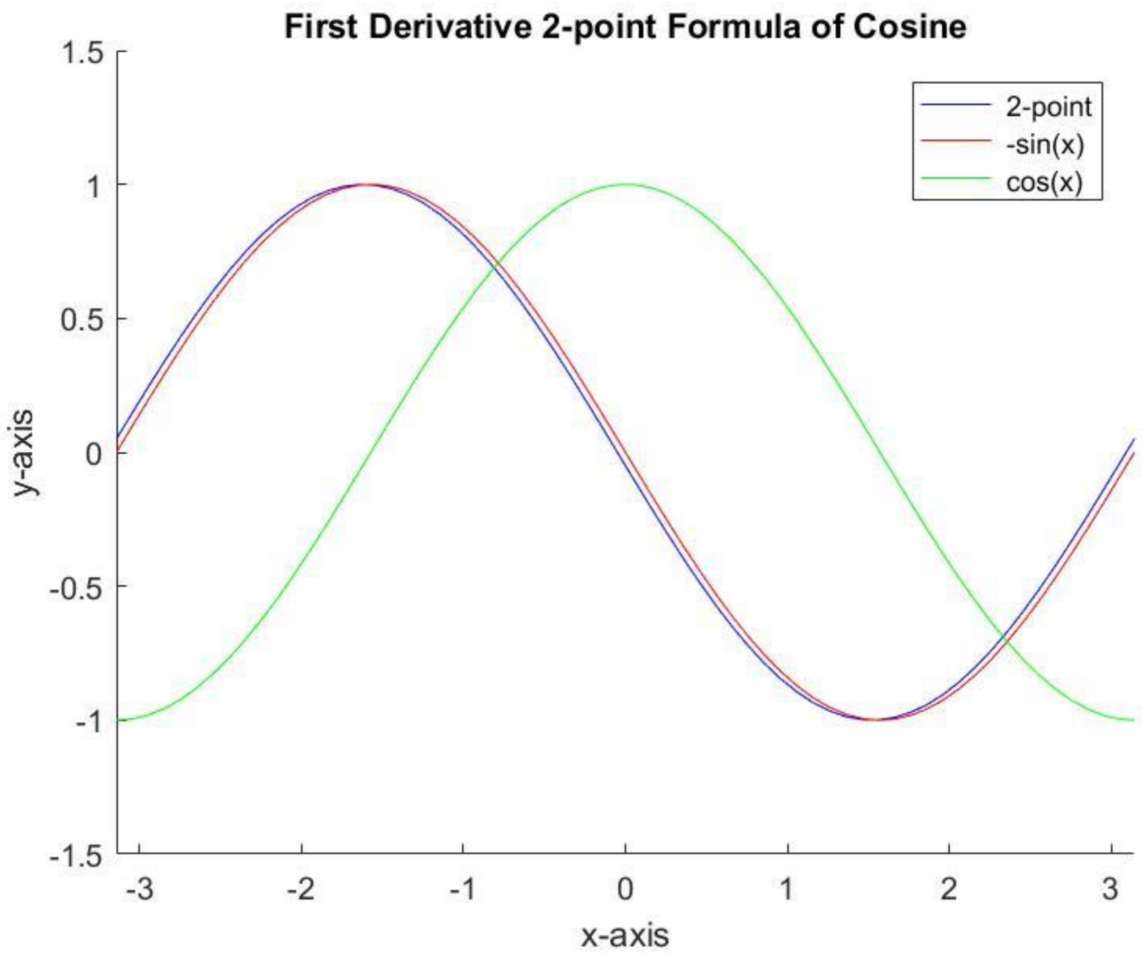
Andrew Camps

ECE 330

Homework 2

9/11/2016

Problem 1:



About Graph:

The graph above is the derivative of the cosine function F(x) = cos(x) using the two-point formula shown in the function code below. You can see the leading error from the two point formula on the blue graph. There is a value of h = 0.1 and about 63 test points.

