Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Кафедра информационных технологий и автоматизированных систем

**ОТЧЕТ ПО**

**ЛАБОРАТОРНОЙ РАБОТЕ**

**Тема:** решение нелинейных уравнений тремя методами

Выполнила студентка РИС-23-2б:

Куклина Юлия Витальевна

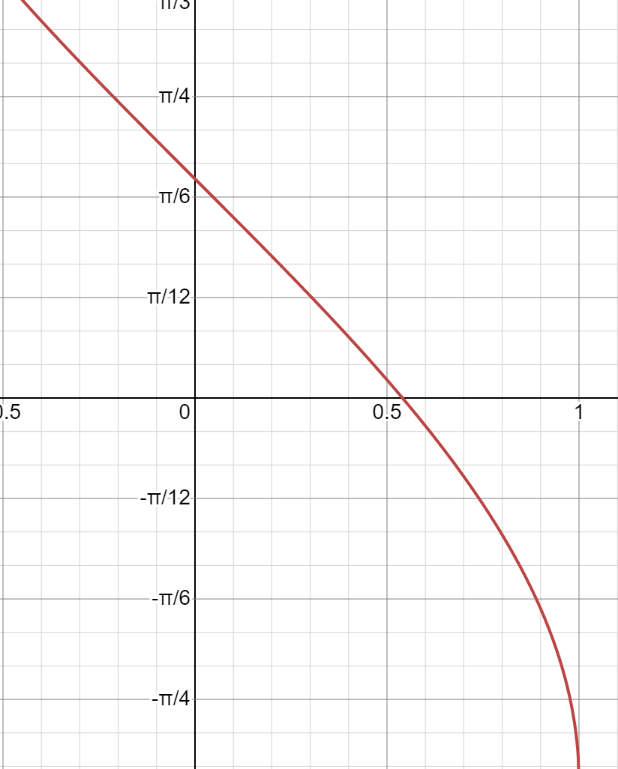
Проверила: доцент кафедры ИТАС

О.А. Полякова

2023

**Задача**

Разработать алгоритм и написать три программы на С++ для решения нелинейного уравнения . Отрезок [a;b], a=0, b=1; точное значение 0,5629; заданная точность epsilon = 0.0001.



**Метод половинного деления**

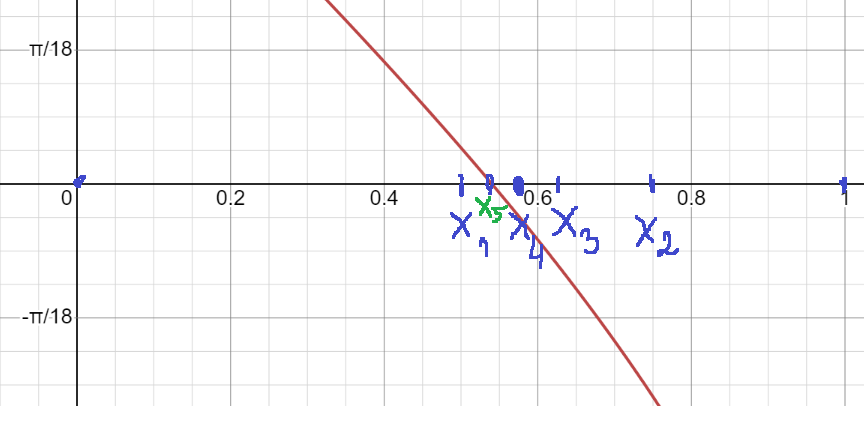
**(метод дихотомии)**

**Словесный алгоритм:**

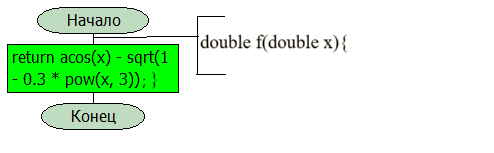
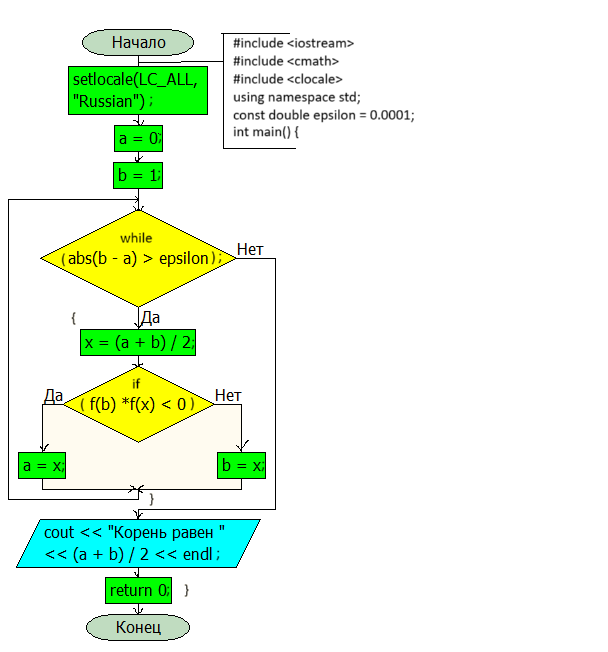
1. На заданном отрезке существует корень функции вида f(x)=0.
2. Если непрерывная функция на концах некоторого интервала имеет значения разных знаков, то внутри этого интервала у нее есть корень. Первый корень находится по формуле

x = (a+b)/2.

