Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Томский государственный университет систем управления и радиоэлектроники»

Факультет систем управления (ФСУ)

Кафедра автоматизированных систем управления (АСУ)

**РЕШЕНИЕ УРАВНЕНИЙ С ОДНОЙ ПЕРЕМЕННОЙ**

**Отчет по лабораторной работе №1**

**По дисциплине**

**«Численные методы»**

Обучающийся гр. 431-3

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Андреев Д.П.

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Руководитель: ст. Преподаватель каф. АСУ

Косова А. Е.

« » 2023 г.

Томск 2023

Оглавление

[1 Цель и Задачи 4](#_Toc137074055)

[2 Теория 5](#_Toc137074056)

[3.Листинг программы 8](#_Toc137074057)

[4.Пример решения 13](#_Toc137074058)

[5.Вывод 17](#_Toc137074059)

# 1 Цель и Задачи

Для решения уравнения f(x)=0 необходимо реализовать три обязательных метода (дихотомии, хорд и Ньютона) и, по желанию, три дополнительных (комбинированный метод, метод итераций и золотого сечения).

# 2 Теория

**Итерационные методы**

Методы Ньютона (касательных) и итераций являются итеративными (итерационными), на основе некоторого приближения корня они позволяют на каждой итерации получать новое приближение . При этом используется информация о первой производной функции. Вместо условия (2.1.3) в итеративных методах оценивается расстояние между последним и предпоследним приближениями корня:

. (2.1.13)

При этом нужно знать начальное приближение x0, а дальнейшие приближения на каждой k+1-й итерации находятся по итеративной формуле:

. (2.1.14)

В методе Ньютона начальное приближение выбирается в соответствии со следующим условием: если в некоторой точке x произведение f (x) f "(x) > 0, то точка x является подходящей для начала итерационного процесса. Проверяются границы интервала:

(2.1.15)

На практике может наблюдаться ситуация, когда оба условия (2.1.15) не выполняются. В этом случае вместо второго условия можно использовать оператор «иначе», либо воспользоваться вторым критерием. Если вторая производная функции не известна, можно воспользоваться другим критерием. Вычислим точку c по формуле (2.1.9), и далее

(2.1.16)

Если начальная точка определена неправильно, то найденное решение уравнения (2.1.1) может находиться за пределами отрезка [a, b].

Функция φ() в (2.1.14) для метода Ньютона выглядит следующим образом:

(2.1.17)

В методе итераций, если выполняется неравенство |φ'(x)| < 1, процесс сходится независимо от выбора начальной точки. Поэтому можно брать любую из границ интервала, его середину и т.п. А функция φ() в (2.1.14) выглядит следующим образом:

(2.1.18)

В отличие от интервальных методов, длина исследуемого отрезка в которых на каждой итерации гарантированно уменьшается (например, для метода дихотомии – в два раза, для метода золотого сечения – в γ раз), в итеративных методах, в общем случае, расстояние между последовательными приближениями корня может иногда и увеличиваться. То же самое касается и значения функции в этих точках – оно может как уменьшаться, так и увеличиваться. Поэтому для некоторых функций условия (2.1.3) и (2.1.4) могут не выполняться в течение довольно большого числа итераций (или вообще никогда). В этом случае итерации следует прекращать при выполнении хотя бы одного условия.

**Комбинированный метод**

Комбинированный метод сочетает в себе сильные стороны методов хорд и Ньютона, и поэтому является достаточно эффективным для большого класса функций. Т.к. он является интервальным, то для него применимы выражения (2.1.5) и (2.1.8). Исключение интервалов выполняется по следующему алгоритму.

Сначала по формуле (2.1.9) ищется точка пересечения хорды с осью x. Далее, согласно (2.1.15), если f () f "() > 0 то точку можно переместить ближе к корню по формуле Ньютона (2.1.14) и (2.1.17). Тогда точка перемещается по формуле метода хорд (2.1.9):

(2.1.19)

Если же f () f "() > 0, то, наоборот, точку можно переместить ближе к корню по формуле Ньютона, а точку ak – по формуле метода хорд:

(2.1.20)

Два упомянутых условия достаточно проверять только один раз, если вторая производная не меняет своего знака на отрезке [a, b]. Но, т.к. это выполняется не для всех функций, лучше их проверять на каждой итерации. Аналогично (2.1.15), вместо второго условия можно использовать оператор «иначе», чтобы не возникла ситуация, когда оба условия не выполняются.