Function Code:

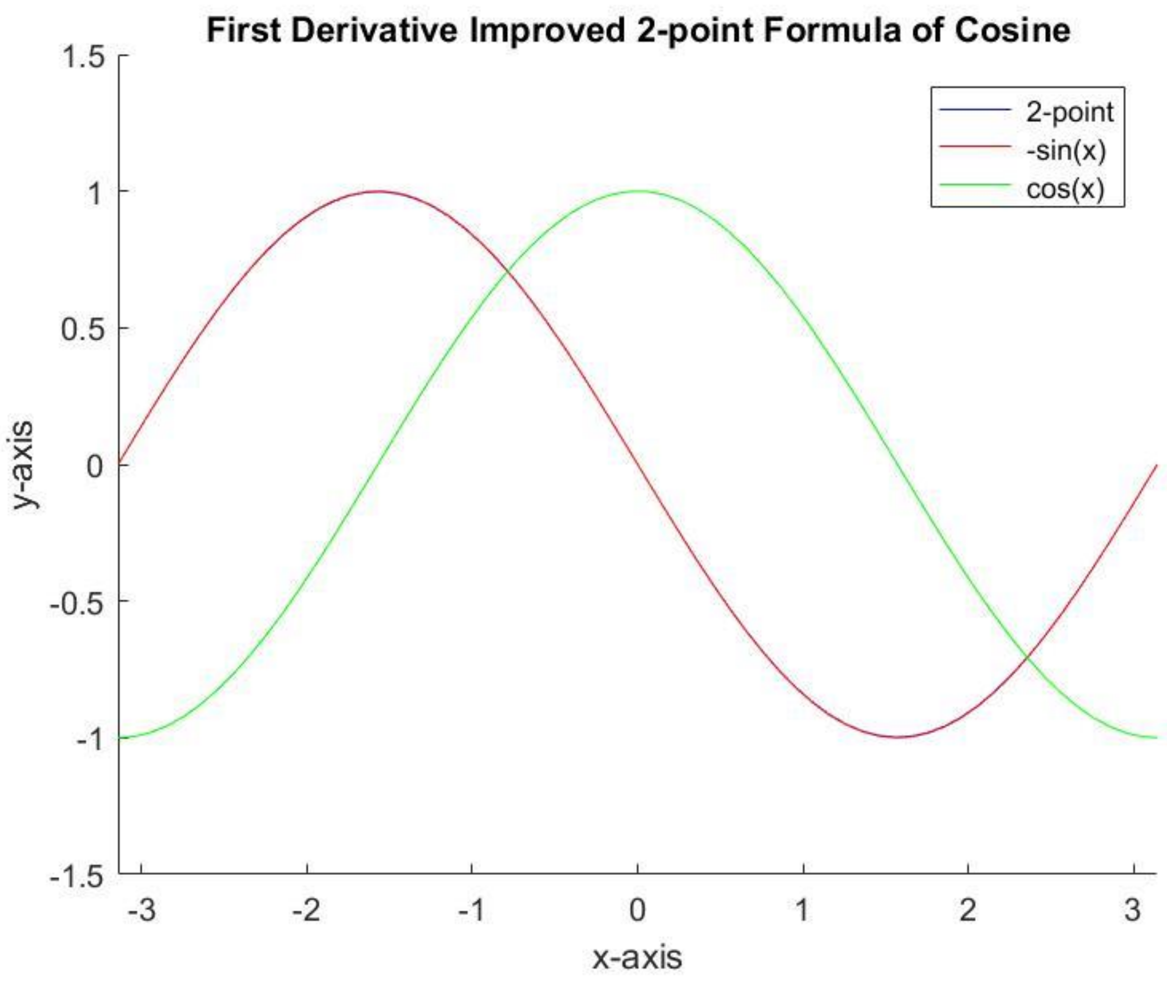
```
function [y] = firstDeriv( fh, h, lowerBound, upperBound )
%First Derivative Two-Point Formula
```

```
%   Calculates the derivative of a function when provided a function, h the
step value, upper and lower bounds

numPoints = (upperBound - lowerBound) / (h) + 1; %Number of points
yA = zeros(1, ceil(numPoints)); %inizialization
x = zeros(1, ceil(numPoints));
k = 1;
x(1) = lowerBound; %Sets first value in array

while k <= numPoints + 1 %goes through all the points
   yA(k) = (fh(x(k) + h) - fh(x(k))) / h; %Actual calculation
   k = k + 1;
   x(k) = x(k - 1) + h; %Increase the step

end

y = yA; %Set output

end
```

