Московский Авиационный Институт

(Национальный Исследовательский Университет)

Курсовая работа

по курсу

«Вычислительные системы»

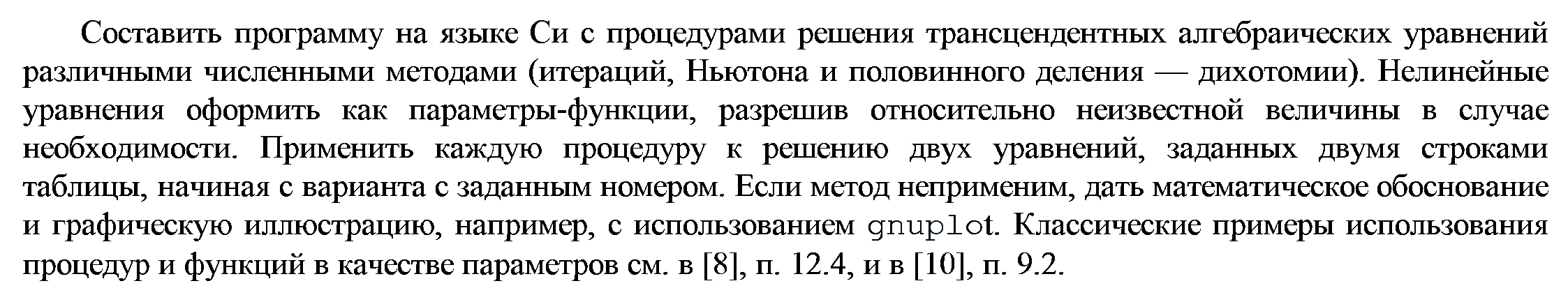
1 семестр

Задание 4

«Процедуры и функции в качестве параметров»

|  |  |
| --- | --- |
| Студент: | Жилов А.А. |
| Группа: | М8О-105Б-21 |
| Руководитель: | Титов В.К. |
| Оценка: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| Дата: | 28.12.21 |
| Подпись: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

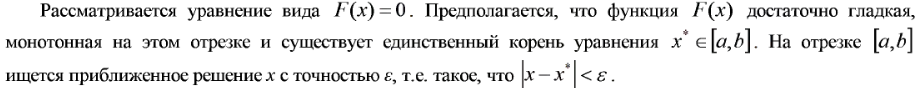
**Введение**



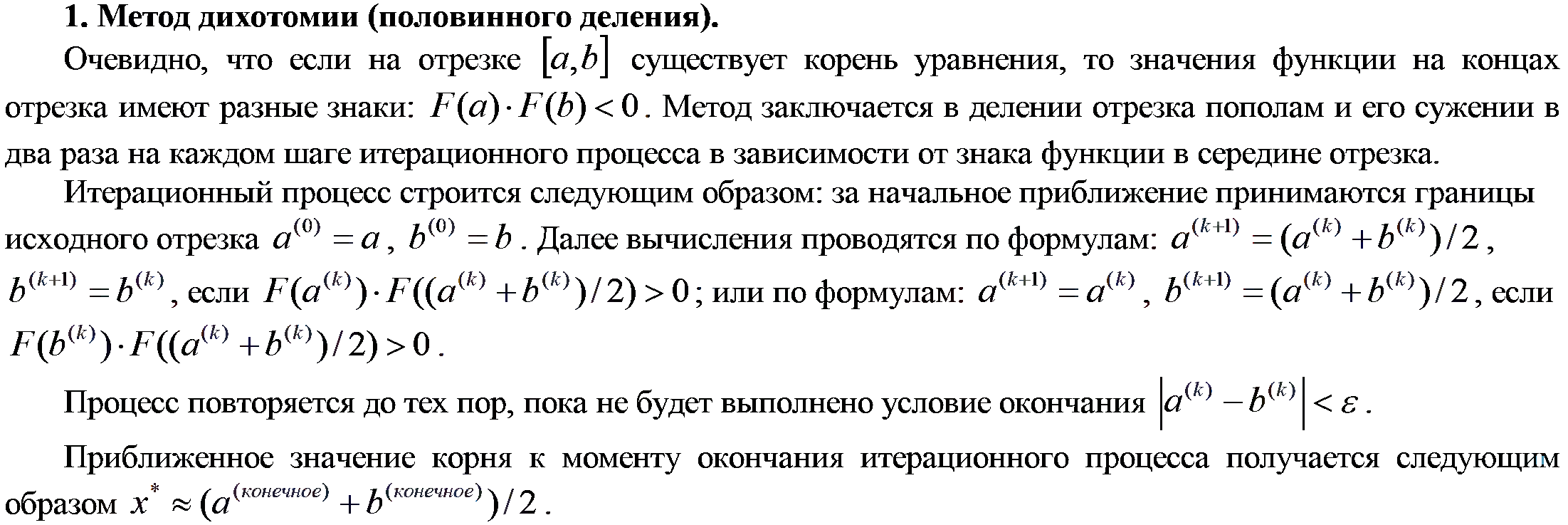
**Задание**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Обозначение функций в программе | Вариант | Функция | Отрезок | Метод | Приближенное значение корня |
| f1 | 4 |  | [1;3] | все | 2.0692 |
| f2 | свой |  | [4;5] | все | 4.4987 |
| f3 | свой |  | [4;5] | все | 4.4557 |

