Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Курсовая работа
по курсу «Вычислительные системы»
I Семестр

Задание 4
Процедуры и функции в качестве параметров

Студент:	Сыроежкин К.Г
Группа:	М8О-104Б-18
Преподаватель:	Никулин С.П.
Оценка:	
Дата:	

Москва, 2018 г.

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.

Общий метод решения

8	$0{,}6 \cdot 3^x - 2{,}3x - 3 = 0$	[2, 3]	дихотомии	2.4200
9	$x^2 - \ln(1+x) - 3 = 0$	[2, 3]	итераций	2.0267

1.Метод дихотомии (половинного деления).

Очевидно, что если на отрезке $[a,b]$ существует корень уравнения, то значения функции на концах отрезка имеют разные знаки: $F(a) \bullet F(b) < 0$. 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* $F(a^{(k)}) \bullet F((a^{(k)} + b^{(k)})/2) > 0$*; или по формулам:* $a^{(k+1)} = a^{(k)}$, $b^{(k+1)} = (a^{(k)} + b^{(k)})/2$, *если* $F(b^{(k)}) \bullet F((a^{(k)} + b^{(k)})/2) > 0$.

Процесс повторяется до тех пор, пока не будет выполнено условие окончания $|a^{(k)} - b^{(k)}| < \varepsilon$.

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: $|f'(x)| < 1$, $x \in [a,b]$. 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: $x^{(0)} = (a + b)/2$ (середина исходного отрезка).

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

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

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

3.Метод Ньютона.

Метод Ньютона является частным случаем метода итераций.

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

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

Для начала необходимо описать возможностями языка Си вычисление приближенного значения функции методом итерации, дихотомии (половинного деления), Ньютона, хорд (касательных). Далее подставить в значения переменных функцию и вывести высчитанные значения.

Для запуска программы необходим рабочий компилятор и интерпретатор языка Си. Для выполнения программы требуется любая ОС с рабочими компилятором и интерпретатором. Файл с исходным кодом находится в директории /168038/k4.cpp. Для компиляции программы необходимо в терминале написать следующее: c++ –o k4.cpp.out k4.cpp, для запуска необходимо написать: ./k4.cpp.out После этого в терминале отобразятся результаты работы программы.

Программа предназначена для решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона, дихотомии и хорд).

Программа получает требуемый отрезок, далее на промежутках вычисляет значения функции требуемым методом, находит то, при котором функция равна нулю и выводит значение на экран пользователю.

Описание переменных

X	**Аргумент функции**
A	**Левая граница значений**
B	**Правая граница значений**
eps	**Машинное эпсилон**

Описание функций

ep	**Вычисление машинного эпсилон**
func_1/2	**функции**
funcit_1/2	**Выраженные функции**
funcitdiff_1/2	**Производные выраженных функций**
diff1_1/2	**1 производная функции**
diff2_1/2	**2 производная функции**
Dich	**Расчет значения по дихотомии**
iter	**Расчет значения по итерациям**
newt	**Расчет значения по Ньютону**

Программа

```
#include <stdio.h>
#include <math.h>
double ep(void)
{
   double e = 1.0;
   while ((1 + (e / 2.0))>1) e /= 2.0;
   return e;
}
double func_1(double x)
{
```

```
    return 0.6*pow(3,x)-2.3*x-3;
}
double func_2(double x)
{
    return x*x-logl(1+x)-3;
}
double funcit_1(double x)
{
    return log((2.3*x+3)/0.6)/log(3);
}
double funcit_2(double x)
{
    return sqrt(logl(1+x)+3);
}
double funcitdiff_1(double x)
{
    return 23/(10*(2.3*x+3)*log(3));
}
double funcitdiff_2(double x)
{
    return 1/(2*(x+1)*sqrt(logl(1+x)+3));
}
double diff1_1(double x)
{
    return 0.6*pow(3,x)*logl(3)-2.3;
}
```

```
double diff1_2(double x)
{
    return 2*x-1/(x+1);
}
double diff2_1(double x)
{
    return 0.6*pow(3,x)*pow(logl(3),2);
}
double diff2_2(double x)
{
    return 2+1/pow((x+1),2);
}
double Dich(double(*F)(double))
{
    double a = 2, b = 3, eps = ep() * 100;
    while (fabs(a - b)>eps)
    {
        if ((F(a)*F((a + b) / 2))>0)
            a = (a + b) / 2;
        else if ((F(b)*F((a + b) / 2))>0)
            b = (a + b) / 2;
    }
    return (a + b) / 2;

}
```

```
double iter(double(*F1)(double), double(*F2)(double))
{
    double a = 2, b = 3;
    if (abs(F2(a))>1 || abs(F2(b))>1)
        return 0;
    else
    {
        double eps = ep() * 100;
        double x = (a + b) / 2.0;
        while (fabs(F1(x) - x) > eps)
            x = F1(x);
        return F1(x);
    }
}
```

