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»

**КУРСОВОЙ ПРОЕКТ**

По курсу «Вычислительные системы»

I семестр

Тема:

«Процедуры и функции в качестве параметров»

|  |  |
| --- | --- |
| Группа: | М8О-106Б-21 |
| Студент: | Лохматов Н. И. |
| Преподаватель: | Дубинин А.В. |
| Оценка: |  |
| Дата: |  |

Москва, 2021

Оглавление

[Введение 3](#_Toc92109898)

[Теория 4](#_Toc92109899)

[Метод дихотомии 4](#_Toc92109900)

[Метод итераций 5](#_Toc92109901)

[Метод Ньютона 6](#_Toc92109902)

[Практика 8](#_Toc92109903)

[Задание 8](#_Toc92109904)

[Описание программы 8](#_Toc92109905)

[Использованные переменные 8](#_Toc92109906)

[Тесты 8](#_Toc92109907)

[Заключение 9](#_Toc92109908)

[Источники 9](#_Toc92109909)

# Введение

Нахождение корней трансцендентных уравнений очень часто является довольно сложной задачей, которую нельзя решить аналитически с помощью конечных формул. Также бывают случаи, когда на практике уравнение содержит коэффициенты, значения которых заданы приблизительно, так что говорить о точном решении уравнений в таких случаях не стоит, поэтому задачи приближенного определения корней уравнения и соответствующей оценки их точности имеют большое значение. Цель этой работы: составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления – дихотомии). Применить каждую процедуру к решению двух уравнений.

# Теория

## Метод дихотомии

Перед применением метода для поиска корней функции необходимо отделить корни одним из известных способов, например, графическим методом. Отделение корней необходимо в случае, если неизвестно на каком отрезке нужно искать корень.

Будем считать, что корень t функции f(x)=0 отделён на отрезке [a,b]. Задача заключается в том, чтобы найти и уточнить этот корень методом половинного деления. Другими словами, требуется найти приближённое значение корня с заданной точностью \eps.

Пусть функция f непрерывна на отрезке [a,b],

f(a)\cdot f(b)<0, \; \eps=0,01 и t\in[a,b] - единственный корень уравнения f(x)=0, \; a\le t\le b.

(Мы не рассматриваем случай, когда корней на отрезке [a,b] несколько, то есть более одного. В качестве \eps можно взять и другое достаточно малое положительное число, например, 0,001.)

Поделим отрезок [a,b] пополам. Получим точку c= \frac {a+b}{2}, \; a<c<b и два отрезка [a,c], \; [c,b].

* Если f(c)=0, то корень t найден (t=c).
* Если нет, то из двух полученных отрезков [a,c] и [c,b] надо выбрать один [a_1;b_1] такой, что f(a_1)\cdot f(b_1)<0, то есть
  + [a_1;b_1] = [a,c], если f(a)\cdot f(c)<0 или
  + [a_1;b_1] = [c,b], если f(c)\cdot f(b)<0.

Новый отрезок [a_1;b_1] делим пополам. Получаем середину этого отрезка c_1=\frac {a_1+b_1}{2} и так далее.

Для того, чтобы найти приближённое значение корня с точностью до  \eps >0, необходимо остановить процесс половинного деления на таком шаге n, на котором |b_n-c_n|<\eps и вычислить x=\frac {a_n+b_n}{2}. Тогда можно взять t\approx x.

