

Mathematics 3 File

Name: Ashish Gupta

Roll Number: 2K16/MC/023

Department of Applied Mathematics

Delhi Technological University

Content:

|  |  |  |
| --- | --- | --- |
| **S. No** | **Practical** | **Signature** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |

Practical 1:

Write a program to determine the largest two Eigen values of the following matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 1 | -1 |
| 0 | 2 | 3 | 5 | 0 |
| -1 | 0 | 0 | 0 | 1 |
| 6 | 8 | 1 | 2 | -2 |
| 1 | 1 | 1 | 1 | 1 |

**Theory:**

In [linear algebra](https://en.wikipedia.org/wiki/Linear_algebra), an **eigenvector** or **characteristic vector** of a linear transformation is a non-zero [vector](https://en.wikipedia.org/wiki/Vector_space) that only changes by an overall scale when that linear transformation is applied to it. More formally, if *T* is a linear transformation from a [vector space](https://en.wikipedia.org/wiki/Vector_space) *V* over a [field](https://en.wikipedia.org/wiki/Field_(mathematics)) *F* into itself and **v** is a vector in *V* that is not the zero vector, then **v** is an eigenvector of *T* if *T*(**v**) is a scalar multiple of **v**.

**Function Used:**

Ans=eig(A), evalues all the eigen values in the given matrix and saves it as a list.

**Matlab Code:**

A=[1 0 0 -1 1; 0 2 3 5 0; -1 0 0 0 1; 1 8 1 2 -2 ; 1 1 1 1 1];

Ans=eig(A);

sort(Ans);

for i=1:2

Ans(i,1)

end

**Output:**

ans = 8.1330

ans = -4.2354

Practical 2 and Practical 3:

W.A.P to show the consistency/ non consistency of the system of linear equations. If the system is consistent, then write a program to solve the given system of equations for unique/ infinite solutions:

**System Given: AX=B**

**A= B=**

|  |  |  |
| --- | --- | --- |
| **1** | **2** | **1** |
| **2** | **3** | **5** |
| **7** | **1** | **2** |

|  |
| --- |
| **5** |
| **7** |
| **0** |

**Function Used:**

**RankofA=rank(A);** Evaluates the rank of matrix A

**RankOfAug=rank(Aug);** Evaluates the rank of matrix Aug(Augmented Matrix)

**X=inv(A);** Evaluates the inverse of matrix A

**Matlab Code:**

A=[1 2 1;2 3 5;7 1 2];

B=[5;7;0];

Aug=[A,B];

flag=0;

RankofA=rank(A);

RankOfAug=rank(Aug);

if(RankofA~=RankOfAug)

fprintf('The given system of equations is inconsistent')

flag=1;

else

fprintf('The given system is consistent\n')

end

clear X;

if(flag==0)

% fprintf('\n')

X=inv(A);

Ans=(X\*B)

End

**Output:**

>> Practical2\_ConsistencyOfMatrix

The given system is consistent

Ans =

-0.3636

2.7273

-0.0909

Practical 4:

Using inbuilt ode solver 23 and ODE 45, find y(0.3), where y is solution of the following initial value problem abd hence compare this value to the original value:

## System Given: dy/dx= y+x, y(0) = 1

**Function Used:**

**Eqn(y,t)** gets the differential equation to be input into the ode23 inbuilt solver.

**ODE 23 and ODE 45** A function handle that evaluates the right side of the differential equations. All solvers solve systems of equations in the form *y*′ = *f*(*t*,*y*) or problems that involve a mass matrix, *M*(*t*,*y*)*y*′ = *f*(*t*,*y*). The ode23s solver can solve only equations with constant mass matrices. ode15s and ode23t can solve problems with a mass matrix that is singular, i.e., differential-algebraic equations (DAEs).

**Tspan:** Defines a range for ode solvers.

**Matlab Code:**

**ODE-45 Solver:**

function f = fun1( y,t )

f=y+t;

end

% in new script file ...

syms x y t;

y0=1;

Value=eqn(x,y);

yspan=[-5,5];

[x,y]=ode45(value,yspan,y0);

%output

plot(x,y,'r\*');

y1=y(3)

grid on;

