Program-1

Name: Divyashree HB

This program demonstrate the implementation of bisection and newton’s method for the function

* User provides the max iteration, error bound and the intervals for bisection & initial value for newton’s method.
* This report contains the source code, screenshot of the console and the output file.
* The following is the source code to calculate the given F(x) and G(x) using bisection and newton method.

#include<iostream>

#include<iomanip>//setpression

#include<vector>

#include<fstream>//write to file

#include <cmath> // std::abs

#include <math.h> /\* sin \*/

using namespace std;

std::ofstream out;

vector<long double> bisec\_arr;

vector<long double> newton\_arr;

long double f\_x(long double x) {

return ((x\*x\*x) - (5 \* x) + 1);

}

long double f1\_x(long double x) {

return ( (3\*x\*x) - 5);

}

long double g\_x(long double x) {

return ((3 \* sin(x)) - x);

}

long double g1\_x(long double x) {

return ((3 \* cos(x)) - 1);

}

long double bisection\_f(long double err, int NMAX, long double a, long double b) {

//INPUT: Function f, endpoint values a, b, tolerance TOL, maximum iterations NMAX

//CONDITIONS: a < b, either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0

//RETURN: value which differs from a root of f(x)=0 by less than TOL

long double c;

long double E = 0.0001;

long double TOL = err, fc; //initialize error to large value

int N = 1;

bisec\_arr.clear();

while (N <= NMAX && abs(E)>TOL) { // limit iterations to prevent infinite loop

setprecision(8);

c = (a + b) / 2;// new midpoint

bisec\_arr.push\_back(c);

if (f\_x(c) \* f\_x(a) <= 0) { // compare function values to determine interval side

E = c - b; //compute new error

b = c; // new interval

}

else

{

E = c - a; //compute new error

a = c; // new interval

}

N = N + 1; // increment step counter

}//endWhile

out << "The number of itrations take to run F(x) using bisection method is\t" << N-1 << endl;

return c;

}

//Let f : R → R be a differentiable function. The following algorithm computes

// an approximate solution x ∗ to the equation f(x) = 0.

long double newton\_f(long double err, int NMAX, long double xo) {

long double TOL = err, x = xo;

long double temp, E = 0.0001;

int N = 1;

newton\_arr.clear();

while (N <= NMAX && abs(E) > TOL) { // limit iterations to prevent infinite loop

temp = x - (f\_x(x) / f1\_x(x));//newton formula

//Compute new error | xk + 1 − xk |

newton\_arr.push\_back(temp);

E = (temp - x);

x = temp;

N = N + 1;

}

out << "The number of itrations take to run F(x) using newton's method is\t" << N -1<< endl;

return x;

}

long double bisection\_g(long double err, int NMAX, long double a, long double b) {

//INPUT: Function f, endpoint values a, b, tolerance TOL, maximum iterations NMAX

//CONDITIONS: a < b, either f(a) < 0 and f(b) > 0 or f(a) > 0 and f(b) < 0

//RETURN: value which differs from a root of f(x)=0 by less than TOL

long double c;

long double E = 0.0001;

long double TOL = err, fc; //initialize error to large value

int N = 1;

bisec\_arr.clear();

while (N <= NMAX && abs(E)>TOL) { // limit iterations to prevent infinite loop

setprecision(8);

c = (a + b) / 2;// new midpoint

bisec\_arr.push\_back(c);

if (g\_x(c) \* g\_x(a) <= 0) { // compare function values to determine interval side

E = c - b; //compute new error

b = c; // new interval

}

else

{

E = c - a; //compute new error

a = c; // new interval

}

N = N + 1; // increment step counter

}//endWhile

out << "The number of itrations take to run G(x) using bisection method is\t" << N-1 << endl;

return c;

}

//Let f : R → R be a differentiable function. The following algorithm computes

// an approximate solution x ∗ to the equation f(x) = 0.

long double newton\_g(long double err, int NMAX, long double xo) {

long double TOL = err, x = xo;

long double temp, E = 0.0001;

int N = 1;

newton\_arr.clear();

while (N <= NMAX && abs(E) > TOL) { // limit iterations to prevent infinite loop

temp = x - (g\_x(x) / g1\_x(x));//newton formula

//Compute new error | xk + 1 − xk |

E = (temp - x);

newton\_arr.push\_back(temp);

x = temp;

N = N + 1;

}

out << "The number of itrations take to run G(x) using newton's method is\t" << N-1 << endl;

return x;

