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

Институт: №8 «Информационные технологии 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	Степанов А.Н.
Преподаватель	Сысоев М.А.
Оценка	
Дата	

Москва, 2022

Постановка задачи

Составить программу на Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и дихотомии). Нелинейные уравнения оформить как параметры-функции, разрешив относительно неизвестной величины при необходимости. Применить каждую процедуру к решению двух уравнений, заданных двумя строками таблицы, начиная с варианта с заданным номером.

Вариант 12 и 13:

12	$\ln x - x + 1{,}8 = 0$	[2, 3]	итераций	2.8459
13	$x \cdot \operatorname{tg} x - \frac{1}{3} = 0$	[0.2, 1]	Ньютона	0.5472

Теоретическая часть

Метод дихотомии (половинного деления).

Очевидно, что если на отрезке [a, b] существует корень уравнения, то значения на концах отрезка имеют разные знаки: F(a)*F(b) <0. Метод основан на док-ве т. Больцано-Коши и заключается в делении отрезка пополам и его сужении в 2 раза на каждом шаге итерационного процесса в зависимости от знака итерационного процесса.

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 $F(b^{(k)}) \cdot F((a^{(k)} + b^{(k)})/2) > 0$.

Процес требуется повторить до тех пор, пока не будет выполнено условие окончания,а именно : $|a^{(k)}-b^{(k)}|< \varepsilon$

Корень равен середине конечного отрезка

Метод итераций.

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

Достаточное условие сходимости метода: $|f^{(1)}(x)|<1$, $\forall x \in [a,b]$ *(следует проверить перед началом решения, иначе сходиться ничего не будет).*

Начальное приближение корня: $x(o)=(a+b)/2$ *(середина отрезка).*

Итерационный процесс: $x_{k+1}=f(x_k)$

Условие окончания: $|x_k-x_{k-1}|< \varepsilon$

Корень равен конечному x.

Метод Ньютона.

Является частным случаем выше представленного метода.

Метод Ньютона является частным случаем метода итераций.

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

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

Машинное эпсилон — числовое значение, меньше которого невозможно задавать относительную точность для любого алгоритма, возвращающего вещественные числа. Абсолютное значение для машинного эпсилон зависит от разрядности сетки применяемой ЭВМ и от разрядности используемых при расчёте чисел. Формально это машинное эпсилон определяют как число, удовлетворяющее равенству $1 + \varepsilon = 1$. Фактически, два отличных от нуля числа являются равными с точки зрения машинной арифметики, если их модуль разности меньше или не превосходит машинное эпсилон.

В языке Си машинные эпсилон определено для следующих типов: float – $1.19 * 10^{-7}$, double – $2.20 * 10^{-16}$, long double – $1.08 * 10^{-19}$.

Описание алгоритма

Рассмотрим алгоритм решения. Сперва нужно найти машинное эпсилон, на котором будет основываться точность вычисления. Это можно сделать, просто деля 1 на 2, пока это число не будет удовлетворять условию машинного эпсилон: $1 + \varepsilon = 1$.

Проанализировав данные нам методы нахождения корня уравнения, мы можем заметить, что для применения метода итераций уравнение должно быть приведено к виду f(x)=x, причем подобное приведение может быть неоднозначно. Аналитическим путем было выявлено, что подобной функцией в уравнении под номером 12 будет x=log(x)+1.8, а для 13 – x=arctg(1/(3*x)). Также для вычисления методом ньютона нам потребовалась первая производная, которую мы нашли аналитическим методом. В конце нам осталось реализовать требуемые методы решения уравнений в виде функций, применимых для произвольной F(x), а также вывести корни на экран, чтобы сравнить их с ответом и между собой.

Использованные в программе переменные

Название переменной	Тип переменной	Смысл переменной
kerr	Int64_t	Степень, регулирующая погрешность вычисления(не больше 16, иначе уйдете в машинный epsilon и не вернетесь)
x(main)	double	Использовалась для тестирования работоспособности функций. Забыли убрать, потом привыкли. Символ проекта.
rate_error	double	ЕЕ благородие, госпожа Погрешность.

Использованные в программе не библиотечные функции и их переменные.

1. double Eps() – нахождение машинного эпсилон, что потребуется в дальнейшем для вычисления погрешности.

Название переменной	Тип переменной	Смысл переменной
epsilon	double	Переменная, что будет уменьшаться относительно единицы и в конечном итоге достигнет машинного эпсилон.

2. void TEST_EPS() – тестирование double Eps() на правильность нахождения эпсилон относительно библиотечного значения для double.
3. double f_12(double x) – вычисляет значение функции, полученной из уравнения 12 варианта, относительно аргумента x, необходима в методе Ньютона и дихотомии.
4. double f_12_derivative_first(double x) - вычисляет значение производной функции, полученной из уравнения 12 варианта, относительно аргумента x, необходима в методе Ньютона.
5. double f_13(double x) – вычисляет значение функции, полученной из уравнения 13 варианта, относительно аргумента x, необходима в методе Ньютона и дихотомии.
6. double f_13_derivative_first (double x) - вычисляет значение производной функции, полученной из уравнения 13 варианта, относительно аргумента x, необходима в методе Ньютона.
7. double f_12_for_itter(double x) -вычисляет значение функции, полученной преобразованиями исходного уравнения в вид f(x)=x(левую часть) 12 варианта, необходима для метода итераций.

