**MA510 HW#**

**Mehmet Sinan INCI – 393784824**

**Q1. Part a)**

Using the fzero function of Matlab, I calculated the zero/root of the function as **1.671699881657161**

Also, I have tried the **bisection** **method** to get a good initial guess for Newton-Rhapson method as suggest. Searching between the range of 0 and 10, in only 10 iterations, I obtained the initial guess value as **1.669921875000000**. As you can see, the value calculated using Bisection is very close to the value calculated using the **fzero** function.

**Q1. Part b)**

When I use the ‘0’ as the initial guess,

1. The method does not converge. Estimated values jump between 4 values in the range of -3.000498253808222 to -1.961896665716206
2. I notice that there are actually 4 values that estimation has. After some research, I found out that this is due to the selected initial value and is called **Newton’s Fractal** where for certain functions, the initial value strongly determines next guesses and necessary number of iterations to get a good estimate. Even further, there are initial values of **non-convergence** where the root of the function cannot be accurately estimated even with large number of iterations.

The graphs are as follows;







**Q1. Part c)**

When I use ‘2’ as the initial guess, the method converges and the value within error boundaries is calculated in just 5 iterations as seen in the following figures;





The code for this question is as follows;

function [ out ] = fx( x )

% fx function given in the problem

out = x^3 - x -3;

end

function [ out ] = gx( x )

% gx is the derivative of the function fx

out = (3\*x^2) - 1;

end

% Mehmet Sinan INCI @ WPI

% Numerical Methods HW3 Question 1

% Finding the nontrivial root of

% f(x) = x^3 - 25

% using the Simple Fixed-Point Iteration

%% Bisection Method

% to calculate a good initial guess in few iterations

clear all;

clc;

% To pick a good initial guess, I will perform bisection for a few

% iterations

a = 0; % left border value

b = 10; % right border value

for i=1:10 % do 10 iterations, benefit of more iterations degrades after 10

fa = fx(a); % calculate the fx value for left border value

fb = fx(b); % calculate the fx value for right border value

mid = (a+b)/2;

fmid = fx(mid);

if fmid == 0 % stop if root is found

break;

end

if fa\*fmid < 0 % if true, the root is in range [a fmid]

b = mid;

else % the root is in range [fmid b]

a = mid;

end

end

x(1) = mid; % starting point - initial guess

fzero(@fx,2) % using Matlab's fzero tool to find root around '2'

%% Newton Rhapson Algorithm

clear all;

close all;

clc;

format long;

N = 1; % iteration counter

x(1) = 2; % starting point - initial guess

error(1) = 1; % randomly large relative approximate error

tolerance = 10^-6; % our error tolerance

for N=1:100

fN = fx(x(N)); % fN is f(Xn)

gN = gx(x(N)); % gN is f’(Xn)

x(N+1) = x(N) - (fN/gN);

error(N) = abs(x(N+1)-x(N) ) / abs(x(N+1)); % error calculation

if error(N) <= tolerance % break if tolerance goal is reached

break

end

N = N + 1

end

% N % number of iterations

plot(x);

title('Estimated root values over 5 iterations');

xlabel('Iteration') % x-axis label

ylabel('Estimated root value') % y-axis label

figure;

plot(error);

title('Error values over 5 iterations');

xlabel('Iteration') % x-axis label

ylabel('Error') % y-axis label

figure;

scatter(1:101,x)

title('Scatter plot of estimated root values over 5 iterations');

xlabel('Iteration') % x-axis label

ylabel('Estimated root value') % y-axis label

**Q2.**

Error graph for each method are as follows. Matlab code is included at the end.

**a)**



**b)**



**c)**



% Mehmet Sinan INCI @ WPI

% Numerical Methods HW3 Question 2

% Finding the nontrivial root of

% f(x) = cos(x^2) - x in [0,pi/2] to within TOL = 10-7

% with X0 starting point = 1.5

%% Part A

% Newton Rhapson Algorithm

clear all;

% close all;

clc;

format long;

N = 1; % iteration counter

x(1) = 1.5; % starting point - initial guess

TOL = 10^-7; % our error tolerance

for N=1:1000

fN = myfx(x(N)); % fN = f(Xn)

gN = mygx(x(N)); % gN = f'(Xn)

x(N+1) = x(N) - (fN/gN); % Next X estimation calculated

% using Newton Rhapson method

error(N) = abs(x(N+1)-x(N) ) / abs(x(N+1));

if error(N) <= TOL % stop when the error is within range

break

end

N = N + 1;

end

% N % number of iterations

% plot(x);

% title('Estimated root values over 6 iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Estimated root value') % y-axis label

%

% figure;

% plot(error);

% title('Error values over 6 iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Error') % y-axis label

figure;

esize = size(error,2);

temp1 = error(1:esize-1);

temp2 = error(2:esize);

plot(log(temp1),log(temp2));

title('Error Graph with Newtons Method');

%% Part B

% Secant method

clear all;

% close all;

clc;

format long;

N = 1; % iteration counter

x(1) = 1.5; % starting point - initial guess

x(2) = 1.4; % 2nd starting point - initial guess

TOL = 10^-7; % our error tolerance

for N=2:100

fN = myfx(x(N)); % f(Xn)

prevfN = myfx(x(N-1)); % f(Xn-1)

x(N+1) = x(N) - (fN/( (fN-prevfN)/(x(N)-x(N-1)) ) );

% next Xn calculated using the secant method in above line

error(N-1) = abs(x(N+1)-x(N) ) / abs(x(N+1));

if error(N-1) <= TOL % stop when the error is within range

break

end

N = N + 1

end

% N % number of iterations

% plot(x);

% title('Estimated root values over 7 iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Estimated root value') % y-axis label

%

% figure;

% plot(error);

% title('Error values over 7 iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Error') % y-axis label

figure;

esize = size(error,2);

temp1 = error(1:esize-1);

temp2 = error(2:esize);

plot(log(temp1),log(temp2));

title('Error Graph with Secant Method');

%% Part C

% Fixed point iteration method

% close all;

clear all;

clc;

x(1) = 1.5; % starting point - initial guess

TOL = 10^-7; % our error tolerance

for N=1:1000

x(N+1) = fpgx(x(N)); % our gx function, cos(x^2) that solves fx

error(N) = abs(x(N+1)-x(N) ) / abs(x(N+1));

if error(N) <= TOL % stop when the error is within range

break

end

N = N + 1;

end

% % N % number of iterations

% plot(x);

% title('Estimated root values over iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Estimated root value') % y-axis label

%

% figure;

% plot(error);

% title('Error values over iterations');

% xlabel('Iteration') % x-axis label

% ylabel('Error') % y-axis label

figure;

esize = size(error,2);

temp1 = error(1:esize-1);

temp2 = error(2:esize);

plot(log(temp1),log(temp2));

title('Error Graph with Fixed Point Method');

function [ out ] = fx( x )

% fx function given in the problem

out = x^3 - x -3;

end

function [ out ] = gx( x )

% gx is the derivative of the function fx used in Newton's method

out = (3\*x^2) - 1;

end

function [ out ] = myfx( x )

% f(x) function used for the secant method

% f(x) = cos(x^2) - x in [0,pi/2] to within TOL = 10-7

out = cos(x^2) - x;

end

function [ out ] = mygx( x )

% f'(x) function used for the secant method

% f'(x) = derivative of fx = cos(x^2) - x

out = -2\*x\*sin(x^2) - 1;

end

function [ out ] = fpgx( x )

% gx function used for fixed point method

out = cos(x^2);

end

**Q3. Part a)**