

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 «Вычислительные системы»
I семестр
Задание 4
«Процедуры и функции в качестве параметров»

| | |
|---------------|-------------|
| Группа | М8О-109Б-22 |
| Студент | Дударь Ю.М. |
| Преподаватель | Сысоев М.А. |
| Оценка | |
| Дата | |

Постановка задачи

Составить программу на 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`.

Вариант 1:

Функция:

$$x + \sqrt{x} + \sqrt[3]{x} - 2,5 = 0$$

Отрезок содержащий корень: [0.4, 1.0]

Вариант 1:

Функция:

$$x - \frac{1}{3 + \sin 3.6x} = 0$$

Отрезок содержащий корень: [0.0, 0.85]

Теоретическая часть

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: $x^* \approx x^{(\text{конечное})}$.

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.

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

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

Численное дифференцирование — Так как возможности компьютера не позволяют проводить вычисления с бесконечно малыми, для расчетов будем брать просто очень маленькие значения. Так, для вычисления производной через предел возьмем `prb` равное `1e-6`

Описание алгоритма

Делаем функцию для высчитывания корня методом дихотомии. После чего выводим его значение. Аналогично поступаем и для метода итераций, но для него отдельно выгодно будет сделать проверку.

Использованные в программе переменные

| Название переменной | Тип переменной | Смысл переменной |
|---------------------|----------------|------------------------|
| prib | long double | Маленькое значение |
| step | long double | Шаг для проверки |
| a | long double | Левая граница отрезка |
| b | long double | Правая граница отрезка |
| x1 | long double | Следующее значение x |

Исходный код программы:

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

//Метод дихотомии
long double func_18(long double x){
    //  $x + \sqrt{x} + \sqrt[3]{x} - 2.5$ 
    return x + sqrtl(x) + cbrtl(x) - 2.5;
}

long double dixit(long double (*f)(long double), long double a, long double b){
    long double c;
    long double ans;
    while (fabsl(a - b) > LDBL_EPSILON) {
        c = (a + b) / 2.0;
        if (f(c) * f(a) < 0) {
            b = c;
        } else {
            a = c;
        }
    }
    ans = c;
    return ans;
}

//Метод итерации
//Численное выражение первой производной
long double derive(long double (*f)(long double), long double x){
    long double prib = 1e-6;
    long double ans = (f(x + prib) - f(x)) / prib;
    return ans;
}

long double func_19(long double x) {
    //  $x = 1 / (3 + \sin(3.6 * x))$ 
    return 1 / (3 + sinl(3.6 * x));
}

long double func_19_hands(long double x) {
    return -(18*cosl(3.6*x)/(5*powl((3+sinl(3.6*x)),2)));
}

//Проверка на сходимость итерации
```

```

int is_iterat(long double (*f)(long double), long double (*hands)(long double), long double a,
long double b) {
    long double step = (b-a)/100000;
    for (long double x=a; x<=b; x+=step) {
        if (derive(f, x) >= 1 || hands(x) >= 1) {
            return 0;
        }
    }
    return 1;
}

```

// Компьютерный метод

```

long double iterat(long double (*f)(long double), long double a, long double b) {
    long double x = (a + b) / 2.0;
    long double x1 = f(x);
    while (fabs(x1 - x) >= LDBL_EPSILON) {
        x = x1;
        x1 = f(x);
    }
    return x1;
}

```

```

int main() {
    //18 функция
    long double a = 0.4,
        b = 1;
    printf("Func 18:  $x + \sqrt{x} + \sqrt[3]{x} - 2.5 = 0$  Method: dixit.\n%.19Lf", dixit(func_18, a,
b));

```

```

    printf("\n\n");

```

//19 функция

```

a = 0.0;
b = 0.85;
printf("Func 19:  $x - 1 / (3 + \sin(3.6x)) = 0$  Method: iterations.\n");
if (is_iterat(func_19, func_19_hands, a, b)) {
    printf("Method is covergent.\n");
    printf("Approximated root of the equation: %.19Lf\n", iterat(func_19, a, b));
} else {
    printf("Method is not covergent.\n");
}
return 0;
}

```