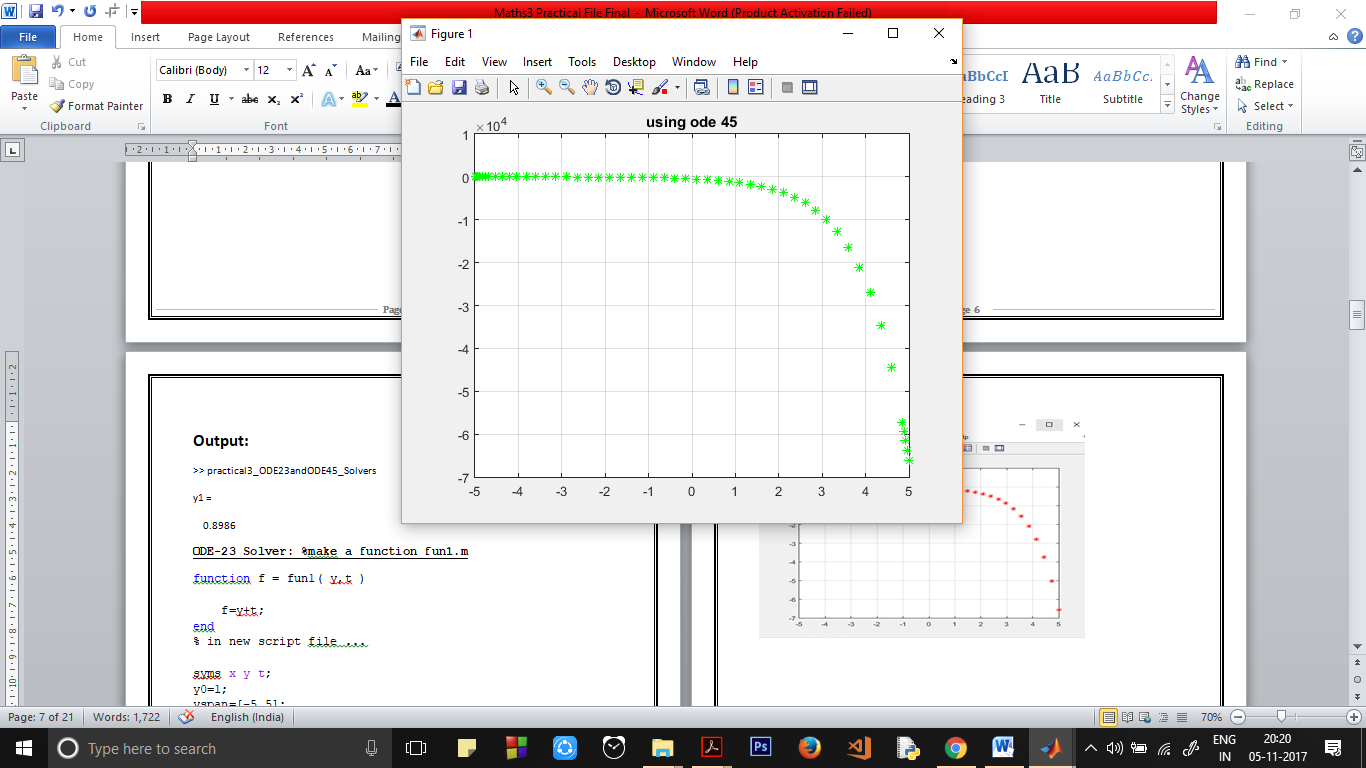
title('using ode 45')

**Output and Graph:**

>> practical3\_ODE23andODE45\_Solvers

y1 =

0.8986



**ODE-23 Solver: %make a function fun1.m**

function f = fun1( y,t )

f=y+t;

end

% in new script file ...

syms x y t;

y0=1;

yspan=[-5,5];

[x,y]=ode23(‘fun1’,yspan,y0);

%output

plot(x,y,'r\*');

y1=y(3)

grid on;

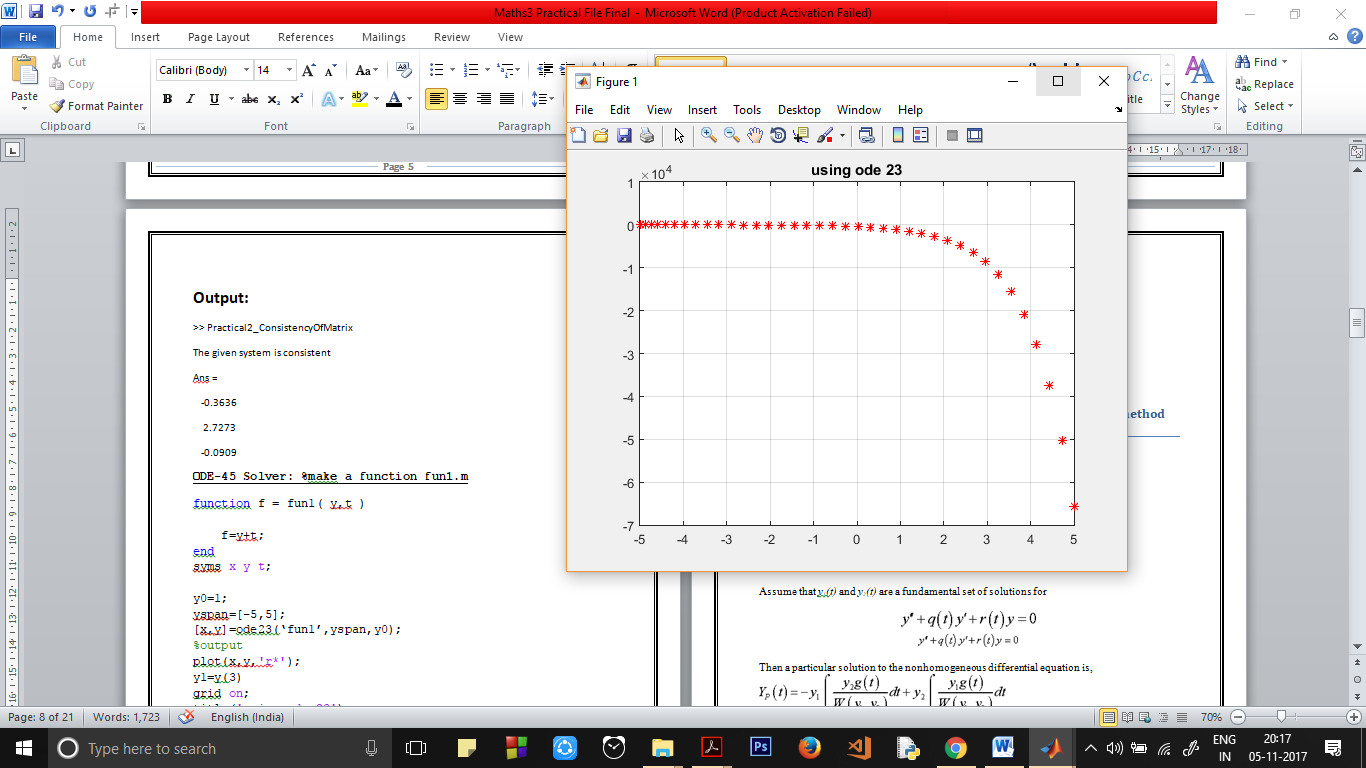
title('using ode 23')