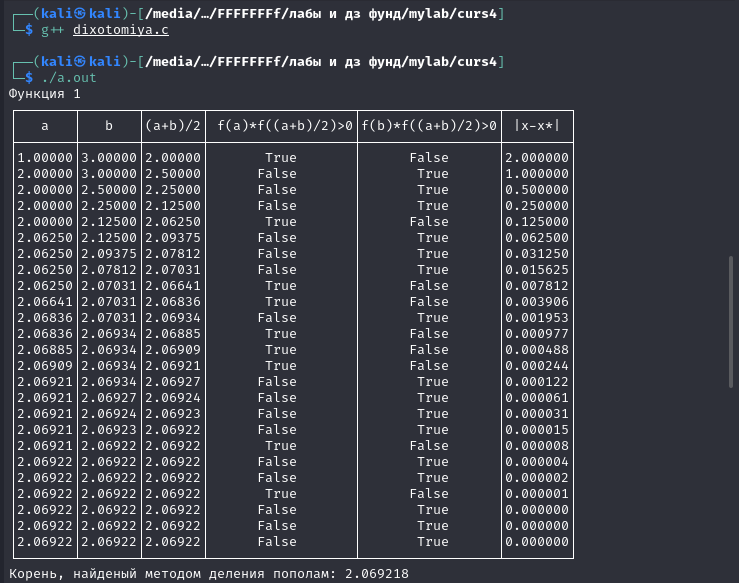
**Описание решения**

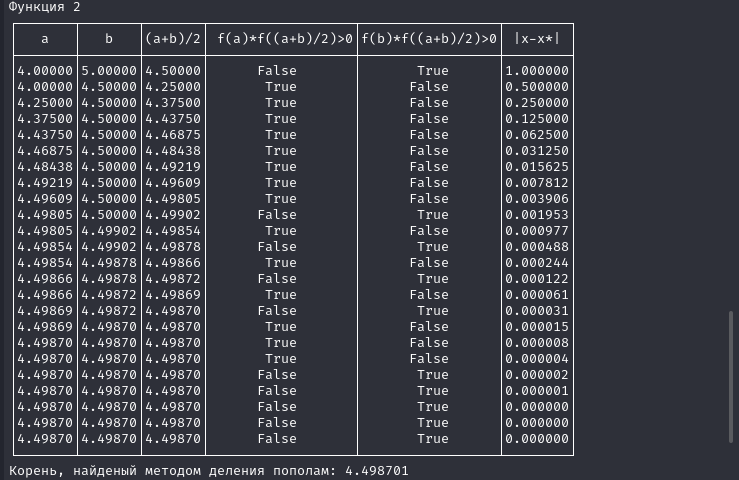


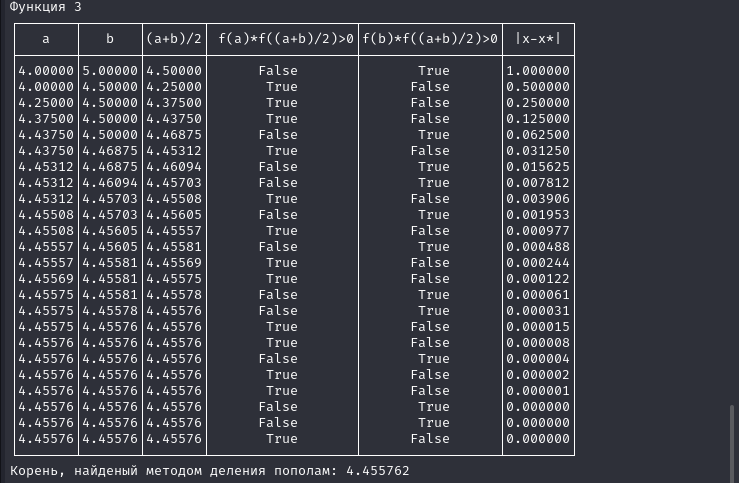
**Описание методов нахождения корня**

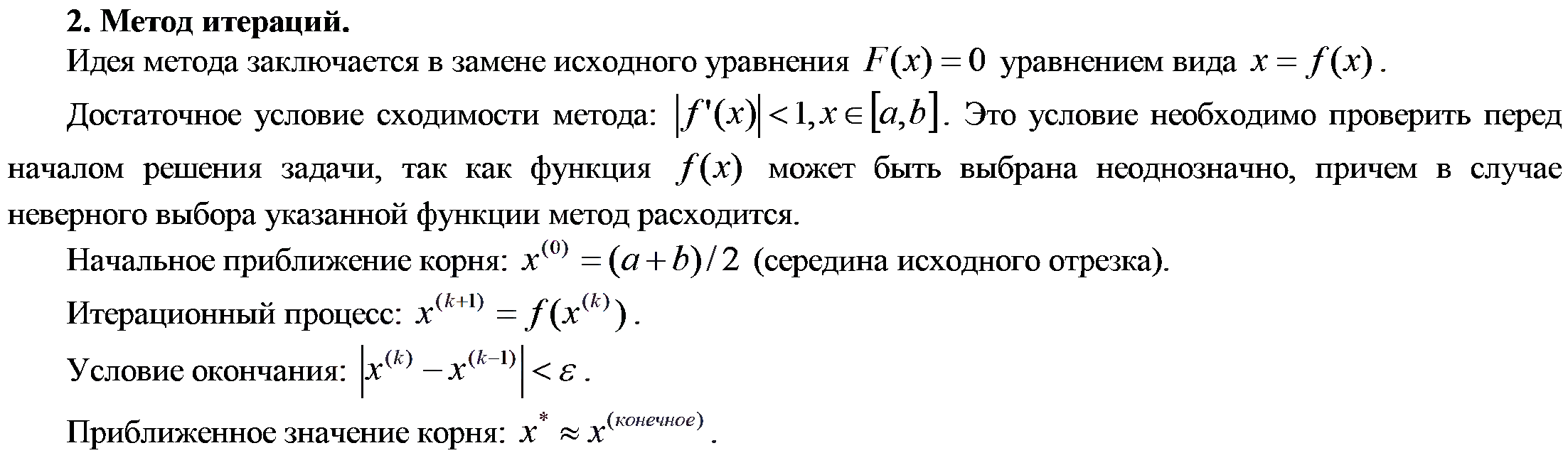








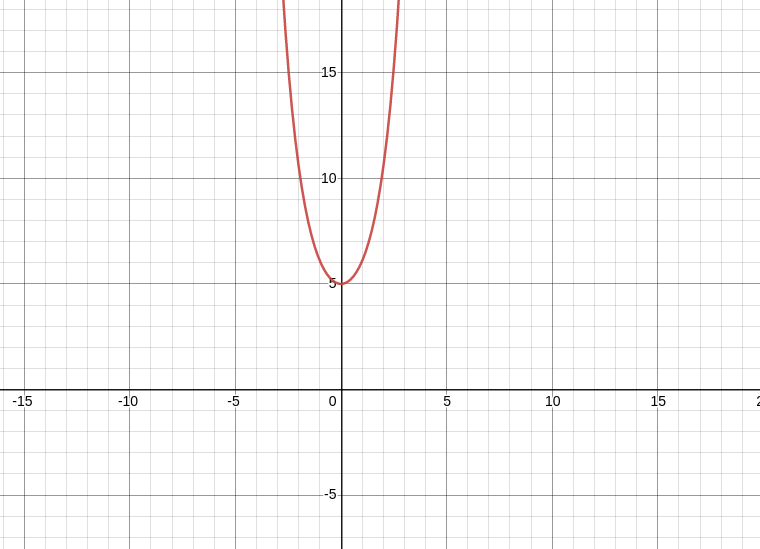




**Проверка сходимости метода итерации для функции f1(x)=**

f1′(x)=f1\_p(x)=

График f1′(x):



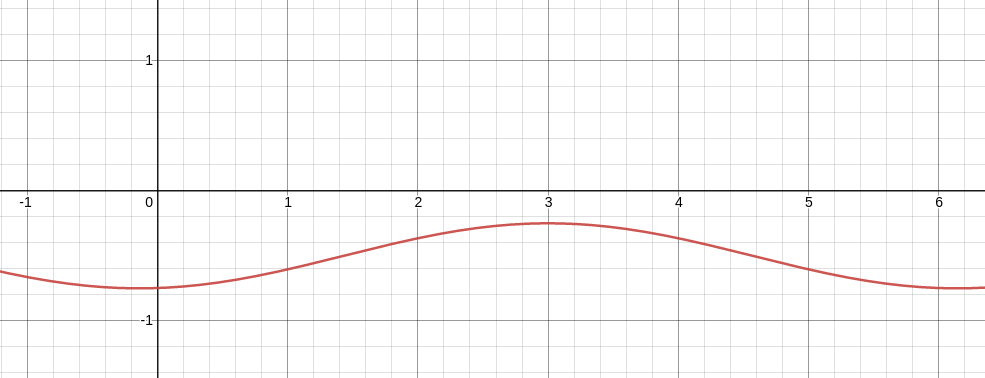
При ⁡x ∈ [1;3] |f1′(x)|>1

**Следовательно достаточное условие сходимости метода итераций не выполнено.**

**Проверка сходимости метода итерации для функции f2(x)=**

f2′(x)=f2\_p(x)=

График f2′(x):



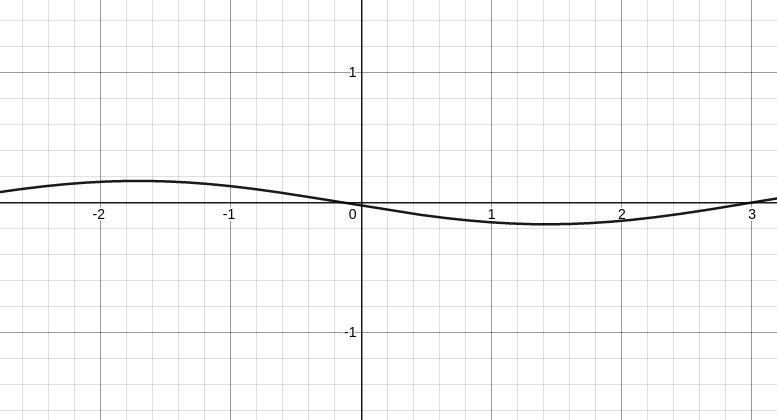
При ⁡(x ∈ [4;5]) |f2′(x)|<1

**Следовательно достаточное условие сходимости метода итераций выполнено.**

Заменяем уравнение вида f2(x)=0 уравнением типа x=f2\_x(x).

x=f2\_x(x)=

График f2\_x(x):



**Проверка сходимости метода итерации для функции f3(x)=**

f3′(x)=f3\_p(x)=

График f3′(x):



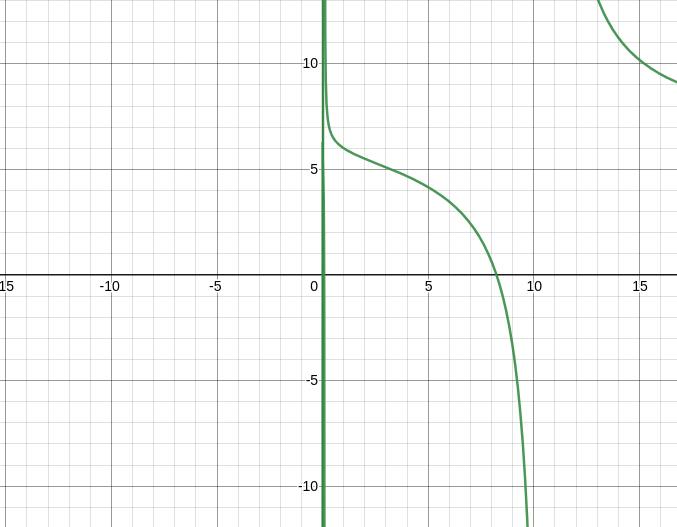
При ⁡x ∈ [2; 3] |f3′(x)|<1

**Следовательно достаточное условие сходимости метода итераций выполнено.**

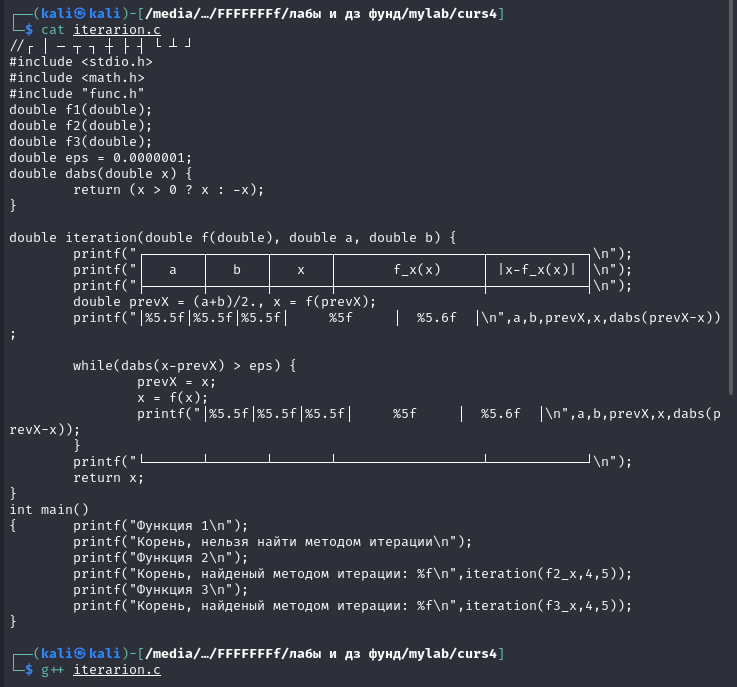
Заменяем уравнение вида f3(x)=0 уравнением типа x=f3\_x(x).

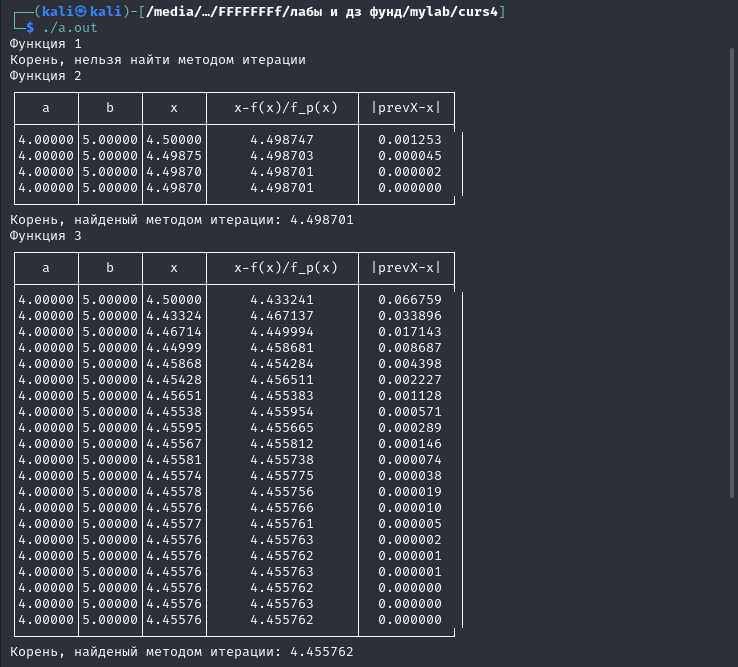
x=f3\_p(x)=

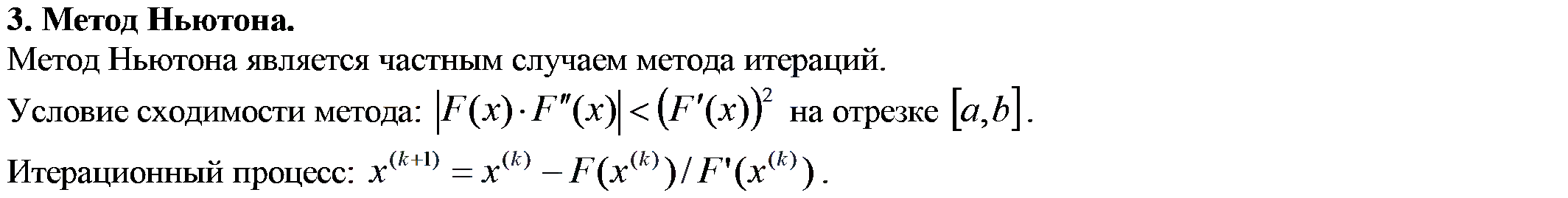
График f3\_p(x):



**Нахождение корня с помощью метода итераций:**





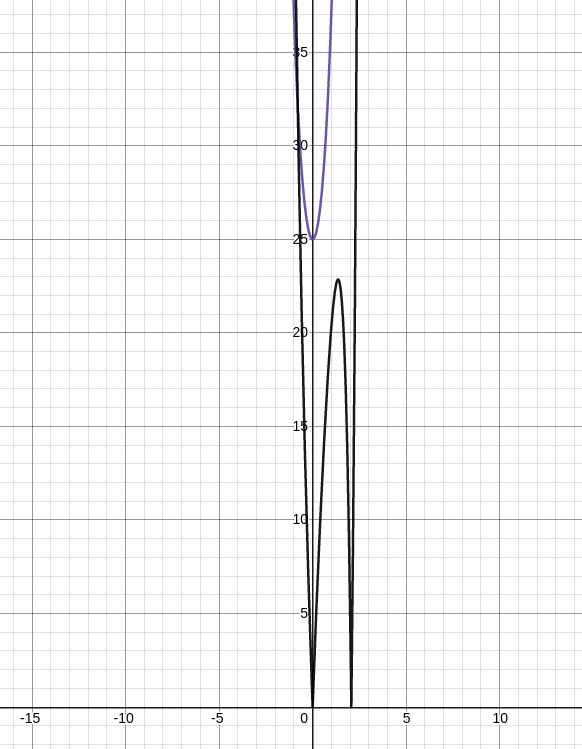


**Проверка сходимости метода Ньютона для функции f1(x)=**

f1′(x)=f1\_p(x)=

f1 ′′ (x)=

**График функции |f1(x) ∗ f1 ′′ (x)|<(f1′(x)) 2:**



**неравенство: |f1(x) ∗ f1 ′′ (x)| < при x ∈ [1;3] выполняется,**

**значит метод Ньютона сходится.**

**Проверка сходимости метода Ньютона для функции f2(x)=**

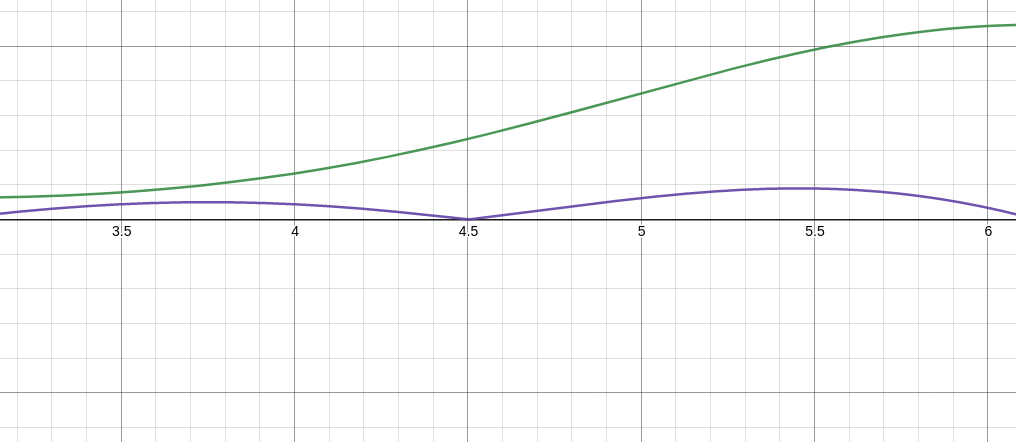
f2′(x)=f2\_p(x)=

=

=

|f2(x) ∗ f2 ′′ (x)|=

**График функции|f2(x) ∗ f2 ′′ (x)| < :**



**неравенство: |f2(x) ∗ f2 ′′ (x)| < при x ∈ [4;5] выполняется,**

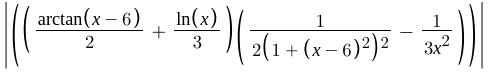
**значит метод Ньютона сходится.**

**Проверка сходимости метода Ньютона для функции f3(x)=**

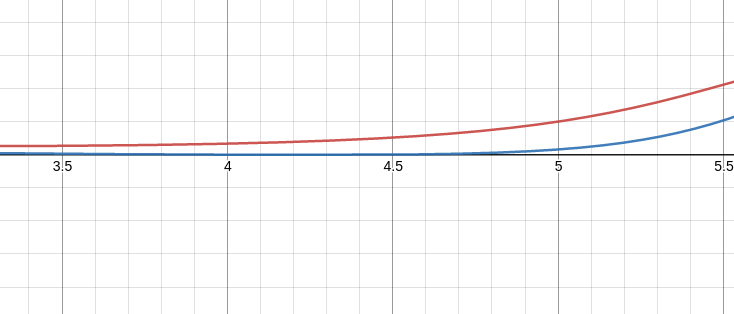
f3′(x)=f3\_p(x)=

=

=

|f3(x) ∗ f3′′ (x)|=

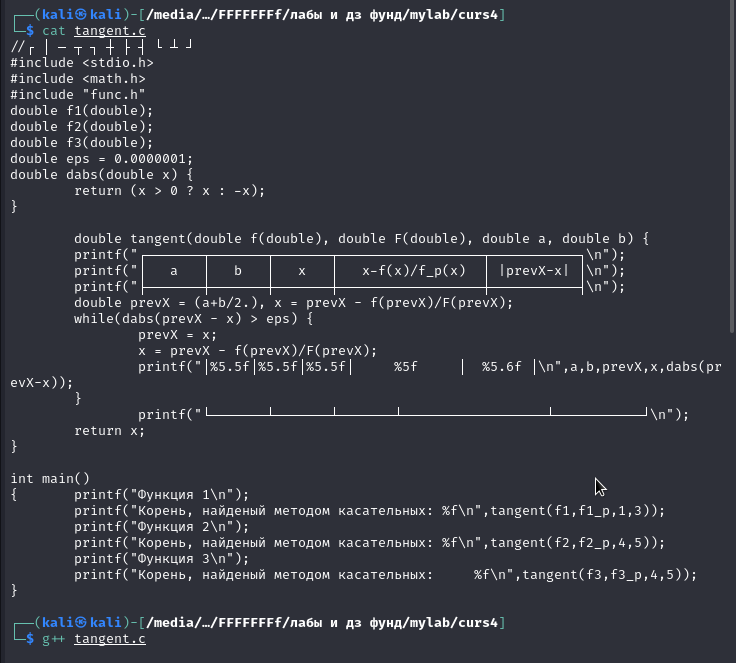
**График функции|f3(x) ∗ f3 ′′ (x)| < :**

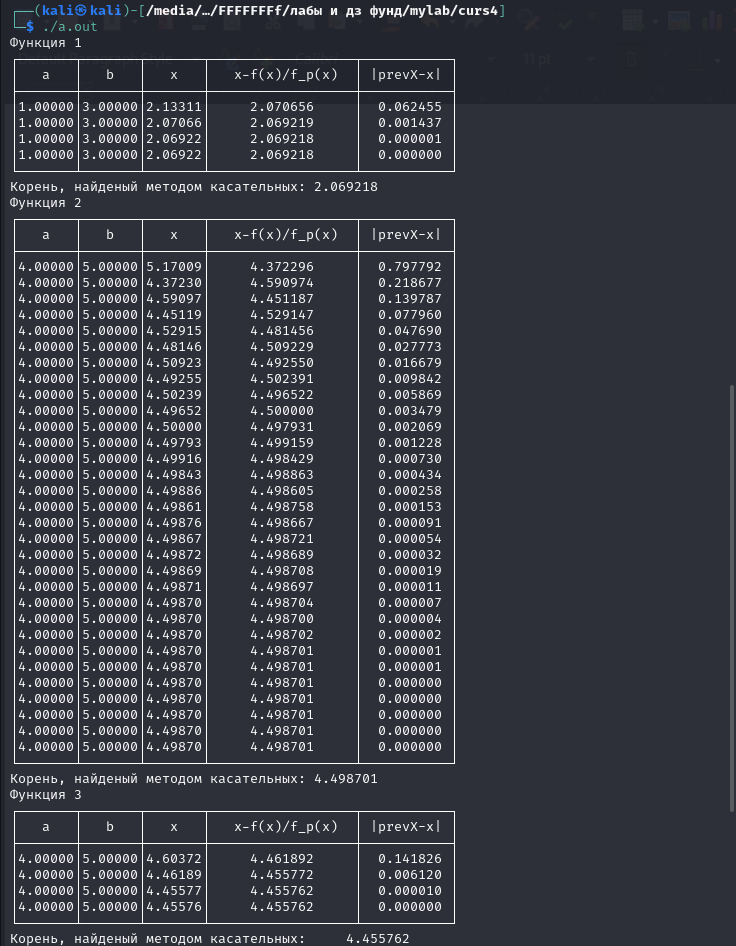


**неравенство: |f3(x) ∗ f3 ′′ (x)| < при x ∈ [4;5] выполняется,**

**значит метод Ньютона сходится.**

**Нахождение корня с помощью метода Ньютона для функций**





**4. Метод хорд**

Суть метода хорд состоит в разбиении отрезка **[a; b]** (при условии **f(a)f(b) < 0**) на два отрезка с помощью хорды и выборе нового отрезка от точки пересечения хорды с осью абсцисс до неподвижной точки, на котором функция меняет знак и содержит решение, причём подвижная точка приближается к **ε**-окрестности решения.

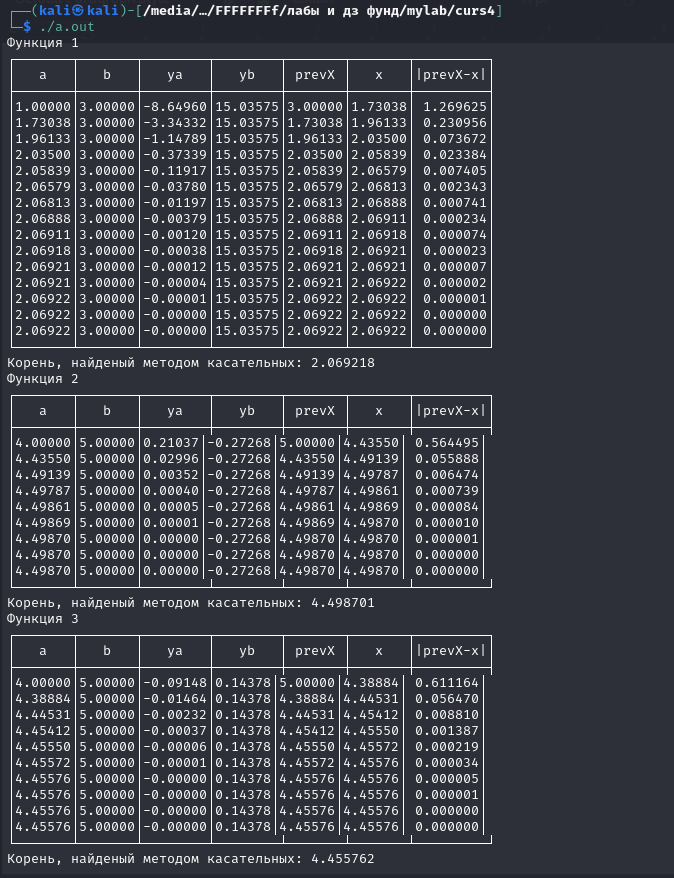
Входные данные: **f(x), f’’(x), a, b, ε**.

1. *Если* **f(a) · f’’(a) > 0**, *то* **c = a**, *иначе если* **f(b) · f’’(b) > 0**, *то* **c = b**.
2. *Если* **f(a) · f’’(a) < 0**, *то* **x = a**, *иначе если* **f(b) · f’’(b) < 0**, *то* **x = b**.
3. **Δx = f(x) · (x − c) / (f(x) − f(c))**.
4. **x = x — Δx**.
5. *Если* **|Δx| > ε**, *то идти к* 3.

Выходные данные: **x**. Значение **x** является решением с заданной точностью **ε** нелинейного уравнения вида **f(x) = 0**. Если **f(x) = 0**, то **x** — точное решение.

**Нахождение корня с помощью метода хорд для функций**



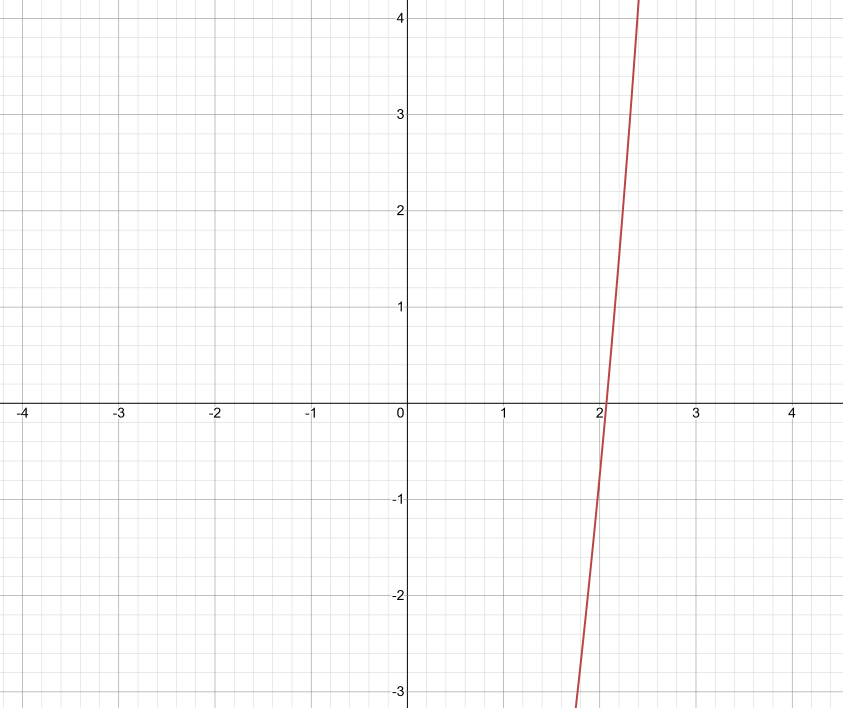


**Реализация програмной алгоритм**

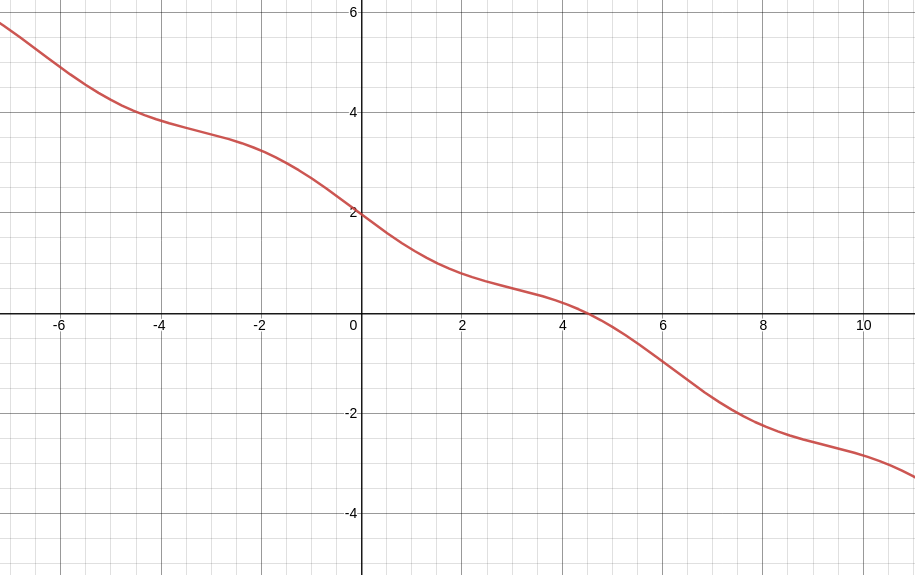
Сначала мы определяем все функции, которые используются в коде. Далее выводи на экран значения корня каждой функции, каждым из 4 методов. Где f-функция, f\_x-функция вида x=f(x), f\_p- производная функции.

**Графики функций**

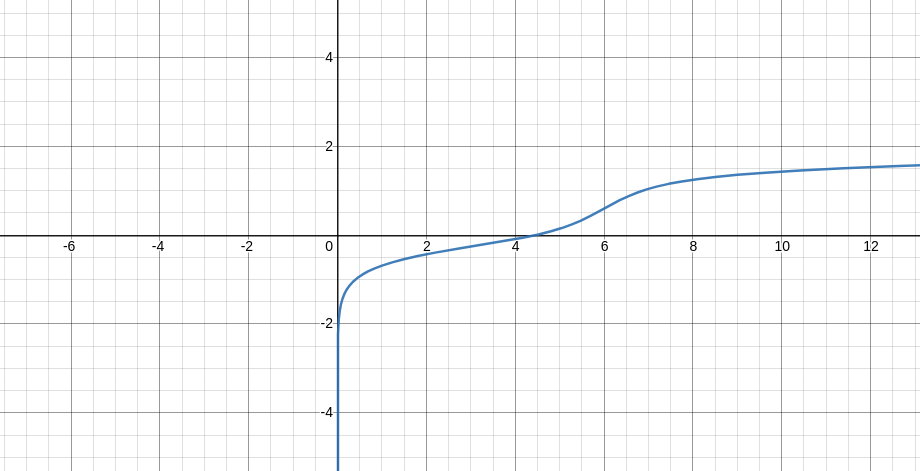
**1)** 



**2)** 



**3)**



**Протокол**

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/curs4]

└─$ cat curs4.c

#include <math.h>

#include <stdio.h>

#include "locale.h"

#include "func.h"

double dabs(double);

double dichotomy(double f(double), double, double);

double iteration(double f(double), double, double);

double tangent(double f(double), double F(double), double, double);

double chord(double f(double), double, double);

double f1(double);

double f1\_x(double);

double f1\_p(double);

double f2(double);

double f2\_x(double);

double f2\_p(double);

double f3(double);

double f3\_x(double);

double f3\_p(double);

int main()

{

printf("Корень для функции f1\n");

printf("Корень, найденый методом деления пополам: %f\n",dichotomy(f1,1,3));

printf("Корень, найденый методом итераций: не удовлетворяет условию достаточности\n");

printf("Корень, найденый методом касательных: %f\n",tangent(f1,f1\_p,1,3));

printf("Корень, найденый методом хорд: %f\n",chord(f1,1,3));

printf("Корень для функции f2\n");

printf("Корень, найденый методом деления пополам: %f\n",dichotomy(f2,4,5));

printf("Корень, найденый методом итераций: %f\n",iteration(f2\_x,4,5));

printf("Корень, найденый методом касательных: %f\n",tangent(f2,f2\_p,4,5));

printf("Корень, найденый методом хорд: %f\n",chord(f2,4,5));

printf("Корень для функции f3\n");

printf("Корень, найденый методом деления пополам: %f\n",dichotomy(f3,4,5));

printf("Корень, найденый методом итераций: %f\n",iteration(f3\_x,4,5));

printf("Корень, найденый методом касательных: %f\n",tangent(f3,f3\_p,4,5));

printf("Корень, найденый методом хорд: %f\n",chord(f3,4,5));

}

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/curs4]

└─$ cat func.h

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*f1\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double f1(double x)

{

return 3\*x-14+pow(exp(1.),x)-pow(exp(1.),-x);

}

double f1\_p(double x)

{

return 3+pow(exp(1.),x)+pow(exp(1.),-x);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*f2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double f2(double x)

{

return sin(x-3)/4-(x-4)/2;

}

double f2\_x(double x)

{

return sin(x-3)/2+4;

}

double f2\_p(double x)

{

return cos(x)/4-(x-4)/2;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*f3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

double f3(double x)

{

return atan(x-6)/2+log(x)/3;

}

double f3\_x(double x)

{

return tan(-2\*log(x)/3)+6;

}

double f3\_p(double x)

{

return 1/(2\*(1+(x-6)\*(x-6)))+1/(3\*x);

}

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/curs4]

└─$ cat locale.h

double eps = 0.0000001;

double dabs(double x) {

return (x > 0 ? x : -x);

}

double dichotomy(double f(double), double a, double b) {

double prevX = b, x = (a + b) / 2.;

while(dabs(prevX - x) > eps) {

if(f(x)\*f(a) > 0)

a = x;

else

b = x;

prevX = x;

x = (a + b) / 2.;

}

return x;

}

double iteration(double f(double), double a, double b) {

double prevX = (a+b)/2., x = f(prevX);

while(dabs(x-prevX) > eps) {

prevX = x;

x = f(x);

}

return x;

}

double tangent(double f(double), double F(double), double a, double b) {

double prevX = (a+b/2.), x = prevX - f(prevX)/F(prevX);

while(dabs(prevX - x) > eps) {

prevX = x;

x = prevX - f(prevX)/F(prevX);

}

return x;

}

double chord(double f(double), double a, double b) {

double prevX = b, ya = f(a), yb = f(b);

double x = (ya\*b-yb\*a)/(ya-yb);

while(dabs(prevX - x) > eps) {

if(ya\*f(x) > 0)

a = x;

else

b = x;

ya = f(a), yb = f(b);

prevX = x;

x = (ya\*b-yb\*a)/(ya-yb);

}

return x;

}

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/curs4]

└─$ g++ curs4.c

┌──(kali㉿kali)-[/media/…/FFFFFFFf/лабы и дз фунд/mylab/curs4]

└─$ ./a.out

Корень для функции f1

Корень, найденый методом деления пополам: 2.069218

Корень, найденый методом итераций: не удовлетворяет условию достаточности

Корень, найденый методом касательных: 2.069218

Корень, найденый методом хорд: 2.069218

Корень для функции f2

Корень, найденый методом деления пополам: 4.498701

Корень, найденый методом итераций: 4.498701

Корень, найденый методом касательных: 4.498701

Корень, найденый методом хорд: 4.498701

Корень для функции f3

Корень, найденый методом деления пополам: 4.455762

Корень, найденый методом итераций: 4.455762

Корень, найденый методом касательных: 4.455762

Корень, найденый методом хорд: 4.455762

**Дневник отладки**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Дата | Время | Событие | Действие по исправлению |
| 1 | 28.12.21 | 18:00 | Написал arctan(x-6) | Исправил на atan(x-6) |

**Заключение**

В ходе выполнения была написана программа для нахождения приближенного значения

корней трех разных функций четырьмя способами для каждой: метод Дихотомии, метод

итераций, метод Ньютона(касательных), метод хорд.