Table of Contents

G12ISC 2018-2019 Coursework 5

Student ID: 4336432 Subject: Numerical ODE's

```
clear all
close all
clc
```

Question 1

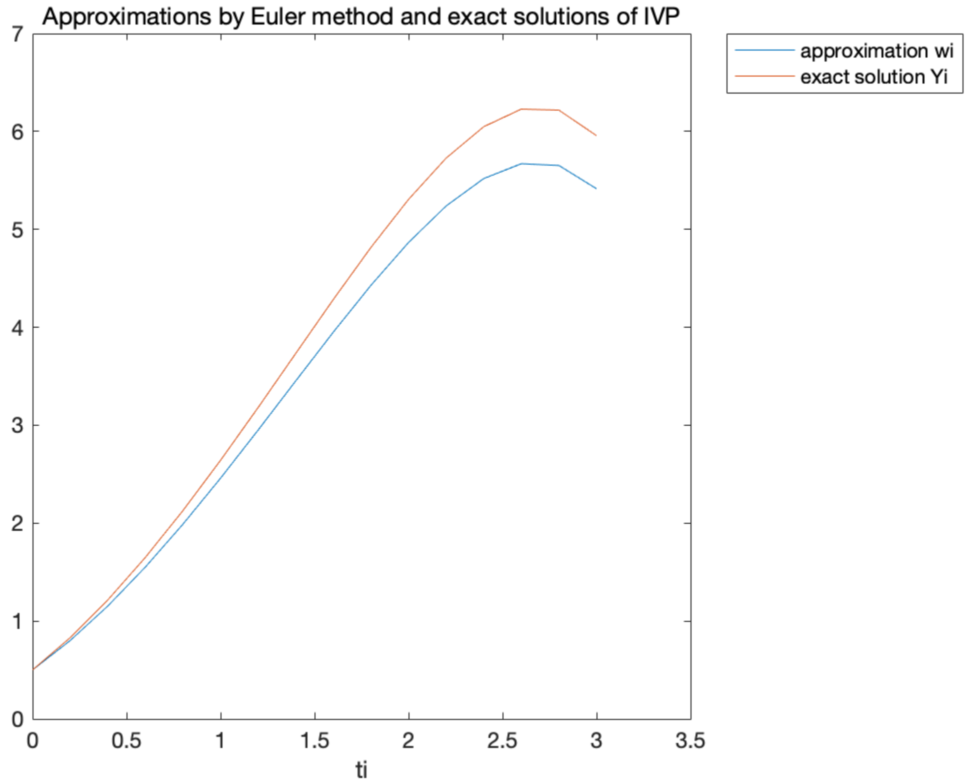
```
% Following code produces a table and a figure comparing the
 approximations
% by Euler's method with exact solutions.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
h = 0.2;
w0 = 0.5;
N = (b-a)/h;

% Create table
format long g
[t,w]=Euler(a,b,N,w0,f);
Y = zeros(N+1,1);
error = zeros(N+1,1);
for i = 1:N+1
    Y(i) = y(t(i));
    error(i) = abs(w(i)-Y(i));
end
table(t,w,Y,error)

% Create figure
plot(t,w)
```

```
hold on
plot(t,Y)
% Format figure
title('Approximations by Euler method and exact solutions of IVP')
legend('approximation wi','exact solution
 Yi','location','bestoutside')
xlabel('ti')
```

ans =

16×4 table

t	w	Y	error
0	0.5	0.5	0
0.2	0.8	0.829298620919915	0.029298620919915
0.4	1.152	1.21408765117936	0.0620876511793644
0.6	1.5504	1.64894059980475	0.0985405998047457
0.8	1.98848	2.12722953575377	0.138749535753766
1	2.458176	2.64085908577048	0.182683085770477
1.2	2.9498112	3.17994153863173	0.230130338631727
1.4	3.45177344	3.73240001657766	0.280626576577662
1.6	3.950128128	4.28348378780244	0.333355659802439
1.8	4.4281537536	4.81517626779353	0.387022514193526
2	4.86578450432	5.30547195053468	0.439687446214674
2.2	5.238941405184	5.72749325028294	0.488551845098938
2.4	5.5187296862208	6.0484118096792	0.529682123458396
2.6	5.67047562346496	6.22813098249916	0.557655359034192
2.8	5.65257074815796	6.21767661445147	0.565105866293519
3	5.41508489778955	5.95723153840616	0.542146640616616

