University of Dhaka

Third Year BS (Honours) 2022-23

Assignment 2A

Subject: Applied Mathematics

Course No.: AMTH 350 Course Title: Math Lab III

Name: Aminul Islam Maruf Roll: SH-123-071 Group: B

1. Consider the following linear system of equations:

$$\begin{aligned} 4x_1 + x_2 + x_3 + x_5 &= 6 \\ -x_1 - 3x_2 + x_3 + x_4 &= 6 \\ -x_1 - x_2 - x_3 + 4x_4 &= 6 \\ 2x_1 + x_2 + 5x_3 - x_4 - x_5 &= 6 \\ 2x_2 - x_3 + x_4 + 4x_5 &= 6 \end{aligned}$$

[illegible] i) First, store the coefficients and constants as *.txt* or *.dat* files. Load those data in two matrices named A and b and then find the exact solution of the system using Gaussian elimination with Gauss Jordan method.

ii) Check whether A is positive definite or not. If so, then solve the above system, correct up-to 5 significant digits within the initial guess $x_0 = 0$ using the following methods:

a) Jacobi

b) Gauss-Seidel

c) SOR with relaxation coefficient $\omega = 1.1$

iii) Create a table showing the comparison of the methods in (b) with headings as follows: "Exact Solution", "Jacobi", "G-S", "SOR", "Total iterations for Jacobi", "Total iterations for G-S", "Total iterations for SOR",

2. [illegible] i) Use the Bisection method, Secant method and Regula-Falsi method to find an approximate root of the equation $x^3 = x + e^x + 2$, in the interval $[2, 3]$ correct up to 6 significant digits with their corresponding errors. Show your results in a table with headings as follows:
"$Iteration$", "a_n", "b_n", "$f(a_n)$", "$f(b_n)$", "x_{rfp}", "x_{secant}", "x_{bisec}", "Abs_Error_rfp", "Abs_Error_secant", "Abs_Error_bisec".

ii) Test the rate of convergence of these methods by plotting the errors in the same graph with respect to the number of iterations. Comment on their convergence.

3. Let T be the total period of the planet, and let t be the time required for the planet to go. Then Kepler's equation from orbital mechanics, relating x and t, is

$$x - \epsilon \sin x = \frac{2\pi t}{T}$$

Here ϵ is the eccentricity of the elliptical orbit (the extent to which it deviates from a circle). For an orbit of eccentricity $\epsilon = 0.01$ (roughly equivalent to that of the Earth),

i) what is the value of x corresponding to $t = \frac{T}{4}$?

ii) What is the value of x corresponding to $t = \frac{T}{8}$?

Use Newton-Raphson method and Fixed point method to solve the required equation.

iii) Divide the interval $t = \left[\frac{T}{8}, \frac{T}{4}\right]$ in 10 equally spaced intervals and show the solution in a suitable table.

iv) Also plot each of the solutions with respect to the number of iterations in a single figure.

University of Dhaka

Fourth Year BS (Honours) 2022-23

Assignment 02B

Subject: Applied Mathematics

Course No.: AMTH 350 Course Title: Math Lab III

Name: Aminul Islam Maruf Roll: SH-123-071 Group: B

1. The study of mathematical models for predicting the population dynamics of competing species has its origin in independent works published in the early part of the 20th century by A. J. Lotka and V. Volterra. Consider the problem of predicting the population of two species, one of which is a predator, whose population at time t is $x_2(t)$, feeding on the other, which is the prey, whose population is $x_1(t)$. We will assume that the prey always has an adequate food supply and that its birth rate at any time is proportional to the number of prey alive at that time; that is, birth rate (prey) is $k_1x_1(t)$. The death rate of the prey depends on both the number of prey and predators alive at that time. For simplicity, we assume death rate (prey) $= k_2x_1(t)x_2(t)$. The birth rate of the predator, on the other hand, depends on its food supply, $x_1(t)$, as well as on the number of predators available for reproduction purposes. For this reason, we assume that the birth rate (predator) is $k_3x_1(t)x_2(t)$. The death rate of the predator will be taken as simply proportional to the number of predators alive at the time; that is, death rate (predator) $= k_4x_2(t)$. Since x_1' and x_2' represent the change in the prey and predator populations, respectively, with respect to time, the problem is expressed by the system of nonlinear differential equations

$$x_1' = k1x1(t) - k2x1(t)x2(t)$$

$$x_2' = k3x1(t)x2(t) - k4x2(t)$$

Solve this system for $0 \leq t \leq 4$, assuming that the initial population of the prey is 1000 and of the predators is 500 and that the constants are $k_1 = 3, k_2 = 0.002, k_3 = 0.0006$ & $k_4 = 0.5$. Sketch a graph of the solutions to this problem, plotting both populations with time, and describe the physical phenomena represented.