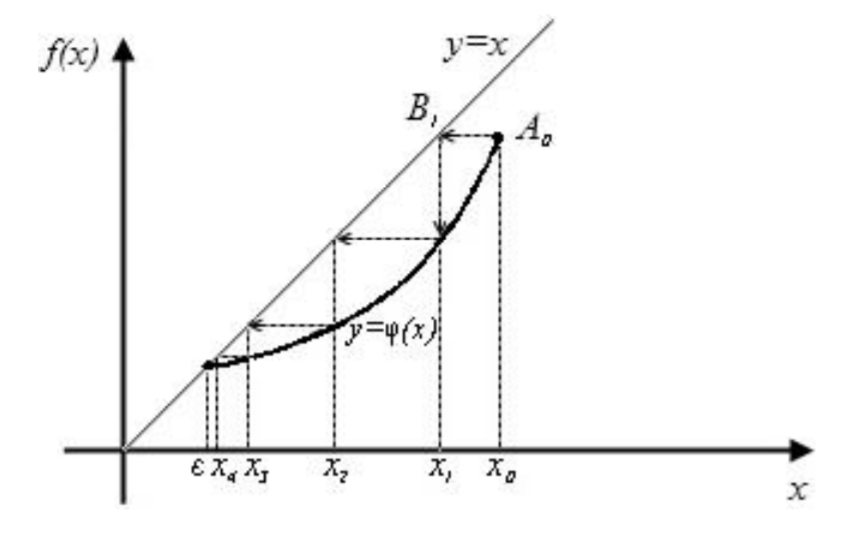


## Метод итераций

**Метод итерации** — [численный метод](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B) решения [системы линейных алгебраических уравнений](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D1%85_%D0%B0%D0%BB%D0%B3%D0%B5%D0%B1%D1%80%D0%B0%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D1%85_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D0%B9). Суть метода заключается в нахождении по приближённому значению величины следующего приближения, являющегося более точным.

Метод позволяет получить значения корней системы с заданной точностью в виде [предела последовательности](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%B5%D0%BB_%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8) некоторых векторов (в результате итерационного процесса). Характер сходимости и сам факт сходимости метода зависит от выбора начального приближения корня.

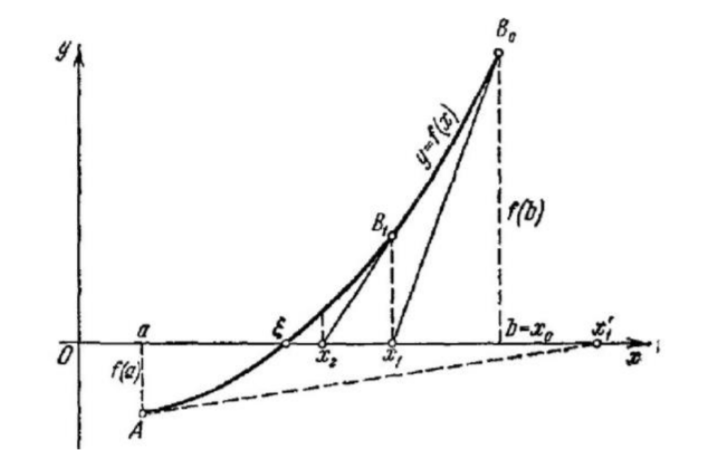
Идея метода заключается в замене исходного уравнения 𝐹(𝑥) = 0 уравнением вида 𝑥 = 𝑓(𝑥). Достаточное условие сходимости метода |𝑓′(𝑥)| < 1, 𝑥 ∈ [𝑎; 𝑏]. Это условие необходимо проверить перед началом решения задачи, так как функция 𝑓(𝑥) может быть выбрана неоднозначно, причем в случае неверного выбора указанной функции метод расходится.



## Метод Ньютона

**Метод Ньютона** — это итерационный [численный метод](https://ru.wikipedia.org/wiki/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B) нахождения корня ([нуля](https://ru.wikipedia.org/wiki/%D0%9D%D1%83%D0%BB%D1%8C_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8)) заданной [функции](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)). Метод был впервые предложен английским [физиком](https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B7%D0%B8%D0%BA), [математиком](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA) и [астрономом](https://ru.wikipedia.org/wiki/%D0%90%D1%81%D1%82%D1%80%D0%BE%D0%BD%D0%BE%D0%BC) [Исааком Ньютоном](https://ru.wikipedia.org/wiki/%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD,_%D0%98%D1%81%D0%B0%D0%B0%D0%BA). Поиск решения осуществляется путём построения последовательных приближений и основан на принципах [простой итерации](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8). Метод обладает квадратичной [сходимостью](https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D0%BE%D1%80%D0%BE%D1%81%D1%82%D1%8C_%D1%81%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%D1%81%D1%82%D0%B8). Модификацией метода является [метод хорд и касательных](https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D1%85%D0%BE%D1%80%D0%B4_%D0%B8_%D0%BA%D0%B0%D1%81%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D1%85). Также метод Ньютона может быть использован для решения [задач оптимизации](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%BE%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8), в которых требуется определить ноль первой [производной](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) либо [градиента](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D0%B8%D0%B5%D0%BD%D1%82_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)) в случае многомерного пространства.

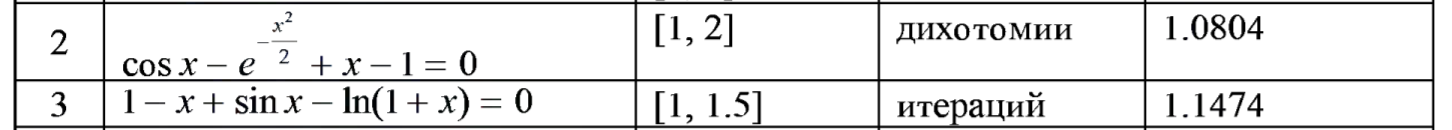
Метод Ньютона является частным случаем метода итераций. Условие сходимости метода: |𝐹(𝑥) ∙ 𝐹"(𝑥)| < (𝐹 ′ (𝑥))² на отрезке [𝑎; 𝑏].