1. Цикл while работает до тех пор, пока модуль разности между концами отрезка не станет меньше или равен заданной точности ;
2. Eсли , то значение a заменяется на x, иначе значение b заменяется на x;
3. Выводится точное значение корня x = (a+b)/2.



**Блок-схема**



**Код**

#include <iostream>

#include <cmath>

#include <clocale>

using namespace std;

const double epsilon = 0.0001;

double f(double x)

{

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

}

int main()

{

setlocale (LC\_ALL, "Russian");

double a, b, x;

a = 0;

b = 1;

while (abs(b - a) > epsilon)

{

x = (a + b) / 2;

if(f(b) \* f(x) < 0)

a = x;

else

b = x;

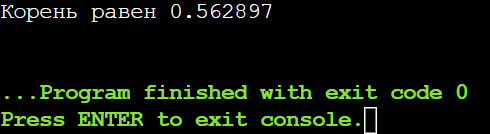
}

cout<<"Корень равен "<< (a + b) / 2 << endl;

return 0;

}

**Работа кода**



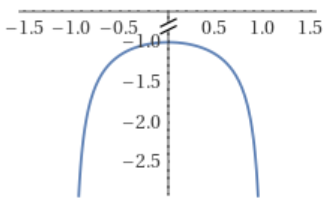
**Метод Ньютона**

**Словесный алгоритм:**

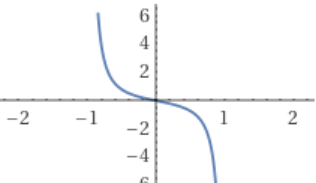
1. Функция вида на заданном отрезке [a;b] имеет корень
2. Выбирается начальное приближение x\_0. Значение x\_0 определяется в зависимости от характеристик функции: если , то x\_0 = a, если , то x\_0 = b
3. x находится по формуле
4. Цикл while работает до тех пор, пока модуль разности x\_0 и x не станет меньше или равен заданной точности
5. Значение x\_0 меняется на x;
6. Выводится ответ.

**Условия применимости:**

1. Первая производная функции не равна нулю на заданном отрезке [0;1]



1. Вторая производная функции монотонна и непрерывна на отрезке



**Код**

#include <iostream>

#include <math.h>

#include <clocale>

using namespace std;

double f(double x)

{

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

}

double diff(double x)

{

const double h = 1e-10;

return (f(x + h) - f(x - h)) / (2.0 \* h);

}

int main()

{

setlocale(LC\_ALL, "ru");

double x = 0, x0 = 1;

double e = 0.0001;

do

{

x0 = x;

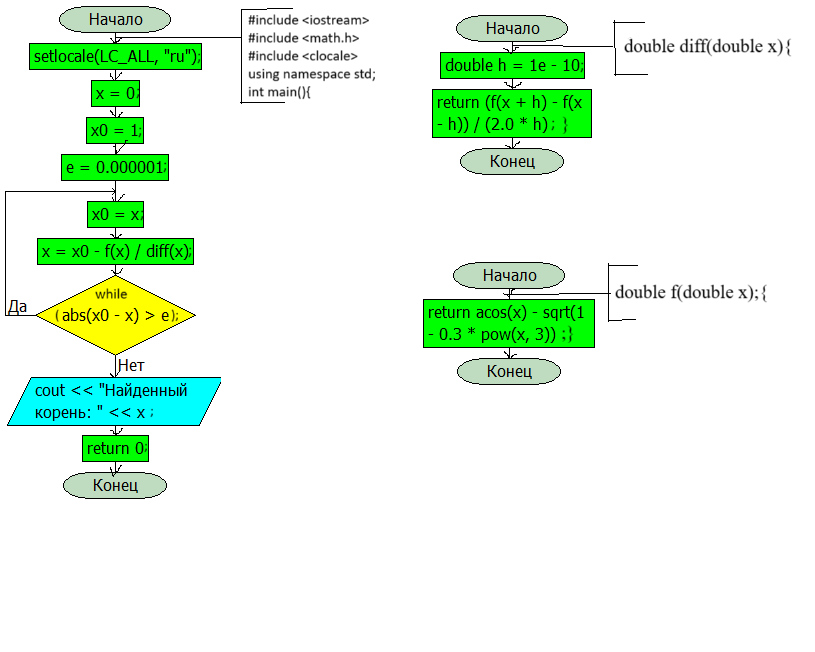
x = x0 - f(x) / diff(x);