}

int main() {

out.open("test.txt");

out << "----------------------------------------------project1--------------------------------------------------------" << endl;

out << "-------------------------------------------Divyashree H B-----------------------------------------------------" << endl;

out << "---------------------------------------------Feb 13,2017------------------------------------------------------" << endl;

cout << "----------------------------------------------project1--------------------------------------------------------" << endl;

cout << "-------------------------------------------Divyashree H B-----------------------------------------------------" << endl;

cout << "---------------------------------------------Feb 13,2017------------------------------------------------------" << endl;

int NMAX,i,j;//i->size for bisection array/vector and j for newton

long double err, a, b,guess;

cout << "Enter the following details" << endl;

cout << "The maximum number of iterations: ";

cin >> NMAX;

cout << "The maximum error tolarance value: ";

cin >> err;

out << "The following Method uses " << NMAX << " as its max iteration and " << err << "as its error precission" << endl;

cout << "To calculate f(x)=x^3-5x+1 using bisection method, enter the intervals" << endl;

out << "calculating f(x)=x^3-5x+1 using bisection method" << endl;

cin >> a >> b;

out << "For the interval [ " << a << " , " << b << " ]" << endl;

bisec\_arr.clear();

long double r = bisection\_f(err, NMAX, a, b);

cout << "result is " << r << endl;

out << "result = " << r << endl;

cout << "To calculate f(x)=x^3-5x+1 using newton's method, enter the initial guess" << endl;

out << "calculating f(x)=x^3-5x+1 using newton's method" << endl;

cout << "Enter the initial guess" << endl;

cin >> guess;

out << "With initial guess " << guess << endl;

newton\_arr.clear();

long double r1 = newton\_f(err, NMAX, guess);

cout << "result is " << r1 << endl;

out << "result = " << r1 << endl;

out << "iteration " << " Bisection\t\t" << "Newton" << endl;

i = 0; j = 0;

while (i < bisec\_arr.size())

{

while (j<newton\_arr.size()) {

out << i + 1 << "\t\t" << bisec\_arr[i] << "\t\t" << newton\_arr[j] << endl;

i++;

j++;

}

out << i + 1 << "\t\t" << bisec\_arr[i] << endl;

i++;

}

cout << "-----------------------------------------------------------------------------------------------" << endl;

cout << "To calculate f(x)=x^3-5x+1 using bisection method, enter the intervals" << endl;

out << "calculating f(x)=x^3-5x+1 using bisection method" << endl;

cin >> a >> b;

out << "For the interval [ " << a << " , " << b << " ]" << endl;

bisec\_arr.clear();

long double r3 = bisection\_f(err, NMAX, a, b);

cout << "result is " << r3 << endl;

out << "result = " << r3 << endl;

cout << "To calculate f(x)=x^3-5x+1 using newton's method, enter the initial guess" << endl;

out << "calculating f(x)=x^3-5x+1 using newton's method" << endl;

cout << "Enter the initial guess" << endl;

cin >> guess;

out << "With initial guess " << guess << endl;

newton\_arr.clear();

long double r2= newton\_f(err, NMAX, guess);

cout << "result is " << r2 << endl;

out << "result = " << r2 << endl;

out << "iteration " << " Bisection\t\t" << "Newton" << endl;

i = 0; j = 0;

while (i < bisec\_arr.size())

{

while (j<newton\_arr.size()) {

out << i + 1 << "\t\t" << bisec\_arr[i] << "\t\t" << newton\_arr[j] << endl;

i++;

j++;

}

out << i + 1 << "\t\t" << bisec\_arr[i] << endl;

i++;

}

cout << "----------------------------------------------------------------------------------------------" << endl;

cout << "To calculate g(x)=3sin(x)-x using bisection method, enter the intervals" << endl;

out << "calculating g(x)=3sin(x)-x using bisection method" << endl;

cin >> a >> b;

out << "For the interval [ " << a << " , " << b << " ]" << endl;

bisec\_arr.clear();

long double r4 = bisection\_g(err, NMAX, a, b);

cout << "result is " << r4 << endl;

out << "result = " << r4 << endl;

cout << "To calculate g(x)=3sin(x)-x using newton's method, enter the initial guess" << endl;

out << "calculating g(x)=3sin(x)-x using newton's method" << endl;

cout << "Enter the initial guess" << endl;

cin >> guess;

out << "With initial guess " << guess << endl;

newton\_arr.clear();

long double r5 = newton\_g(err, NMAX, guess);

cout << "result is " << r5 << endl;

out << "result = " << r5 << endl;

out << "iteration " << " Bisection\t\t" << "Newton" << endl;

i = 0; j = 0;

while (i < bisec\_arr.size())

