МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский Авиационный Институт
(Национальный Исследовательский Университет)»
ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И ПРИКЛАДНОЙ МАТЕМАТИКОЙ
Кафедра вычислительной математики и программирования

Курсовой проект
по курсу вычислительные системы 1 семестра
Задание 4. Процедуры и функции в качестве параметров

Студент:	Саженов К.С.
Группа:	М80 - 108Б - 19
Преподаватель:	Поповкин А.В.
Подпись:	
Оценка:	

Москва, 2019

СОДЕРЖАНИЕ

ЗАДАЧА

Составить программу на языке 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.

21 вариант

21	$\operatorname{tg} x - \frac{1}{3}\operatorname{tg}^3 x + \frac{1}{5}\operatorname{tg}^5 x - \frac{1}{3} = 0$	[0, 0.8]	дихотомии	0.3333
22	$\arccos x - \sqrt{1 - 0{,}3x^3} = 0$	[0, 1]	итераций	0.5629

ОБЩИЙ МЕТОД РЕШЕНИЯ

Каждое уравнение решаем 3 методами: итераций, дихотомии и Ньютона.

Метод дихотомии

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

До тех пор, пока не будет выполнено условие $|a^k-b^k|<\varepsilon$, процесс будет выполняться.

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

Метод итераций

Идея метода заключается в замене исходного уравнения F(x) = 0 уравнением вида $x=f(x)$. Достаточное условие сходимости данного 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. Достаточно неплохим выбором является функция $x-sign(dx(x))*F(x)$ где $dx(x)$ – производная функции $F(x)$.

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

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

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

Метод Ньютона

Метод Ньютона является частным случаем метода итераций.

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

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

Метод обладает квадратичной сходимостью. Модификацией метода является метод хорд и касательных. Также метод Ньютона может быть использован для решения задач оптимизации, в которых требуется определить ноль первой производной либо градиента в случае многомерного пространства.

В начале программы составляем функции для решения уравнения определённым методом и функции, послужащии им аргументом. Для каждой формулы необходимо запрограммировать исходную функцию. Производную в точку можно принять за дифференциал от функции в данной точке, который считается по формуле $\lim\limits_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}$. В теле основной программы делаем только вызов подпрограмм и вывод.

Для удобства и лаконичности в прогрмаме с помощью оператора typedef введен тип pFunc, который расшифровывается как pointer function и ld, который расшифровывается long double.

ОБЩИЕ СВЕДЕНИЯ О ПРОГРАММЕ

Язык и система программирования: GNU C

Местонахождение файлов: sakost@sakost-pc: ~/university/course_work/

Способ вызова и загрузки: gcc c4.c -lm -Wall -pedantic -std=c99 -o c4.out && c4.out

ОПИСАНИЕ ПЕРЕМЕННЫХ И ФУНКЦИЙ

Таблица 1.А. Описание переменных

Имя	Тип	Описание
k	ld	Точность вычислений
epsilon	ld	Машинное эпсилон
a1	ld	Левая граница 1го отрезка
b1	ld	Правая граница 1го отрезка
a2	ld	Левая граница 2го отрезка
b2	ld	Права граница 2го отрезка

Таблица 1.Б. Описание функций

Название функции	Выходной тип	Входные параметры	Описание
machineeps	ld	отсутствуют	Функциця, считающая машинное эпсилон
dx	ld	pFunc f, ld x	Функция, считающая дифференциал в данной точке
sign	ld	ld x	Возвращает 1, если знак x положительный, -1, если отрицательный и 0, если x близок к нулю.
dht_method	ld	pFunc f, ld a, ld b	Находит корень функции методом дихотомии
iter_method	ld	pFunc f, ld a, ld b	Находит корень функции методом итераций
newton_method	ld	pFunc f, ld a, ld b	Находит корень функции методом Ньютона
func1	ld	ld x	Считает значение 1 функции в данной точке
func2	ld	ld x	Считает значение 2 функции в данной точке

