МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Московский Авиационный Институт»

(Национальный Исследовательский Университет)

Институт №8: «Информационные технологии и прикладная математика»

Кафедра 806: «Вычислительная математика и программирование»

**КУРСОВАЯ РАБОТА №4**

По курсу «Вычислительные системы»

I семестр

Тема:

«Процедуры и функции в качестве параметров»

**Группа:** М80-106Б-22

**Студент:** Ларченко А.О

**Преподаватели:** Дубинин А.В.

**Оценка: \_\_\_\_\_\_\_\_\_ Подпись: \_\_\_\_\_\_\_\_**

**Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Москва, 2023

# Содержание

[**Содержание**](#_expp2mfnf04p) **2**

[**Введение**](#_omob7rkxyj9e) **3**

[**Глава 1. Теоретическая часть**](#_8xbddmaxtdid) **4**

[Метод дихотомии(половинного деления)](#_pj6kfaf8l4n6) 4

[Метод итераций](#_8dt6tjl42twu) 5

[Метод Ньютона](#_hpxisfymi7sl) 6

[**Глава 2. Практическая реализация**](#_dx6halrk1e1r) **9**

[Задание](#_56rkx3b441wy) 9

[Алгоритм](#_ugkufkcwr5jv) 9

[Сценарий](#_wd7whw5n321) 9

[Аналитическое решение](#_fltqoxdhjit8) 10

[Описание переменных](#_vmhy8m97g351) 14

[Протокол](#_nhlrk6qyfuk6) 15

[**Заключение**](#_g9jpzbuebvqx) **23**

[**Список используемой литературы**](#_iegueqqpucgk) **24**

# Введение

Зачастую перед нами встаёт задача решить какое-либо уравнение и не всегда эта задача является простой. Но существуют методы нахождения приближенного корня таких уравнений(методы итераций, Ньютона и дихотоми), которые можно реализовать через программу.

Цель 4 курсового проекта - составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и дихотомии). Применить каждый метод к решению двух уравнений.

\*\*\*Если функция представляет собой многочлен, то уравнение называют алгебраическим, если же в функцию входят элементарные (тригонометрические, логарифмические, показательные и т. п.) функции, то такое уравнение называют трансцендентным\*\*\*

# Глава 1. Теоретическая часть

## Метод дихотомии(половинного деления)

Из курса матана мы знаем теорему Больцано-Коши о о нуле непрерывной функции, она звучит так: “Если функция непрерывна на некотором отрезке и на концах этого отрезка принимает значения противоположных знаков, то существует точка, в которой значение функции равно нулю (иными словами существует корень в данной точке)”. Именно на этой теореме основан метод дихотомии,т.е. деление отрезка на две части. Обобщенный алгоритм выглядит так:

1. Задать начальный интервал ;
2. Убедиться, что на концах функция имеет разный знак( );
3. Повторять
   * выбрать внутри интервала точку ;
   * сравнить знак функции в точке со знаком функции в одном из концов;
   * сместить концы отрезка так:

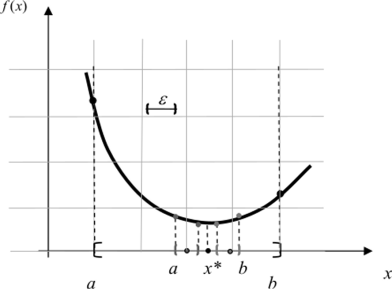
, если ;

, иначе

, если ;

, иначе;

Для того, чтобы найти приближённое значение корня с точностью до , необходимо остановить процесс половинного деления на таком шаге , на котором и вычислить , где - маленькое число, равное погрешности нашего вычисления.



метод дихотомии

## Метод итераций

Это способ численного решения математических задач. Его суть – нахождение алгоритма поиска по известному приближению (приближенному значению) искомой величины следующего, более точного приближения. Применяется в случае, когда последовательность приближений по указанному алгоритму сходится.

