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	Горохов М.С.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2022

Постановка задачи

Составить программу на Си с процедурами решения трансцендентных алгебраических уравнений резличными 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 19:

Функция:

$$x - \frac{1}{3 + \sin 3.6x} = 0$$

Отрезок содержащий корень: [0, 0,85]

Метод итераций

Вариант 20:

Функция:

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

Отрезок содержащий корень: [1, 2]

Метод Ньютона

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 итераций

Идея метода заключается в замене исходного уравнения $F(x)=0$ уравнением вида $x=f(x)$.

Достаточное условие сходимости метода: $|f'(x)|<1, x\in[a,b]$. 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.

Начальное приближение корня: $x^{(0)} = (a+b)/2$ (середина исходного отрезка).

Итерационный процесс: $x^{(k+1)} = f(x^{(k)})$.

Условие окончания: $\left|x^{(k)} - x^{(k-1)}\right| < \varepsilon$.

Приближенное значение корня: $x^* \approx x^{(конечное)}$.

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 итераций.

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
inf	long double	Малое значение

Исходный код программы:

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


long double derive(long double (*f)(long double), long double x){
    long double inf = 1e-6;
    long double ans = (f(x + inf) - f(x)) / inf;
    return ans;
}

long double func_1(long double x) {
    return 1 / (3 + sinl(3.6 * x));
}

long double func_1_hands(long double x) {
    return -(18*cosl(3.6*x)/(5*powl((3+sinl(3.6*x)),2)));
}

int is_iteration(long double (*f)(long double), long double (*hands)(long double), long
double a, long double b) {
    long double step = (b-a)/100000;
    for (long double x=a; x<=b; x+=step) {
        if (derive(f, x) >= 1 || hands(x) >= 1) {
            return 0;
        }
    }
    return 1;
}

long double iteration(long double (*f)(long double), long double a, long double b) {
    long double x = (a + b) / 2.0;
    long double x1 = f(x);
    while (fabsl(x1 - x) >= LDBL_EPSILON) {
        x = x1;
        x1 = f(x);
    }
    return x1;
}

long double func_2(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(func_2(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){
```

```c
    while (fabsl(x - x_0) >= LDBL_EPSILON){
        printf("%Lf %Lf", x_0, x);
        x_0 = x;
        x = x_0 - func_2(x_0) / first_derivative(x_0);
    }
    return x;
}

int main() {
    //19
    long double a = 0.0, b = 0.85;
    printf("Function: x - 1 / (3 + sin(3.6x)) = 0\nMethod: iterations.\n");
    if (is_iteration(func_1, func_1_hands, a, b)) {
        printf("Method is convergent.\n");
        printf("x = %Lf", iteration(func_1, a, b));
    } else {
        printf("Method doesn't convergent.\n");
    }
    // 20
    a = 1;
    b = 2;

    long double x_0 = (a + b) / 2;
    long double x= x_0 - func_2(x_0) / first_derivative(x_0);

    printf("\nFunction: (0.1 * x * x) - (x * logbl(x)) = 0\nMethod: Newton.\n");

    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", func_2(x));
    }
    else{
        printf("Method doesn't convergent\n");
    }
    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

```
Function: x - 1 / (3 + sin(3.6x)) = 0
Method: iterations.
Method is convergent.
x = 0.262441
Function: (0.1 * x * x) - (x * logbl(x)) = 0
Method: Newton.
Method doesn't convergent
```

Вывод

В работе описаны и использованы различные численные методы для решения трансцендентных алгебраических уравнений. На основе программы на языке Си проверена применимость методов для конкретных функций. Работа является 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)