} while (abs(x0 - x) > e);

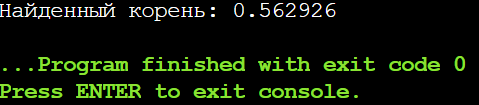
cout << "Найденный корень: " << x ;

return 0;}

**Блок-схема**



**Работа кода**



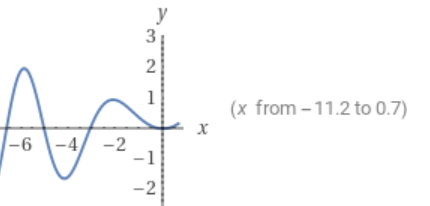
**Метод итераций**

**Словесный алгоритм:**

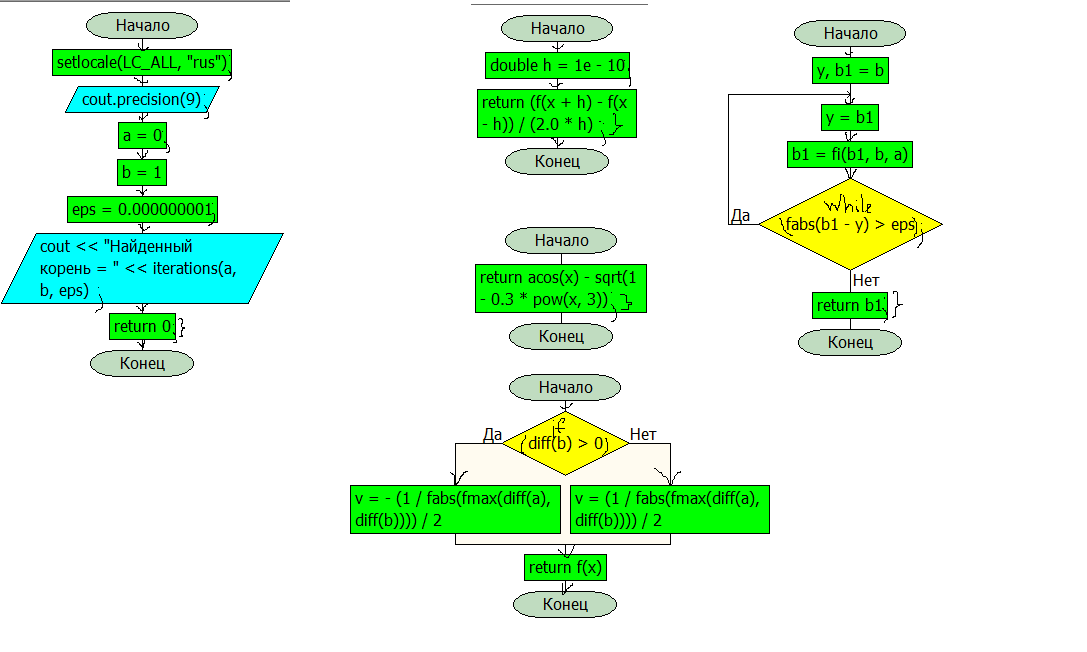
1. Уравнение f(x)=0 представляется в виде x=φ(x)
2. На заданном интервале выбирается начальное приближение x0
3. Следующее приближение )
4. Цикл повторяется, пока
5. После завершения цикла искомым значением становится

**Условия применения алгоритма:**

1. Интервал, в котором находится корень уравнения [0;1]. Значение производной новой функции должно быть меньше 1
2. Производная новой функции удовлетворяет условию применимости.



**Блок-схема**

****

**Код**

#include <iostream>

#include <cmath>

using namespace std;

double f(double x)

{

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

}

double diff(double x)

{

const double h = 1e-10;

return (f(x + h) - f(x - h)) / (2.0 \* h);

}

double fi(double x, double b, double a)

{

double v;

if (diff(b) > 0)

v = -(1 / fabs(fmax(diff(a), diff(b)))) / 2;

else

v = (1 / fabs(fmax(diff(a), diff(b)))) / 2;

return x + v \* f(x);

}

double iterations(double a, double b, double eps)

{

double y, b1 = b;

do {

y = b1;

b1 = fi(b1, b, a);

}

while (fabs(b1 - y) > eps);

return b1;

}

int main()

{

setlocale(LC\_ALL, "rus");

cout.precision(9);

double a = 0;

double b = 1;

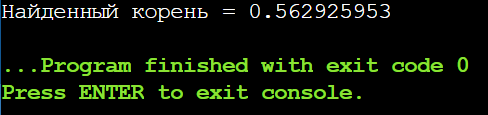
double eps = 0.0001;

cout << "Найденный корень = " << iterations(a, b, eps);

return 0;

}

**Работа кода**

****

**Вывод**

Все программы правильно считают корень нелинейного уравнения. Наиболее эффективным методом оказался метод итераций, который наиболее точно находит корень заданного уравнения.

**GIT**