Идея метода заключается в замене исходного уравнения уравнением вида . Это можно сделать разными способами:

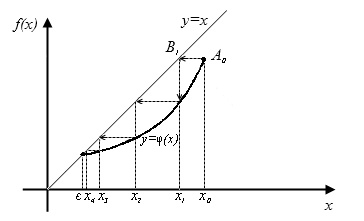
1. выделить x из исходного уравнения , остальное перенести в правую часть (это и будет f(х)).
2. Умножить левую и правую части на произвольную константу и прибавить к левой и правой частям x, т. е. представить в виде . При этом .

Я буду использовать 2 способ.

Достаточное условие сходимости метода: . Это условие необходимо проверить перед началом решения задачи, так как функция может быть выбрана неоднозначно, причем в случае неверного выбора указанной функции метод расходится.

Обобщенный алгоритм выглядит так:

1. Задать начальный интервал ;
2. Проверить условие сходимости;
3. Задать начальное приближение, равное середине отрезка ();
4. Итерационный процесс:
5. Выполнять процесс до условия ;
6. Приближённое значение корня - .

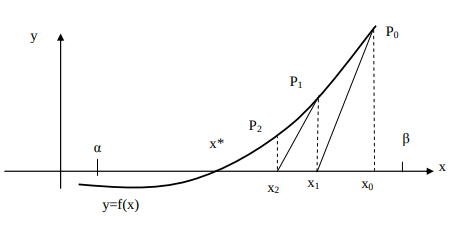


Метод итераций

## Метод Ньютона

Пусть уравнение f(x)=0 имеет один корень на отрезке [α, β], причем f '(х) и f "(х) определены, непрерывны и сохраняют постоянные знаки на отрезке [α, β].

Рассмотрение метода Ньютона начнем с его геометрического представления. Возьмем некоторую точку отрезка [α, β] и проведем в точке {, f()} графика функции касательную к кривой y=f(x) до пересечения с осью Ох. Абсциссу точки пересечения можно взять в качестве приближенного значения корня. Проведя касательную через новую точку {, f()} и находя точку ее пересечения с осью Ox, получим второе приближение корня . Аналогично определяются последующие приближения.



Метод Ньютона

Выведем формулу для последовательных приближений к корню. Уравнение касательной, проходящей через точку , имеет вид: )+(x-).

Подставляя y=0, находим абсциссу точки пересечения касательной с осью Ох: . Получаем зависимость: .

Условие сходимости можно проверить через уравнение: на [a,b].

Обобщенный алгоритм выглядит так:

1. Задать начальный интервал ;
2. Проверить условие сходимости;
3. Задать начальное приближение, равное середине отрезка ();
4. Итерационный процесс:;
5. Выполнять процесс до условия ;
6. Приближённое значение корня - .

# Глава 2. Практическая реализация

## Задание

Необходимо составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления – дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию.

Вариант 16-17



## Алгоритм

## Сценарий

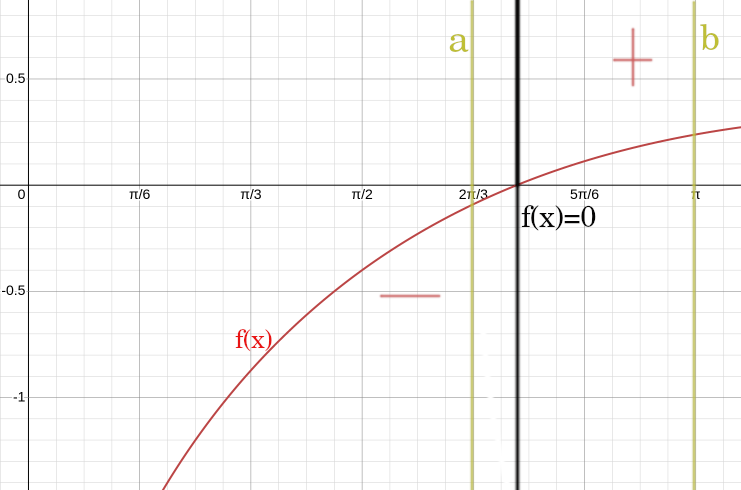
Первым делом проверим аналитически применимость методов для каждой из функций. Затем приступим к программной реализации: разобьем нашу программу на подзадачи и объявим глобальные переменные(x, count\_i, eps).

