**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная №1

**«Численное решение нелинейных уравнений»**

**Выполнил:**

Крученков Евгений Андреевич

студент 2 курса 9 группы,

специальность

“прикладная математика”

**Преподаватель:**

Ассистент кафедры вычислительной

математики ФПМИ,

Ю.Н. Горбачёва

Минск, 2022

**Содержание:**

Постановка задачи ------------------------------------------------------------------ 2-4

Краткие теоретические сведения ------------------------------------------------ 5

Листинг программы ---------------------------------------------------------------- 5-8

Результаты --------------------------------------------------------------------------- 8-11

Выводы ------------------------------------------------------------------------------- 11

**Постановка задачи**

**Задание 1**

Написать программу, которая находит решение уравнения f (x) = 0 c точностью ε = методами, указанными в варианте задания. Корень отделяем сначала графически, затем с помощью метода половинного деления с точностью ε = 0.1. Провести сравнительный анализ полученных результатов.

В содержание отчета должна быть включена следующая информация:

• Графики, которые использовались для отделения корня. Отрезок отделенного корня.

• Алгоритм метода половинного деления. Сводные данные по результатам работы метода половинного деления, оформленные в виде таблицы 1 (см. ниже).

• Алгоритмы методов, применяемые для нахождения корня уравнения с заданной точностью ε. Использовать в качестве отрезка отделенного корня суженный отрезок, полученный с помощью метода половинного деления.

• Проверка условий теоремы о сходимости метода простой итерации. Проверка условий теоремы о сходимости метода Ньютона.

• Сводные данные по результатам работы методов, оформленные в виде таблицы(см. ниже).

