: flipvec(vec)
% % Returns flipped vector or unchanged
% [r, c] = size(vec);
% if r == 1 && c > 1
% out = fliplr(vec);
% elseif c == 1 && r > 1
% out = flipud(vec);
% else
% out = vec;
% end
% end

% flipvec([1 2 3 4 5])
%
% ans =
%
```

```
% 5 4 3 2 1
%
% flipvec([1; 2; 3; 4; 5])
%
% ans =
%
% 5
% 4
% 3
% 2
% 1
```

5. Write a function *eqfn* that will calculate  $f(x) = x^2 + \frac{1}{x}$  for all elements of *x*. Since division by 0 is not possible, if any element in *x* is zero, the function will instead return a flag of -99.

```
% function fofx = eqfn(x)
% if any(any(x==0))
% fofx = -99;
% else
% fofx = x.^2 + 1./x;
% end
% end

% eqfn(5)
%
% ans =
%
% 25.2000
```

### for-end loop

6. In the Command Window, write a for loop that will iterate through the integers from 32 to 255. For each, show the corresponding character from the character encoding. Play with this! Try printing characters beyond the standard ASCII, in small groups. For example, print the characters that correspond to integers from 300 to 340.

```
for i = 32:255
 disp(char(i))
end
```

```
!
"
#
$
%
```

&  
,  
(  
)  
\*  
+  
,  
-  
.  
/  
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
:  
;  
<  
=  
>  
?  
@  
A  
B  
C  
D  
E  
F  
G  
H  
I  
J  
K  
L  
M  
N  
O  
P  
Q  
R  
S  
T  
U  
V  
W  
X  
Y  
Z  
[  
\  
]  
^  
\_  
a  
b  
c  
d  
e



f  
g  
h  
i  
j  
k  
l  
m  
n  
o  
p  
q  
r  
s  
t  
u  
v  
w  
x  
y  
z  
{  
|  
}  
~

i  
¢  
£  
¤  
¥

¡  
§  
..  
©  
a  
«  
¬  
®  
-  
o  
±  
2  
3  
/  
µ  
¶  
.  
,  
1  
o  
»  
¼  
½  
¾  
¿  
À  
Á  
Â  
Ã  
Ä  
Å  
Æ  
Ç  
È  
É  
Ê  
Ë  
Ì  
Í  
Î  
Ï  
Ð  
Ñ  
Ò  
Ó  
Ô  
Õ  
Ö  
×  
Ø  
Ù  
Ú  
Û  
Ü  
Ý  
Þ  
ß  
à  
á  
â  
ã  
ä

æ  
ç  
è  
é  
ê  
ë  
ì  
í  
î  
ï  
ö  
ñ  
ò  
ó  
ô  
õ  
ö  
÷  
ø  
ù  
ú  
û  
ü  
ý  
þ  
ÿ

**7. Write a function *sumsteps2* that calculates and returns the sum of 1 to *n* in steps of 2, where *n* is an argument passed to the function. For example, if 11 is passed, it will return 1 + 3 + 5 + 7 + 9 + 11. Do this using a for loop. Calling the function will look like this:**