Plotting Code:

```
%Problem 1: First derivative of cosine with Two-point formula
clear
h = 0.1;
lowerBound = -pi;
upperBound = pi;

x = lowerBound:0.1:upperBound + 0.1; %x included step values
xDer = lowerBound:h:upperBound + h; %x for two point formula

y1 = firstDeriv(@cos, h, lowerBound, upperBound); %two point formula
y2 = -sin(x);
y3 = cos(x);

hold on

plot(xDer, y1, 'b-')
plot(x, y2, 'r-')
plot(x, y3, 'g-')

axis([lowerBound upperBound -1.5 1.5])
xlabel('x-axis')
ylabel('y-axis')
title('First Derivative 2-point Formula of Cosine')
legend(' 2-point', ' -sin(x)', ' cos(x)')
```

Problem 2:



About Graph:

The graph above is the derivative of the cosine function F(x) = cos(x) using the improved two-point formula. It is very hard to see the error from this formula in the blue graph because the error is much smaller in the improved formula. There is a value of h = 0.1 and about 63 test points.

Function Code:

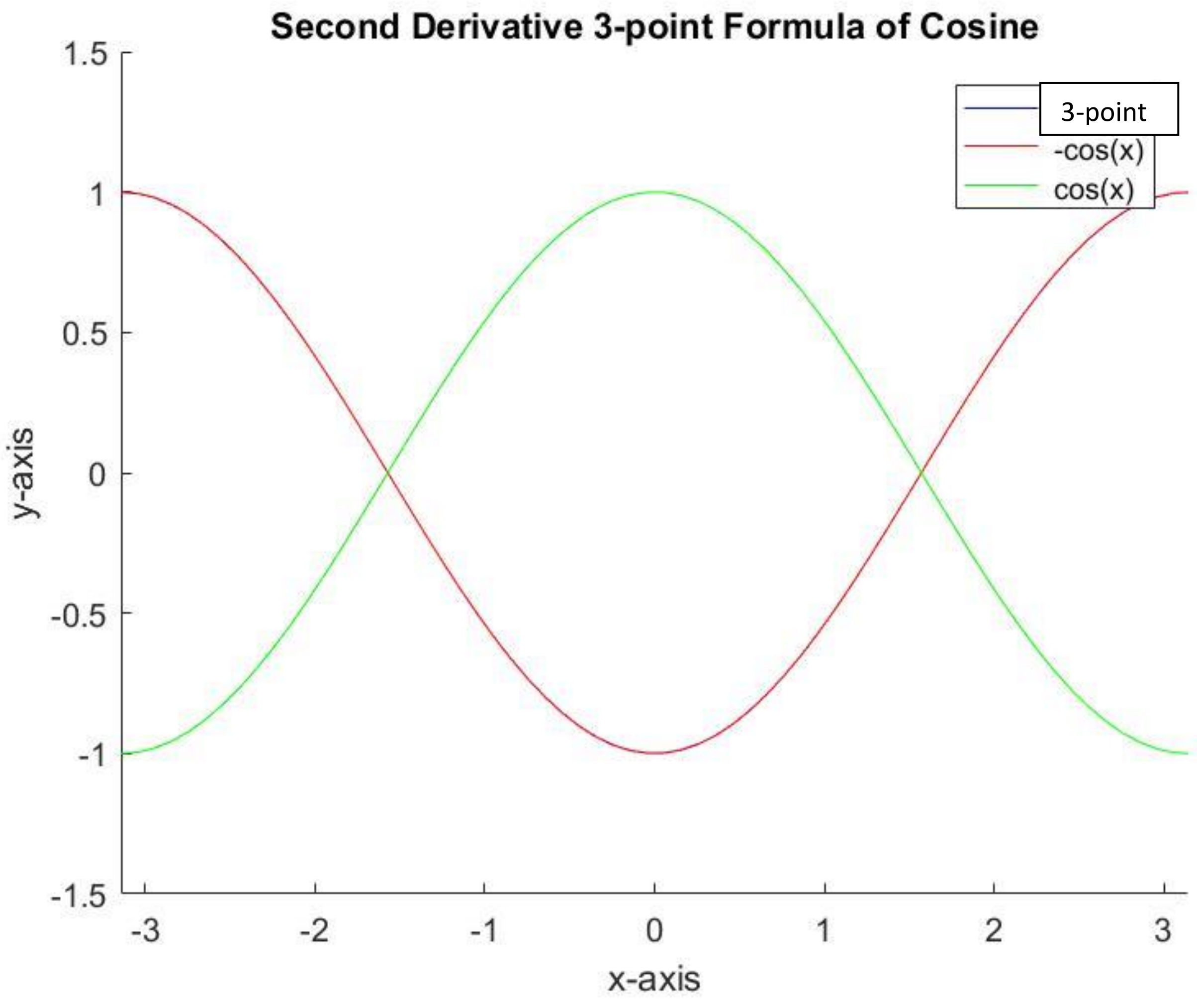
```
function [ y ] = firstDerivImproved( fh, h, lower, upper )
%First Derivative Improved Two-Point Formula
%   Calculates the derivative of a function when provided a function, h the
step value, upper and lower bounds

numPoints = (upper - lower) / (h) + 1; %Number of points
yA = zeros(1, ceil(numPoints)); %Inizalization
x = zeros(1, ceil(numPoints));
k = 1;
x(1) = lower;

while k <= numPoints + 1
```

```
   yA(k) = (fh(x(k) + h) - fh(x(k) - h)) / (2 * h); %Improved two point
calculation
   k = k + 1;
   x(k) = x(k - 1) + h;

end

y = yA;

end
```

