Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра вычислительной математики и программирования

Курсовая работа

По курсу «Фундаментальная информатика»

I семестр

Задание №4 «Процедуры и функции в качестве параметров»

Выполнил студент

1-го курса, 105-ой группы

Махмудов О. С.

____

(подпись)

Научный руководитель

Доцент кафедры 806

Сластушенский Ю. В.

____

(подпись)

Работа защищена

«__»___________ 2019

Оценка____________

ПОСТАНОВКА ЗАДАЧИ

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 и графическую иллюстрацию, например, с использованием gnuplot.

13 и 14 вариант

13	$x \cdot \mathrm{tg}\, x - \frac{1}{3} = 0$	[0.2, 1]	Ньютона	0.5472
14	$\mathrm{tg}\frac{x}{2} - \mathrm{ctg}\frac{x}{2} + x = 0$	[1, 2]	дихотомии	1.0769

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Краткие сведения из численных методов

Рассматривается уравнение вида Предполагается, что функция F(x) достаточно гладкая, монотонная на этом отрезке и существует единственный корень уравнения x* ∈ [a,b]. На отрезке [a,b] ищется приближенное решение x с точностью ε, т.е. такое, что $|x - x*| < \varepsilon$.

При решении реальных задач, где поведение функции F(x) неизвестно, сначала производят исследование функции (аналитическое, численное, или графическое), например, с помощью программ qnuplot, MathLab, MathCAD, Maple. Также выполняют т. н. отделение корней, т. е. разбивают область определения функции на отрезки монотонности, на каждом из которых имеется ровно один корень и выполняются другие условия применимости численных методов (гладкость). Различные численные методы предъявляют разные требования к функции F(x), обладают различной скоростью сходимости и поведением. В данном задании предлагается изучить и запрограммировать три простейших численных метода решения алгебраических уравнений и провести вычислительные эксперименты по определению корней уравнений на указанных в задании отрезках монотонности.

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.

За начальное приближение принимаются границы исходного отрезка $a(0)=a$, $b(0)=b$

Итерационный процесс:

$a(k+1)=(a(k)+b(k))/2$, $b(k+1)=b(k)$, если $F(a(k))\cdot F((a(k)+b(k))/2)>0$

$a(k+1= a(k)$, $b(k+1)=(a(k)+b(k))/2$, если $F(b(k))\cdot F((a(k)+b(k))/2)>0$

Условия окончания: $| a(k)+b(k)|< \varepsilon$.

Приближенное значение корня: $x^*\approx(a(\text{конечное})+b(\text{конечное}))/2$.

Метод итераций

Идея метода заключается в замене исходного уравнения $F(X) = 0$ уравнением вида $x = f(x)$. Достаточное условие сходимости 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.

Начальное приближение корня: $x(0)= (a + b) /2$

Итерационный процесс: $x(k+1) =f(x(k))$ Условие окончания: $| x(k))- x(k-1) |< \varepsilon$

Приближенное значение корня: $x^*\approx x(\text{конечное})$

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)-F(x(k))/ F'(x(k)).

ОБОРУДОВАНИЕ И СП

Используется ноутбук с операционной системой семейства Windows 10-ой версии (64 bit), с процессором Intel Core i3 8130U и оперативной памятью в 4 ГБ. Для компиляции и отладки программы используется подсистема Linux для Windows, после чего программа компилируется с помощью GNU CC.

ОПИСАНИЕ ФУНКЦИЙ И ПЕРЕМЕННЫХ

Функции

double eps() – вычисляет машинное эпсилон. К сожалению, невозможно создать переменную, равную значению этой функции в глобальной области, поэтому приходится пользоваться этой функцией в каждом месте, где необходим машинный эпсилон. Также в языке Си существует константа, равная машинному эпсилон, но мы не будем ее использовать, так как значение эпсилон зависит от аппаратной части.

double F1 – первая функция из дано, корень которой нужно найти

double F2 – вторая функция из дано, корень которой нужно найти

double dih – функция, которая ищет корень методом дихотомии

double iter - функция, которая ищет корень методом итерации

double newton - функция, которая ищет корень методом Ньютона

}

Переменные функции main

double a1 - первая координата отрезка, на котором ищется корень первой функции

double b1 – вторая координата отрезка, на котором ищется корень первой функции

double a2 - первая координата отрезка, на котором ищется корень второй функции

double b2 – вторая координата отрезка, на котором ищется корень второй функции

ВХОДНЫЕ ДАННЫЕ И ВЫХОДНЫЕ ДАННЫЕ

Входных данных нет, на выход программа выводит таблицу в которой находятся функции из дано и значения корней этих функций (первые 4 цифры после запятой) на данном интервале рассчитанные 3 разными методами. Вторая таблица выводит тоже самое, только значения корней более точные (16 цифр после запятой).

