Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

ПНИПУ

Лабораторная работа

**«Численные методы решения уравнений»**

Выполнила:

студентка группы РИС-23-3б

Шуракова Анастасия Андреевна

Проверила:

доцент кафедры ИТАС

О.А. Полякова

2023 г.

**Создание алгоритма для выполнения задачи**

**Постановка задачи:**

Создать алгоритм и разработать программу на языке C++ для решения данного уравнения:

**Метод половинного деления**

**Словесный алгоритм:**

1. В условии задачи уточняется отрезок [a; b], на котором функция имеет корень, сам вид функции , а также требуемая точность epsilon;
2. С помощью теоремы[1]определяется первый корень, который находится как ;
3. Далее запускается цикл while, который продолжает работу, пока модуль разности между концами отрезка не станет меньше или равен заданной точности . Если , то значение b заменяется на x, иначе значение a заменяется на x;
4. В результате точное значение корня выводится в виде .

[1] Теорема: «Если непрерывная функция на концах некоторого интервала имеет значения разных знаков, то внутри этого интервала у нее есть корень».

**Смысловые значения переменных:**

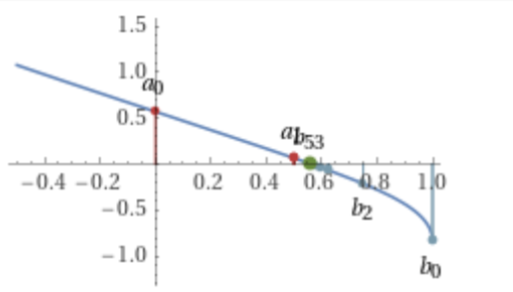
a – левый конец отрезка;

b – правый конец отрезка;

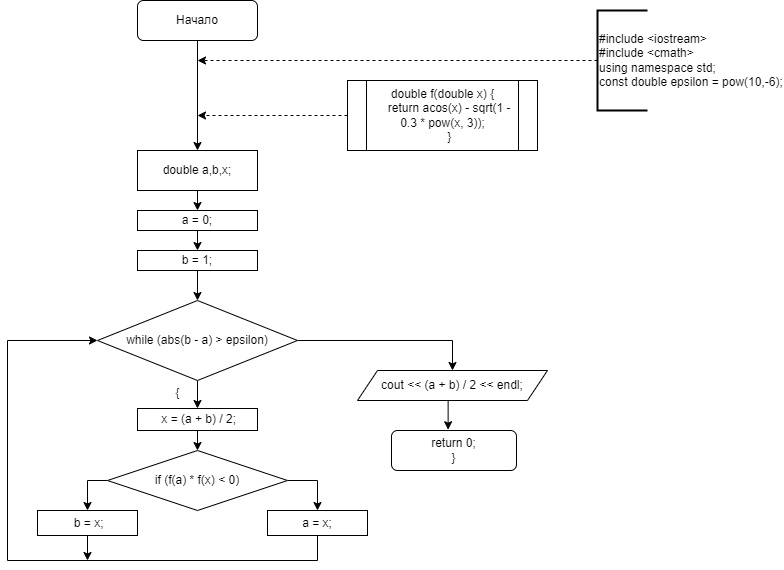
x – корень уравнения;

epsilon – точность.

**Геометрическая интерпретация:**

****

**Блок-схема:**



**Код:**

#include <iostream>

#include <cmath>

using namespace std;

const double epsilon = pow(10,-6);

// Метод итерации

double f(double x) {

return acos(x) - sqrt(1 - 0.3 \* pow(x, 3));

}

int main() {

setlocale(LC\_ALL, "Rus");

double a,b,x;

a = 0;

b = 1;

while (abs(b - a) > epsilon) {

x = (a + b) / 2;

if (f(a) \* f(x) < 0) {

b = x;

}

else {

a = x;

}

}

cout << (a + b) / 2 << endl;

return 0;

