Chapter-2

# Lesson 1:

**Code:**

% 1. Arithmatic operators Area program

radius = pi^(1/3) - 1

AREA = pi \* radius^2

% 2. Exp and log (Solve 3^x = 17 for x and check the result)

x = log(17) / log(3)

% 3. Trigo (y = cosh^2 x − sinh^2 x)

x = 32\*pi;

y = (cosh(x) )^2 - (sinh(x) )^2

% 4. Complex nos (Check the Euler’s Formula)

x = (pi/4)

res\_1 = exp(j\*x)

res\_2 = cos(x) + j\*sin(x)

if res\_1 == res\_2

disp('verified euler formula')

end

**Output:**

diary on

L2\_1\_PG\_23

radius =

0.4646

AREA =

0.6781

x =

2.5789

y =

0

x = 0.7854

res\_1 =

0.7071 + 0.7071i

res\_2 =

0.7071 + 0.7071i

verified euler formula

diary off

# Lesson 2:

**Code:**

disp('Equation of a straight line:')

    x = [0, 1.5, 3 ,4 ,5, 7, 9 , 10];

    m = 0.5;

    c = -2;

    y = m.\*x + c

    disp('Multiply, divide, and exponentiate vectors: ')

    t = 1:10;

    x = t.\*sin(t)

    y = (t-1) ./ (t+1)

    z = sin( t.^2  ) ./ t.^2

disp('Points on a circle')

    RADIUS = 2;

    theta = 0:(pi/4):(5\*pi/4) ;

    x = cos(theta)\*RADIUS

    y = RADIUS\*sin(theta)

    condition = sqrt(x.^2 + y.^2);

    if (condition == RADIUS)

         disp('Equation of circle is verified')

    end

disp('Geometric series')

    n = 0:1:10;

    r = 0.5;

    x = r.^n ;

    s = sum(x);

    result\_by\_formula = 1 / (1-r);

    deviation\_1 = s - result\_by\_formula

    n = 0:1:50;

    x = r.^n ;

    s = sum(x);

    deviation\_2 = s - result\_by\_formula

    n = 0:1:100;

    x = r.^n ;

    s = sum(x);

    deviation\_3 = s - result\_by\_formula

**Output:**

diary on

L2\_2\_PG\_27

Equation of a straight line:

y =

-2.0000 -1.2500 -0.5000 0 0.5000 1.5000 2.5000 3.0000

Multiply, divide, and exponentiate vectors:

x =

0.8415 1.8186 0.4234 -3.0272 -4.7946 -1.6765 4.5989 7.9149 3.7091 -5.4402

y =

0 0.3333 0.5000 0.6000 0.6667 0.7143 0.7500 0.7778 0.8000 0.8182

z =

0.8415 -0.1892 0.0458 -0.0180 -0.0053 -0.0275 -0.0195 0.0144 -0.0078 -0.0051

Points on a circle

x =

2.0000 1.4142 0.0000 -1.4142 -2.0000 -1.4142

y =

0 1.4142 2.0000 1.4142 0.0000 -1.4142

Equation of circle is verified

Geometric series

deviation\_1 =

-9.7656e-04

deviation\_2 =

-8.8818e-16

deviation\_3 =

0

diary off

# Lesson 3:

**Code:**

disp('An exponentially decaying sine curve')

n = input('10, 50 or 100 points? ');

switch n

case 10

    disp('using 10 points')

        theta = linspace(0, 4\*pi, 10);

        y\_sin = sin(theta);

        expo = exp( theta.\*(-0.4) ) ;

        y = expo.\*y\_sin;

        ylabel('Y')

        xlabel('theta')

        title('using 10 points ')

        plot(theta, y);

case 50

    disp('using 50 points')

        theta = linspace(0, 4\*pi, 50);

        y\_sin = sin(theta);

        expo = exp( theta.\*(-0.4) ) ;

        y = expo.\*y\_sin;

        ylabel('Y')

        xlabel('theta')

        title('using 50 points ')

        plot(theta, y);

case 100

    disp('using 100 points')

        theta = linspace(0, 4\*pi, 100);

        y\_sin = sin(theta);

        expo = exp( theta.\*(-0.4) ) ;

        y = expo.\*y\_sin;

        ylabel('Y')

        xlabel('theta')

        title('using 100 points ')

        plot(theta, y);