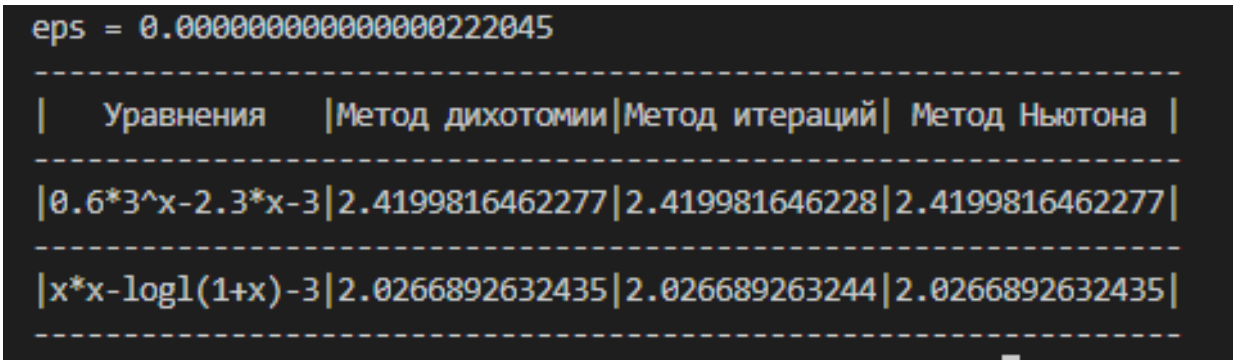
```
double newt(double(*F1)(double), double(*F2)(double),
double(*F3)(double))
{
    double a = 2, b = 3;
    if (abs(F1(a)*F3(a))<(F1(a)) || abs(F1(b)*F3(b))<(F1(b)))
        return 0;
    double eps = ep() * 100;
    double x = (a + b) / 2.0;
    while (fabs((x - F1(x) / F2(x)) - x) > eps)
        x = x - F1(x) / F2(x);
```

```
    return (x - F1(x) / F2(x));
}

int main()
{
    printf("eps = %.21lf\n", ep());
    printf("-----------------------------------------------------------------\n");
    printf("|   Уравнения   |Метод дихотомии|Метод итераций| Метод Ньютона |\n");
    printf("-----------------------------------------------------------------\n");
    printf("|0.6*3^x-2.3*x-3|%.13lf|", Dich(func_1));
    if (iter(funcit_1, funcitdiff_1) == 0)
        printf("  Невозможно  |");
    else
        printf("%.12lf|",  iter(funcit_1, funcitdiff_1));
    if (newt(func_1, diff2_1, diff1_1) == 0)
        printf("  Невозможно  |\n");
    else
        printf("%.13lf|\n", newt(func_1, diff2_1, diff1_1));
    printf("-----------------------------------------------------------------\n");
    printf("|x*x-logl(1+x)-3|%.13lf|", Dich(func_2));
    if ( iter(funcit_2, funcitdiff_2) == 0)
        printf("  Невозможно  |");
    else
        printf("%.12lf|", iter(funcit_2, funcitdiff_2));
    if (newt(func_2, diff2_2, diff1_2) == 0)
```

```
        printf("  Невозможно  |\n");
    else
        printf("%.13lf|\n", newt(func_2, diff2_2, diff1_2));
    printf("-----------------------------------------------------------------\n");
}
```

Результат работы программы



Вывод

Составленная программа на языке Си решает заданное алгебраическое уравнение методом дихотомии и Ньютона. Метод итераций не подходит для подбора корня данного уравнения. Полученные в результате вычислений программы значения совпадают до 3 знаков после запятой с заданным приближённым значением корня в исходных данных задания курсовой работы.