}

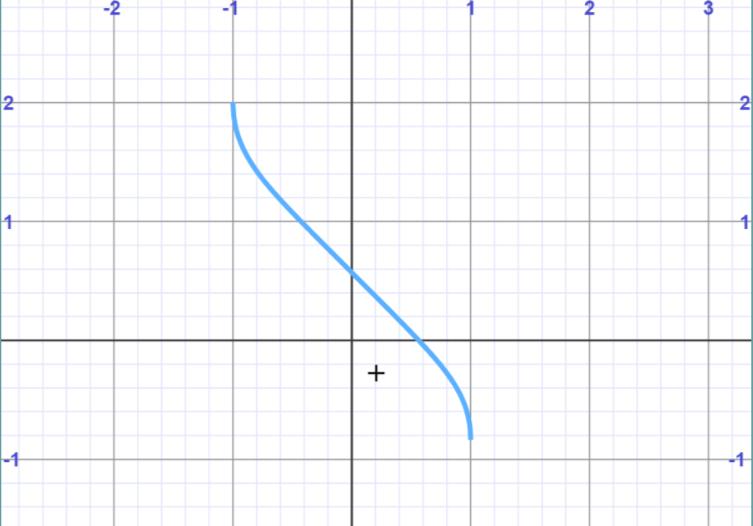
**Работа программы:**

****

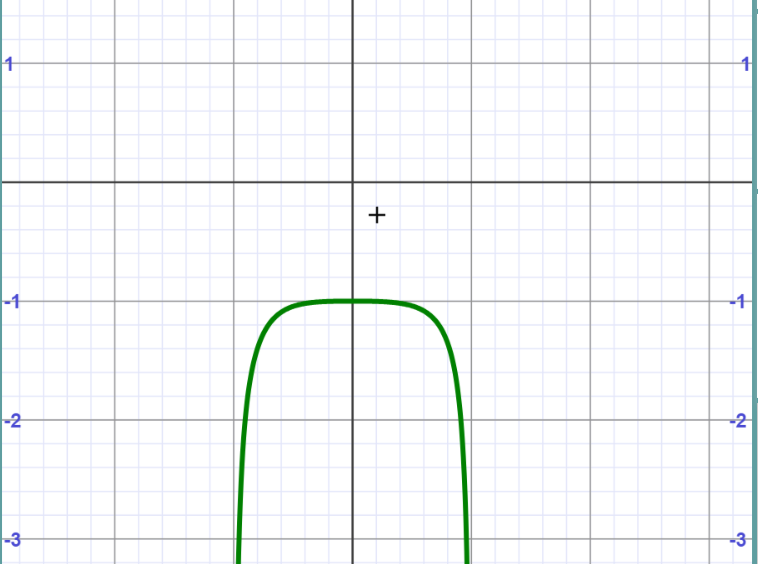
**Метод Ньютона**

**Условия применения алгоритма:**

Данный алгоритм требует, чтобы функция была монотонной и непрерывной, что подтверждается графической проверкой существования второй производной на интервале [0, 1]:

****

Также необходимо удостовериться, что первая производная не равна нулю для всех x на интервале, что также подтверждается графически:

****

**Словесный алгоритм:**

1. В условии задачи уточняется отрезок [a; b], на котором функция имеет корень, сам вид функции , и необходимая точность epsilon;
2. Алгоритм начинается с выбора начального приближения x\_0. Значение x\_0 определяется в зависимости от характеристик функции: если , то x\_0 = a, если , то x\_0 = b;
3. Затем находится x\_1 по формуле ;
4. После этого запускается цикл, который работает до тех пор, пока модуль разности x\_0 и x\_1 не станет меньше или равен заданной точности;

a.Значение x\_0 меняется на x\_1;

b.Повторяется третий шаг;

1. Выводится ответ.

**Смысловые значения переменных:**

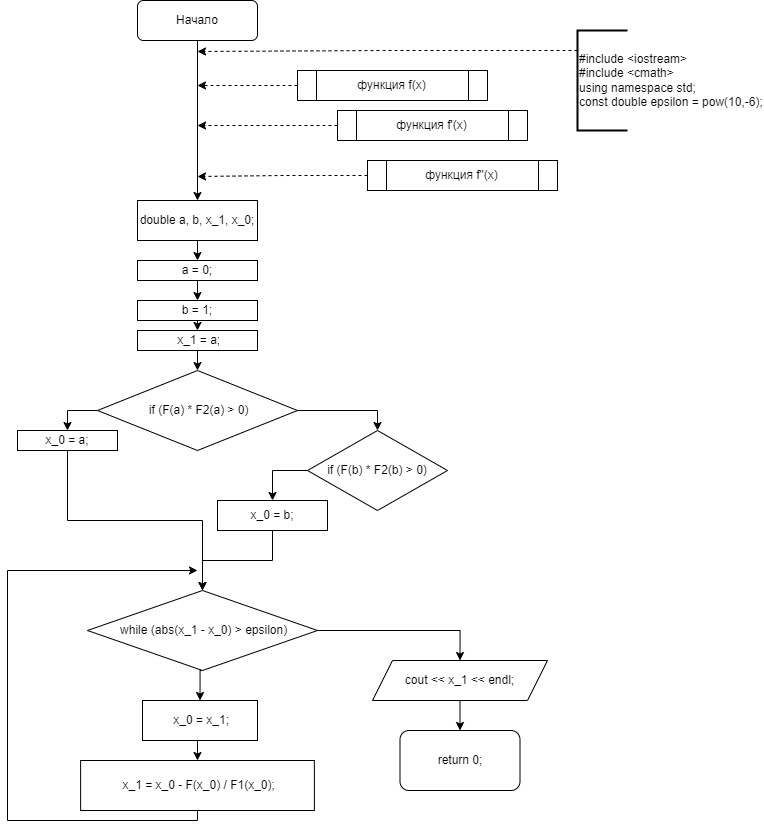
a – левый конец отрезка;