# Практика

## Задание

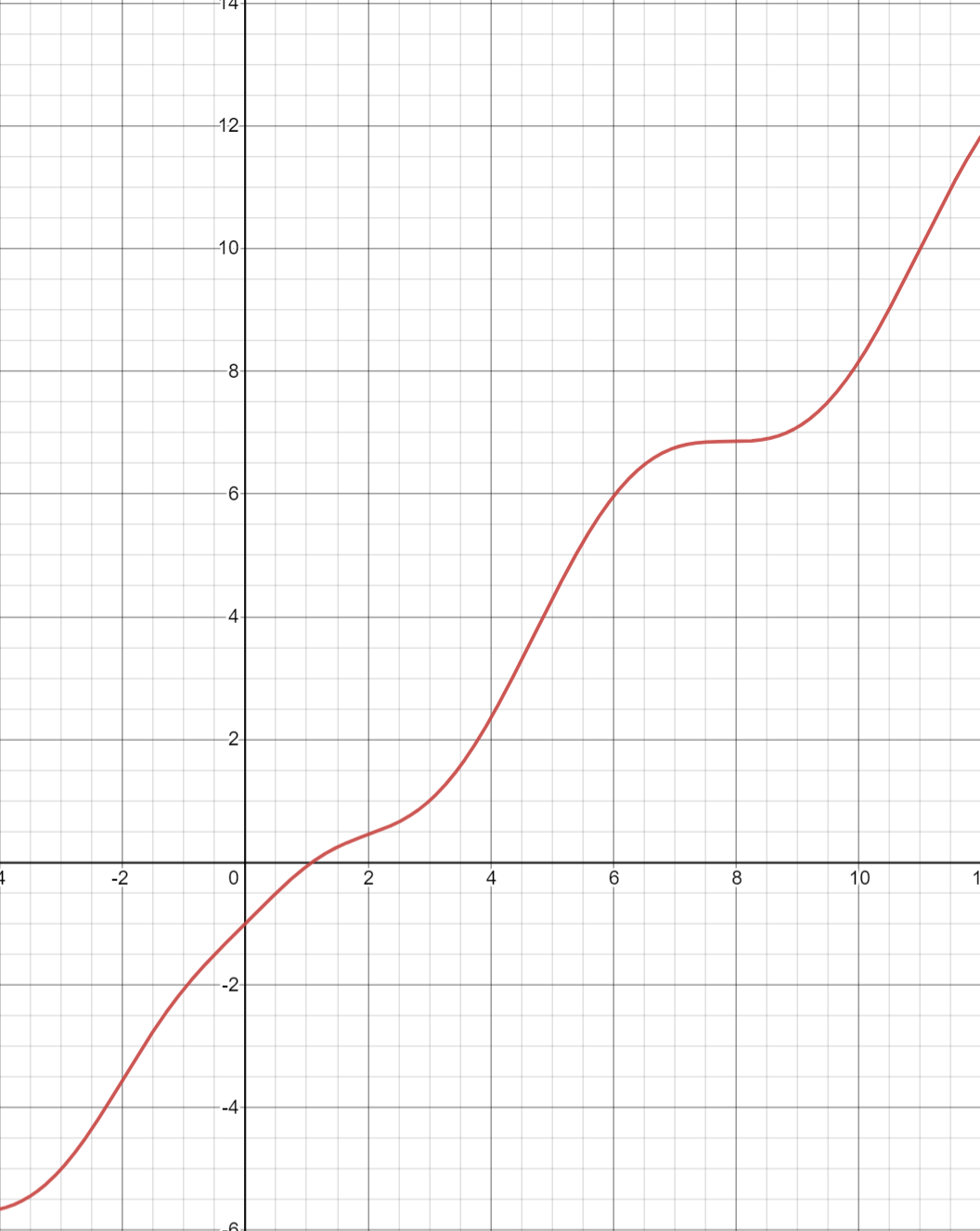
Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления – дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. 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.



## Графики функций

**Вариант 2:**

F(x) = cos(x) – e-(x^2/2) + x - 1, [1, 2]



Производная f’(x) = e-(x^2/2) \* x – sin(x) + 1:



Условие |𝑓′ (𝑥)| < 1 выполняется на всём отрезке [1; 2], следовательно, условие сходимости метода итераций выполнено.

