Lee Sutton
48507107

Mech 358 - Computer Lab 1

Problem 1

A)

The m – files for problem 1a are shown below. The plot of the analytical solution versus the numerical solution is shown below in Figure 1.

```
%this M-file will calculate the solution to the ODE and plot it

%initial conditions and constants
x0 = [0 2];
b=1;

%time interval
t=[-b b]

%solve the ODE
[T,X] = ode45(@poise,t,x0)

%plot the solution with blue stars
plot(X(:,1),T,'*')

%compare with the analytical soln.
y = linspace(-1,1);
u = 1-y.^2;

%plot the analytical soln. in green on the same plot
hold on;
plot(u,y,'g')

%add labels
xlabel('u')
ylabel('y')
title('Numerical Solution and Analytical Solution to Flow in a Channel')
legend('Numerical Solution','Analytical Solution')
```

```
%This function will calculate f(u,y)

function f = poise(t,x)
%initialize G
G = -2;
f=zeros(2,1)

f(1) = x(2);
f(2) = G;
```

Lee Sutton
48507107

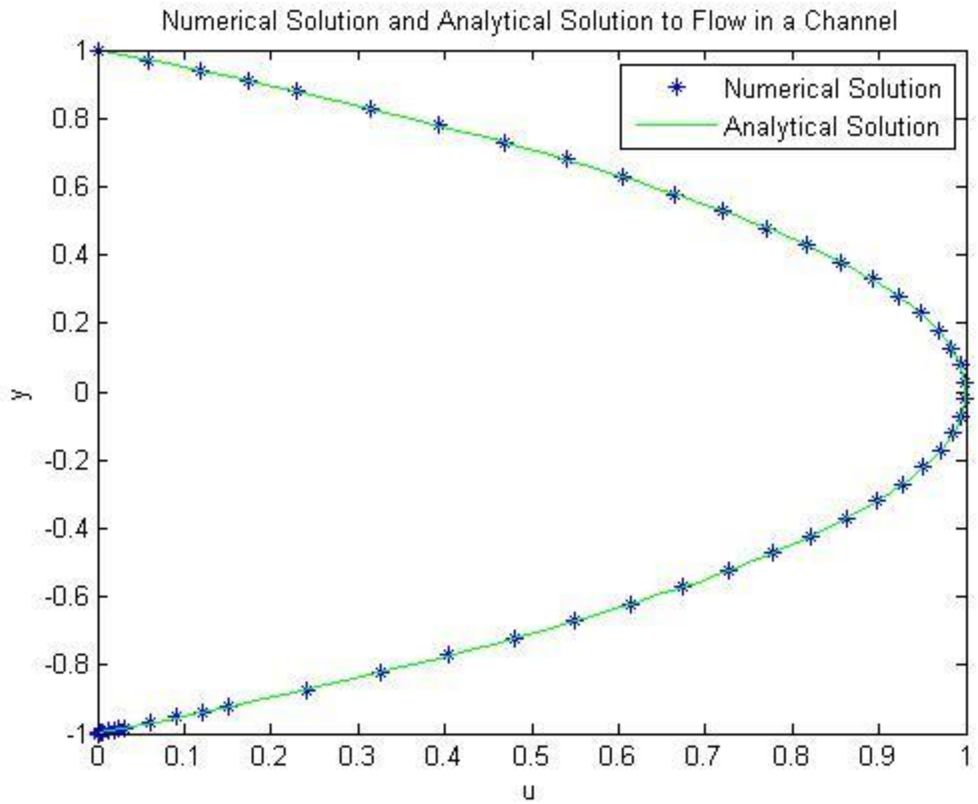

Figure 1: Plot of the Numerical and Analytical Solution for Flow in a Channel (Problem 1A)

B)

The m-file used for problem 1b is shown below. The poise function shown in part A was also used in this part of the lab

```
%This m-file will solve the ODE using a shooting method with ODE 45
clear;
clc;
%constants
b=1;
G=-2;
V=1;
y = [-b b];


%initially we have two guesses
x01 = [0 1];      %intial guess #1 u(-b)=1 & u'(-b)=1
x02 = [0 4];      %intial guess #1 u(-b)=1 & u'(-b)=4

%need to solve each ODE

%solve for the initial guess #1
[Y1,U1] = ode45(@poise_flow,y,x01);
```