**Output and Graph:**

>> practical3\_ODE23andODE45\_Solvers

y1 =

0.4975



Practical 5:

Write a program to solve D^2(y) +4\*y=Sec(x) by using the method of variation of parameters

**Theory:**

**Variation of Parameters**

Consider the differential equation,

http://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/eq0028MP.gif

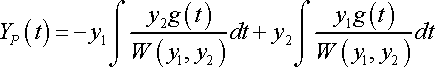
http://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/eq0028M.gifhttp://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/empty.gif

Assume that *y1(t)* and *y2(t)* are a fundamental set of solutions for

http://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/eq0029MP.gif

http://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/eq0029M.gifhttp://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/empty.gif

Then a particular solution to the nonhomogeneous differential equation is,



http://tutorial.math.lamar.edu/Classes/DE/VariationofParameters_files/eq0030M.gif

**Function Used:**

**Eqn(y,t)** gets the differential equation to be input into the ode23 inbuilt solver.

**ODE 23 and ODE 45** A function handle that evaluates the right side of the differential equations. All solvers solve systems of equations in the form *y*′ = *f*(*t*,*y*) or problems that involve a mass matrix, *M*(*t*,*y*)*y*′ = *f*(*t*,*y*). The ode23s solver can solve only equations with constant mass matrices. ode15s and ode23t can solve problems with a mass matrix that is singular, i.e., differential-algebraic equations (DAEs).

**Tspan:** Defines a range for ode solvers.

**Matlab Code:**

syms x t;

cf=dsolve('D2y+4\*y=0');

y1dot=diff('cos(2\*t)',t);

y2dot=diff('sin(2\*t)',t);

w=[cos(2\*t) sin(2\*t) ;y1dot y2dot];

wdet=det(w);

w1=[0 sin(2\*t); sec(t) y2dot];

w1det=det(w1);

w2=[cos(2\*t) 0 ;y1dot sec(t)];

w2det=det(w2);

u=int(w1det/wdet);

v=int(w2det/wdet);

perticular\_solution=u\*cos(2\*t)+v\*sin(2\*t);

Answer=cf+perticular\_solution

**Output:**

>> method\_VariationOfParameters

Answer =

cos(2\*t)\*cos(t) + C3\*cos(2\*t) + C4\*sin(2\*t) - sin(2\*t)\*(atanh(sin(t))/2 - sin(t))

Practical 6:

Graphically compare the function sin(x) and Taylor series expansion of sin(x) up to degree 10 in the neighbourhood of 1.

**Theory:**

The formula for the Taylor series expansion for sin(x) is :

http://wiki.ubc.ca/images/math/4/4/7/447a79826774707026bbefcd76962d3a.png

In mathematics, a **Taylor series** is a representation of a function as an infinite sum of terms that are calculated from the values of the function's derivatives at a single point.

**Function Used:**

**Taylor (**[**f**](file:///C:\Program%20Files\MATLAB\MATLAB%20Production%20Server\R2015a\help\symbolic\taylor.html?searchHighlight=taylor#inputarg_f)**)**computes the taylor’s series expansion of ‘f’ up to the fifth order. The expansion point is 0.

**Taylor** (**[f](file:///C:\\Program%20Files\\MATLAB\\MATLAB%20Production%20Server\\R2015a\\help\\symbolic\\taylor.html?searchHighlight=taylor" \l "inputarg_f),[Name,Value](file:///C:\\Program%20Files\\MATLAB\\MATLAB%20Production%20Server\\R2015a\\help\\symbolic\\taylor.html?searchHighlight=taylor" \l "namevaluepairarguments)**)uses additional options specified by one or more Name, Value pair arguments.

