МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт»
(Национальный Исследовательский Университет)

Институт: №8 «Компьютерные науки и прикладнаяматематика»
Кафедра: 806 «Вычислительная математика и программирование»

Курсовая работа
по курсу
«Фундаментальная информатика»
I семестр
Задание 4
«Процедуры и функции в качестве параметров»

Группа	М8О-109Б-22
Студент	Концебалов О.С.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2022

Постановка задачи

Составить программу на 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 и графическую иллюстрацию, например, с использованием gnuplot.

Вариант 20:

Функция:

$$0{,}1x^2 - x\ln x = 0$$

Отрезок содержащий корень: [1, 2]

Метод Ньютона

Вариант 21:

Функция:

$$\operatorname{tg} x - \frac{1}{3}\operatorname{tg}^3 x + \frac{1}{5}\operatorname{tg}^5 x - \frac{1}{3} = 0$$

Отрезок содержащий корень: [0, 0.8]

Метод Дихотомии

Теоретическая часть

Метод Ньютона

Метод Ньютона является частным случаем метода итераций.

Условие сходимости метода: $|F(x)\cdot F''(x)| < (F'(x))^2$ на отрезке $[a,b]$.

Итерационный процесс: $x^{(k+1)} = x^{(k)} - F(x^{(k)})/F'(x^{(k)})$.

Метод дихотомии (половинного деления)

Очевидно, что если на отрезке $[a,b]$ существует корень уравнения, то значения функции на концах отрезка имеют разные знаки: $F(a)\cdot F(b) < 0$. Метод заключается в делении отрезка пополам и его сужении 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 $a^{(0)} = a$, $b^{(0)} = b$. Далее вычисления проводятся по формулам: $a^{(k+1)} = (a^{(k)} + b^{(k)})/2$, $b^{(k+1)} = b^{(k)}$, если $F(a^{(k)})\cdot F((a^{(k)} + b^{(k)})/2) > 0$; или по формулам: $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = (a^{(k)} + b^{(k)})/2$, если $F(b^{(k)})\cdot F((a^{(k)} + b^{(k)})/2) > 0$.

Процесс повторяется до тех пор, пока не будет выполнено условие окончания $|a^{(k)} - b^{(k)}| < \varepsilon$.

Приближенное значение корня к моменту окончания итерационного процесса получается следующим образом $x^* \approx (a^{(конечное)} + b^{(конечное)})/2$.

Описание алгоритма

Составляю программу для нахождения корня с помощью метода Ньютона и проверяю найденный корень, либо вывожу, что метод не применим. Аналогично поступаю и с методом дихотомии.

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
LDBL_EPSILON	long double	Машинный эпсилон 1.0842e-19
step	long double	Шаг для проверки
a	long double	Левая граница отрезка
b	long double	Правая граница отрезка
x_0	long double	значение x
x	long double	Следующее значение x

Исходный код программы:

Вариант 20:

```
#include <stdio.h>
#include <math.h>
#include <float.h>

long double function(long double x){
    return (0.1 * x * x) - (x * logbl(x));
}

long double first_derivative(long double x){
    return (0.2 * x) - logbl(x) - 1;
}

long double second_derivative(long double x){
    return 0.2 - (1 / x);
}

int check_convergence(long double a, long double b){
    long double step = (b - a) / 10000;
    for (long double x = a; x <= b; x += step){
        if (fabsl(function(x) * second_derivative(x)) < first_derivative(x) *
first_derivative(x)){
            return 0;
        }
    }
    return 1;
}

long double find_x(long double x_0, long double x){
    while (fabsl(x - x_0) >= LDBL_EPSILON){
        printf("%Lf %Lf", x_0, x);
        x_0 = x;
        x = x_0 - function(x_0) / first_derivative(x_0);
    }
    return x;
}

int main() {
    long double a = 1;
    long double b = 2;

    long double x_0 = (a + b) / 2;
    long double x= x_0 - function(x_0) / first_derivative(x_0);

    if (check_convergence(a, b) == 1){
        printf("Method is convergent\n");
        printf("x = %Lf", find_x(x_0, x));
        printf("The value of the function for such x: %Lf", function(x));
    }
    else{
        printf("\nMethod doesn't convergent\n");
    }

    return 0;
}
```

Вариант 21:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <float.h>

long double function(long double x){
    return tanl(x) - ((1.0 / 3.0) * powl(tanl(x), 3)) + ((1.0 / 5.0) * powl(tanl(x), 5)) -
(1.0 / 3);
}

int main(){
    long double a = 0;
    long double b = 0.8;

    printf("\nDichotomy Method");

    if (function(a) * function(b) > 0){
        printf("No roots on the segment");
        exit(0);
    }

    while (fabsl(a - b) > LDBL_EPSILON){
        if (function(a) * function((a + b) / 2) > 0){
            a = (a + b) / 2;
            b = b;
        }
        else if (function(b) * function((a + b) / 2) > 0){
            a = a;
            b = (a + b) / 2;
        }
    }

    printf("\nx = %Lf\n", (a + b) / 2);
    printf("The value of the function for such x: %Lf\n", function((a + b) / 2));

    return 0;
}
```

Входные данные

Нет

Выходные данные

Программа должна вывести для первого уравнения сходится метод или нет. В случае, если сходится, вывести его значение. Для второго уравнения вывести найденный корень и значение уравнения при таком корне.

Тест №1

Вариант 20:

```
Newton method
Method doesn't convergent
```

Вариант 21:

```
Dichotomy Method
x = 0.333255
The value of the function for such x: 0.000000
```

Вывод

В работе описаны и использованы различные численные методы для решения трансцендентных алгебраических уравнений. Даны обоснования сходимости и расходимости тех или иных методов. Имплементирована функция вычисления производной от заданной функции в точке. На основе алгоритма составлена программа на языке Си, сделана проверка полученных значений путем подстановки. Работа представляется довольно полезной для понимания принципов работы численных методов и способов их имплементации.

Список литературы

1. Численное дифференецирование – URL:

   Численное дифференцирование — Википедия (wikipedia.org)

2. Конечная разность – URL:

   Численное дифференцирование — Википедия (wikipedia.org)