**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по практическому заданию №2**

**по дисциплине «Вычислительная математика»**

Тема: **РЕШЕНИЕ НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**МЕТОДОМ БИСЕКЦИИ**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Лисс А.Р. |

Санкт-Петербург

2017

**Цель работы.**

Исследовать обусловленность задачи нахождения корня уравнения для нелинейной функции.

**Основные теоретические положения.**

Метод бисекции – частный случай метода вложенных отрезков для вычисления корней нелинейных непрерывных уравнений. Роль вложенных отрезков здесь играет результаты деления исходного отрезка пополам; на каждом шаге итерации данного метода выбирается та половина отрезка, на концах которой функция имеет различные знаки – поскольку корень будет находиться именно в этом промежутке. Если на конце какого-либо отрезка значение функции равно нулю, то корень уравнения найден.

Задача нахождения корней нелинейных уравнений вида f(x)=0 (где f(x)- некоторая непрерывная функция) встречается в различных областях инженерной и научной деятельности. Нелинейные уравнения делятся на два класса - алгебраические и трансцендентные. Алгебраическими называются уравнения, содержащие только алгебраические функции (целые, рациональные, иррациональные). Уравнения, которые содержат другие функции (тригонометрические, показательные, логарифмические и т.п.), называются трансцендентными.

Методы решения нелинейных уравнений подразделяются на прямые и итерационные. Прямые методы позволяют записать корни в виде некоторого конечного соотношения. Итерационные методы, при которых алгоритм нахождения корня уравнения в общем случае включает два этапа:

- отыскания приближенного значения корня или содержащего его отрезка;

- уточнения приближенного значения до некоторой заданной степени точности.

Пусть задана непрерывная функция f вещественного аргумента x и требуется численным методом решить уравнение f(x)=0, т.е. найти приближение x\* к вещественному корню этого уравнения. Если уравнение имеет несколько вещественных корней, то сначала производят их отделение (изоляцию), а затем уточняют положение отдельного корня. Считается, что отделение корня произведено, если выделен такой интервал [a0, b0] области определения функции, на концах которого значения функции f(a0) и f(b0) имеют разные знаки и внутри которого имеется ровно один корень уравнения f(x)=0.

Если найден отрезок [a,b], такой, f(a)f(b)<0, существует точка c, в которой значение функции равно нулю, т.е f(с)=0, с∈(a,b). Метод бисекции состоит в построении последовательности вложенных друг в друга отрезков, на концах которых функция имеет разные знаки. Каждый последующий отрезок получается делением пополам предыдущего. Процесс построения последовательности отрезков позволяет найти нуль функции f(x)(корень уравнения f(x)=0 с любой заданной точностью.

Рассмотрим один шаг итерационного процесса. Пусть на (n-1)-м шаге найден от-резок [an-1, bn-1]⊂[a, b], такой, что f(an-1)f(bn-1)<0. Разделим его пополам точкой ξ=(an-1 +bn-1)/2 и вычислим f(ξ). Если f(ξ)=0, то ξ=( an-1+bn-1)/2- корень уравнения. Если f(ξ)≠0, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине, т.е.

an=an-1, bn=ξ , если f(ξ)f(an-1) < 0 ;

an=ξ, bn= bn-1 , если f(ξ)f(an-1) > 0 .

Если требуется найти корень с точностью ε, то деление пополам продолжается до тех пор, пока длина отрезка не станет меньше 2ε. Тогда координата середины отрезка есть значение корня с требуемой точностью ε.

Метод бисекции является простым и надежным методом поиска простого корня уравнения f(x)=0 (простым называется корень x=c дифференцируемой функции f(x), если f(с) и f `(с)≠0). Этот метод сходится для любых непрерывных функций f(x), в том числе недифференцируемых. Скорость его сходимости невысока. Для достижения точности ε необходимо совершить N≈log2(b-a)/ε итераций. Это означает, что для получения каждых трех верных десятичных знаков необходимо совершить около 10 итераций.

**Постановка задачи.**

Используя функции BISECT и Round найти корень нелинейного уравнения f(x)=ln(x)-1/(1+x2) методом бисекции с заданной точностью Eps, исследовать зависимость числа итераций от точности Eps при изменении Eps от 0.1 до 0.000001, исследовать обусловленность метода (чувствительность к ошибкам в исходных данных).

**Выполнение работы.**

1. Графически найдем отрезки [Left, Right], на которых функция f(x) удовлетворяет условиям теоремы Коши. График представлен на рис. 1.

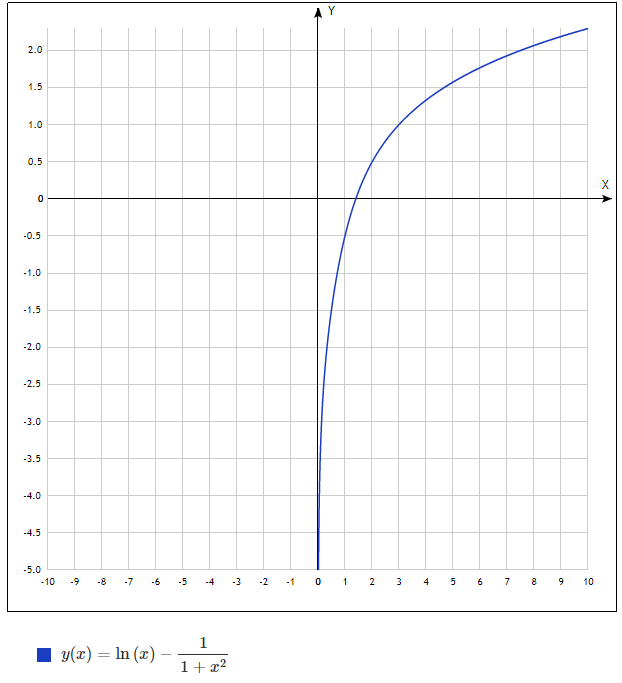


Рисунок 1 – График уравнения

Нужный отрезок ­­– [1.0; 2.0].