**Ezplot(x,f)** plots the function’s graph.

**Matlab Code:**

syms f x;

f=taylor(sin(x),'Order',11);% it represents the powers from 0

%all the way upto 10

h=ezplot(x,f);

set(h,'color','r');

grid on;

hold on;

y=sin(x);

plot2=ezplot(y);

set(plot2,'color','g');

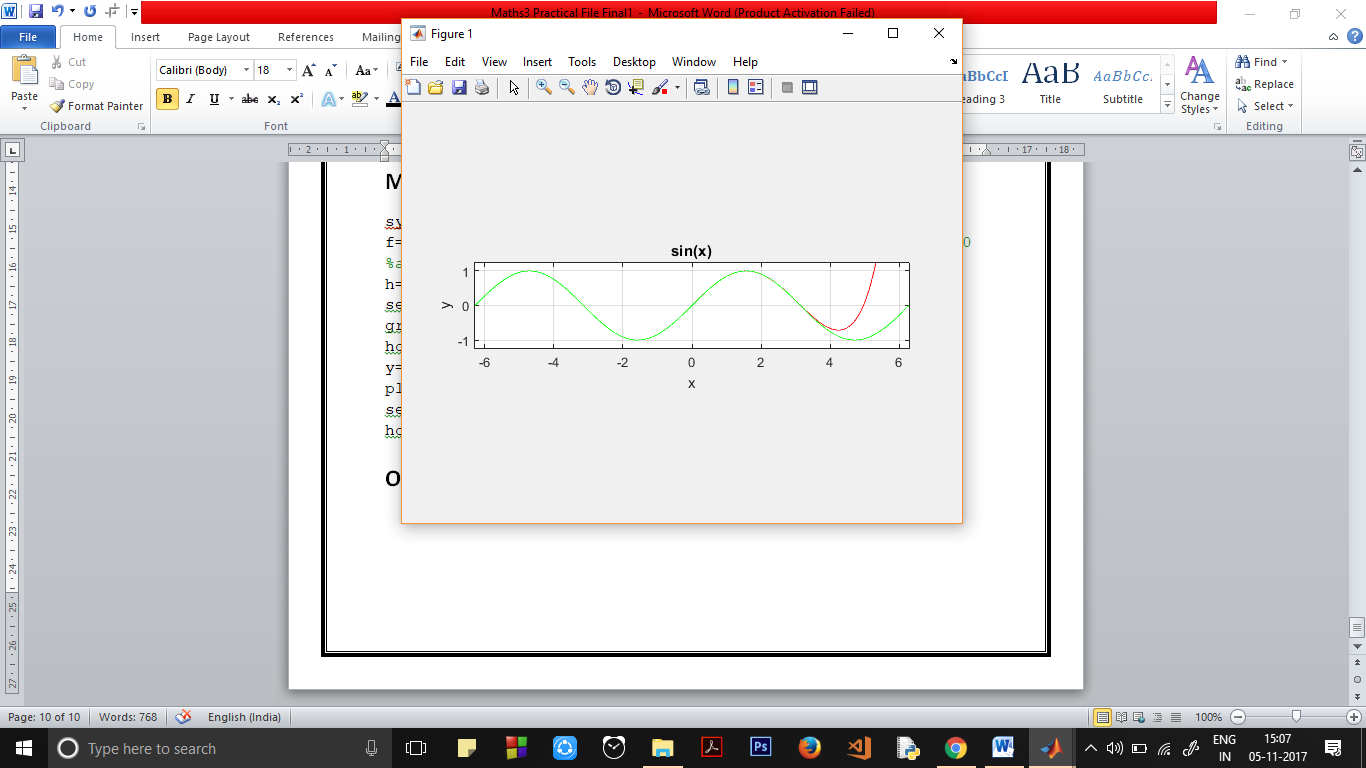
hold off;

**Output and Graph:**

>> taylorSeries\_Sinx

f =

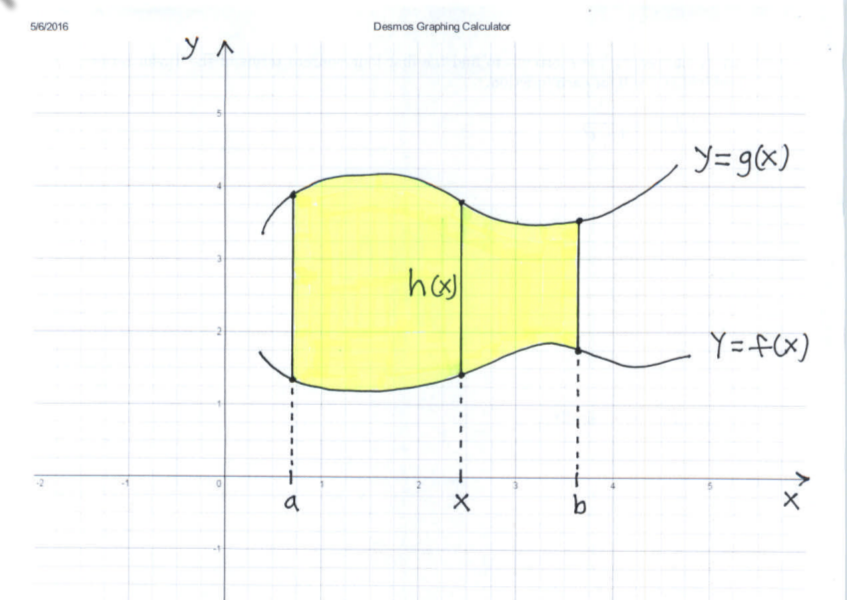
x^9/362880 - x^7/5040 + x^5/120 - x^3/6 + x



Practical 7:

Sketch the area/ region enclosed by the curves f(x)= x^3 – 3\*x^2 + 3\*x and g(x)=x^2 and find the area of the enclosed region.