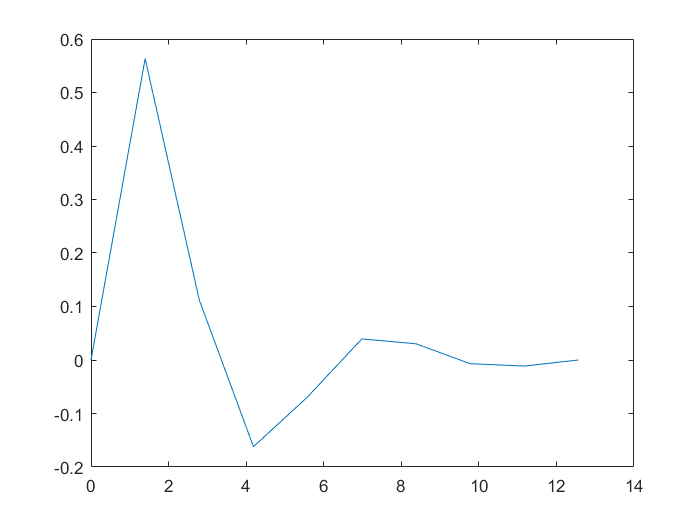
end

**Output:**

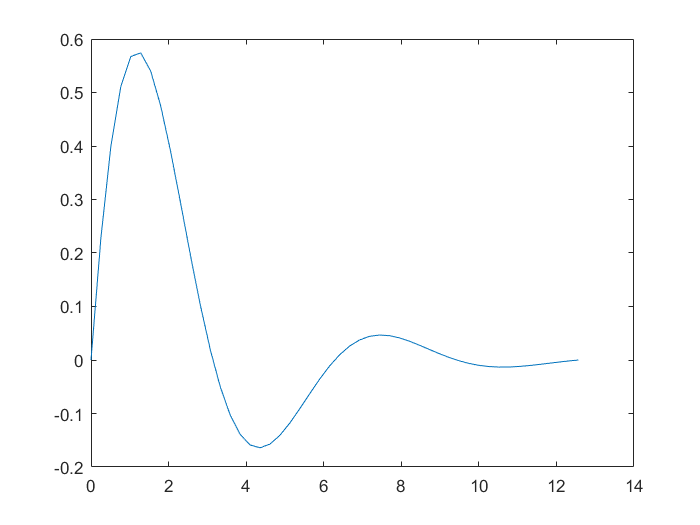
An exponentially decaying sine curve

10, 50 or 100 points?

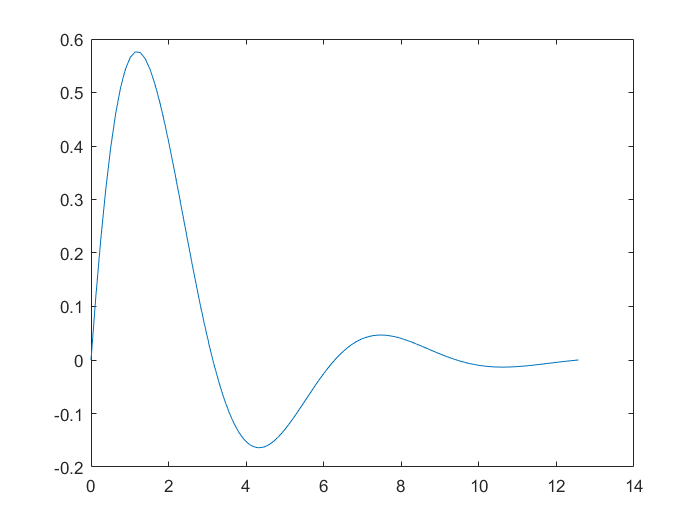
10



50



100



# Lesson 4

**Code:**

disp('CIRCLE - A script file to draw a unit circle')

    r = input(’Enter the radius of the circle: ’)

    theta = linspace(0,2\*pi,100); % create vector theta

    x = r\*cos(theta); % generate x-coordinates

    y = r\*sin(theta); % generate y-coordinates

    plot(x,y); % plot the circle

    axis(’equal’);