[illegible]

2. Consider the initial-value problem

$$y' = \frac{2-2ty}{t^2+1}, \quad 0 \le t \le 1, \quad y(0) = 0.025$$

The analytic solution is given by

$$y(t) = \frac{2t+1}{t^2+2}$$

i) Use the *RK-4* or *Euler* method to obtain starting values and step size $h = 0.05$ to obtain approximations using

a) Adams-Bashforth four step explicit method.

b) Adams-Moulton three step implicit method.

c) Adams fourth order predictor-corrector method.

Show your result in a table with headings as follows:

"t", "Exact", "A-B", "A-M", "Predictor- Corrector"

ii) Compare the three methods by finding the errors at each step by showing your comparison in a table with headings as follows:

"t", "Exact", "Error in AB-4", "Error in AM-3", "Error in A-4 PC"

Also plot the exact solution along with the approximations obtained by the above three methods using different colors and legends. Which method do you think approximated the solution best?

```
%a2q1(a)
A = [4, 1, 1, 0, 2; -1, -3, 1, 1, 0; 2, 1, 5, -1, -1; -1, -1, -1, 4, 0; 0,
2,-1, 1, 4];
t=issymmetric(A);
d=eig(A)>0;
if t && d
 disp('the matrix A is positive definite')
else
 disp('the matrix A is not positive definite')
end
b = [6; 6; 6; 6; 6];
% Solve the system of linear equations Ax = b
exact_solution = A \ b;
[x_j,k1] = jacobi(A,b);
[x_s,k2] = sor(A,b);
[x_g,k3] = gauss_seidel(A,b);
total=[k1,k2,k3];
table(exact_solution,x_j,x_g,x_s)
fprintf(' ___iteation:: %d \t%d \t %d',k1,k3,k2)

function [x,k] =gauss_seidel(mat,b)
 x=zeros(5,1);
 error=inf;
 tol=0.00001;
 k=0;
 xold=x;
 while error>tol
 for i=1:size(mat,1)
 s=0;
 for j=1:size(mat,1)
 if i~=j
 s=s+mat(i,j)*xold(j);
 end
 end
 x(i)=(b(i)-s)/mat(i,i);
 error=abs(x(i)-xold(i));
 xold(i)=x(i);
 end
 k=k+1;
 end
end

function [c,x] = gaussjordanmethod(mat,b)
c = [mat, b];
for i = 1:size(c, 1)-1
 for j = i+1:size(c, 1)
 c(j, :) = c(j, :) - (c(j, i) .* c(i, :))/c(i,i);
 end
end
x=zeros(size(c, 1),1);
x(5,1)=c(5,6)/c(5,5);
disp('solution')
```

```
for i = 5:-1:1
 s=0;
 for j=5:-1:1
 if i<j
 s=s+c(i,j)*x(j);
 end
 end
 x(i,1)=(c(i,6)-s)/c(i,i);
end
end

function [x,k] = jacobi(mat,b)
 x=zeros(5,1);
 error=inf;
 tol=0.00001;
 k=0;
 while error>tol
 xold=x;
 for i=1:size(mat,1)
 s=0;
 for j=1:size(mat,1)
 if i~=j
 s=s+mat(i,j)*xold(j);
 end
 end
 x(i)=(b(i)-s)/mat(i,i);
 error=abs(x-xold);
 end
 k=k+1;
 end
end

function [x,k] = sor(mat,b)
 x=zeros(5,1);
 error=inf;
 tol=0.00001;
 k=0;
 xold=x;
 w=1.1;
 while error>tol
 for i=1:size(mat,1)
 s=0;
 for j=1:size(mat,1)
 if i~=j
 s=s+mat(i,j)*xold(j);
 end
 end
 x(i)=(1-w)*x(i)+w*(b(i)-s)/mat(i,i);
 error=abs(x(i)-xold(i));
 xold(i)=x(i);
 end
 k=k+1;
 end
end
```

