МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Воронежский государственный технический университет»

(ФГБОУ ВО «ВГТУ», ВГТУ)

**Факультет** экономики, менеджмента и информационных технологий

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

Отчет по лабораторной работе 3

по дисциплине: «Информатика»

Выполнил студент: Олейников Борис Александрович

Группа: бИСТ- 213

Руководитель: доцент, к.т.н. Ефимова О.Е.

Работа защищена « » 2022г.

С оценкой

(подпись)

Воронеж 2022

**Лабораторная работа № 3**

**«Вычисление корня нелинейного уравнения 4 методами»**

Цель работы: изучить методы последовательных приближений (метод хорд, метод касательных, метод дихтомии и комбинированный метод). Для выполнения этого задания, необходимо создать программу, которая позволит всё автоматизировать и вычислить с заданной погрешностью.

**Задания Варианта 24 (Вариант 6):**

***Задание***

Самостоятельно рассмотреть метод секущих. Составить алгоритм и блок схему метода секущих. Объяснить чем метод секущих отличается от метода касательных, и что у них общего. Самостоятельно рассмотреть комбинированный метод хорд и касательных. Объяснить специфику применения комбинированного метода и касательных.

Используя методы нахождения корней нелинейного уравнения, найти все корни уравнения в соответствии с индивидуальным вариантом с заданной точностью:

Вариант 24 (Рисунок 1)



Метод Хорд отличается от Метода Касательных скоростью сходимости и некоторыми формулами. Метод хорд проще ( не нужно находить производную), так же он надёжнее и применим к не дифференцируемым функциям. Метод касательных не применим к не дифференцируемым функциям, но он быстрее.

Метод Хорд (метод секущих). Один из методов решения нелинейных уравнений, основанный на последовательном сужении интервала, содержащего единственный корень уравнения. Итерационный процесс выполняется до того момента, пока не будет достигнута заданная точность ε. Блок схема алгоритма показана на рисунке 2.

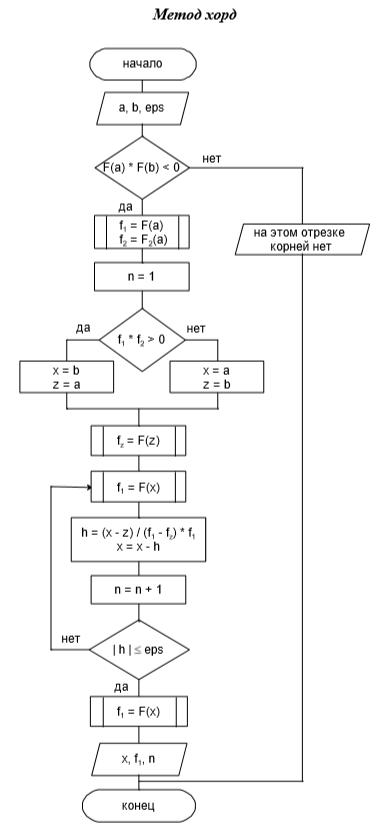


Рисунок 2 – Блок схема метода Хорд

Комбинированный метод. Суть комбинированного метода состоит в разбиении отрезка [a;b] на три отрезка с помощью хорды и касательной и выборе нового отрезка от точки пересечения хорды с осью абсцисс до точки пересечения касательной с осью абсцисс, на котором функция меняет знак и содержит решение.

Алгоритмы всех 4 методов показаны на рисунках 3, 4, 5 и 6.

Метод касательных. Его же суть состоит в разбиении отрезка [a;b] на два отрезка с помощью касательной и выборе нового отрезка от точки пересечения касательной с осью абсцисс до неподвижной точки, на которой функция меняет знак и содержит решение, причём подвижная точка приближается к **ε-окресности**  решения.

