**Problem 1**

1. Analytical Solution

f(x) = sin(x^(1/2))

f'(x) = 0.5 \* x^(-1/2) \* cos(x^(1/2))

f''(x) = -1/(4x) \* sin(x^(1/2)) - 1/(4x^(3/2)) \* cos(x^(1/2))

1. M.files

testfuncion

function f = testfunction(x)

%f = sin(x.^(1/2));

f=sin(sqrt(x));

end

newfunction

function f = newfunction(x)

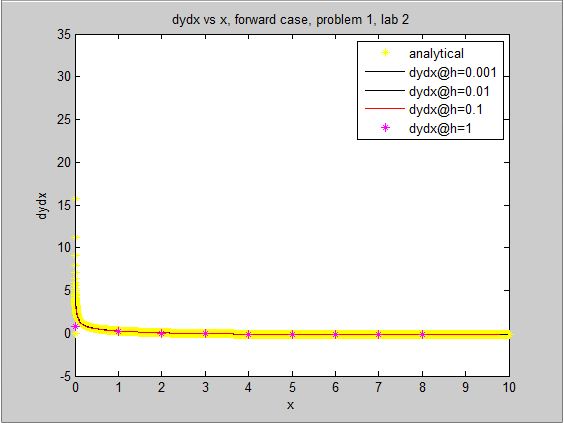
f = x^2 - x + 1;

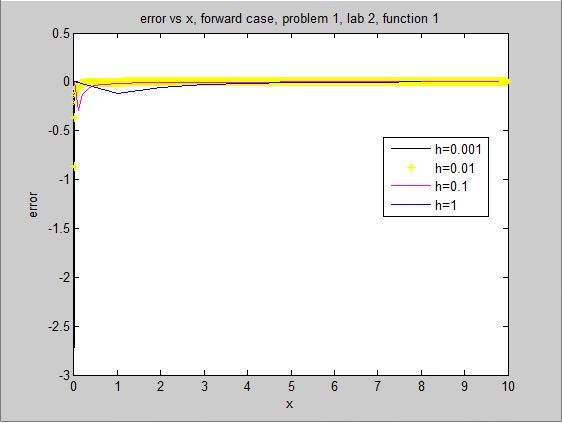
end

fwd1

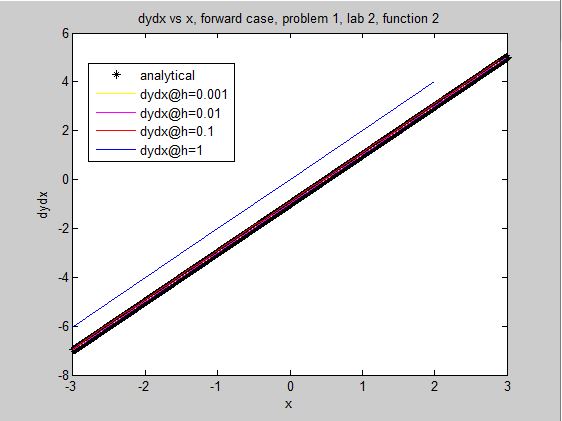
1. Plots

**FORWARD CASE**

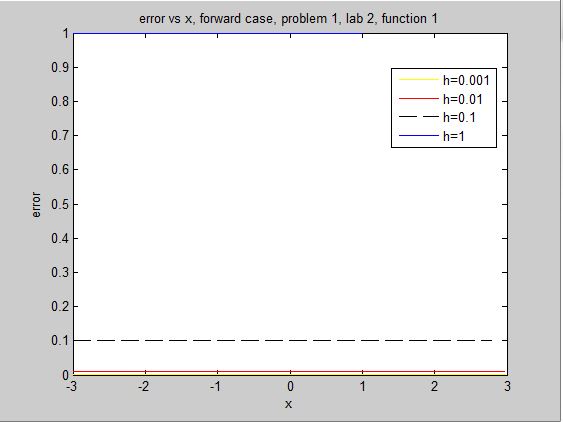




As can be seen above, the errors all converge to zero making the numerical approximation an accurate representation for the function f’(x).



Function 2 - g(x), below. Error in title.



The error for the second derivative of g(x) also becomes zero at very small values of h, making it a good approximation to the analytical solution.

