```
% MCEN 3030
% Project 6
% MEID: 650-703
% HANWEN ZHAO
function [] = CP6_650703()
clear all
close all
clc
% Problem 1
fprintf('Problem 1\n')
y_init = 0; h1 = 0.1; h2 = 0.01; t_end = 4;
t1 = 0:h1:t_end;
t2 = 0:h2:t_end;
imax1 = length(t1);
imax2 = length(t2);

f = @(t,y) -200000*y + 199000*y^(2/3)*exp(-t) + exp(-t);
g = @(x,y,t,h) x + 200000*x*h - 199000*x.^(2/3)*exp(-t)*h - exp(-t)*h
 - y;
%Take derivative of this g(x) with respect to x
g_p = @(x,y,t,h) 1 + 200000*h - 199000*(2/3)*x^(-1/3)*exp(-t)*h;

% Implicit (Backward) Euler
y_imp1 = Euler_Backward(t1,imax1,h1,g,g_p); % calculate y_imp with
 step size 0.1
y_imp2 = Euler_Backward(t2,imax2,h2,g,g_p); % calculate y_imp with
 step size 0.01

figure
semilogx(t1,y_imp1,'b.-') % plot with setp size h = 0.1
axis([0.1 4 0 0.8])
hold on
semilogx(t2,y_imp2,'r.-') % plot with setp size h = 0.01
legend('h=0.1','h=0.01')
hold off
Error = sum(abs(y_imp1-y_imp2(1:10:end)))/length(y_imp1);
fprintf('Since we dont know the true solution of ODE, we can use more
 costly approach to compare.\n')
fprintf('From the plot above, we can say our Euler solution was an
 accurate solution to the ODE.\n')
fprintf('The average error is %4.8f.\n\n',Error)

% Problem 2
fprintf('Problem 2\n')
fprintf('Also we can use other high oder methods to solve this ODE,
 such as 5th order Runge Kutta or Adams-Moulton method.\n\n')

% Problem 3
fprintf('Problem 3 \n')
[vl] = ProblemThreeFunction(20,40,1.6); % when the bottom of the
 projectile block knife
```

```
[vh] = ProblemThreeFunction(20,40,1.8); % when the top of the
 projectile block knife
[ve] = ProblemThreeFunction(20,40,1.7); % when the middle of the
 projectile block knife
fprintf('The lowest initial velocity we can use is %4.2f.\n',vl)
fprintf('The highest initial velocity we can use is %4.2f.\n',vh)
fprintf('The excate initial velocity we can use is %4.2f.\n\n',ve)

% Problem 4
fprintf('Problem 4 \n')
solinit=bvpinit([0:0.4:2.8],[20,40]);
sol=bvp4c(@odefun,@bcfun,solinit);
figure
plot(sol.x,sol.y(1,:),'r')
xlabel('time')
ylabel('position')

vf = (sol.y(1,2)-sol.y(1,1))/(sol.x(2)-sol.x(1));
fprintf('The calculated initial velocity through finite difference
 approach is %4.2f m/s. \n\n',vf)
end

function dydx=odefun(x,y)
dydx=[y(2) -9.8-0.5/5*y(2)];
end

function res=bcfun(ya,yb)
BCa=-1;
BCb=1.7;
res=[ya(1)-BCa yb(1)-BCb];
end

function [v] = ProblemThreeFunction(v1,v2,height)
timespan = [0 2.8]; % desried time range for solution
IC1 = [-1 v1]; % initial condition
IC2 = [-1 v2];
%Solve ODE via ode45
i = 1;
while(1)
    [t1,y1] = ode45(@ODE3,timespan,[-1 v1]); % calculate positon
 throught the first guess
    [t2,y2] = ode45(@ODE3,timespan,[-1 v2]); % calculate positon
 throught the second guess
    error1 = y1(end,1)-height;
    error2 = y2(end,1)-height;
    if abs(error1) < 0.0001
       break;
    end
    tmp = v2;
    v2 = v1 - error1*(v1-v2)/(error1-error2); % secant method
    v1 = tmp;
end
v = v2;
end
```