Lee Sutton
48507107

```
%solve for the initial guess #2
[Y2,U2] = ode45(@poise_flow,y,x02);

% %plot velocity vs. y
% plot(U1(:,1),Y1)        %for first guess soln.
% hold on;
% plot(U2(:,1),Y2)


%check the relative errors for each at u(b)
a = size(U1);    %need to evaluate U at its last value
B = a(1);        %need the number of rows in U1 to evaluate the last value
rel1 = abs((U1(B)-V)/V);
rel2 = abs((U2(B)-V)/V);

%we need to classify each as either positive or negative so we can decide
%which point to discard after converging we assume each guess gives us
%opposite signs around the root
if (U1(B)-V)>0
   Upos = U1;
   Uneg = U2;
else
    Uneg = U1;
    Upos = U2;
end

rel_error = min(rel1,rel2)

%create a while loop to converge the solution

while rel_error> 10^-4      %we want the soln. to 4 sig figs

    %converge the soln. using false position method
    U3_prime = -Upos(B)*(Uneg(1,2)- Upos(1,2))/(Uneg(B)-V - Upos(B)-V) +
Upos(1,2)

    %new intial conditions
    x03 = [0 U3_prime];

    %solve with new intial conditions
    [Y3,U3] = ode45(@poise_flow,y,x03);

    %classify U3(B) as a positive or negative root
    if (U3(B)-V)>0
       Upos = U3;

    else
        Uneg = U3;

    end

    %find new relative error
    rel_error = abs((U3(B)-V)/V);
```

```
end

%plot the final solution with the initial guesses
plot(U3(:,1),Y3,'*r')
hold on

%plot it against the actual solution
y = linspace(-b,b);
u = -1*y.^2+0.5*y+1.5;
plot(u,y,'g');

%add labels
xlabel('u')
ylabel('y')
title('Numerical and Analytical Solution for Flow in a Channel')
legend('Numerical Solution', 'Analytical Solution')
```

The plot for part 1b is shown below in Figure 2

The analytical solution was solved by integrating the equation twice and using the boundary conditions to find the integration constants. The analytical solution is shown below in Equation 1

