Bangladesh University of Engineering and Technology

Numerical Technique Laboratory

Numerical Technique Laboratory

EEE 212

Experiments No.: 08 & 09

Name of the Experiments:

- **Experiment 08:** Solutions to Non-linear Equations
- **Experiment 09:** Solutions to Linear Differential Equations

Department: EEE

Section: C1

Group: 01

Student No.: 1406131

1406132

Date of Performance:

Date of Submission:

Experiment 08: Solutions to Non-linear Solutions

Bisection Method

Exercise 1:

Question: Find the real root of the equation d(x) = x5 + x +1using Bisection Method.
X_{low} = -1, x_{up} =0 and ε = selected x tolerance =10^{-4}.

Code:

```
clc , clear all ;
close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Exercise 1
disp('Bisection Method') ;

f = @(x) x.^5+x+1 ;
xlow =  -1 ;
xup = 0 ;
xtol = 10e-4 ;
% calculating Actual value using built in fzero funcion
actual_value = fzero(f,[-1,0]) ;
fprintf('Actual value is : %f \n',actual_value) ;

% Bisection Method Algorithm
xmid = (xlow+xup)/2 ;
ylow = f(xlow) ;
ymid = f(xmid) ;
iteration = 0 ;

while ((xup - xlow)/2) > xtol
    iteration = iteration + 1 ;
    if (ylow*ymid)>0
        xlow = xmid;
    else
        xup = xmid ;
    end
    xmid = (xup + xlow) /2;
    ymid = f(xmid) ;
    iteration = iteration + 1 ;
end

fprintf('Real root of x using Bisection Method : %f\n',xmid) ;
fprintf('Total Iteration number : %d\n',iteration) ;
```

Command Window Output:

```
Command Window
Bisection Method
Actual value is : -0.754878
Real root of x using Bisection Method : -0.754883
Total Iteration number : 18
fx >>
```

False Position Method

Exercise 2:

Question: Find the root of the equation d(x) = x5 + x +1using False Position Method. X_{low} = -1, x_{up} = 0 and ε = selected x tolerance =10^{-4}.

Code:

```
clc , clear all ;
close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Exercise 2
disp('False Position Method') ;
f = @(x) x.^5+x+1 ;
xlow =  -1 ;
xup = 0 ;
xtol = 10e-4 ;
% calculating Actual value using built in fzero funcion
actual_value = fzero(f,[-1,0]) ;
fprintf('Actual value is : %f \n',actual_value) ;
% False Position Method Algorithm
x = xup - ( (f(xup)*(xlow-xup))/(f(xlow)-f(xup)) ) ;
ylow = f(xlow) ;
y = f(x) ;
iteration = 0 ;
while abs(f(x)) > 10e-10
    if (ylow*y)>0
        xlow = x;
    else
        xup = x ;
    end
    x = xup - ( (f(xup)*(xlow-xup))/(f(xlow)-f(xup)) );
    y = f(x) ;
    iteration = iteration + 1 ;
end
fprintf('Real root of x using False Position Method : %f\n',x) ;
fprintf('Total Iteration number : %d\n',iteration) ;
```

Command Window Output:

```
Command Window
False Position Method
Actual value is : -0.754878
Real root of x using False Position Method : -0.754878
Total Iteration number : 20
fx >>
```

Newton Raphson Method

Exercise 3:

Question: Use the Newton Raphson method to estimate the root of f (x) = $e^{-x} - 1$, employing an initial guess of $x_0 = 0$. The tolerance is = 10^{-4}.

Code:

```
clc , clear all ;
close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exercise 03
disp('Newton Raphson Method') ;
% defining function
f = @(x) exp(-x)-1 ;
d = @(x) -exp(-x) ;
% calculating Actual value using built in fzero funcion
actual_value = fzero(f,0) ;
fprintf('Actual value is : %d \n',actual_value) ;

% setting initail guess
x0 = 0 ;
tol = 10e-8 ;
iteration = 0 ;
root = x0 - ( f(x0)/d(x0) ) ;
while abs((f(x0)/d(x0))) > tol
    x0 = root;
    root = x0 - (f(x0)/d(x0)) ;
    iteration = iteration + 1 ;
end
fprintf('Real root of x using Newton Raphson Method : %d\n',root) ;
fprintf('Total Iteration number : %d\n',iteration) ;
```

Command Window Output:

```
Command Window
Newton Raphson Method
Actual value is : 0
Real root of x using Newton Raphson Method : 0
Total Iteration number : 0
fx >>
```

Remark:

Here, root itself has been given in the question as initial guess and that's why total iteration number is 0.

The secant Method

Exercise 4:

Question: Find the root of the equation f (x) = 3x + sin(x) − e^x, starting values are 0 and 1. The tolerance limit is 0.0000001.

Code:

```
clc , clear all ;
close all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Exercise 4
% The Secant Method
disp('The Secant Method') ;
f = @(x) 3*x + + sin(x) - exp(x) ;

% calculating Actual value using built in fzero funcion
actual_value = fzero(f,[0,1]) ;
fprintf('Actual value is : %f \n',actual_value) ;
% setting initial guesses
xk = 0 ;
xkminus = 1 ;
tol = 10e-7 ;
iteration = 1 ;
yk = f(xk) ;
ykminus = f(xkminus) ;
root = (xkminus*yk-xk*ykminus)/(yk-ykminus) ;
ykplus = f(root) ;
```

```
while abs(root - xk) > tol
    xkminus = xk ;
    ykminus = yk ;
    xk = root ;
    yk = ykplus ;
    root = (xkminus*yk-xk*ykminus)/(yk-ykminus) ;
    ykplus = f(root) ;
    iteration = iteration + 1 ;
end

fprintf('Real root of x using the secant method : %f\n',root) ;
fprintf('Total Iteration number : %d\n',iteration) ;
```

Command Window Output:

```
Command Window
The Secant Method
Actual value is : 0.360422
Real root of x using the secant method : 0.360422
Total Iteration number : 6
>>
```

Experiment 09: Solutions to Linear Differential Equations

Euler's Method

Exercise 1:

Question: Use Euler's method to solve the IVP 2, y' = $\frac{x-y}{2}$ on [0, 3] with y(0) =1. Compare solutions for h = 1, 1/2/, 1/4 and 1/8.
The exact solution is $y = 3e^{-x/2} - 2 + x$.