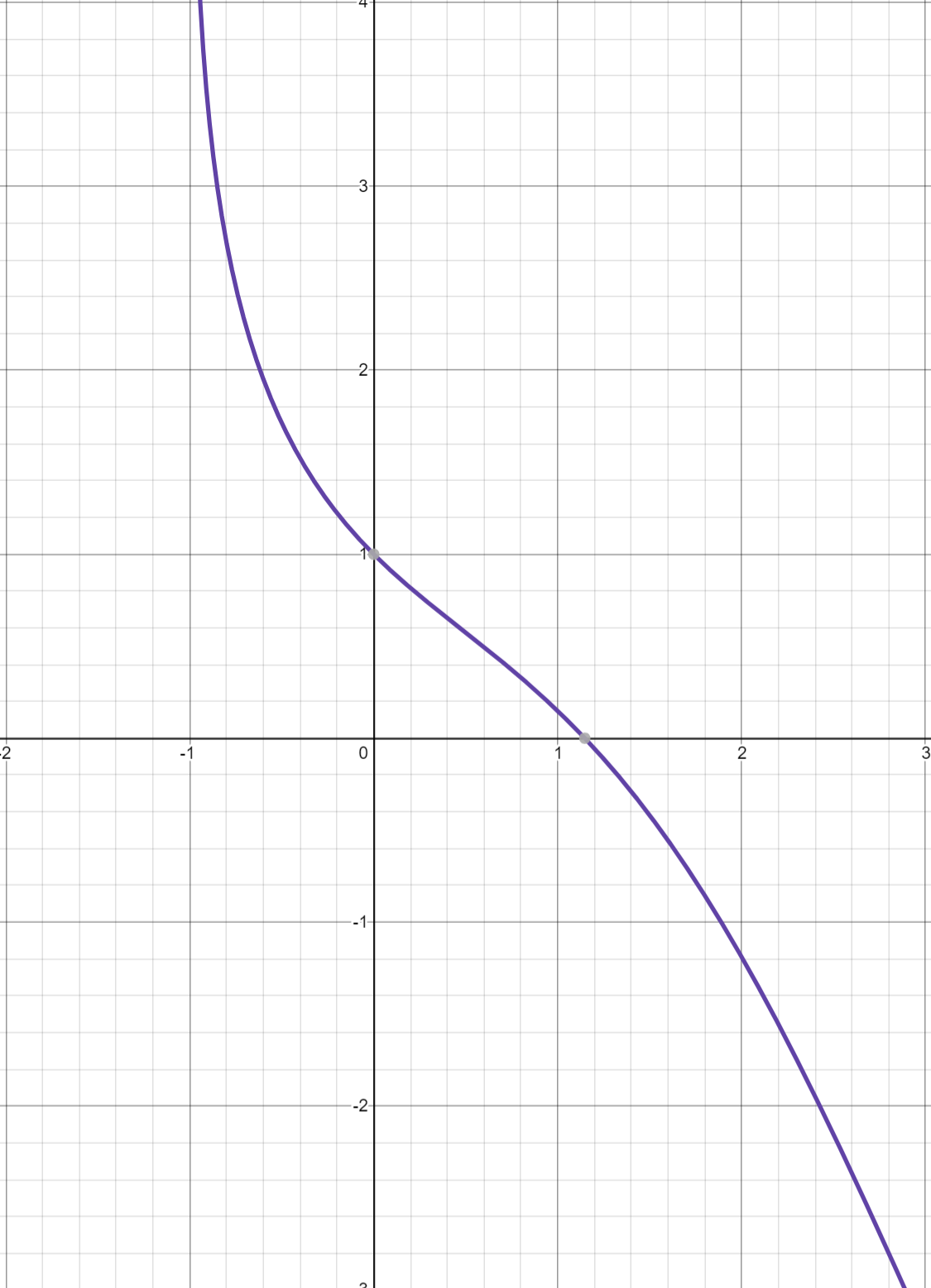
Графики |𝐹(𝑥) ∙ 𝐹"(𝑥)| и (𝐹′(𝑥))^2 (синий и зеленый):