1. функция f\_1(x) возвращает значение функции 3\*sin(sqrt(x1))+0.35\*x1-3.8;
2. функция f\_2(x) возвращает значение функции 0.25\*x1\*x1\*x1+x1-1.2502;
3. функции f\_1\_iter(x) и f\_2\_iter(x) необходимы для метода итерации
4. функция derivative1() возвращает производную функции первого порядка;
5. функция derivative2() возвращает производную функции второго порядка;
6. функция dich() получает на вход функцию и границы, в которых она определена, и ищет её приближенное значение по методу дихотомии
7. функция iter() получает на вход функцию и границы, в которых она определена, и ищет её приближенное значение по методу итерации;
8. функция newton() получает на вход функцию и границы, в которых она определена, и ищет её приближенное значение по методу Ньютона;
9. главная функция main(), отвечающая за проверку применимости каждого метода к 2 функциям и за вывод результатов.

После этого сверим результаты работы программы с аналитическим предположением.

## Аналитическое решение

1)Функция , [2,3]

a)Для применения метода дихотомии нам необходимо проверить условие: (функция должна иметь разные знаки на концах отрезках). 

Исходя из графика, это условие выполняется, также при подставлении численных значений, получаем нужный результат (берём приближённые значения на отрезке, ):

f(2)= 2.963 +0.7 -3.8= -0.136<0;

f(3)=2.961+1.05-3.8=0.211>0;

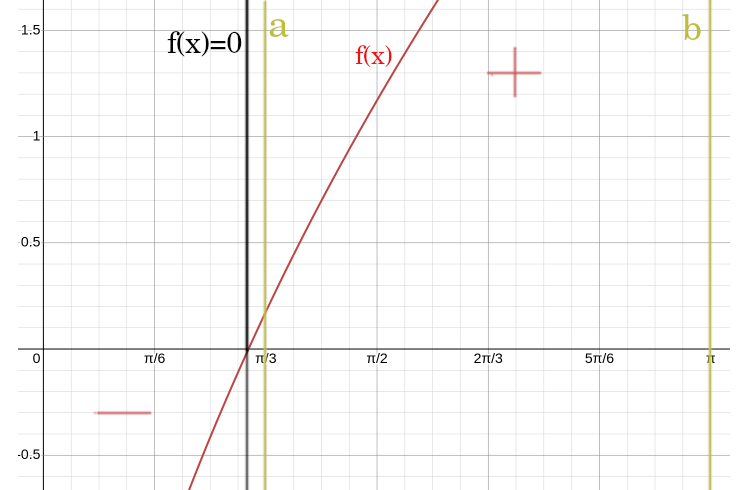
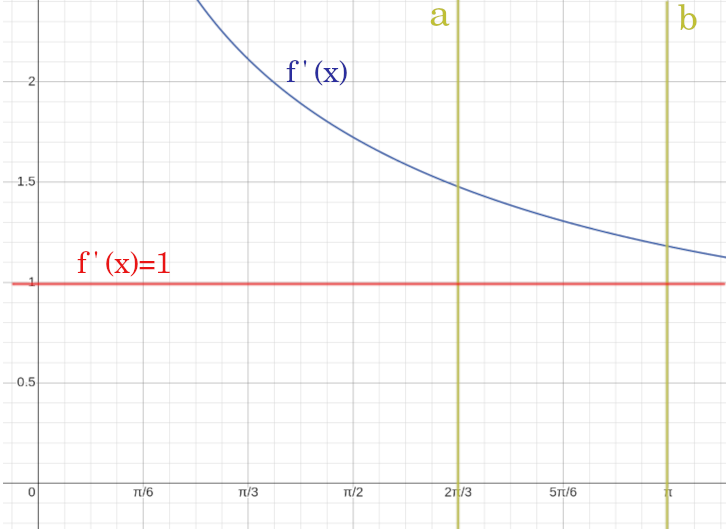
b)Для применения метода итерации нам необходимо, чтобы условие выполнялось на всём отрезке, при этом для метода итераций, мы берём функцию 