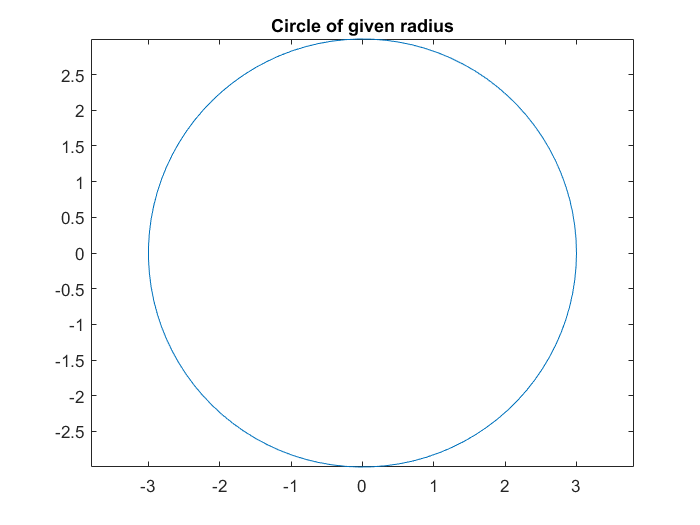
    title(’Circle of given radius’)

**Output:**

Enter the radius of the circle: 3

r =

3



# Lesson 5:

**Code:**

function [temperature] = my\_temp\_conv(Ti, Tf)

%Enter the lower and upper limit respectively

% and this function shall return a tabular conversion between those

%values in differences of 1

%

% my\_temp\_conv (float, float)

%

celcius = Ti:1:Tf ;

farhenite = (9/5)\*celcius + 32 ;

temperature = [celcius; farhenite]

**Output:**

my\_temp\_conv(15,45)

temperature =

ans =

Columns 1 through 16

15.0000 16.0000 17.0000 18.0000 19.0000 20.0000 21.0000 22.0000 23.0000 24.0000 25.0000 26.0000 27.0000 28.0000 29.0000 30.0000

59.0000 60.8000 62.6000 64.4000 66.2000 68.0000 69.8000 71.6000 73.4000 75.2000 77.0000 78.8000 80.6000 82.4000 84.2000 86.0000

Columns 17 through 31

31.0000 32.0000 33.0000 34.0000 35.0000 36.0000 37.0000 38.0000 39.0000 40.0000 41.0000 42.0000 43.0000 44.0000 45.0000

87.8000 89.6000 91.4000 93.2000 95.0000 96.8000 98.6000 100.4000 102.2000 104.0000 105.8000 107.6000 109.4000 111.2000 113.0000

# Lesson 6:

-Has no given exercise-

Chapter-3

**Q1. Entering matrices:** Enter the following three matrices. (check pdf for the asked matrices)

**Code:**

 A = [ [2,6]; [3,9] ] ;

 B = [ [1,2]; [3,4] ] ;

 C = [ [-5,5]; [5,3] ] ;

**Q2. Check some linear algebra rules:**

**Code:**

 disp('Is matrix addition commutative?')

    disp('A+B = ')

        A+B

    disp('B+A = ')

        B+A

 disp('Is matrix addition associative?')

    disp('(A + B) + C = ')

        (A + B) + C

    disp('A + (B + C) =')

        A + (B + C)

 disp('Matrices are different from scalars!')

    disp('A\*B = ')

        A\*B

    disp('A\*C = ')

        A\*C

    disp('But, B clearly is not equal to C!')

disp('matrix products do not commute either')

    disp('A\*B = ')

        A\*B

    disp('B\*A = ')

        B\*A

**Output:**

Is matrix addition commutative?

A+B =

ans =

3 8

6 13

B+A =

ans =

3 8

6 13

Is matrix addition associative?