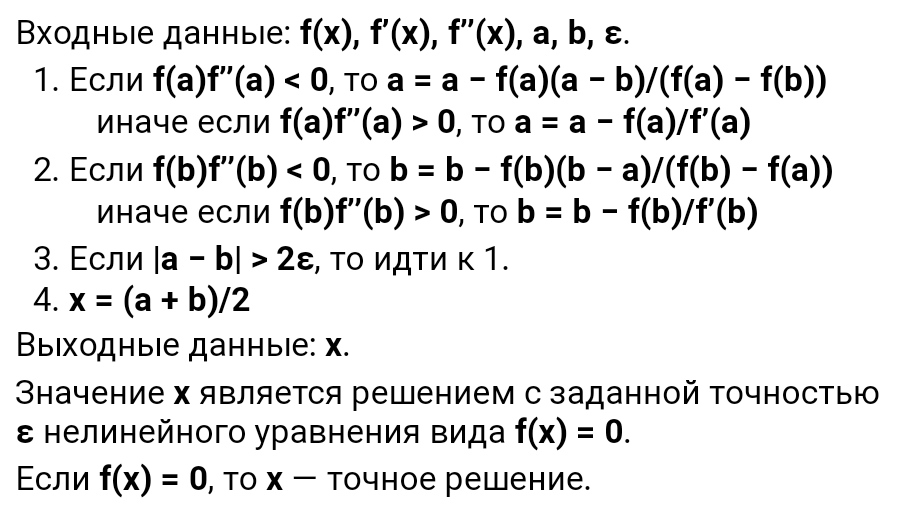


Рисунок 3 – Алгоритм Комбинированного метода

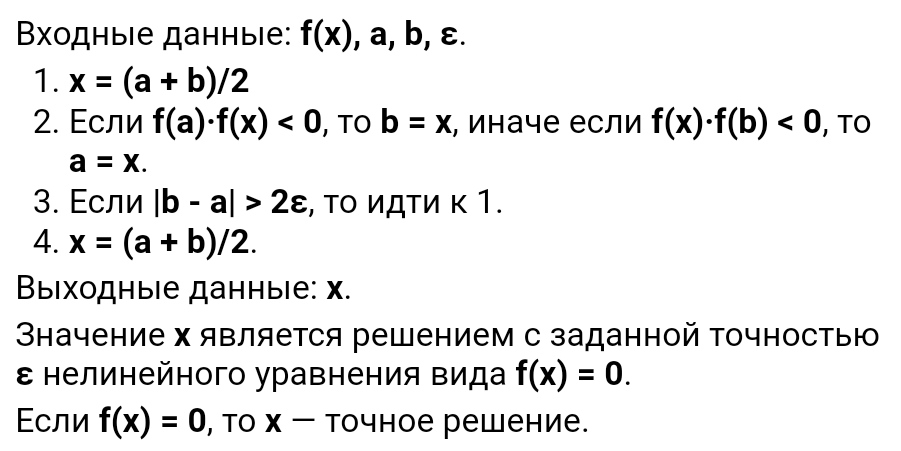


Рисунок 4 – Алгоритм метода дихтомии

