Table of Contents


```
% Joshua Oates - Hw 6 matlab portion
```

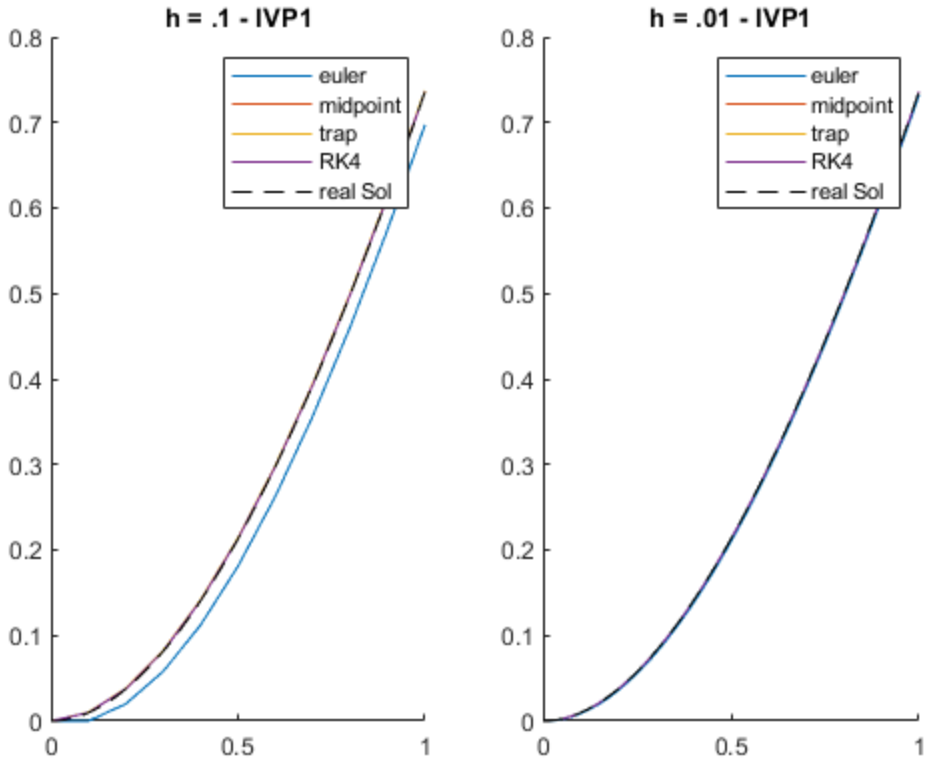
section 0 - clean up

```
clear all;
close all;
clc
```

section 1 - Problem 1 - IVP 1

```
y0 =  0; % set up problem
t0 = 0;
T = 1;
n1 = 1/.1;
n2 = 1/.01;
f = @ode1;

myFig1 = figure; % set up figure for ivp1

subplot(1,2,1)
hold on
title("h = .1 - IVP1")
subplot(1,2,2)
hold on
title("h = .01 - IVP1")


% use euler
clear t1 t2 w1 w2
[t1,w1] = euler(f,t0,y0,T,n1);
[t2,w2] = euler(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use midpoint
clear t1 t2 w1 w2
[t1,w1] = midpoint(f,t0,y0,T,n1);
[t2,w2] = midpoint(f,t0,y0,T,n2);
```