**Theory:**



The area of RR is given by:

**AREA=∫ba h(x)dx=∫ba (g(x)−f(x))dx** (ba is integration limits from a to b)

**Function Used:**

**Ezplot(x,f)** plots the function’s graph.

**area1=int(f1-f2,0,1);**

**area2=int(f2-f1,1,3);**

**int=int(x)** (integrate is used to integrate a function from limits initial to final)

**Matlab Code:**

syms x t;

f1=x^3- 3\*x^2 +3\*x;

f2=x^2;

plot1=ezplot(f1);

set(plot1,'color','r');

hold on;

plot2=ezplot(f2);

set(plot2,'color','g');

grid on;

legend('x^3- 3\*x^2 +3\*x','x^2');

xlabel('X-Axis');

ylabel('Y-Axis');

title('Area Enclosed');

area1=int(f1-f2,0,1);

area2=int(f2-f1,1,3);

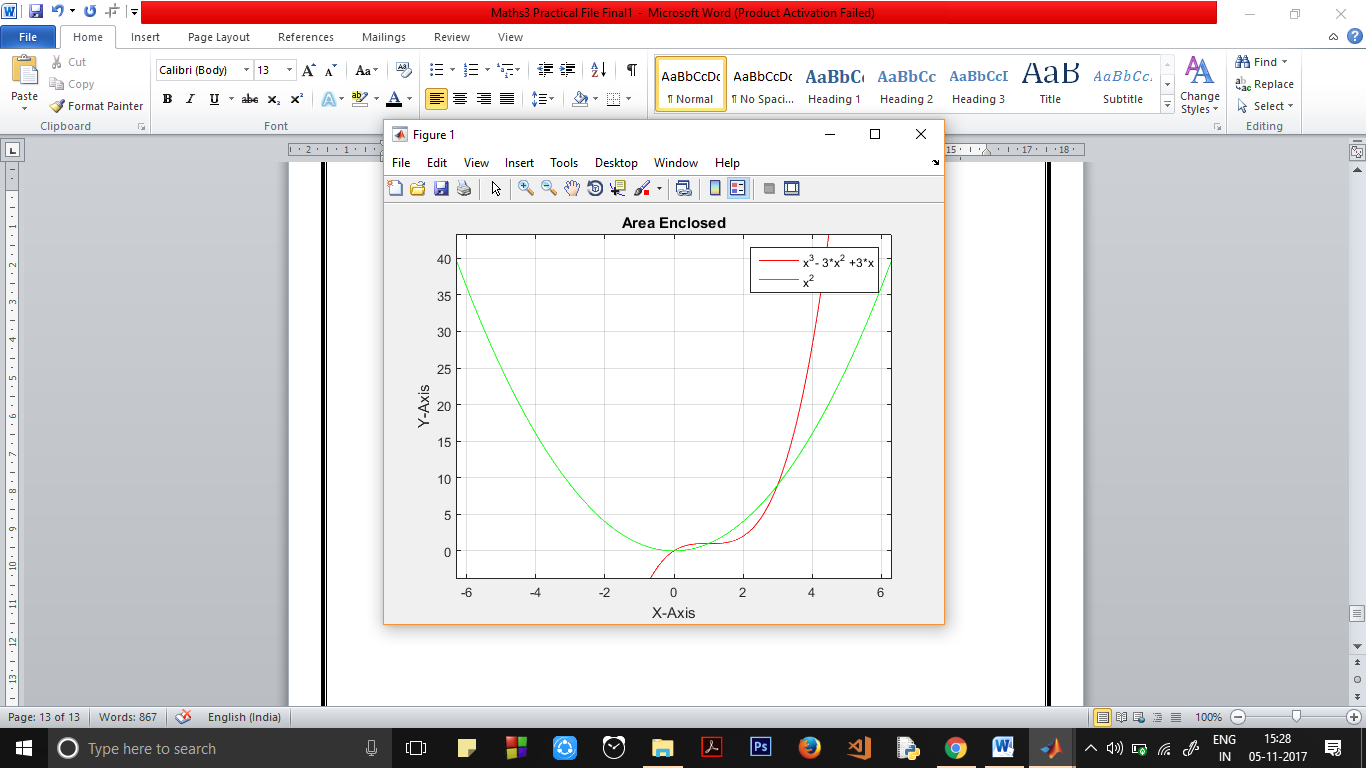
area=area1+area2

**Output and Graph:**

>> area\_Enclosed

area =

37/12



Practical 8:

**To draw the tangent line at point on a given curve** 𝐲=𝟏+𝐱𝟐 **at the point (2,5) and also find the Radius of curvature at that point.**.

**Theory:**

A **tangent line** is a line that touches a curve at a single point and does not cross through it. The point where the curve and the tangent meet is called the point of tangency. We know that for a line **y=m\*x+c** its slope at any point is **m**. The same applies to a curve. When I say the slope of a curve, I mean the slope of tangent to the curve at a point.

Radius of curvature is :



**Function Used:**

**ezplot**(FUN) plots the function FUN(X) over the default domain -2\*PI < X < 2\*PI, where FUN(X) is an explicitly defined function of X.

**ezplot**(FUN2) plots the implicitly defined function FUN2(X,Y) = 0 over the default domain -2\*PI < X < 2\*PI and -2\*PI < Y < 2\*PI.