Equation 1

$$u = -y^2 + 0.5y + 1.5$$

Lee Sutton
48507107

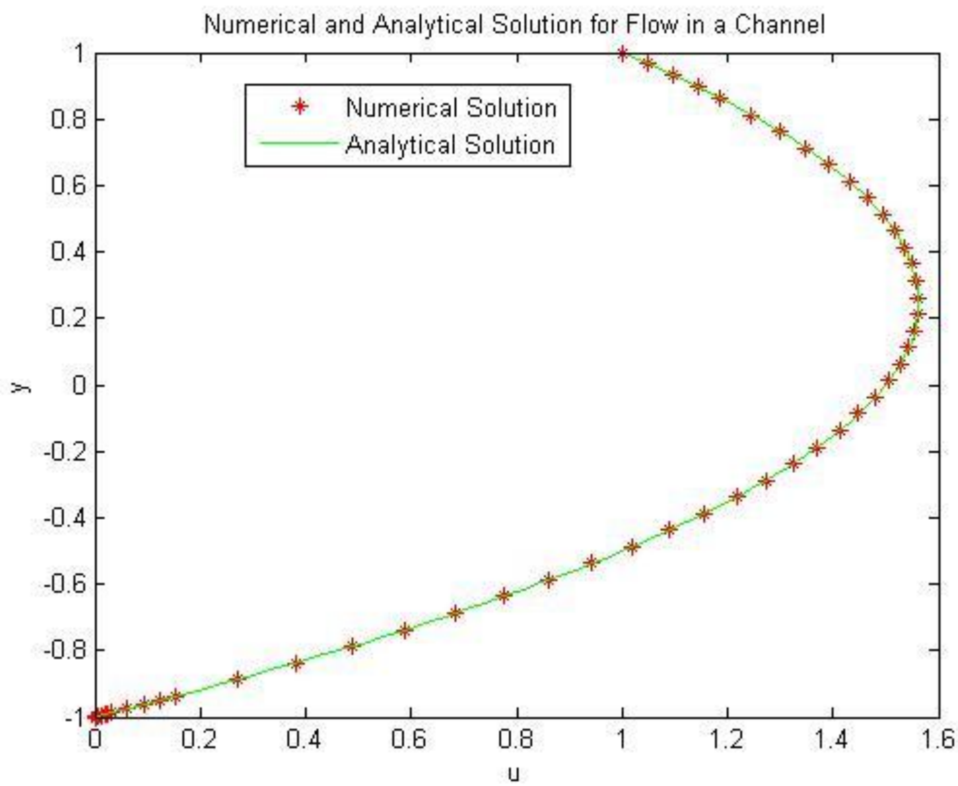

Figure 2: Analytical and Numerical Solution for Flow in a Channel

The numerical value at u’(-b) was also found for V=1

$$U(-1) = 2.500$$

Problem 2

To find guesses that bracket y(1) = 1.5 I used trial and error. I guessed 2 values for y’(0) and plotted their solutions. Then adjusted the guesses until both solutions bracketed y(1)=1.5. The plot for the final guesses is shown below in Figure 3.

Lee Sutton
48507107

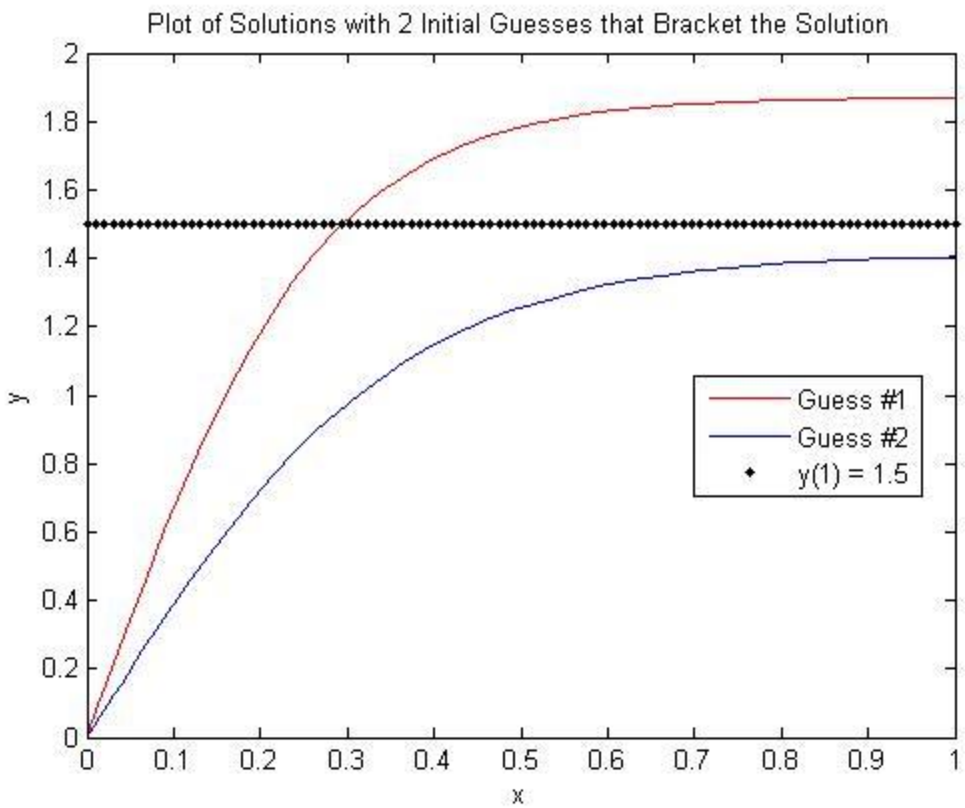

Figure 3: Plot of Numerical Solutions to the ODE that Bracket the Correct Solution

The m file that was used to do this is shown below along with the function used to evaluate y'.

```
%This m-file will solve the ODE using a shooting method with ODE 45
clear;
clc;
%constants
V=1.5;      % y(1)=1.5 call this V
x = [0 1];


%initially we have two guesses
x01 = [0 7];     %intial guess #1 u(-b)=1 & u'(-b)=1
x02 = [0 4];     %intial guess #1 u(-b)=1 & u'(-b)=4

%need to solve each ODE

%solve for the initial guess #1
%note that U = Y it was left this way after using this mfile to solve
%problem 1
[X1,U1] = ode45(@yprime,x,x01);

%solve for the initial guess #2
```

