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-108Б-22
Студент	Иванов А.К.
Преподаватель	Сахарин Н.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.

Функции:

10	$2x \cdot \sin x - \cos x = 0$	[0.4, 1]	Ньютона	0.6533
11	$e^x + \sqrt{1+e^{2x}} - 2 = 0$	[-1, 0]	дихотомии	-0.2877

Теоретическая часть

Метод дихотомии:

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 Ньютона

Метод Ньютона является частным случаем метода итераций.

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

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

Более совершенное с программистской точки зрения решение задачи может быть получено с помощью изучаемого в курсе «Языки программирования» (II семестр) процедурного типа данных. В этом случае различные уравнения и методы как переменные процедурного типа подставляются в качестве фактических параметров соответствующих подпрограмм. Решение задачи на языке Си, фактически базирующееся на указателях на функции, близко к этому.

Численное дифференцирование

Так как возможности компьютера не позволяют проводить вычисления с бесконечно малыми, для расчетов будем брать просто очень маленькие значения. Так, для вычисления производной через предел возьмем prib равное 1e-6.

Описание алгоритма

Делаем функцию для высчитывания корня методом дихотомии. После чего выводим его значение. Для метода же Ньютона нужно найти производные первого и второго порядка. Прописываем команду проверки на сходимость, а затем и сам способ нахождения корня (условия прописаны в самом задании). Выводим результаты.

Используемые переменные

Название переменной	**Тип переменной**	**Смысл переменной**
step	long double	Шаг при проверке на сходимость
x0	long double	Временная переменная для хранения значения x при расчете методом
x1	long double	Результат работы методов
a	long double	Начало отрезка
b	long double	Конец отрезка

Исходный код программы:

Вариант 10:

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

long double var10(long double x){
return (2.0 * x * sin(x) - cos(x));
}

long double var10_pr1(long double x){
    return ( 2.0 * sin(x) + 2.0 * x * cos(x) + sin(x));
}

long double var10_pr2(long double x){
return (-2.0 * x * sin(x) + 5 * cos(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(var10(x) * var10_pr2(x)) < var10_pr1(x) * var10_pr1(x)){
return 0;
        }     }
return 1;
}

long double find_x(long double x0, long double x){
while (fabsl(x - x0) >= LDBL_EPSILON){
```

```
        printf("%Lf %Lf", x0, x);
       x0 = x;
        x = x0 - var10(x0) / var10_pr1(x0);
    }
return x;
}

int  main()  {         long
double a = 0.4;     long
double b = 1;

    long double x0 = (a + b) / 2;
    long double x= x0 - var10(x0) / var10_pr1(x0);

    printf("\nNewton method\n");

    if (check_convergence(a, b) == 1){
printf("Method is convergent\n");         printf("x
= %Lf", find_x(x0, x));
        printf("The value of the function for such x: %Lf", var10(x));
    }
else{
        printf("Method doesn't convergent\n");
    }

    return 0;
}
```

<u>Вариант 11:</u>

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

//Метод дихотомии

long double var_11(long double x){
```

```
    // exp(x) + sqrt(1.0 + exp(2.0 * x)) - 2.0
    return (exp(x) + sqrt(1.0 + exp(2.0 * x)) - 2.0);
}

long double dixit(long double (*f)(long double), long double a,
longdouble b){
    long double
    c; long
    double ans;
    while (fabsl(a - b) >
        LDBL_EPSILON) {c = (a + b) /
        2.0;
        if (f(c) * f(a) <
            0) {b = c;
        } else {
            a = c;
        }
    }
    ans = c;
    return
    ans;
}
int main() {
    long double a
    = -1;
    long double b
    = 0;
    printf("variant 11: exp(x) + sqrt(1.0 + exp(2.0 * x)) - 2.0
Method:dixit.\n%.19Lf", dixit(var_11, a, b));

    printf("\n\n");
}
```

Входные данные:

Нет

Выходные данные:

Программа должна вывести для второго уравнения сходится метод или нет. В случае, если сходится, вывести его значение. Для первого уравнения вывести его значение.

Тест №1

```
variant 11: exp(x) + sqrt(1.0 + exp(2.0 * x)) - 2.0 Method: dixit.
-0.2876820724517799023
```

```
Newton method
Method doesn't convergent

Process returned 0 (0x0)   execution time : 0.018 s
Press any key to continue.
```

Вывод

В работе описаны и использованы различные численные методы для решения трансцендентных алгебраических уравнений. Даны обоснования сходимости и расходимости тех или иных методов. Данное задание познакомило меня с такими методами решения, как метод дихотомии, итерации и Ньютона. Так же цель лабораторной работы достигнута, т.к. я составила программу на языке Си с процедурами решения трансцендентных алгебраических уравнений.

Источники

1. Конечные разности. Разделенные разности. // Studme.org URL: https://studme.org/288569/informatika/konechnye_raznosti_razdelennye_raznosti (дата обращения 30.12.2022)

2. Методы численного дифференцирования функций // aco.info.ru URL: //http://aco.ifmo.ru/el_books/numerical_methods/lectures/glava1.html (дата обращения 30.12.2022)
3. Метод Ньютона // Алгоритмика URL: https://ru.algorithmica.org/cs/numerical/newton/ (дата обращения 30.12.2022)