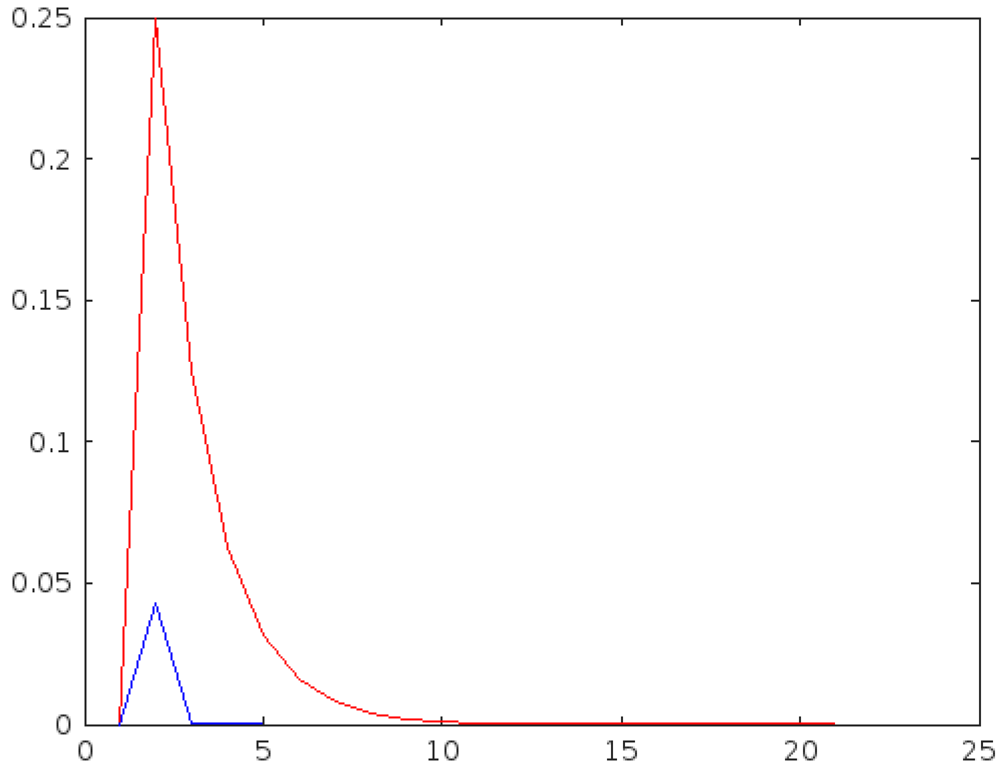
```
the matrix A is not positive definite

ans =

  5x4 table

    exact_solution      x_j         x_g         x_s    
    ______________    ________    ________    _______

        0.22416        0.22407     0.22416    0.22416
       -0.77709       -0.77699    -0.77708    -0.7771
         2.0249          2.025      2.0249     2.0249
          1.868          1.868       1.868      1.868
         1.9278         1.9278      1.9278     1.9278

 ___iteation:: 17     10      11
```