Lee Sutton
48507107

```
[X2,U2] = ode45(@yprime,x,x02);

%plot the functions to check if the solved equations bracket the solution
plot(X1,U1(:,1),'r')         %for first guess soln.
hold on;
plot(X2,U2(:,1))         %for the second guess

%add in a plot for y(1)=1.5 so we can see clearly that the solutions
%bracket y(1)
xbracket = linspace(0,1);
ybracket = 1.5;
plot(xbracket,ybracket,'.k')

%Label the plot
xlabel('x')
ylabel('y')
title('Plot of Solutions with 2 Initial Guesses that Bracket the Solution')
legend('Guess #1', 'Guess #2','y(1) = 1.5')
```

Function to evaluate y'

```
function f = yprime(t,y)

%intialize as a column vector
f = zeros(2,1);

f(1) = y(2);
f(2) = -4*y(1)*y(2);
```

After bracketing the solution, the m-file from problem 1 had to be adjusted to solve this ODE the mfile used for this problem is shown below.

```
%This m-file will solve the ODE using a shooting method with ODE 45
clear;
clc;
%constants
V=1.5;      % y(1)=1.5 call this V
x = [0 1];
iteration_num = 1;       %this is a counter for relative error etc.

%initially we have two guesses
x01 = [0 7];     %intial guess #1 u(-b)=1 & u'(-b)=1
x02 = [0 4];     %intial guess #1 u(-b)=1 & u'(-b)=4

%need to solve each ODE

%solve for the initial guess #1
%note that U = Y it was left this way after using this mfile to solve
%problem 1
[X1,U1] = ode45(@yprime,x,x01);
```

Lee Sutton
48507107

```
%solve for the initial guess #2
[X2,U2] = ode45(@yprime,x,x02);

% %plot the functions to check if the solved equations bracket the solution
% plot(X1,U1(:,1),'r')          %for first guess soln.
% hold on;
% plot(X2,U2(:,1))          %for the second guess



%check the relative errors for each at u(b)
a = size(U1);    %need to evaluate U at its last value
B = a(1);        %need the number of rows in U1 to evaluate the last value
rel1 = abs((U1(B)-V)/V);
rel2 = abs((U2(B)-V)/V);

%we need to classify each as either positive or negative so we can decide
%which point to discard after converging we assume each guess gives us
%opposite signs around the root
if (U1(B)-V)>0
   Upos = U1;
   Uneg = U2;
else
    Uneg = U1;
    Upos = U2;
end

rel_error = min(rel1,rel2);

%create a while loop to converge the solution

while rel_error> 10^-3      %we want the soln. to 3 sig figs

    %converge the soln. using false position method
    U3_prime = -Upos(B)*(Uneg(1,2)- Upos(1,2))/(Uneg(B)-V - Upos(B)-V) +
Upos(1,2);

    %new intial conditions
    x03 = [0 U3_prime];

    %solve with new intial conditions
    [X3,U3] = ode45(@yprime,x,x03);

    %classify U3(B) as a positive or negative root
    if (U3(B)-V)>0
       Upos = U3;

    else
        Uneg = U3;

    end

    %find new relative error store it as a vector so it can be plotted
    rel_error(iteration_num) = abs((U3(B)-V)/V);
    iteration_num = iteration_num + 1;
```

Lee Sutton
48507107

```
    %vectors to save y'(0) and y(1) vs the iteration number
    yprime0(iteration_num) = U3(1,2);
    y1(iteration_num) = U3(B);

    %need the successive relative error in y'(0) and y(1) so we can plot it
    %later

    %for error in y(1)
    err_y0(iteration_num) = abs((y1(iteration_num)-V)/V);

    %for the error  in y' we dont have an intial value to subtract from, we
    %have to wait until the second iteration to calculate the error in y'

    if iteration_num>1
    err_in_yprime(iteration_num) = abs((yprime0(iteration_num)-
yprime0(iteration_num-1))/yprime0(iteration_num));
    end

end

%plot the final solution as a blue line
plot(X3,U3(:,1))
hold on

%add labels
xlabel('x')
ylabel('y')
title('Numerical Solution to the BVP in Problem 2')
legend('Numerical Solution')

%create a new figure to plot y'(0) and y(1) as a function of the iteration
%number
figure;

plot(1:iteration_num,yprime0,'xb');

%add labels
xlabel('Iteration Number')
title('dy/dx at x=0 and y(1) Plotted Against the Iteration Number')

%plot it on the same figure
hold on
plot(1:iteration_num,y1,'or');

%add labels
xlabel('Iteration Number')
legend('Yprime(0)','Y(1)')
title('y(1) Plotted Against the Iteration Number')

%now we plot the relative errors vs the iteration number

figure
```

Lee Sutton
48507107

```
plot(1:iteration_num, err_y0,'or')
hold on
plot(1:iteration_num, err_in_yprime,'xk')

%add labels
xlabel('Iteration Number')
legend('Error in Y(1)','Error in Yprime(0)')
title('Error in y(1) and y"(0) Plotted Against iteration number')
```

The plot of the numerical solution is shown below in Figure 4.