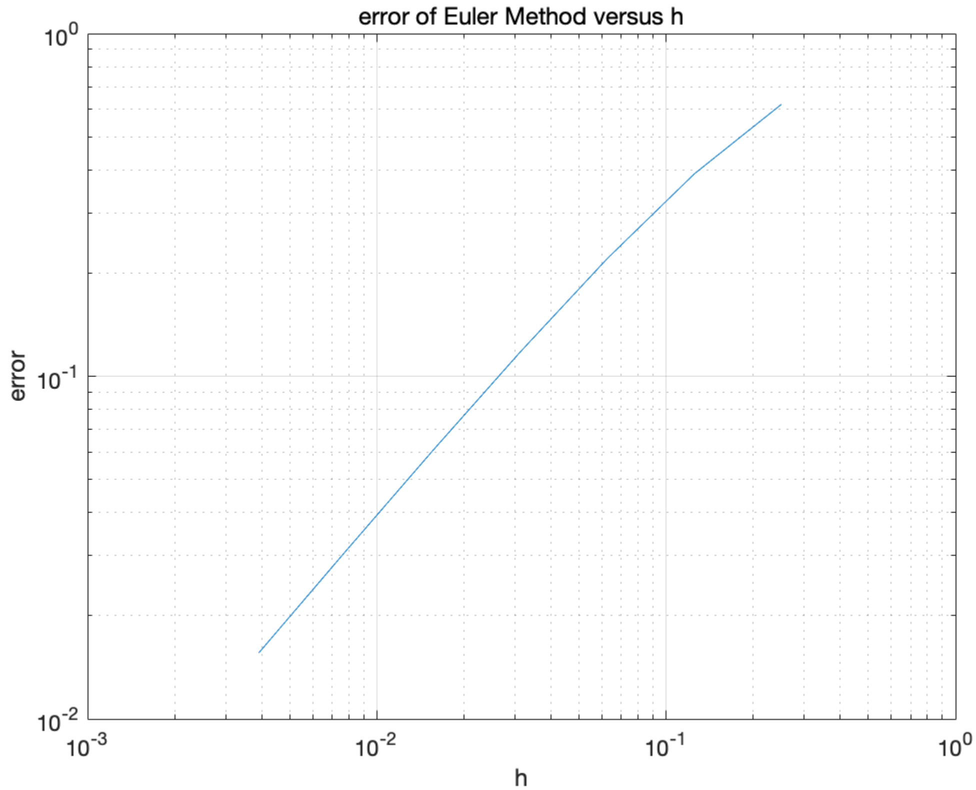


Question 2

```
clear all
close all
clc

% Following code produces a table and a figure to invesgate the
 asymptotic
% rate of convergence of Euler's method.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
w0 = 0.5;

h = zeros(7,1); % initialise
e = zeros(7,1);

% Create table
format long g
for i = 2:8
    h(i-1) = 1/(2^i);
```

```
    N = (b-a)/h(i-1);
    [t,w] = Euler(a,b,N,w0,f);
    e(i-1) = abs(w(N+1)-y(3));
end
table(h,e)

% Create figure
loglog(h,e)
% Format figure
title('error of Euler Method versus h')
xlabel('h')
ylabel('error')
grid on

% Therefore, from the plot, errors of Euler's Method converge
 linearly.
```

```
ans =

  7×2 table

        h                 e
    __________    _________________

          0.25     0.621167959681305
         0.125     0.389232377686741
        0.0625     0.220243293587793
       0.03125     0.117508380031466
      0.015625    0.0607426223098138
     0.0078125    0.0308875300289664
    0.00390625    0.0155753024066279
```



