Московский Авиационный Институт (Национальный Исследовательский Университет)

Курсовой проект  
по курсу «Вычислительные системы»

1 семестр

Задание 4 «Процедуры и функции в качестве параметров»

Выполнил: Бровина В.Д.

Группа: М8О-111Б-22

Руководитель: Аносова Н.П.

Оценка: \_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_

Москва, 2022

Оглавление

[Теоретическая часть 3](#_Toc122709163)

[Метод дихотомии (половинного деления) 3](#_Toc122709164)

[Метод итераций 4](#_Toc122709165)

[Метод Ньютона 5](#_Toc122709166)

[Практическая часть 6](#_Toc122709167)

[Задание 6](#_Toc122709168)

[Анализ применимости методов для функций 6](#_Toc122709169)

[Таблица используемых переменных 10](#_Toc122709170)

[Таблица используемых функций 11](#_Toc122709171)

[Листинг программного кода 12](#_Toc122709172)

[Выходные данные 18](#_Toc122709173)

[Вывод 18](#_Toc122709174)

[Список литературы 18](#_Toc122709175)

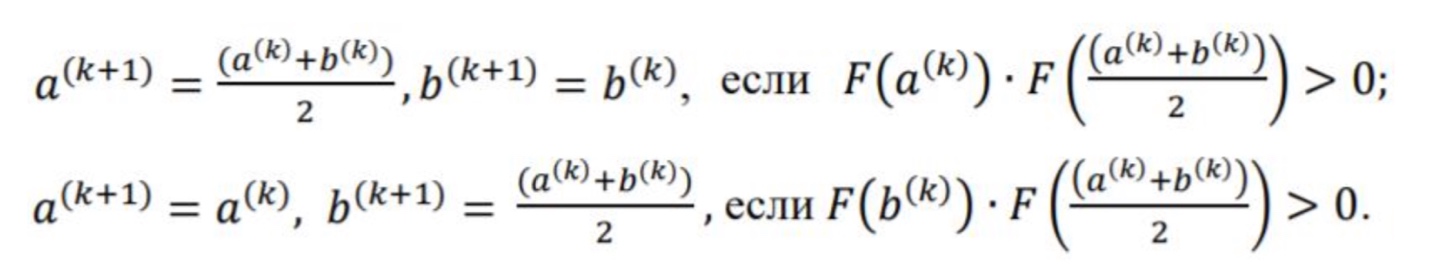
# **Теоретическая часть**

## **Метод дихотомии (половинного деления)**

Если на отрезке существует корень уравенеия, то значения функции на концах отрезка будут иметь разные знаки: . Метод заключается в делении данного отрезка пополам => в его сужении в два раза на каждом шаге процесса итерации в зависимости от знака функции на середине отрезка.

Итерационный процесс строится таким образом: за начальное приближение принимаются границы исходного отрезка .

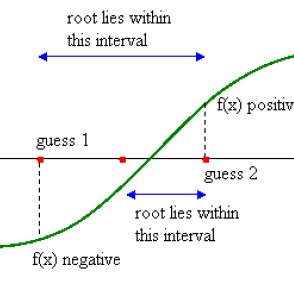
Дальнейшие вычисления производятся по формулам:



Процесс выполняется до тех пор, пока не будет выполнено условие окончания:

Приближенное значение корня к моменту окончания итерационного процессы получается следующим образом:

Иллюстрация метода дихотомии:



## **Метод итераций**

Идея метода заключается в замене исходного уравнения уравнением вида .

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

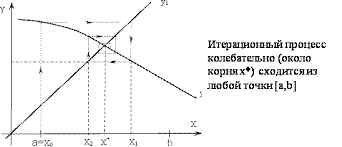
Начально приближение корня:

Итерационный процесс:

Условие окончания:

Приближенное значение корня:

Иллюстрация метода итераций:



## **Метод Ньютона**

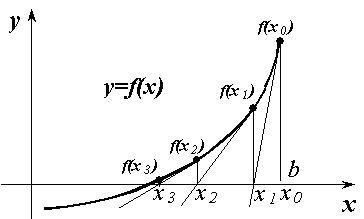
Метод Ньютона является частным случаем метода итераций.

Условие сходимости метода:

на отрезке

Итерационный процесс:

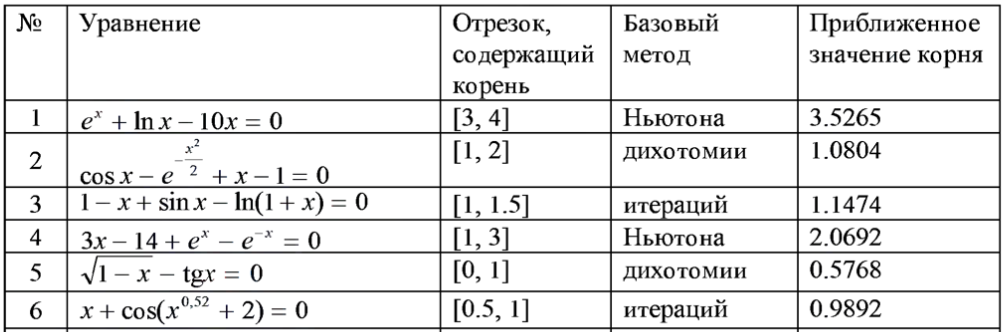
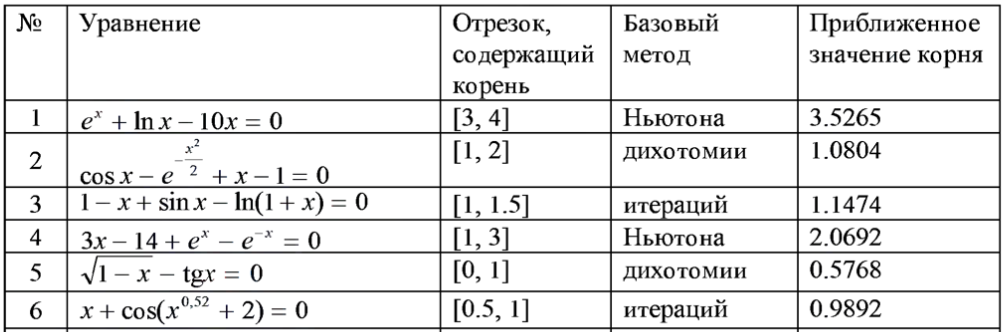
Иллюстрация метода Ньютона:



# **Практическая часть**

## **Задание**

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений с различными численными методами (дихотомии (половинного деления), итераций, Ньютона). 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.

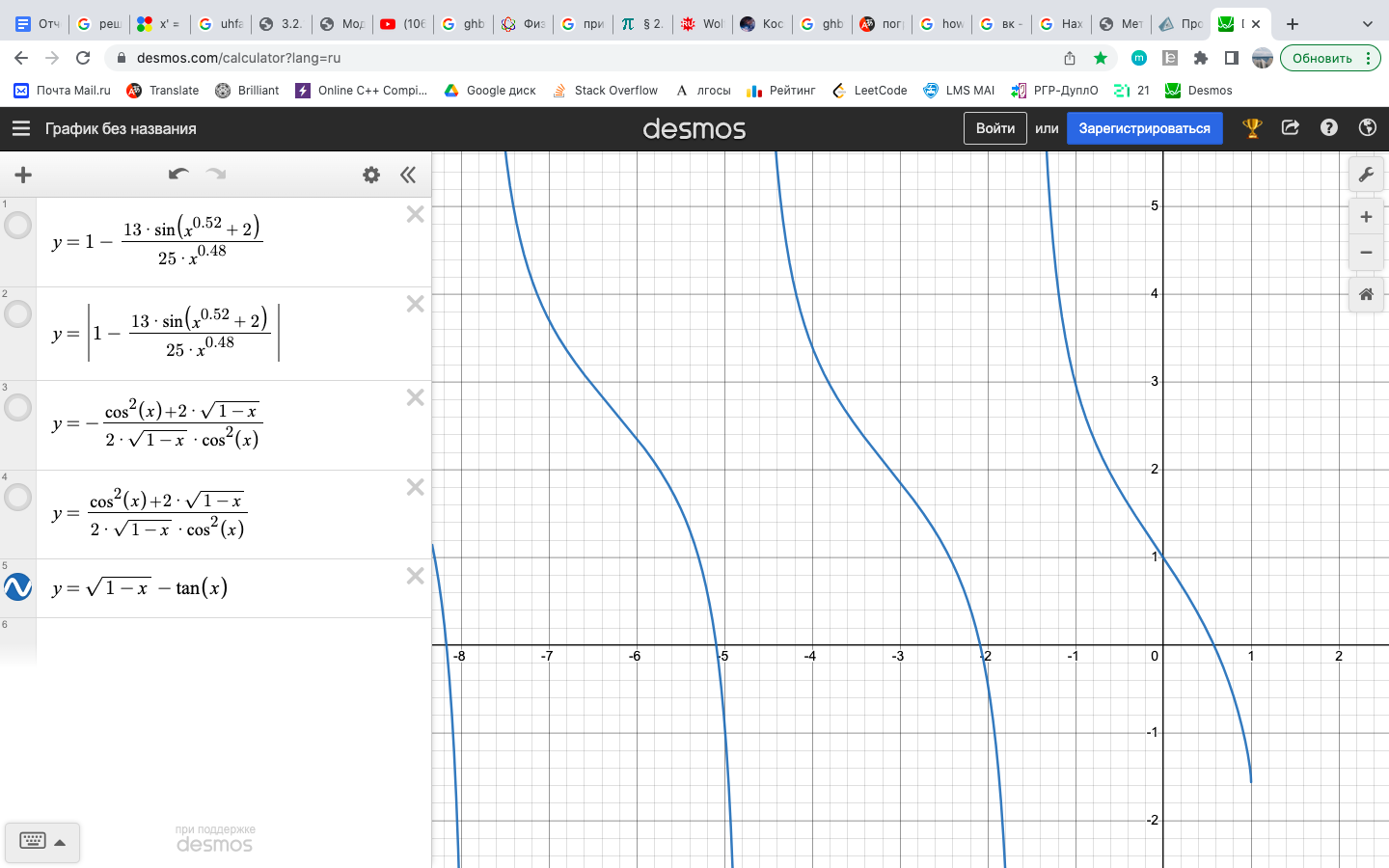
**Вариант №5, 6**

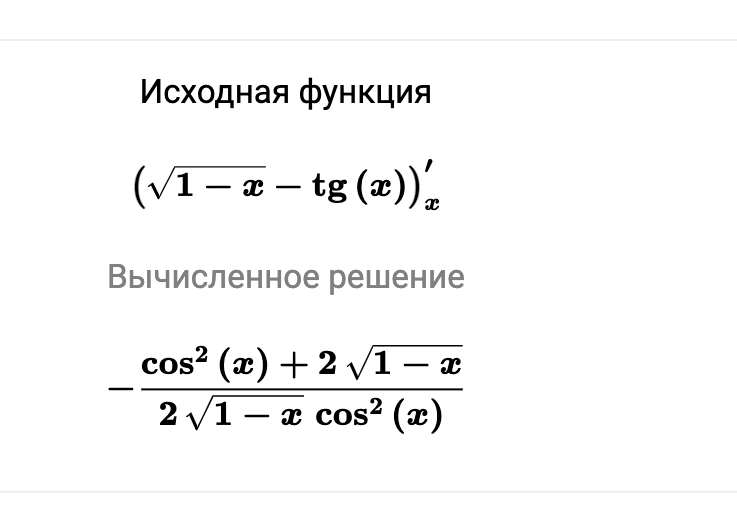
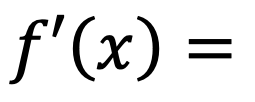
## **Анализ применимости методов для функций**

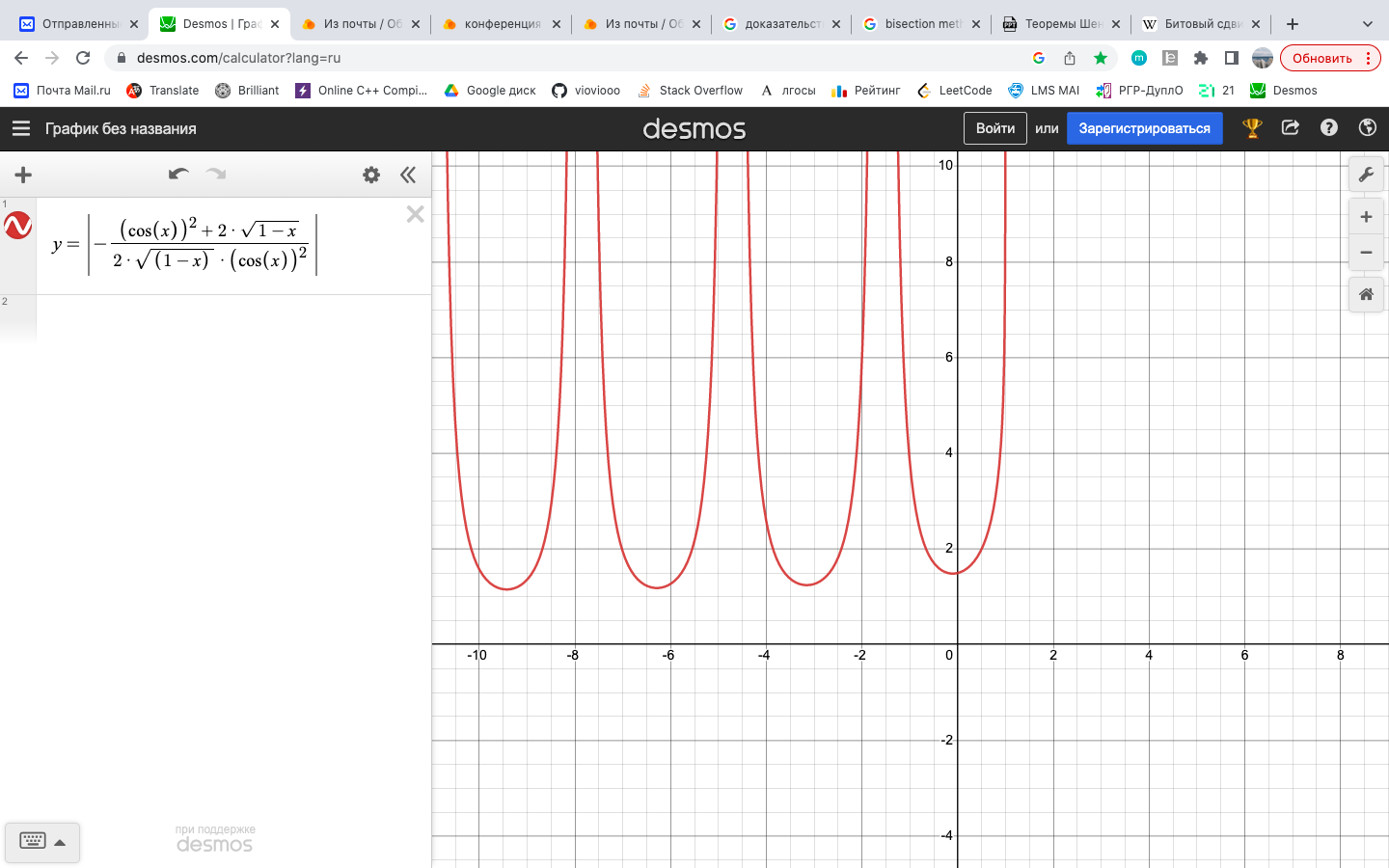
**Графики функций и их производных**

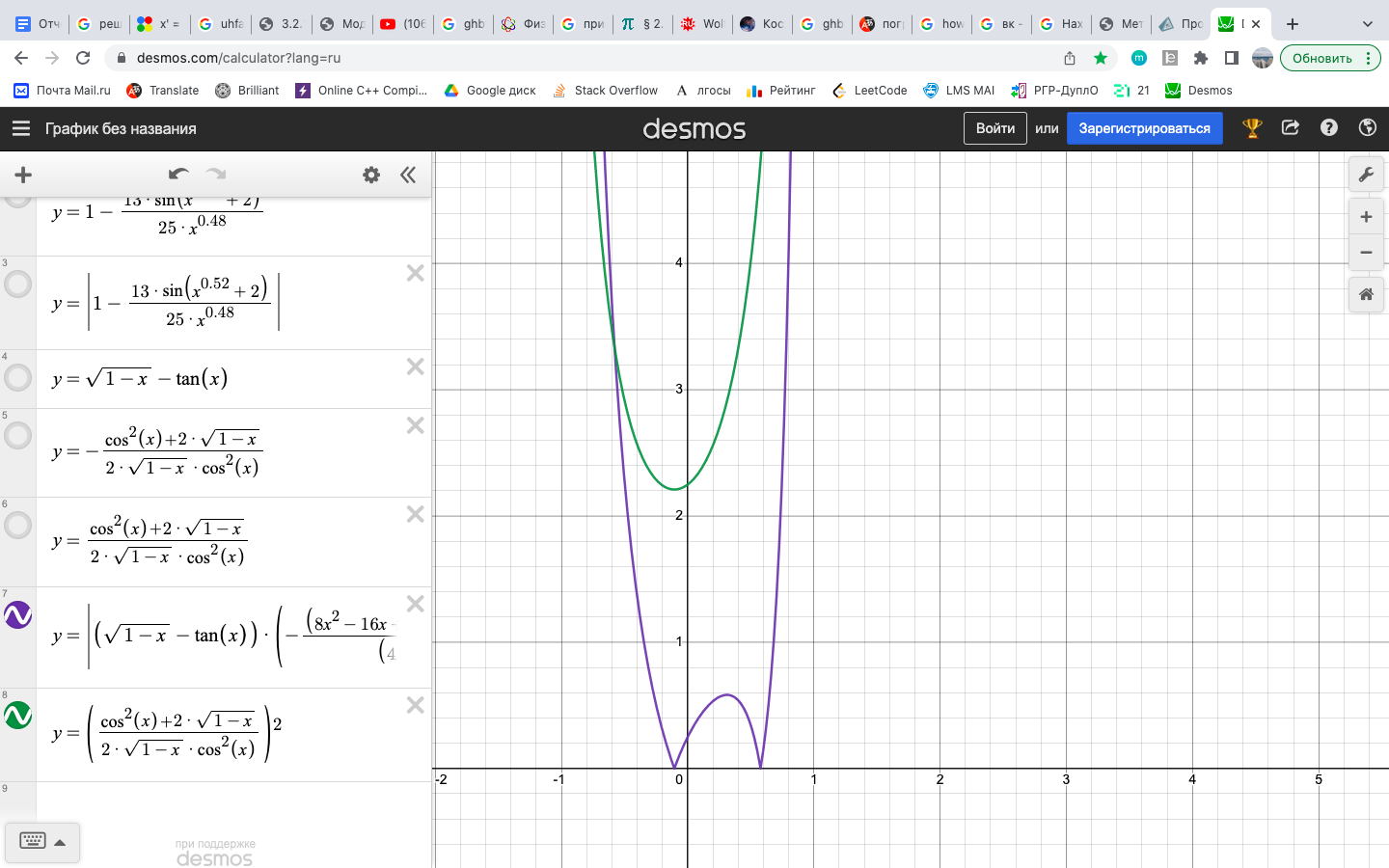
*Функция из варианта №5*

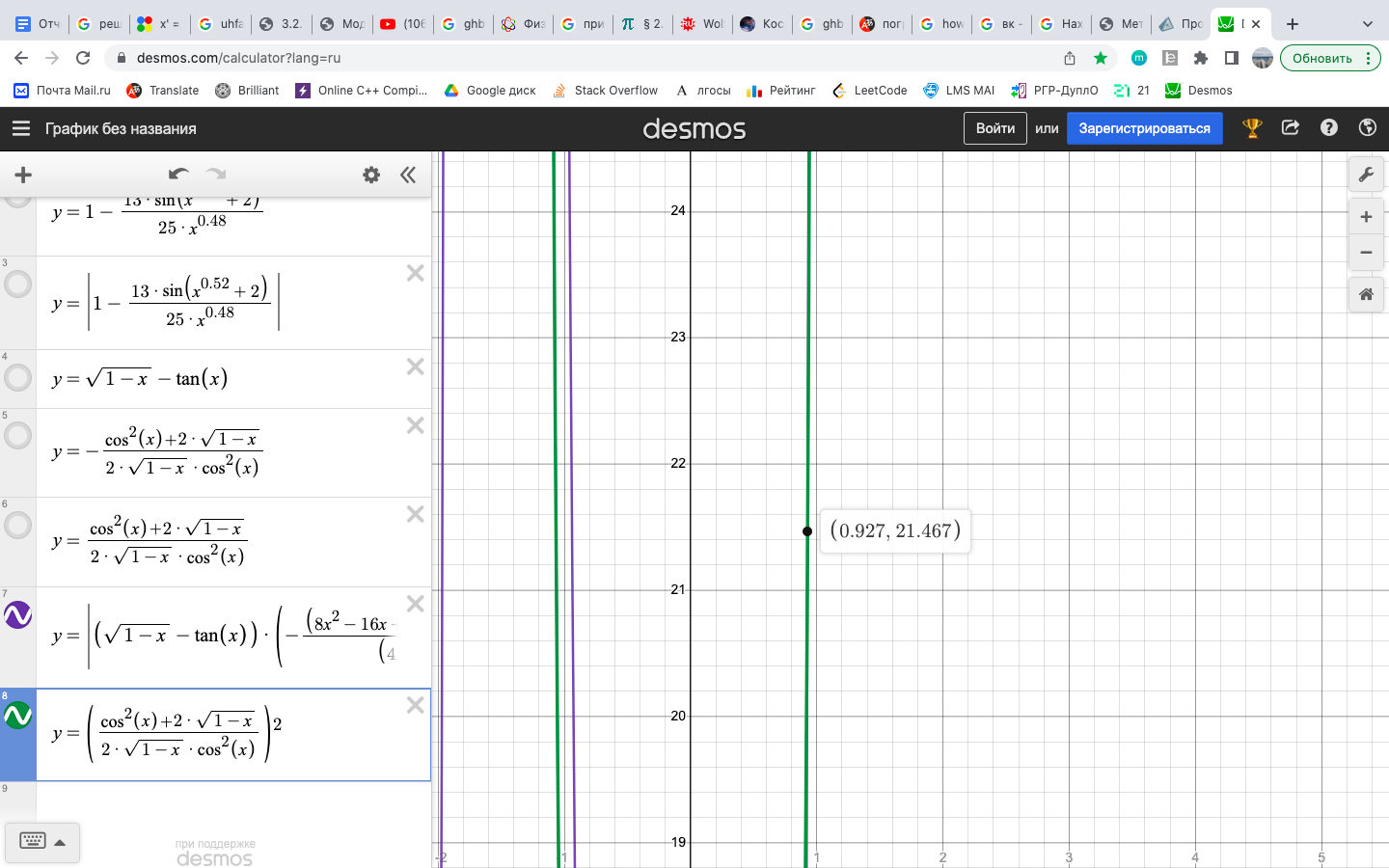
График функции :





График производной функции под модулем:

****Графики (фиолетовый) и (зеленый):

****

**Имеем:**

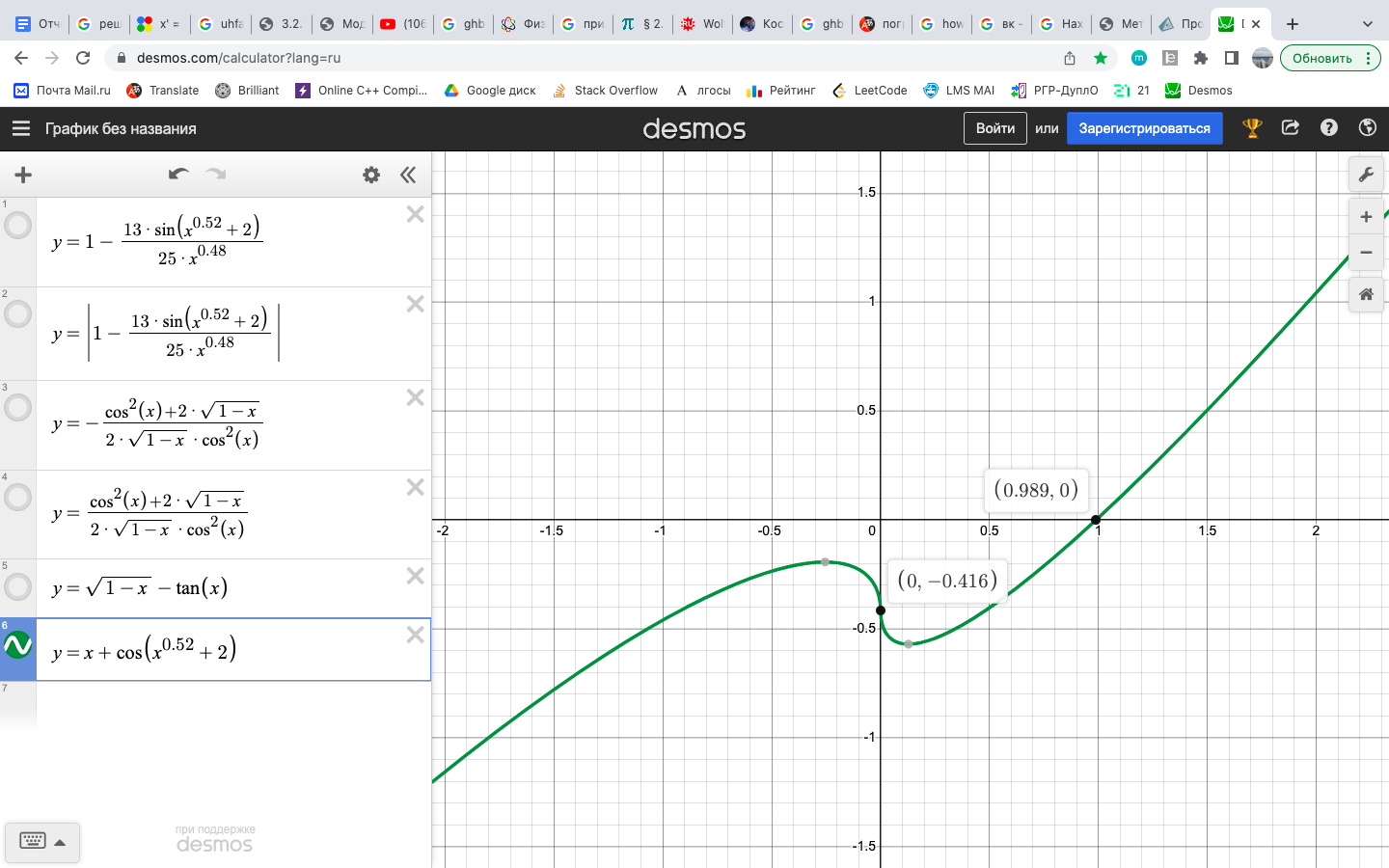
*Метод дихотомии* **применим** к решению данного уравнения, поскольку из графика функции видно, что выполняется условие .

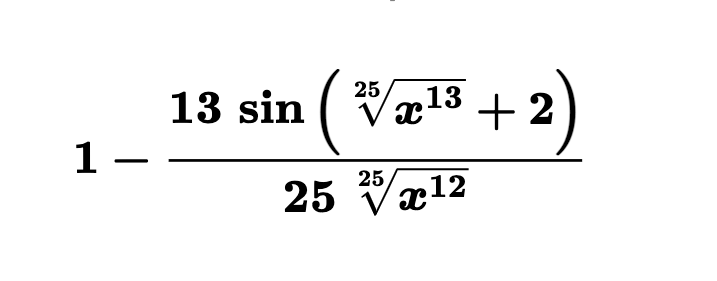
*Метод итераций* **не применим** к решению данного уравнения, поскольку существую значения модуля производной функции на отрезке, содержащем корень.

*Метод Ньютона* **применим**, поскольку корень данного уравнения () лежит в области выполнения условия .

*Функция из варианта №6*

График функции :





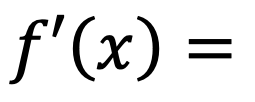
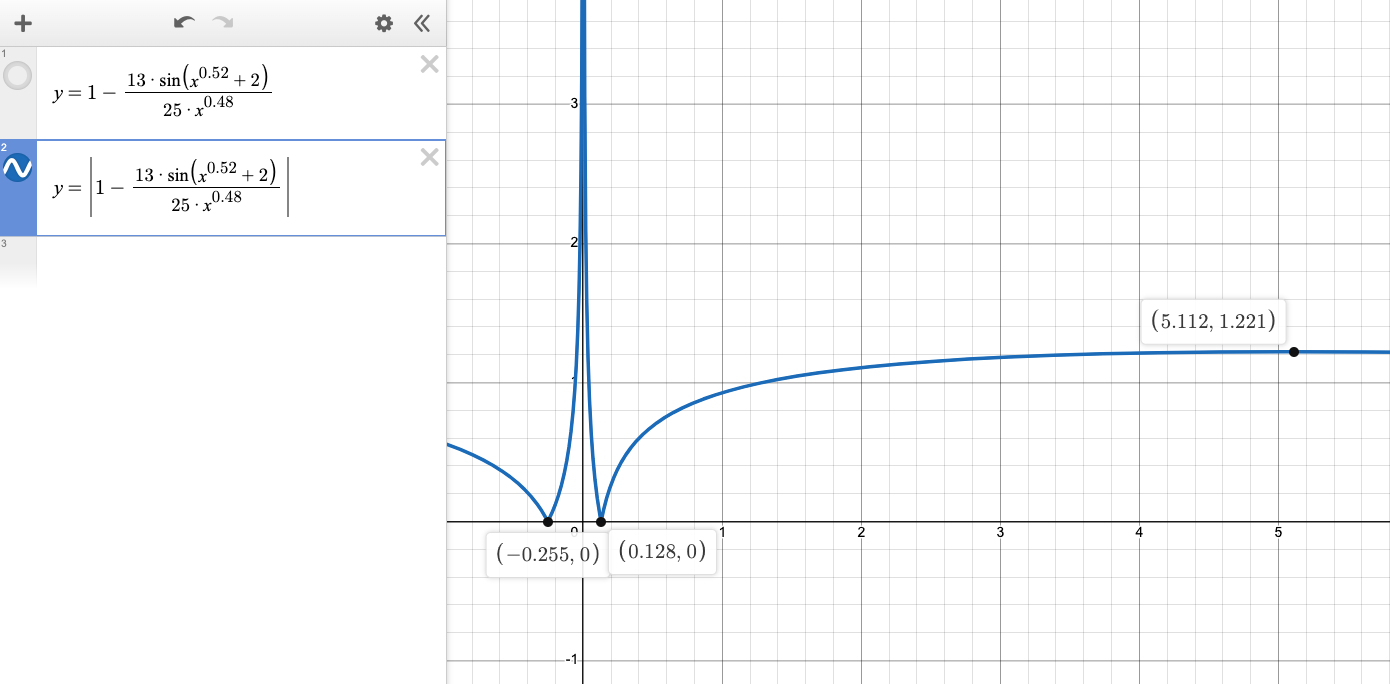
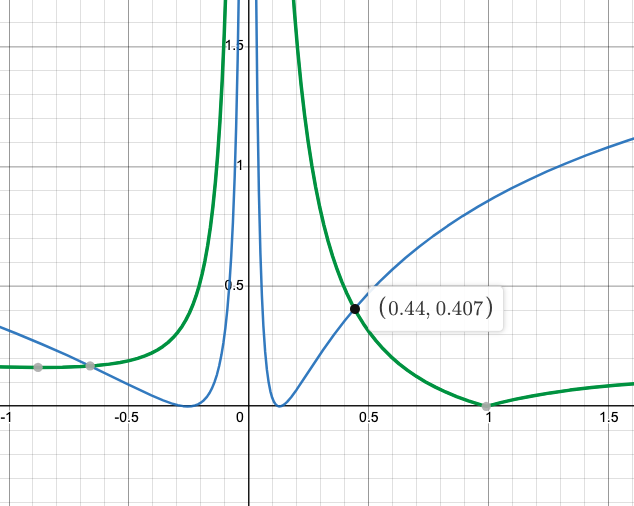


График производной под модулем:

****

Графики функций :

****

**Имеем:**

*Метод дихотомии* **применим** к решению данного уравнения, поскольку из графика функции видно, что выполняется условие .

*Метод итераций* **применим** к решению данного уравнения, поскольку выполнимо условие на отрезке, содержащем корень.

*Метод Ньютона* **применим**, поскольку условие выполнимо на всем отрезке .

## **Таблица используемых переменных**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| cnt | int | счетчик итераций |
| a1 | long double | начало отрезка первой функции |
| b1 | long double | конец отрезка первой функции |
| c | long double | промежуточное значение |
| EPS | long double | точность вычисления корней |
| a | long double | начало отрезка |
| b | long double | конец отрезка |
| flag | bool | контроль выхода из цикла |
| a2 | long double | начало отрезка первой функции |
| b2 | long double | конец отрезка первой функции |

## **Таблица используемых функций**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Тип** | **Описание** |
| fi\_function |  | первая функция |
| se\_function |  | вторая функция |
| fi\_iter\_func |  | первая функция вида f(x) = x |
| se\_iter\_func | long double | вторая функция вида f(x) = x |
| fi\_derivative |  | производная первой функции |
| se\_derivative |  | производная второй функции |
| fi\_dichotomy |  | метод дихотомии для первой функции |
| se\_dichotomy |  | метод дихотомии для второй функции |
| fi\_iter | void | метод итераций для первой функции |
| se\_iter |  | метод итераций для второй функции |
| fi\_newton |  | метод Ньютона для первой функции |
| se\_newton |  | метод Ньютона для второй функции |

## **Листинг программного кода**

#include <math.h>

#include <stdio.h>

#include <stdbool.h>

long double fi\_function(long double a) {

return sqrt(1 - a) - tan(a);

}

long double se\_function(long double a) {

return a + cos(pow(a, 0.52) + 2);

}

long double fi\_iter\_func(long double a) {

return 1 - tan(a) \* tan(a); // запись 1 функции вида: f(a) = a

}

long double se\_iter\_func(long double a) {

return -cos(pow(a, 0.52) + 2); // запись 2 функции вида: f(a) = a

}

long double fi\_derivative(long double a) { // производная от первой функции

return -(cos(a) \* cos(a) + 2 \* sqrt(1 - a))/(2 \* sqrt(1 - a) \* cos(a) \* cos(a));

}

long double se\_derivative(long double a) { // производная от второй функции

return 1 - (13 \* sin(pow(a, 0.52) + 2))/(25 \* pow(a, 0.48));

}

void fi\_dichotomy(long double a, long double b, long double EPS) {

bool flag = true;

long double c = 0;

int cnt = 0;

while (fabs(a - b) > EPS && flag) {

c = (a + b) / 2; // middle point

if (fi\_function(c) == 0) {

break;

}

else {

if (fi\_function(c) \* fi\_function(a) < 0) {

b = c;

}

else if (fi\_function(b) \* fi\_function(c) < 0) {

a = c;

}

cnt++;

}

if (cnt > 100) {

flag = false;

}

}

if (!flag) {

printf("Метод дихотомии не применим. \n");

}

else {

printf("Метод дихотомии: %.20Lf; Кол-во итераций: %d \n", c, cnt);

}

}

void se\_dichotomy(long double a, long double b, long double EPS) {

bool flag = true;

long double c = 0;

int cnt = 0;

while (fabs(a - b) > EPS && flag) {

c = (a + b) / 2; // middle point

if (se\_function(c) == 0 && flag) {

break;

}

else {

if (se\_function(c) \* se\_function(a) < 0) {

b = c;

}

else if (se\_function(b) \* se\_function(c) < 0) {

a = c;

}

}

cnt++;

if (cnt > 100) {

flag = false;

}

}

if (!flag) {

printf("Метод дихотомии не применим. \n");

}

else {

printf("Метод дихотомии: %.20Lf; Кол-во итераций: %d \n", c, cnt);

}

}

void fi\_iter(long double a, long double b, long double EPS) {

bool flag = true;

long double c;

int cnt = 0;

while (fabs(b - a) > EPS && flag) {

c = fi\_iter\_func(b);

b = a;

a = c;

cnt++;

if (cnt > 100) {

flag = false;

break;

}

}

if (!flag) {

printf("Метод итераций не применим. \n");

}

else {

printf("Метод итераций: %.20Lf; Кол-во итераций: %d \n", a, cnt);

}

}

void se\_iter(long double a, long double b, long double EPS) {

bool flag = true;

long double c;

int cnt = 0;

while (fabs(b - a) > EPS && flag) {

c = se\_iter\_func(b);

b = a;

a = c;

cnt++;

if (cnt > 100) {

flag = false;

break;

}

}

if (!flag) {

printf("Метод итераций не применим. \n");

}

else {

printf("Метод итераций: %.20Lf; Кол-во итераций: %d \n", a, cnt);

}

}

void fi\_newton(long double a, long double b, long double EPS) {

bool flag = true;

int cnt = 0;

long double c;

while (fabs(b - a) > EPS && flag) {

c = a - (fi\_function(a) / fi\_derivative(a));

b = a;

a = c;

cnt++;

if (cnt > 100) {

flag = false;

}

}

if (!flag) {

printf("Метод Ньютона не применим. \n");

}

else {

printf("Метод Ньютона: %.20Lf; Кол-во итераций: %d \n", a, cnt);

}

}

void se\_newton(long double a, long double b, long double EPS) {

bool flag = true;

int cnt = 0;

long double c;

while (fabs(b - a) > EPS && flag) {

c = b - (se\_function(b) / se\_derivative(b));

b = a;

a = c;

cnt++;

if (cnt > 100) {

flag = false;

}

}

if (!flag) {

printf("Метод Ньютона не применим. \n");

}

else {

printf("Метод Ньютона: %.20Lf; Кол-во итераций: %d \n", a, cnt);

}

}

int main(void) {

long double EPS = 1.0;

while (1.0 + EPS / 2.0 > 1) {

EPS /= 2;

}

long double a1 = 0, b1 = 1;

long double a2 = 0.5, b2 = 1;

printf("Результаты вычислений для 1 функции: \n");

fi\_dichotomy(a1, b1, EPS);

fi\_newton(a1, b1, EPS);

fi\_iter(a1, b1, EPS);

printf("\n");

printf("Результаты вычислений для 2 функции: \n");

se\_dichotomy(a2, b2, EPS);

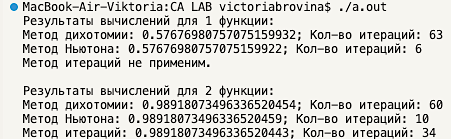
se\_newton(a2, b2, EPS);

se\_iter(a2, b2, EPS);

return 0;

}

## **Выходные данные**

****

# **Вывод**

В ходе работы над заданием мы рассмотрели три численных метода решения трансцендентных уравнений: метод дихотомии (половинного деления), метод итераций, метод Ньютона. Я составила программу на языке Си с процедурами решения трансцендентных алгебраических уравнений. Используя графики функций, производные функции первого и второго порядка, я доказала применимость и неприменимость тех или иных численных методов к каждому из двух данных уравнений.

# **Список литературы**

* [**https://www.desmos.com/calculator?lang=ru**](https://www.desmos.com/calculator?lang=ru)
* [**https://en.cppreference.com/w/**](https://en.cppreference.com/w/)
* [**https://core.ac.uk/download/pdf/162891844.pdf**](https://core.ac.uk/download/pdf/162891844.pdf)