```
>> sumsteps2(11)
```

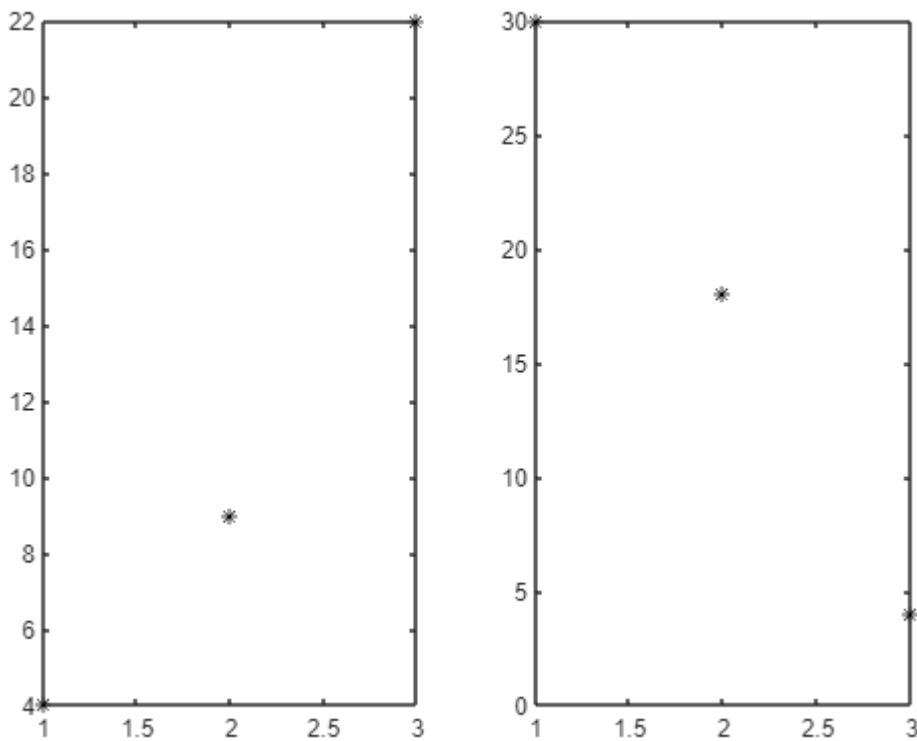
```
ans = 36
```

```
% function outsum = sumsteps2(n)
% % sum from 1 to n in steps of 2
% % Format of call: sumsteps2(n)
% % Returns 1 + 3 + ... + n
%
% outsum = 0;
% for i = 1:2:n
% outsum = outsum + i;
% end
% end

% sumsteps2(18)
%
% ans =
%
% 81
```

8. Write a script that will load data from a file into a matrix. Create the data file first, and make sure that there is the same number of values on every line in the file so that it can be loaded into a matrix. Using a for loop, it will then create a subplot for every row in the matrix, and will plot the numbers from each row element in the Figure Window.

```
row1 = [4 9 22];
row2 = [30 18 4];
mat = [row1;row2];
% Save the matrix as an ASCII file.
save xfile.dat mat -ascii
% load data from a file and plot data
% from each line in a separate plot in a subplot
load xfile.dat
[r, c] = size(xfile);
for i = 1:r
 subplot(1,r,i)
 plot(xfile(i,:), 'k*')
end
```



9. Write a script that will print the following multiplication table:

1

2 4

3 6 9

4 8 12 16

5 10 15 20 25

```
% Prints a multiplication table
rows = 5;
for i = 1:rows
% for j = 1:i
% fprintf('%d ', i*j)
% end
% fprintf('\n')
 linspace(i,i*i,i)
end
```

```
ans = 1
ans = 1x2
 2 4
ans = 1x3
 3 6 9
ans = 1x4
 4 8 12 16
ans = 1x5
 5 10 15 20 25
```

**10. A machine cuts N pieces of a pipe. After each cut, each piece of pipe is weighed and its length is measured; these 2 values are then stored in a file called *pipe.dat* (first the weight and then the length on each line of the file). Ignoring units, the weight is supposed to be between 2.1 and 2.3, inclusive, and the length is supposed to be between 10.3 and 10.4, inclusive. The following is just the beginning of what will be a long script to work with these data. For now, the script will just count how many rejects there are. A reject is any piece of pipe that has an invalid weight and/or length. For a simple example, if N is 3 (meaning three lines in the file) and the file stores:**

2.14 10.30

2.32 10.36

2.20 10.35

**there is only one reject, the second one, as it weighs too much. The script would print: There were 1 rejects.**

```
% Counts pipe rejects. Ignoring units, each pipe should be
% between 2.1 and 2.3 in weight and between 10.3 and 10.4
% in length
row1 = [2.44 10.40];
row2 = [2.23 10.36];
```

```

row3 = [2.20 12.35];
mat = [row1;row2;row3];
% Save the matrix as an ASCII file.
save pipe.dat mat -ascii
% read the pipe data and separate into vectors
load pipe.dat
weights = pipe(:,1);
lengths = pipe(:,2);
N = length(weights);
% the programming method of counting
count = 0;
for i=1:N
 if weights(i) < 2.1 || weights(i) > 2.3 || lengths(i) < 10.3 || lengths(i) > 10.4
 count = count + 1;
 end
end
fprintf('There were %d rejects.\n', count)

```

There were 2 rejects.