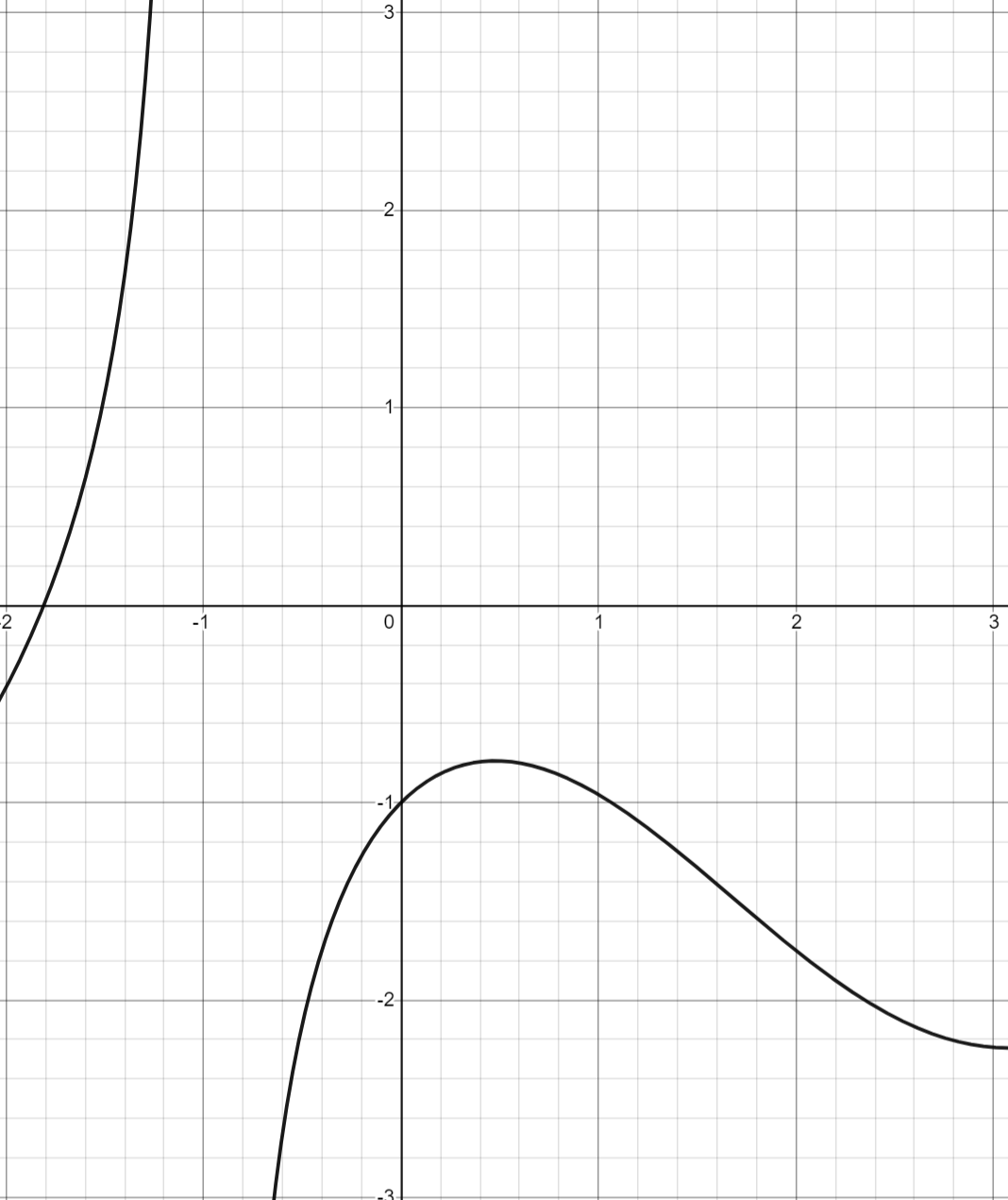
Как видно, на отрезке [1; 2] |𝐹(𝑥) ∙ 𝐹"(𝑥)| < (𝐹′(𝑥))^2, значит метод Ньютона выполняется.

