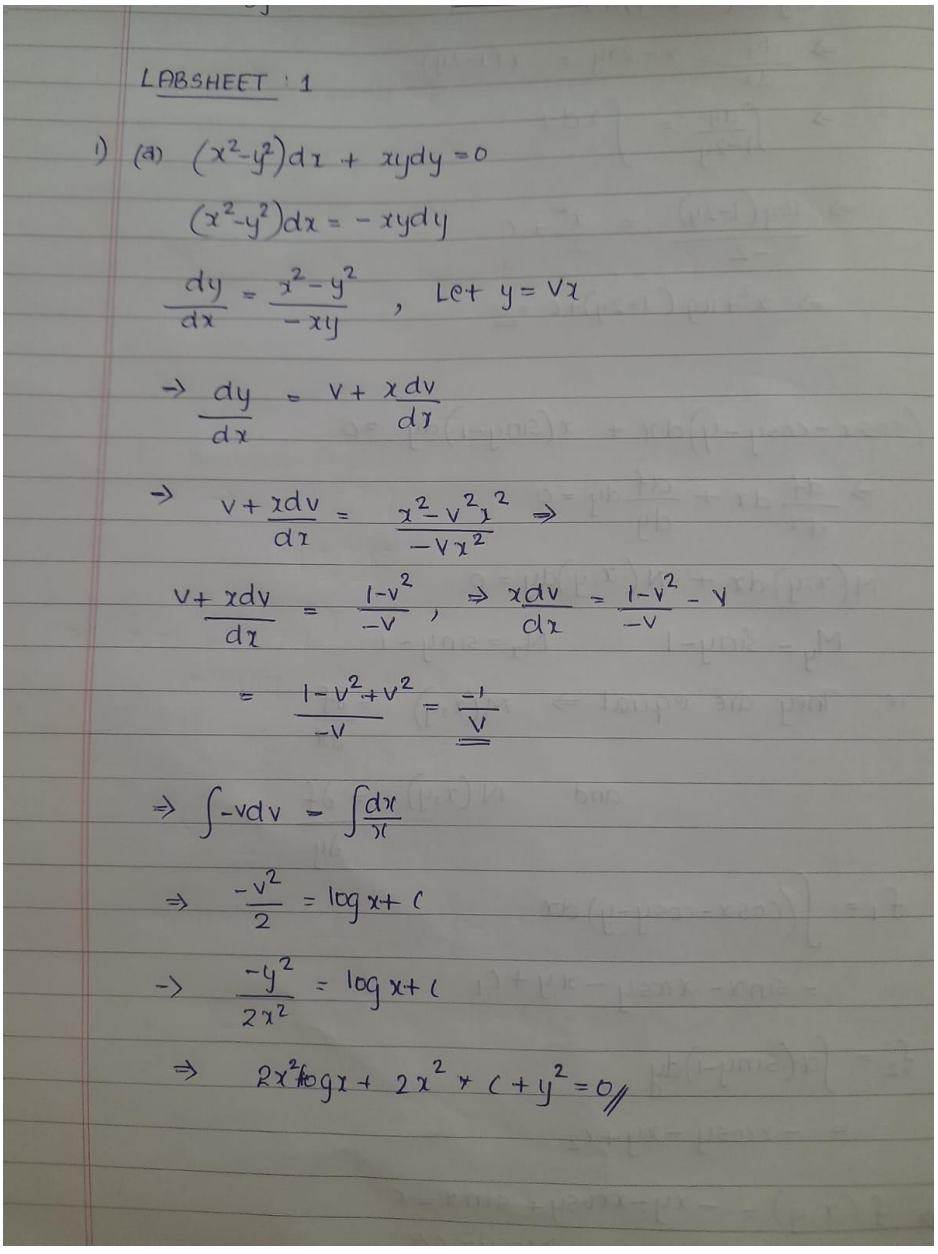
MATHS
Assignment

Anuvind MP
AM.EN.U4AIE22010

1.	Solve the following ODE using pen and paper and plot the solution (in Matlab) for the range of x from 1 to 1.5 a. $(x^2 - y^2)dx + xydy = 0$ b. $dy + (2xy - x)dx = 0$ c. $(\cos x - \cos y - y)dx + x(\sin y - 1)\,dy = 0$ d. $(3e^x y + x)dx + e^x dy = 0; y(0) = 1$



LABSHEET : 1

1) (a) $(x^2-y^2)dx + xydy = 0$

$(x^2-y^2)dx = -xydy$

$\frac{dy}{dx} = \frac{x^2-y^2}{-xy}$, Let $y = vx$

$\rightarrow \frac{dy}{dx} = v + \frac{x\,dv}{dx}$

$\rightarrow v + \frac{x\,dv}{dx} = \frac{x^2 - v^2x^2}{-vx^2} \Rightarrow$

$v + \frac{x\,dv}{dx} = \frac{1-v^2}{-v}$, $\Rightarrow \frac{x\,dv}{dx} = \frac{1-v^2}{-v} - v$

