МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №3

по дисциплине «Вычислительная математика»

Тема: Метод бисекции

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2018

Цель работы.

Изучить метод бисекции для нахождения простого корня уравнения, обусловленность этого метода, зависимость числа итераций от точности задания исходных данных.

Основные теоретические положения.

Если найден отрезок [*a*,*b*], такой, что $f(a)f(b)<0$, существует точка *c*, в которой значение функции равно нулю, т.е. $f(c)=0$, $c\in(a,b)$. 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 отрезок $[a_{n-1},\ b_{n-1}]\subset[a,\ b]$, такой, что $f(a_{n-1})f(b_{n-1})<0$. Разделим его пополам точкой $\xi=(a_{n-1}+b_{n-1})/2$ и вычислим $f(\xi)$. Если $f(\xi)=0$, то $\xi=(a_{n-1}+b_{n-1})/2$ - корень уравнения. Если $f(\xi)\neq 0$, то из двух половин отрезка выбирается та, на концах которой функция имеет противоположные знаки, поскольку искомый корень лежит на этой половине, т.е.

$$a_n=a_{n-1},\ b_n=\xi\text{, если } f(\xi)f(a_{n-1})<0;$$

$$a_n=\xi,\ b_n=b_{n-1}\text{, если } f(\xi)f(a_{n-1})>0.$$

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.

Пусть между абсолютными погрешностями входных данных *X* и решения *У* установлено неравенство

$$\Delta(y^*)\le\nu_\Delta\Delta(x^*),\qquad(1)$$

где x^* и y^* - приближенные входные данные и приближенное решение. Тогда величина ν_Δ называется абсолютным числом обусловленности.

Если рассматривать задачу вычисления корня уравнения $Y = f(X)$, то роль числа обусловленности будет играть величина

$$\nu_\Delta = \frac{1}{\left| f'(x^0) \right|}, \tag{2}$$

где x^0 — корень уравнения.

Постановка задачи.

1) Графически или аналитически отделить корень уравнения $f(x) = 0$ (то есть найти отрезки [*Left, Right*], на которых функция $f(x)$ удовлетворяет условиям теоремы Коши).

2) Составить подпрограмму вычисления функции $f(x)$.

3) Составить головную программу, содержащую обращение к подпрограмме `F`, `BISECT`, `Round` и индикацию результатов.

4) Провести вычисления по программе. Построить график зависимости числа итераций от `Eps`.

5) Исследовать чувствительность метода к ошибкам в исходных данных. Ошибки в исходных данных моделировать с использованием программы `Round`, округляющей значения функции с заданной точностью `Delta`.

Выполнение работы.

Файлы программ methods.cpp и methods.h взяты из директории LIBR1. В текст программы methods.cpp добавлена подпрограмма вычисления функции $f(x)$. Программа main.cpp написана согласно требованиям, описанным в постановке задачи. Текст программы main.cpp представлен в приложении А, текст программы methods.cpp представлен в приложении Б, текст программы methods.h представлен в приложении В.

В задании, согласно варианту, использовалась функция

$$f(x) = 2x^2 - x^4 - 1 - \ln(x) \quad (3)$$

На рис. 1 представлен график функции (3). По нему можно найти границы отрезка, на котором расположен корень функции. Левая граница a равна 0.1, так как в 0 функция стремится к ∞. Правая граница b равна 2.

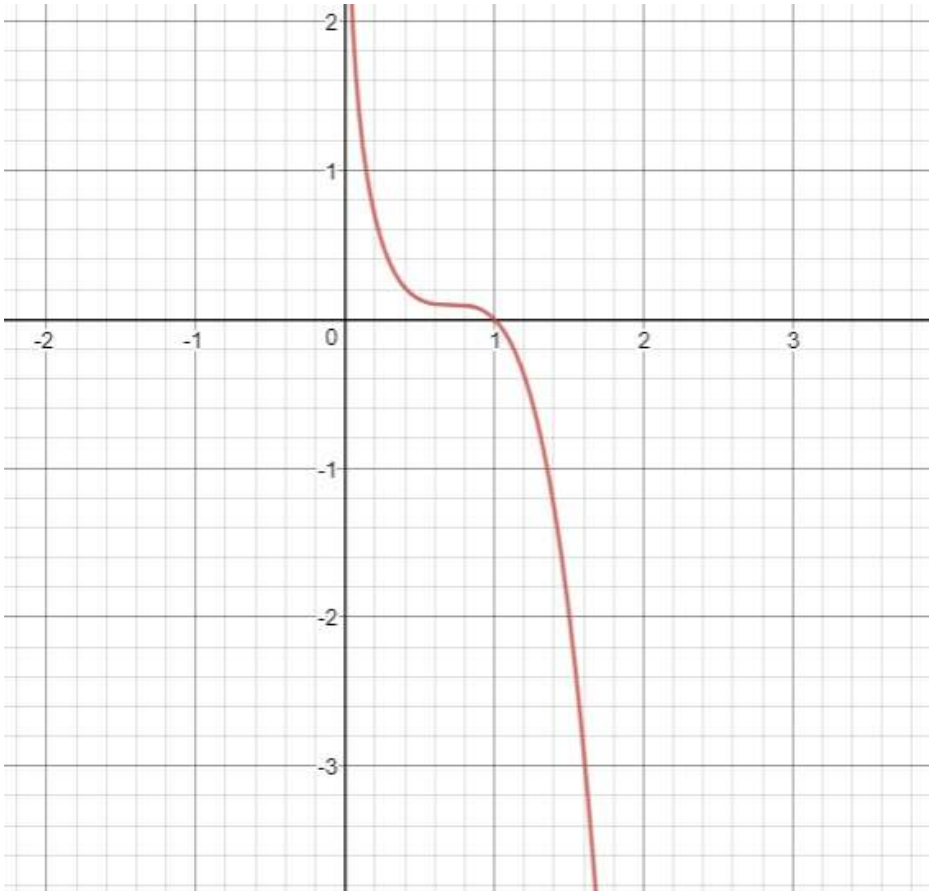

Рисунок 1 – График функции

В качестве теоретических данных выступает число обусловленности ν_Δ, рассчитываемое по формуле (2). Обусловленность задачи определяется выражением (1), в котором $\Delta(y^*)$ = `Eps`, а $\Delta(x^*)$ = `Delta`.