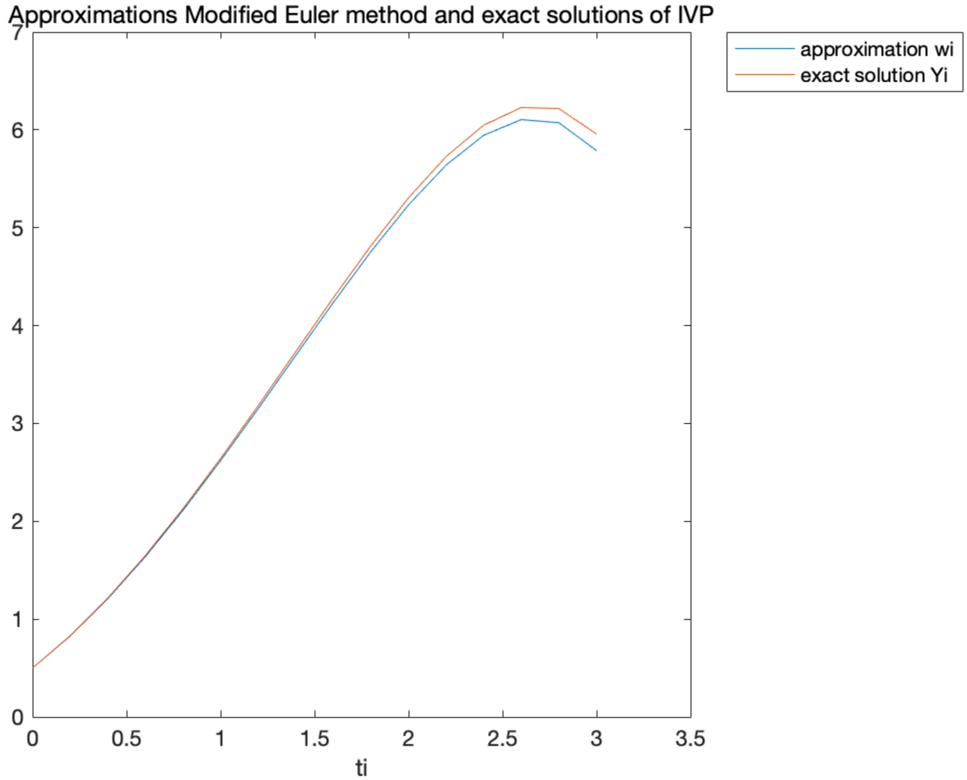
Question 4

```
clear all
close all
clc

% Following code produces a table and a figure comparing the
 approximations
% by Modified Euler's method with exact solutions.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
h = 0.2;
w0 = 0.5;
N = (b-a)/h;

% Create table
[t,w]=ModifiedEuler(a,b,N,w0,f);
Y = zeros(N+1,1);
error = zeros(N+1,1);
for i = 1:N+1
```

```
    Y(i) = y(t(i));
    error(i) = abs(w(i)-Y(i));
end
table(t,w,Y,error)

% Create figure
format long g
plot(t,w)
hold on
plot(t,Y)
% Format figure
title('Approximations Modified Euler method and exact solutions of
 IVP')
legend('approximation wi','exact solution
 Yi','location','bestoutside')
xlabel('ti')
```

```
ans =

  16×4 table

     t               w                      Y                      error
    ___     ________________     ________________     ____________________

      0                  0.5                  0.5                        0
    0.2                0.826    0.829298620919915      0.00329862091991495
    0.4              1.20692     1.21408765117936      0.00716765117936435
    0.6            1.6372424     1.64894059980475       0.0116981998047456
    0.8          2.110235728     2.12722953575377       0.0169938077537659
      1        2.61768758816     2.64085908577048       0.0231714976104764
    1.2      3.1495788575552     3.17994153863173       0.0303626810765261
    1.4     3.69368620621735     3.73240001657766       0.0387138103603171
    1.6     4.23509717158516     4.28348378780244       0.0483866162172788
    1.8      4.7556185493339     4.81517626779353       0.0595577184596294
      2     5.23305463018736     5.30547195053468        0.07241732034732
    2.2     5.64032664882857     5.72749325028294       0.0871666014543653
    2.4     5.94439851157086      6.0484118096792        0.104013298108338
```

```
   2.6     6.10496618411645      6.22813098249916
0.123164798382706
   2.8     6.07285874462207      6.21767661445147
0.144817869829406
     3     5.78808766843892      5.95723153840616
0.16914386996724
```