$= \frac{1-v^2+v^2}{-v} = \frac{-1}{v}$

$\Rightarrow \int -v\,dv = \int \frac{dx}{x}$

$\Rightarrow \frac{-v^2}{2} = \log x + c$

$\rightarrow \frac{-y^2}{2x^2} = \log x + c$

$\Rightarrow 2x^2 \log x + 2x^2 \times c + y^2 = 0$ //

b) $dy + (2xy - x)dx = 0$

$\Rightarrow dy = (x - 2xy)dx$

$\Rightarrow \frac{dy}{dx} = x - 2xy = x(1-2y)$

$\Rightarrow \int \frac{dy}{1-2y} = \int x\,dx$

$\Rightarrow \frac{\log(1-2y)}{-2} = \frac{x^2}{2} + c$

$\Rightarrow x^2 + \log(1-2y) + c = 0$

c) $(\cos x - \cos y - y)dx + x(\sin y - 1)dy = 0$

$\Rightarrow \frac{df}{dx}dx + \frac{df}{dy}dy = 0$

$M(x,y)\,dx + N(x,y)\,dy = 0$

$M_y = \sin y - 1 \qquad N_x = \sin y - 1$

ie, They are equal $\Rightarrow M(x,y) = \frac{\partial f}{\partial x}$

and $N(x,y) = \frac{\partial f}{\partial y}$

$f_1 = \int (\cos x - \cos y - y)\,dx$

$= \sin x - x\cos y - xy + c_1$

$f_2 = \int x(\sin y - 1)\,dy$

$= -x\cos y - xy + c_2$

$\Rightarrow f(x,y) = -xy - x\cos y + \sin x = c$

$\Rightarrow \sin x - xy - x\cos y = c$ //

d) $(3e^x y + x)dx + e^x dy = 0 \qquad ; y(0) = 1$

$\rightarrow \frac{dy}{dx} = \frac{-(3e^x y + x)}{e^x} = -3y - xe^{-x}$

$\Rightarrow \frac{dy}{dx} + 3y = -xe^{-x}$

which is of the form $\frac{dy}{dx} + py = Q$

$\Rightarrow p = 3 \; ; \; Q = -xe^{-x}$

2.	Try to solve the above equations and plot the function using the analytic solver dsolve. What do you observe?

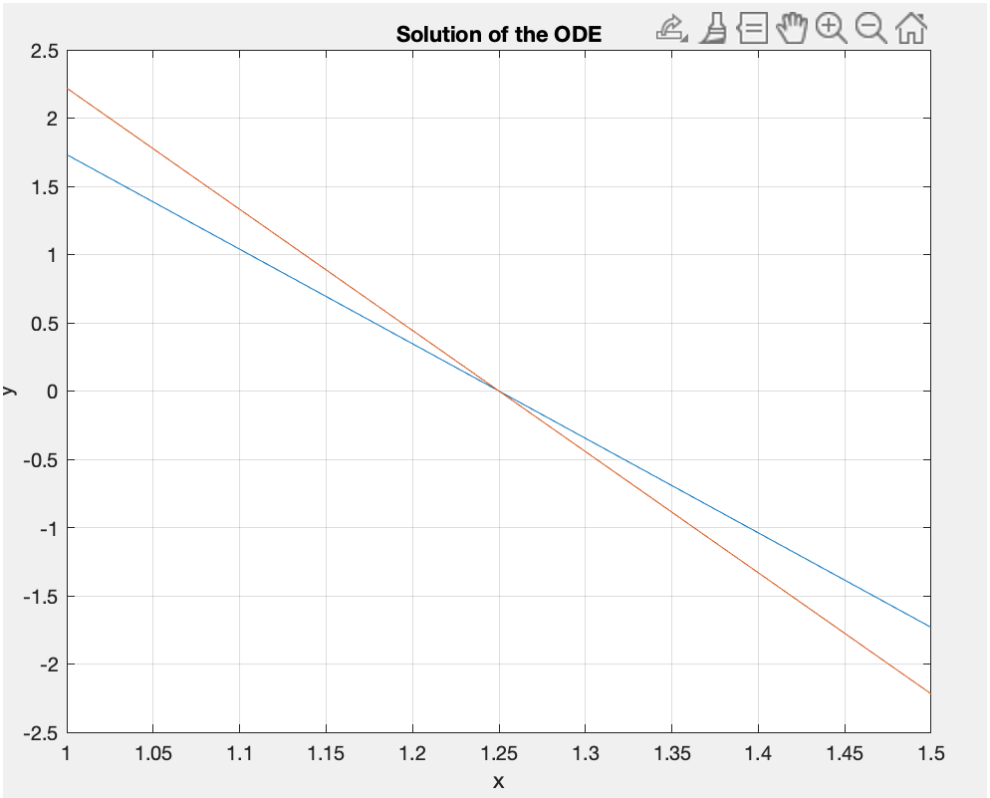
a)

CODE

```
% Define the ODE
syms x y(x)
ode = (x^2 - y^2)*diff(x) + x*y*diff(y) == 0;

% Solve the ODE
sol = dsolve(ode);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Define a range of x values
x_range = 1:0.01:1.5; % Adjust the range accordingly

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q2a
 2^(1/2)*x*(C1 - log(x))^(1/2)
-2^(1/2)*x*(C1 - log(x))^(1/2)

>>
```



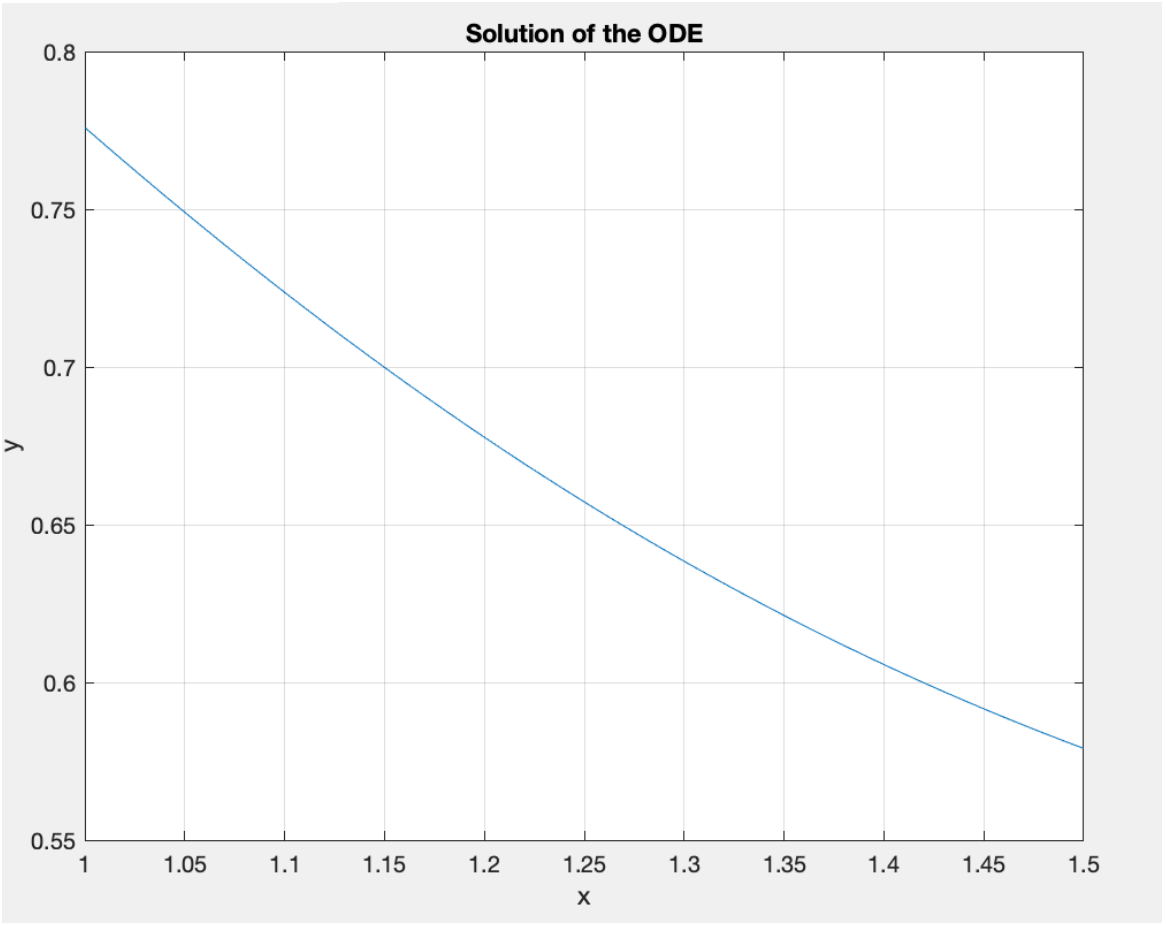
b)

CODE

```
% Define the ODE
syms x y(x)
ode = diff(y)+(2*x*y-x)*diff(x) == 0;

% Solve the ODE
sol = dsolve(ode);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Define a range of x values
x_range = 1:0.01:1.5; % Adjust the range accordingly

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q2b
(C1*exp(-x^2))/2 + 1/2
```



c)

