**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**“Московский Авиационный Институт**

**(Национальный Исследовательский Университет)”**

**Факультет информационных технологий и прикладной математики**

**Кафедра 806 “Вычислительная математика и программирование”**

**Курсовая работа**

**по курсу “Вычислительные системы”**

1 семестр

Задание 4. Процедуры и функции в качестве параметров

Студент: Цирулев Н.В.

Группа: М8О-108Б-22,

№ по списку 22

Руководитель: Сахарин Н.А.

Дата: 09.01.23

Оценка:

Москва, 2023

**ОГЛАВЛЕНИЕ**

[Задача 2](#_Toc124186989)

[общий метод решения 3](#_Toc124186990)

[Метод дихотомии 3](#_Toc124186991)

[Метод итераций 3](#_Toc124186992)

[Метод Ньютона 4](#_Toc124186993)

[Общие сведения о программе 5](#_Toc124186994)

[Описание переменных и функций 6](#_Toc124186995)

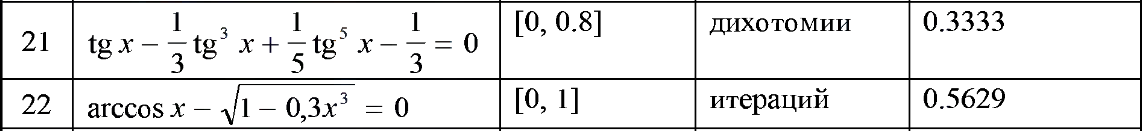
[Протокол 7](#_Toc124186996)

[Заключение 10](#_Toc124186997)

[Список использованных источников 11](#_Toc124186998)

# Задача

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления – дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером. Если метод неприменим, дать математическое обоснование.

22 вариант

# общий метод решения

Каждое уравнение решаем 3 методами: итераций, дихотомии и Ньютона.

### **Метод дихотомии**

Очевидно, что если на отрезке [a,b] существует корень уравнения, то значения функции на концах отрезка имеют разные знаки F(a)\*F(b) < 0. Метод заключается в делении отрезка пополам и его сужения в два раза на каждом шаге итерационного процесса в зависимости от знака функции в середине отрезка.

Итерационный процесс строится следующим образом: за начальное приближение принимаются границы исходного отрезка . Далее вычисления проводятся по формулам: , если ; или по формулам:, если .

До тех пор, пока не будет выполнено условие , процесс будет выполняться.

Приближенное значение корня к моменту окончания итерационного процесса получается следующим образом.

### **Метод итераций**

Идея метода заключается в замене исходного уравнения F(x) = 0 уравнением вида . Достаточное условие сходимости данного метода . Это условие необходимо проверить перед началом решения задачи, так как функция f(x) может быть выбрана неоднозначно, причем в случае неверного выбора указанной функции, метод расходится. Достаточно неплохим выбором является функция где – производная функции .

Начальное приближение корня: (середина исходного отрезка).

Итерационный процесс: .

Условие окончания: .

### **Метод Ньютона**

Метод Ньютона является частным случаем метода итераций.

Условие сходимости метода: на отрезке .

Итерационный процесс: .

Метод обладает квадратичной сходимостью. Модификацией метода является метод хорд и касательных. Также метод Ньютона может быть использован для решения задач оптимизации, в которых требуется определить ноль первой производной либо градиента в случае многомерного пространства.

В начале программы составляем функции для решения уравнения определённым методом и функции, служащим им аргументом. Для каждой формулы необходимо запрограммировать исходную функцию. Производную в точку можно принять за дифференциал от функции в данной точке, который считается по формуле . В теле основной программы делаем только вызов подпрограмм и вывод.

Для удобства и лаконичности в программе с помощью оператора typedef введен тип pFunc, который расшифровывается как pointer function и ld, который расшифровывается long double.