Question 5

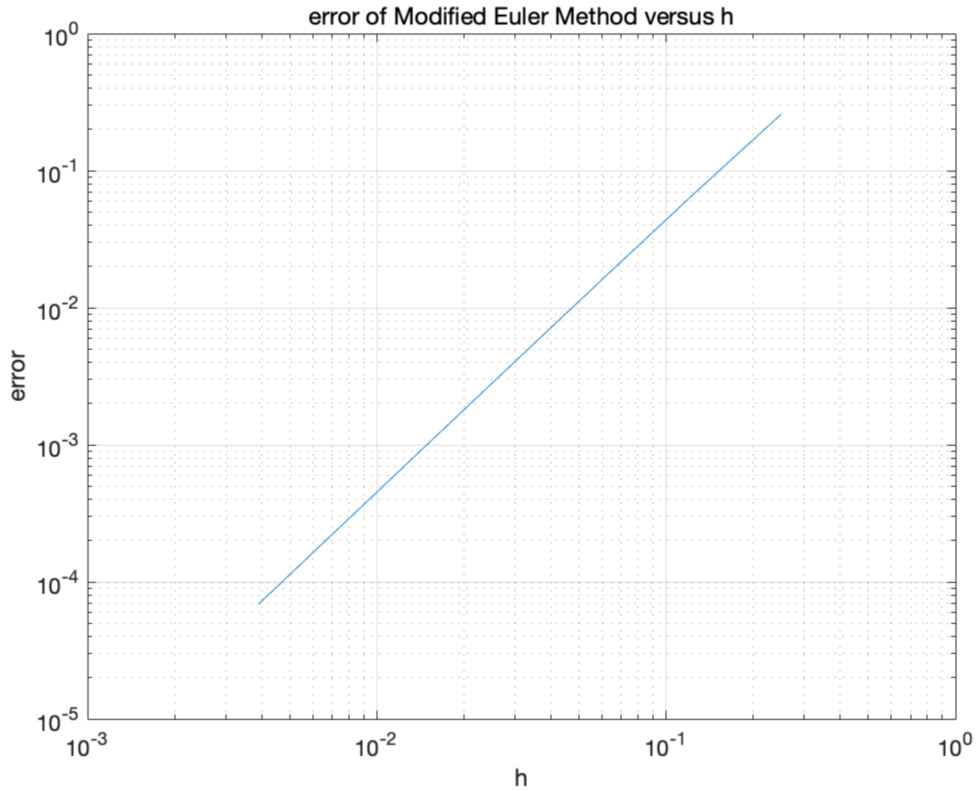
```
clear all
close all
clc

% Following code produces a table and a figure to invesgate the
 asymptotic
% rate of convergence of Modified Euler's method.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
w0 = 0.5;
```

```
h = zeros(7,1); % initialise
e = zeros(7,1);

% Create table
format long g
for i = 2:8
    h(i-1) = 1/(2^i);
    N = (b-a)/h(i-1);
    [t,w] = ModifiedEuler(a,b,N,w0,f);
    e(i-1) = abs(w(N+1)-y(3));
end
table(h,e)

% Create figure
loglog(h,e)
% Format figure
title('error of Modified Euler Method versus h')
xlabel('h')
ylabel('error')
grid on

% Therefore, from the plot, errors of Modified Euler's Method converge
% quadratically.
```

ans =

7×2 table

h	*e*
0.25	*0.258438623911498*
0.125	*0.0680717214848423*
0.0625	*0.0173751519773679*
0.03125	*0.00438207495903953*
0.015625	*0.00109984938291596*
0.0078125	*0.00027547334552569*
0.00390625	*6.89302556988736e-05*



