Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра 806 “Вычислительная математика и программирование”

Курсовая работа

по курсу “Архитектура компьютера”

1 семестр

Задание 4. Процедуры и функции в качестве параметров

Студент: Старостина А.А.
Группа: М8О-108Б-22,
№ по списку: 19
Руководитель: Сахарин Н.А.
Дата: 08.01.2023
Оценка: _____

г. Москва, 2023

Содержание

Задание

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными способами(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.

Варианты

№	Уравнение	Отрезок, содержащий корень	Базовый метод	Приближенное значение корня
19	$x - \frac{1}{3 + sin3.6x} = 0$	[0,0.85]	итерации	0.2624
20	$0.1x^2 - xlnx = 0$	[1,2]	Ньютона	1.1183

Общий метод решения

Вычисление приближенного значений функций при помощи метода дихотомии, метода итераций и метода Ньютона.

Метод дихотомии - деление отрезка пополам, учитывает что знак функции должен быть разным. До тех пор, пока длина отрезка не будет меньше значения машинного эпсилон, процесс деления будет выполняться. Приближенное значение корня к моменту окончания итерационного процесса будет находиться примерно в середине заданного отрезка.

Метод итераций заключается в замене исходного уравнения F(x) = 0 уравнением f(x) = x. Начальным приближенным значением корня является середина заданного отрезка. Итерационный процесс имеет вид: $x^{k+1} = f\left(x^{(k)}\right)$. Процесс выполняется пока разность $x^{(k)}$ и $x^{(k+1)}$ не станет меньше значения машинного эпсилон.

Метод Ньютона - частный случай метода дихотомии. Итерационный процесс представляет собой: $x^{(k+1)} = x^{(k)} - F\left(x^{(k)}\right) / F'\left(x^{(k)}\right)$.

Общие сведения о программе

Аппаратное обеспечение: домашний ноутбук
Операционная система: Linux Ubuntu, версия 22.04.1 LTS
Язык и система программирования: C, GNU
Местонахождение файлов: /home/ann
Компиляция программы: gcc -lm kp4.c
Вызов программы: ./a.out

Функциональное назначение

Программа предназначена для вычисления приближенного значения трансцендентных алгебраических уравнений с использованием различных

численных методов и при помощи встроенных программных функций библиотеки языка Си.

Описание логической структуры

Программа получает на вход заданный отрезок, находит значение уравнения F(x) = 0 различными численными методами и выводит полученный корень уравнения.

Описание переменных, констант и подпрограмм

Функция	**Входные аргументы**	**Описание**
machine_epsilon		Функция для подсчета машинного ε
F19,F20	long double x	Вычисляет значение входной функции
F19_first_derivative, F20_first_derivative	long double x	Функция, вычисляющая первую производную от входной функции
F19_second_derivative, F20_second_derivative	long double x	Функция, вычисляющая вторую производную от входной функции

f19_first_derivative, f20_first_derivative	long double x	Функция, вычисляющая первую производную от уравнения, в котором выражен x
dichotomya	long double (*F)(long double), long double a, long double b, long double abs_epsilon, long double otn_epsilon	Функция, вычисляющая значение уравнения F(x) = 0 методом дихотомии
iteration	long double (*f)(long double), long double (*f_first_derivative)(long double), long double a, long double b, long double abs_epsilon, long double otn_epsilon	Функция, вычисляющая значение уравнения F(x) = 0 методом итерации
newton	long double(* F)(long double), long double (*F_first_derivative)(lo	Функция, вычисляющая значение уравнения F(x) = 0 методом Ньютона

	ng double), long double (*F_second_derivative) (long double), long double a, long double b, long double abs_epsilon, long double otn_epsilon	
f19, f20	long double x	Функция, вычисляющая выраженный x

Таблица 1. Описание функций программы

Переменная	**Значение**
long double abs_epsilon	Машинный эпсилон (абсолютный)
long double otn_epsilon	Машинный эпсилон (относительный)
long double a,b	Границы отрезка
long double x	Значение аргумента функции

Таблица 2. Описание переменных

Протокол

Код программы:

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

```
typedef long double ldbl;

ldbl machine_epsilon() {
    ldbl eps = 1.0;
    while (1 + eps / 2.0 != 1)
        eps /= 2.0;
    return eps;
}

ldbl F19(ldbl x) {
    return x - 1/(3 + sin(3.6*x));
}

ldbl f19(ldbl x) {
    return 1/(3 + sin(3.6*x));
}

ldbl f19_first_derivative(ldbl x) {
    return -(3.6*cos(3.6*x)/(3+sin(3.6*x)*(3+sin(3.6*x))));
}

ldbl F19_first_derivative(ldbl x) {
    return 1 + (3.6*cos(3.6*x)/(3+sin(3.6*x)*(3+sin(3.6*x))));
}

ldbl F19_second_derivative(ldbl x) {
                return      (-3.6*3.6*sin(3.6*x)*(3+sin(3.6*x))*(3+sin(3.6*x))      -
(3.6*cos(3.6*x))*2*3.6*cos(3.6*x)*(3+sin(3.6*x)))/(pow(3+sin(3.6*x),4));
```

```
}

ldbl F20(ldbl x) {
   return 0.1*x*x-x*logl(x);
}

ldbl f20(ldbl x) {
   return sqrt(10*x*logl(x));
}

ldbl f20_first_derivative(ldbl x) {
   return (10*logl(x)+1)/(2*sqrt(10*x*logl(x)));
}

ldbl F20_first_derivative(ldbl x) {
   return 0.2*x-logl(x)-1;
}

ldbl F20_second_derivative(ldbl x) {
   return 0.2-1/x;
}

ldbl dichotomya(ldbl (*F)(ldbl), ldbl a, ldbl b, ldbl abs_epsilon, ldbl
otn_epsilon) {
   ldbl x = a + (b - a) / 2;
   if (F(a) * F(b) < 0){
      while (fabs(a - b) > fmax(otn_epsilon * fabs(x), abs_epsilon)) {
         x = (a + b) / 2;
         if (F(x) * F(a) < 0)
```

```
            b = x;
        else
            a = x;
    }
    return x;
  }
  else
    return 0;
}

ldbl iteration(ldbl (*f)(ldbl), ldbl (*f_first_derivative)(ldbl), ldbl a, ldbl b, ldbl abs_epsilon, ldbl otn_epsilon)
{
  ldbl x = a + (b - a) / 2;
  if (fabs(f_first_derivative(x)) < 1) {
    while (fabs(f(x) - x) >= fmax(otn_epsilon * fabs(x), abs_epsilon))
      x = f(x);
    return x;
  }
  else
    return 0;
}

ldbl newton(ldbl (*F)(ldbl),ldbl (*F_first_derivative)(ldbl),ldbl (*F_second_derivative)(ldbl),ldbl a,ldbl b, ldbl abs_epsilon, ldbl otn_epsilon) {
  ldbl x = a + (b - a) / 2;
      if (fabs(F(x) * F_second_derivative(x)) < (F_first_derivative(x) * F_first_derivative(x))) {
        while (fabs(F(x) / F_first_derivative(x)) > fmax(otn_epsilon * fabs(x),
```

```
abs_epsilon))
            x -= F(x) / F_first_derivative(x);
        return x;
    }
    else
        return 0;
}

void result(ldbl d, ldbl i, ldbl n) {
   if (d != 0) printf("The dichotomya method: %.10Lf\n", d);
   else printf("The dechotomya method isn't suitable\n");
   if (i != 0) printf("The iteration method: %.10Lf\n", i);
   else printf("The iteration method isn't suitable\n");
   if (n != 0) printf("The Newton's method: %.10Lf\n", n);
   else printf("The Newton's method isn't suitable\n");
}

int main() {
    ldbl a19 = 0, b19 = 0.85;
    ldbl a20 = 1, b20 = 2;
    ldbl abs_epsilon = machine_epsilon();
    ldbl otn_epsilon = sqrt(abs_epsilon);
    ldbl d19 = dichotomya(F19, a19, b19, otn_epsilon, abs_epsilon);
      ldbl i19 = iteration(f19, f19_first_derivative, a19, b19, otn_epsilon,
abs_epsilon);
      ldbl n19 = newton(F19, F19_first_derivative, F19_second_derivative, a19,
b19, otn_epsilon, abs_epsilon);
    printf("Machine epsilon is %.40Lf\n",abs_epsilon);
    printf("Function 19 var: x - 1/(3+sin(3.6x))\n");
```

```
    result(d19, i19, n19);
    printf("\n");
    ldbl d20 = dichotomya(F20, a20, b20, otn_epsilon, abs_epsilon);
        ldbl i20 = iteration(f20, f20_first_derivative, a20, b20, otn_epsilon, abs_epsilon);
     ldbl n20 = newton(F20, F20_first_derivative, F20_second_derivative, a20, b20, otn_epsilon, abs_epsilon);
    printf("Function 20 var: 0.1*x^2-x*ln(x)\n");
    result(d20, i20, n20);
    return 0;
}
```

Входные данные

Отсутствуют

Выходные данные

```
ann@ann:~$ gcc kp4.c -lm
ann@ann:~$ ./a.out
Machine epsilon is 0.000000000000000000000000000000001925930
Function 19 var: x - 1/(3+sin(3.6x))
The dichotomya method: 0.2624414651
The iteration method: 0.2624414651
The Newton's method: 0.2624414651


Function 20 var: 0.1*x^2-x*ln(x)
```

The dichotomya method: 1.1183255916

The iteration method isn't suitable

The Newton's method: 1.1183255916

Вывод

В результате выполнения данной курсовой работы были получены навыки работы по получению корня уравнения. Было изучено вычисление машинного эпсилона и различных численных методов, таких как: метод дихотомии, метод итераций и метод Ньютона. Оценивая полученные данные, можно сказать, что каждый из методов неидеален, так как для поиска корня необходимо знать точные границы отрезка.

Ответы совпали везде, где могли. Только в 20м варианте не удалось вычислить метод итерации, потому что не выполняется главное условие – модуль производной выбранной функции получается больше единицы (при любом x).