# Общие сведения о программе

Язык и система программирования: GNU C

Местонахождение файлов \\wsl.localhost\Ubuntu\home\hackerman\

Способ вызова и загрузки:

gcc cw.c -Wall -std=c18 -pedantic -lm

./a.out

# Описание переменных и функций

Таблица 1.А. Описание переменных

| Имя | Тип | Описание |
| --- | --- | --- |
| a | ld | Левая граница отрезка |
| b | ld | Права граница отрезка |

Таблица 1.Б. Описание функций

| Название функции | Выходной тип | Входные параметры | Описание |
| --- | --- | --- | --- |
| machineeps | ld | отсутствуют | Функиця, считающая машинное эпсилон |
| dx | ld | pFunc f, ld x | Функция, считающая дифференциал в данной точке |
| sign | ld | ld x | Возвращает 1, если знак x положительный, -1, если отрицательный и 0, если x близок к нулю. |
| dht\_method | ld | pFunc f, ld a, ld b | Находит корень функции методом дихотомии |
| iter\_method | ld | pFunc f, ld a, ld b | Находит корень функции методом итераций |
| newton\_method | ld | pFunc f, ld a, ld b | Находит корень функции методом Ньютона |
| func | ld | ld x | Считает значение функции в данной точке |

# Протокол

hackerman@WARMACHINE\_mini:~$ cat cp4.c

#include <stdio.h>

#include <math.h>

typedef long double ld;

typedef ld(\*pFunc)(ld);

ld machineeps() {

ld epsilon = 1, prev;

ld expression;

do {

prev = epsilon;

epsilon = epsilon / 2;

expression = 1 + epsilon;

} while (expression > 1);

return prev;

}

ld dx(pFunc f, ld x) {

ld eps = machineeps() \* 2;

return (f(x + eps / 2) - f(x - eps / 2)) / eps;

}

ld sign(ld x) {

return x > machineeps() ? 1 : x < -machineeps() ? -1 : 0;

}

ld dht\_method(pFunc f, ld a, ld b) {

ld x = (a + b) / 2;

while (fabsl(a - b) > machineeps()) {

x = (a + b) / 2;

if(f(a) \* f(x) > 0.l) a = x;

else b = x;

}

return x;

}

ld iter\_method(pFunc f, ld a, ld b) {

ld x = (a + b) / 2;

while(fabsl(f(x)) > machineeps()){

x = x - f(x) \* sign(dx(f, x));

}

return x;

}

ld newton\_method(pFunc f, ld a, ld b) {

ld x = (a + b) / 2;

while(fabsl(f(x) / dx(f, x)) > machineeps()){

x = x - f(x) / dx(f, x);

}

return x;

}

ld a = 0, b = 1;

ld func(ld x) {

return acosl(x) - sqrtl(1 - 0.3 \* powl(x, 3));

}

int main() {

printf("dichotomy method result for func: %Lf\n", dht\_method(func, a, b));

printf("iteration method result for func: %Lf\n", iter\_method(func, a, b));

printf("newton method result for func: %Lf\n", newton\_method(func, a, b));

}

hackerman@WARMACHINE\_mini:~$ gcc cp4.c -Wall -std=c18 -pedantic -lm

hackerman@WARMACHINE\_mini:~$ ./a.out

dichotomy method result for func: 0.562926

iteration method result for func: 0.562926

newton method result for func: 0.562926

hackerman@WARMACHINE\_mini:~$

# заключение

Данное задание курсового проекта показывает суть некоторых численных методов и их практическое применение для вычисления приближенного значения корней, однако в абсолюте ни один из вышеприведённых методов не идеален, так как требуется заранее определённые границы поиска искомого корня и при увеличении параметра точности затраты по времени растут слишком быстро. Также были использованы универсальные функции, которые принимают в качестве аргументов указатели на другие функции. Это решение позволяет избежать дублирования кода.

# Список использованных источников

1) Математический энциклопедический словарь. — М.: «Сов. энциклопедия », 1988. — С. 847.

2) Волков Е. А. Численные методы. — М. : Физматлит, 2003.

3) Максимов Ю. А. Алгоритмы линейного и дискретного программирования – М.: МИФИ, 1980.