# 3.Листинг программы

using System;

using PolStrLib;

using System.IO;

namespace lab\_1

{

class Program

{

static int Sxod(double x,double a, double b)

{

if (a < x && x < b)

{

return 1;

}

else

{

return 0;

}

}

static int Degree(double e)

{

int count=0;

while (e != 1)

{

e = e \* 10;

count++;

}

return count;

}

static void Main(string[] args)

{

FileStream FileIn = new FileStream("IN.txt",FileMode.Open);

StreamReader reader = new StreamReader(FileIn);

int n = Convert.ToInt32(reader.ReadLine());

string pstr, expr = reader.ReadLine();

PolStr.StrToPolStr(expr, out pstr, 0);

double a = Convert.ToDouble(reader.ReadLine());

double b = Convert.ToDouble(reader.ReadLine());

double e = Convert.ToDouble(reader.ReadLine());

FileIn.Close();

reader.Close();

if (n == 1)//1:Метод Дихотомии

{

Console.WriteLine("Метод Дихотомии:");

if (PolStr.Error == Error.OK)

{

double c = 0.5 \* (a + b),ex=1;

while (Math.Abs((b - a)/2) >= e|| Math.Abs((PolStr.EvalPolStr(pstr, b, 0) - PolStr.EvalPolStr(pstr, a, 0)) / 2)>=e)

{

if (PolStr.EvalPolStr(pstr, a, 0) \* PolStr.EvalPolStr(pstr, c, 0) < 0)

{

b = c;

}

else

{

a = c;

}

c = 0.5 \* (a + b);

ex = Math.Abs((b - a) / 2);

}

c = Math.Round(c, Degree(e));

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Дихотомии");

writer.WriteLine("x= {0}", c);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, c, 0), Degree(e)));

writer.WriteLine("e= +-{0}", ex.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", c);

}

else

{

Console.WriteLine(PolStr.Error);

}

}

if (n == 2)//2:Метод Ньютона

{

Console.WriteLine("Метод Ньютона:");

if (PolStr.Error == Error.OK)

{

double x0=0, xk, xkp1,e1=1;

if(PolStr.EvalPolStr(pstr, a, 0)\* PolStr.EvalPolStr(pstr, a, 2) > 0)

{

x0 = a;

}

if (PolStr.EvalPolStr(pstr, b, 0) \* PolStr.EvalPolStr(pstr, b, 2) > 0)

{

x0 = b;

}

xk = x0 - (PolStr.EvalPolStr(pstr, x0, 0)) / (2 \* x0);

xkp1 = xk - (PolStr.EvalPolStr(pstr, xk, 0)) / (2 \* xk);

while (Math.Abs(xk - xkp1) >= e || Math.Abs(PolStr.EvalPolStr(pstr, xk, 0) - PolStr.EvalPolStr(pstr, xkp1, 0)) >=e)

{

xk = xkp1;

xkp1 = xk - (PolStr.EvalPolStr(pstr, xk, 0)) / (2 \* xk);

}

xkp1 = Math.Round(xkp1, Degree(e));

e1 = Math.Abs(xkp1 - x0);

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Ньютона");

writer.WriteLine("x= {0}", xkp1);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, xkp1, 0), Degree(e)));

writer.WriteLine("e= +-{0}", e1.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", xkp1);

}

else

{

Console.WriteLine(PolStr.Error);

}

}

if (n == 3)//3:Метод Хорд

{

Console.WriteLine("Метод Хорд");

if (PolStr.Error == Error.OK)

{

double xkm1 = a, xk = b, xkp1 = 0, e1 = 0,tmp;

do

{

tmp = xkp1;

xkp1 = xk - PolStr.EvalPolStr(pstr, xk, 0) \* (xkm1 - xk) /

(PolStr.EvalPolStr(pstr, xkm1, 0) - PolStr.EvalPolStr(pstr, xk, 0));

xkm1 = xk;

xk = tmp;