Question 7

```
clear all
close all
clc

% Following code produces a table and a figure comparing the
 approximations
% by Runge-Kutta Order Four method with exact solutions.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
h = 0.2;
w0 = 0.5;
N = (b-a)/h;

% Create table
format long g
[t,w]=RuKuMeth(a,b,N,w0,f);
Y = zeros(N+1,1);
error = zeros(N+1,1);
```

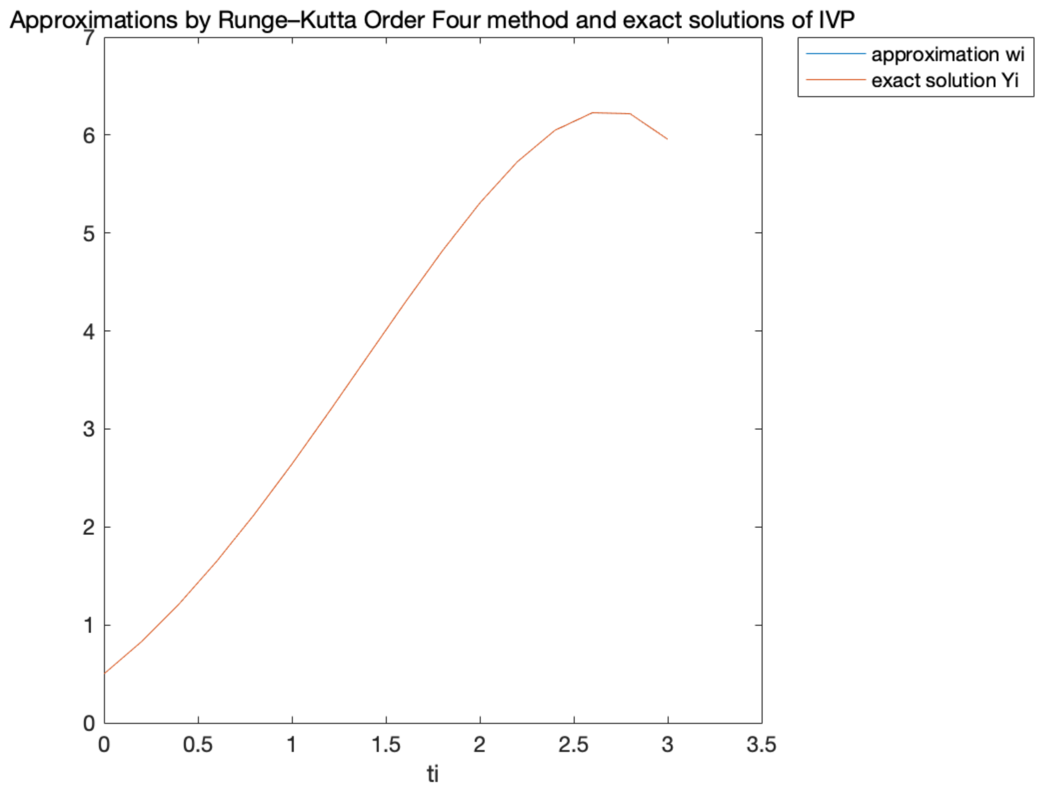
```
for i = 1:N+1
    Y(i) = y(t(i));
    error(i) = abs(w(i)-Y(i));
end
table(t,w,Y,error)

% Create figure
plot(t,w)
hold on
plot(t,Y)
% Format figure
title('Approximations by Runge–Kutta Order Four method and exact
 solutions of IVP')
legend('approximation wi','exact solution
 Yi','location','bestoutside')
xlabel('ti')
```