b – правый конец отрезка;

x\_0, x\_1 – точки пересечения касательной и оси абсцисс, корни уравнения;

epsilon – точность.

**Блок-схема:**

****

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

double const epsilon = pow(10, -6);

double F(double x) {

return acos(x) - sqrt(1 - 0.3 \* pow(x, 3));

}

double F1(double x) {

return (9 \* pow(x, 2)) / (20 \* sqrt(1 - (3 \* pow(x, 3)) / 10)) - (1 / (sqrt(1 - pow(x, 2))));

}

double F2(double x) {

return (9 \* (2 \* x \* sqrt(1 - (3 \* pow(x, 3)) / 10) + (9 \* pow(x, 4)) / (20 \* sqrt(1 - (3 \* pow(x, 3)) / 10)))) / (20 \* (1 - (3 \* pow(x, 3)) / 10)) - x / (pow(1 - pow(x, 2), 3 / 2));

}

int main() {

double a, b, x\_1, x\_0;

setlocale(LC\_ALL, "Rus");

a = 0;

b = 1;

x\_1 = a;

if (F(a) \* F2(a) > 0) {

x\_0 = a;

}

else {

if (F(b) \* F2(b) > 0) {

x\_0 = b;

}

}

while (abs(x\_1 - x\_0) > epsilon) {

x\_0 = x\_1;

x\_1 = x\_0 - F(x\_0) / F1(x\_0);

}

cout << x\_1 << endl;

return 0;

}

**Работа программы:**

****

**Метод итераций**

**Условия применения алгоритма:**

Этот метод может быть использован в том случае, если известен интервал, в котором находится корень уравнения (x ∈ [a; b]), данный интервал уточнен в условии задачи (x ∈ [0; 1]). Важным требованием также является то, что абсолютное значение производной новой функции должно быть меньше 1: .Новая функция:

**Словесный алгоритм:**

1. Уравнение f(x)=0 переписывается в виде x=φ(x) для дальнейших вычислений.
2. На заданном отрезке [a; b] устанавливается начальное приближение х\_0 (x\_0 ∈ [0; 1]), ;
3. Следующее приближение вычисляется по формуле )
4. Этот процесс повторяется, пока . Как только , цикл завершается и становится искомым значением.
5. Ответ выводится на экран.

**Смысловые значения переменных:**

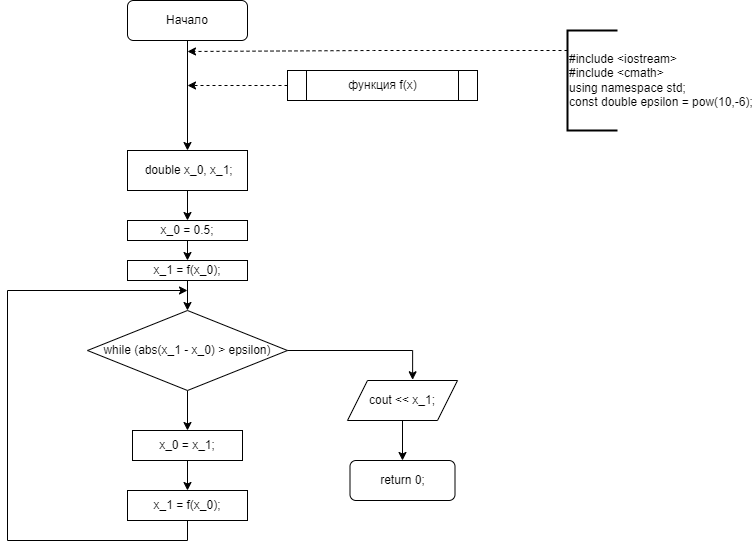
a – левый конец отрезка;

b – правый конец отрезка;

x\_0, x\_1 – точки пересечения касательной и оси абсцисс, корни уравнения;

epsilon – точность.

