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: Немкова А.Р.

Группа: М8О-108Б-22,

№ по списку 14

Руководитель: Сахарин Н.А.

Дата:

Оценка:

Подпись преподавателя:

Москва, 2023

**Содержание**

ЗАДАЧА ……………………………………………………………………..... 3

ВАРИАНТ …………………………………………………………………..... 3

ОБЩИЙ МЕТОД РЕШЕНИЯ .……………………………………………... 3

ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ ……………………………………. 4

ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ ……………………………………. 4

ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ ……………………………… 5

ОПИСАНИЕ ПЕРЕМЕННЫХ, КОНСТАНТ И ПОДПРОГРАММ ………. 5

ПРОТОКОЛ …………………………………………………………………. 7

ВХОДНЫЙ ДАННЫЕ ……………………………………………………... 11

ВЫХОДНЫЕ ДАННЫЕ ………………………………………………...…. 11

ВЫВОД ...…………………………………………………………………… 11

**1. Задача**

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными способами(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 и графическую иллюстрацию.

**2. Вариант**

| № | Уравнение | Отрезок, содержащий корень | Базовый метод | Приближенное значение корня |
| --- | --- | --- | --- | --- |
| 14 |  | [1, 2] | дихотомии | 1.0769 |
| 15 |  | [1, 2] | итераций | 1.2388 |

Таблица 1а. Вариант №14, 15

**3. Общий метод решения**

Вычисление приближенного значений функций при помощи метода дихотомии, метода итераций и метода Ньютона.

Метод дихотомии - деление отрезка пополам, учитывае что знак функции должен быть разным. До тех пор, пока длина отрезка не будет меньше значения машинного эпсилон, процесс деления будет выполняться. Приближенное значение корня к моменту окончания итерационного процесса будет находиться примерно в середине заданного отрезка.

Метод итераций заключается в замене исходного уравнения F(x) = 0 уравнением f(x) = x. Начальным приближенным значением корня является середина заданного отрезка. Итерационный процесс имеет вид: . Процесс выполняется пока разность и не станет меньше значения машинного эпсилон.

Метод Ньютона - частный случай метода дихотомии. Итерационный процесс представляет собой: ).

**4. Общие сведения о программе**

Аппаратное обеспечение: домашний ноутбук

Операционная система: Linux Ubuntu, версия 22.04.1 LTS

Язык и система программирования: С, GNU

Местонахождение файлов: /home/anastasia

Компиляция программы: g++ -lm kp4.c

Вызов программы: ./a.out

**5. Функциональное назначение**

Программа предназначена для вычисления приближенного значения трансцендентных алгебраических уравнений с использованием различных численных методов и при помощи встроенных программных функций библиотеки языка Си.

**6. Описание логической структуры**

Программа получает на вход заданный отрезок, находит значение уравнения F(x) = 0 различными численными методами и выводит полученный корень уравнения.

**7. Описание переменных, констант и подпрограмм**

Таблица 1. Описание функций программы

| Функция | Входные аргументы | Описание |
| --- | --- | --- |
| machine\_eps | - | Функция для подсчета машинного ε |
| F1, F2 | double x | Вычисляет значение входной функции, подставляя значение x |
| exp\_x1, exp\_x2 | double x | Функция, вычисляющая выраженный x |
| derivative\_fx1,  derivative\_fx2 | double x | Функция, вычисляющая первую производную f(x) |
| derivative\_F1, derivative\_F2 | double x | Функция, вычисляющая первую производную от заданного уравнения |
| second\_derivative\_F1, second\_derivative\_F2 | double x | Функция, вычисляющая вторую производную от заданного уравнения |
| dichot | double (\*F)(double), double a, double b, double abs\_eps, double otn\_eps | Функция, вычисляющая значение уравнения F(x) = 0 методом дихотомии |
| itter | double (\*derivative\_fx)(double), double (\*exp\_x)(double), double a, double b, double abs\_eps, double otn\_eps | Функция, вычисляющая значение уравнения F(x) = 0 методом итераций |
| newton | double (\*F)(double), double (\*derivative\_F)(double), double (\*second\_derivative\_F)(double), double a, double b, double abs\_eps, double otn\_eps | Функция, вычисляющая значение уравнения F(x) = 0 методом Ньютона |
| result | double (\*F)(double), double (\*derivative\_fx)(double), double (\*derivative\_F)(double), double (\*second\_derivative\_F)(double), double (\*exp\_x)(double), double a, double b, double abs\_eps, double otn\_eps | Функция, выводящая ответ уравнения F(x) = 0, решенное тремя различными способами |