**Вариант 3:**

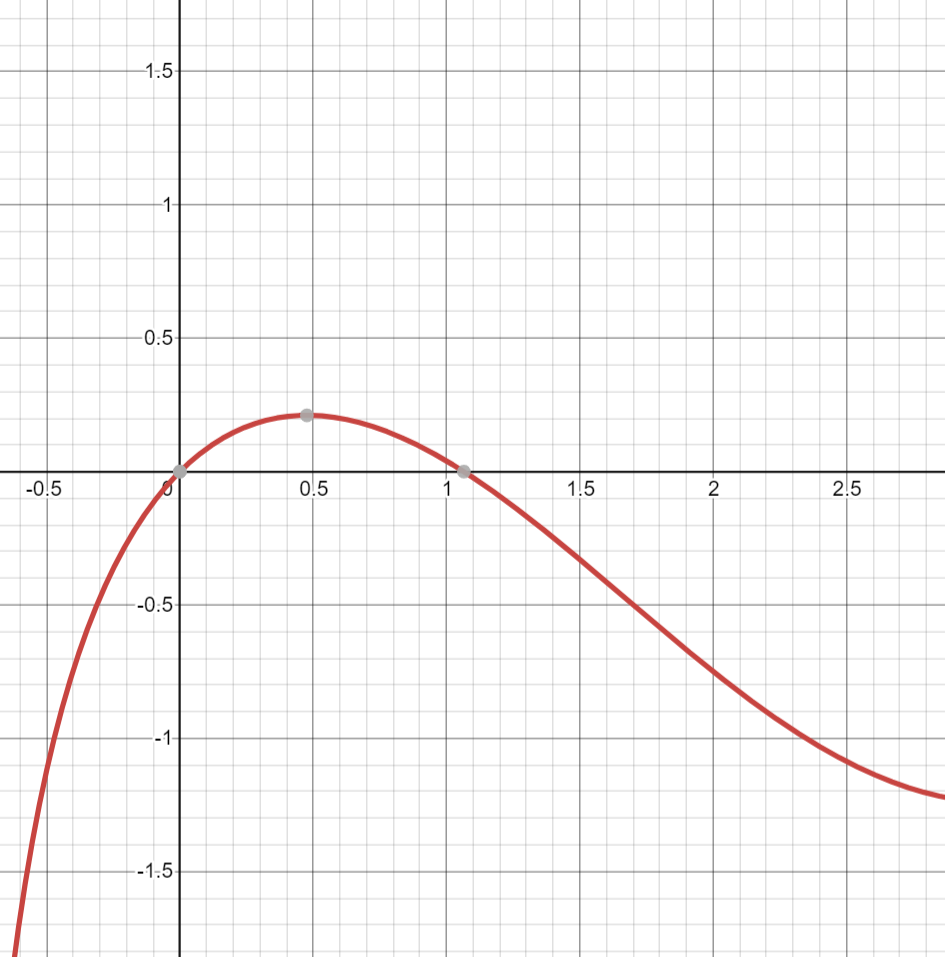
F(x) =1 – x + sin(x) – ln(1+x), [1, 1.5]

****

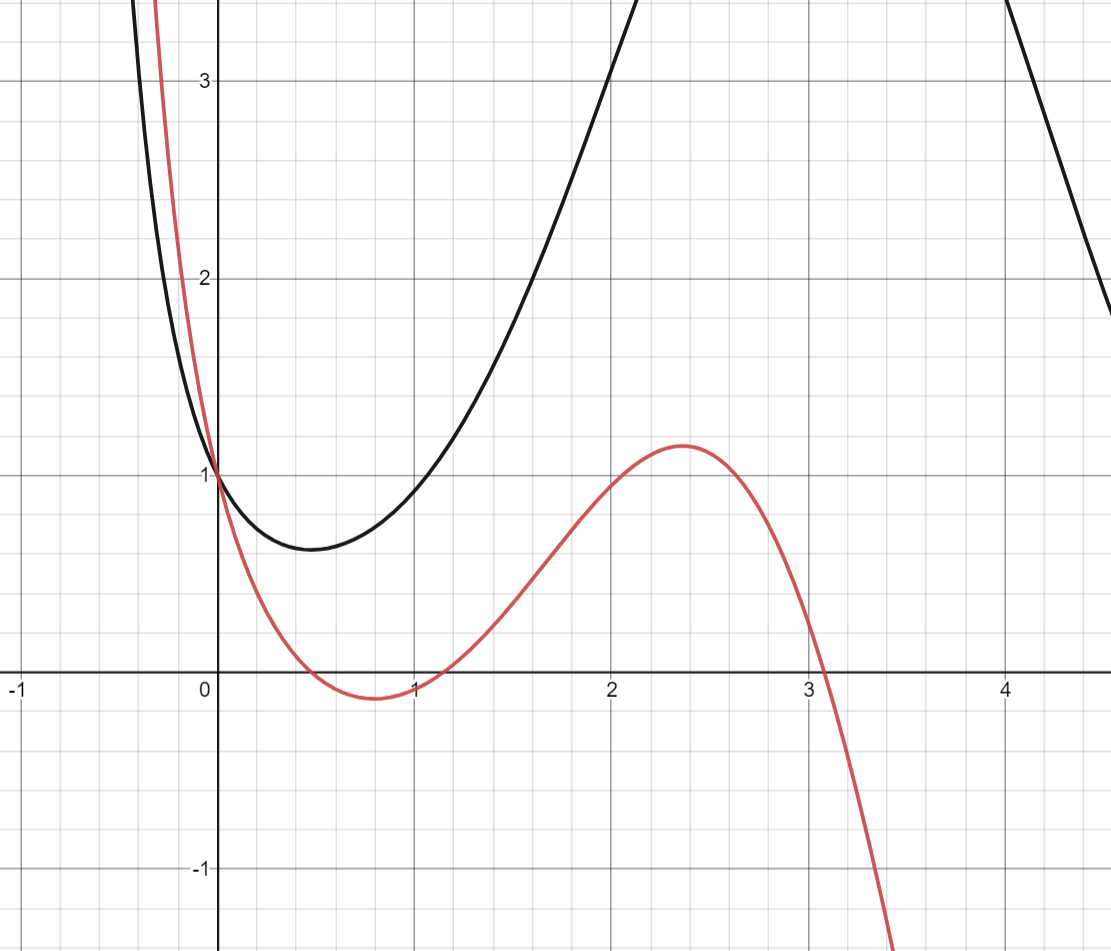
Производная f’(x) = -1/(1+x) + cos(x) – 1:



Метод итераций неприменим к решению данного уравнения, так как существуют значения производных функций ≥ 1 на отрезке, содержащем корень.



Графики |𝐹(𝑥) ∙ 𝐹"(𝑥)| и (𝐹′(𝑥))^2 (красный и чёрный):

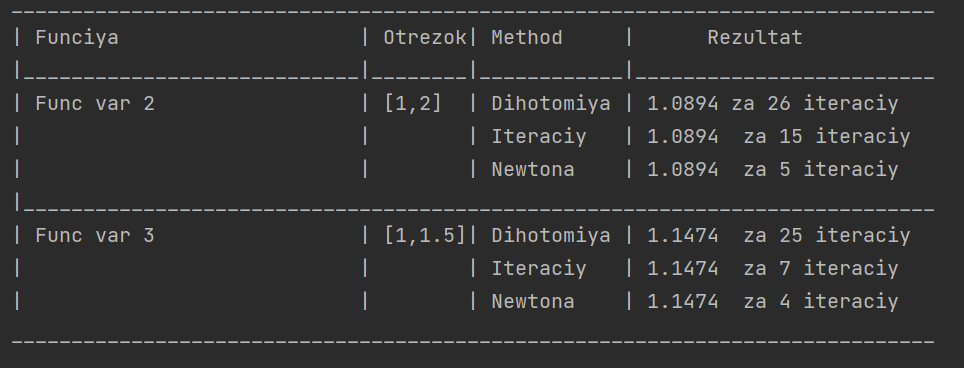


Как видно, на отрезке [1; 1.5] |𝐹(𝑥) ∙ 𝐹"(𝑥)| < (𝐹′(𝑥))^2, значит метод Ньютона выполняется.

## Протокол

#include <stdio.h>  
#include <math.h>  
#include <stdbool.h>  
  
typedef double (\*func)(double);  
  
typedef struct {  
 double root;  
 int iters;  
 bool success;  
} result;  
  
double Fx\_2(double x){  
 return cos(x) - exp(-x\*x/2) + x - 1;  
}  
double dFx\_2(double x){  
 return exp(-x\*x/2)\*x - sin(x) + 1;  
}  
double d2Fx\_2(double x){  
 return -exp(-x\*x/2)\*(x\*x + exp(x\*x/2)\* cos(x) - 1);  
}  
  
double fx\_2(double x){  
 return exp(-x\*x/2) - cos(x) + 1;  
}  
double dfx\_2(double x){  
 return sin(x) - exp(-x\*x/2)\*x;  
}  
  
  
double Fx\_3(double x){  
 return 1 - x + sin(x) - log(1 + x);  
}  
double dFx\_3(double x){  
 return -1/(1+x) + cos(x) - 1;  
}  
double d2Fx\_3(double x){  
 return 1/((1+x)\*(1+x)) - sin(x);  
}  
double fx\_3(double x){  
 return 1 + sin(x) - log(1 + x);  
}  
double dfx\_3(double x){  
 return cos(x) - 1/(1+x);  
}  
  