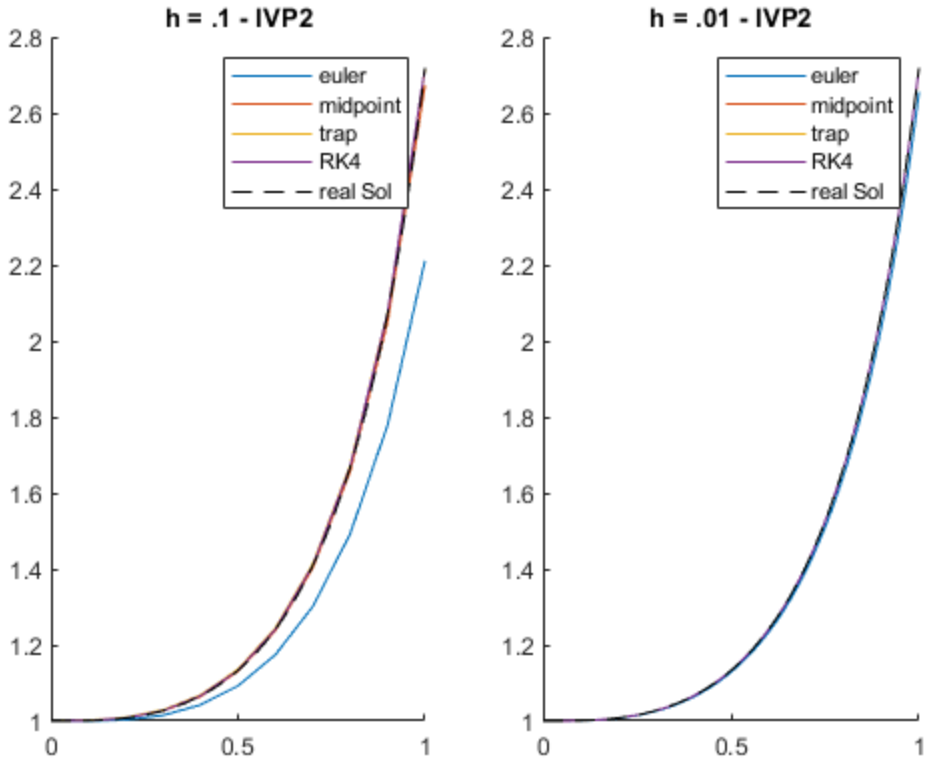
```
subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use trap
clear t1 t2 w1 w2
[t1,w1] = explicitTrapezoid(f,t0,y0,T,n1);
[t2,w2] = explicitTrapezoid(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use RK4
clear t1 t2 w1 w2
[t1,w1] = RK4(f,t0,y0,T,n1);
[t2,w2] = RK4(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% plot real solution
y = @(t) t.*2-2+((exp(t).^-1)*2);
x = linspace(t0,T);
subplot(1,2,1)
plot(x,y(x),"-- k")
subplot(1,2,2)
plot(x,y(x),"-- k")
clear x y

% finish figure 1
subplot(1,2,1)
legend("euler","midpoint","trap","RK4","real Sol")
subplot(1,2,2)
legend("euler","midpoint","trap","RK4","real Sol")
```



section 2 - Problem 1 - IVP 2

```
y0 =  1;% set up problem
t0 = 0;
T = 1;
n1 = 1/.1;
n2 = 1/.01;
f = @ode2;

myFig2 = figure; % set up figure for ivp2

subplot(1,2,1)
hold on
title("h = .1 - IVP2")
subplot(1,2,2)
hold on
title("h = .01 - IVP2")


% use euler
clear t1 t2 w1 w2
[t1,w1] = euler(f,t0,y0,T,n1);
[t2,w2] = euler(f,t0,y0,T,n2);

subplot(1,2,1)
```