}while (Math.Abs(xkp1 - xk) > e|| Math.Abs(PolStr.EvalPolStr(pstr, xkp1, 0) - PolStr.EvalPolStr(pstr, xk, 0))>=e);

xk = Math.Round(xk, Degree(e));

e1 = Math.Abs(xk - xkm1);

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Хорд");

writer.WriteLine("x= {0}", xk);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, xk, 0), Degree(e)));

writer.WriteLine("e= +-{0}", e1.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", xk);

}

else

{

Console.WriteLine(PolStr.Error);

}

}

if (n == 4)//Метод Комбинированный

{

Console.WriteLine("Метод Комбинированный");

if (PolStr.Error == Error.OK)

{

double e1 = 0;

do

{

if (PolStr.EvalPolStr(pstr, a, 0) \* PolStr.EvalPolStr(pstr, a, 2) < 0)

{

a += (b - a) / (PolStr.EvalPolStr(pstr, a, 0) - PolStr.EvalPolStr(pstr, b, 0)) \* PolStr.EvalPolStr(pstr, a, 0);

}

else

{

a -= PolStr.EvalPolStr(pstr, a, 0) / PolStr.EvalPolStr(pstr, a, 1);

}

if (PolStr.EvalPolStr(pstr, b, 0) \* PolStr.EvalPolStr(pstr, b, 2) < 0)

{

b += (a - b) / (PolStr.EvalPolStr(pstr, b, 0) - PolStr.EvalPolStr(pstr, a, 0)) \* PolStr.EvalPolStr(pstr, b, 0);

}

else

{

b -= PolStr.EvalPolStr(pstr, b, 0) / PolStr.EvalPolStr(pstr, b, 1);

}

e1 = Math.Abs((b - a) / 2);

} while (Math.Abs(b - a) > 2 \* e || Math.Abs(PolStr.EvalPolStr(pstr, b, 0) - PolStr.EvalPolStr(pstr, a, 0) / 2) >= e);

double x = Math.Round((a + b) / 2, Degree(e));

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Комб");

writer.WriteLine("x= {0}", x);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, x, 0), Degree(e)));

writer.WriteLine("e= +-{0}", e1.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", x);

}

else

{

Console.WriteLine(PolStr.Error);

}

}

if (n == 5)//Метод Золотого Сечения

{

Console.WriteLine("Метод Золотого Сечения");

if (PolStr.Error == Error.OK)

{

double x, e1;

double Y = (Math.Sqrt(5) + 1) / 2;

do

{

double d = a + (b - a) / Y;

double c = a + (b - a) / Math.Pow(Y, 2);

if ((PolStr.EvalPolStr(pstr, a, 0) \* PolStr.EvalPolStr(pstr, d, 0)) > 0)

{

a = c;

}

else

{

b = d;

}

x = (a + b) / 2;

e1 = Math.Abs((b - a) / 2);

}

while (Math.Abs((b - a) / 2) >= e|| Math.Abs(PolStr.EvalPolStr(pstr, b, 0)- PolStr.EvalPolStr(pstr, a, 0)/2)>=e);

x = Math.Round((a + b) / 2, Degree(e));

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Золотого Сечения");

writer.WriteLine("x= {0}", x);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, x, 0), Degree(e)));

writer.WriteLine("e= +-{0}", e1.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", x);

}

else

{

Console.WriteLine(PolStr.Error);

}

}

if (n == 6)//Метод Итераций

{

Console.WriteLine("Метод Итераций");

if (PolStr.Error != Error.OK)

{

Console.WriteLine(PolStr.Error);

return;

}

double x = a + b / 2;

double x0;

do

{

x0 = x;

x = x - (PolStr.EvalPolStr(pstr, x0, 0) / Math.Max(PolStr.EvalPolStr(pstr, a, 1), PolStr.EvalPolStr(pstr, b, 1)));

}

while ((Math.Abs(x - x0) >= e) || Math.Abs(PolStr.EvalPolStr(pstr, x, 0)- PolStr.EvalPolStr(pstr, x0, 0))>=e);

x = Math.Round(x, Degree(e));

FileStream FileOut = new FileStream("OUT.txt", FileMode.Create);

StreamWriter writer = new StreamWriter(FileOut);

writer.WriteLine("Метод Золотого Сечения");

writer.WriteLine("x= {0}", x);

writer.WriteLine("f(x)= {0}", Math.Round(PolStr.EvalPolStr(pstr, x, 0), Degree(e)));

writer.WriteLine("e= +-{0}", e.ToString("e4"));

writer.Close();

FileOut.Close();

Console.WriteLine("{0}", x);

}

Console.ReadKey();

}

}

}

# 4.Пример решения

Входной файл IN.txt для всех методов на рисунке 4.1.