CODE

```
% Define the ODE
syms x y(x)
ode = (cos(x) - cos(y) - y)*diff(x) + x*(sin(y) - 1)*diff(y) == 0;

% Solve the ODE
sol = dsolve(ode);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Define a range of x values
x_range = 1:0.01:1.5; % Adjust the range accordingly

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q2c
Warning: Unable to find symbolic solution.
> In dsolve>assignOutputs (line 235)
In dsolve (line 221)
In Q2c (line 6)
Error using sym/subs
Number of elements in NEW must match number in OLD.

Error in Q2c (line 21)
y_range = subs(sol_numeric, x, x_range);
```

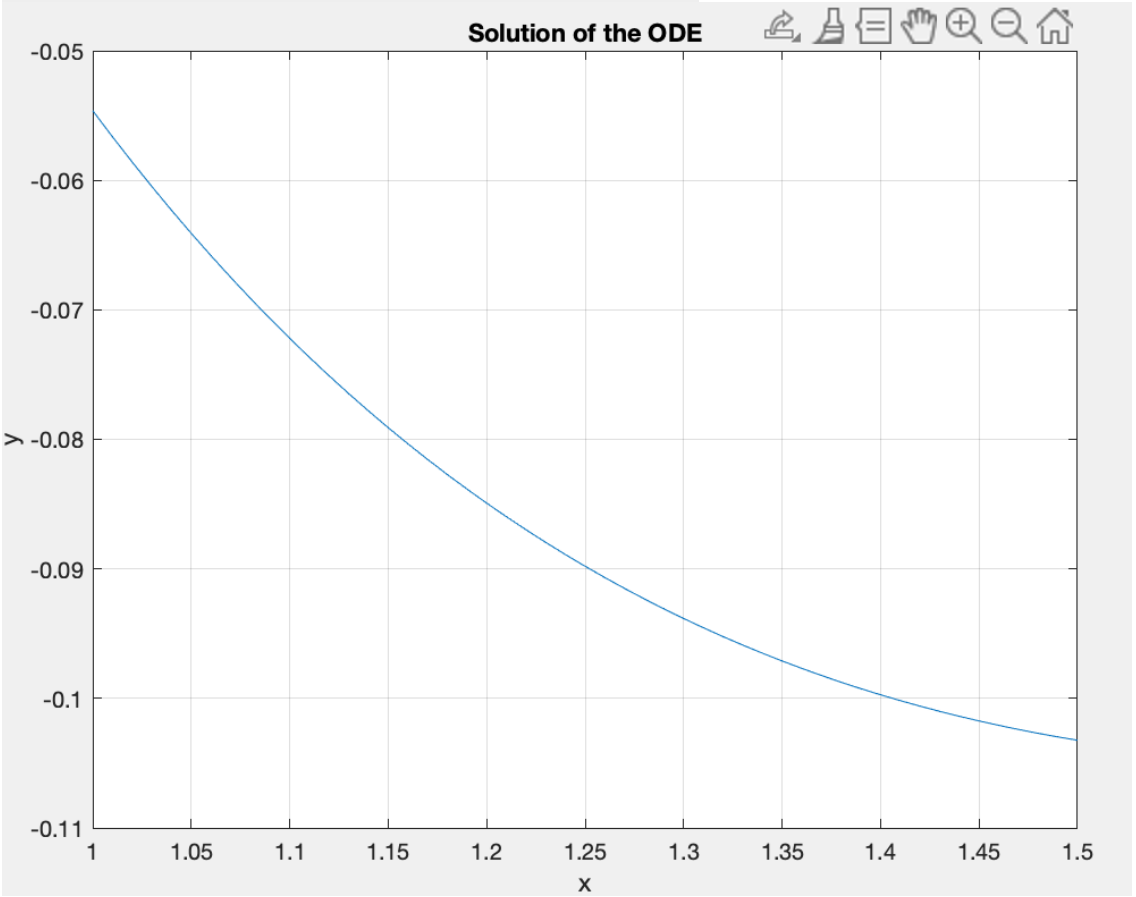
d)

CODE

```
% Define the ODE
syms x y(x)
ode = (3*exp(x)*y + x)*diff(x) + exp(x)*diff(y) == 0;
cond = y(0)==1;
% Solve the ODE
sol = dsolve(ode,cond);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q2d
(3*exp(-3*x))/4 - (exp(-x)*(2*x - 1))/4

>>
```



OBSERVATION

If the dsolvefunction is unable to find a symbolic solution for the given ordinary differential equation (ODE) here question ‘c’, it could be due to the complexity ofthe equation or the absence of explicit symbolic solutions. This commonly occurs when dealing with nonlinear or highly intricate ODEs.

When faced with such a situation, it becomes necessary to resort to numerical methods for solving the ODE. Numerical methods, like the ode45 function in MATLAB, can provide approximate solutions by discretizing the problem and iteratively solving it over a specified range.

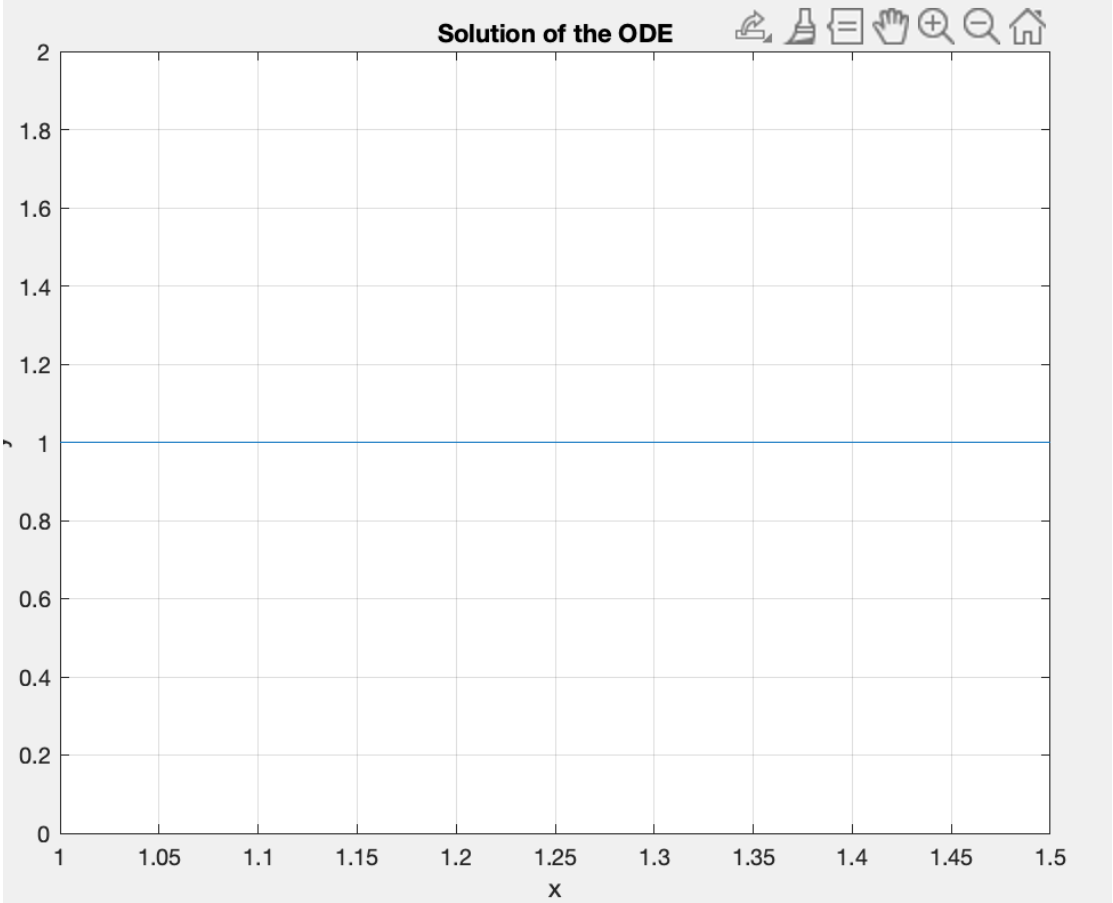
3.	Now try to solve the following ODEs using dsolve. What do you observe? a. $dy + (xy - xy^3)dx = 0$ b. $dy + e^{-y^2}dx = 0$