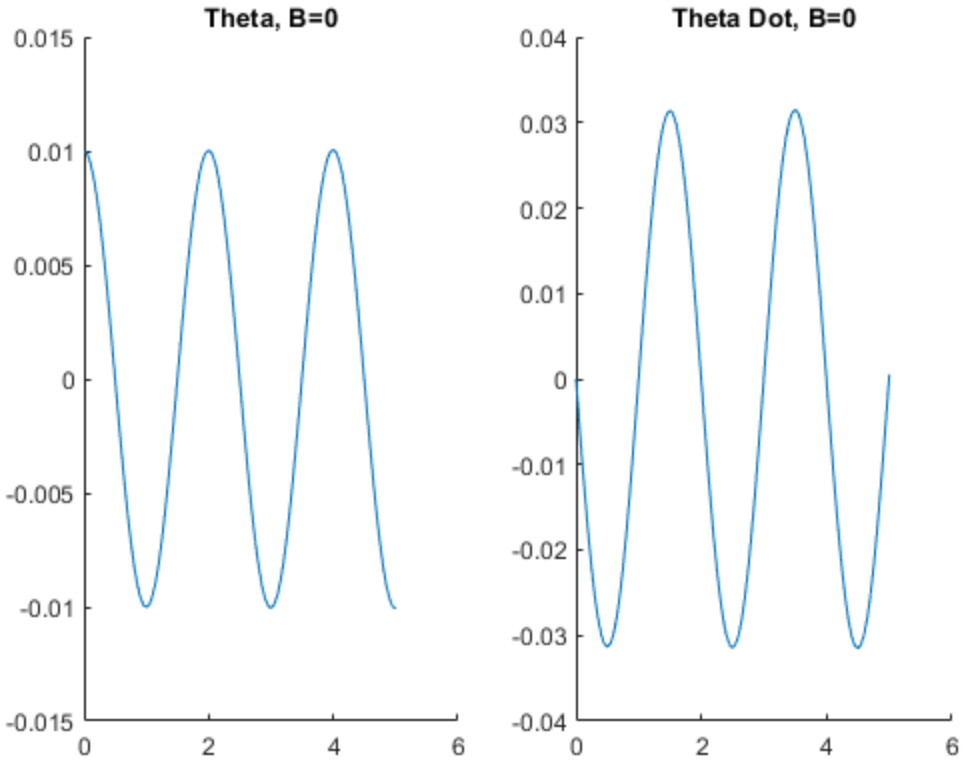
```
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use midpoint
clear t1 t2 w1 w2
[t1,w1] = midpoint(f,t0,y0,T,n1);
[t2,w2] = midpoint(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use trap
clear t1 t2 w1 w2
[t1,w1] = explicitTrapezoid(f,t0,y0,T,n1);
[t2,w2] = explicitTrapezoid(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% use RK4
clear t1 t2 w1 w2
[t1,w1] = RK4(f,t0,y0,T,n1);
[t2,w2] = RK4(f,t0,y0,T,n2);

subplot(1,2,1)
plot(t1,w1)
subplot(1,2,2)
plot(t2,w2)

% plot real solution
y = @(t) exp(t.^3);
x = linspace(t0,T);
subplot(1,2,1)
plot(x,y(x),"-- k")
subplot(1,2,2)
plot(x,y(x),"-- k")
clear x y

% finish figure 2
subplot(1,2,1)
legend("euler","midpoint","trap","RK4","real Sol")
subplot(1,2,2)
legend("euler","midpoint","trap","RK4","real Sol")
```



