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

ОТЧЕТ

по практической работе №4

по дисциплине «Вычислительная математика»

Тема: Метод хорд

Студент гр. 7383 _____ Кирсанов А.Я.

Преподаватель _____ Сучков А.И.

Санкт-Петербург

2018

Цель работы.

Изучить метод хорд для нахождения простого корня уравнения, обусловленность и скорость сходимости этого метода.

Основные теоретические положения.

Пусть найден отрезок [*a, b*], на котором функция $f(x)$ меняет знак. Для определенности положим $f(a) > 0$, $f(b) < 0$. В методе хорд процесс итераций состоит в том, что в качестве приближений к корню уравнения $f(x) = 0$ принимаются значения c_0, c_1, . . . точек пересечения хорды с осью абсцисс, как это показано на рис.1.

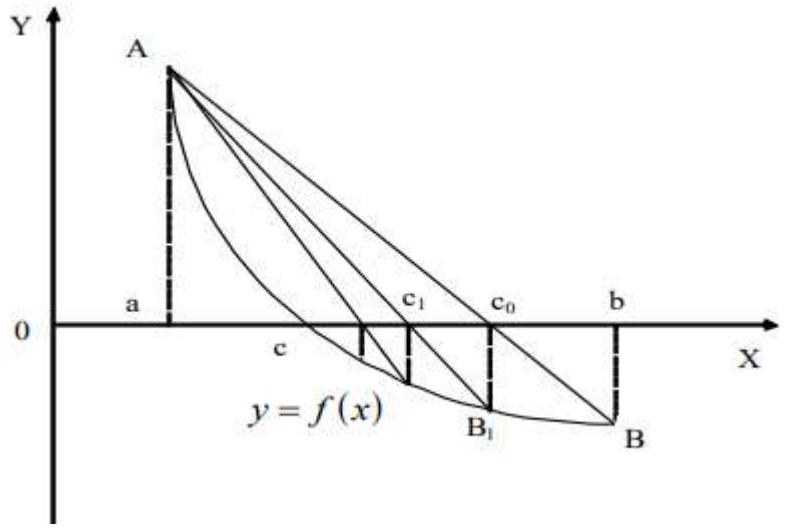

Рисунок 1 – Метод хорд

Сначала находится уравнение хорды AB

$$\frac{y - f(a)}{f(b) - f(a)} = \frac{x - a}{b - a}.$$

Для точки пересечения ее с осью абсцисс ($x = c_0$, $y = 0$) получается уравнение

$$c_0 = a - \frac{b - a}{f(b) - f(a)} f(a).$$

Далее сравниваются знаки величин $f(a)$ и $f(c_0)$ для рассматриваемого случая оказывается, что корень находится в интервале (a, c_0), так как

$f(a)f(c_0) < 0$. Отрезок [c_0, b] отбрасывается. Следующая итерация состоит в определении нового приближения c_1 как точки пересечения хорды AB_1 с осью абсцисс и т.д. Итерационный процесс продолжается до тех пор, пока значение $f(c_n)$ не станет по модулю меньше заданного числа ε.

Пусть между абсолютными погрешностями входных данных X и решения Y установлено неравенство

$$\Delta(y^*) \le \nu_\Delta \Delta(x^*), \tag{1}$$

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.

Метод хорд обладает скоростью сходимости α

$$\alpha = \frac{q}{1-q}, \tag{3}$$

где

$$q = \left| x^* - \frac{x^* - a}{f(x^*) - f(a)} f'(x^*) \right|, \tag{4}$$

при этом должно выполняться неравенство

$$\left| x_n - x^* \right| \le \alpha \left| x_n - x_{n-1} \right|, \tag{5}$$

Где x^* – корень уравнения, x_n и x_{n-1} – приближения корня.

Постановка задачи.

Порядок выполнения:

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

2) Составить подпрограмму – функцию вычисления функции $f(x)$, предусмотрев округление значений функции с заданной точностью `Delta` с использованием программы `Round`.

3) Составить головную программу, вычисляющую корень уравнения $f(x)=0$ и содержащую обращение к подпрограмме $f(x)$, `HORDA`, `Round` и индикацию результатов.