Code:

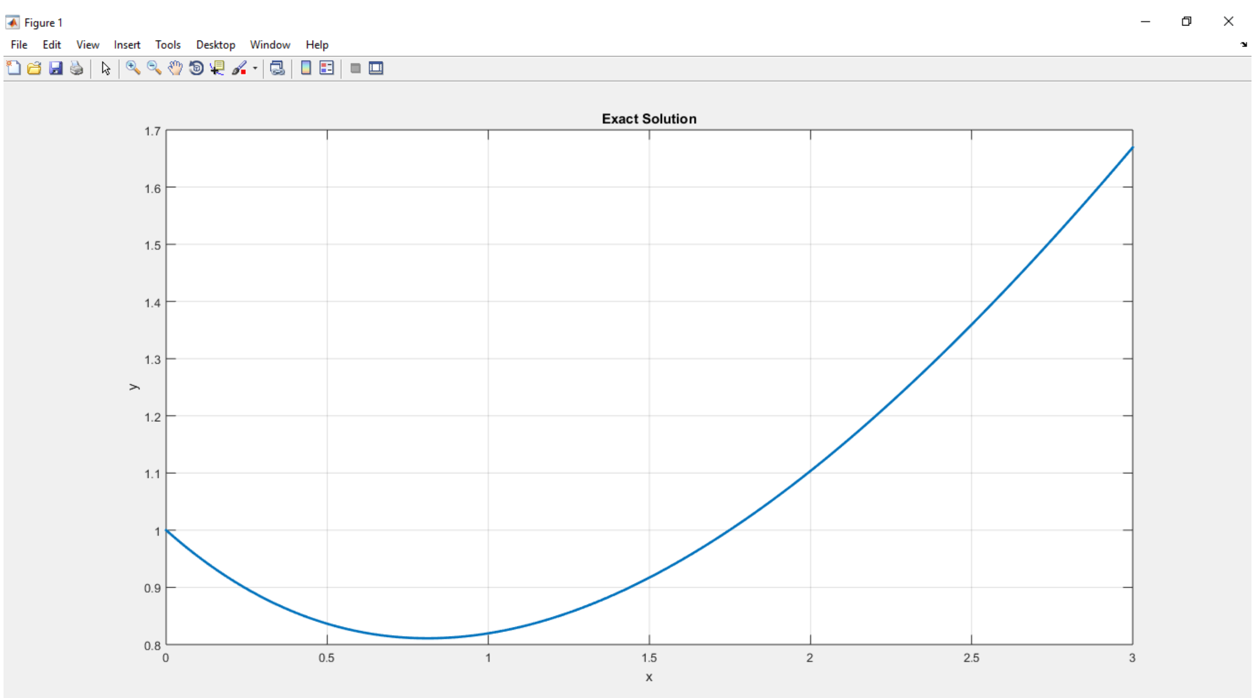
```
clc , close all ;
clf , clear all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Euler's Method
f = @(x,y) (x-y)/2 ;
% Plotting actual value using exact solution
actual_ans = @(x) 3*exp(-x/2) - 2 + x ;
x = 0 : .01 : 3 ;
y = actual_ans(x) ;
plot(x,y,'LineWidth',2) ;
title('Exact Solution','LineWidth',2) ;
xlabel('x','LineWidth',2) ;
ylabel('y','LineWidth',2) ;
grid on ;
actual_value = actual_ans(3) ;
fprintf('Actual value of y(3) : %f\n\n',actual_value) ;
% Code for Euler's Method to solve ODE
for k = 1 : 4           % Loop for changing step size
    step_size = [1 1/2 1/4 1/8] ;
    max_value = 3 ;
    n = max_value / step_size(k) ;
    x = [] ;
    y = [] ;
    % Initial value
    x(1) = 0 ;
    y(1) = 1 ;
    for i = 1 : n ;
        y(i+1) = y(i) + f(x(i),y(i)) * step_size(k) ;
        x(i+1) = x(i) + step_size(k) ;
    end
    figure ;
    plot(x,y,'LineWidth',2) ;
    title('Solution using Euler Method','LineWidth',2) ;
    xlabel('x') ;
    ylabel('y') ;
    % adding step size to the figure
    string = num2str(step_size(k)) ;
    string = [ 'Step size : ' string ] ;
    text(1.33,1.3,string,'FontSize',12) ;
    grid on ;
```

```
    fprintf('For step size %.3f y(3) = %f\n',step_size(k),y(i+1)) ;
    error = (abs(actual_value-y(i+1))/actual_value) * 100 ;
    fprintf('Percentage of error %.2f%%\n\n',error) ;
end
```

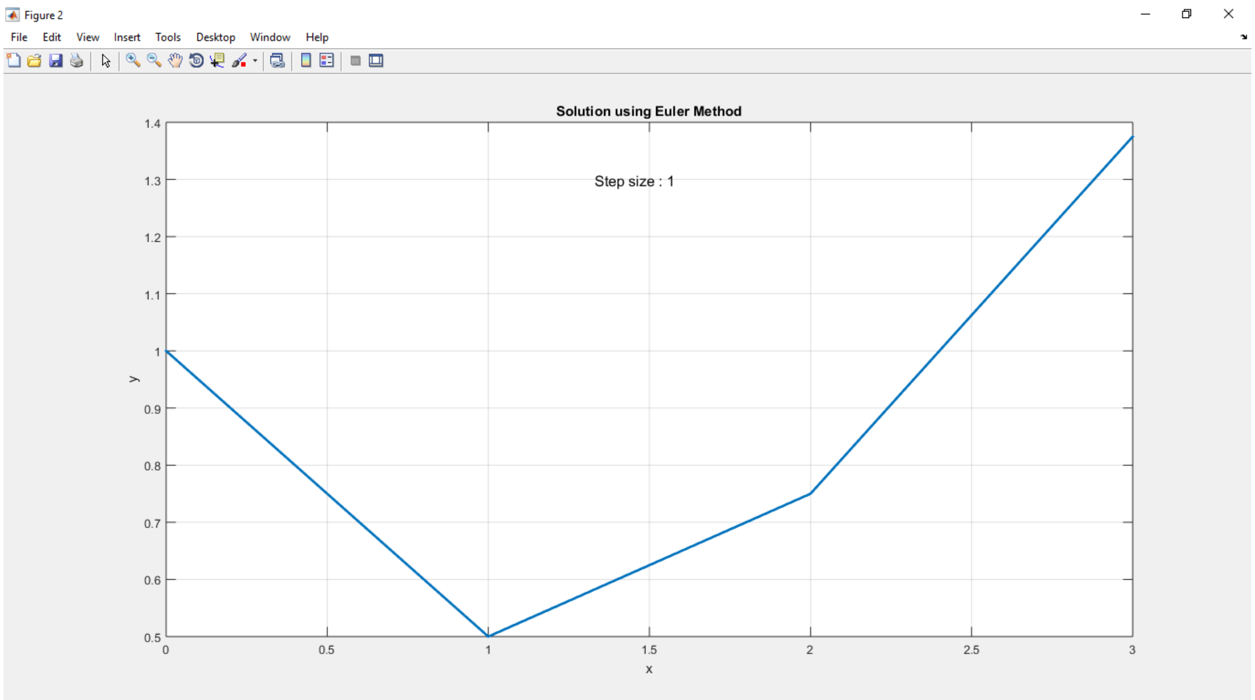
Command Window Output:

```
Command Window
Actual value of y(3) : 1.669390

For step size 1.000 y(3) = 1.375000
Percentage of error 17.63%

For step size 0.500 y(3) = 1.533936
Percentage of error 8.11%

For step size 0.250 y(3) = 1.604252
Percentage of error 3.90%

For step size 0.125 y(3) = 1.637429
Percentage of error 1.91%

fx >>
```