a) CODE

```
% Define the ODE
syms x y(x)
ode = diff(y) + (x*y - x*y^3)*diff(x) == 0;
cond = y(0) == 1;

% Solve the ODE
sol = dsolve(ode, cond);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q3a
1

>>
```

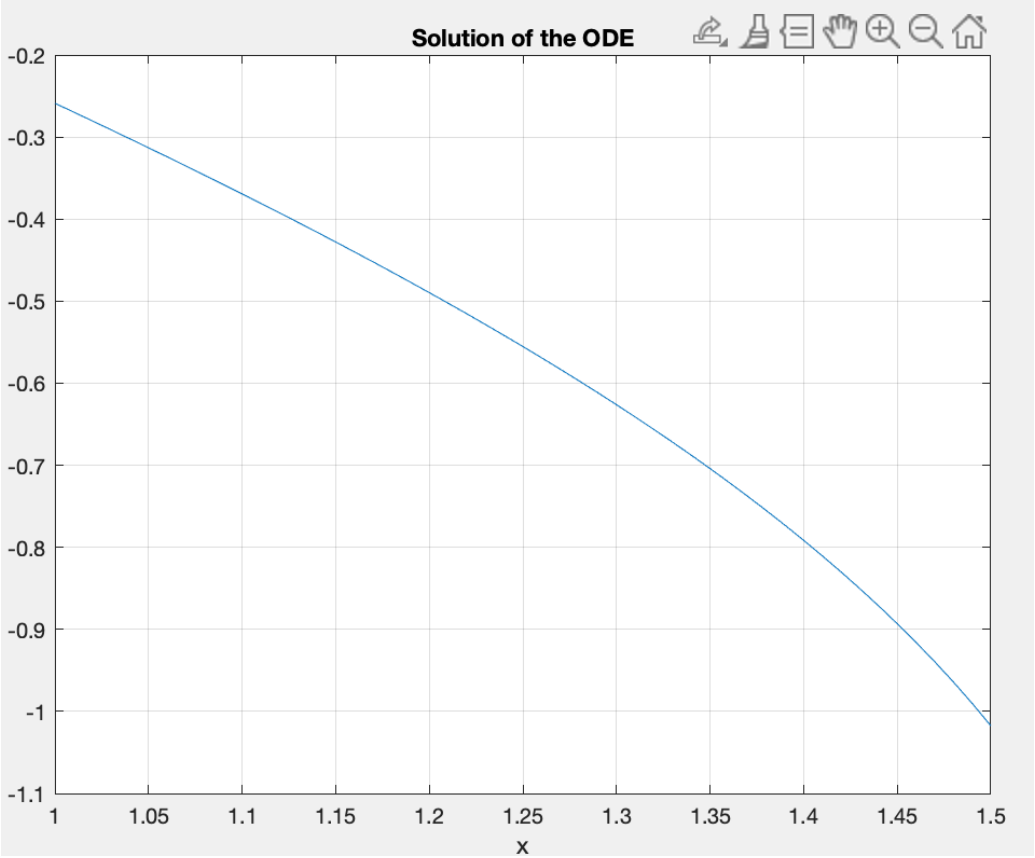


b) **CODE**

```
% Define the ODE
syms x y(x)
ode = diff(y) + exp((-y)^2)*diff(x) == 0;
cond = y(0)==1;
% Solve the ODE
sol = dsolve(ode,cond);

% Display the solution
disp(sol)

% Define the symbolic constant C1 (replace this with the actual constant in your solution)
C1 = 1.5;

% Substitute the constant into the symbolic solution
sol_numeric = subs(sol, 'C1', C1);

% Substitute x values into the symbolic solution
y_range = subs(sol_numeric, x, x_range);

% Plot the solution
figure
plot(x_range, y_range)
title('Solution of the ODE')
xlabel('x')
ylabel('y')
grid on
```