4) Провести вычисления по программе. Теоретически и экспериментально исследовать скорость сходимости и обусловленность метода.

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. Тексты программы main.cpp, methods.cpp methods.h и представлены в приложениях А, Б и В соответственно.

В задании, согласно варианту, использовалась функция

$$f(x)=2x^2-x^4-1-\ln(x) \qquad (6)$$

На рис. 2 представлен график функции (6). По нему можно найти границы отрезка, на котором расположен корень функции. Левая граница *a* равна 0.3, так как в 0 функция стремится к ∞. Правая граница *b* равна 1.2.

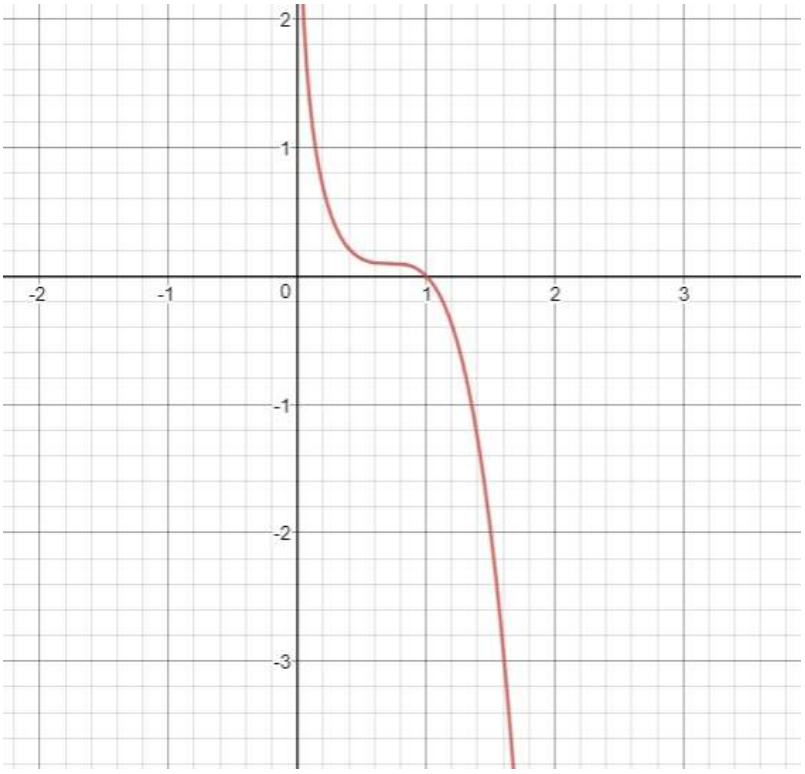

Рисунок 2 – График функции (6)

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

В табл. 2 представлены экспериментальные значения корня и числа итераций, теоретические значения 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	0.990314	15	1	Хорошая
0.0001	0.990325	15	1	Хорошая

Окончание таблицы 1

0.001	0.990250	15	1	Хорошая
0. 01	0.996131	16	1	Хорошая
0.1	0.968257	9	1	Плохая

Таблица 2 – Результаты вычислений при меняющемся `Eps`

Значение `Eps`	Значение x	Значение k	Значение ν_Δ	Обусловленность
0.00001	0.999566	22	1	Плохая
0.0001	0.999566	22	1	Плохая
0.001	0.999566	22	1	Хорошая
0.01	0.990250	15	1	Хорошая
0.1	0.621218	2	1	Плохая

На рис. 3 представлен график зависимости числа итераций k от точности вычисления корня `Eps`.

