МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

**Факультет** экономики, менеджмента и информационных технологий

**Кафедра** систем управления и информационных технологий в строительстве

**Отчет по лабораторной работе**

### Тема: «Решение нелинейных алгебраических уравнений методом деления отрезка пополам, методом хорд, методом простых итераций, методом касательных, методом секущих, комбинированным методом хорд и касательных.»

По дисциплине: Информатика

Выполнил студент: Курганников Р.А.

Группа: № БИСТ-225

Вариант: 12

Руководитель: доц. Ефимова О. Е.

Работа защищена « »\_\_\_\_\_\_\_\_\_2023г.

С оценкой \_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_

Воронеж 2023

**Постановка задачи**

### Условие задачи:

* Найти все интервалы изоляции корней заданного уравнения, и корни на найденных отрезках
* Каждый метод должен быть оформлен в виде процедуры
* В программе должен осуществляться выбор метода нахождения приближенного корня
* Составить алгоритм и блок-схему метода секущих
* Объяснить, чем метод секущих отличается от метода касательных, и что у них общего
* Составить алгоритм комбинированного метода хорд и касательных
* Объяснить специфику применения комбинированного метода хорд и касательных

**Методы**:

Метод деления отрезка пополам (дихотомии)

Метод хорд

Метод Ньютона (метод касательных)

Метод последовательных приближений (метод простых итераций)

Метод секущих

Комбинированный метод хорд и касательных

**Исходные данные**:

Нелинейное алгебраическое уравнение, корни которого требуется найти и уточнить, используя различные методы:

f(x) = 3x4+4x3-12x2+1

**Код программы**

#include<iostream>

#include<cmath>

#include<iomanip>

#define EXP 2.71828182

using namespace std;

double fun(double x) {

return 3\*pow(x,4)+4\*pow(x,3)-12\*pow(x,2)+1;

}

double fun1(double x) {

return 12\*pow(x,3)+12\*pow(x,2)-24\*x;

}

double fun2(double x) {

return 36\*pow(x,2)+24\*x-24;

}

double DivSegmentHalf(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

while (true) {

double c = (a + b) / 2;

double a\_values = fun(a);

double b\_values = fun(b);

double c\_values = fun(c);

if (fabs(c\_values) < eps) return c;

if (a\_values \* c\_values < 0) b = c;

else a = c;

}

}

double DivHord(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

double c = 0;

while (fabs(fun(b) - fun(a)) > eps) {

c = (fun(b) \* a - fun(a) \* b) / (fun(b) - fun(a));

if ((fun(a) \* fun(c)) > 0) a = c;

else b = c;

}

return c;

}

double DivNuton(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

int n = 0;

double x = 0;

if (fun(a) \* fun1(a) < 0) x = a;

else x = b;

double counter = 0;

counter = fabs(fun1(x));

while (counter > eps)

{

x = x - (fun(x) / fun1(x));

n += 1;

counter--;

}

return x;

}

double DivSucApp(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

double x = a;

for (double iter = 1; eps < fabs(fun(x)); ++iter)

{

if (fun(x) \* fun(fun1(x)) > 0) x = fun1(x);

else break;

}

return x;

}

double DivHordKos(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

do {

if (fun(a) \* fun2(a) < 0) a += (b - a) / (fun(a) - fun(b)) \* fun(a);

else a -= fun(a) / fun1(a);

if (fun(b) \* fun2(b) < 0) b += (a - b) / (fun(b) - fun(a)) \* fun(b);

else b -= fun(b) / fun1(b);

} while (fabs(a - b) > 2 \* eps);

return (a + b) / 2.0;

}

double DivSek(double a, double b, double eps) {

if (fun(a) \* fun(b) >= 0) throw false;

double res = 0, y = 0;

do

{

y = res;

res = b - ((b - a) / (fun(b) - fun(a))) \* fun(b);

a = b;

b = res;

} while (fabs(y - res) >= eps);

return res;