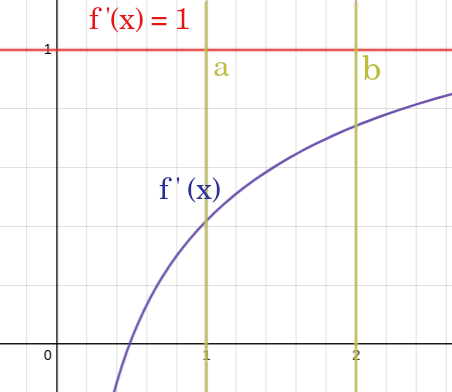
рис. 1

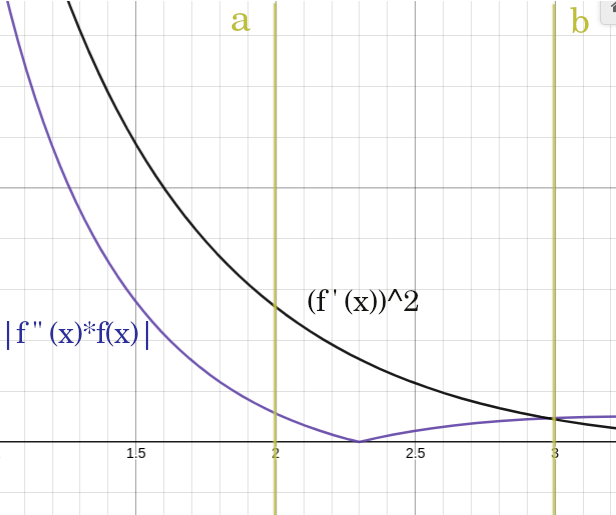
f(x)== , которая принимает вид рис.(1)

А её производная:

Видим, что производная больше 1 на всём отрезке, следовательно, метод неприменим.

Но по условию, нам сказано, что мы можем умножать функцию на произвольную константу , возьмём =-0.5, получим f(x)=-0.5= , тогда производная будет равна:

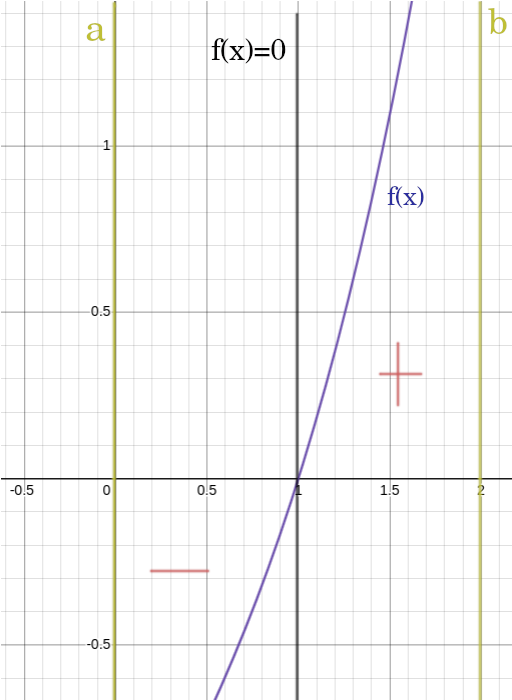
 Условие выполняется на всём промежутке, а значит метод применим.