**m.file for forward case**

clear all; clc;

% h = 1; 0:1; 0:01; 0:001;

h=1; % time steps

x = -3:h:3; % for newfunction

%x = 0:h:10; % for testfunction - vector of x points

for i = 1:length(x) % from 1 to 1000, for loop puts each individual value into function

y(i) = newfunction(x(i)); % for new function

dy(i) = 2\*x(i) - 1; % derivative of newfunction

% y(i)= testfunction(x(i)); % length y = 1000

% dy(i) = 0.5\*(x(i)^(-0.5))\*cos(sqrt(x(i))); % analytical derivative for plotting error

end

%y = testfunction(x); % i think this works fine too

for i = 1:(length(x)-1) % for forward case --> i = 1:(length(x)-2)for testfunction

dydx(i)=(y(i+1)-y(i))/h; % derivative for the FORWARD case w/Dan

end

% for i=1:(length(x)-2)

for i = 1:(length(x)-2)

error(i) = dydx(i) - dy(i); % error(301) is messed up besause dy has 0 in denominator at error(301)

end

% plot(x,dy,'k\*'); % plot analytical derivative, USED THIS ONE

% hold on;

%xlim([0 4])

plot(x(1:(length(x)-2)),error); % error plot GOOD

%plot(x(2:length(x)-1),error) % plots the error for testfunction error only since newfunction derivative has a discontinuity at x=0

%plot(x(2:(length(x)-300)),error(2:(length(x)-300))); % for newfunction

hold on;

%plot(x(1:(length(x)-300)), dy(1:(length(x)-300))); % for newfunction

%hold on;

%plot(x(1:length(x)-1),y); % plots x vs y for testfunction

% hold on;

% plot(x(1:length(x)-1),dydx); % for forward case for testfunction, USED THIS ONE

% hold on;

% ylim([-10 10]); % sets the minimum and maximums for the y-axis

%plot(x(1:length(x)),y); % for newfunction

%plot(x(1:(length(x)+1)/2-1), dy(1:(length(x)+1)/2-1));%for newfunction -3<x<0 for x vs dy

% hold on; % above plot finds the x=0 point of all h timesteps

% plot(x(((length(x)+1)/2+1):601), dy(((length(x)+1)/2+1):601));%for newfunction 0<x<-3 for x vs dy

% these two plots show a very small error from -3<x<0 with h = 0.01

% plot(x,dy); % turns out I can just do this, fuck my life and the last wasted hour

% hold on;

%Plotting dydx vs x including analytical solution

% legend('analytical','dydx@h=0.001','dydx@h=0.01','dydx@h=0.1','dydx@h=1');

% xlabel('x');

% ylabel('dydx');

% title('dydx vs x, forward case, problem 1, lab 2, function 2');

%Plotting the error vs x

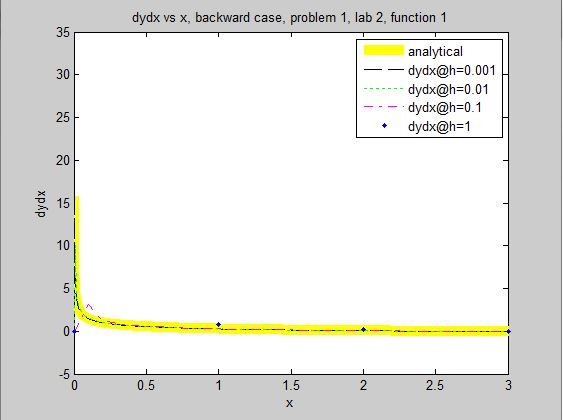
legend('h=0.001','h=0.01','h=0.1','h=1');

xlabel('x');

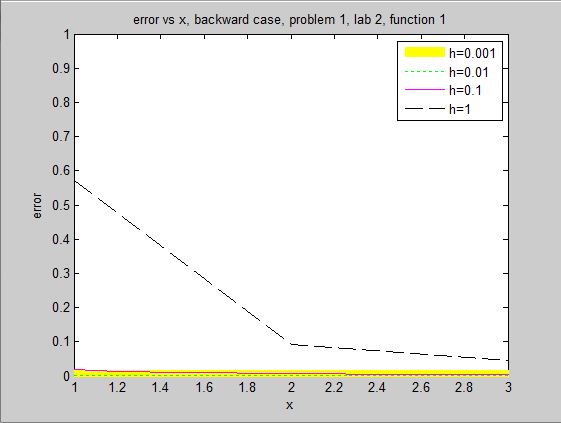
ylabel('error');

title('error vs x, forward case, problem 1, lab 2, function 1');