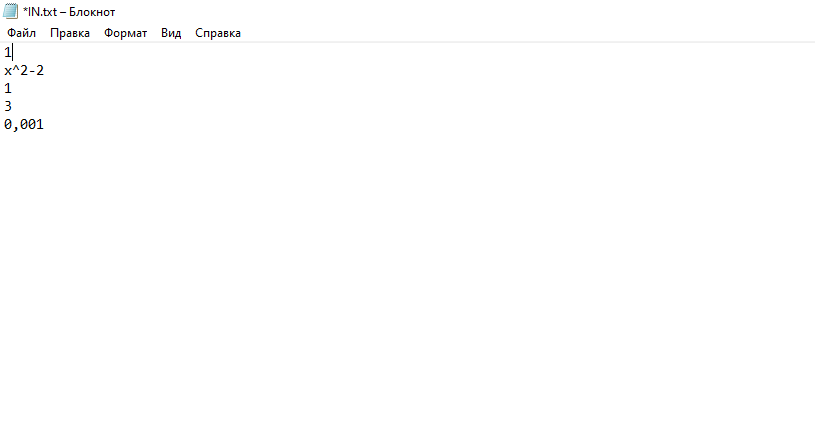


Рисунок 4.1— Входные данные из файла

Результат работы программы, записанный в файл OUT.txt при использовании метода Дихотомии на рисунке 4.2