{

while (j<newton\_arr.size()) {

out << i + 1 << "\t\t" << bisec\_arr[i] << "\t\t" << newton\_arr[j] << endl;

i++;

j++;

}

out << i + 1 << "\t\t" << bisec\_arr[i] << endl;

i++;

}

cout << "-----------------------------------------------------------------------------------------------" << endl;

cout << "To calculate g(x)=3sin(x)-x using bisection method, enter the intervals" << endl;

out << "calculating g(x)=3sin(x)-x using bisection method" << endl;

cin >> a >> b;

out << "For the interval [ " << a << " , " << b << " ]" << endl;

bisec\_arr.clear();

long double r6 = bisection\_g(err, NMAX, a, b);

cout << "result is " << r6 << endl;

out << "result = " << r6 << endl;

cout << "To calculate g(x)=3sin(x)-x using newton's method, enter the initial guess" << endl;

out << "calculating g(x)=3sin(x)-x using newton's method" << endl;

cout << "Enter the initial guess" << endl;

cin >> guess;

out << "With initial guess " << guess << endl;

newton\_arr.clear();

long double r7 = newton\_g(err, NMAX, guess);

cout << "result is " << r7 << endl;

out << "result = " << r7 << endl;

out << "iteration " << " Bisection\t\t" << "Newton" << endl;

i = 0; j = 0;

while (i < bisec\_arr.size())

{

while (j<newton\_arr.size()) {

out << i+1 << "\t\t" << bisec\_arr[i] << "\t\t" << newton\_arr[j] << endl;

i++;

j++;

}

out << i+1 << "\t\t" << bisec\_arr[i] << endl;

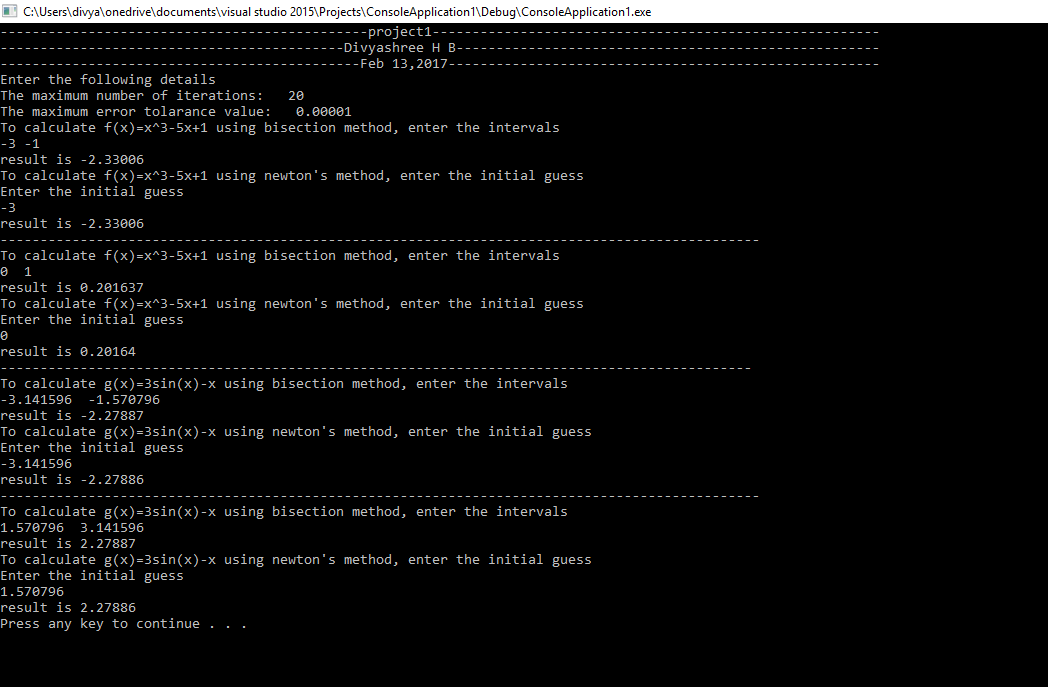
i++;

}

out.flush();

out.close();

return 0;

}