**sqrt(X)** is the square root of the elements of X.

**Matlab Code:**

syms x t;

f1=1+x^2;

d(x)=diff(f1);

m=d(2);

c=5-m\*2;

f2=m\*x+c;

plot1=ezplot(f1);

set(plot1,'color','r');

hold on;

grid on;

plot2=ezplot(f2);

legend('1+x^2','y=m\*x+c');

xlabel('X-axis');

ylabel('Y-axis');

title('Tangent to the curve');

ydot2=diff(diff(f1));

roc=sqrt((1+m^2)^3)/2

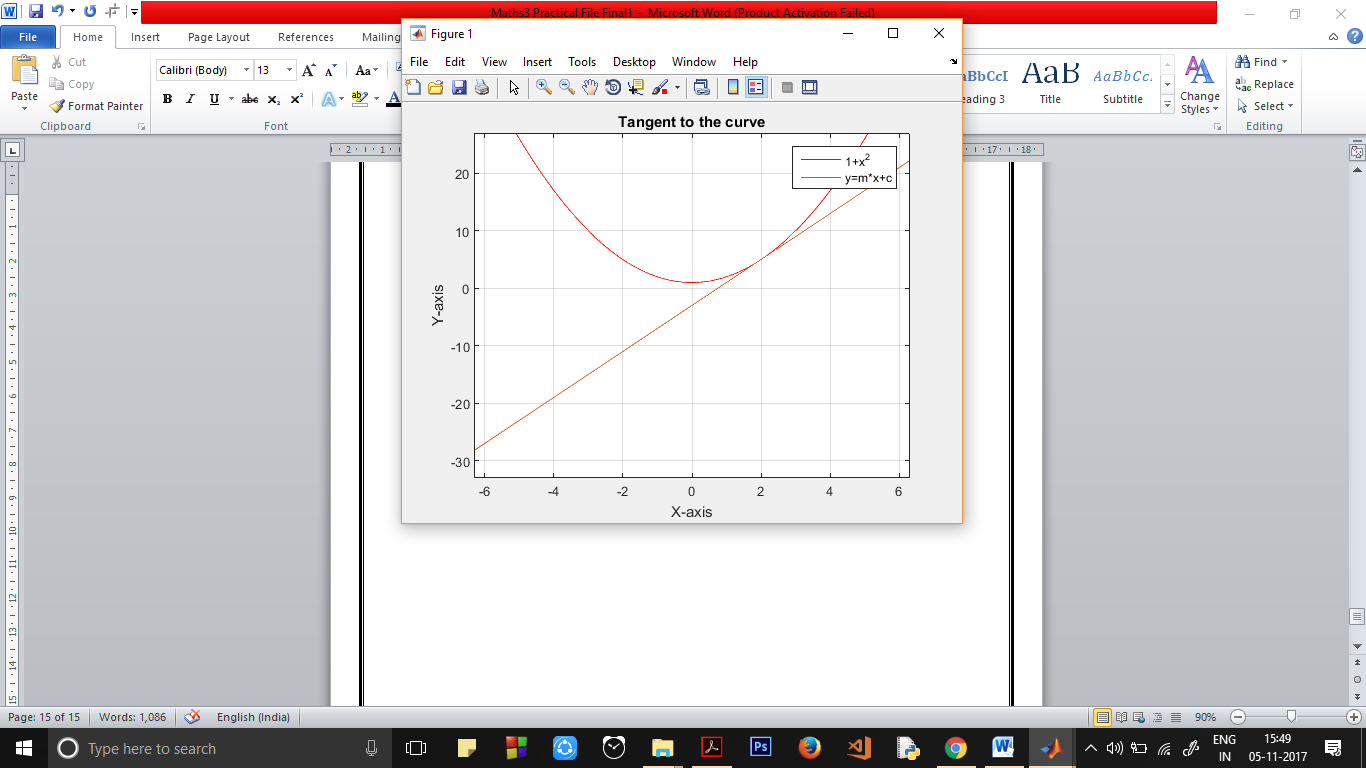
hold off;

**Output and Graph:**

>> tangent\_to\_a\_curve

roc =

(17\*17^(1/2))/2



Practical 9:

**Plot the surface defined by the function f(x, y) = −xye-2(x^2+y^2) on the domain −2 ≤ x ≤ 2 and − 2 ≤ y ≤ 2. Find the values and locations of the maxima and minima of the function.**

**Theory:**

# Maxima and Minima of Functions of Two Variables:

|  |  |
| --- | --- |
| |  | | --- | | Locate relative maxima, minima and saddle points of functions of two variables. Several examples with detailed solutions are presented. 3-Dimensional graphs of functions are shown to confirm the existence of these points. More on Optimization Problems with Functions of Two Variables in this web site.  Theorem:  Let f be a function with two variables with continuous second order **partial derivatives** fxx, fyy and fxy at a critical point (a,b). Let  D = fxx(a,b) fyy(a,b) - fxy2(a,b)     * + - 1. If D > 0 and fxx(a,b) > 0, then f has a relative minimum at (a,b).       2. If D > 0 and fxx(a,b) < 0, then f has a relative maximum at (a,b).       3. If D < 0, then f has a saddle point at (a,b).       4. If D = 0, then no conclusion can be drawn. | |

**Function Used:**

**ezplot**(FUN) plots the function FUN(X) over the default domain -2\*PI < X < 2\*PI, where FUN(X) is an explicitly defined function of X.