с)Для применения метода Ньютона необходимо выполнения условия , где

f’’(x)=

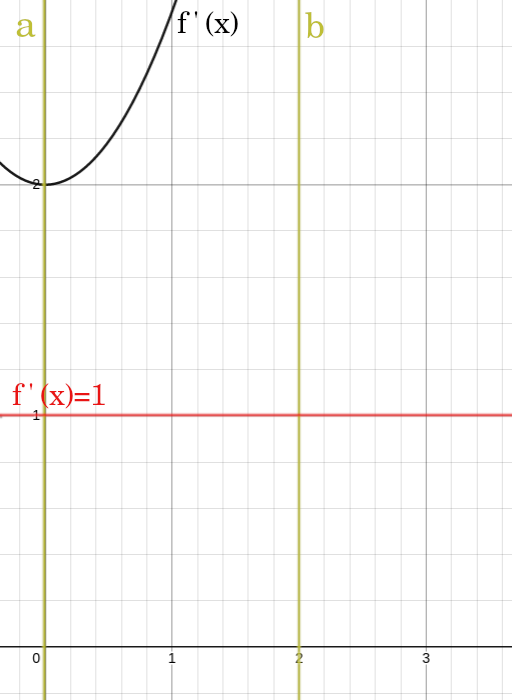
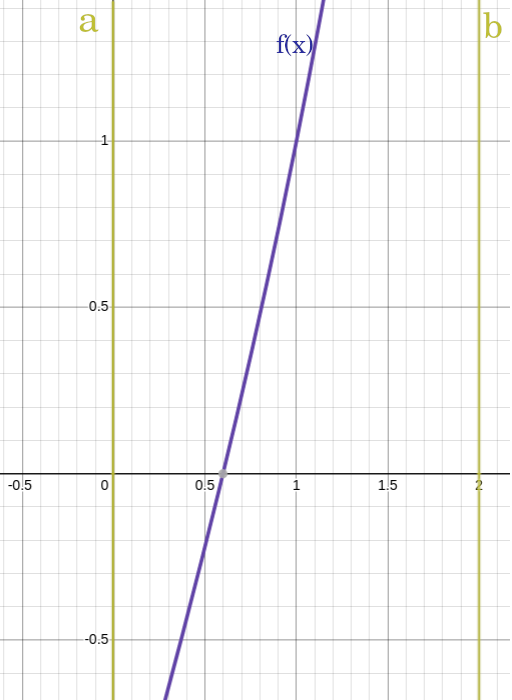
Условие выполняется на всём промежутке, следовательно метод применим.

2)Функция , [0,2].

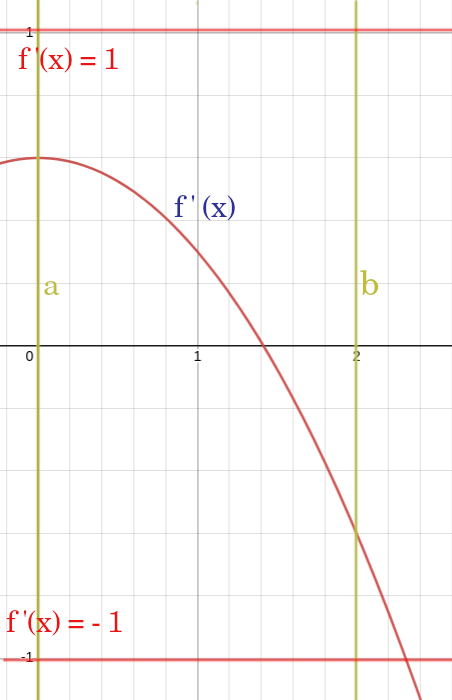
a)Для применения метода дихотомии нам необходимо проверить условие: (функция должна иметь разные знаки на концах отрезках). Исходя из графика, это условие выполняется, также при подставлении численных значений, получаем нужный результат:

f(0)= -1.2502<0;

f(2)= 2.7498 >0;

b)Для применения метода итерации нам необходимо, чтобы условие выполнялось на всём отрезке, при этом для метода итераций, мы берём функцию , которая принимает вид: А её производная f’(x)=

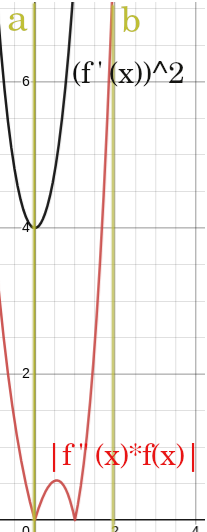
Видим, что производная больше 1 на всём отрезке, следовательно, метод неприменим.



Но по условию, нам сказано, что мы можем умножать функцию на произвольную константу , возьмём =-0.4, получим

f(x)= , тогда её производная

f’(x)=, получаем функцию, удовлетворяющую наши условия



c) Для применения метода Ньютона необходимо выполнения условия , где

f’(x)=

Условие выполняется на всём промежутке, следовательно метод применим.