8. double f_13_for_itter(double x) -вычисляет значение функции, полученной преобразованиями исходного уравнения в вид f(x)=x(левую часть) 13 варианта, необходима для метода итераций.
9. double separation(double f(double),double left, double right ,double dbleps) – функция, ~~позволяющая поделить Польшу~~ находящая корень для уравнения вида F(x)=0 на отрезке, ограниченном значениями left и right, с погрешностью, равной dbleps, методом дихотомии.

Название переменной	Тип переменной	Смысл переменной
x	double	Переменная, что будет содержать в себе середину отрезка и как и в док-ве первой теоремы Больцано-Коши, легшей в основу этого метода, будет приравниваться к 1 из двух переменных, обозначающих границы отрезка.

10. double iterations(double f(double),double left, double right ,double dbleps)- функция, ~~позволяющая поделить Польшу~~ находящая корень для уравнения вида F(x)=0 на отрезке, ограниченном значениями left и right, с погрешностью, равной dbleps, методом итераций.

Название переменной	Тип переменной	Смысл переменной
x	double	Переменная, что принимает значение аргумента для функции(x=f(x)) и приравнивается к функции при итерации.Содержит корень в конце работы функции, с чем возвращается

11. double Newtons(double f(double),double f_frst(double),double left, double right ,double dbleps)- функция, ~~позволяющая поделить Польшу~~ находящая корень для уравнения вида F(x)=0 на отрезке, ограниченном значениями left и right, с погрешностью, равной dbleps, методом Ньютона с использованием первой производной F(x).

Название переменной	Тип переменной	Смысл переменной
x	double	Переменная, что принимает значение аргумента для функций и изменяется при итерациях цикла while. Содержит корень в конце работы функции, с чем возвращается

Исходный код программы:

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

double Eps(){
    double epsilon=1.0;
    while(1.0+(epsilon/2.0)!=1.0){
        epsilon/=2.0;
    }
return epsilon;
}

void TEST_EPS(){
assert(Eps()<=_DBL_EPSILON_);
}

double f_12(double x){
    return log(x)- x +1.8;
}
double f_12_derivative_first(double x){
    return 1.0/x-1;



}
double f_13(double x){
    return x*tan(x)- ((double)1/(double)3) ;
}
double f_13_derivative_first(double x){
    return x/(cos(x)*cos(x)) +tan(x);
```

```
}
double f_12_for_itter(double x){
    return log(x)+1.8;

}
double f_13_for_itter(double x){
    return atan(1.0/(3.0*x));

}
double separation(double f(double),double left, double right ,double dbleps){
    double x;
    while(fabs(left - right)>dbleps){
        x=(left + right)/2.0;
        if(f(x)*f(left)<0){
            right=x;
        }
        else {
            left=x;
        }

    }
return (left+right)/2.0;
}
double iterations(double f(double),double left, double right ,double dbleps){
    double x = (left+right)/2.0;

    while(fabs(f(x)-x)>dbleps){




            x=f(x);


    }
    return x;
```

```
}

double Newtons(double f(double),double f_frst(double),double left, double
right ,double dbleps){
double x=(left+right)/2.0;
while (fabs(f(x)/f_frst(x)) > dbleps) {
        x-=f(x)/f_frst(x);

}
     return x;

}
int main()
{  double x=2.0;
   int64_t kerr=0;
   scanf("%d",&kerr);
   assert(kerr<16);
   double rate_error=Eps()*pow(10,16-kerr);
   printf("rate_error=%.16f¥n",rate_error);
   void TEST_EPS();
   printf("The root of 12th  equation ln(x)- x +1.8=0:¥n");
   printf("%.16f|%.16f|%.16f¥n",iterations(f_12_for_itter,2.0,3.0,rate_error)
,Newtons(f_12,f_12_derivative_first,2.0,3.0,rate_error),
separation(f_12,2.0,3.0,rate_error));
   printf("The root of 13th  equation x*tg(x) - 1/3=0:¥n");
   printf("%.16f|%.16f|%.16f¥n",iterations(f_13_for_itter,0.2,1.0,rate_error)
,Newtons(f_13,f_13_derivative_first,0.2,1.0,rate_error),
separation(f_13,0.2,1.0,rate_error));
   return 0;
}
```

Входные данные

Отсутствуют (кроме точности, в примере 15, на общую работы программы не влияет)

Выходные данные (Протокол исполнения)

rate_error=0.0000000000000022
The root of 12th equation ln(x)- x +1.8=0:
2.8458681814741791|2.8458681814741822|2.8458681814741817
The root of 13th equation x*tg(x) - 1/3=0:
0.5471607572603289|0.5471607572603300|0.5471607572603296

Тесты

Не предусмотрены заданием.

Вывод

В работе описано определение машинного эпсилон, приведены его значения для разных переменных языка Си, описана методы решения уравнений некоторыми численными методами и составлен алгоритм реализации вычисления корня уравнения с заданной точностью для заданного отрезка. На основе алгоритма составлена программа на языке Си, проведено её тестирование, где это было возможно, составлен протокол исполнения программы. В целом, работа понравилась. Приятно применять знания из других областей(матана) для решения какой-либо задачи по программированию.

Список литературы

1. Машинный ноль – URL: https://ru.wikipedia.org/wiki/Машинный_ноль
2. Ряд Тейлора – URL: https://ru.wikipedia.org/wiki/Ряд_Тейлора