Published with MATLAB® R2024a

```
clc;clear all;

f=@(x) -x^3+x+exp(x)+2;
a=2;b=3;tolerance=.5*.000001
title=(' i   a      b         roots     error');
fprintf('%s\n',title);
for i=1:50
 x(i)=(a+b)/2;
 if f(x(i))*f(a)>0
 a=x(i);
 else
 b=x(i);
 end
 if i>1
 error(i)=abs(x(i)-x(i-1)) ;
 end
 if i==1
 fprintf('%2d %3.6f %.6f %.6f \n ',i,a,b,x(i));
 else
 fprintf('%d %.6f %.6f %.6f %.6f \n ',i,a,b,x(i),error(i));
 if tolerance>error(i)
 break
 end
 end
end
a=2;b=3;
fprintf('secant method\n')
title=(' i   a      b         roots     error');
fprintf('%s\n',title);
for i=1:50
 x(i)=(f(b)*a-f(a)*b)/(f(b)-f(a));
 if f(x(i))*f(a)>0
 a=x(i);
 else
 b=x(i);
 end
 if i>1
 error1(i)=abs(x(i)-x(i-1)) ;
 end
 if i==1
 fprintf('%d %.6f %.6f %.6f \n ',i,a,b,x(i));
 else
 fprintf('%d %.6f %.6f %.6f %.6f \n ',i,a,b,x(i),error1(i));
 if tolerance>error1(i)
 break
 end
 end
end
a=2;b=3;
fprintf('regular falsi\n');
title=(' i   a      b         roots     error');
fprintf('%s\n',title);
```

```
for i=1:50
 x(i)=(f(b)*a-f(a)*b)/(f(b)-f(a));
 if f(x(i))*f(a)>0
 a=x(i);
 else
 b=x(i);
 end
 if i>1
 error2(i)=abs(x(i)-x(i-1)) ;
 end
 if i==1
 fprintf('%d %.6f %.6f %.6f \n ',i,a,b,x(i));
 else
 fprintf('%d %.6f %.6f %.6f %.6f \n ',i,a,b,x(i),error2(i));
 if tolerance>error2(i)
 break
 end
 end
end
x=1:length(error);
plot(x,error,'red')
hold on
x1=1:length(error1);
plot(x1,error1,'y')
hold on
x2=1:length(error2);
plot(x1,error2,'b')
```

```
tolerance =

   5.0000e-07

 i   a       b          roots     error
 1 2.500000 3.000000 2.500000
 2 2.500000 2.750000 2.750000 0.250000
 3 2.625000 2.750000 2.625000 0.125000
 4 2.625000 2.687500 2.687500 0.062500
 5 2.656250 2.687500 2.656250 0.031250
 6 2.671875 2.687500 2.671875 0.015625
 7 2.679688 2.687500 2.679688 0.007812
 8 2.679688 2.683594 2.683594 0.003906
 9 2.681641 2.683594 2.681641 0.001953
 10 2.682617 2.683594 2.682617 0.000977
 11 2.682617 2.683105 2.683105 0.000488
 12 2.682617 2.682861 2.682861 0.000244
 13 2.682617 2.682739 2.682739 0.000122
 14 2.682678 2.682739 2.682678 0.000061
 15 2.682709 2.682739 2.682709 0.000031
 16 2.682724 2.682739 2.682724 0.000015
 17 2.682724 2.682732 2.682732 0.000008
 18 2.682724 2.682728 2.682728 0.000004
 19 2.682726 2.682728 2.682726 0.000002
 20 2.682726 2.682727 2.682727 0.000001
 21 2.682726 2.682727 2.682726 0.000000
 secant method
 i   a       b          roots     error
1 2.639020 3.000000 2.639020
 2 2.682071 3.000000 2.682071 0.043050
 3 2.682719 3.000000 2.682719 0.000648
 4 2.682726 3.000000 2.682726 0.000007
 5 2.682727 3.000000 2.682727 0.000000
 regular falsi
 i   a       b          roots     error
1 2.639020 3.000000 2.639020
 2 2.682071 3.000000 2.682071 0.043050
 3 2.682719 3.000000 2.682719 0.000648
 4 2.682726 3.000000 2.682726 0.000007
 5 2.682727 3.000000 2.682727 0.000000
```