ans =

16×4 table

t	w	Y	error
0	0.5	0.5	0
0.2	0.829293333333333	0.829298620919915	5.28758658158157e-06
0.4	1.21407621066667	1.21408765117936	1.14405126978578e-05
0.6	1.6489220170416	1.64894059980475	1.85827631458135e-05
0.8	2.12720268494794	2.12722953575377	2.68508058227646e-05
1	2.64082269272875	2.64085908577048	3.63930417255354e-05
1.2	3.17989417023223	3.17994153863173	4.73683994965945e-05
1.4	3.73234007285498	3.73240001657766	5.99437226829203e-05
1.6	4.28340949831841	4.28348378780244	7.42894840346509e-05
1.8	4.81508569457943	4.81517626779353	9.05732140923377e-05
2	5.30536300069265	5.30547195053468	0.000108949842021033
2.2	5.72736370237934	5.72749325028294	0.000129547903597427
2.4	6.04825935941946	6.0484118096792	0.000152450259736 98

```
   2.6      6.22795331492826      6.22813098249916
0.000177667570891771
   2.8      6.21747151218671      6.21767661445147
0.00020510226475956
     3      5.95699703831819      5.95723153840616
0.000234500087977096
```

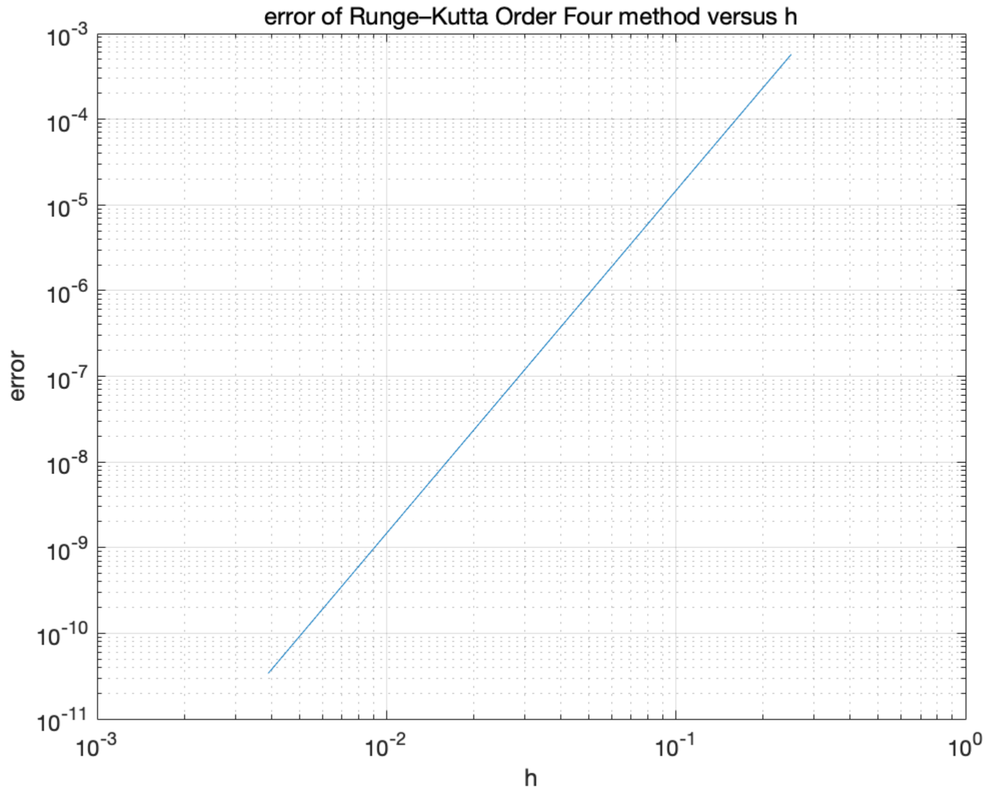


Question 8

```
clear all
close all
clc

% Following code produces a table and a figure to invesgate the
 asymptotic
% rate of convergence of Runge–Kutta Order Four method.

f = @(t,y) y-t^2+1; % y'
y = @(t) (t+1)^2-0.5*exp(t); % exact solution

% Data
a = 0;
b = 3;
w0 = 0.5;
```

```
h = zeros(7,1); % initialise
e = zeros(7,1);

% Create table
format long g
for i = 2:8
    h(i-1) = 1/(2^i);
    N = (b-a)/h(i-1);
    [t,w] = RuKuMeth(a,b,N,w0,f);
    e(i-1) = abs(w(N+1)-y(3));
end
table(h,e)

% Create figure
loglog(h,e)
% Format figure
title('error of Runge-Kutta Order Four method versus h')
xlabel('h')
ylabel('error')
grid on

% Therefore, from the plot, the order of convergence of this method is
% four.
```