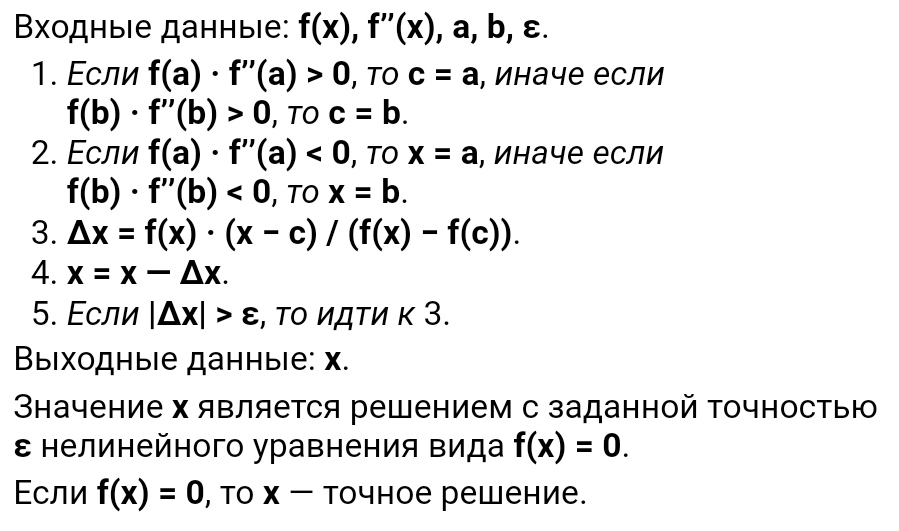


Рисунок 5 – Алгоритм метода хорд

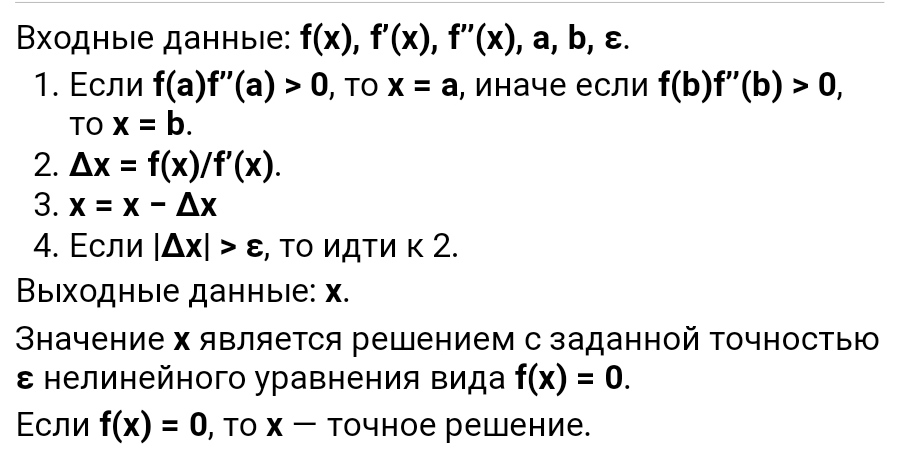
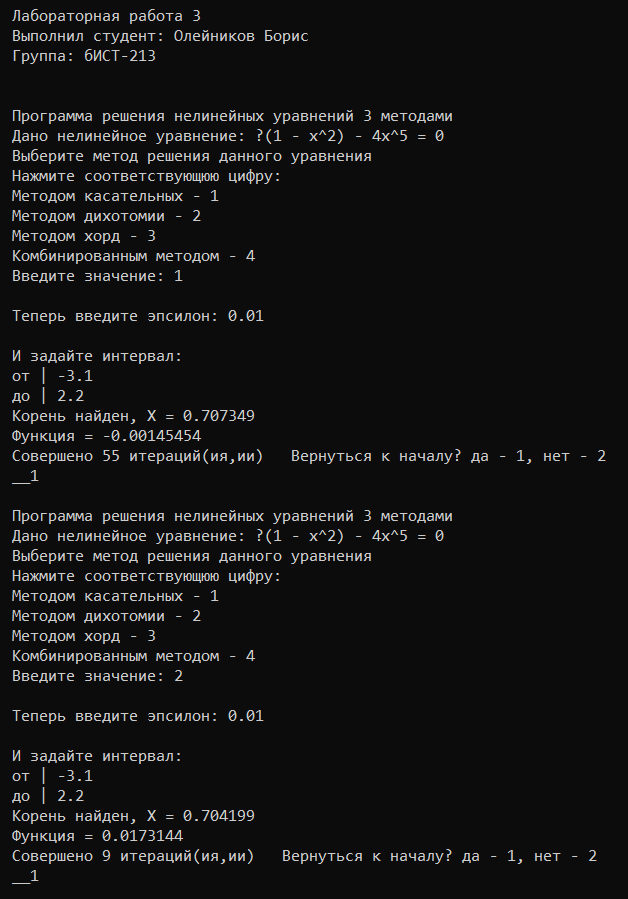
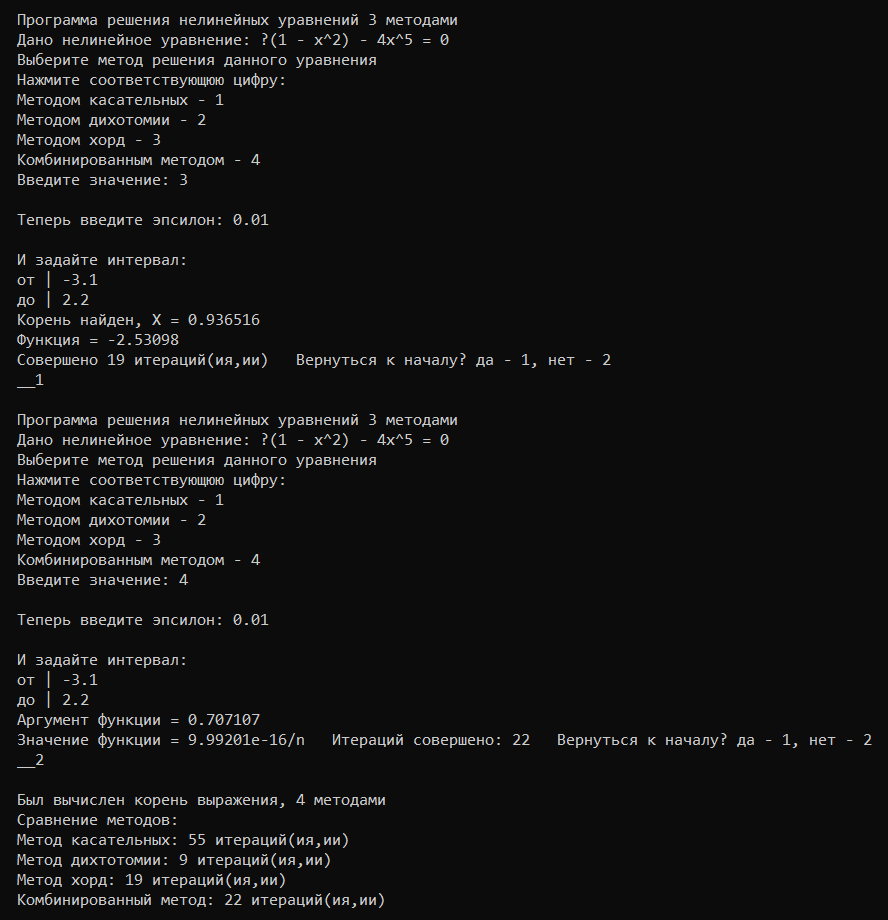