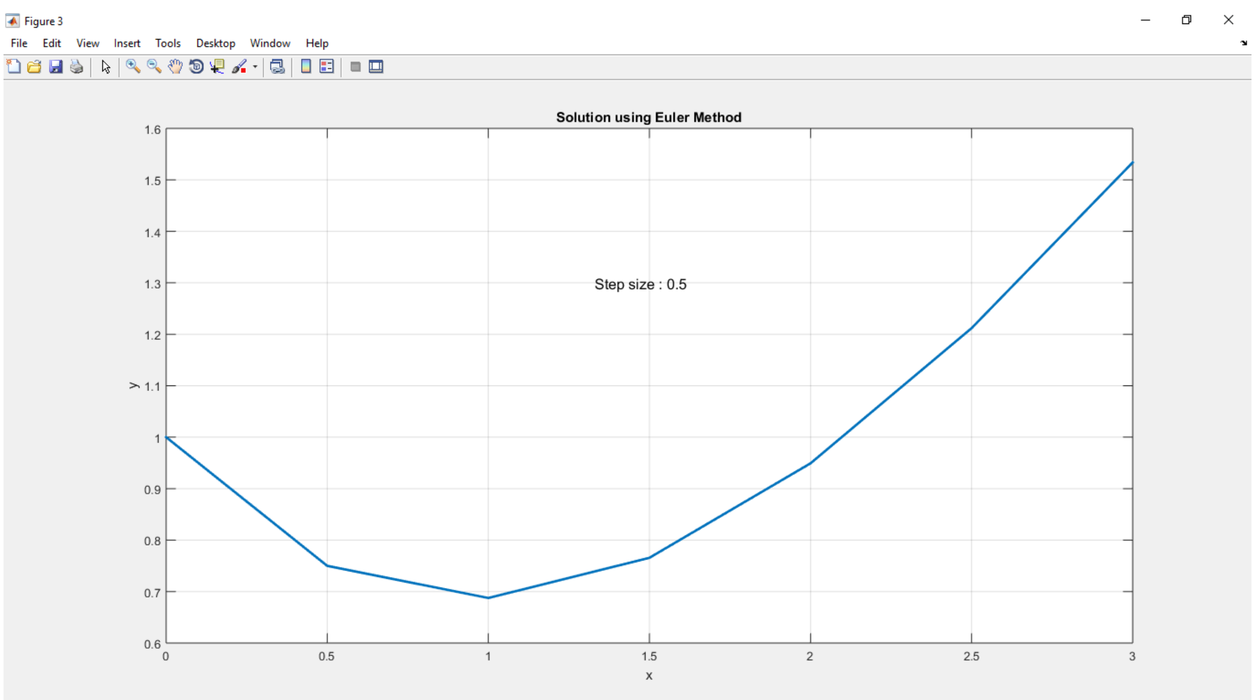
Plot Output: Exact solution



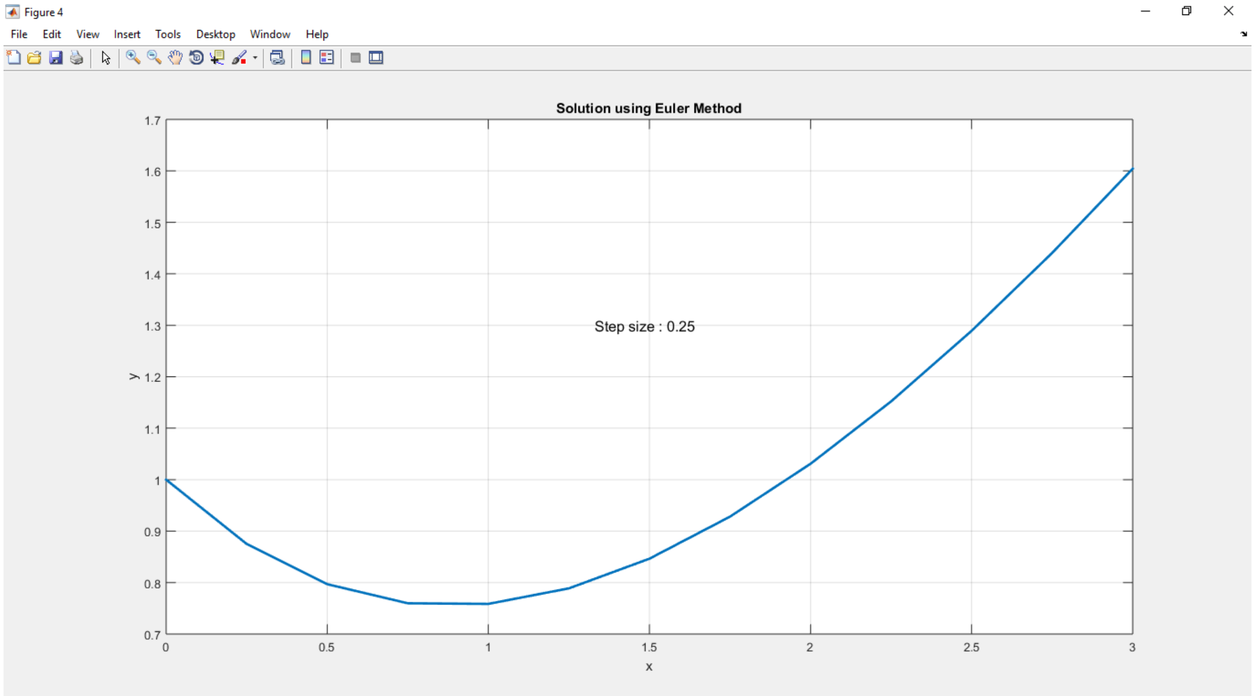
For step size = 1



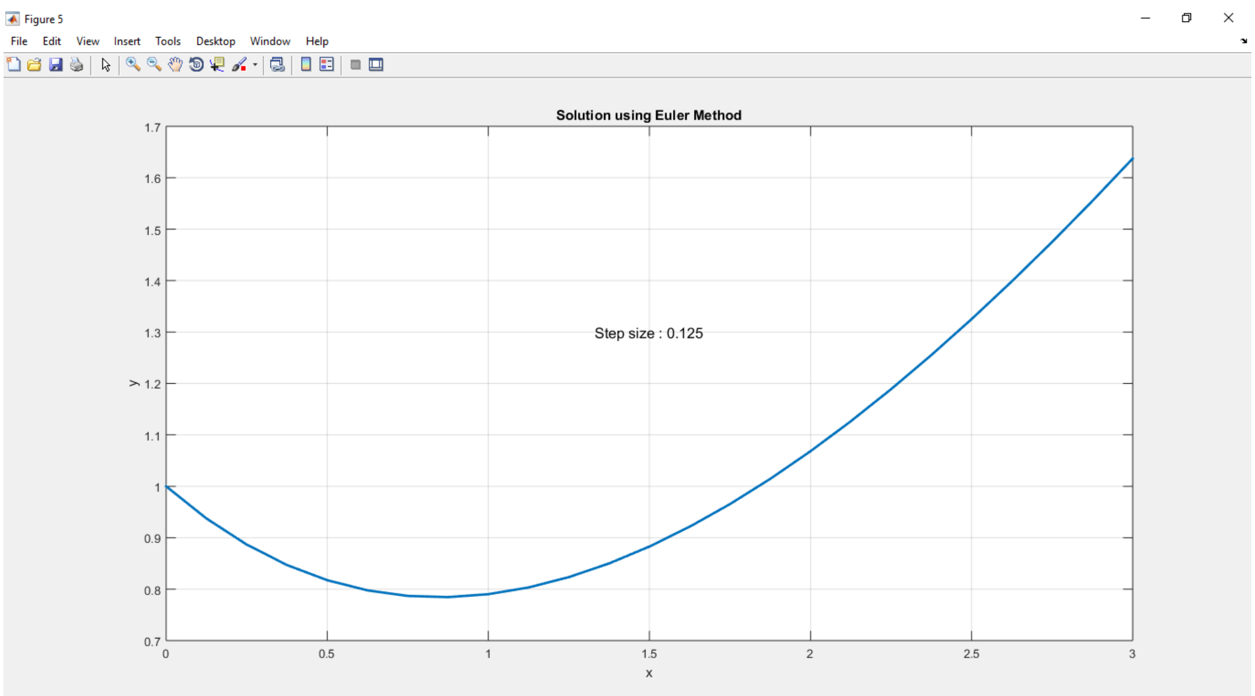
For step size = ½



For step size = ¼

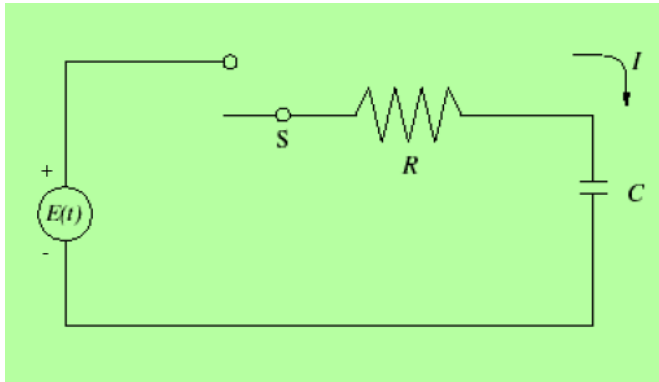


For step size = 1/8



Exercise 2:

Question: Consider the following circuit:



In this circuit, R = 20KΩ, C = 10μ F, E = 117V and Q(0)=0. Find Q for t = 0 to t = 3sec.

Here, linear differential equation is: $\frac{dq}{dt} = \frac{E}{R} - \frac{Q}{RC}$ and its exact solution is:

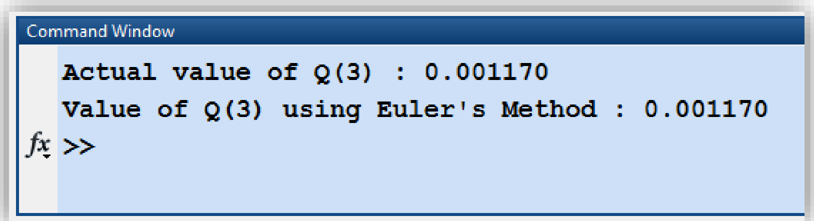
$Q(t) = EC\ (1\text{-}e^{-t/\tau})$ where $\tau = RC$

Code:

```
clc , close all ;
clf , clear all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Given
R = 20000 ;
C = 10e-6 ;
E = 117 ;

f = @(t,q) (E/R - q/(R*C)) ;
% Plotting actual value using actual answer
tau = R*C ;
actual_ans = @(t) E*C*(1-exp(-t/tau)) ;
t = 0 : .01 : 3 ;
q = actual_ans(t) ;
plot(t,q,'LineWidth',2) ;
title('Exact Solution','LineWidth',2) ;
xlabel('t','LineWidth',2) ;
ylabel('Q','LineWidth',2) ;
grid on ;
figure ;
% calculating actual value
actual_value = actual_ans(3) ;
fprintf('Actual value of Q(3) : %f\n',actual_value) ;

% Euler Method
t(1) = 0 ;
```