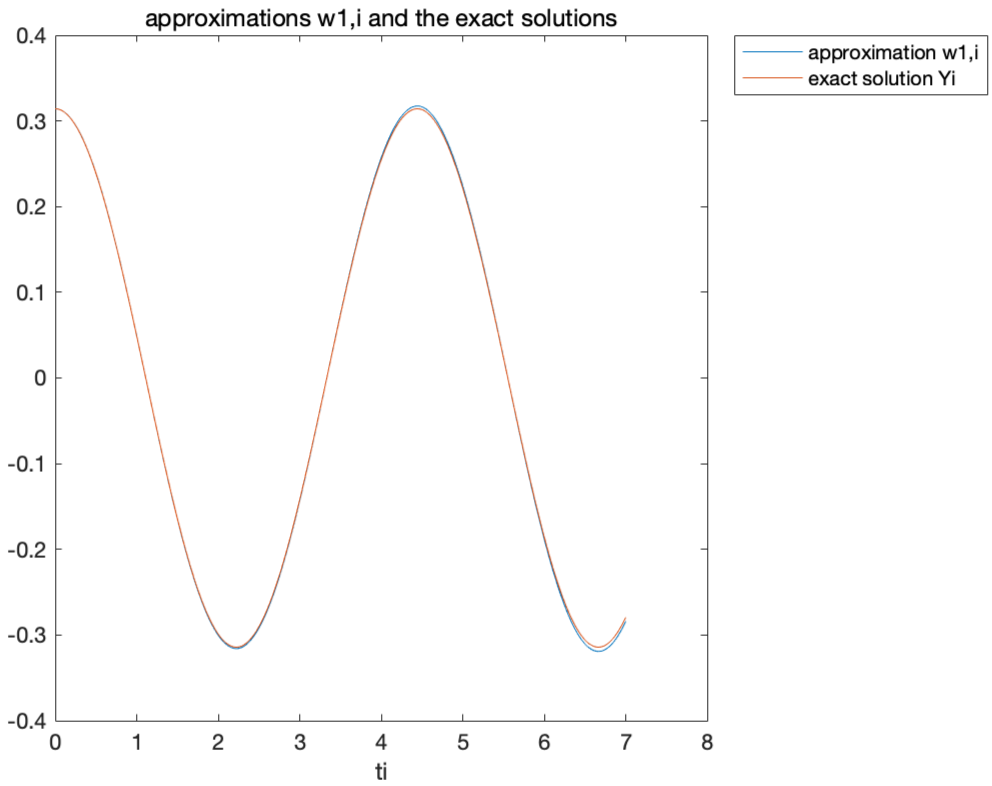
```
ans =

  7×2 table

        h                    e
    __________    ____________________

          0.25    0.000570275089891936
         0.125    3.58919966352289e-05
        0.0625    2.24277622251634e-06
       0.03125    1.40015814942274e-07
      0.015625    8.74371597348045e-09
     0.0078125    5.46208411833504e-10
    0.00390625    3.41220385280394e-11
```



Question 10

```
clear all
close all
clc

% Following code produces a plot of Euler's approximations w1,i and
 the
% exact solutions and figure out for what value of N will the computed
% approximation become completely unreliable.

T = 7;
N = 3000;
theta = 2^(1/2);
alpha = pi/10;
delta = 0;

[t,w1,w2]=EulerSys(T,N,theta,alpha,delta);
y = @(t) alpha*cos(theta*t); % exact solution
Y = zeros(N+1,1); % initialise

% Create figure
for i=1:N+1
    Y(i) = y(t(i));
end
```

