Московский Авиационный Институт
(Национальный исследовательский университет)

Факультет информационных технологий и прикладной математики
Кафедра №806 Вычислительная математика и программирование

Курсовой проект

по курсам
«Фундаментальная информатика», «Архитектура компьютера и информационных систем»
I семестр

Задание 3

Вещественный тип. Приближенные вычисления.

Табулирование функций.

Студент: Ибрагимов Д. М.

Группа: М8О-108Б-22

Руководитель: Сахарин Н. С.

Оценка:

Дата:

Подпись преподавателя:

Москва, 2022

СОДЕРЖАНИЕ

Задание

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.

Вариант:

№	Уравнение	Отрезок, содержащий корень	Базовый метод	Приближенное значение корня
9	$x^2 - \ln(1+x) - 3 = 0$	[2, 3]	итераций	2.0267
10	$2x \cdot sinx - cosx = 0$	[0.4, 1]	Ньютона	0.6533

Общий метод решения

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

Рассматривается уравнение вида $F(x)= 0$. Предполагается, что функция $F(x)$ достаточно гладкая, монотонная на этом отрезке и существует единственный корень уравнения $x^* \in [a, b]$. на отрезке $[a, b]$ ищется приближенное решение x с точностью ε, т.е. такое, что $|x - x^*| < \varepsilon$.

Метод дихотомии - деление отрезка пополам с учётом того, что знак функции на концах отрезка должен быть разным: $F(a) * F(b) < 0$. До тех пор, пока длина отрезка не будет меньше значения ε, 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 $x^{(0)} = (a + b)/2$. Итерационный процесс имеет вид: $x^{(k+1)} = f(x^{(k)})$. Процесс выполняется пока $|x^{(k+1)} - x^{(k-1)}| < \varepsilon$

Метод Ньютона - частный случай метода итераций. Итерационный процесс представляет собой: $x^{(k+1)} = x^{(k)} – F(x^{(k)})/F'(x^{(k)})$.

Оборудование

Процессор	AMD Ryzen 5 5600H (12) @ 3.600GHz
ОП	32GiB 3200 MHz LPDDR4
SSD	512 GiB
Монитор	27-дюймовый (1920 x 1080)
Графика	AMD Radeon™ RX 6600M 2177 MHz 8GiB GDDR6

Таблица 2 – оборудование

Программное обеспечение

Операционная система семейства	Microsoft Windows 11 Pro
Компилятор	GNU Compiler Collection
Текстовый редактор	Visual Studio Code версия 1.73.0

Таблица 3 – программное обеспечение

Функциональное назначение

Программа предназначена для высокоточного вычисления вещественного значения трансцедентных функций в алгебраической форме с использованием ряда Тейлора и при помощи встроенных программных функций библиотеки языка Си.

Описание логической структуры

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

Описание переменных, констант и подпрограмм

Функция	Входные аргументы	Описание
epsilon	double x	Функция для подсчета машинного ε
F9, F10	double x	Вычисляет значение входной функции, подставляя значение x
F9_x, F10_x	double x	Функция, вычисляющая выраженный x
F9_first_derivative, F10_first_derivat	double x	Функция, вычисляющая первую производную от заданного уравнения

ive		
F9_second_derivative, F10_second_derivative	double x	Функция, вычисляющая вторую производную от заданного уравнения
F9_x_first_derivative, F10_x_first_derivative	double x	Функция, вычисляющая первую производную от уравнения, в котором выражен x
dichotomy	double F(double), double a, double b	Функция, вычисляющая значение уравнения F(x) = 0 методом дихотомии
iterations	double F_x(double), double F_x_first_derivative(double), double a, double b	Функция, вычисляющая значение уравнения F(x) = 0 методом итераций
newton	double F(double), double F_first_derivative(double), double F_second_derivative(double), double a, double b	Функция, вычисляющая значение уравнения F(x) = 0 методом Ньютона

Таблица 1. Описание функций программы

Переменная	Значение
double abs_eps	Абсолютный эпсилон
double relative_eps	Относительный эпсилон
double a, b	Границы отрезка
long double x	Значение аргумента функции