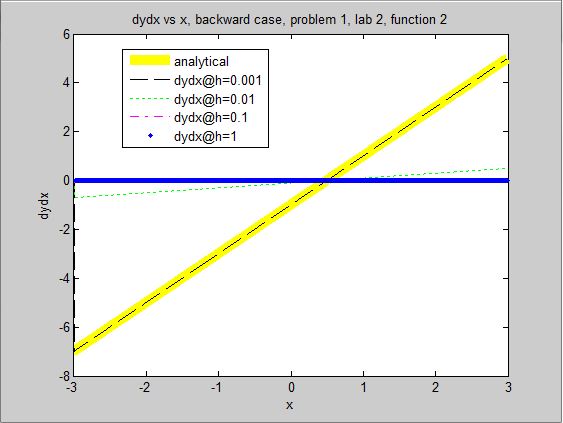
**BACKWARD CASE**

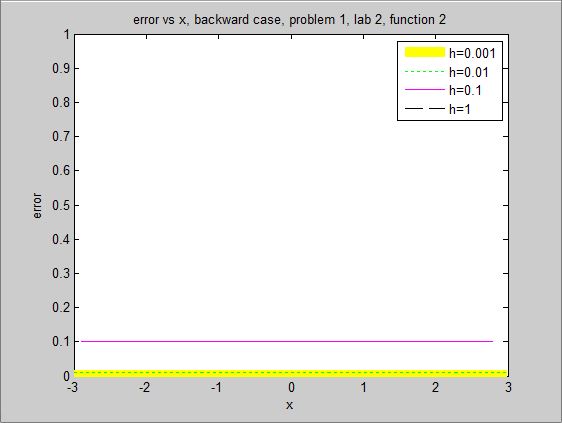
****

The truncation error remains as I could not figure out how to rid myself of it.



As can be seen above in the error plot for f(x) in the backward numerical solution, nearly all error converges to zero. Again, for f(x), the numerical solution is a good approximation of the solution.





Errors are constant and are near zero as h 🡪 0.

**m.files for backward case**

clear all; clc;

h = 1; % time steps

x = -3:0.001:3; % for newfunction

%x = 0:h:10; % length x = 1001 at time interval 0.01

counter = 0;

for i = 1:length(x)-1 % from 1 to 1000, for loop puts each individual value into function

y(i) = newfunction(x(i)); % for new function

%y(i)= testfunction(x(i)); % length y = 1000

%dy(i) = 0.5\*(x(i)^(-0.5))\*cos(sqrt(x(i))); % derivative fnc1

end

counter = 0;

for i = 2:(length(x)-1) % for backward case

counter = counter+1;

dy(i) = 2\*x(i) - 1; % derivative of newfunction

dydx(i)=(y(i)-y(i-1))/h; % derivative for the BACKWARD case --> works

end

counter = 0;

for i = 2:(length(x)-2)

counter = counter+1;

error(counter) = dydx(i) - dy(i);

end

%plot(x(1:length(x)-1),dy,'y','linewidth',8); % plot analytical derivative

%plot(x(1:length(x)-1),dydx,'k--') % h = 0.001

%plot(x(1:length(x)-1),dydx,'g:') % h = 0.01

%plot(x(1:length(x)-1),dydx,'m-.') % h = 0.1

%plot(x(1:length(x)-1),dydx,'b.') % h = 1

% xlim([0 3])

% hold on;

%Plotting dydx vs x including analytical solution

% legend('analytical','dydx@h=0.001','dydx@h=0.01','dydx@h=0.1','dydx@h=1');

% xlabel('x');

% ylabel('dydx');

% title('dydx vs x, backward case, problem 1, lab 2, function 2');

%plot(x(1:(length(x)-2)),error,'y','linewidth',8); % h = 0.001

%plot(x(1:(length(x)-2)),error,'g:'); % h = 0.01

%plot(x(1:(length(x)-2)),error,'m-'); % h = 0.1

plot(x(1:(length(x)-2)),error,'k--'); % h = 1

xlim([-3 3])

%ylim([0 1])

hold on;

%Plotting the error vs x

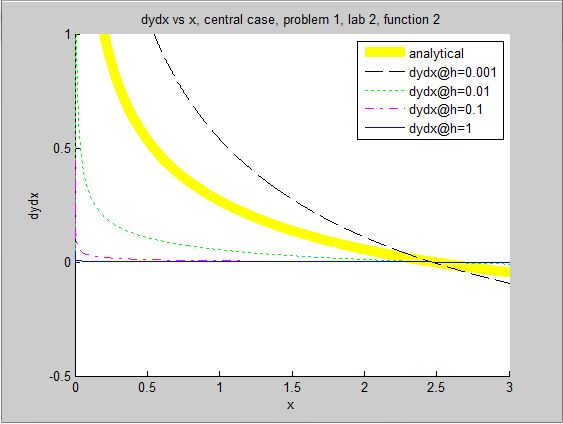
legend('h=0.001','h=0.01','h=0.1','h=1');