## Описание переменных

| dx | double | приращение х |
| --- | --- | --- |
| a1,b1; a2,b2 | double | границы отрезков |
| res.x | double | приближённое значение корня уравнения |
| res.count\_i | int | число итераций |
| res.state | int | состояние выполнения |
| f\_1(x); f\_2(x) | функции double | возвращают значения функций |
| f\_1\_iter(x); f\_2\_iter(x) | функции double | возвращают значения функций для метода итераций |
| derivative1() | функция double | возвращает производную функции первого порядка |
| derivative2() | функция double | возвращает производную функции второго порядка |
| dich() | функция double | возвращает приближённое значение функции по методу дихотомии |
| iter() | функция double | возвращает приближённое значение функции по методу итерации |
| newton() | функция double | возвращает приближённое значение функции по методу Ньютона |

## 

## Протокол

#include <stdio.h>

#include <math.h>

#include <inttypes.h>

struct result{

double x;

int state;

int count\_i;

} res;

double f\_1(double x1){ //f\_1()-num 16

return 3\*sin(sqrt(x1))+0.35\*x1-3.8;

}

double f\_2(double x1){ //f\_2() -num 17

return 0.25\*x1\*x1\*x1+x1-1.2502;

}

double f\_1\_iter(double x1){

return -0.5\*(3\*sin(sqrt(x1))+0.35\*x1-3.8)+x1;

}

double f\_2\_iter(double x1){

return -0.4\*(0.25\*x1\*x1\*x1+x1-1.2502)+x1;

}

double derivative1(double (\*f)(double), double x0){

double dx = 0.00001;

return (f(x0+dx)-f(x0))/dx;

}

double derivative2(double (\*f)(double), double x0){

double dx = 0.00001;

return (derivative1(f, x0+dx) -derivative1(f, x0))/dx;

}

struct result dich(double (\*f)(double), double a, double b){

double c;

res.count\_i=0;

double e= 0.00001;

while (fabs(b-a>e)){

c=(a+b)/2;

if(f(a)\*f(c)<0){

b=c;

} else{

a=c;

}

res.count\_i++;

}

res.x=(a+b)/2;

res.state=1;

return res;

}

struct result iter(double (\*f)(double), double a, double b){

double x0=(a+b)/2;

double x1=x0;

double e= 0.00001;

res.count\_i=0;

do {

if (fabs(derivative1(f,x0))>=1){

res.state=0;

return res;

}

x0=x1;

x1=f(x0);

res.count\_i++;

// printf("%.5lf %.5lf \n", x0, x);

} while(fabs(x1-x0)>e);

res.state=1;

res.x=x1;

return res;

}

struct result nyton(double (\*f)(double), double a, double b){

double x0=(a+b)/2;

double e= 0.000001;

res.count\_i=0;

double x1=x0;

do{

if (fabs(f(x0)\*derivative2(f, x0))>= derivative1(f, x0)\*derivative1(f, x0)){

res.state=0;

return res;

}

x0=x1;

x1=x0-f(x0)/derivative1(f, x0);

res.count\_i++;

} while(fabs(x1-x0)>e);

res.state=1;

res.x=x1;

return res;

}

int main(){

double a1=2, b1=3, a2=0, b2=2;

int n;

printf("Функция 3\*sin(sqrt(x1))+0.35\*x1-3.8 \n \n");

struct result dich\_1=dich(f\_1,a1, b1);

double x1d=dich\_1.x;

if (f\_1(a1)\*f\_1(b1)<0){

printf("Метод дихотомии применим. Корень = %.5lf. Число итераций: %d \n", x1d, dich\_1.count\_i );

} else {

printf("Метод дихотомии неприменим \n");

x1d=0;

}

struct result iter\_1=iter(f\_1\_iter,a1, b1);

double x1i=iter\_1.x;

if (iter\_1.state==1){

printf("Метод итерации применим. Корень = %.5lf. Число итераций: %d \n", x1i, iter\_1.count\_i);

} else{

printf("Метод итерации неприменим. \n");

x1i=0;

}

struct result nyton\_1=nyton(f\_1,a1, b1);

double x1n=nyton\_1.x;

if (nyton\_1.state==1){

printf("Метод Ньютона применим. Корень = %.5lf. Число итераций: %d \n \n", x1n, nyton\_1.count\_i);

} else{

printf("Метод Ньютона неприменим. \n");

x1n=0;