Result

```
>> Q3b
-erfinv((2*(x - (pi^(1/2)*erf(1))/2))/pi^(1/2))

>>
```

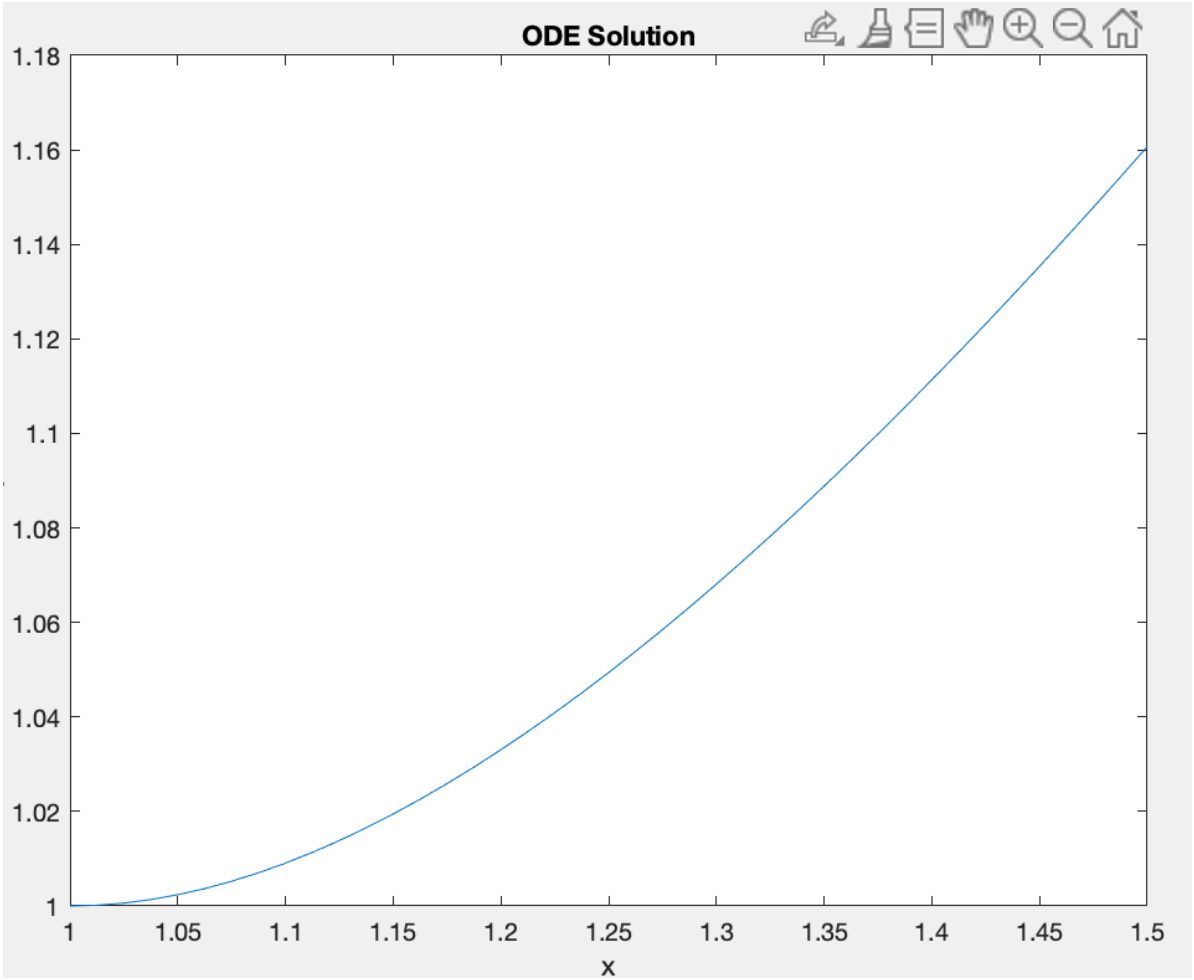


4.	Now solve all the above ODEs (ode45) numerically and plot the solution for 1 to 1.5

a) CODE

```
% Define the ODE
dydx = @(x, y) (x^2 - y^2)/(x*y);

% Define the range of x
xspan = [1 1.5];

% Provide an initial condition at x = 1
y0 = 1; % You can choose any initial value

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

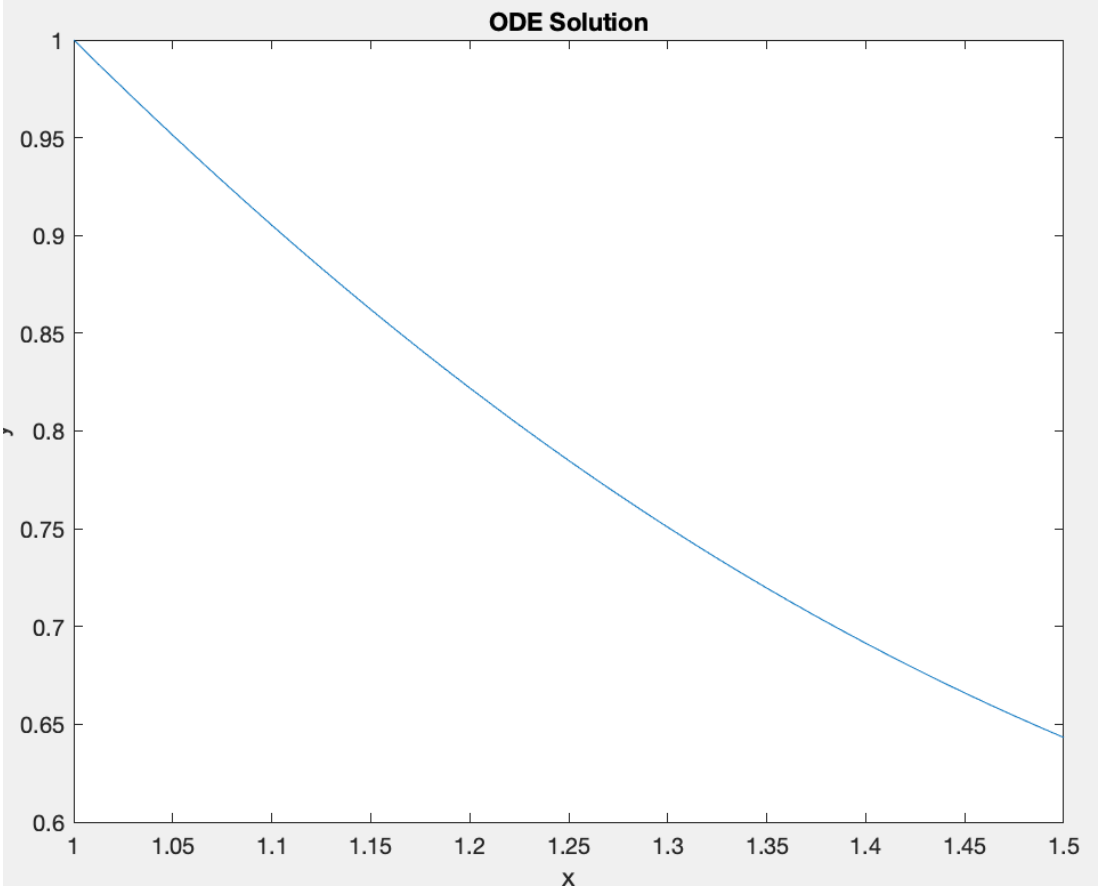
Result



b) **CODE**

```
% Define the ODE
dydx = @(x, y) -(2*x*y - x);

% Define the range of x
xspan = [1 1.5];

% Provide an initial condition at x = 1
y0 = 1; % You can choose any initial value