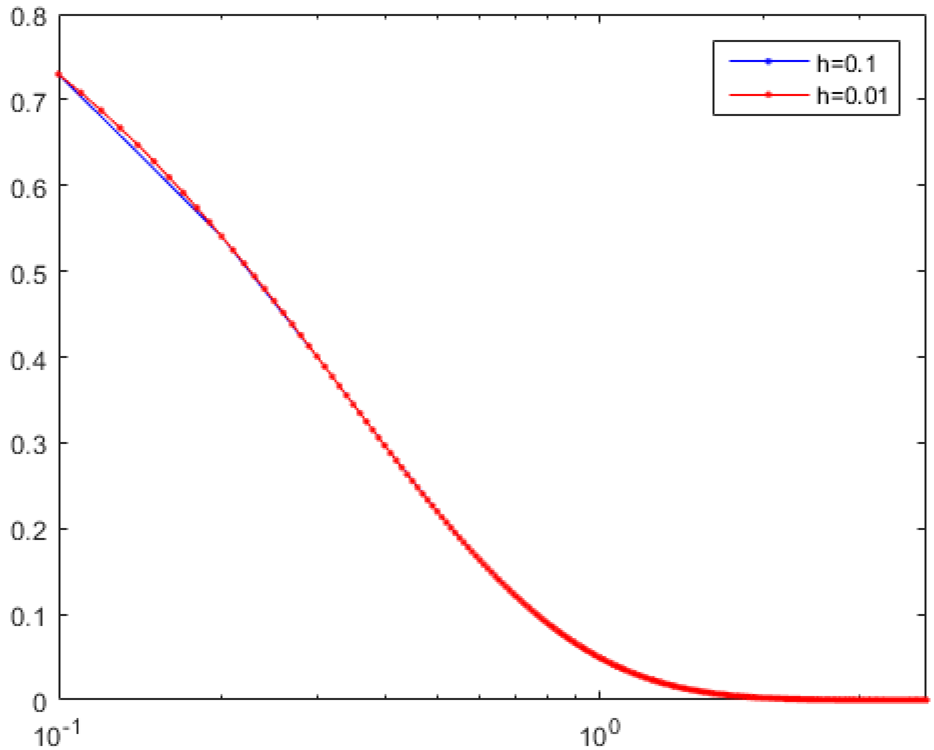
```
function [y_imp] = Euler_Backward(t,imax,h,g,g_p) % euler backward
 function
y_imp = zeros(1,imax);
y_imp(1) = 0.0; %need reasonable initial condition for Newton's method
for i = 1:imax-1
    if i == 1, x = 0.7; else  x = y_imp(i); end %can't divide by zero
    %so don't start Newton iteration at y0 to determine y1, instead
 pick
    %another guess
    while true
        x_new = x - g(x,y_imp(i),t(i+1),h) / g_p(x,y_imp(i),t(i+1),h);
        if abs((x_new-x)/x) < 100*eps  %once we converge
            break
        else
            x = x_new;
        end
    end
    y_imp(i+1) = x;     %set our y value
end
end

function [phi] = ODE3(t,y)
m=5;  %set constants
g=9.81;
c=0.5;
if y(2) > 0  %drag is in negative y when moving up
    phi = [y(2); -g - c/m*(y(2))^2];
else         %other wise it is positive (none when v=0)
    phi = [y(2); -g + c/m*(y(2))^2];
end
end
```

Problem 1
Since we dont know the true solution of ODE, we can use more costly approach to compare.
From the plot above, we can say our Euler solution was an accurate solution to the ODE.
The average error is 0.00000383.

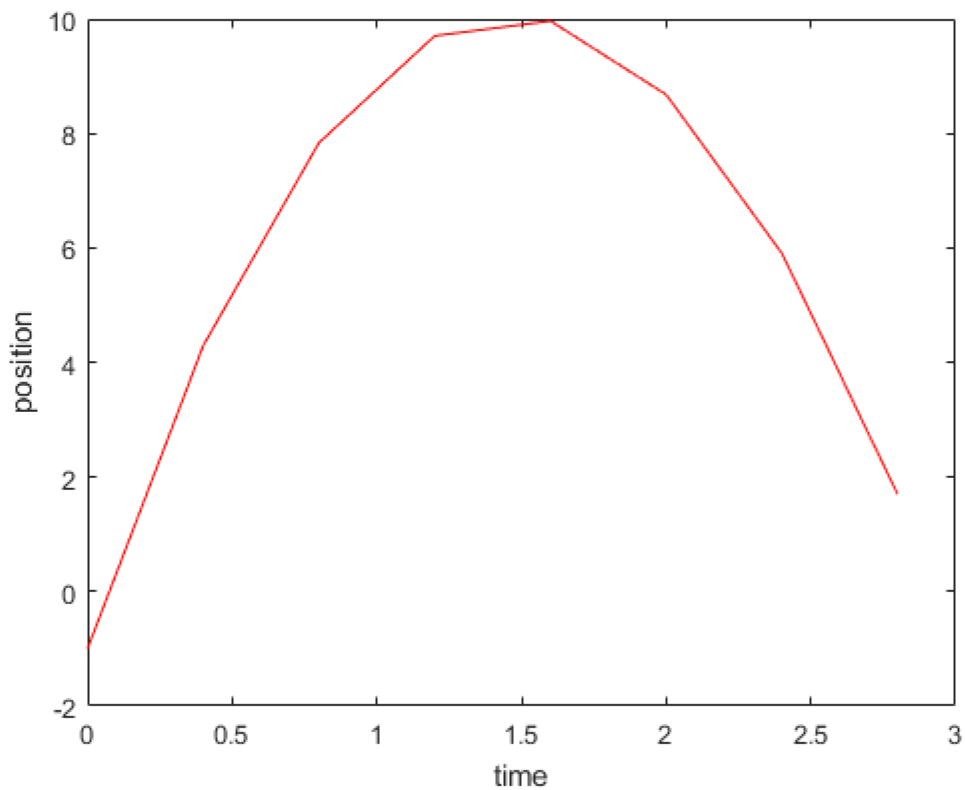
Problem 2
Also we can use other high oder methods to solve this ODE, such as 5th order Runge Kutta or Adams-Moulton method.

Problem 3
The lowest initial velocity we can use is 29.36.
The highest initial velocity we can use is 29.86.
The excate initial velocity we can use is 29.61.

Problem 4
The calculated initial velocity through finite difference approach is 13.23 m/s.



0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10^{-1}
10^{0}
h=0.1
h=0.01



position
time
10
8
6
4
2
0
-2
0
0.5
1
1.5
2
2.5
3

Published with MATLAB® R2015b