ПРОГРАММА

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

double eps() {
	double e = 1;
	while (1 + e != 1) {
		e /= 2;
	}
	return e;
}

double F1 (double x) {
	return (3 * x * tan(x) - 1) / 3;
}
```

```
double F2 (double x) {
	return tan(x / 2) - 1 / tan(x / 2) + x;
}

double dih(double (*F)(double), double a, double b,double e) {
	double c;
	int i = 0;
	while (fabs(a - b) > e && i < 100) {
		c = (a + b) / 2;
		if ((F(a) * F(c)) > 0) {
			a = c;
		}
		if ((F(b) * F(c)) > 0) {
			b = c;
		}
		i++;
	}
	c = (a + b) / 2;
	return c;
}

double iter(double (*F)(double), double k, double a, double b, double e){
	double x = (a + b) / 2;
	int i = 0;
	while (fabs(F(x)) > e && i < 100) {
		x = x - k * F(x);
		i++;
	}
	return x;
}

double newton(double(*F)(double), double k, double a, double b, double e){
	double x = (a + b) / 2;
	int i = 0;
	while (fabs(F(x) / k ) > e && i < 100) {
		x = x - F(x) / k ;
		i++;
	}
	return x;
}


int main() {
	double a1, b1, a2, b2;
```

```
printf("eps = %e\n", eps());
a1 = 0.2;
b1 = 1;
a2 = 1;
b2 = 2;
printf("---------------------------------------------------------------------------------
\n");
printf("      Function       | Dihotomy_method  | Iteration_method  |
Newton_method  \n");
printf("    x * tg(x) - 1/3    | %.4lf           | %.4lf            | %.4lf    \n",
dih(F1, a1, b1, eps()), iter(F1, 1, a1, b1, eps()), newton(F1, 1, a1, b1, eps()));
printf(" tg(x/2) -ctg(x/2) + x |  %.4lf          | %.4lf            | %.4lf    \n",
dih(F2, a2, b2, eps()), iter(F2, 0.1, a2, b2, eps()), newton(F2, 10, a2, b2, eps()));
printf("---------------------------------------------------------------------------------
\n");
printf("\n");
printf("----------------------------------------------------------------------------------
-\n");
printf("      Function       | Dihotomy_method  | Iteration_method |
Newton_method  \n");
printf("    x * tg(x) - 1/3    | %.16lf | %.16lf | %.16lf  \n", dih(F1, a1, b1,
eps()), iter(F1, 1, a1, b1, eps()), newton(F1, 1, a1, b1, eps()));
printf(" tg(x/2) -ctg(x/2) + x | %.16lf | %.16lf | %.16lf  \n", dih(F2, a2, b2,
eps()), iter(F2, 0.1, a2, b2, eps()), newton(F2, 10, a2, b2, eps()));
printf("----------------------------------------------------------------------------------
-\n");
return 0;
}
```

ПРОТОКОЛ РАБОТЫ

```
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ gcc kursach4.c -lm
Admin@LAPTOP-Q5U6S2UH:/mnt/c/Users/Admin/Desktop/Все для вуза$ ./a.out
eps = 1.110223e-16
-----------------------------------------------------------------------------------------
      Function          | Dihotomy_method  | Iteration_method  | Newton_method
    x * tg(x) - 1/3     | 0.5472           | 0.5472            | 0.5472
 tg(x/2) -ctg(x/2) + x  | 1.0769           | 1.0769            | 1.0769
------------------------------------------------------------------------------------------

-------------------------------------------------------------------------------------------------
      Function          |  Dihotomy_method   |   Iteration_method  | Newton_method
    x * tg(x) - 1/3     | 0.5471607572603301 | 0.5471607572603300 | 0.5471607572603300
 tg(x/2) -ctg(x/2) + x  | 1.0768739863118038 | 1.0768739863118038 | 1.0768739863118040
--------------------------------------------------------------------------------------------------
```

ВЫВОД

В курсовой работе было продемонстрировано решение двух нелинейных уравнений с применением различных методов:

1. Дихотомии
2. Итерации
3. Ньютона

Наибольшую точность показал метод Ньютона (он же-самый экономный по памяти), наименьшую-метод итераций (он же самый затратный), что говорит о пользе первого, но при решении по нему требуется намного больше высчитывать первоначальных данных: производных, перемножений функций и т.д.