Numerical Computing II, Homework I

Questions Answered

II. Does the bisection method find the same zero as fzero does?

Yes, it does.

III. How does convergence depend on an initial guess?

The closer the guess, the faster the convergence -- usually.

IV. Does global Newton's method improve convergence significantly?

In my experience not really, unless your initial guess was horribly off.

V. Which of the algorithms was fastest / most robust?

This seems like a tie between GNM and NM to me.

Source Code

HW1.m

%%%%%%%%%%%%%

% Problem 1 %

%%%%%%%%%%%%%

% Plot the function exp(2sinx) - x

x = linspace(-10,10);

figure(1)

plot(x,exp(2\*sin(x))-x)

% Find a solution to exp(2sinx) - x = 0 using fzero

disp('The solution to the nonlinear equation using fzero is ')

func = @(x) exp(2\*sin(x))-x;

x0 = 2;

fzero(func,x0)

% Write a Matlab function which returns the function value and its

% derivative for a given point

% > The derivative of exp(2sinx)-x is 2cosx\*exp(2sinx)-1

% Find the matlab code for the function in h1p1ret.m

%%%%%%%%%%%%%

% Problem 2 %

%%%%%%%%%%%%%

% Write a Matlab function which implements the bisection method on exp(2sinx)-x

% with a tolerance of 10^-12.

% Find the matlab code for the function in h1p2ret.m

figure(2)

disp('Bisection method: ')

h1p2ret(-8,8,10^-12)

usrin = input('Enter the x value whose function value and derivative you want to know: ');

h1p1ret(usrin)

clearvars usrin;

hold on;

%%%%%%%%%%%%%

% Problem 3 %

%%%%%%%%%%%%%

% Implement Newton's method on exp(2sinx)-x. You will need the function

% and derivative values.

disp('Newton''s Method:')

usrin = input('Enter an initial guess: ')

h1p3ret(usrin,10^-12)

clearvars usrin;

% Try it out for multiple initial guesses.

% % for i=-6:6

% % h1p3ret(i,10^-12)

% % end

%%%%%%%%%%%%%

% Problem 4 %

%%%%%%%%%%%%%

% Implement global Newton's method on exp(2sinx)-x. You will need to

% provide an initial interval, and then mash the bisection and Newton

% algorithms together.

disp('Global Newton''s Method:')

h1p4ret(-8,8,10^-12)

hold off;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% h1p1ret.m %

% Write a Matlab function which returns the function value and its %

% derivative for a given point %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function v = h1p1ret(x)

format long;

% init vector v as storage

v = zeros(2,1);

% put function value in first component

v(1,1) = exp(2\*sin(x)) - x;

% put derivative value in second component

v(2,1) = 2\*cos(x)\*exp(2\*sin(x))-1;

% ////////////////////////////////////////////////////////////////////

% The original form of this function was to display the results to usr

% in a nice way. The next few lines contain that code, for my health.

% % gift for usr:

% % disp('Function value at selected point:')

% % disp(v(1,1))

% % disp('Derivative at selected point:')

% % disp(v(2,1))

% ////////////////////////////////////////////////////////////////////

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% h1p2ret.m %

% Write a Matlab function which implements the bisection method on %

% exp(2sinx)-x with a tolerance of 10^-12. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function h1p2ret(a,b,tol)

% Check to see if the function changes sign on the interval

if (((exp(2\*sin(a))-a)\*(exp(2\*sin(b))-b))>0)

% If it doesn't, throw an error

disp('Bad bracket')

return

else

% If it does, start iter and setup midpoint.

k=1;

xk = (a+b)/2;

while (abs((exp(2\*sin(xk))-xk)) > tol)

% If they have the same sign on the contracted interval,

if (((exp(2\*sin(a))-a)\*(exp(2\*sin(xk))-xk))>0)

% Move left bound to preset midpoint.

a = xk;

% If their signs are different on the contracted interval,

else

% Move right bound to preset midpoint.

b = xk;

end

% iter k each while step, save xk-x to vector for later

v(1,k) = xk - 2.635713222271392;

k = k+1;

% move bounds each while step

xk = (a+b)/2;

end

end

disp('Your numerical solution using the interval method is ')

disp(xk)

[m,n]=size(v);

axis = 1:n;

figure(2)

plot(axis,v);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% h1p3ret.m %

% Implement Newton's method on exp(2sinx)-x. You will need the function %

% and derivative values. Hopefully you wrote h1p1ret.m correctly. %

% (not like I did initially) %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function h1p3ret(x,tol)

format long;

% Display initial guess.

disp('Initial Guess for Newton''s method: ')

disp(x)

v = h1p1ret(x);

k = 1;

% Until the function value approaches tolerance,

while(abs(v(1,1))>tol)

% Get the function and derivative values,

v = h1p1ret(x);

% And perform Newton's method.

x = x-(v(1,1)/v(2,1));

% iter k each while step, save xk-x to vector for later

w(1,k) = x - 2.635713222271392;

k = k+1;

end

% Once we're done, celebrate by printing.

disp('Newton''s method solution: ')

disp(x)

k = num2str(k);

a = strcat('It took',{' '},k,' iterations.');

disp(a);

% Plot the errors

[m,n]=size(w);

axis = 1:n;

figure(2)

plot(axis,w);

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% h1p4ret.m %

% Implement global Newton's method on exp(2sinx)-x. You will need to %

% provide an initial interval, and then mash the bisection and Newton %

% algorithms together. %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function h1p4ret(a,b,tol)

format long;

% Display initial guess.

disp('Initial Guess for Global Newton''s method is left end of interval:')

disp(a)

v = h1p1ret(a);

k = 1;

% Check to see if the function changes sign on the interval

if (((exp(2\*sin(a))-a)\*(exp(2\*sin(b))-b))>0)

% If it doesn't, throw an error

disp('Bad bracket')

return

else

x = a;

% If it does, go ahead with the algorithm

% Until the function value approaches tolerance,

while(abs(v(1,1))>tol)

% Get the function and derivative values,

v = h1p1ret(x);

% And perform Newton's method.

x = x-(v(1,1)/v(2,1));

% iter k each while step, save xk-x to vector for later

w(1,k) = x - 2.635713222271392;

k = k+1;

% If x is in the interval,

if (x >= a) && (x <= b)

% leave it alone!

x;

% And if it ain't,

else

% Change it to the midpoint of the interval.

x = (a+b)/2;

end

% If they have the same sign on the contracted interval,

if (((exp(2\*sin(a))-a)\*(exp(2\*sin(x))-x))>0)

% Move left bound to preset midpoint.

a = x;

% If their signs are different on the contracted interval,

else

% Move right bound to preset midpoint.

b = x;

end

end

% Once we're done, celebrate by printing.

disp('Global Newton''s method solution: ')

disp(x)

k = num2str(k);

a = strcat('It took',{' '},k,' iterations.');

disp(a);

% Plot the errors

[m,n]=size(w);

axis = 1:n;

figure(2)

plot(axis,w);

end