(A + B) + C =

ans =

-2 13

11 16

A + (B + C) =

ans =

-2 13

11 16

Matrices are different from scalars!

A\*B =

ans =

20 28

30 42

A\*C =

ans =

20 28

30 42

But, B clearly is not equal to C!

matrix products do not commute either

A\*B =

20 28

30 42

B\*A =

8 24

18 54

Q3. **Create matrices with** zeros, eye, **and** ones**:**

**Code:**

D = zeros(2,3)

E = 3\*eye(3)

F = 3\*Ones(2)

Q4. **Create a big matrix with submatrices** G is created

by putting matrices A, B, and C given above, on its diagonal.

Code:

  A = [ [2,6]; [3,9] ] ;

 B = [ [1,2]; [3,4] ] ;

C = [[-5,5];[5,3]] ;

blkdiag(A,B,C)

Output:

ans =

2 6 0 0 0 0

3 9 0 0 0 0

0 0 1 2 0 0

0 0 3 4 0 0

0 0 0 0 -5 5

0 0 0 0 5 3

Q5. **Manipulate a matrix:**

**Code:**

G = blkdiag(A,B,C)

    disp('Delete the last row and last column of the matrix.')

        G(end, :) = []

    disp('Extract the first 4 × 4 submatrix from G.')

        G([1:4],[1:4])

Output:

Delete the last row and last column of the matrix.

G =

2 6 0 0 0 0

3 9 0 0 0 0

0 0 1 2 0 0

0 0 3 4 0 0

0 0 0 0 -5 5

Extract the first 4 × 4 submatrix from G.

ans =

2 6 0 0

3 9 0 0

0 0 1 2

0 0 3 4

Q7. **Create a symmetric matrix:** Create an upper triangular matrix

Code:

A = diag(1:6) + diag(7:11,1) + diag(12:15,2)

A = A + triu(A,1)'

Output:

A =

1 7 12 0 0 0

0 2 8 13 0 0

0 0 3 9 14 0

0 0 0 4 10 15

0 0 0 0 5 11

0 0 0 0 0 6

A =

1 7 12 0 0 0

7 2 8 13 0 0

12 8 3 9 14 0

0 13 9 4 10 15

0 0 14 10 5 11

0 0 0 15 11 6

Q8. Do some cool operations:

Code: A = rand(10)

Output:

0.1622 0.4505 0.1067 0.4314 0.8530 0.4173 0.7803 0.2348 0.5470 0.9294

0.7943 0.0838 0.9619 0.9106 0.6221 0.0497 0.3897 0.3532 0.2963 0.7757

0.3112 0.2290 0.0046 0.1818 0.3510 0.9027 0.2417 0.8212 0.7447 0.4868

0.5285 0.9133 0.7749 0.2638 0.5132 0.9448 0.4039 0.0154 0.1890 0.4359

0.1656 0.1524 0.8173 0.1455 0.4018 0.4909 0.0965 0.0430 0.6868 0.4468

0.6020 0.8258 0.8687 0.1361 0.0760 0.4893 0.1320 0.1690 0.1835 0.3063

0.2630 0.5383 0.0844 0.8693 0.2399 0.3377 0.9421 0.6491 0.3685 0.5085

0.6541 0.9961 0.3998 0.5797 0.1233 0.9001 0.9561 0.7317 0.6256 0.5108

0.6892 0.0782 0.2599 0.5499 0.1839 0.3692 0.5752 0.6477 0.7802 0.8176

0.7482 0.4427 0.8001 0.1450 0.2400 0.1112 0.0598 0.4509 0.0811 0.7948

Code: A = fix(A)

Output:

64 20 31 59 8 96 3 10 3 18

37 30 92 26 26 54 88 65 74 23

81 47 43 60 80 52 91 49 50 88

53 23 18 71 2 23 79 77 47 2

35 84 90 22 92 48 9 71 90 48

93 19 97 11 73 62 26 90 60 16

87 22 43 29 48 67 33 89 61 97

55 17 11 31 57 39 67 33 85 71

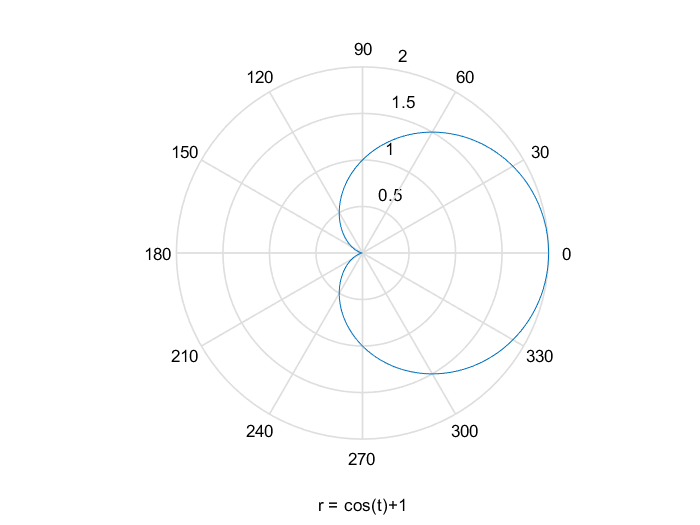
62 22 25 42 23 36 13 69 80 50

58 43 40 50 45 98 72 19 57 47

Q9. Fun with plotting: plot the cardioid.

Code: ezpolar('cos(t)+1',[0 2\*pi])

Output:



CHAPTER 4:

Q1. A script file to compute sine series:

Code:

function y = myFun(x,n)

%SINE\_SERIES: computes sin(x) from series expansion

%type sine\_series(angle in radians, no. of terms for computation)

        for k = 1 : n

                sine\_ans(k) =   ( (-1)^(k-1) ) \*   (x ^(2\*k-1)  )  /  factorial(2\*k-1) ;

        end

        sum\_sine = sum(sine\_ans);

        disp('sin(x) = ')

        disp(sum\_sine)

end

Output:

>> sine\_series(pi/2,5)

sin(x) =

1.0000

>> sine\_series(pi/3,50)

sin(x) =

0.8660

>> sine\_series(pi/6,100)

sin(x) =

0.5000

Q2. function file to compute sine series:

Code:

function y = myFun(x,n)

    %SINE\_SERIES: computes sin(x) from series expansion

    %type sine\_series(angle in radians, no. of terms for computation)

    if nargin<2

        n = 10;

    end

        if floor(n)==n && n>0

            for index = 1 : length(x)

                for k = 1 : floor(n)

                    sine\_ans(k) = (  ( (-1)^(k-1) ) \* (  (x(index)) ^(2\*k-1)  )  /  factorial(2\*k-1) );

                end

            sum\_sine(index) = sum(sine\_ans);

            end

            disp('sin(x) = ')

            disp(sum\_sine)

        else

            disp('n must be greater than 0 and must be an integer only')

        end

    y\_2 = sin(x);

    err = y\_2 - sum\_sine;

    disp('error percent is:')

    100\*err

end

Output:

>>help vector\_input\_sine\_series

SINE\_SERIES: computes sin(x) from series expansion

type sine\_series(angle in radians, no. of terms for computation)

>> my\_vect = [0, pi/6, pi/4, pi/2];

>> vector\_input\_sine\_series(my\_vect) %no n specified, takes default 10

sin(x) =

0 0.5000 0.7071 1.0000

error percent is:

ans =

0 0 0 0

>> vector\_input\_sine\_series(my\_vect,2)

sin(x) =

0 0.4997 0.7047 0.9248

error percent is:

ans =

0 0.0326 0.2454 7.5168

Q6. Calculate factorial of a number using recursion In MATLAB:

Code:

function r = myFun(n)

    if n<=0

        r=1;

    else

        r =  n \* CH4\_qn6\_Recursion(n-1);

    end

end

Output:

>> CH4\_qn6\_Recursion(5)

ans =

120

>> CH4\_qn6\_Recursion(3)

ans =

6