Таблица 2. Описание переменных

| Переменная | Значение |
| --- | --- |
| double abs\_eps | Абсолютное значение машинного эпсилон |
| double otn\_eps | Относительное значение машинного эпсилон |
| double b, а | Границы отрезка |
| long double x | Значение аргумента функции |
| double d | Корень уравнения, подсчитанный методом дихотомии |
| double i | Корень уравнения, подсчитанный методом итераций |
| double n | Корень уравнения, подсчитанный методом Ньютона |

**8. Протокол**

Код программы:

#include <stdio.h>

#include <math.h>

double mashine\_eps() {

double eps = 1.0;

while (1 + eps / 2.0 != 1) {

eps /= 2.0;

}

return eps;

}

double F1(double x) {

return tan(x / 2) - (1 / tan(x / 2)) + x;

}

double F2(double x) {

return 0.4 + atan(sqrt(x)) - x;

}

double exp\_x1(double x) {

return (1 / tan(x / 2)) - tan(x / 2);

}

double exp\_x2(double x) {

return 0.4 + atan(sqrt(x));

}

double derivative\_fx1(double x) {

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

}

double derivative\_fx2(double x) {

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

}

double derivative\_F1(double x) {

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

}

double derivative\_F2(double x) {

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

}

double second\_derivative\_F1(double x) {

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

}

double second\_derivative\_F2(double x) {

return -((1 + 3 \* x) / (sqrt(x) \* (2 \* sqrt(x) + 2 \* sqrt(x) \* fabs(x)) \* (2 \* sqrt(x) + 2 \* sqrt(x) \* fabs(x))));

}

double dichot(double (\*F)(double), double a, double b, double abs\_eps, double otn\_eps) {

double x = (a + b);

if (F(a) \* F(b) <= 0) {

while (fabs(a - b) > fmax(otn\_eps \* fmax(fabs(a), fabs(b)), abs\_eps)) {

x = a + (b - a) / 2;

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

b = x;

} else {

a = x;

}

}

return x;;

} else {

return NAN;

}

}

double itter(double (\*derivative\_fx)(double), double (\*exp\_x)(double), double a, double b, double abs\_eps, double otn\_eps) {

double x = (a + b) / 2;

if (fabs(derivative\_fx(x)) < 1) {

while (fabs(exp\_x(x) - x) >= fmax(otn\_eps \* fmax(fabs(exp\_x(x)), fabs(x)), abs\_eps)) {

x = exp\_x(x);

}

return x;

} else {

return NAN;

}

}

double newton(double (\*F)(double), double (\*derivative\_F)(double), double (\*second\_derivative\_F)(double), double a, double b, double abs\_eps, double otn\_eps) {

double x = (a + b) / 2;

if (fabs(F(x) \* second\_derivative\_F(x)) < pow(derivative\_F(x), 2)) {

while (fabs(F(x) / derivative\_F(x)) > fmax(otn\_eps \* fabs(F(x) / derivative\_F(x)), abs\_eps) ) {

x -= F(x) / derivative\_F(x);

}

return x;

} else {

return NAN;

}

}

void result(double (\*F)(double), double (\*derivative\_fx)(double), double (\*derivative\_F)(double), double (\*second\_derivative\_F)(double), double (\*exp\_x)(double), double a, double b, double abs\_eps, double otn\_eps){

double d = dichot(F, a, b, abs\_eps, otn\_eps);

double i = itter(derivative\_fx, exp\_x, a, b, abs\_eps, otn\_eps);

double n = newton(F, derivative\_F, second\_derivative\_F, a, b, abs\_eps, otn\_eps);

if (d != NAN) {

printf("The root obtained by the dichotomy method:%11.7f\n", d);

} else {

printf("The dichotomy method is not applicable \n");

}

if (i != NAN) {

printf("The root obtained by iteration method:%11.7f\n", i);

} else {

printf("The iteration method is not applicable \n");

}

if (n != NAN) {

printf("The root obtained by Newton's method:%11.7f\n", n);

} else {

printf("Newton's method is not applicable \n");

}

}

int main() {

double a = 1;

double b = 2;

double abs\_eps = mashine\_eps();

double otn\_eps = sqrt(abs\_eps);

printf("Function tan(x / 2) - ctg(x / 2) + x \n");

result(F1, derivative\_fx1, derivative\_F1, second\_derivative\_F1, exp\_x1, a, b, abs\_eps, otn\_eps);

printf("Function 0.4 + atan(sqrt(x)) - x\n");

result(F2, derivative\_fx2, derivative\_F2, second\_derivative\_F2, exp\_x2, a, b, abs\_eps, otn\_eps);

return 0;