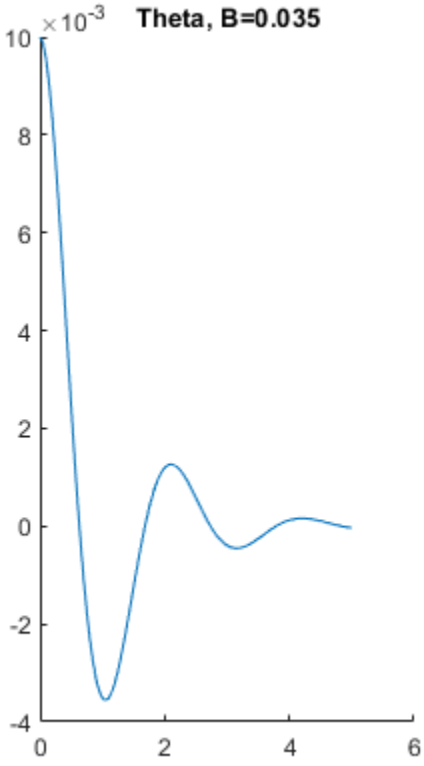
section 3 - pendulum problem 1

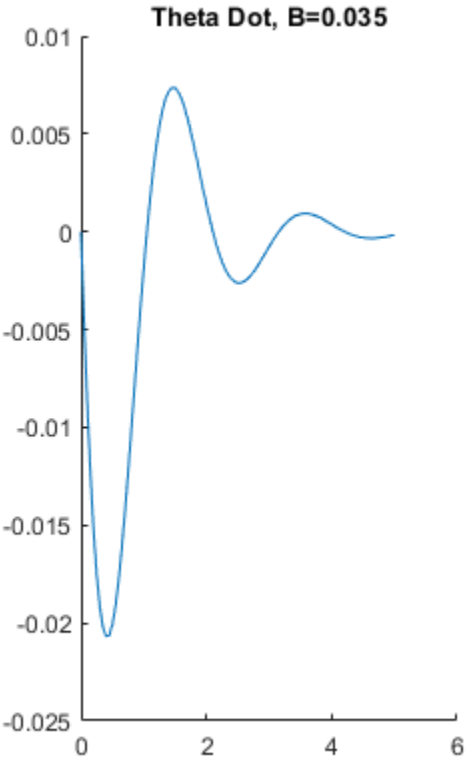
```
clear all

Y0 =  [.01;0];% set up problem
t0 = 0;
T = 5;
n = 100;
m = 2;
f1 = @pendulum1;
f2 = @pendulum2;
f3 = @pendulum3;
f4 = @pendulum4;



clear t1 t2 w1 w2
[t1,w1] = explicitTrapezoidVec(m,f1,t0,Y0,T,n);
[t2,w2] = explicitTrapezoidVec(m,f2,t0,Y0,T,n);
[t3,w3] = explicitTrapezoidVec(m,f3,t0,Y0,T,n);
[t4,w4] = explicitTrapezoidVec(m,f4,t0,Y0,T,n);

myFig3 = figure; % set up figure for B = 0
subplot(1,2,1)
hold on
```

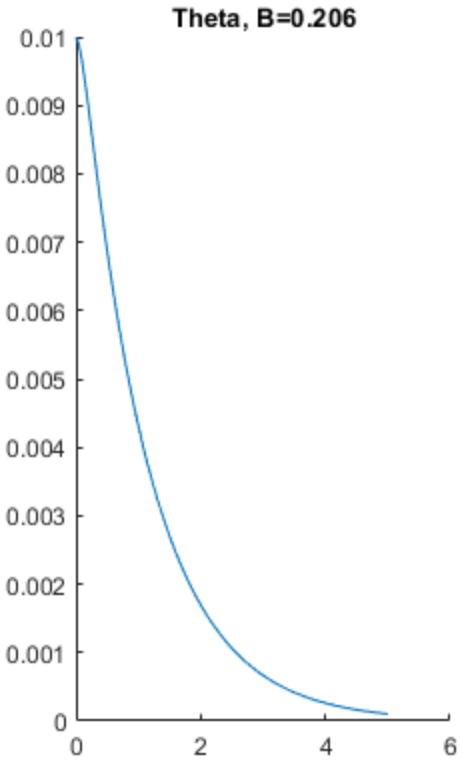
```
title("Theta, B=0")
subplot(1,2,2)
hold on
title("Theta Dot, B=0")

subplot(1,2,1)
plot(t1,w1(:,1))
subplot(1,2,2)
plot(t1,w1(:,2))

myFig4 = figure; % set up figure for B = 0.035
subplot(1,2,1)
hold on
title("Theta, B=0.035")
subplot(1,2,2)
hold on
title("Theta Dot, B=0.035")

subplot(1,2,1)
plot(t2,w2(:,1))
subplot(1,2,2)
plot(t2,w2(:,2))

myFig5 = figure; % set up figure for B = 0.206
subplot(1,2,1)
hold on
title("Theta, B=0.206")
subplot(1,2,2)
hold on
title("Theta Dot, B=0.206")

subplot(1,2,1)
plot(t3,w3(:,1))
subplot(1,2,2)
plot(t3,w3(:,2))

myFig6 = figure; % set up figure for B = 0.25
subplot(1,2,1)
hold on
title("Theta, B=0.25")
subplot(1,2,2)
hold on
title("Theta Dot, B=0.25")

subplot(1,2,1)
plot(t4,w4(:,1))
subplot(1,2,2)
plot(t4,w4(:,2))
```