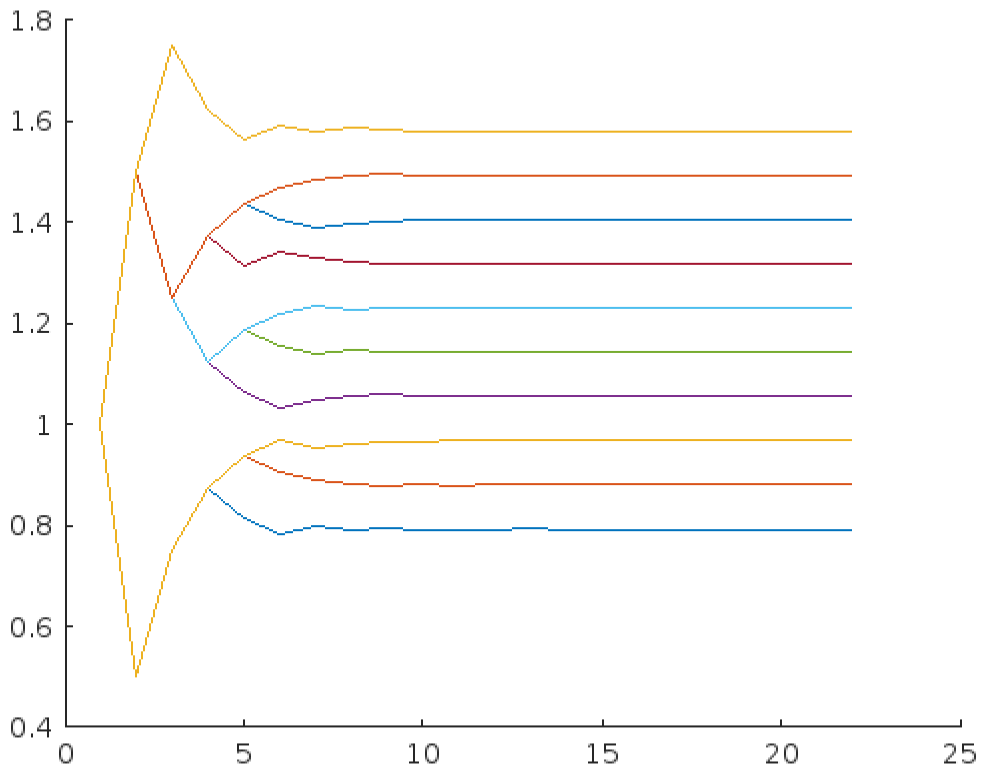


Published with MATLAB® R2024a

```
%a2q3
%assume that t/T=a
%(i)
a1=1/4;
a2=1/8;
x2=2;
disp('newton raphson method t=T/4')
[x1,k,error] = newtonRapson(x2,a1);
table(k,x1,error)
disp('fixPoint method t=T/4')
[x1,k,error] = fixedPoint(x2,a1);
table(k,x1,error)
 % ii
disp('newton raphson method t=T/8')
[x1,k,error] = newtonRapson(x2,a2);
table(k,x1,error)
disp('fixPoint method t=T/8')
[x1,k,error] = fixedPoint(x2,a2);
table(k,x1,error)
a=linspace(1/8,1/4,10);
intial_x=0;
% iii
for i=1:10
 [x,k] = bisection(intial_x,x2,a(i));
 hold on
 plot(k,x)
end

function [x1,k,error] = newtonRapson(x2,a)
epsilon=0.01;
f=@(x) x-epsilon*sin(x)-2*pi*a;
df=@(x) 1-epsilon*cos(x);
error=inf;
tol=0.00001;
i=0;
while error>tol
 i=i+1;
 x1(i)=x2-(f(x2)/df(x2));
 error(i)=abs(x1(i)-x2);
 x2=x1(i);
 k(i)=i;
end
error=error';
x1=x1';
k=k';
end
function [x1,k,error] = fixedPoint(x2,a)
epsilon=0.01;
g=@(x) epsilon*sin(x)+2*pi*a;
error=inf;
tol=0.00001;
i=0;
```

