# Exercise 1

# Write a script file to find the 1st, 2nd and 3rd derivative of

**z = Sin2(α) + Cos2(α)**

Solution:

**%Create a script file with name derivative.m**

syms a;

z = sin(a)^2 + cos(a)^2;

d1 = diff(z);

d2 = diff(z,2);

d3 = diff(z,3);

disp('Main Function');

disp(z);

disp('1st deriavative');

disp(d1);

disp('2nd deriavative');

disp(d2);

disp('3rd deriavative');

disp(d3);

**>> derivative**

## Output:-

Main Function

cos(a)^2 + sin(a)^2

1st deriavative

0

2nd deriavative

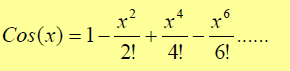
0

3rd deriavative

0

# Exercise 2

**The Taylor series expansion of cos(x) is given by**

****

**a. Plot cos(x) vs x from -3<x<3 using increments of x equal to 0.1.**

**b. On the same plot, compare cos(x) with 1-x2/2, and 1-x2/2+x4/24 (3 curves) over the same range of x considered in part (a)**

**c. Make a table similar to the one shown bellow of the text for the case of x =1. You may want to use the Matlab program calc\_exp.m which performs a similar operation for the function f(x) = ex.**

**Terms Result eps\_t(%) eps\_a(%)**

**1.0000 1.0000 -85.0816 0**

**2.0000 0.5000 7.4592 -100.0000**

**3.0000 0.5417 -0.2525 7.6923**

**4.0000 0.5403 0.0045 -0.2571**

**5.0000 0.5403 -0.0001 0.0046**

**6.0000 0.5403 0.0000 -0.0001**

# d. How many terms are needed before the true percent relative error, εt, is less than 0.01%?

Solution:

1. >> x = [-3:0.1:3];

>> plot(x,cos(x))

>> xlabel('x');

>> ylabel('Cos(x)');

1. >> syms x

>> f1 = cos(x);

>> f2 = 1-x^2/2;

>> f3 = 1-x^2/2+x^4/24;

>> x = [-3:.1:3];

>> plot(x,eval(f1),'r-+',x,eval(f2),'g:\*',x,eval(f3),'b-\*')

>> legend('cos(x)','1-(x.^2)/2','1-(x.^2)/2+(x.^4)/2',0)

1. % This program computes the cosine

% function by means of a Taylor series

% Input x and n

x=input('Enter x... ');

n=input('Enter number of terms... ');

% initialize variables and vectors

exact\_value = cos(x);

eps\_t = 100\*(exact\_value - 1)/exact\_value;

EPS\_t = zeros(1,n);

EPS\_t(1) = eps\_t;

EPS\_a = zeros(1,n);

SUM = ones(1,n);

sum=1; % the first term in the sum is 1

fact = 1;

% Begin loop to compute and sum terms 2 through n

for k=2:n;

p=(2\*k-2);

sum = sum + ((-1)^(k+1))\*(x^p)/factorial(p);

eps\_t = 100\*(exact\_value - sum)/exact\_value;

eps\_a = 100\*(sum - SUM(k-1))/sum;

% Store results

SUM(k) = sum;

EPS\_t(k) = eps\_t;

EPS\_a(k) = eps\_a;

end

approximate\_value = sum;

% Display results

disp(' ')

disp(' ')

disp(' Terms Result eps\_t(%) eps\_a(%)')

disp(' ')

disp([(1:n)' SUM' EPS\_t' EPS\_a'])

1. We need 4 terms before the magnitude of εt, in order to make error smaller than 0.01% when x=1.

# Exercise 3

**Write a script to generate the Taylor series as created by the built in function.**

Solution:

**%Create a function file with name tylr.m**

function [Taylor\_Series]=tylr()

func=input(' Enter Function : ','s');

deg=input (' Enter Pol Degree : ');

syms y x

y=0; n=0;

while n <= deg

eq=diff(func,n);

eq=subs(eq,0);

eq = (eq\*(x-0)^n)/factorial(n);

y=y+eq;

n=n+1;

end

disp(' Taylor Series ');

Taylor\_Series=y;

**>>tylr**