**ezplot**(FUN2) plots the implicitly defined function FUN2(X,Y) = 0 over the default domain -2\*PI < X < 2\*PI and -2\*PI < Y < 2\*PI.

**Plot:** Plots the graph of the desired function, with valid inputs

**Meshgrid:** Used to plot graph of 3D plots, functions of 2 independent variables.

**Max & Min:** Calculate max/min of a set of values.

**Matlab Code:**

clc;

clear;

syms x y;

[x,y]=meshgrid(-2:0.03:2,-2:0.03:2);

f=-x.\*y.\*exp(-2\*(x.^2+y.^2));

figure(1)

mesh(x,y,f),xlabel('X'),ylabel('y'),grid

figure(2)

contour(x,y,f)

xlabel('X'),ylabel('y'),grid,hold on

fmax=max(max(f))

kmax=find(f==fmax)

pos=[x(kmax) y(kmax)]

%pos = -0.5000 0.5000 0.5000 -0.5000

plot(x(kmax),y(kmax),'\*')

text(x(kmax),y(kmax),'Maximum')

%plotting the maximum value on the graph

fmin=min(min(f))

kmin=find(f==fmin)

pos1=[x(kmin) y(kmin)]

plot(x(kmin),y(kmin),'\*')

% We are plotting the minimum value now

text(x(kmin),y(kmin),'Minimum')