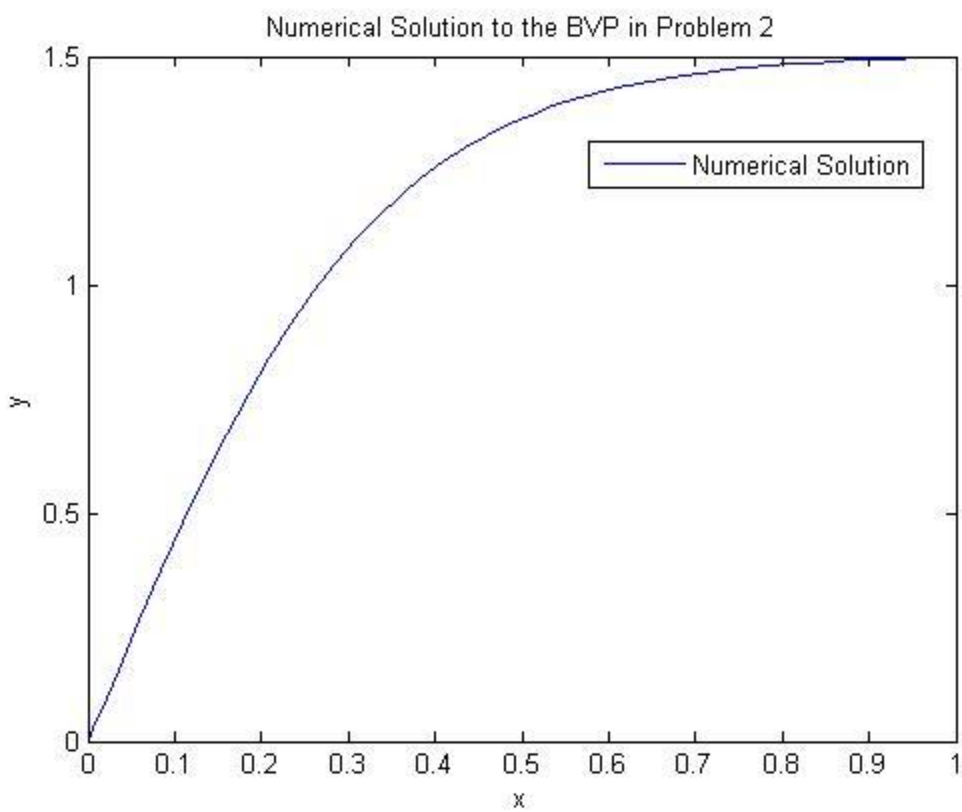

Figure 4: Numerical Solution to the BVP for Problem 2

The numerical value for y'(0):

Y'(0) = 4.5369

Lee Sutton
48507107

A plot of y'(0) and y(1) versus the iteration number is shown below in Figure 5.