Plotting Code:

```
%Problem 2: First derivative of cosine with improved Two-Point formula
clear
h = 0.1;

lowerBound = -pi;
upperBound = pi;

x = lowerBound:0.1:upperBound + 0.1;
xDer = lowerBound:h:upperBound + h;

y1 = firstDerivImproved(@cos, h, lowerBound, upperBound);
y2 = -sin(x);
y3 = cos(x);

hold on

plot(xDer, y1, 'b-')
plot(x, y2, 'r-')
plot(x, y3, 'g-')

axis([lowerBound upperBound -1.5 1.5])
xlabel('x-axis')
ylabel('y-axis')
title('First Derivative Improved 2-point Formula of Cosine')
legend(' 2-point', ' -sin(x)', ' cos(x)')
```

Problem 3:



About Graph:

The graph above is the second derivative of the cosine function F(x) = cos(x) using the three point formula. The red and blue graphs are of the second derivative. It is very hard to see the blue graph which is from the 3-point formula, because the error is very small as well. There is a value of h = 0.1 and about 63 test points.

Function Code:

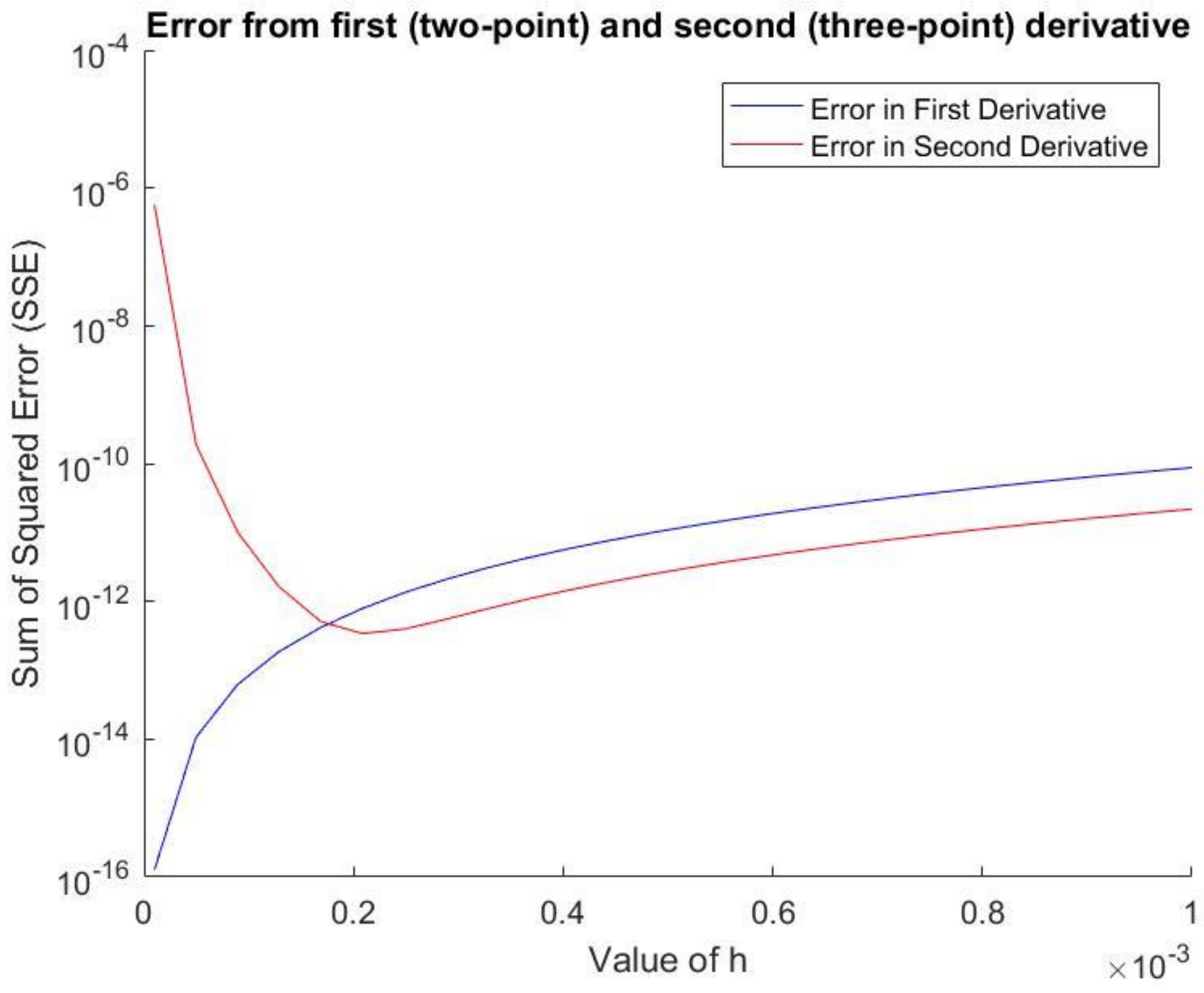
```
function [ y ] = secondDeriv( fh, h, lower, upper )