Рисунок 6 – Алгоритм метода касательных

**Тестирование программы:**





Метод дихотомии оказался самым быстро действенным при решении данного выражения. На этом тестирование закончено.

**Листинг Программы (С/С++)**

#include <stdlib.h>

#include <iostream>

#include <locale.h>

#include <math.h>

#define exp = 2,7182818284

using namespace std;

double F1(double x);

double PF1(double x);

double PF2(double x);

int main()

{

setlocale(LC\_ALL, "RUS");

puts("\n Лабораторная работа 3\n Выполнил студент: Олейников Борис\n Группа: бИСТ-213\n");

int ch2 = 1;

int ch1 = 0, n1 = 0, n2 = 0, n3 = 0, n4 = 0;

while (ch2 == 1)

{

double E, Y = 0,b1 = 0, a1 = 0, X = 0, DX = 0, f11 = 0, f22 = 0, fcc = 0, f1 = 0, f2 = 0, fc = 0, pf1 = 0, pf2 = 0;

cout << "\n Программа решения нелинейных уравнений 3 методами\n Дано нелинейное уравнение: √(1 – x^2) – 4x^5 = 0\n Выберите метод решения данного уравнения\n";

while (1 != 0)

{

cout << " Нажмите соответствующюю цифру:\n Методом касательных - 1\n Методом дихотомии - 2\n Методом хорд - 3\n Комбинированным методом - 4\n Введите значение: ";

cin >> ch1;

cout << endl;

if (ch1 != 1 && ch1 != 2 && ch1 != 3 && ch1 != 4)puts(" Вы ввели неправильное занчение... Повторите\n");

else break;

}

cout << " Теперь введите эпсилон: ";

cin >> E;

double a, b, c;

cout << "\n И задайте интервал:\n от | ";

cin >> a;

cout << " до | ";

cin >> b;

switch (ch1)

{

case 1:

if ((F1(a)\*F1(b)) >= 0)cout << " На данном отрезке корней нет";

else

{

f1 = F1(b);

pf2 = PF2(b);

n1++;

if ((f1 \* pf2) > 0)X = b;

else X = a;

while (1 != 0)

{

f1 = F1(X);

pf1 = PF1(X);

DX = f1 / pf1;

X -= DX;

n1++;

if (abs(DX) <= E)