В таблице 1 представлены экспериментальные вычисления корня и числа итераций, теоретические вычисления абсолютного числа обусловленности и обусловленность задачи при меняющемся от 0.00001 до 0.1 значении `delta`, `Eps` = 0.01.

В таблице 2 представлены экспериментальные вычисления корня и числа итераций, теоретические вычисления абсолютного числа обусловленности и

обусловленность задачи при меняющемся от 0.00001 до 0.1 значении `Eps`, `delta` = 0.001.

Таблица 1 — Результаты вычислений при меняющемся `delta`

Значение `delta`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	1.007813	7	1	Хорошая
0.0001	1.007813	7	1	Хорошая
0.001	1.007813	7	1	Хорошая
0. 01	0.996875	5	1	Плохая
0.1	0.975000	2	1	Плохая

Таблица 2 — Результаты вычислений при меняющемся `Eps`

Значение `Eps`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	0.996875	5	1	Плохая
0.0001	0.996875	5	1	Плохая
0.001	0.996875	5	1	Плохая
0.01	0.996875	5	1	Хорошая
0.1	0.975000	3	1	Хорошая

На рис. 2 представлен график зависимости числа итераций k от точности вычисления корня `Eps`.

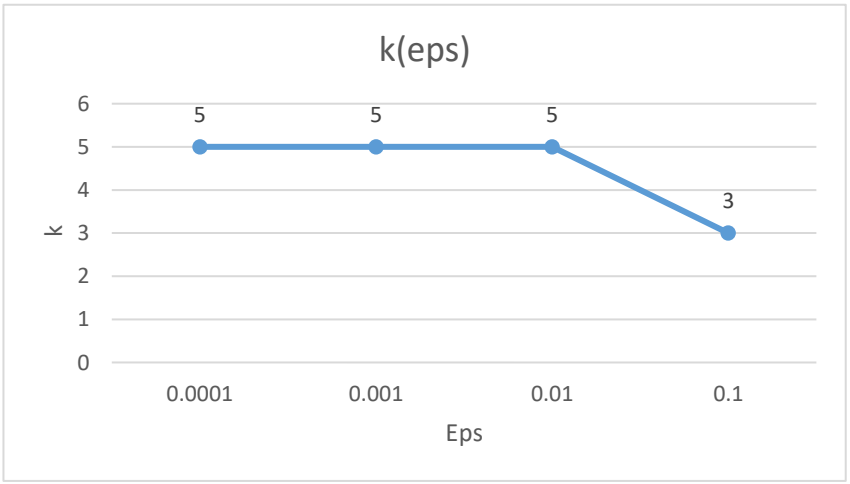

Рисунок 2 – График зависимости k от `Eps`

Выводы.

В работе изучен метод бисекции для нахождения простого корня уравнения, обусловленность этого метода, зависимость числа итераций от точности задания исходных данных. Задача хорошо обусловлена, если выполняется неравенство (1). Это подтверждается экспериментальными результатами (см. табл. 1 и табл. 2). При повышении точности вычисления корня Eps число итераций увеличивается (см. рис. 2).

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include<cmath>
#include<cstdoio>
#include<cstdlib>
#include<methods.h>
{
    int k;
    float eps1,delta1;
    double a,b,eps,x;
    printf("Введите eps:");
    scanf("%f",&eps1);
    eps = eps1;
    a = 0;
    b = 1.5;
    printf("Введите delta:");
    scanf("%f",&delta1);
    delta = delta1;
    x = BISECT(a,b,eps,k);
    printf("x=%f    k=%d\n",x,k);
    return 0;
}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЙ КОД ФУНКЦИЙ

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <iostream>
#include "METHODS.H"
using namespace std;
double Round (double X,double Delta)
{
 if (Delta<=1E-9) {puts("3 \n");exit(1);}
 if (X>0.0) return (Delta*(long((X/Delta)+0.5)));
    else    return (Delta*(long((X/Delta)-0.5)));
}

double F(double x)
{
extern double delta;
double s;
long int S;
s = 2*pow(x, 2)-pow(x, 4)-1-log(x);
if( s/delta < 0 )
S = s/delta - .5;
else
S = s/delta + .5;
s = S*delta;
s = Round( s,delta );
return(s);
}

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("2\n");exit(1);}
  if (Eps<=0.0) {puts("1\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);
     Y = F(X);
     if (Y == 0.0) return (X);
     if (Y*FLeft < 0.0)
        Right=X;
     else
      { Left=X; FLeft=Y; }
     N++;
  };
  return X;
}
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ЗАГОЛОВОЧНОГО ФАЙЛА

```
extern double F(double);
double Round (double X,double Delta);
double BISECT(double Left,double Right,double Eps, int &N);
```