%Second Derivative using Three-Point Formula
%   Calculates the second derivative of a function given the function, h
%   the step size, lower and upper bounds

numPoints = (upper - lower) / (h) + 1; %Number of points
yA = zeros(1, ceil(numPoints)); %Initialization
x = zeros(1, ceil(numPoints));
k = 1;
x(1) = lower;
```

```
while k <= numPoints + 1
   yA(k) = (fh(x(k) + h) + fh(x(k) - h) - (2 * fh(x(k)))) / (h * h);
%Calculation of three point formula
   k = k + 1;
   x(k) = x(k - 1) + h;

end

y = yA;

end
```

Plotting Code:

```
%Problem 3: Second Derivative of Cosine with three-point formula
clear
h = 0.1;
lowerBound = -pi;
upperBound = pi;
x = lowerBound:0.1:upperBound + 0.1;
xDer = lowerBound:h:upperBound + h;
y1 = secondDeriv(@cos, h, lowerBound, upperBound);
y2 = -cos(x);
y3 = cos(x);

hold on

plot(xDer, y1, 'b-')
plot(x, y2, 'r-')
plot(x, y3, 'g-')

axis([lowerBound upperBound -1.5 1.5])
xlabel('x-axis')
ylabel('y-axis')
title('Second Derivative 3-point Formula of Cosine')
legend(' 2-point', ' -cos(x)', ' cos(x)')
```

Problem 4:



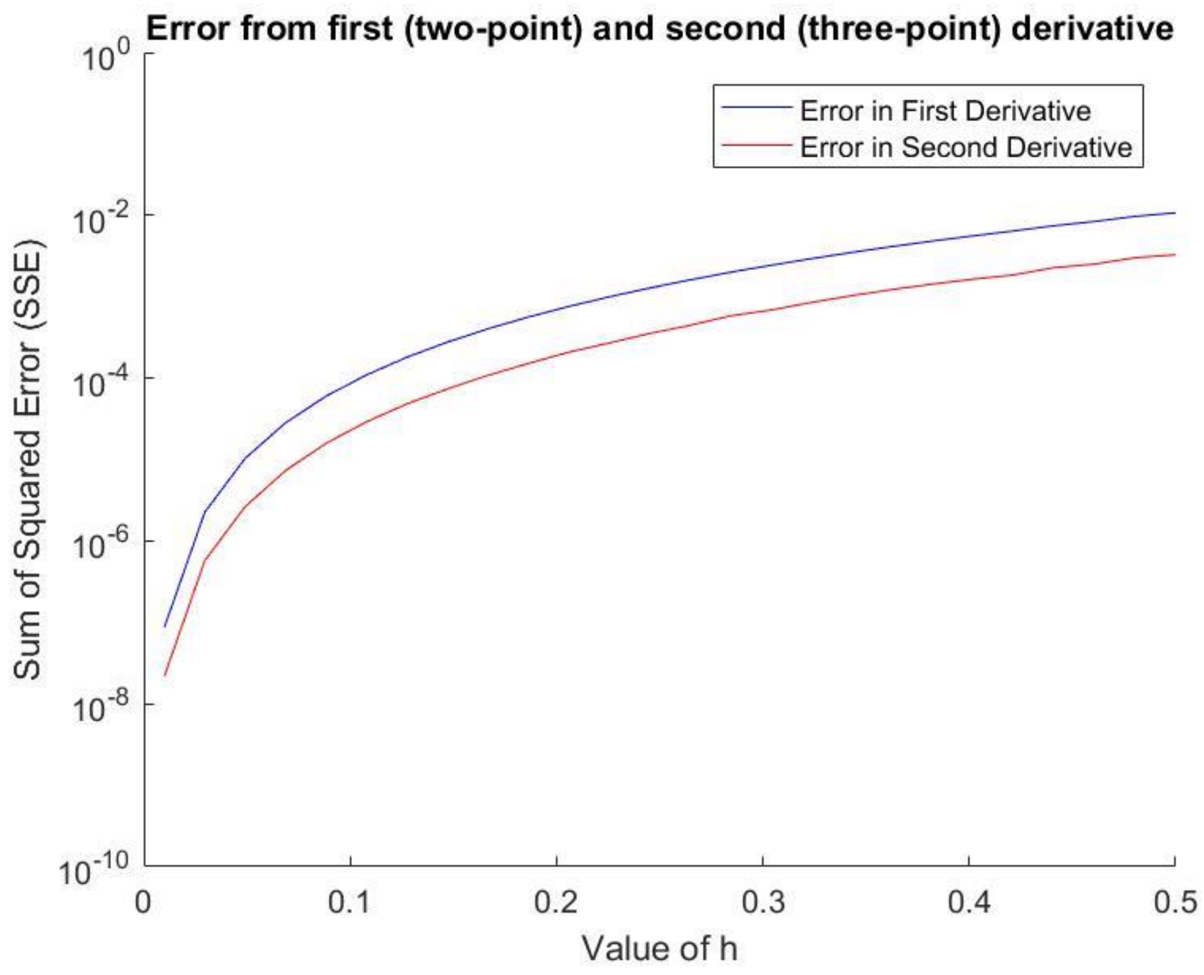
About Graph:

Problem 4 asked us to graph the sum of the squared error between the calculated derivative and the analytical derivative with different values of h. In the question it said to use the not improved version of the two point formula, but I was unable to get the same results as they had in the prompt unless I used the improved version of the derivative which is shown above. The scale of h is from 0.00001 – 0.001 and is cycled over these different values of h in the code below and the error is calculated for each value of h.