}

int main() {

setlocale(LC\_ALL, "RUS");

cout << "Task 1\n" << endl;

bool flag = true;

while (flag) {

int button = 0;

cout << "1. Метод деления отрезка пополам (дихотомии)" << endl;

cout << "2. Методхорд" << endl;

cout << "3. Метод Ньютона (метод касательных)" << endl;

cout << "4. Метод последовательных приближений (метод простых итераций)" << endl;

cout << "5. Метод секущих" << endl;

cout << "6. Комбинированный метод хорд и касательных" << endl;

cout << "0. Выход" << endl;

cout << "Выбор: ";

cin >> button;

cout << endl;

double (\*fcnPtr)(double, double, double) = nullptr;

switch (button) {

case 1: fcnPtr = DivSegmentHalf; break;

case 2: fcnPtr = DivHord; break;

case 3: fcnPtr = DivNuton; break;

case 4: fcnPtr = DivSucApp; break;

case 5: fcnPtr = DivSek; break;

case 6: fcnPtr = DivHordKos; break;

default: flag = false; break;

}

if (!flag) break;

double a\_1 = 0, b\_1 = 0, err\_1 = 0, y\_1 = 0, h\_1 = 0.1;

cout << "Enter a: ";

cin >> a\_1;

cout << "Enter b: ";

cin >> b\_1;

cout << "Enter err: ";

cin >> err\_1;

double min\_1 = 0, max\_1 = 0;

if (a\_1 < b\_1) {

min\_1 = a\_1;

max\_1 = b\_1;

}

else {

min\_1 = b\_1;

max\_1 = a\_1;

}

cout << endl << "-------------------------------------" << endl;

printf("| %15s | %15s |\n", "x", "f(x)");

cout << "-------------------------------------" << endl;

for (; min\_1 < max\_1; ) {

min\_1 += h\_1;

try

{

y\_1 = fcnPtr(min\_1, min\_1 + h\_1, err\_1);

printf("| %15e | %15e |\n", y\_1, fun(y\_1));

cout << "-------------------------------------" << endl;

}

catch (bool flag)

{

if (!flag) continue;

}

}

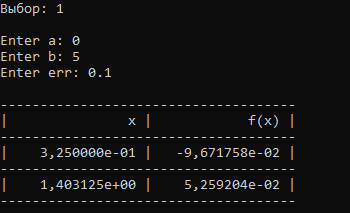
cout << endl;

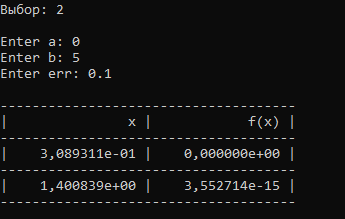
}

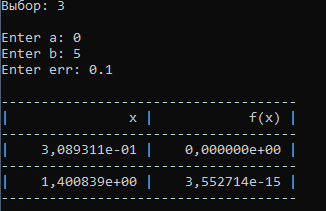
return 0;

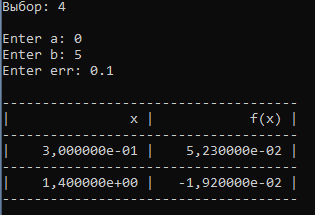
}

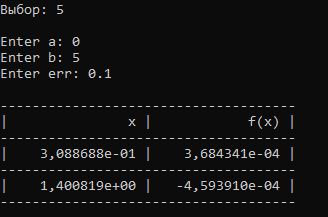
**Результаты работы программы**

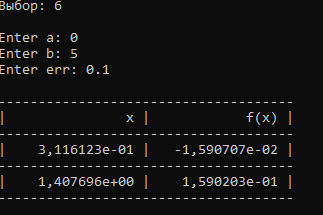












**Вывод:**

В ходе лабораторной работы, мы изучили численные методы уточнения корней уравнений: метод деления отрезка пополам (дихотомии), метод хорд, метод Ньютона (касательных), метод последовательных приближений (простых итераций), метод касательных, метод хорд. Исходя из полученных данных (в экспоненциальной форме), мы можем сделать вывод, что метод Ньютона самый точный.