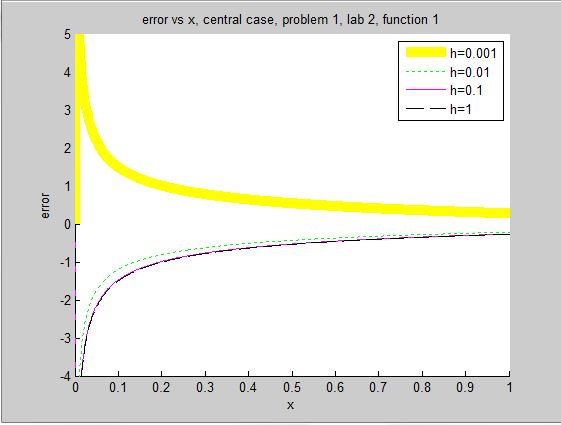
xlabel('x');

ylabel('error');

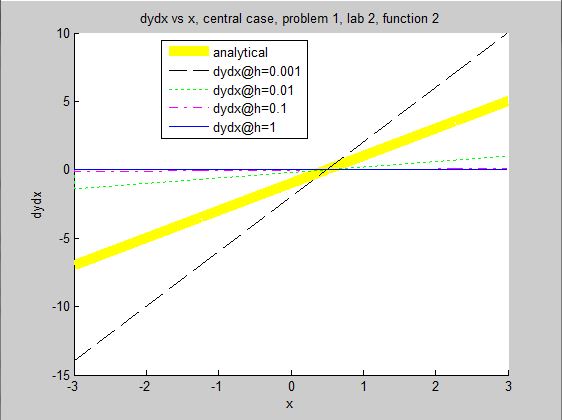
title('error vs x, backward case, problem 1, lab 2, function 2');

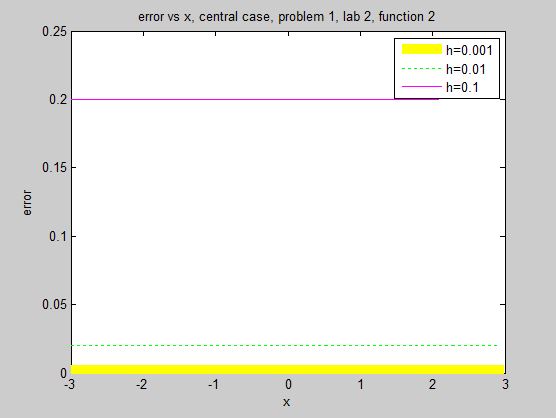
**CENTRAL CASE**





All error converges to zero for f(x) in the central 1 case. It follows that this is a decent approximation for all values of h.





As h 🡪 0 the error becomes smaller.

**m.file for the central case**

clear all; clc;

h=0.99; % time steps

x = -3:h:3; % for newfunction

% x = 0:0.001:10; % length x = 1001 at time interval 0.01

counter = 0;

for i = 2:(length(x)-2) % from 1 to 1000, for loop puts each individual value into function

counter = counter + 1;

y(counter) = newfunction(x(i)); % for new function

% y(i)= testfunction(x(i)); % length y = 1000

% dy(i) = 0.5.\*(x(i).^(-0.5))\*cos(sqrt(x(i))); % analytical derivative fnc1

end

counter = 0;

for i = 2:(length(x)-4) % for central case and 2nd derivative

counter = counter + 1;

dy(counter) = 2.\*x(i) - 1; % derivative of newfunction

dydx(counter)=(y(i+1)-y(i-1))./(2\*h); % derivative for the central case

end

counter = 0;

for i = 2:(length(x)-8)

counter = counter + 1;

error(counter) = dydx(i) - dy(i);

end

%plot(x(1:length(x)),y)

% plot(x(1:length(x)-5),dy,'y','linewidth',8)

% hold on;

% plot(x(1:length(x)-5),dydx)

