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

Институт: №8 «Компьютерные науки и прикладная математика»
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	Фомин И.Д.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2023

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 24:

Функция:

$$\cos\frac{2}{x} - 2\sin\frac{1}{x} + \frac{1}{x} = 0$$

Отрезок содержащий корень: [1,2]

Метод дихотомии

Вариант 25:

Функция:

$$\sqrt{1-0{,}4x^2} - \arcsin x = 0$$

Отрезок содержащий корень: [0,1]

Метод итераций

Теоретическая часть

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 $\left|a^{(k)}-b^{(k)}\right|<\varepsilon$.

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

Метод итераций

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

Достаточное условие сходимости метода: $\left|f'(x)\right|<1, x\in[a,b]$. 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 итераций 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
newX	long double	Новый x
a	long double	Левая граница отрезка
b	long double	Правая граница отрезка
x0	long double	значение x из таблицы
x	long double	рассчитанное значение x

Исходный код программы: (вар. 24)

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

long double function(long double x){
    return cosl(2 / x) - 2 * sinl(1 / x) + 1 / x;
}

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

    printf("\nDichotomy Method");

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

    for (int i = 0; i < 100; ++i){
        if (fabsl(a - b) < LDBL_EPSILON){
            printf("\n%Lf %Lf", a, b);
            break;
        }

        if (function(a) * function((a + b) / 2) > 0)
            a = (a + b) / 2;

        else if (function(b) * function((a + b) / 2) > 0)
            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;
}
```

Исходный код программы: (вар. 25)

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

// Решить методом итераций
long double funcVar25(long double x) {
    // sqrt(1-0.4x^2) - arcsin(x) = 0
    // x = arcsin(x) / sqrt(1/(x*x) - 0.4)
    // return sqrtl(1 - 0.4 * x * x) - asinhl(x);
    return asinhl(x) / sqrtl(1 / (x * x) - 0.4);
}

long double firstDerivative25(long double x) {
    long double a = asinhl(x) / (x * x * x) + (1 / x * x - 0.4) / sqrtl(1 - x * x);
    long double sub = (1 / x * x - 0.4);
```

```
    long double b = sub * sqrtl(sub);
    return a / b;
}

// Для funcVar25
long double iterativeSearch(double_t left, double_t right) {
    long double x;
    long double newX;

    if (fabsl(firstDerivative25(left)) < 1)
        x = left;
    else if (fabsl(firstDerivative25(right)) < 1)
        x = right;
    else {
        x = (left + right) / 2;
        printf("ERROR. Func for #25 Does not converge.\n");
        return x;
    }

    newX = firstDerivative25(x);
    while (fabsl(newX - x) > LDBL_EPSILON) {
        x = newX;
        newX = firstDerivative25(x);
    }

    return x;
}

int main() {
    long double x25 = 0.7672;

    long double ans25 = iterativeSearch(0, 1);

    printf("#25 - x0: %Lf, x: %Lf", x25, ans25);

    return 0;
}
```

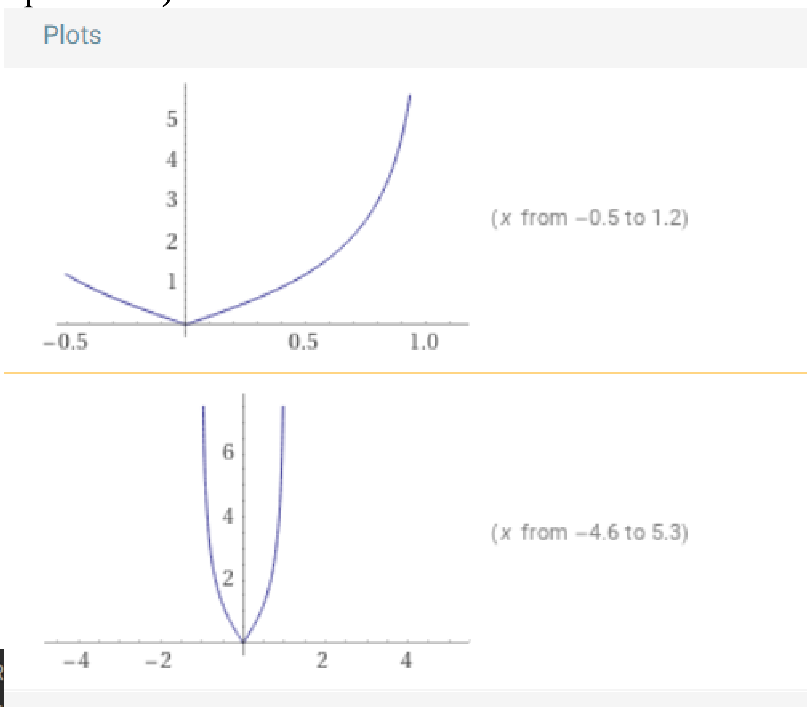
Входные данные

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.

Для варианта 25 вычислить приближенное значение, пользуясь итерациями, не получится т.к. функция на отрезке [0, 1] расходится (т.е. её производная < 1).
Лучше всего это заметно в визуальном представлении функции. (y -> беск. при x -> 1).




```
ERROR
Process finished with exit code 0
```

```
Dichotomy Method
x = 1.875617
The value of the function for such x: -0.000000
```

```
ERROR. Func for #25 Does not converge.
#25 - x0: 0.767200, x: 0.500000
Process finished with exit code 0
```

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)