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

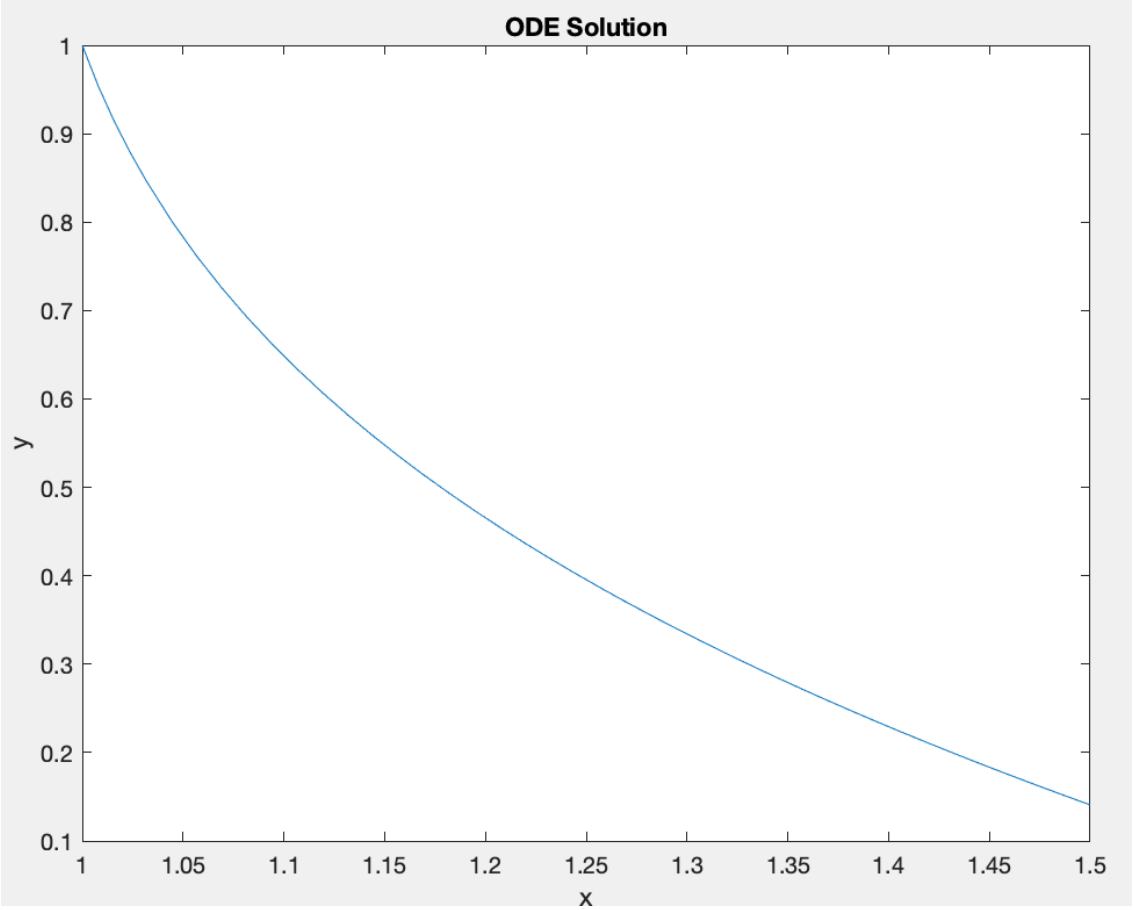
Result



c) **CODE**

```
% Define the ODE
dydx = @(x, y) (cos(x) - cos(y) - y) / (x * (-sin(y) + 1));

% Define the range of x
xspan = [1 1.5];

% Provide an initial condition at x = 1
y0 = 1; % You can choose any initial value

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

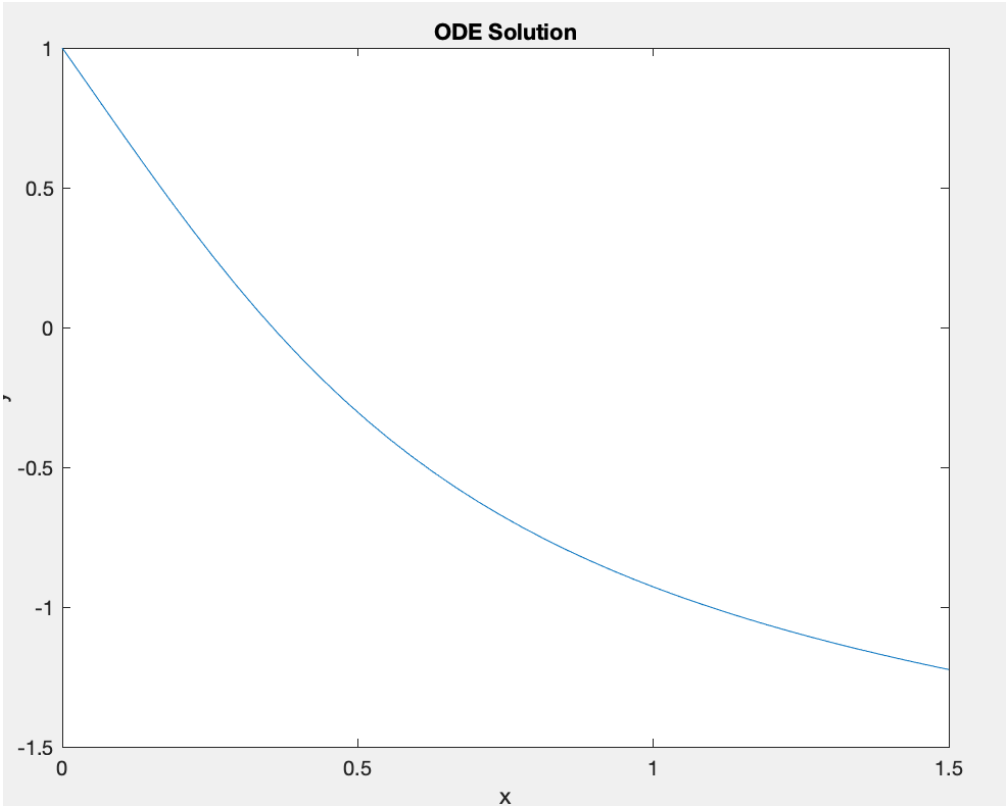
Result



d) **CODE**

```
% Define the ODE
dydx = @(x, y) -(3 * exp(x * y) + x) / exp(x);

% Define the range of x
xspan = [0 1.5];

% Provide the initial condition at x = 0
y0 = 1;

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

Result

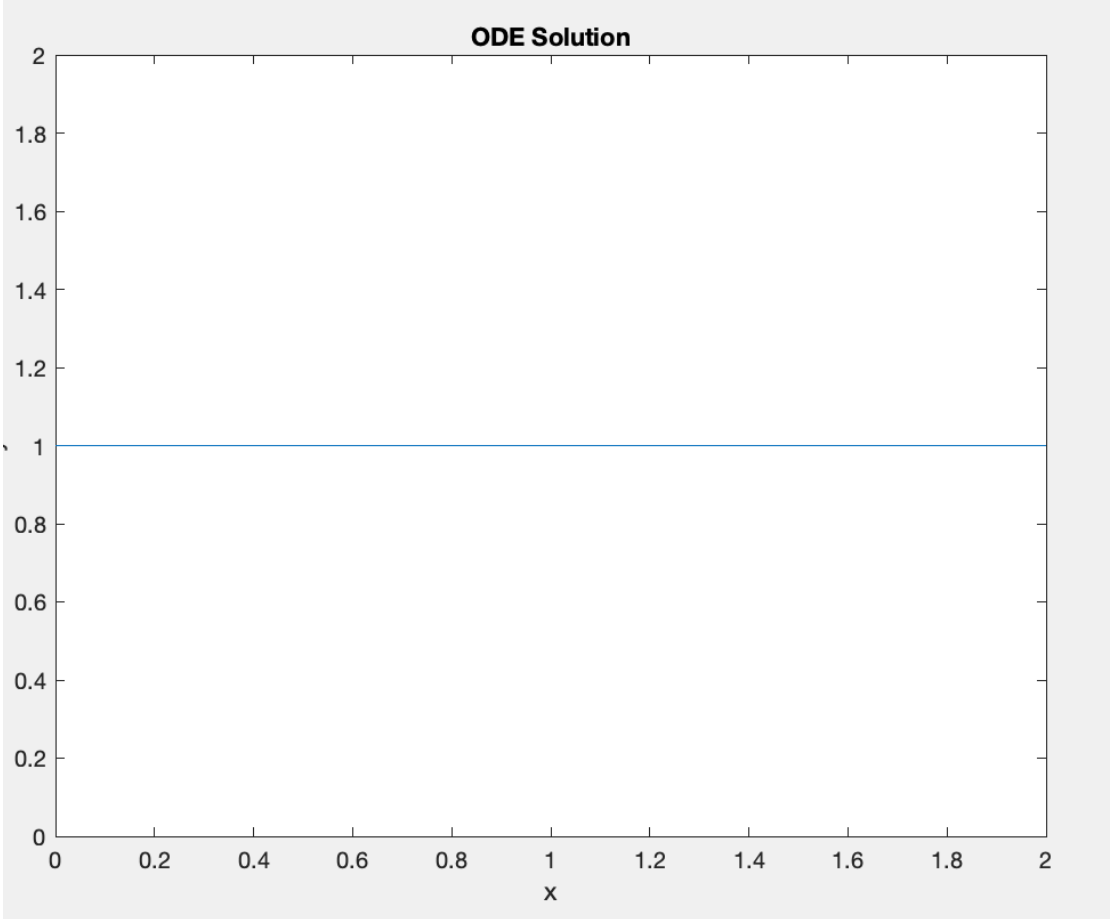


e)