1. Составим подпрограмму вычисления функции f(x)=ln(x)-1/(1+x2). Код подпрограммы находится в приложении А.
2. Составим головную программу содержащую обращение к функциям f(x), BISECT, Round и индикации результатов. Реализовано считывание данных из файла data.txt и вывод результатов вычислений в файл result.txt. Код программы находится в приложении А.
3. Проведем вычисления по программе, варьируя значения параметра  
   Eps (точность вычисления корня, результаты в табл. 1) и Delta (точность задания исходных данных, результаты в табл. 2). Построим график зависимости количества итераций k от точности вычисления Eps, результаты приведены на рис. 2.

Таблица 1 – Вычисления при постоянном Delta и переменном Eps

|  |  |  |  |
| --- | --- | --- | --- |
| Eps | Delta | x | k |
| 0.1 | 0.000001 | 1.375 | 3 |
| 0.01 | 0.000001 | 1.39062 | 6 |
| 0.001 | 0.000001 | 1.40039 | 9 |
| 0.0001 | 0.000001 | 1.40125 | 13 |
| 0.00001 | 0.000001 | 1.40132 | 15 |
| 0.000001 | 0.000001 | 1.40132 | 15 |

Таблица 2 – Вычисления при постоянном Eps и переменном Delta

|  |  |  |  |
| --- | --- | --- | --- |
| Eps | Delta | x | k |
| 0.001 | 0.1 | 1.375 | 2 |
| 0.001 | 0.01 | 1.39844 | 6 |
| 0.001 | 0.001 | 1.40039 | 9 |
| 0.001 | 0.0001 | 1.40039 | 9 |
| 0.001 | 0.00001 | 1.40039 | 9 |
| 0.001 | 0.000001 | 1.40039 | 9 |

Рисунок 2 – Зависимость k от Eps

1. Исследуем чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных смоделируем с использованием функции Round, округляющей значения функции с заданной точностью Delta. Данные представлены на рис. 3.



Рисунок 3 – Содержимое файла вывода result.txt

**Выводы.**

В ходе выполненной работы было установлено, что количество итераций метода бисекции зависит от требуемой точности результата и погрешности входных данных. Чем больше требуемая точность, тем больше будет выполнено итераций. В случаях, когда требуемая точность меньше, чем погрешность входных данных, увеличивать точность нет прямой необходимости, поскольку при этом возрастает количество итераций. Однако, количество итераций не зависит от погрешности входных данных, если она меньше, чем требуемая точность.

Приложение А

Код программы

#include <fstream>

#include <math.h>

#include <conio.h>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

double Round (double X,double Delta);

double BISECT(double Left,double Right,double Eps,int &N);

double F(double x);

double delta;

int main()

{

ifstream data("data.txt");

ofstream result("result.txt");

setlocale(LC\_ALL, "Rus");

int k;

double a, b, eps, x, v;

while (data >> eps)

{

result <<"eps= "<< eps << "\t";

data >> a;

result << "a= " << a << "\t";

data >> b;

result << "b= " << b << "\t";

data >> delta;

result << "delta= " << delta << "\t";

x = BISECT(a, b, eps, k);

v = log(x)-1/(1+x\*x);

result << "v= " << 1/v << "\t";

if ((1 / v) <= (eps / delta))

{

result << "Задача хорошо обусловлена" << "\t";

}

else

{

result << "Задача плохо обусловлена" << "\t";

}

result <<"x= "<< x << "\t" <<"k= "<< k << endl;

}

data.close();

result.close();

return 0;

}

double F(double x)

{

extern double delta;

double s;

long int S;

s = log(x)-1/(1+x\*x);

if (s / delta < 0)

S = s / delta - .5;

else

S = s / delta + .5;

s = S\*delta;

s = Round(s, delta);

return(s);

}

double Round (double X,double Delta)

{

if (Delta<=1E-9) {puts("Неверное задание точности округления\n");exit(1);}

if (X>0.0) return (Delta\*(long((X/Delta)+0.5)));

else return (Delta\*(long((X/Delta)-0.5)));

}

double BISECT(double Left,double Right,double Eps,int &N)

{

double E = fabs(Eps)\*2.0;

double FLeft = F(Left);

double FRight = F(Right);

double X = (Left+Right)/2.0;

double Y;

if (FLeft\*FRight>0.0) {puts("Неверное задание интервала\n");exit(1);}

if (Eps<=0.0) {puts("Неверное задание точности\n");exit(1);}

N=0;

if (FLeft==0.0) return Left;

if (FRight==0.0) return Right;

while ((Right-Left)>=E)

{

X = 0.5\*(Right + Left); /\* вычисление середины отрезка \*/

Y = F(X);

if (Y == 0.0) return (X);

if (Y\*FLeft < 0.0)

Right=X;

else

{ Left=X; FLeft=Y; }

N++;

};

return X;

}