result diho(func f, double a, double b){  
 result res;  
 double delta = pow(2, -26);  
 res.iters = 0;  
 while (b-a > delta){  
 double c = (a+b)/2;  
 if (f(a)\*f(c) < 0){  
 b = c;  
 } else {  
 a = c;  
 }  
 res.iters++;  
 }  
 res.root = (a+b)/2;  
 res.success = 1;  
 return res;  
}  
  
result newton(func f, func df, func d2f, double a, double b){  
 result res;  
 double x = (a+b)/2;  
 if (fabs(f(x)\*d2f(x)) >= df(x)\*df(x)){  
 res.success = 0;  
 return res;  
 }  
 double x1 = x - (f(x)/df(x));  
 if (fabs(f(x1)\*d2f(x1)) >= df(x1)\*df(x1)){  
 res.success = 0;  
 return res;  
 }  
 double delta = pow(2, -26);  
 res.iters = 1;  
 while (fabs(x - x1) > delta){  
 x = x1;  
 if (fabs(f(x)\*d2f(x)) >= df(x)\*df(x)){  
 res.success = 0;  
 return res;  
 }  
 x1 = x - (f(x)/df(x));  
 res.iters++;  
 }  
 res.root = x1;  
 res.success = 1;  
 return res;  
}  
  
result iters(func f, func fx, func dfx, double a, double b){  
 result res;  
 double x = (a+b)/2;  
 double delta = pow(2, -26);  
 if (dfx(x) >= 1){  
 res.success = 0;  
 return res;  
 }  
 res.iters = 1;  
 double x1 = fx(x);  
 while (fabs(x - x1) > delta){  
 x = x1;  
 if (dfx(x) >= 1){  
 res.success = 0;  
 return res;  
 }  
 x1 = fx(x);  
 res.iters++;  
 }  
 res.root = x1;  
 res.success = 1;  
 return res;  
}  
  
int main() {  
   
 result x1 = diho(Fx\_2, 1, 2),   
 x2 = iters(Fx\_2, fx\_2, dfx\_2, 1, 2),   
 x3 = newton(Fx\_2, dFx\_2, d2Fx\_2, 1, 2);  
   
 result x4 = diho(Fx\_3, 1, 1.5),   
 x5 = iters(Fx\_3, fx\_3, dfx\_3, 1, 1.5),   
 x6 = newton(Fx\_3, dFx\_3, d2Fx\_3, 1, 1.5);  
  
 printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 printf("| Funciya | Otrezok| Method | Rezultat \n");  
 printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
   
 if (x1.success == 1){  
 printf("| Func var 2 | [1,2] | Dihotomiya | %.4lf za %d iteraciy \n", x1.root, x1.iters);  
 } else {  
 printf("| Func var 2 | [1,2] | Dihotomiya | Method ne primenim \n");  
 }  
 if (x2.success == 1){  
 printf("| | | Iteraciy | %.4lf za %d iteraciy \n", x2.root, x2.iters);  
 } else {  
 printf("| | | Iteraciy | Method ne primenim \n");  
 }  
 if (x3.success == 1){  
 printf("| | | Newtona | %.4lf za %d iteraciy \n", x3.root, x3.iters);  
 } else {  
 printf("| | | Ньютона | Method ne primenim \n");  
 }  
 printf("|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
 if (x4.success == 1){  
 printf("| Func var 3 | [1,1.5]| Dihotomiya | %.4lf za %d iteraciy \n", x4.root, x4.iters);  
 } else {  
 printf("| Func var 3 | [1,1.5] | Dihotomiya | Method ne primenim \n");  
 }  
 if (x5.success == 1){  
 printf("| | | Iteraciy | %.4lf za %d iteraciy \n", x5.root, x5.iters);  
 } else {  
 printf("| | | Iteraciy | Method ne primenim \n");  
 }  
 if (x6.success == 1){  
 printf("| | | Newtona | %.4lf za %d iteraciy \n", x6.root, x6.iters);  
 } else {  
 printf("| | | Newtona | Method ne primenim \n");  
 }  
 printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");  
   
}

## Тесты



# Заключение

В ходе работы над курсовым проектом были рассмотрены три численных метода решения трансцендентных уравнений: метод дихотомии, метод итераций, метод Ньютона. Используя сведения из численных методов, я составила программу на языке Си с процедурами решения трансцендентных алгебраический уравнений, а также математически, используя графики функций и их производных, доказала неприменимость метода итераций к решению уравнения.

# Источники

* Метод Ньютона <http://statistica.ru/branches-maths/chislennye-metody-resheniya-uravneniy/>
* Метод итераций <https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B8%D1%82%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8>
* Метод дитохомии <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>