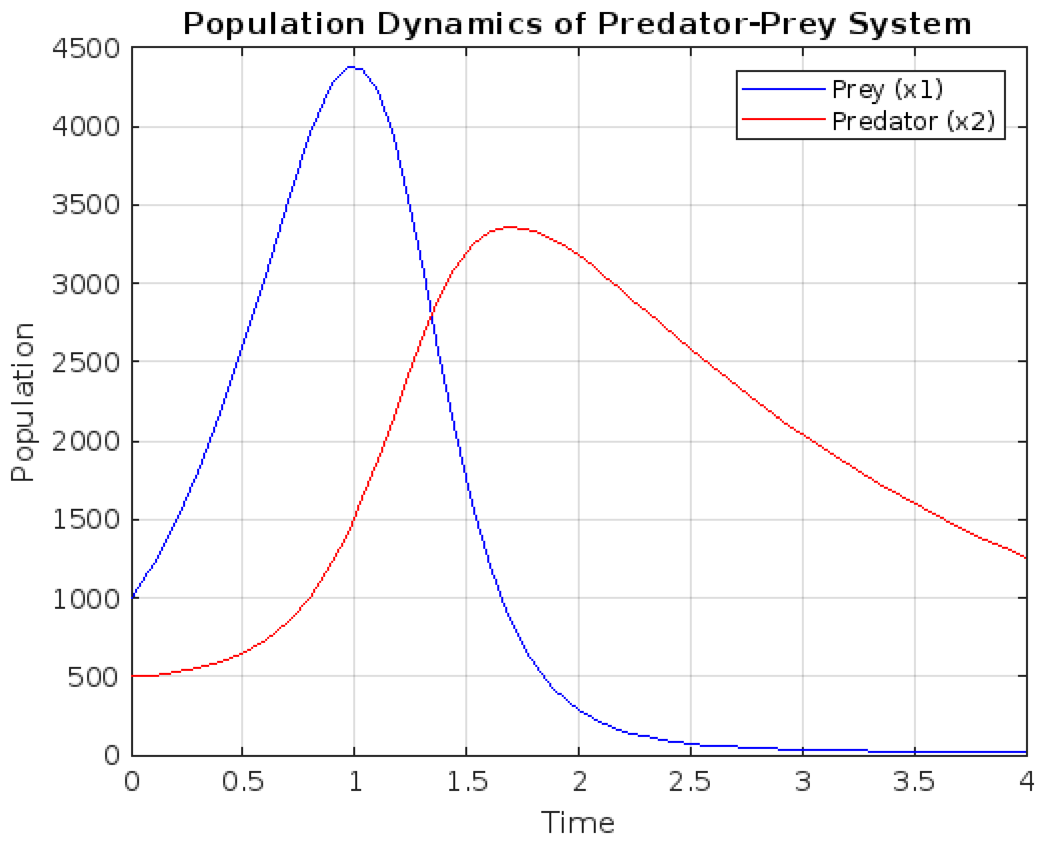
```
while error>tol
 i=i+1;
 x1(i)=g(x2);
 error(i)=abs(x1(i)-x2);
 x2=x1(i);
 k(i)=i;
end
 error=error';
 x1=x1';
 k=k';
end

function [x1,k] = bisection(a,b,a1)
epsilon=0.01;
f=@(x) x-epsilon*sin(x)-2*pi*a1;
tolerance=.5*.000001;
error=inf;
for i=1:50
 x1(i)=(a+b)/2;
 if f(x1(i))*f(a)>0
 a=x1(i);
 else
 b=x1(i);
 end
 if i>1
 error(i)=abs(x1(i)-x1(i-1)) ;
 end
 k(i)=i;
 if tolerance>error(i)
 break
 end
end
end
```

```
newton raphson method t=T/4

ans =

  3x3 table

    k      x1         error   
    _    ______    ___________

    1    1.5816        0.41837
    2    1.5808      0.0008345
    3    1.5808     3.4814e-09

fixPoint method t=T/4

ans =

  3x3 table

    k      x1         error   
    _    ______    __________

    1    1.5799       0.42011
    2    1.5808    0.00090661
    3    1.5808    8.6546e-08

newton raphson method t=T/8

ans =

  3x3 table

    k      x1          error
    _    _______    __________

    1    0.79949        1.2005
    2    0.79252     0.0069675
    3    0.79252    1.7487e-07

fixPoint method t=T/8

ans =

  4x3 table

    k      x1          error
    _    _______    __________

    1    0.79449        1.2055
    2    0.79253     0.0019579
    3    0.79252    1.3732e-05
    4    0.79252    9.6403e-08
```



Published with MATLAB® R2024a

```
clc;clear all;
k1 = 3;
k2 = 0.002;
k3 = 0.0006;
k4 = 0.5;
x1_0 = 1000;
x2_0 = 500;
tspan = [0 4];
ode = @(t, x) [k1*x(1) - k2*x(1)*x(2); k3*x(1)*x(2) - k4*x(2)];
[t, populations] = ode45(ode, tspan, [x1_0, x2_0]);
plot(t, populations(:, 1), 'b', t, populations(:, 2), 'r');
xlabel('Time');
ylabel('Population');
legend('Prey (x1)', 'Predator (x2)');
title('Population Dynamics of Predator-Prey System');
grid on
```