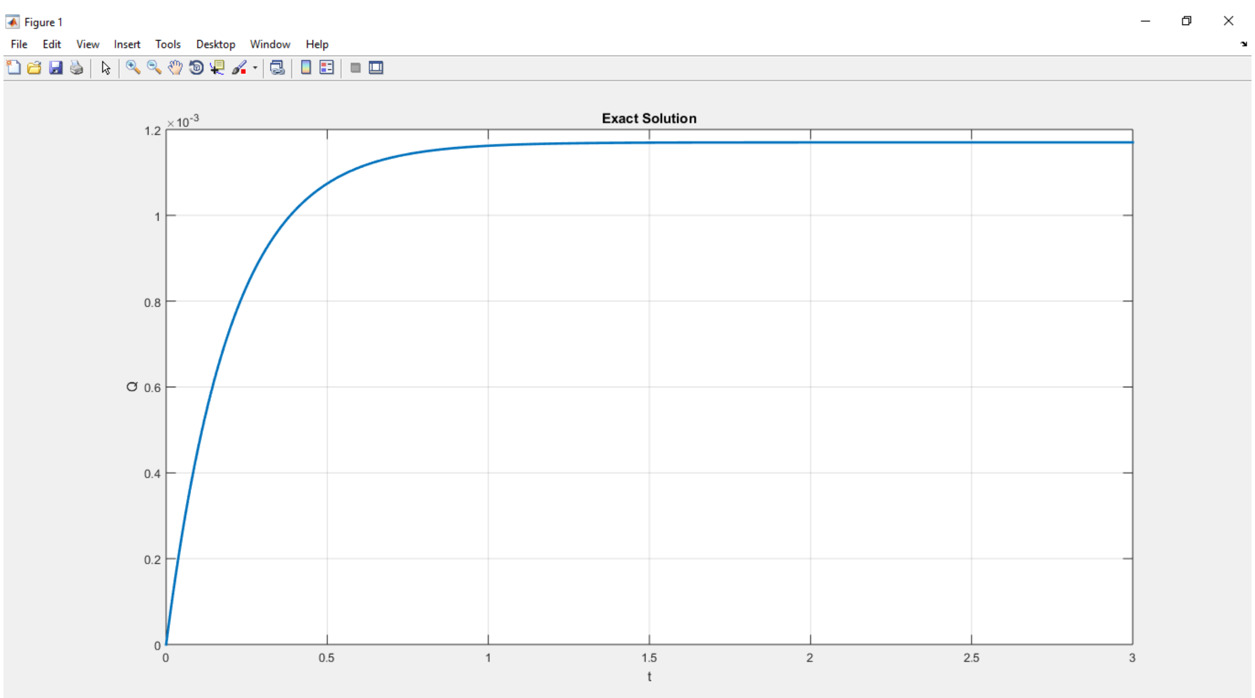
```
q(1) = 0 ;
max_value = 3 ;
step_size = .01 ;
n = max_value / step_size ;
for i = 1 : n
    q(i+1) = q(i) + f(t(i),q(i)) * step_size ;
    t(i+1) = t(i) + step_size ;
end
plot(t,q,'LineWidth',2) ;
title('Solution using Euler Method','LineWidth',2) ;
xlabel('t','LineWidth',2) ;
ylabel('Q','LineWidth',2) ;
grid on ;
fprintf('Value of Q(3) using Euler''s Method : %f\n',q(i+1)) ;
```

Command Window Output:

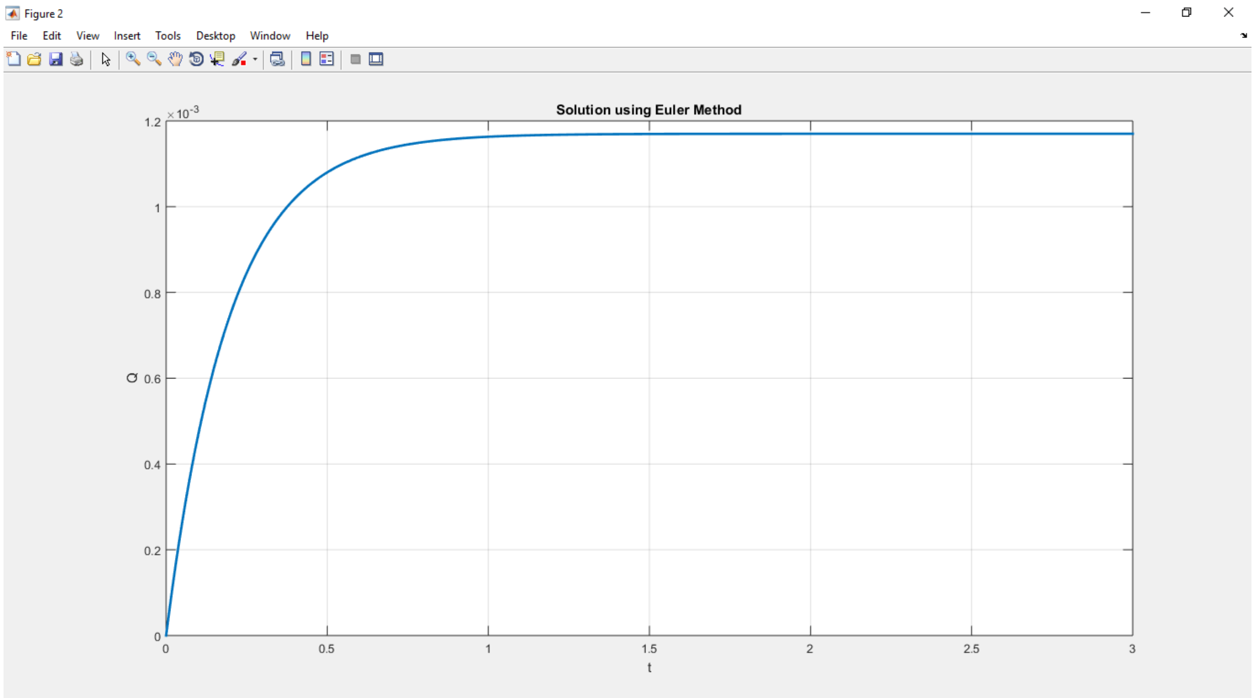


Plot Output:

Exact solution:



Solution using Euler Method:



Improved Euler's Method

Exercise 3:

Question: Solve exercise 1 and 2 using Improved Euler's method.

Exercise 1 Code:

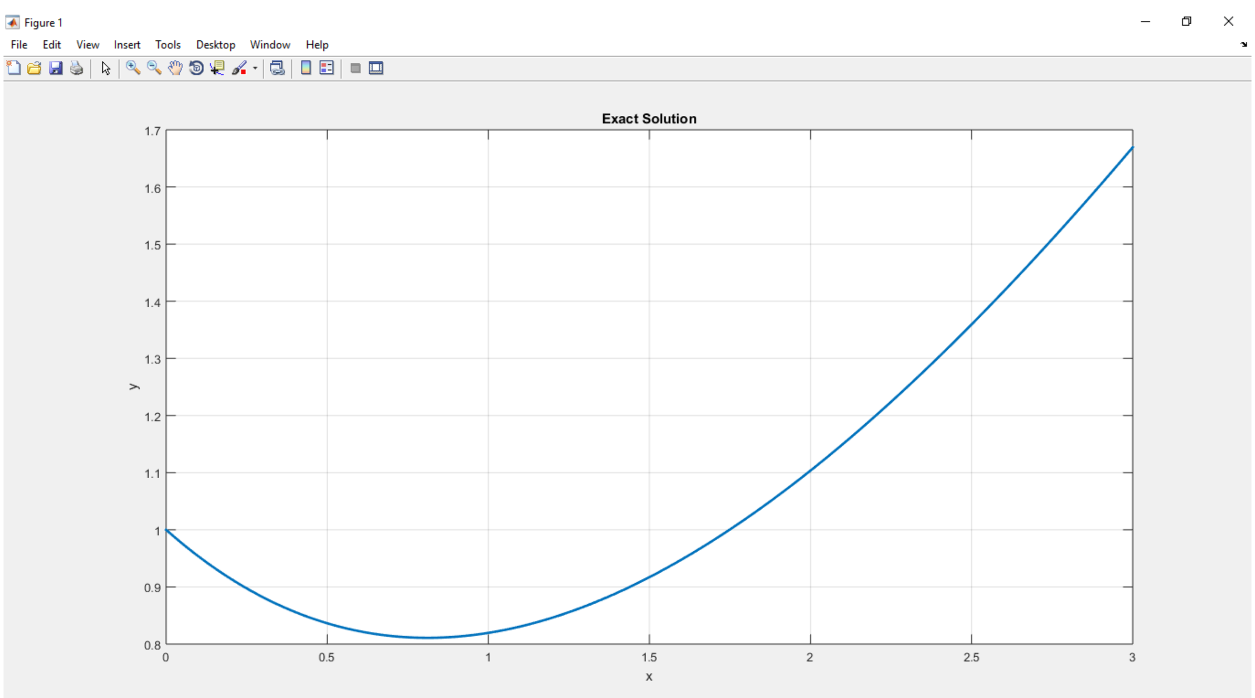
```
clc , close all ;
clf , clear all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Improved Euler's Method
f = @(x,y) (x-y)/2 ;
% Plotting actual value using exact solution
actual_ans = @(x) 3*exp(-x/2) - 2 + x ;
x = 0 : .01 : 3 ;
y = actual_ans(x) ;
plot(x,y,'LineWidth',2) ;
title('Exact Solution','LineWidth',2) ;
xlabel('x','LineWidth',2) ;
ylabel('y','LineWidth',2) ;
grid on ;
actual_value = actual_ans(3) ;
fprintf('Actual value of y(3) : %f\n\n',actual_value) ;

% Code for Euler's Method to solve ODE
for k = 1 : 4           % Loop for changing step size
    step_size = [1 1/2 1/4 1/8] ;
    max_value = 3 ;
    n = max_value / step_size(k) ;
    x = [] ;
    y = [] ;
    % Initial value
    x(1) = 0 ;
    y(1) = 1 ;
    for i = 1 : n ;
        p = y(i) + step_size(k) * f(x(i),y(i)) ;
        x(i+1) = x(i) + step_size(k) ;
        y(i+1) = y(i) + step_size(k) * .5 * (f(x(i),y(i))+f(x(i+1),p)) ;
    end
    figure ;
    plot(x,y,'LineWidth',2) ;
    title('Solution using Improved Euler Method','LineWidth',2) ;
    xlabel('x') ;
    ylabel('y') ;
    % adding step size to the figure
    string = num2str(step_size(k)) ;
    string = [ 'Step size : ' string ] ;
    text(1.33,1.3,string,'FontSize',12) ;
    grid on ;
    fprintf('For step size %.3f y(3) = %f\n',step_size(k),y(i+1)) ;
    error = (abs(actual_value-y(i+1))/actual_value) * 100 ;
    fprintf('Percentage of error %.2f%%\n\n',error) ;
end
```

Command Window Output:

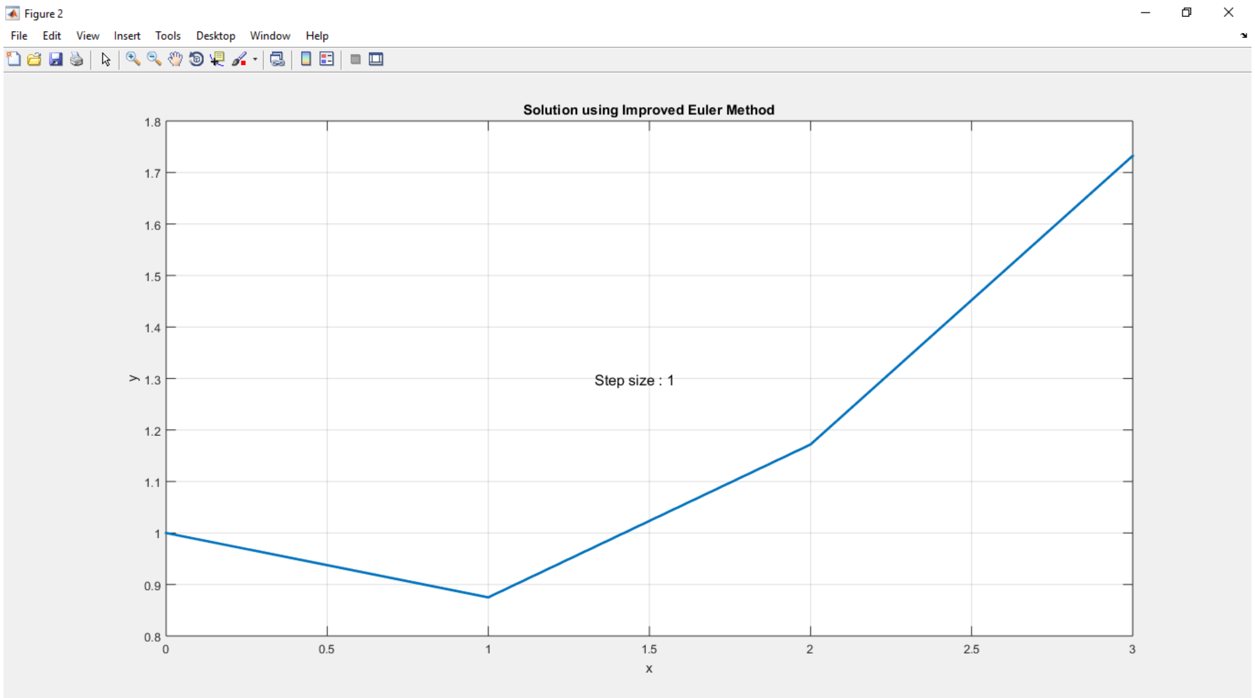
```
Command Window
Actual value of y(3) : 1.669390

For step size 1.000 y(3) = 1.732422
Percentage of error 3.78%

For step size 0.500 y(3) = 1.682121
Percentage of error 0.76%

For step size 0.250 y(3) = 1.672269
Percentage of error 0.17%

For step size 0.125 y(3) = 1.670076
Percentage of error 0.04%

fx >>
```

Error shows that it is more accurate than Euler's Method.