}

printf("Разность результатов метода Ньютона и метода итераций %e \n", fabs(x1n-x1i));

printf("Разность результатов метода дихотомии и метода итераций %e \n", fabs(x1d-x1i));

printf("Разность результатов метода Ньютона и метода дихотомии %e \n", fabs(x1d-x1n));

printf("---------------------------------------------------------------------\n \n");

printf("Функция 0.25\*x1\*x1\*x1+x1-1.2502 \n \n");

struct result dich\_2=dich(f\_2,a2, b2);

double x2d=dich\_2.x;

if (f\_2(a2)\*f\_2(b2)<0){

struct result dich\_2=dich(f\_2,a2, b2);

double x2d=dich\_2.x;

printf("Метод дихотомии применим. Корень = %.5lf. Число итераций: %d \n", x2d, dich\_2.count\_i );

} else {

printf("Метод дихотомии неприменим \n");

x2d=0;

}

struct result iter\_2=iter(f\_2\_iter,a2, b2);

double x2i=iter\_2.x;

if (iter\_2.state==1){

printf("Метод итерации применим. Корень = %.5lf. Число итераций: %d \n", x2i, iter\_2.count\_i);

} else{

printf("Метод итерации неприменим. \n");

x2i=0;

}

struct result nyton\_2=nyton(f\_2,a2, b2);

double x2n=nyton\_2.x;

if (nyton\_2.state==1){

printf("Метод Ньютона применим. Корень = %.5lf. Число итераций: %d \n", x2n, nyton\_2.count\_i);

} else{

printf("Метод Ньютона неприменим. \n");

x2n=0;

}

printf("\n");

printf("Разность результатов метода Ньютона и метода итераций %e \n", fabs(x2n-x2i));

printf("Разность результатов метода дихотомии и метода итераций %e \n", fabs(x2d-x2i));

printf("Разность результатов метода Ньютона и метода дихотомии %e \n", fabs(x2d-x2n));

printf("\n ");

}

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog$ gcc -std=c99 -pedantic kp4.c -lm

arsenii@LarchCompu:~/Documents/prog/projects/ones/Cprog$ ./a.out

Функция 3\*sin(sqrt(x1))+0.35\*x1-3.8

Метод дихотомии применим. Корень = 2.29853. Число итераций: 17

Метод итерации применим. Корень = 2.29857. Число итераций: 39

Метод Ньютона применим. Корень = 2.29854. Число итераций: 4

Разность результатов метода Ньютона и метода итераций 3.372083e-05

Разность результатов метода дихотомии и метода итераций 3.549845e-05

Разность результатов метода Ньютона и метода дихотомии 1.777612e-06

---------------------------------------------------------------------

Функция 0.25\*x1\*x1\*x1+x1-1.2502

Метод дихотомии применим. Корень = 1.00011. Число итераций: 18

Метод итерации применим. Корень = 1.00011. Число итераций: 3

Метод Ньютона применим. Корень = 1.00011. Число итераций: 2

Разность результатов метода Ньютона и метода итераций 3.083938e-06

Разность результатов метода Ньютона и метода итераций 5.699585e-07

Разность результатов метода Ньютона и метода дихотомии 3.653896e-06

[1] + Done

# Заключение

В ходе выполнения задания я проделал большую работу: первым делом я ознакомился с тремя методами решения трансцендентных уравнений: методом дихотомии, методом итераций и методом Ньютона; потом я аналитически через построение графиков проверил применимость каждого из методов к данным функциям, после этого я составил алгоритм выполнения программы, а затем перешёл к её реализации на СИ.

В этом КП я научился решать трансцендентные уравнения через программу, составленную на языке СИ.

Особых сложностей не возникло, единственное, пришлось долго осознавать теорию…

В конечном итоге у меня получилось написать исправно-работающую программу, результаты её работы соответствуют аналитическому решению, поэтому я считаю, что справился с поставленной задачей.

# Список используемой литературы

1)Дихотомия <http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4%D1%8B_%D0%B4%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%D0%B8>

2)Метод итераций <http://school-collection.edu.ru/catalog/res/21808a0c-95ed-448f-9098-68061e6e8fe0/view/>

3)Метод Ньютона

<https://core.ac.uk/download/pdf/50576492.pdf>