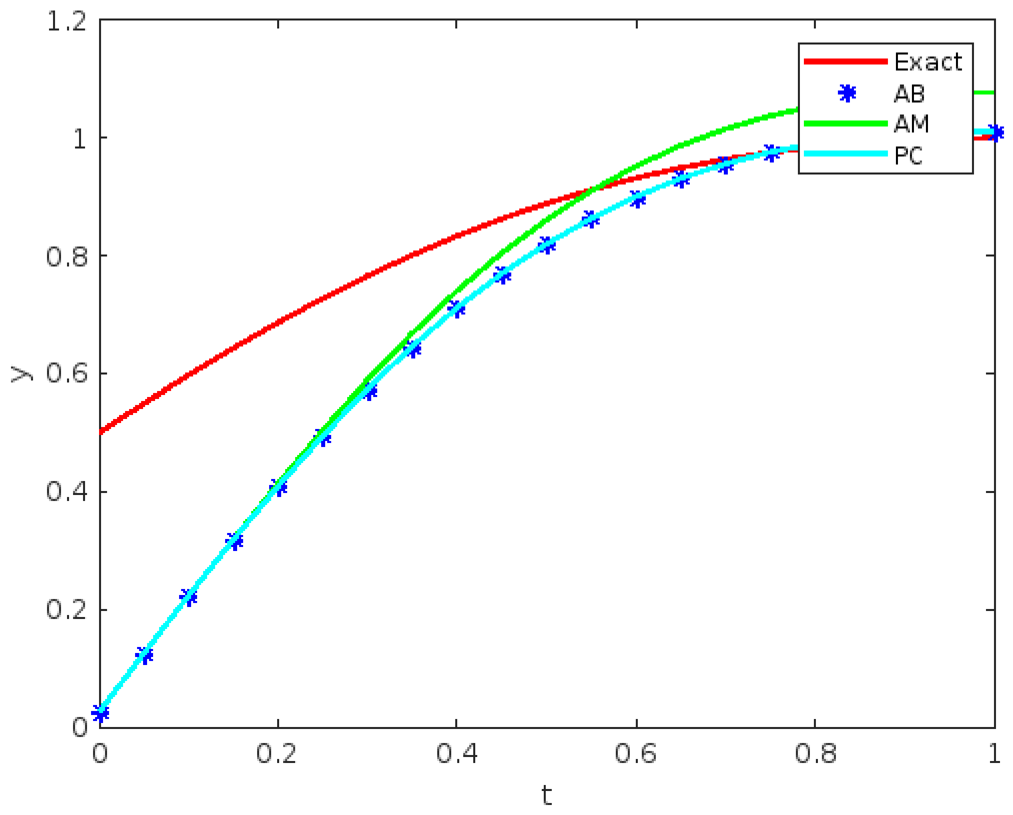


Published with MATLAB® R2024a

```
f=@(t,y)((2-2*t*y)/(t^2+1));
ext=@(t)((2*t+1)/(t^2+2));
h=0.05;u=1;l=0;t(1)=0;w(1)=0.025;
num_iter=(u-l)/h;

for i=2:num_iter+1
    if i<5
        k1=h*f(t(i-1),w(i-1));
        k2=h*f(t(i-1)+h/2,w(i-1)+k1/2);
        k3=h*f(t(i-1)+h/2,w(i-1)+k2/2);
        k4=h*f(t(i-1)+h,w(i-1)+k3);
        w(i)=w(i-1)+1/6*(k1+2*k2+2*k3+k4);
    else
                w(i)=w(i-1)+(h/24)*(55*f(t(i-1),w(i-1))-59*f(t(i-2),w(i-2))
+37*f(t(i-3),w(i-3))-9*f(t(i-4),w(i-4)));
    end
    t(i)=t(i-1)+h;
end
w1=w.';

for i=2:num_iter+1
    if i<5
        k1=h*f(t(i-1),w(i-1));
        k2=h*f(t(i-1)+h/2,w(i-1)+k1/2);
        k3=h*f(t(i-1)+h/2,w(i-1)+k2/2);
        k4=h*f(t(i-1)+h,w(i-1)+k3);
        w(i)=w(i-1)+1/6*(k1+2*k2+2*k3+k4);
    else
        w(i)=w(i-1)+(h/24)*(9*f(t(i-1),w(i-1))
+19*f(t(i-2),w(i-2))-5*f(t(i-3),w(i-3))+f(t(i-4),w(i-4)));
    end
    t(i)=t(i-1)+h;
end
w2=w.';

for i=2:num_iter+1
    if i<5
        k1=h*f(t(i-1),w(i-1));
        k2=h*f(t(i-1)+h/2,w(i-1)+k1/2);
        k3=h*f(t(i-1)+h/2,w(i-1)+k2/2);
        k4=h*f(t(i-1)+h,w(i-1)+k3);
        w(i)=w(i-1)+1/6*(k1+2*k2+2*k3+k4);
    else
        w(i)=w(i-1)+(h/24)*(55*f(t(i-1),w(i-1))-59*f(t(i-2),w(i-2))
+37*f(t(i-3),w(i-3))-9*f(t(i-4),w(i-4)));
        w(i)=w(i-1)+(h/24)*(9*f(t(i),w(i))
+19*f(t(i-1),w(i-1))-5*f(t(i-2),w(i-2))+f(t(i-3),w(i-3)));

    end
    t(i)=t(i-1)+h;
end
w3=w.';
```

```
tvalue=[0:0.05:1];
for i=1:num_iter+1
    exactvalue(i)=ext(tvalue(i));
end
Exact=exactvalue.';
t=tvalue.';
AB=w1;
AM=w2;
PC=w3;
table(t,Exact,AB,AM,PC)

e_AB=abs(AB-Exact);
e_AM=abs(AM-Exact);
e_PC=abs(PC-Exact);
table(t,e_AB,e_AM,e_PC)

plot(tvalue,exactvalue,'r','DisplayName','Exact','LineWidth',2);
hold on
plot(tvalue,w1,'b*','DisplayName','Adam Bashforth 4 step','LineWidth',2);
hold on
plot(tvalue,w2,'g','DisplayName','Adam-Moulton 3 step','LineWidth',2);
hold on
plot(tvalue,w1,'c','DisplayName','4th order predictor-
corrector','LineWidth',2);
set(gca,'Xaxislocation','origin','YAxisLocation','origin');
xlabel('t');
ylabel('y');
legend('Exact','AB','AM','PC')
```

```
ans =

  21x5 table

     t        Exact        AB          AM          PC
    _____    _______    _______    _______    _______

       0        0.5      0.025      0.025      0.025
    0.05    0.54931    0.12469    0.12469    0.12469