CODE

```
% Define the ODE
dydx = @(x, y) -(x*y - x*y^3);

% Define the range of x
xspan = [0 2];

% Provide an initial condition at x = 0
y0 = 1;

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

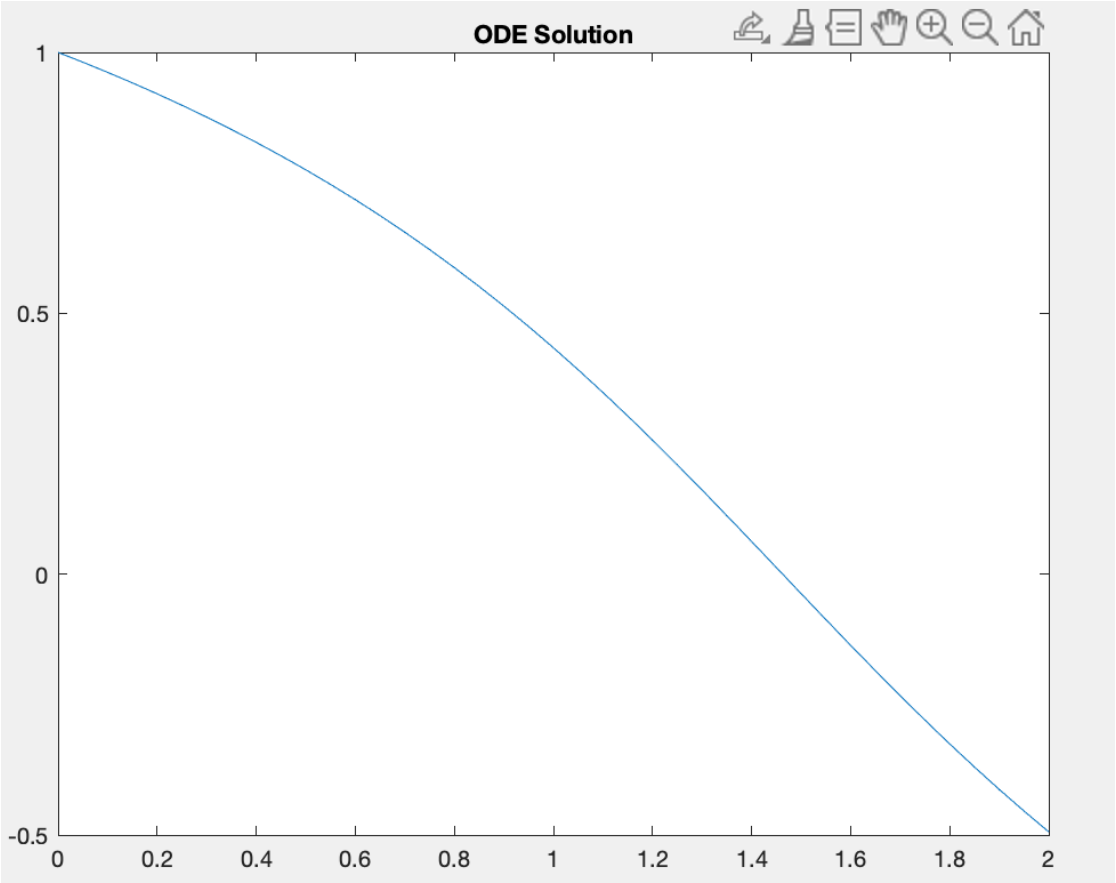
Result



f) **CODE**

```
% Define the ODE
dydx = @(x, y) -exp(-y^2);

% Define the range of x
xspan = [0 2];

% Provide an initial condition at x = 0
y0 = 1;

% Solve the ODE using ode45
[xSol, ySol] = ode45(dydx, xspan, y0);

% Plot the solution
plot(xSol, ySol);
xlabel('x');
ylabel('y');
title('ODE Solution');
```

Result

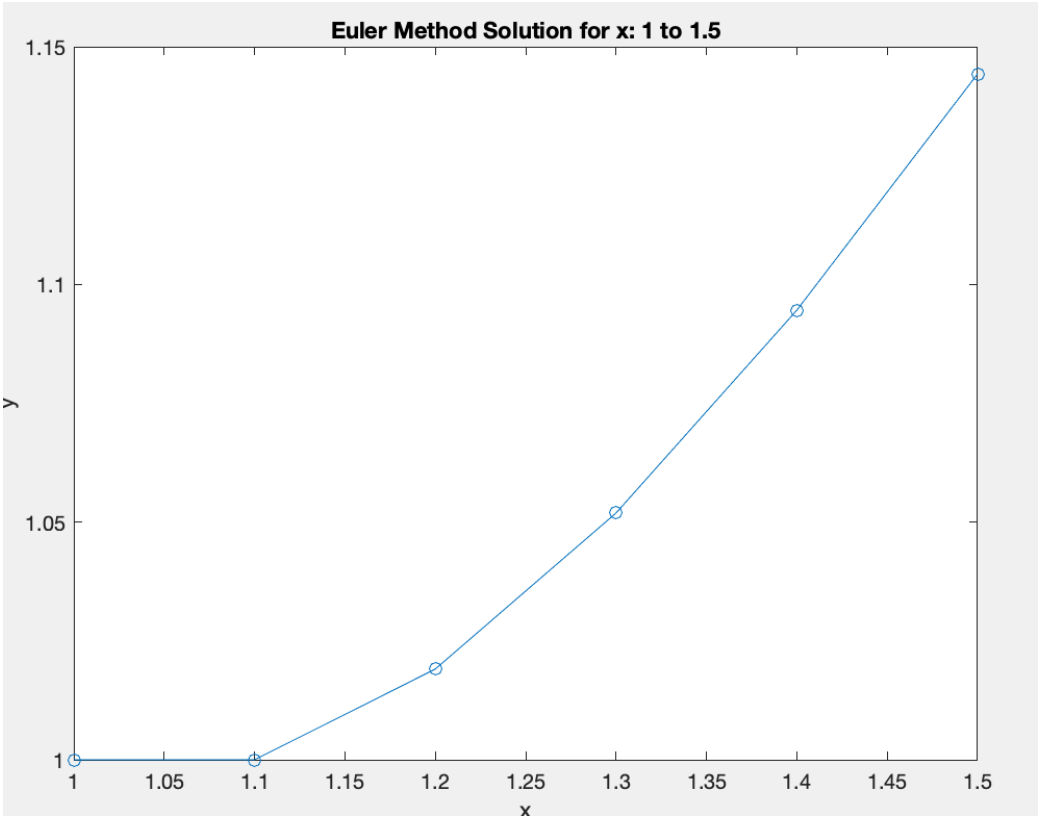


5.	Implement Euler and RK methods to solve ODEs numerically and use these functions to plot the solution for the range x: 1 to 1.5

Euler Method