}

**9. Входные данные**

Отсутствуют

**10. Выходные данные**

anastasia@anastasia-VirtualBox:~$ g++ -lm kp4.c

anastasia@anastasia-VirtualBox:~$ ./a.out

Function tan(x / 2) - ctg(x / 2) + x

The root obtained by the dichotomy method: 1.0768740

The iteration method is not applicable

The root obtained by Newton's method: 1.0768740

Function 0.4 + atan(sqrt(x)) - x

The root obtained by the dichotomy method: 1.2388400

The root obtained by iteration method: 1.2388400

The root obtained by Newton's method: 1.2388400

**11. Вывод**

В ходе выполнения данной работы были изучены некоторые численные методы решения нелинейных уравнений. Также было освоено практическое применение методов дихотомии, итераций и Ньютона для нахождения приближенного значения корней уравнений F(x) = 0. Было выяснено, что для уравнения неприменим метод итераций, в связи с тем, что производная f(x) = 0 и сама функция f(x) убывает на всем заданном промежутке:

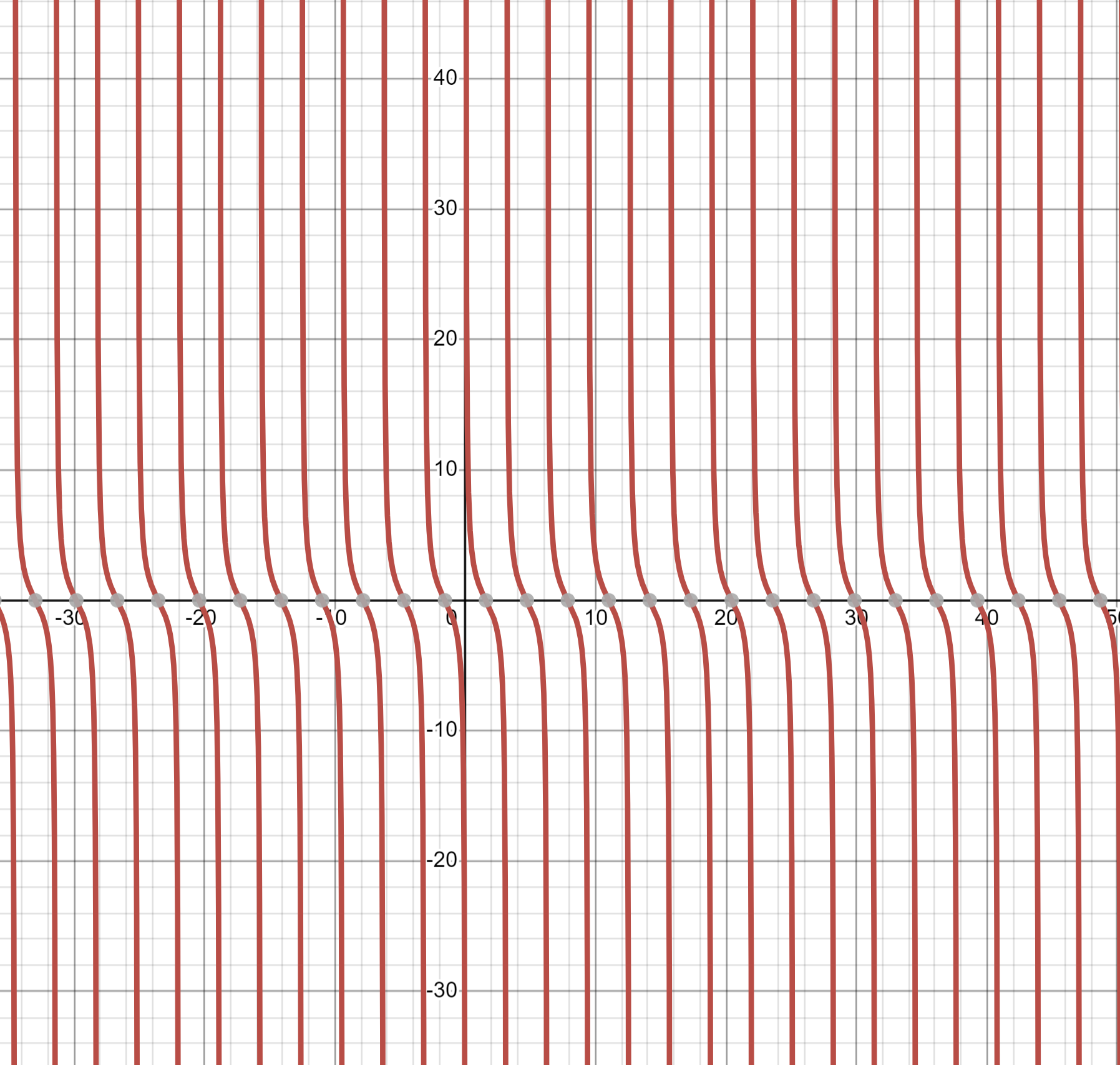


Рис. 1. График функции