{

f1 = F1(X);

cout << " Корень найден, Х = " << X << "\n Функция = " << f1 << "\n Совершено " << n1 << " итераций(ия,ии)";

break;

}

}

break;

}

case 2:

{

if ((F1(a)\*F1(b)) >= 0)cout << " На данном отрезке корней нет";

else

{

while (1 != 0)

{

X = (a + b) / 2;

fc = F1(a);

fcc = F1(b);

f1 = F1(X);

if ((fc\*f1) < 0)

{

b = X;

}

else if ((f1\*fcc) < 0) a = X;

n2++;

if (abs(b - a) <= 2\*E)

{

X = (a + b) / 2;

f1 = F1(X);

cout << " Корень найден, Х = " << X << "\n Функция = " << f1 << "\n Совершено " << n2 << " итераций(ия,ии)";

break;

}

}

}

break;

}

case 3:

{

if ((F1(a)\*F1(b)) >= 0)cout << " На данном отрезке корней нет";

else

{

fc = F1(a);

f1 = PF1(a);

f2 = PF2(a);

fcc = F1(b);

f11 = PF1(b);

f22 = PF2(2);

if ((fc\*f2) > 0)

{

c = a;

}

else if ((fcc \* f22)>0)

{

c = b;

}

if ((fc\*f2) < 0)

{

X = a;

}

else if ((fcc \* f22) < 0)

{

X = b;

}

while (1 != 0)

{

DX = F1(X) \* ((X - c) / (F1(X) - F1(c)));

X -= DX;

n3++;

if (abs(DX) <= E)

{

f1 = F1(X);

cout << " Корень найден, Х = " << X << "\n Функция = " << f1 << "\n Совершено " << n3 << " итераций(ия,ии)";

break;

}

}

}

break;

}

case 4:

{

if ((F1(a)\*F1(b)) >= 0)cout << " На данном отрезке корней нет";

else

{

a1 = a; b1 = b;

while (abs(a1 - b1) > 2\*E)

{

f1 = PF1(a1);

f2 = PF2(a1);

fc = F1(a1);

if ((fc \* f2) < 0) a1 = a1 - fc \* (a1 - b1) / (fc - F1(b1));

else a1 = a1 - fc / f1;

f11 = PF1(b1);

f22 = PF2(b1);

fcc = F1(b1);

if ((fcc \*f22) < 0) b1 = b1 - fcc \* (b1 - a1) / (fcc - F1(a1));

else b1 = b1 - fcc / f11;

n4++;

}

Y = (a1 + b1) / 2;

fc = F1(Y);

cout << " Аргумент функции = " << Y << "\n Значение функции = " << fc << "/n Итераций совершено: "<<n4;

}

break;

}

}

cout << " Вернуться к началу? да - 1, нет - 2\n \_\_";

cin >> ch2;

if (ch2 != 1 && ch2 != 2)

{

cout << "\n Значит выходим... (хоть и цифра у вас не та";

}

}

if (n1 != 0, n2 != 0, n3 != 0, n4 != 0)

{

cout << "\n Был вычислен корень выражения, 4 методами\n Сравнение методов:\n Метод касательных: " << n1 << " итераций(ия,ии)\n Метод дихтотомии: " << n2 << " итераций(ия,ии)\n Метод хорд: " << n3 << " итераций(ия,ии)\n Комбинированный метод: " << n4 << " итераций(ия,ии)" << endl;

}

return 0;

}

double F1(double x)

{

double fx = (sqrt(abs(1 - pow(x, 2)))) - 4 \* pow(x, 5);

return fx;

}

double PF1(double x)

{

double pfx = -1\*(x / (sqrt(abs(1 - pow(x, 2))))) - 20 \* pow(x, 4);

return pfx;

}

double PF2(double x)

{

//Извиняюсь за некомпактность функции. Были проблемы с вычислениями, пришлось отслеживать

double a1 = 1-pow(x, 2);

double a2 = pow(abs(a1),3./2);

if (a1 < 0)a2 = -1 \* a2;

double a3 = pow(x,4);

double a4 = pow(x, 2);

double a5 = a3 - a4;

double a6 = a5/a2;

double a7 = -100 \* a6;

double b1 = sqrt(abs(a1));

double b2 = 1. / b1;

double c1 = a7 - b2;

return c1;

}