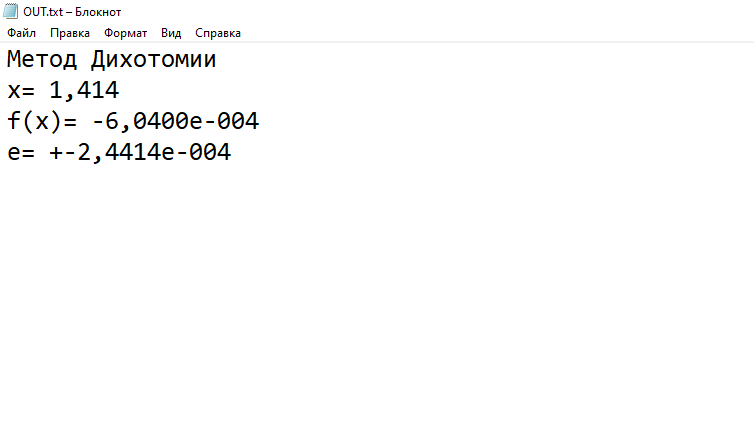


Рисунок 4.2 —Результат работы программы метода Дихотомии

Результат работы программы при использовании метода Ньютона на рисунке 4.3

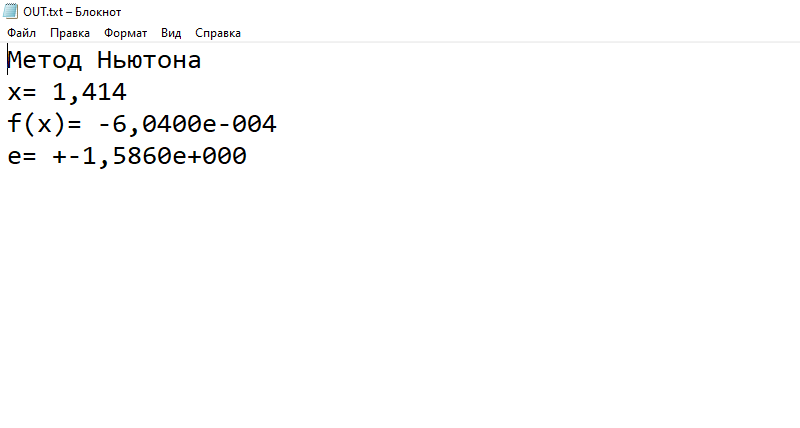


Рисунок 4.3 —Результат работы программы метода Ньютона

Результат работы программы при использовании метода Хорд на рисунке 4.4

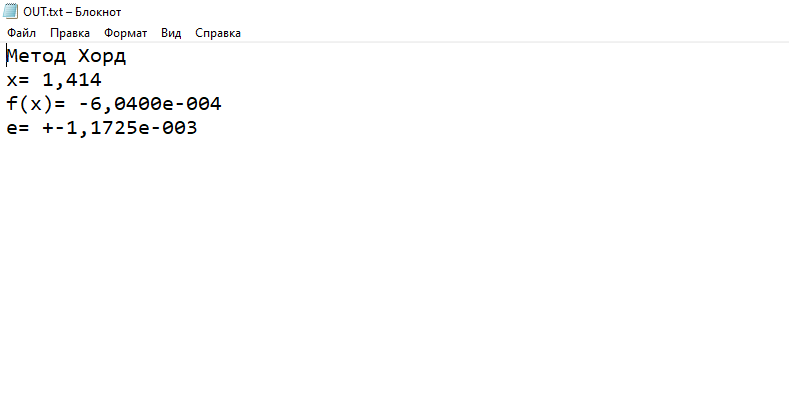


Рисунок 4.4 —Результат работы программы метода Хорд

Результат работы программы при использовании метода Комбинированный на рисунке 4.5



Рисунок 4.5 —Результат работы программы метода Комбинированный

Результат работы программы при использовании метода Золотого Сечения на рисунке 4.6

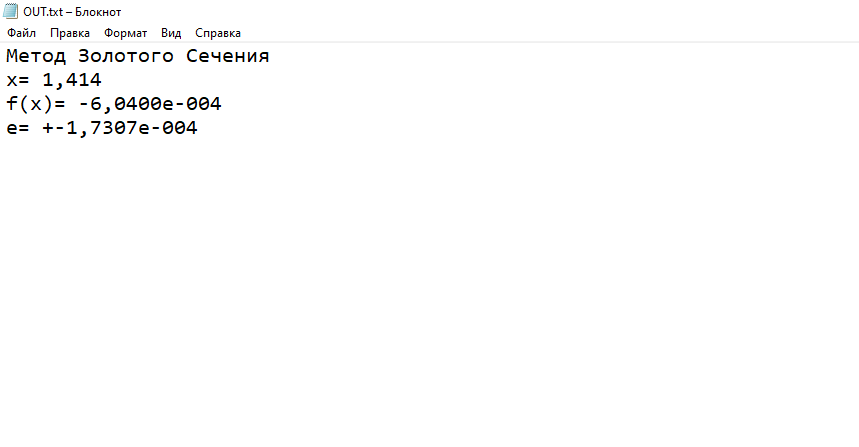


Рисунок 4.6 —Результат работы программы метода Золотого Сечения

Результат работы программы при использовании метода Итераций на рисунке 4.7

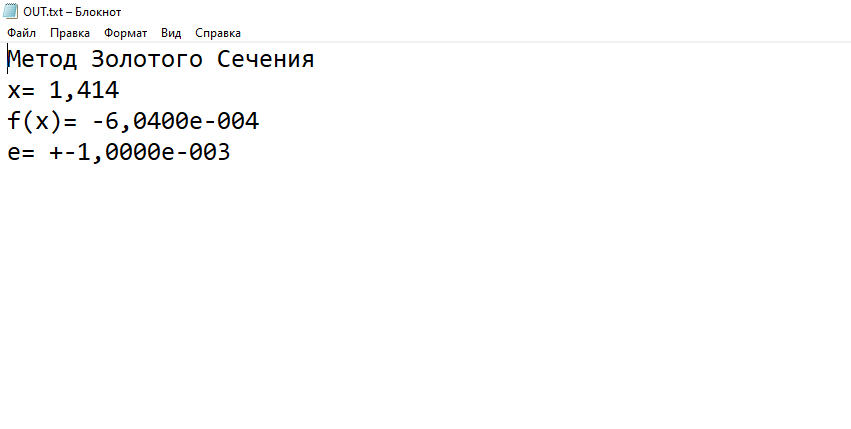


Рисунок 4.7 —Результат работы программы метода Итераций

# 5.Вывод

В результате лабораторной работы были изучены методы решения уравнений с одной переменной, а также библиотека PolStr