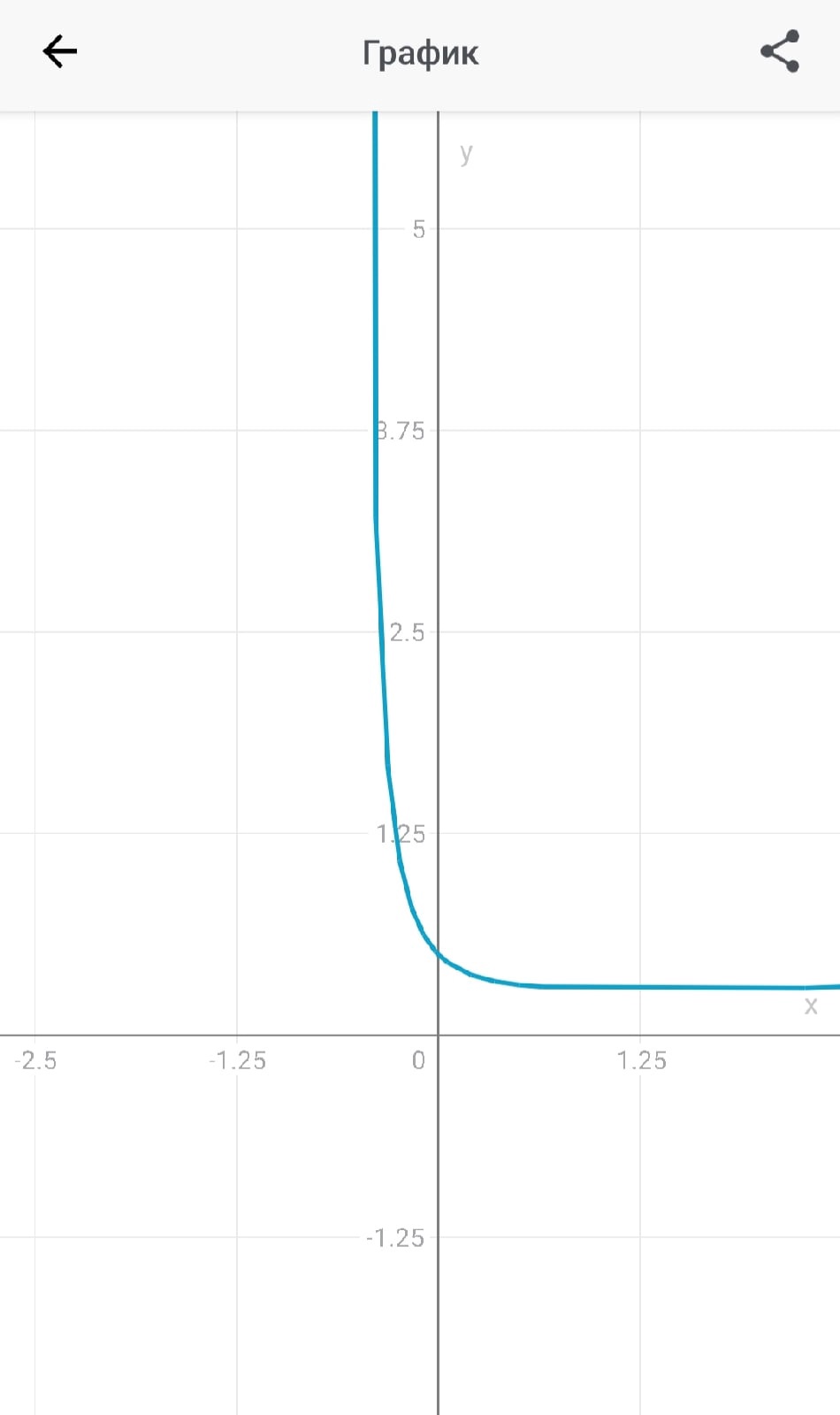
• Листинг программы с комментариями

Вариант 5

Функция для МПИ

Её производная

Её вторая производная (на выбранном промежутке не меняет знак, что видно из графика ниже)



Значит, при проверке условий теоремы о сходимости МПИ можем найти q как максимум значений на концах отрезка

*(q = 0.899631)*

Функция для метода Ньютона

Её первая производная

Её вторая производная

Её третья производная (на выбранном промежутке не меняет знак)

Значит при проверке условий теоремы о сходимости метода Ньютона найдем М как максимум значения второй производной на концах промежутка

*(M = 3.694941)*

*Реализовать: метод простой итерации, метод Ньютона, метод Ньютона с постоянной производной.*

Таблица для метода дихотомии:

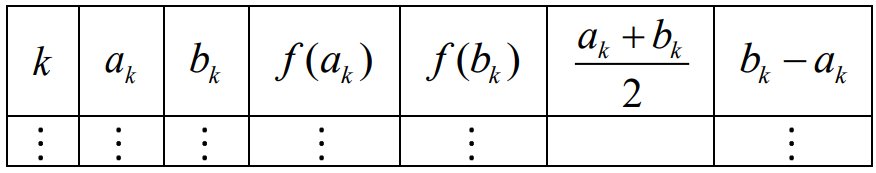
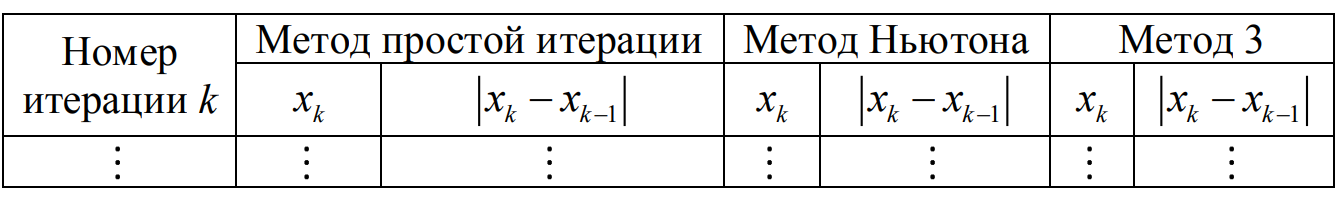


Таблица для других методов:



**Краткие теоретические сведения**

**МЕТОД ПОЛОВИННОГО ДЕЛЕНИЯ**

Если , то . В противном случае .

Алгоритм выполняется, пока .

**МЕТОД НЬЮТОНА**

Условие остановки

**МЕТОД НЬЮТОНА С ПОСТОЯННОЙ ПРОИЗВОДНОЙ**

Условие остановки

**МЕТОД ПРОСТОЙ ИТЕРАЦИИ**

Условие остановки:

**Листинг программы**

**Header.h**

#include <iostream>

#include <stdio.h>

#include <Math.h>

#include <iomanip>

#include <fstream>

#include <string>

using namespace std;

double E = pow(10, -7);

double F(double x) {

double F = sin(x) - 2 \* x \* x + 0.5;

return F;

}

double F\_(double x) {

double F = cos(x) - 4 \* x;

return F;

}

double F\_\_(double x) {

double F = (sin(x)+4)\*(-1);

return F;

}

double f(double x) {

return (-1)\*sqrt((sin(x) + 0.5) / 2);

}

double f\_(double x) {

//return pow((sin(x) + 0.5) / 2,-0.5)\*cos(x)\*0.25;

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

}

double\* Dihotomia(double a, double b) {

double x = (a + b) / 2;

cout << "k" << "\t " << "a\_k " << "\t" << "b\_k " << "\t" << " F(a\_k) " << "\t" << "F(b\_k)" << "\t " << "(a\_k + b\_k) / 2" << "\t " << " | a\_k - b\_k|" << endl;

for (int k = 0; abs(a - b) > 0.2; k++) {

cout << k << "\t " << a << "\t " << b << "\t " << F(a) << "\t " << F(b) << "\t " << (a + b) / 2 << "\t " << abs(b - a) << endl;

if (F(x) \* F(a) < 0) b = x;

else a = x;

x = (a + b) / 2;

}

double\* mass = new double[3];

mass[0] = x;

mass[1] = a;

mass[2] = b;

return mass;

}

bool T1(double x, double a, double b) {

//1

double delta = abs(a - b)/2;

cout << "1. Дельта = " << delta << endl;

//2

//найдём максимум производной фуекции мпи

//заметим, что производная функции монотонна, значит можем найти максимум,

//подставив значение на концах

double q;

if (abs(f\_(a)) > abs(f\_(b))) q = abs(f\_(a));

else q = abs(f\_(b));

cout << "2. q = " << q << endl;

if (q >= 1) {

cout << "Условия теоремы не выполнены: q>=1" << endl;

return false;

}

//3

double m = abs(x - f(x));

cout << "3. m = " << m << endl;

if ((m / (1 - q)) > delta) {

cout << "Условия теоремы не выполнены: m / (1 - q) > delta" << endl;

return false;

}

cout << "Условия теоремы выполнены." << endl;

return true;

}

double MPI(double x, double a, double b) {

cout << "k" << "\t " << "Xk" << "\t " << "|Xk+1-Xk|" << endl;

double x\_ = x;

int k = 0;

do

{

x\_ = x;

x = f(x);

cout <<dec<< k << "\t " <<scientific<< x\_ << "\t " << abs(x - x\_) << endl;

k++;

} while (abs(x - x\_) > E);

return x;

}

bool T2(double x, double a, double b) {

//1

double h = (-1)\*F(x)/F\_(x);

cout << "1. h0 = " << h << endl;

cout << "Значения F(x)\*F\_(x) на концах отрезка: " << F(x) \* F\_(x) << " и " << F(x + 2 \* h) \* F\_(x + 2 \* h) << endl;

if (abs(F(x) \* F\_(x))<0.00001 || abs(F(x + 2 \* h) \* F\_(x + 2 \* h))<0.00001) {

cout << "Условия теоремы не выполнены: значения на концах отрезка близки к 0." << endl;

return false;

}

//2

double M;

if (abs(F\_\_(x)) > abs(F\_\_(x+2\*h))) M = abs(F\_\_(x));

else M = abs(F\_\_(x+2\*h));

cout << "2. M = " << M << endl;

if (2\*abs(h)\*M > abs(F\_(x))) {

cout << "Условия теоремы не выполнены: 2\*|h|\*M > |F\_(x)|" << endl;

return false;

}

//3

cout << "Условия теоремы выполнены." << endl;

return true;

}

double MNpost(double x, double a, double b) {

cout << "k" << "\t " << "Xk" << "\t " << "|Xk+1-Xk|" << endl;

double x\_ = x, F\_post = F\_(x);

int k = 0;

do

{

x\_ = x;

x = x-(F(x)/F\_post);

cout <<dec <<k << "\t " <<scientific<< x\_ << "\t " << abs(x - x\_) << endl;

k++;

} while (abs(x - x\_) > E);

return x;

}

double MN(double x, double a, double b) {

cout << "k" << "\t " << "Xk" << "\t " << "|Xk+1-Xk|" << endl;

double x\_ = x;

int k = 0;

do

{

x\_ = x;

x = x - (F(x) / F\_(x));

cout << dec << k << "\t " << scientific << x\_ << "\t " << abs(x - x\_) << endl;

k++;

} while (abs(x - x\_) > E);

return x;

}

**Main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <fstream>

#include <string>

#include "Header.h"

using namespace std;

ifstream fin;

ofstream fout;

long main()

{

setlocale(LC\_ALL, "ru");

cout << "Дихотомия:" << endl;

double \*mass = Dihotomia(-1, 0);

cout << endl<<"Проверка условий теоремы о сходимости МПИ:"<<endl;

T1(mass[0], mass[1], mass[2]);

cout << endl;

cout << "МПИ:" << endl;

MPI(mass[0], mass[1], mass[2]);

cout << endl << "Проверка условий теоремы о сходимости метода Ньютона:" << endl;

T2(mass[0], mass[1], mass[2]);

cout << endl << "Метод Ньютона:" << endl;

MN(mass[0], mass[1], mass[2]);

cout << endl << "Метод Ньютона c постоянной производной:" << endl;

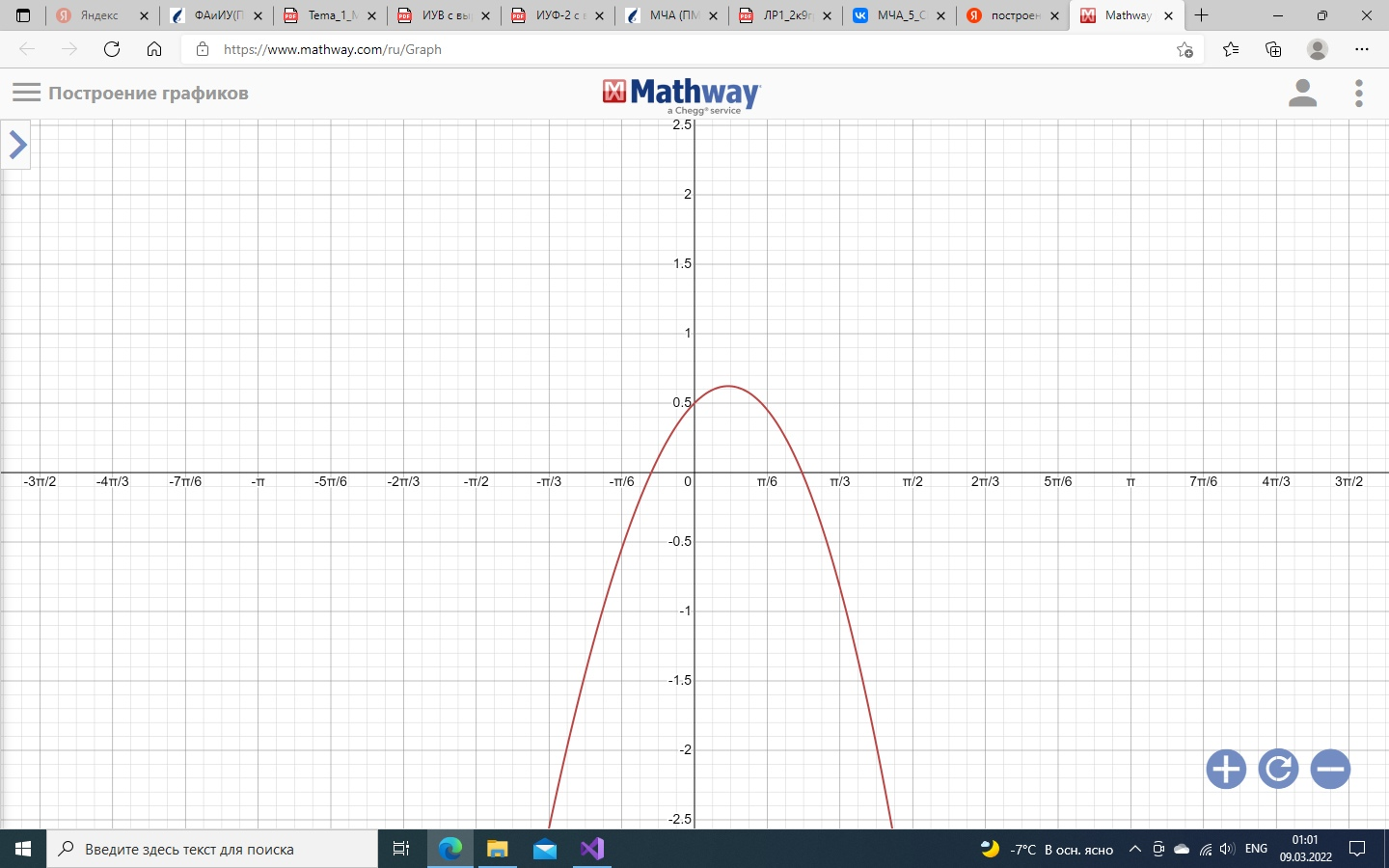
MNpost(mass[0], mass[1], mass[2]);

return 0;

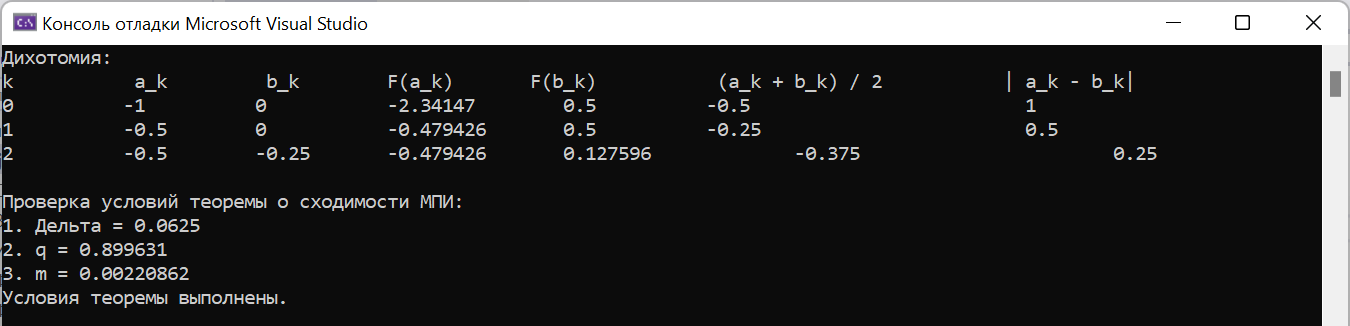
}

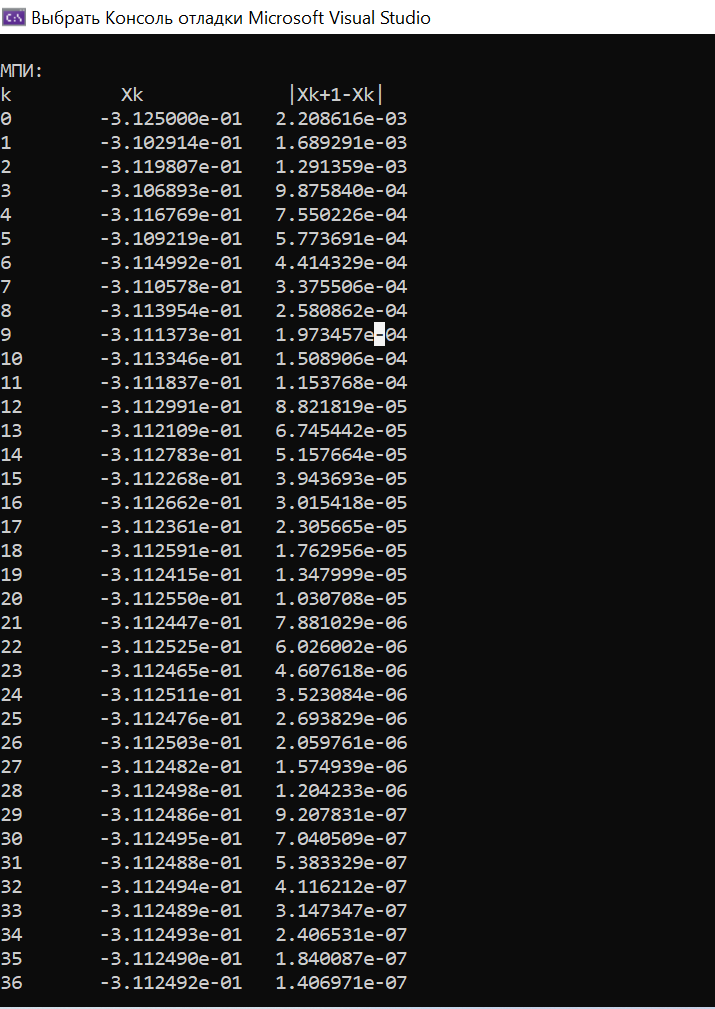
**Результаты**

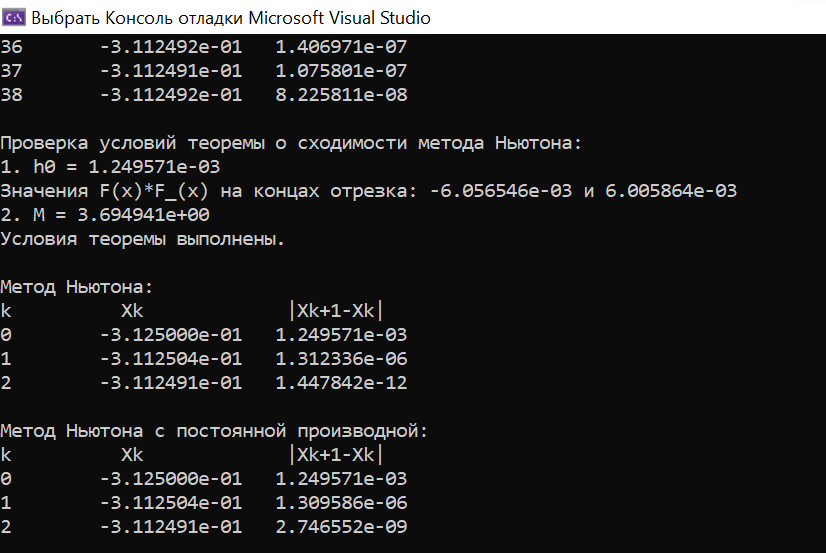
*График функции:*

**

Выберем начальные отрезок [-1;0].

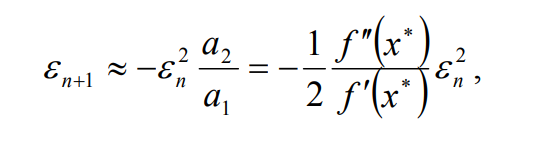
****

****

****

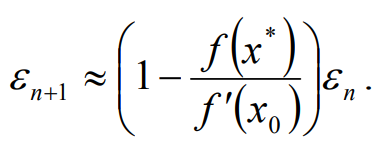
**Выводы**

1. Погрешность метода Ньютона на n+1 шаге выражается через погрешность на n шаге следующим образом:



Следовательно, метод имеет квадратичную сходимость.

1. Для метода Ньютона с постоянной производной:



Скорость сходимости хуже – линейная.

1. МПИ имеет линейную сходимость.