**Output and Graph:**

fmax =

0.0919

kmax =

6784

11173

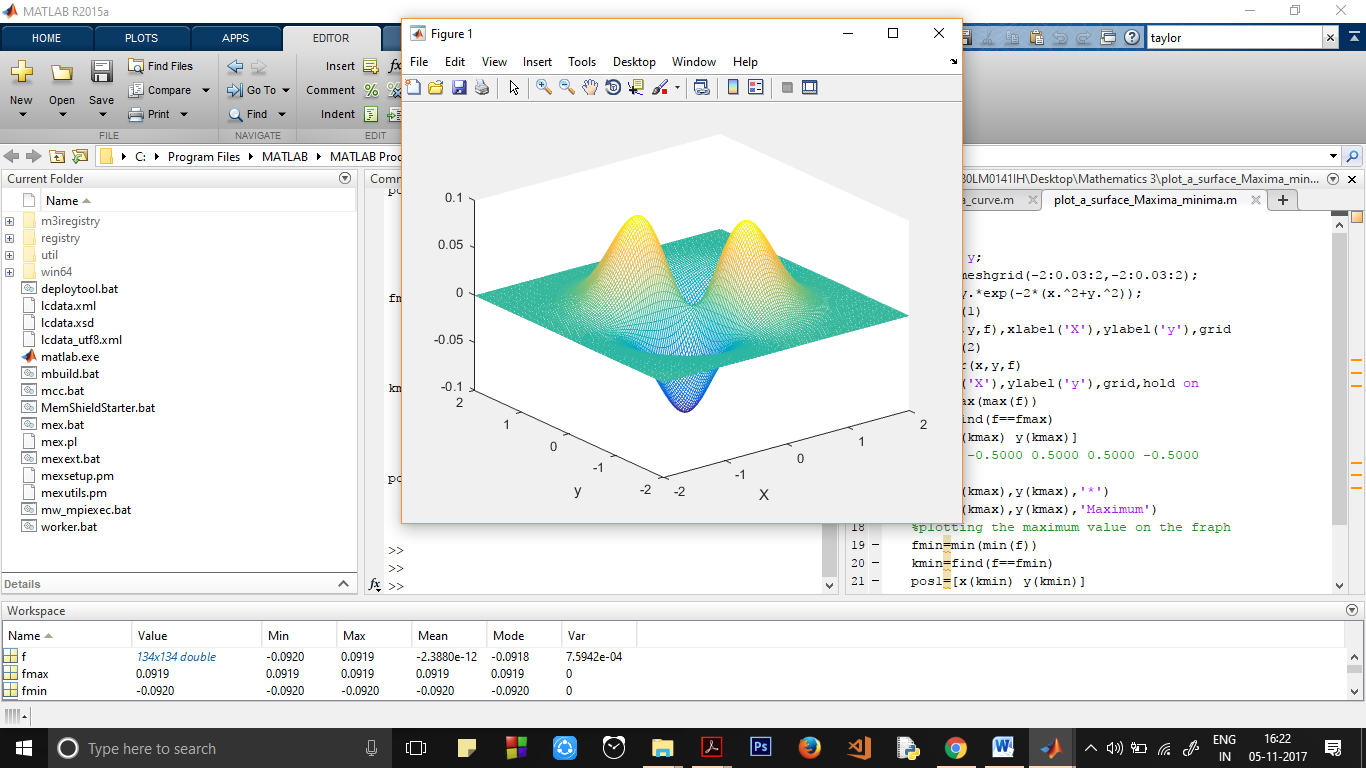
pos =

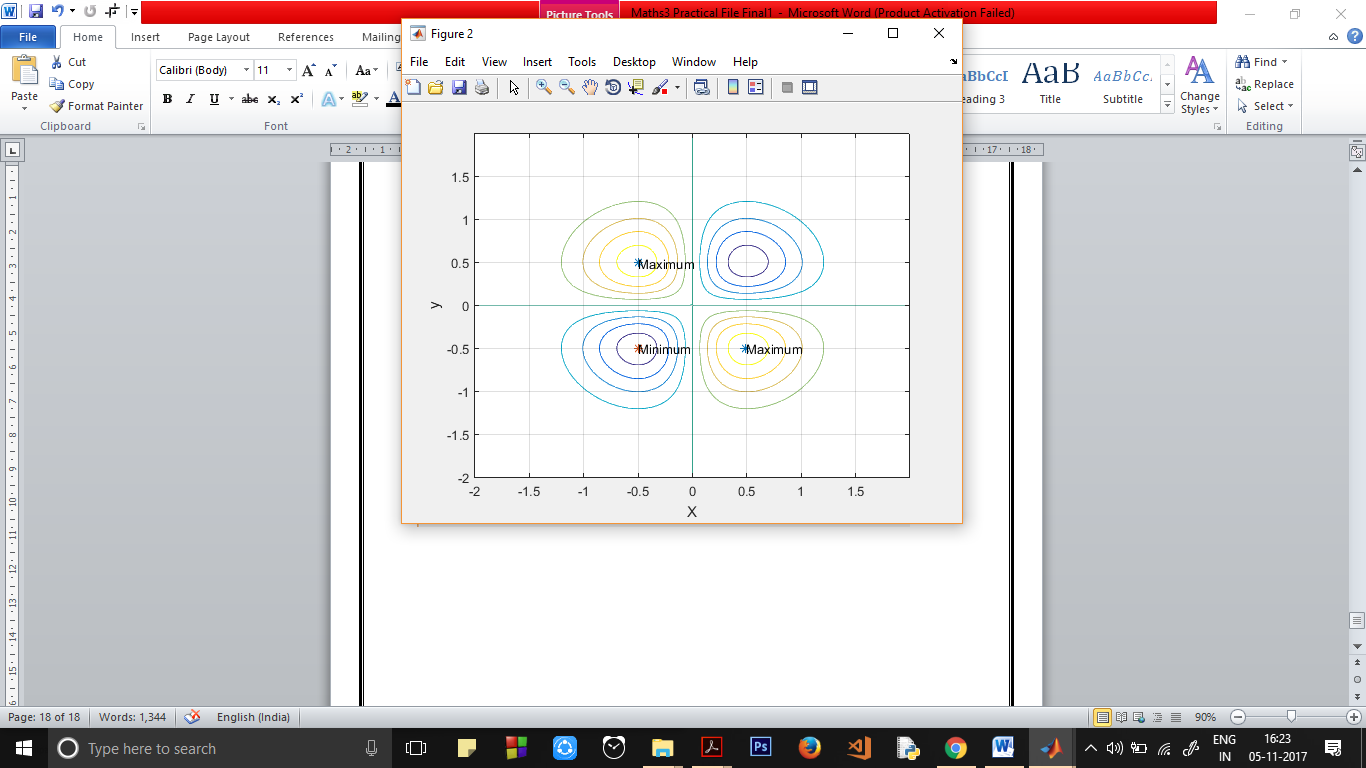
-0.5000 0.4900

0.4900 -0.5000

fmin =

-0.0920





Practical 10:

**Determine the characteristic polynomial of a matrix evaluating the polynomial p(**λ**) at the nth points.**

**Theory:**

In linear algebra, the **characteristic polynomial** of a square matrix is a [polynomial](https://en.wikipedia.org/wiki/Polynomial) which is invariant under [matrix similarity](https://en.wikipedia.org/wiki/Matrix_similarity) and has the [eigenvalues](https://en.wikipedia.org/wiki/Eigenvalues) as [roots](https://en.wikipedia.org/wiki/Root_of_a_polynomial). It has the [determinant](https://en.wikipedia.org/wiki/Determinant) and the [trace](https://en.wikipedia.org/wiki/Trace_(linear_algebra)) of the matrix as coefficients. The **characteristic polynomial** of an [endomorphism](https://en.wikipedia.org/wiki/Endomorphism) of [vector spaces](https://en.wikipedia.org/wiki/Vector_space) of finite dimension is the characteristic polynomial of the matrix of the endomorphism over any base; it does not depend on the choice of a basis.

The **characteristic equation** is the equation obtained by equating to zero the characteristic polynomial.

**Function Used:**

**ezplot**(FUN) plots the function FUN(X) over the default domain -2\*PI < X < 2\*PI, where FUN(X) is an explicitly defined function of X.

**ezplot**(FUN2) plots the implicitly defined function FUN2(X,Y) = 0 over the default domain -2\*PI < X < 2\*PI and -2\*PI < Y < 2\*PI.

**sqrt(X)** is the square root of the elements of X.

**Matlab Code:**

function [ OUT ] = MyFunction( A )

[m n]=size(A) ;

if m~=n

disp('It is not a Square Matrix’)

OUT=[] ;

return

end

for i=1:(n+1)

x(i)=(i-1)\*pi/n;

y(i)=det(A-x(i)\*eye(n));

end

OUT=polyfit(x,y,n);

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

MyFunction(A)

z=length(ans);

syms x;

f=0;

i=4;

for y=1:1:z

f=f+ans(y).\*x.^(i-1);

i=i-1;

end

f %prints the function f

**Output:**

f =

- x^3 + 15\*x^2 + 18\*x