%plot(x(1:length(x)),dy,'y','linewidth',8); % plot analytical derivative

%plot(x(1:length(x)-2),dydx,'k--') % h = 0.001

%plot(x(1:length(x)-2),dydx,'g:') % h = 0.01

%plot(x(1:length(x)-2),dydx,'m-.') % h = 0.1

%plot(x(1:length(x)-2),dydx,'b') % h = 1

%xlim([0 7])

%hold on;

%Plotting dydx vs x including analytical solution

% legend('analytical','dydx@h=0.001','dydx@h=0.01','dydx@h=0.1','dydx@h=1');

% xlabel('x');

% ylabel('dydx');

% title('dydx vs x, central case, problem 1, lab 2, function 2');

%plot(x(1:(length(x)-9)),abs(error),'y','linewidth',8); % h = 0.001

%plot(x(1:(length(x)-9)),abs(error),'g:'); % h = 0.01

%plot(x(1:(length(x)-9)),abs(error),'m-'); % h = 0.1

plot(x(1:(length(x)-9)),abs(error),'k--'); % h = 1

% xlim([-3 3])

% xlim([0 1])

% ylim([-4 5])

hold on;

%Plotting the error vs x

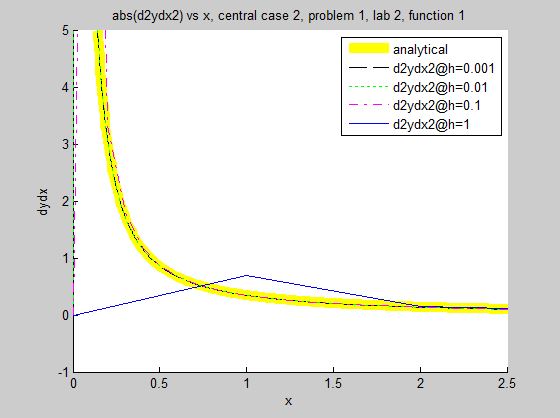
legend('h=0.001','h=0.01','h=0.1','h=1');

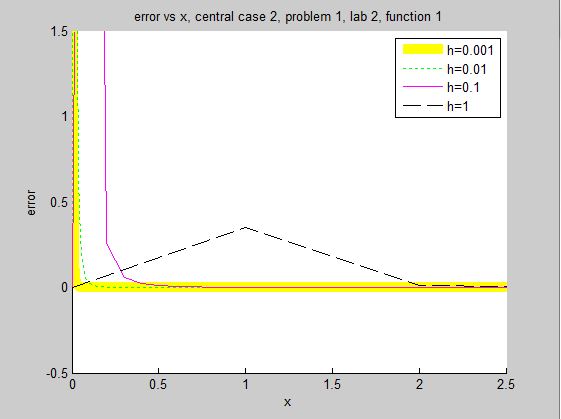
xlabel('x');

ylabel('error');

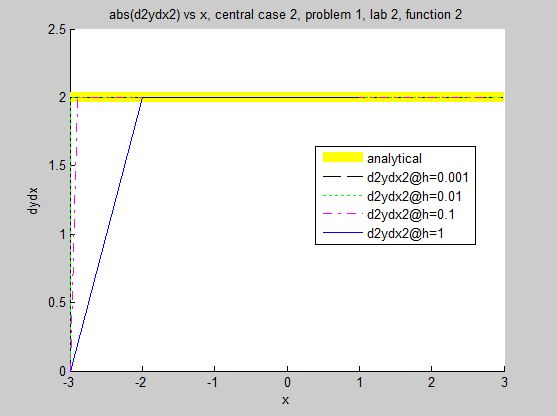
title('error vs x, central case, problem 1, lab 2, function 2');

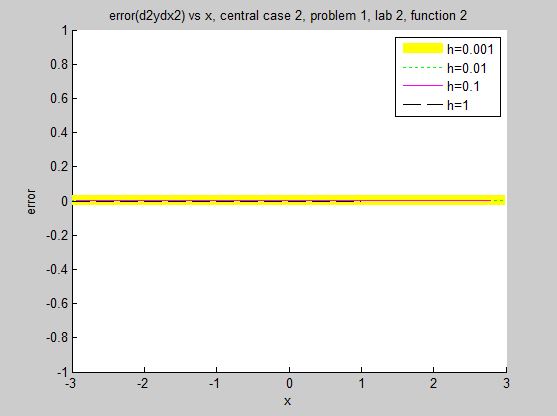
**CENTRAL 2ND DERIVATIVE**





In the central2 case, error converges to zero. There is truncation error before x=1, which is evident in the graph.





The error remains near zero for the 2nd derivative of g(x).