**Блок-схема:**

****

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

const double epsilon = pow(10,-6);

double f(double x)

{

// return acos(x) - sqrt(1 - 0.3 \* pow(x, 3));

return cos(sqrt(1 - (0.3 \* pow(x, 3))));

}

int main()

{

double x\_0, x\_1;

x\_0 = 0.5;

x\_1 = f(x\_0);

while (abs(x\_1 - x\_0) > epsilon)

{

x\_0 = x\_1;

x\_1 = f(x\_0);

}

cout << x\_1;

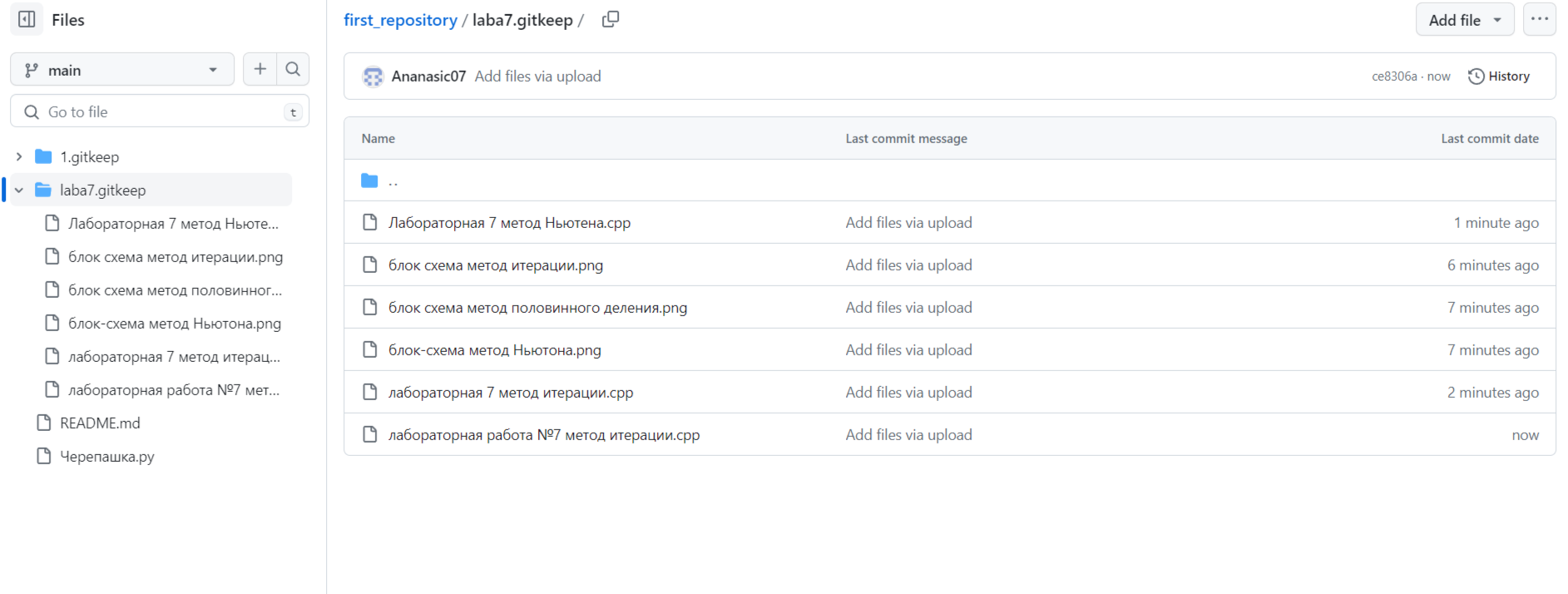
return 0;

}

**Работа программы:**

****

**Скрин из Git-Hub:**

****

**Вывод**

В ходе выполнения лабораторной работы были исследованы различные методы решения нелинейных уравнений, такие как метод половинного деления, метод Ньютона и метод простой итерации. Лабораторная работа позволила углубленно изучить методы их решения.