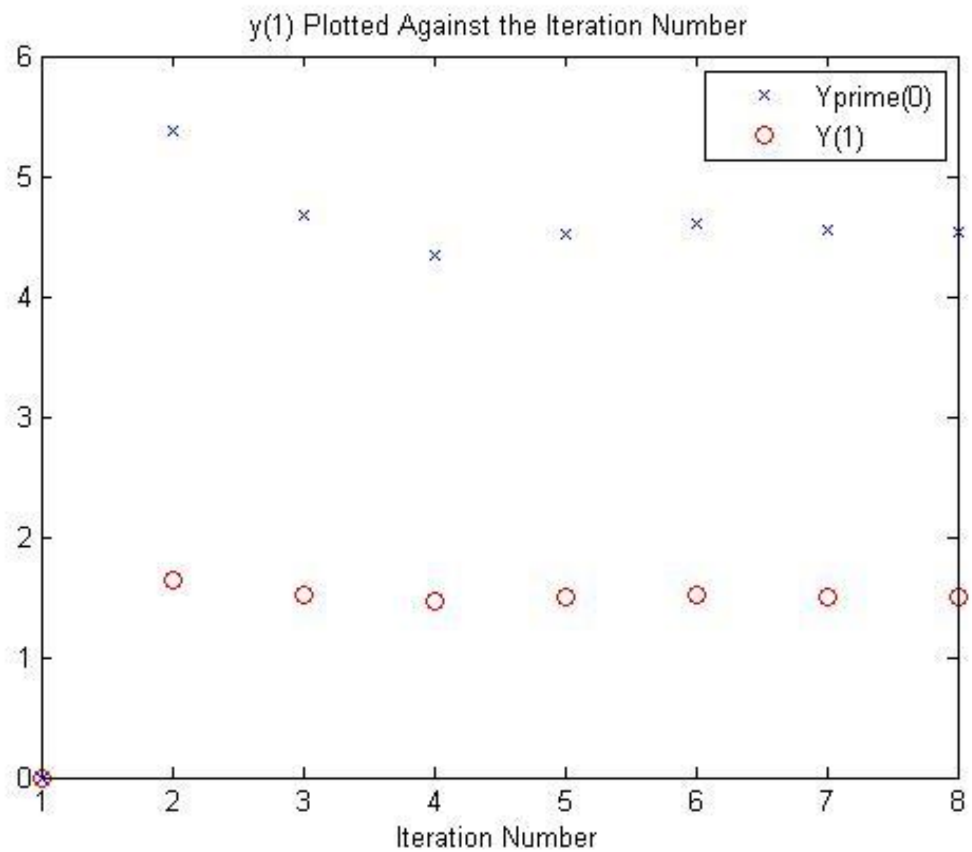

Figure 5: Plot of y'(0) and y(1) Versus the Iteration Number

A plot of the relative error is shown below in Figure 6.

Lee Sutton
48507107

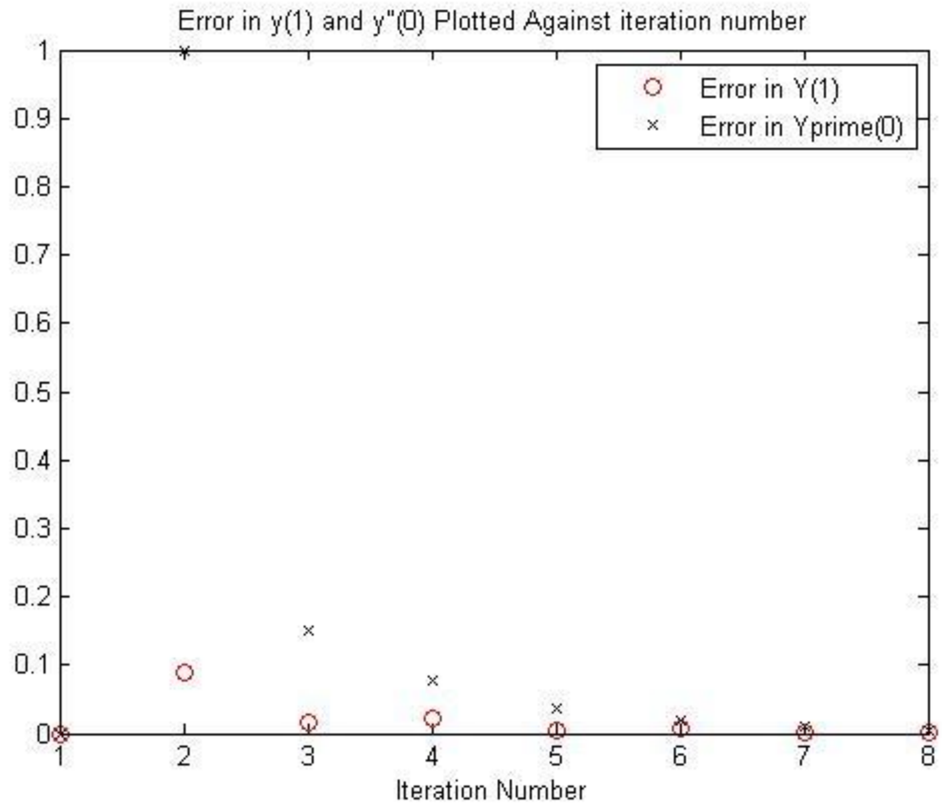

Figure 6: Error in y(1) and y'(0) versus the iteration number