     0.1    0.59701    0.22277    0.22277    0.22277
    0.15    0.64277    0.31785    0.31785    0.31785
     0.2    0.68627    0.40863    0.41293    0.40866
    0.25    0.72727    0.49408     0.5037    0.49412
     0.3    0.76555    0.57335    0.58902     0.5734
    0.35    0.80094    0.64584    0.66797    0.64588
     0.4    0.83333    0.71117     0.7399    0.71121
    0.45    0.86266     0.7692    0.80439    0.76923
     0.5    0.88889    0.81998    0.86126       0.82
    0.55    0.91205    0.86372    0.91055    0.86372
     0.6     0.9322    0.90074    0.95246    0.90073
    0.65    0.94943    0.93148    0.98734    0.93146
     0.7    0.96386     0.9564     1.0156    0.95637
    0.75    0.97561    0.97603     1.0379      0.976
     0.8    0.98485    0.99089     1.0546    0.99085
    0.85    0.99174     1.0015     1.0664     1.0014
     0.9    0.99644     1.0083     1.0739     1.0083
    0.95    0.99914     1.0119     1.0775     1.0118
       1          1     1.0125     1.0777     1.0125


ans =

  21x4 table

     t          e_AB           e_AM           e_PC    
    ____    ____________    __________    ____________

       0           0.475         0.475           0.475
    0.05         0.42463       0.42463         0.42463
     0.1         0.37424       0.37424         0.37424
    0.15         0.32492       0.32492         0.32492
     0.2         0.27764       0.27335         0.27762
    0.25         0.23319       0.22357         0.23315
     0.3          0.1922       0.17653         0.19215
    0.35         0.15511       0.13297         0.15506
     0.4         0.12217      0.093432         0.12212
    0.45        0.093456      0.058264        0.093424
     0.5        0.068908      0.027625        0.068889
    0.55        0.048334     0.0015032         0.04833
     0.6        0.031462      0.020253        0.031471
    0.65        0.017956      0.037903        0.017977
     0.7       0.0074526      0.051782       0.0074837
    0.75      0.00042447      0.062271      0.00038565
     0.8       0.0060445      0.069775       0.0060003
    0.85       0.0097584      0.074697       0.0097108
     0.9         0.01189      0.077426        0.011841
    0.95        0.012732      0.078328        0.012683
       1        0.012544      0.077733        0.012495
```



Published with MATLAB® R2024a