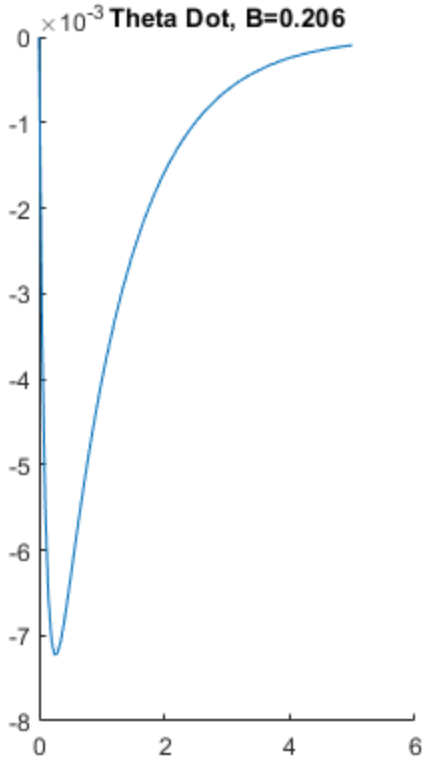

Theta, B=0
0.015
0.01
0.005
0
-0.005
-0.01
-0.015
0
2
4
6

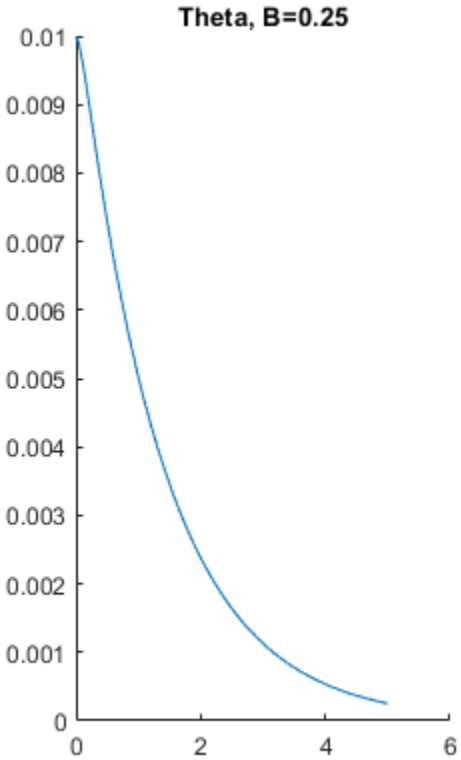
Theta Dot, B=0
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
0
2
4
6

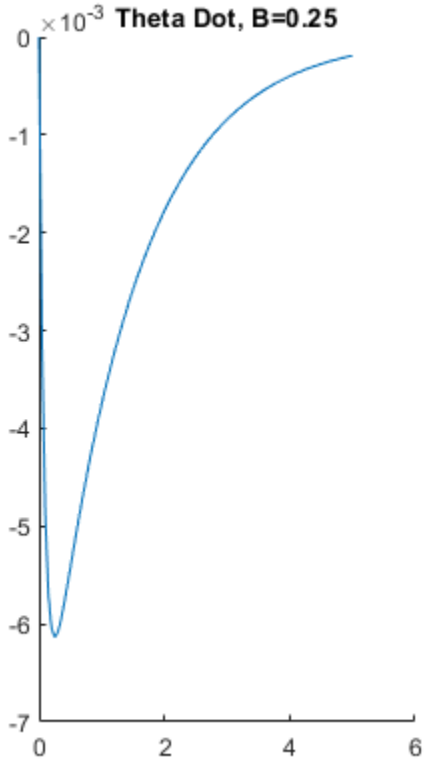

Theta, B=0.035
×10-3
10
8
6
4
2
0
-2
-4
0
2
4
6


Theta Dot, B=0.035
0.01
0.005
0
-0.005
-0.01
-0.015
-0.02
-0.025
0
2
4
6


Theta, B=0.206
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
2
4
6


×10^-3
Theta Dot, B=0.206
0
-1
-2
-3
-4
-5
-6
-7
-8
0
2
4
6


Theta, B=0.25
0.01
0.009
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0
0
2
4
6


