Московский авиационный институт
(национальный исследовательский университет)

Факультет № 8 «Прикладная математика и информатика»

Кафедра 806 «Вычислительная математика и программирование»

КУРСОВОЙ ПРОЕКТ

по дисциплине «Фундаментальная информатика»

1 семестр

на тему “Процедуры и функции в качестве параметров”

Студент:	Воропаев И.К.
Группа:	М8О-109Б-22
Преподаватель:	Сысоев М.
Подпись:	
Оценка:	

Москва, 2023

Постановка задачи

Составить программы на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и половинного деления — дихотомия). 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 13

Уравнение $x \cdot \mathrm{tg}\, x - \frac{1}{3} = 0$

Отрезок [0.2, 1]
Приближённое значение корня 0.5472

Вариант 14

Уравнение $\mathrm{tg}\frac{x}{2} - \mathrm{ctg}\frac{x}{2} + x = 0$

Отрезок [0, 2]
Приближённое значение корня 1.0769

Теоретическая часть

1. 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 $x^{*}\approx(a^{(конечное)}+b^{(конечное)})/2$.

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: $x^{*}\approx x^{(конечное)}$.

3. Метод Ньютона.

Метод Ньютона является частным случаем метода итераций.

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

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

Проверка на условие сходимости:

Наличие корня на отрезке можно проверить из способа Дихтомии. Для обоих вариантов проверяем знаки значений функции на концах отрезка. Если они разные, то программа пойдёт дальше.

Алгоритм выполнения программы:

Для начала нужно вычислить машинное эпсилон. Его мы вычисляем по аналогии с КП3. Затем реализуем функции f(x), f'(x) и g(x) для 13 и 14 вариантов. В них будем передаваться значение x, а возвращать функции будут своё значение от аргумента.

Первым идёт метод Ньютона. Для его рассчёта как раз и понадобилась проивзодная. Реализую строго согласно иструкции (см. Листинг). Передаём значение функции и производной для подсчёта нового икс. Подсчёт корня остановится, когда разница по модулю между следующим и предыдущим иксами будем меньше машинного эпсилон.

Для метода Дихтомии нам нужно лишь значение функции. Передаём его в цикл и считаем новые границы до тех пор, пока модуль разницы между границами отрезка не будет меньше машинного эпсилон, то есть по сути, пока границы не совпадут.

КОД ПРОГРАММЫ

```
#include <stdlib.h>
#include <stdio.h>
#include <math.h>
#include <limits.h>
#include <assert.h>

long double f(long double x) {
    return (x * tanl(x)) - (1.0L / 3.0L);
}

long double f_derivative1(long double x) {
    return (cosl(x) * sinl(x) + x)
            / powl(cosl(x), 2.0L);
}

long double Newton(long double left,
                   long double right) {
    long double x = (left + right) / 2.0L,
        machineEpsilon = 1.0L, previous_x;
    while (1.0L + machineEpsilon > 1.0L) {
        machineEpsilon /= 2.0L;
    }
    do {
        previous_x = x;
        x -= f(x) / f_derivative1(x);
    }
    while (fabsl(x - previous_x) > machineEpsilon);
    return x;
}

long double g(long double x) {
    return tanl(x / 2.0L) -
            (1.0L / tanl(x / 2.0L)) + x;
}

long double dichotomy(long double left,
                      long double right) {
    long double machineEpsilon = 1.0;
    while (1.0L + machineEpsilon > 1.0L) {
        machineEpsilon /= 2.0L;
    }
    while (fabsl(right - left) > machineEpsilon) {
        if ((g(left) * g(left + right) / 2.0L) > 0.0L) {
            left = (left + right) / 2.0L;
            continue;
```

```
        }
        right = (left + right) / 2.0L;
    }
    return left + right;
}

int main () {
    assert(f(0.2L) * f(1.0L) < 0);
    assert(g(0.0L) * g(2.0L) < 0);
    printf("Variant 13: %.4Lf\n", Newton(0.2L, 1.0L));
    printf("Variant 14: %.4Lf\n",
    dichotomy(0.0L, 2.0L));
    return 0;
}
```

Протокол

```
mariavoropaeva@MacBook-Pro-Ivan Project 1 % ./kp4.out
Variant 13: 0.5472
Variant 14: 1.0769
mariavoropaeva@MacBook-Pro-Ivan Project 1 %
```

Вывод

В ходе работы я научился находить приблеженное значение корня на заданном отрезке при помощи метода итераций (метод Ньютона - частный случай итераций) и дихотомии, реализовав данные алгритмы на Си, что поможет мне в будущем работать с функциями на более высоком уровне.

Список литературы

1. Метод бисекции [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_бисекции
2. Метод простой итерации [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_простой_итерации
3. Метод Ньютона [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Метод_Ньютона