Plotting Code:

```
%Problem 4: Graphs of Error from two point and three point formulas
clear
lowerBound = -pi;
upperBound = pi;

k = 1;
h = 0.00001;
x = lowerBound:h:upperBound + h;
```

```matlab
numPoints = 25; %number of h values to try
stepVal = (0.001 - 0.00001) / numPoints; %Spaces them out evenly
xaxis = 0.00001:stepVal:0.001; %for graphing the x

while k <= numPoints + 1; %Loop to try many different h values to graph

    y1 = firstDerivImproved(@cos, h, lowerBound, upperBound); %had to use the
improved first derivative to match the plot in promt. Not improved one had
more error and was higher on the y axis
    y2 = secondDeriv(@cos, h, lowerBound, upperBound); %Three-point formula
    y3 = -cos(x); %Calculated first and second derivatives
    y4 = -sin(x);

    error1 = abs(y4 - y1); %Graps the error of both |actual - calculated|
    error1 = error1.^2; %Squares all the errors
    error2 = abs(y3 - y2);
    error2 = error2.^2;
    sumError1(k) = sum(error1); %Sums the total error
    sumError2(k) = sum(error2);

    k = k + 1;
    h = h + stepVal; %To the next size of h
    x = lowerBound:h:upperBound + h; %update the array for x

end

hold on

plot(xaxis, sumError1, 'b-')
plot(xaxis, sumError2, 'r-')
set(gca, 'yscale', 'log')
xlabel('Value of h')
ylabel('Sum of Squared Error (SSE)')
title('Error from first (two-point) and second (three-point) derivative')
legend('Error in First Derivative', 'Error in Second Derivative')
axis([0 0.001 10^-16 10^-4])
```



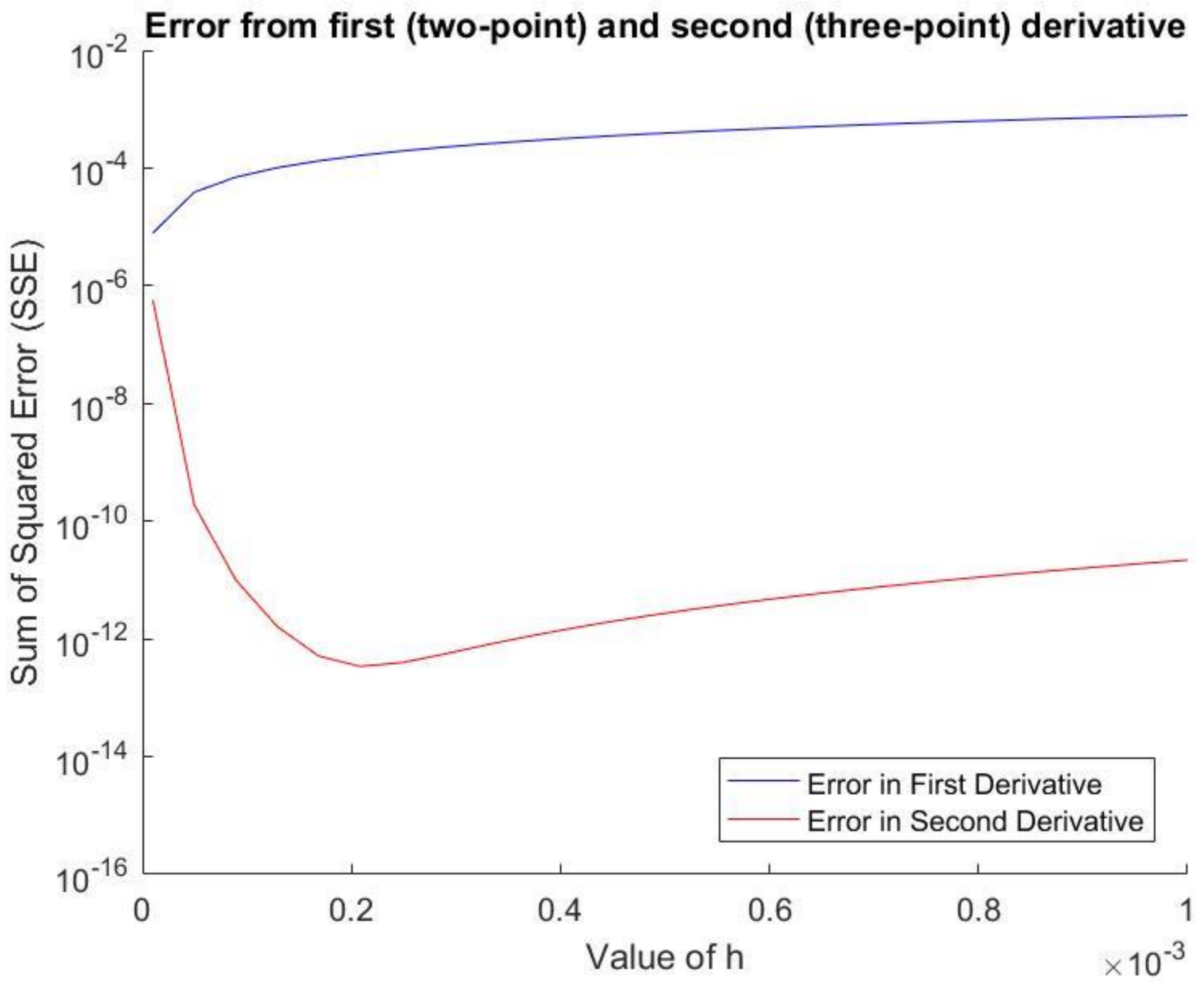
About Graph:

This is the same graph from above but with larger h values used to calculate the error. The values of h are from 0.01 – 0.5.

Plotting Code:

```
%Problem 4: Different x-Axis
%commenting is all the same as above problem only change is larger h values
clear
lowerBound = -pi;
upperBound = pi;

k = 1;
h = 0.01;
x = lowerBound:h:upperBound + h;
numPoints = 25;
stepVal = (0.5 - 0.01) / numPoints; %Larger h values here
xaxis = 0.01:stepVal:0.5;

while k <= numPoints + 1;
```

```
    y1 = firstDerivImproved(@cos, h, lowerBound, upperBound); %had to use the
improved first derivative to match the plot in promt. Not improved one had
more error and was higher on the y axis
    y2 = secondDeriv(@cos, h, lowerBound, upperBound);
    y3 = -cos(x);
    y4 = -sin(x);

    error1 = abs(y4 - y1);
    error1 = error1.^2;
    error2 = abs(y3 - y2);
    error2 = error2.^2;
    sumError1(k) = sum(error1);
    sumError2(k) = sum(error2);

    k = k + 1;
    h = h + stepVal;
    x = lowerBound:h:upperBound + h;

end

hold on

plot(xaxis, sumError1, 'b-')
plot(xaxis, sumError2, 'r-')
set(gca, 'yscale', 'log')
xlabel('Value of h')
ylabel('Sum of Squared Error (SSE)')
title('Error from first (two-point) and second (three-point) derivative')
legend('Error in First Derivative', 'Error in Second Derivative')
axis([0 0.5 10^-10 10^0])
```



About Graph:

These are the results I found when I added the not improved two point formula to the graph. As you can see the error in the not improved two point formula is much more than the improved two point.

Plotting Code:

```
%What problem 4 really asked for
%commenting is all the same as above problem only change is different
%formula graphed
clear
lowerBound = -pi;
upperBound = pi;

k = 1;
h = 0.00001;
x = lowerBound:h:upperBound + h;
numPoints = 25;
stepVal = (0.001 - 0.00001) / numPoints;
xaxis = 0.00001:stepVal:0.001;

while k <= numPoints + 1;
```

```
    y1 = firstDeriv(@cos, h, lowerBound, upperBound); %Not improved two point
formula
    y2 = secondDeriv(@cos, h, lowerBound, upperBound);
    y3 = -cos(x);
    y4 = -sin(x);

    error1 = abs(y4 - y1);
    error1 = error1.^2;
    error2 = abs(y3 - y2);
    error2 = error2.^2;
    sumError1(k) = sum(error1);
    sumError2(k) = sum(error2);

    k = k + 1;
    h = h + stepVal;
    x = lowerBound:h:upperBound + h;

end

hold on

plot(xaxis, sumError1, 'b-')
plot(xaxis, sumError2, 'r-')
set(gca, 'yscale', 'log')
xlabel('Value of h')
ylabel('Sum of Squared Error (SSE)')
title('Error from first (two-point) and second (three-point) derivative')
legend('Error in First Derivative', 'Error in Second Derivative', 'location',
'southeast')
axis([0 0.001 10^-16 10^-2])
```

Problem 4 written [DLV]:

Shown in the graphs in problem 4 the second derivative increase in error as h becomes really small which is strange because with a smaller h value one would think this would improve the accuracy of the derivative. This is caused by the h^2 term in denominator of the three point formula. The error is compounded much faster than the two-point formula which just has h in the denominator.