Таблица 2. Описание переменных

Протокол

Код программы:

```
#include <stdio.h>
#include <math.h>
typedef double dbl;
dbl epsilon() {
    dbl eps = 1.0;
    while (1 + eps / 2.0 != 1) eps /= 2.0;
    return eps;
}
dbl F9(dbl x) {
    return x*x - log(1 + x) - 3;
}
dbl F9_ot_x(dbl x) {
    return (log(1 + x) + 3)/x;
}
dbl F9_ot_x_first_derivative(dbl x) {
    return (-2*x - (1 + x)*log(1+x) - 3)/(x*x + x*x*x);
}
dbl F9_first_derivative(dbl x) {
    return 2*x - 1/(1+x);
}
dbl F9_second_derivative(dbl x) {
    return (2*x*x + 4*x + 3)/((1+x)*(1+x));
}
dbl F10(dbl x) {
    return 2*x*sin(x) - cos(x);
}
dbl F10_ot_x(dbl x) {
    return 1/(2*tan(x));
}
dbl F10_ot_x_first_derivative(dbl x) {
    return (-1/(sin(x)*sin(x)))/2;
}
dbl F10_first_derivative(dbl x) {
    return 3*sin(x) + 2*x*cos(x);
}
dbl F10_second_derivative(dbl x) {
    return 5*cos(x) - 2*x*sin(x);
}
dbl dichotomy(dbl (*F)(dbl), dbl a, dbl b, dbl relative_eps, dbl abs_eps) {
    dbl x = (a + b) / 2;
```

```
    if (F(a) * F(b) >= 0){
        return NAN;
    }
    while (fabs(a - b) > fmax(relative_eps * fabs(x), abs_eps)) {
        x = (a + b) / 2;
        if (F(x) * F(a) < 0) {
            b = x;
        }
        else {
            a = x;
        }
    }
    return x;
}
dbl iterations(dbl (*F_x)(dbl), dbl (*F_x_first_derivative)(dbl), dbl a, dbl b, dbl relative_eps,
dbl abs_eps) {
    dbl x = (a + b) / 2;
    if (fabs(F_x_first_derivative(x)) >= 1) {
        printf("\n x - %lf; f -%lf debug printami\n", x, F_x_first_derivative(x));
        return NAN;
    }
    while (fabs(F_x(x) - x) >= fmax(relative_eps * fabs(x), abs_eps)) x = F_x(x);
    return x;
}
dbl newton(dbl (*F)(dbl), dbl (*F_first_derivative)(dbl), dbl (*F_second_derivative)(dbl), dbl a,
dbl b, dbl relative_eps, dbl abs_eps) {
    dbl x = (a + b / 2);
    if (fabs(F(x) * F_second_derivative(x)) >= (F_first_derivative(x) * F_first_derivative(x))) {
        return NAN;
    }
    while (fabs(F(x) / F_first_derivative(x)) > fmax(relative_eps * fabs(x), abs_eps)) {
        x -= F(x) / F_first_derivative(x);
    }
    return x;
}
void result(dbl d, dbl i, dbl n) {
   if (d != NAN) printf("Dichotomy method: %.10f\n", d);
   else printf("The dichotomy method isn't suitable\n");
   if (i != NAN) printf("Iterations method: %.10f\n", i);
   else printf("The iterations method isn't suitable\n");
   if (n != NAN) printf("Newton's method: %.10f\n", n);
   else printf("The Newton's method isn't suitable\n");
```

```
int main() {
    dbl abs_eps = epsilon();
    dbl relative_eps = sqrt(abs_eps);
    dbl a9 = 2, b9 = 3, a10 = 0.4, b10 = 1;
    dbl d9 = dichotomy(F9, a9, b9, relative_eps, abs_eps);
    dbl i9 = iterations(F9_ot_x, F9_ot_x_first_derivative, a9, b9, relative_eps, abs_eps);
    dbl n9 = newton(F9, F9_first_derivative, F9_second_derivative, a9, b9, relative_eps,
abs_eps);
    printf("Machine epsilon for long double = %.16e\n", abs_eps);
    printf(" x^2 - ln(1 + x) - 3 = 0 \n");
    result(d9, i9, n9);
    printf("\n\n");
    dbl d10 = dichotomy(F10, a10, b10, relative_eps, abs_eps);
    dbl i10 = iterations(F10_ot_x, F10_ot_x_first_derivative, a10, b10, relative_eps, abs_eps);
    dbl n10 = newton(F10, F10_first_derivative, F10_second_derivative, a10, b10, relative_eps,
abs_eps);
    printf("Machine epsilon for long double = %.16e\n", abs_eps);
    printf(" 2*x*sin(x) - cos(x) = 0 \n");
    result(d10, i10, n10);
    return 0;
}
```

Входные данные

Отсутствуют.

Выходные данные

```
PS C:\Users\lock_R\Documents\cc> gcc .\kp4.c -pedantic -Wall
PS C:\Users\lock_R\Documents\cc> .\a.exe
Machine epsilon for long double = 2.2204460492503131e-16
 x^2 - ln(1 + x) - 3 = 0
Dichotomy method: 2.0266892612
Iterations method: 2.0266892794
Newton's method: 2.0266892732



 x - 0.700000; f --1.204772 debug printami
Machine epsilon for long double = 2.2204460492503131e-16
 2*x*sin(x) - cos(x) = 0
Dichotomy method: 0.6532711893
Iterations method: nan
Newton's method: 0.6532711871
```

```
PS C:\Users\lock_R\Documents\cc> gcc .\kp4.c -pedantic -Wall
PS C:\Users\lock_R\Documents\cc> .\a.exe
Machine epsilon for long double = 2.2204460492503131e-16
Function x^2 - ln(1 + x) - 3
The root obtained by the dichotomy method: 2.0266892612
The root obtained by the iterations method: 2.0266892794
The root obtained by the Newton's method: 2.0266892732
```

Рис.1 – выходные данные

Заключение

В результате выполнения данного курсового проекта были получены навыки работы с процедурами и функциями в качестве параметров. Было изучено вычисление машинного эпсилона и различных численных методов, таких как: метод дихотомии (половинного деления), метод итераций и метод Ньютона. Оценивая полученные данные, можно сказать, что каждый из методов неидеален, так как для поиска корня необходимо знать точные границы отрезка. Также при увеличении точности вычислений затраты по времени быстро увеличиваются.

Благодаря использованию универсальных функций, которые принимают в качестве аргументов указатели на другие функции, удалось избежать дублирования кода.