×10^-3
Theta Dot, B=0.25
0
-1
-2
-3
-4
-5
-6
-7
0
2
4
6

function definitions

```
function [dydt] = ode1(t,Y) % for ivp1
    y = Y(1);
    yp = -y + 2*t;
    dydt = yp;
end

function [dydt] = ode2(t,Y) % for ivp2
    y = Y(1);
    yp = 3*y*t^2;
    dydt = yp;
end

function [dydt] = pendulum1(t,Y)
    m=0.2;
    l=0.3;
    g=9.81;
    B=0;
    y = Y(1);
    yp = Y(2);
    ypp = -B/(m*l^2)*Y(2)-(g/1)*Y(1);
    dydt = [yp;ypp];
end
function [dydt] = pendulum2(t,Y)
    m=0.2;
    l=0.3;
    g=9.81;
    B=0.035;
    y = Y(1);
    yp = Y(2);
    ypp = -B/(m*l^2)*Y(2)-(g/1)*Y(1);
    dydt = [yp;ypp];
end
function [dydt] = pendulum3(t,Y)
    m=0.2;
    l=0.3;
    g=9.81;
    B=0.206 ;
    y = Y(1);
    yp = Y(2);
    ypp = -B/(m*l^2)*Y(2)-(g/1)*Y(1);
    dydt = [yp;ypp];
end
function [dydt] = pendulum4(t,Y)
    m=0.2;
    l=0.3;
    g=9.81;
    B=0.25  ;
    y = Y(1);
    yp = Y(2);
    ypp = -B/(m*l^2)*Y(2)-(g/1)*Y(1);
    dydt = [yp;ypp];
```

```
end



% methods implementations
function [t,w] = midpoint(f, t0,y0,T,n)
h = (T-t0)/n;
t = linspace(t0, T, n+1)';
w = zeros(n+1,1);
w(1) = y0;
for i=1:n
    w(i+1)=w(i)+h*f(t(i)+h/2,w(i)+h/2*f(t(i),w(i)));
end
end

function [t, w] = euler(f, t0, y0, T, n)
% Implements Euler method for 1st-order IVPs
%   f: right-hand side of ODE
%   t0: Initial time
%   y0: Function value at the initial time
%   T: final time at which estimate is desired
%   n: number of iterations needed
h = (T-t0)/n;
t = linspace(t0, T, n+1)';
w = zeros(n+1,1);
w(1) = y0;
for i=1:n
    w(i+1) = w(i) + h*f(t(i),w(i));
end
end

function [t, w] = explicitTrapezoid(f, t0, y0, T, n)
% Implements the explicit trapezoid method for 1st-order IVPs
%   f: right-hand side of ODE
%   t0: Initial time
%   y0: Function value at the initial time
%   T: final time at which estimate is desired
%   n: number of iterations needed
h = (T-t0)/n;
t = linspace(t0, T, n+1)';
w = zeros(n+1,1);
w(1) = y0;
for i=1:n
    w(i+1) = w(i) + h/2*(f(t(i), w(i)) + f(t(i+1), w(i) + h*f(t(i), w(i))));
end
end

function [t, w] = RK4(f, t0, y0, T, n)
% Implements the 4th-order Runge-Kutta method for 1st-order IVPs
%   f: right-hand side of ODE
%   t0: Initial time
%   y0: Function value at the initial time
%   T: final time at which estimate is desired
%   n: number of iterations needed
```

```
h = (T-t0)/n;
h2 = h/2;
t = linspace(t0, T, n+1)';
w = zeros(n+1,1);
w(1) = y0;
for i=1:n

    s1 = f(t(i), w(i));
    s2 = f(t(i)+h2, w(i)+h2*s1);
    s3 = f(t(i)+h2, w(i)+h2*s2);
    s4 = f(t(i)+h, w(i)+h*s3);

    w(i+1) = w(i) + h/6*(s1 + 2*s2 + 2*s3 + s4);
end
end


function [t, W] = explicitTrapezoidVec(m, F, t0, Y0, T, n)
% Implements the explicit trapezoid method for 1st-order IVPs
%   m: number of equations in system
%   F: right-hand side column vector of the system
%   t0: Initial time
%   Y0: Column vector solution at initial time
%   T: final time at which estimate is desired
%   n: number of iterations needed
h = (T-t0)/n;
t = linspace(t0, T, n+1)';
W = zeros(n+1,m);
W(1,:) = Y0';
for i=1:n

    W(i+1,:) = W(i,:) + h/2*(F(t(i), W(i,:))' ...
        + F(t(i+1), W(i,:) + h*F(t(i), W(i,:))')');
end
end
```

Published with MATLAB® R2022a