ПРОТОКОЛ

```
[sakost@sakost-pc course work]$ cat main4.c
#include <stdio.h>
#include <math.h>

// 22 var
typedef long double ld;
typedef ld(*pFunc)(ld);

const ld k = 1;

ld machineeps() {
    ld epsilon = 1, prev;
    ld expression;

    do {

        prev = epsilon;
        epsilon = epsilon / 2;
        expression = 1 + epsilon;

    } while (expression > 1);

    return prev;
}
ld epsilon;

ld dx(pFunc f, ld x){
    ld eps = epsilon * 2;
    return (f(x+eps/2) - f(x - eps/2))/eps;
}

ld sign(ld x){
    return x > epsilon ? 1.l : x < -epsilon ? -1.l : 0.l;
}

ld dht_method(pFunc f, ld a, ld b){
```

```
    ld x = (a + b) / 2;
    while (fabsl(a-b) > epsilon * k){
        x = (a + b) / 2;
        if(f(a) * f(x) > 0.l) a = x;
        else b = x;
    }
    return x;
}

ld iter_method(pFunc f, ld a, ld b){
    ld x = (a + b) / 2;
    while(fabsl(f(x)) > epsilon * k){
        x = x - f(x)*sign(dx(f, x));
    }
    return x;
}

ld newton_method(pFunc f, ld a, ld b){
    ld x = (a + b) / 2;
    while(fabsl(f(x)/ dx(f, x)) > epsilon * k){
        x = x - f(x)/dx(f, x);
    }
    return x;
}

ld a1 = 0.l, b1 = 0.8l;
ld func1(ld x){
    return tanl(x) - 1.l/3*powl(tanl(x), 3) + 1.l/5*powl(tanl(x), 5) - 1.l/3;
}

ld a2 = 0.l, b2 = 1.l;
ld func2(ld x){
    return acosl(x) - sqrtl(1.l - 0.3l*powl(x, 3));
}



int main(){
    epsilon = machineeps();
```

```
    printf("for 21 variant:\n");
    printf("dichotomy method result for 21 func: %Lf\n", dht_method(func1, a1, b1));
    printf("iteration method result for 21 func: %Lf\n", iter_method(func1, a1, b1));
    printf("newton method result for 21 func: %Lf\n", newton_method(func1, a1, b1));
    printf("\n");
    printf("for 22 variant:\n");
    printf("dichotomy method result for 22 func: %Lf\n", dht_method(func2, a2, b2));
    printf("iteration method result for 22 func: %Lf\n", iter_method(func2, a2, b2));
    printf("newton method result for 22 func: %Lf\n", newton_method(func2, a2, b2));
}
```

```
[sakost@sakost-pc course work]$ gcc main4.c -lm -Wall -pedantic -std=c99 -o c4.out && ./c4.out
for 21 variant:
dichotomy method result for 21 func: 0.333255
iteration method result for 21 func: 0.333255
newton method result for 21 func: 0.333255

for 22 variant:
dichotomy method result for 22 func: 0.562926
iteration method result for 22 func: 0.562926
newton method result for 22 func: 0.562926
[sakost@sakost-pc course work]$
```

ЗАКЛЮЧЕНИЕ

Данное задание курсового проекта показывает суть некоторых численных методов и их практическое применение для вычисления приближенного значения корней, однако в абсолюте ни один из вышеприведённх методов не идеален, так как требуется заранее определённые границы поиска искомого корня и при увеличении параметра точности затраты по времени растут слишком быстро. Также были использованы универсальные функции, которые принимают в качестве аргументов указатели на другие функции. Это решение позволяет избежать дублирования кода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1) Математический энциклопедический словарь. — М.: «Сов. энциклопедия », 1988. — С. 847.

2) Волков Е. А. Численные методы. — М. : Физматлит, 2003.

3) Максимов Ю. А. Алгоритмы линейного и дискретного программирования – М.: МИФИ, 1980.