```
% Define the ODE function
f = @(x, y) (x^2 - y^2)/(x*y);

% Define the x range and initial condition
x_range = [1, 1.5]; % Adjusted x_range to [1, 1.5]
y0 = 1;

% Choose a step size
step_size = 0.1;

% Call the Euler method function
[x_values, y_values] = my_euler_method(f, x_range, y0, step_size);

% Plot the results
plot(x_values, y_values, '-o');
xlabel('x');
ylabel('y');
title('Euler Method Solution for x: 1 to 1.5');

% Euler method function
function [x_values, y_values] = my_euler_method(f, x_range, y0, step_size) % Renamed function
    x_values = x_range(1):step_size:x_range(2);
    y_values = zeros(size(x_values));

    % Set initial condition
    y_values(1) = y0;

    % Euler method iteration
    for i = 1:length(x_values)-1
        % Update y using Euler method
        y_values(i+1) = y_values(i) + step_size * f(x_values(i), y_values(i));
    end
end
```

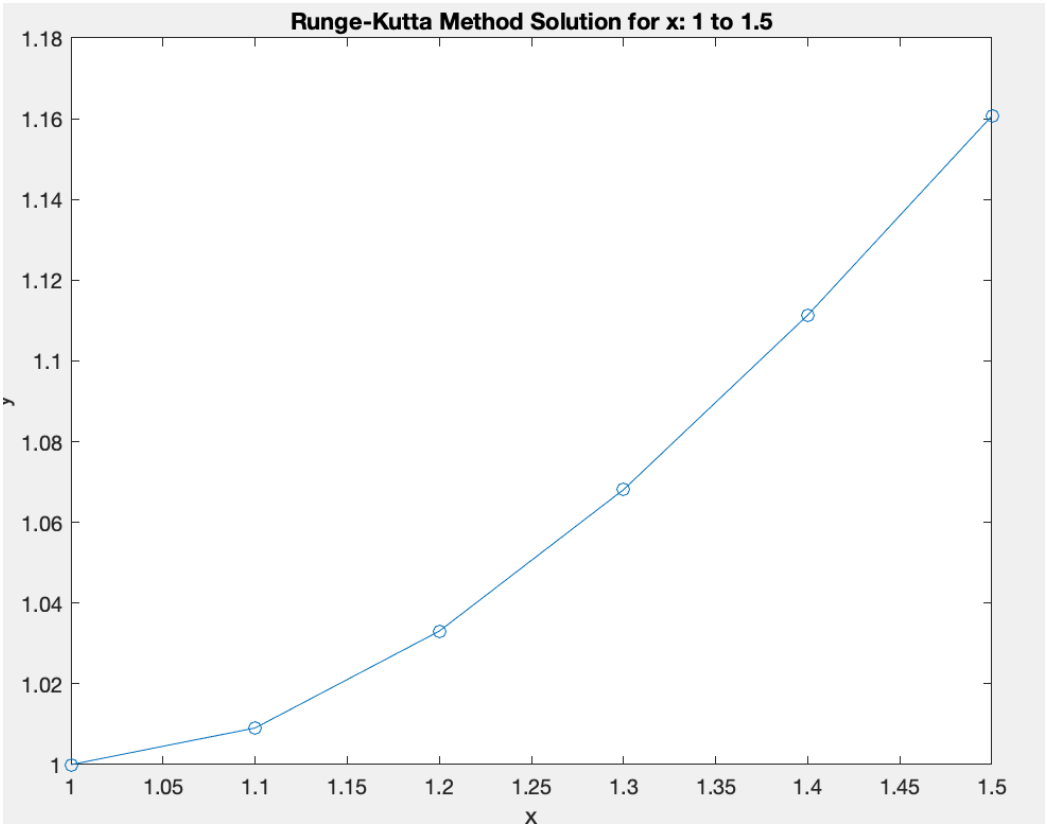
Result



RK Method

```
% Define the ODE function
f = @(x, y) (x^2 - y^2)/(x*y);

% Define the x range and initial condition
x_range = [1, 1.5];
y0 = 1;

% Choose a step size
step_size = 0.1;

% Call the Runge-Kutta method function
[x_values, y_values] = my_runge_kutta_method(f, x_range, y0, step_size);

% Plot the results
plot(x_values, y_values, '-o');
xlabel('x');
ylabel('y');
title('Runge-Kutta Method Solution for x: 1 to 1.5');

% Runge-Kutta method function
function [x_values, y_values] = my_runge_kutta_method(f, x_range, y0, step_size)
    x_values = x_range(1):step_size:x_range(2);
    y_values = zeros(size(x_values));

    % Set initial condition
    y_values(1) = y0;

    % Runge-Kutta method iteration
    for i = 1:length(x_values)-1
        % Update y using Runge-Kutta method
        k1 = step_size * f(x_values(i), y_values(i));
        k2 = step_size * f(x_values(i) + step_size/2, y_values(i) + k1/2);
        k3 = step_size * f(x_values(i) + step_size/2, y_values(i) + k2/2);
        k4 = step_size * f(x_values(i) + step_size, y_values(i) + k3);

        y_values(i+1) = y_values(i) + (k1 + 2*k2 + 2*k3 + k4)/6;
    end
end
```

Result



6.	What is radioactive decay? Formulate and solve the ordinary differential equation corresponding to radioactive decay? Write a Matlab program that takes the % of reduction is C^{14} in an animal fossil to predict the age of the fossil. Now write short notes on the following: 1. How will you assess the current quantity of C^{14} present in the fossil? 2. How will you assess the quantity of fossil available in the fossil when the animal has died?

CODE

```
%% Write a Matlab program that takes the % of reduction is C 14 in an animal fossil to predict the age of the fossil.

% Define the half-life of Carbon-14
half_life = 5730;

% Input the percentage of reduction in C14
reduction_percent = input('Enter the percentage of reduction in C14: ');

% Calculate the number of half-lives that have passed
num_half_lives = (-log(1 - reduction_percent/100)) / log(2);

% Calculate the age of the fossil
age = num_half_lives * half_life;

% Display the age of the fossil
fprintf('The estimated age of the fossil is %.2f years.\n', age);
```

Result

```
Enter the percentage of reduction in C14: 20
The estimated age of the fossil is 1844.65 years.
>>
```

write a short note on How will you assess the current quantity of C14 present in the fossil?

Ans:

Fossils reveal their age through C14, a radioactive timer. Sensitive machines directly measure its remaining whisper, or stable isotopes hint at its past abundance. But beware, contamination and environment can muddle the message. Experts decipher these clues, unlocking ancient secrets one decay at a time.

How will you assess the quantity of fossil available in the fossil when the animal has died?

Ans:

Assessing fossil quantity involves field excavation, estimating volume, utilizing advanced imaging like photogrammetry, conducting laboratory analysis, and employing remote sensing techniques such as LiDAR. These methods help determine the extent, density, and composition of fossil material, aiding in understanding ancient ecosystems and population dynamics.