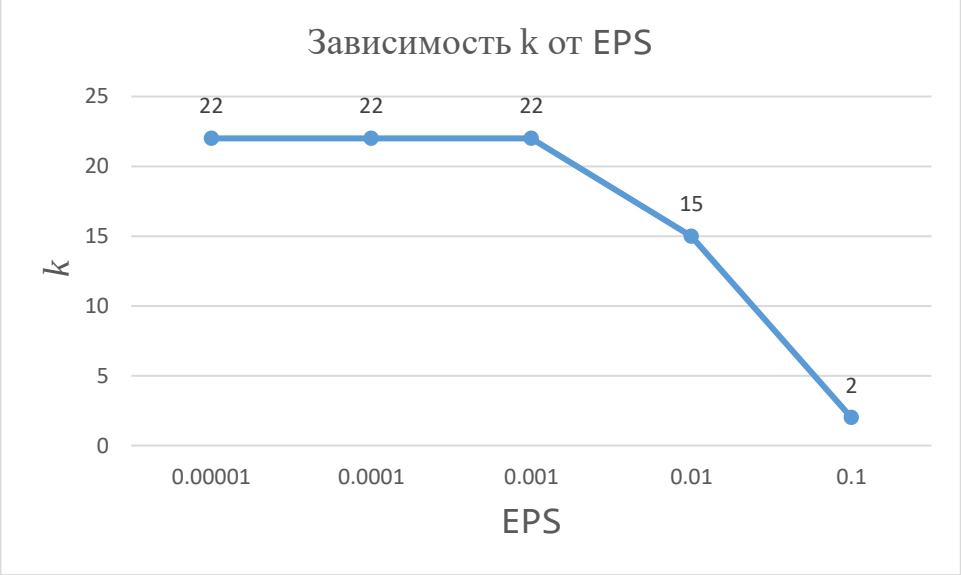

Рисунок 3 – График зависимости k от `Eps`

Вычислена скорость сходимости. Для этого по формуле (4) вычислено значение $q \approx 0.8421$, которое подставлено в формулу (3). Получено значение $\alpha = 5.329$ – скорость сходимости метода хорд. В табл. 3 представлены значения левых и правых частей неравенства (5). По ней можно убедиться, что на каждой

итерации неравенство выполняется, это значит, что скорость сходимости найдена верно. Измерения проводились при `Eps` = 0.0001, `delta` = 0.0001.

Таблица 3 – Результаты вычислений неравенства

Номер итерации	Значение корня на данной итерации	Значение левой части неравенства (5)	Значение правой части неравенства (5)
1	0.75	0.25	3.99675
2	0.841773	0.158227	0.489056
3	0.909288	0.0907118	0.359791
4	0.952203	0.0477972	0.228692
5	0.976197	0.0238034	0.127863
6	0.988518	0.0114824	0.0656586
7	0.99453	0.00546969	0.0320416
8	0.99744	0.00255981	0.0155067
9	0.998778	0.00122154	0.00713161
10	0.999419	0.000581222	0.00341228
11	0.999738	0.00026157	0.00170342
12	0.999898	0.000101872	0.000851032
13	0.999951	4.86537e-005	0.000283602

Выводы.

В работе изучен метод хорд 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). Скорость сходимости метода хорд постоянна при неизменяющемся отрезке поиска корня.

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 <cmath>
#include <stdlib.h>
#include <iostream>
#include <conio.h>
#include "methods.h"
using namespace std;
double n1, n, x1;

double Round (double X,double Delta)
{
 if (Delta<=1E-9) {puts("Invalid rounding precision\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 HORDA(double Left,double Right,double Eps,int &N)
{
  double FLeft = F(Left);
  double FRight = F(Right);
  double X,Y;

  if (FLeft*FRight>0.0) {puts("Invalid interval setting\n");exit(1);}
  if (Eps<=0.0) {puts("Invalid precision setting\n");exit(1);}

  N=0;
```

```
  if (FLeft==0.0)  return Left;
  if (FRight==0.0) return Right;
  X = 0;
  do
  {
    x1 = X;
     X = Left-(Right-Left)*FLeft/(FRight-FLeft);
speed(X, N);
     Y = F(X);
     if (Y == 0.0) return (X);
     if (Y*FLeft < 0.0)
      { Right=X; FRight=Y; }
     else
      { Left=X; FLeft=Y; }
     N++;

  }
  while ( fabs(Y) >= Eps );

  return(X);

}

void speed(double X, int &N){
    cout << N << ": " << X  << ", " << abs(X - 1) << " < " << 5.329 *
abs(X-x1) << endl;
}
```

ПРИЛОЖЕНИЕ В

ИСХОДНЫЙ КОД ЗАГОЛОВОЧНОГО ФАЙЛА

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