**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский Авиационный Институт» (Национальный Исследовательский Университет)**

**Институт№8: «Информационные технологии и прикладная математика»**

**Кафедра: 806 «Вычислительная математика и программирование»**

**Курсовой проект**

по курсу фундаментальная информатика 1 семестра

Задание 3. Процедуры и функции в качестве параметров

Студент: Калюжный М.С.

Группа: М8О-108Б-22

Преподаватель: Сахарин Н.А.

Подпись:

Оценка:

Москва 2022

**СОДЕРЖАНИЕ**

[Задача 3](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc239_2137607056)

[Общий метод решения 4](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc2251_1208348216)

[ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 6](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc2476_1208348216)

[ОПИСАНИЕ ПЕРЕМЕННЫХ И ФУНКЦИЙ 7](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc2478_1208348216)

[Протокол 8](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc2482_1208348216)

[вывод 13](file:///C:\Users\ivank\Downloads\sakost_cw3.odt#__RefHeading___Toc2484_1208348216)

ИСПОЛЬЗОВАННЫЕ ИСТОЧНИКИ…………………………………………..14

2

# 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.

11 вариант задания:

Уравнение: e^x + (1 + e^2x)^0.5 – 2 = 0

Отрезок, содержащий корень: [-1; 0]

Базовый метод: дихотомии

Приближенное значение корня: -0.2877

12 вариант задания:

Уравнение: ln(x) – x + 1.8 = 0

Отрезок, содержащий корень: [2; 3]

Базовый метод: итераций

Приближенное значение корня: 2.8459

3

# 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 .

4

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, служащие им аргументом. Для каждой формулы необходимо запрограммировать исходную функцию. Производную в точку можно принять за дифферинциал от функции в данной точке, который считается по формуле . В теле основной программы делаем только вызов подпрограмм и вывод.

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

5

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**

ОС семейства: UNIX

Наименование: Ubuntu

Версия: Ubuntu 22.04 LTS

Интерпретатор команд: bash

Версия: 5.1-6ubuntu1

Компилятор: gcc

6

**ОПИСАНИЕ ПЕРЕМЕННЫХ И ФУНКЦИЙ**

| Имя | Тип | Описание |
| --- | --- | --- |
| epsilon | ld | Машинное эпсилон |
| a1 | ld | Левая граница 1го отрезка |
| b1 | ld | Правая граница 1го отрезка |
| a2 | ld | Левая граница 2го отрезка |
| b2 | ld | Права граница 2го отрезка |

Таблица 1. Описание переменных

| Название функции | Выходной тип | Входные параметры | Описание |
| --- | --- | --- | --- |
| machineeps | long double | отсутствуют | Функиця, считающая машинное эпсилон |
| dx1(dx2) | long double | pFunc f, long double x | Функция, считающая дифференциал в данной точке |
| sign | long double | long double x | Возвращает 1, если знак x положительный, -1, если отрицательный и 0, если x близок к нулю. |
| dichotomy1 (dichotomy2) | long double | pFunc f, long double a, long double b | Находит корень функции методом дихотомии |
| iteration1 (iteration2) | long double | pFunc f, long double a, long double b | Находит корень функции методом итераций |
| newton1 (newton2) | long double | pFunc f, long double a, long double b | Находит корень функции методом Ньютона |
| func1 | long double | long double x | Считает значение 1 функции в данной точке |
| func2 | long double | long double x | Считает значение 2 функции в данной точке |

Таблица 2. Описание функций

7

**ПРОТОКОЛ**

#include <stdio.h>

#include <math.h>

typedef long double(\*pFunc)(long double);

long double machineeps();

long double dx1(long double x);

long double dx2(long double x);

long double sign(long double x);

long double dichotomy1(pFunc f, long double a, long double b);

long double iteration1(pFunc f, long double a, long double b);

long double newton1(pFunc f, long double a, long double b);

long double dichotomy2(pFunc f, long double a, long double b);

long double iteration2(pFunc f, long double a, long double b);

long double newton2(pFunc f, long double a, long double b);

long double func1(long double x);

long double func2(long double x);

int main(){

long double a1 = -1.l, b1 = 0.l;

long double a2 = 2.l, b2 = 3.l;

printf("dichotomy method result 11 variant: %Lf\n", dichotomy1(func1, a1, b1));

printf("iteration method result 11 variant: %Lf\n", iteration1(func1, a1, b1));

8

printf("newton method result 11 variant: %Lf\n", newton1(func1, a1, b1));

printf("\n");

printf("dichotomy method result 12 variant: %Lf\n", dichotomy2(func2, a2, b2));

printf("iteration method result 12 variant: %Lf\n", iteration2(func2, a2, b2));

printf("newton method result 12 variant: %Lf\n", newton2(func2, a2, b2));

}

long double machineeps(){

long double eps = 1;

while(1 < 1 + eps)

eps /= 2;

return eps;

}

long double dx1(long double x){

return expl(x) + 1 / (2 \* sqrtl(1 + expl(2 \* x))) \* expl(2 \* x) \* 2;

}

long double dx2(long double x){

return ((1 / x) - 1.l);

}

long double sign(long double x){

long double epsilon = machineeps();

return x > epsilon ? 1.l : x < -epsilon ? -1.l : 0.l;

}

9

long double dichotomy1(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

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

x = (a + b) / 2;

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

else b = x;

}

return x;

}

long double iteration1(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

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

x = x - f(x)\*sign(dx1(x));

}

return x;

}

long double newton1(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

while(fabsl(f(x)/ dx1(x)) > epsilon){

x = x - f(x)/dx1(x);

}

return x;

}

10

long double dichotomy2(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

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

a = ceil(a \* 1e+10) / 1e+10;

b = ceil(b \* 1e+10) / 1e+10;

x = (a + b) / 2;

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

else b = x;

}

return x;

}

long double iteration2(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

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

x = (x - f(x)\*sign(dx2(x)));

}

return x;

}

long double newton2(pFunc f, long double a, long double b){

long double epsilon = machineeps();

long double x = (a + b) / 2;

while(fabsl(f(x)/ dx2(x)) > epsilon){

x = x - f(x)/dx2(x);

}

return x;

11

}

long double func1(long double x){

return expl(x) + sqrtl(1.l + expl(2 \* x)) - 2.l;

}

long double func2(long double x){

long double p = logl(x) - x + 1.8l;

p = ceil(p \* 1e+10) / 1e+10;

return p;

}

mimik@mimik-VirtualBox:~$ gcc kp41.c -lm -Wall -pedantic -std=c99 -o c4.out && ./c4.out

dichotomy method result 11 variant: -0.287682

iteration method result 11 variant: -0.287682

newton method result 11 variant: -0.287682

dichotomy method result 12 variant: 2.845868

iteration method result 12 variant: 2.845868

newton method result 12 variant: 2.845868

12

# ВЫвод

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

13

# Использованные источники

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.

14