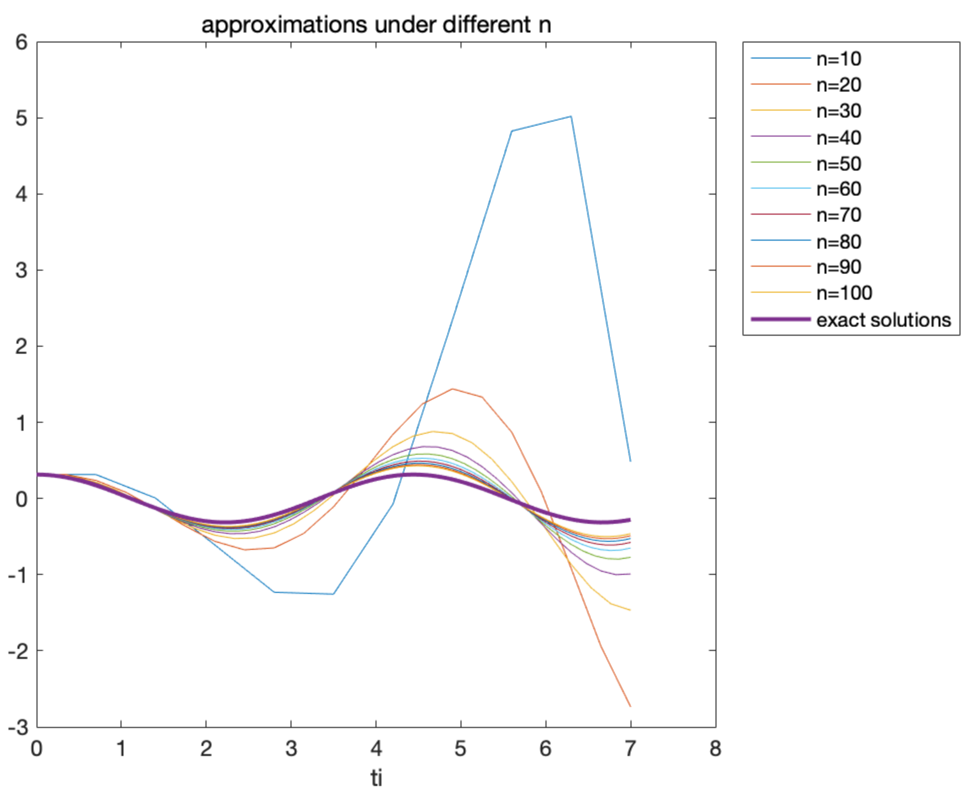
```
plot(t,w1)
hold on
plot(t,Y)
% Format figure
title('approximations w1,i and the exact solutions')
xlabel('ti')
legend('approximation w1,i','exact solution
 Yi','location','bestoutside')
```



Question 10 cont.

```
clear all
close all
clc

% That computed approximation become completely unreliable
 mathematically
% means the approximation go far away from the exact solution. Here 10
% values of n are tested by ploting approximation under each n.

T = 7;
theta = 2^(1/2);
alpha = pi/10;
delta = 0;

% Create figure
```

```
n = linspace(10,100,10);
for i = 1:10
    [ti,w1,w2]=EulerSys(T,n(i),theta,alpha,delta);
    plot(ti,w1)
    hold on
end
t = linspace(0,T,100);
y = alpha*cos(theta*t); % exact solution
plot(t,y,'LineWidth',2)
hold off
% Format figure
title('approximations under different n')
xlabel('ti')
legend('n=10','n=20','n=30','n=40','n=50','n=60','n=70','n=80','n=90','n=100','exa
 solutions','location','bestoutside')

% From the plot, we can see as n becomes smaller, the approximations
 are
% less accurate. When n is less than 20, the tendency of the
 approximation
% goes against the exact solution which is a periodic function.
 Therefore,
% roughly speaking, when n is less than 20, the computed approximation
 will
% become completely unreliable.
```



Published with MATLAB® R2017a