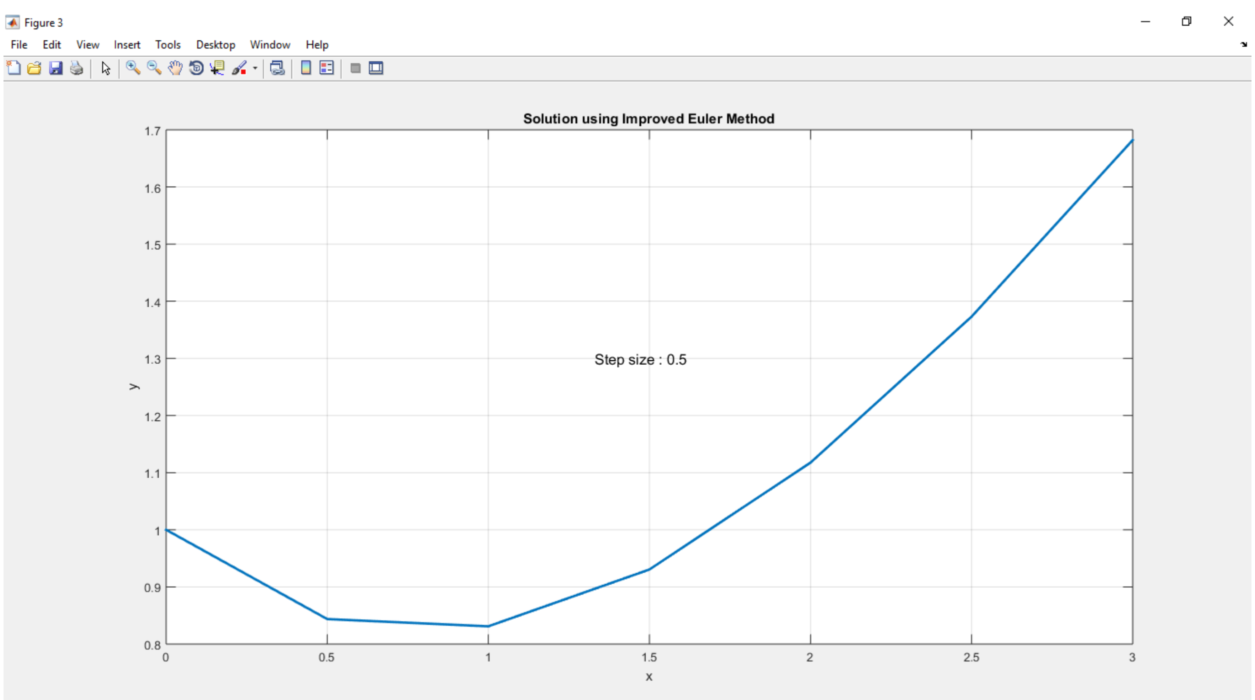
Plot Output:



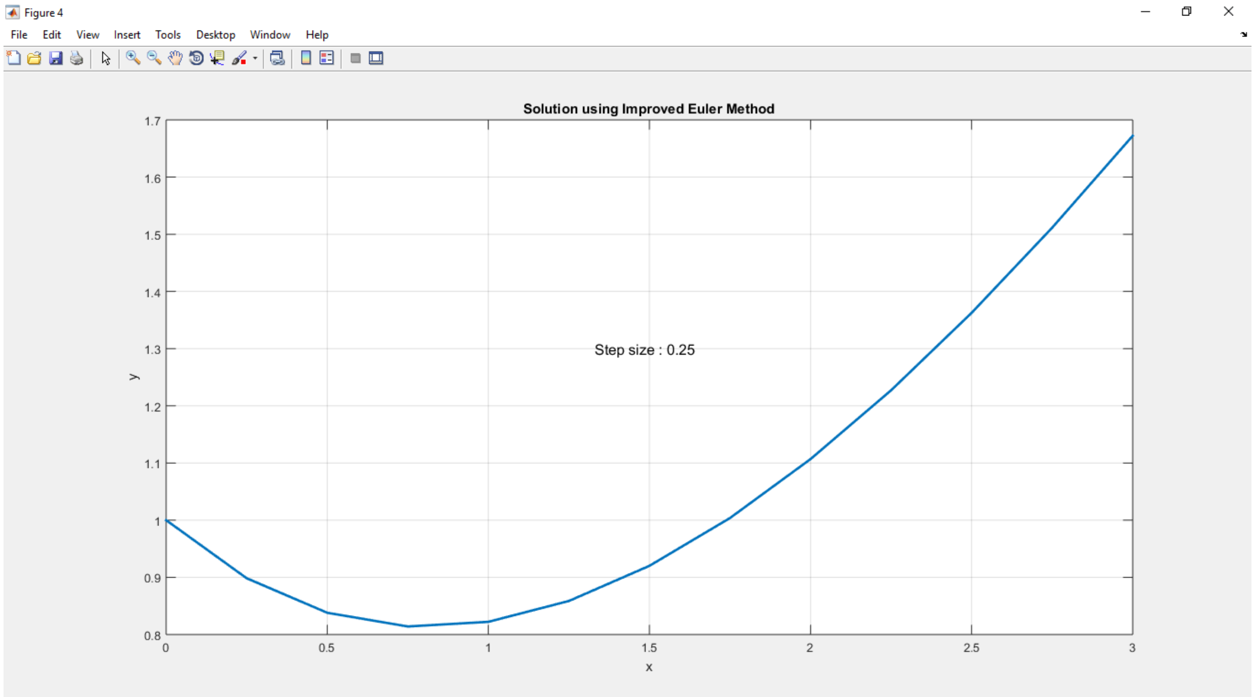
For step size = 1



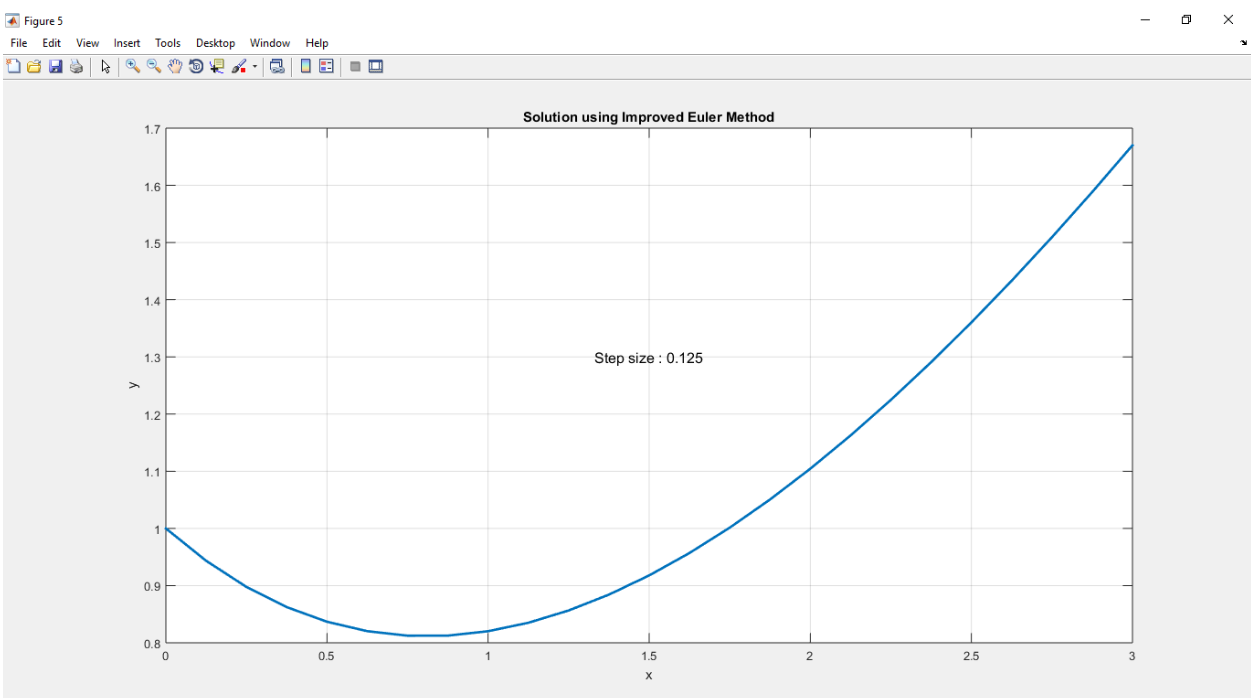
For step size = ½



For step size = ¼



For step size = 1/8



Exercise 2 Code:

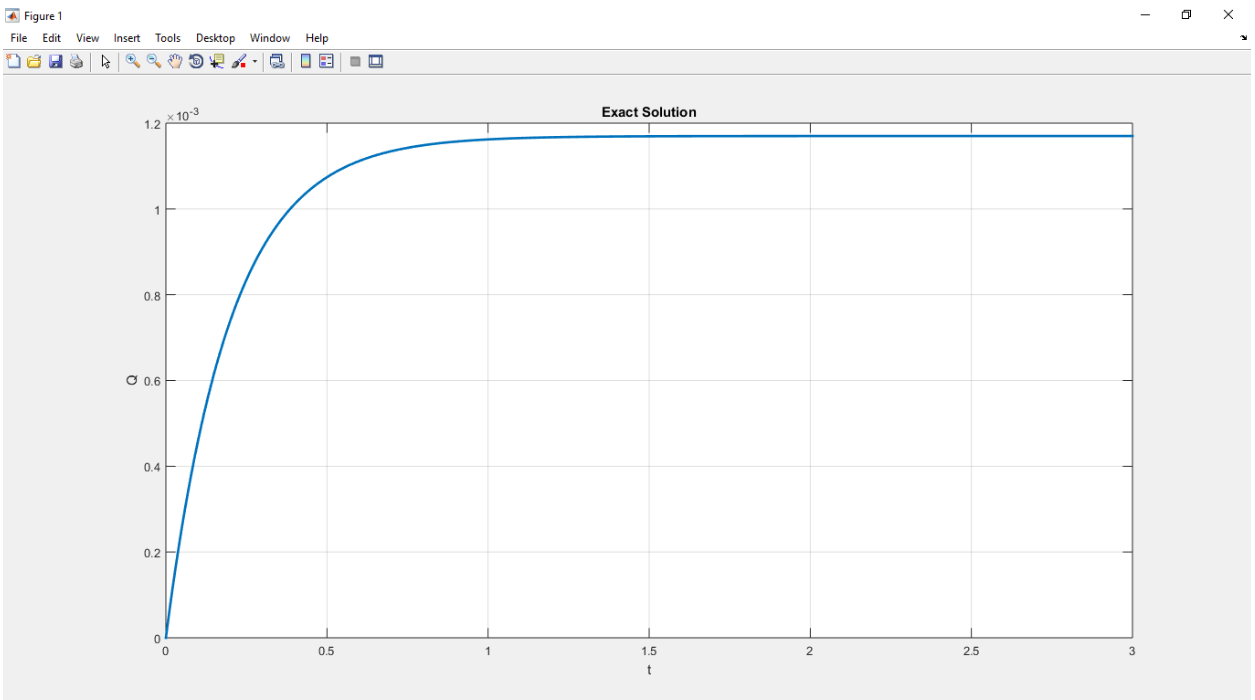
```
clc , close all ;
clf , clear all ;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Given
R = 20000 ;
C = 10e-6 ;
E = 117 ;

f = @(t,q) (E/R - q/(R*C)) ;
% Plotting actual value using actual answer
tau = R*C ;
actual_ans = @(t) E*C*(1-exp(-t/tau)) ;
t = 0 : .01 : 3 ;
q = actual_ans(t) ;
plot(t,q,'LineWidth',2) ;
title('Exact Solution','LineWidth',2) ;
xlabel('t','LineWidth',2) ;
ylabel('Q','LineWidth',2) ;
grid on ;
figure ;
% calculating actual value
actual_value = actual_ans(3) ;
fprintf('Actual value of Q(3) : %f\n',actual_value) ;

% Improved Euler Method
t(1) = 0 ;
q(1) = 0 ;
max_value = 3 ;
step_size = .01 ;
n = max_value / step_size ;
for i = 1 : n
    p = q(i) + step_size * f(q(i),q(i)) ;
    t(i+1) = t(i) + step_size ;
    q(i+1) = q(i) + step_size * .5 * (f(t(i),q(i))+f(t(i+1),p)) ;
end
plot(t,q,'LineWidth',2) ;
title('Solution using Improved Euler''s Method','LineWidth',2) ;
xlabel('t','LineWidth',2) ;
ylabel('Q','LineWidth',2) ;
grid on ;
fprintf('Value of Q(3) using Improved Euler''s Method : %f\n',q(i+1)) ;
```

Command Window Output:

```
Command Window
Actual value of Q(3) : 0.001170
Value of Q(3) using Improved Euler's Method : 0.001170
fx >>
```

Plot Output:

Exact solution:



Solution using Improved Euler's Method: