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:** | Аксенов А. Е. |
| **Группа:** | 8О-108Б, №9 |
| **Преподаватель:** | Поповкин А. В. |
|  |  |
| **Оценка:** |  |
| **Дата:** |  |

# Содержание

## Итерационные процессы. Функции как параметры. Задание………………….3

1. Краткие сведения из численных методов……………………………………….3
2. Метод дихотомии (половинного деления)…………………………………...….3
3. Метод итераций……………………………………..…………………………….4
4. Метод Ньютона………………………………….………………………………..4
5. Описание переменных, констант, функций…………………….......………......5
6. Описание подпрограмм……………..……………………………………………6
7. Работа программы (протокол)……….….………………...……………………..7
8. Замечание…………………..…….………………………………………………..9
9. Заключение………………………………………………………………..………9
10. Список литературы……………………………………………………….……..10

## **Итерационные процессы. Функции как параметры.**

### **Задание**

Составить программу на языке Си с процедурами решения трансцедентных алгебраических уравнений различными численными методам (итераций, Ньютона и половинного деление — дихотомии). Уравнения оформить как функции параметры, разрешив относительно неизвестной величины в случае необходимости. Применить каждую процедуру к решению двух уравнений — заданного вариантом и следующего за ним. Если метод неприменим, дать математическое обоснование и графическую иллюстрацию, например, с использованием Гнуплота.

### **Краткие сведения из численных методов**

Рассматривается уравнение вида F(x)=0. Предполагается, что функций 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*| < ε.*

При решении реальных задач, где поведение функции F(x) неизвестно, сначала производят исследование функции (аналитическое, численное или графическое), например, с помощью программ gnuplot, MathLab, MathCAD, Maple. Также выполняют т. н. отделение корней, т. е. разбивают область опреденения 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 опредению корней уравнений на указанных в задании отрезках монотонности и, в качестве дополнительного упражнения, вне их.

### **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) ⋅ 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 a0 = a, b0 = b.
* Итерационный процесс:
  + ak+1 = (ak + bk) / 2, bk+1 = bk, если F(ak) ⋅ F((ak + bk) / 2) > 0;
  + ak+1 = ak, bk+1 = (ak + bk) / 2, если F(bk) ⋅ F((ak + bk) / 2) > 0.
* Условие окончания: |ak - bk| < ε.
* Приближенное значение корня: x\* ≈ (aконечное + bконечное) / 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 ∈ [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: x0 = (a + b) / 2.
* Итерационный процесс: xk+1 = f(xk).
* Условие окончания: |xk - xk-1| < ε.
* Приближенное значение корня: x\* ≈ xконечное.

### **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: xk+1 = xk - F(xk)/F'(xk).

Вариант 1/2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вар. | Формула | Отрезок | Осн. метод | Приб. знач. |
| 1 |  | [3, 4] | Ньютона | 3.5265 |
| 2 |  | [1, 2] | дихотомии | 1.0804 |

**Описание переменных, констант, функций:**

*Таблица 1 – Описание переменных, констант и функий.*

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| eps | double  (переменные) | Эпсилон \* const (чтобы значения немного различались) |
| res1\_1  res1\_3  res2\_1  rec2\_2  res2\_3 | Результаты вычисления корня различными методами |
| x | Корень, выч. методом дихотомии/Ньютона |
| a | Левая грань |
| b | Правая грань |
| previous | Метод Ньютона (код до итер.) |
| next | Метод Ньютона (код после итер.) |
| func1(double x)  func2(double x) | double  (функции) | Исходная F(x)=0 №1 |
| Исходная F(x)=0 №2 |
| func1\_iter(double x)  func2\_iter(double x) | f(x)=x №1 для Iter(итераций) |
| f(x)=x №2 для Iter(итераций) |
| func1\_der(double x)  func2\_der(double x) | Производная F(x)=0 №1 |
| Производная F(x)=0 №2 |
| double dht(funct func, double a, double b, double epsa) | double | Функция нахождения корня методом дихотомии |
| double iter(funct func, double a, double b, double epsa) | Функция нахождения корня методом итераций |
| double newton(funct func, funct der, double a, double b, double epsa) | Функция нахождения корня методом Ньютона |
| typedef double (\*funct) (double); | double | Тип “funct” для использования подфункций в функциях |
| int main() | int | Главная подпрограмма (для вывода значений) |

**Описание подпрограмм**

double dht(funct func, double a, double b, double epsa)

Функция находит корень F(x)=0 методом дихотомии.

double iter(funct func, double a, double b, double epsa)

Функция находит корень F(x)=0 методом итераций.

double newton(funct func, funct der, double a, double b, double epsa)

Функция находит корень F(x)=0 методом Ньютона.

double func1(double x)

Функция вычисляет значение f1(x).

double func2(double x)

Функция вычисляет значение f1(x).

double func1\_iter(double x)

Функция представляет f1(x) в виде x = f1’(x) и выч. f1’(x) (для метода итераций)

double func2\_iter(double x)

Функция представляет f2(x) в виде x = f2’(x) и выч. f2’(x) (для метода итераций)

double func1\_der(double x)

Функция вычисляет производную f1(x).

double func2\_der(double x)

Функция вычисляет производную f2(x).

**Работа программы (протокол)**

fallfire@vb:~/kursach4$ cat kurs.c

#include<stdio.h>

#include<math.h>

typedef double (\*funct) (double);

double Epsilon(void)

{

double epsilon = 1.0;

while ((1.0 + (epsilon / 2.0)) > 1.0) {

epsilon /= 2.0;

}

return epsilon;

}

double dht(funct func, double a, double b, double epsa) {

double x;

while(fabs(a-b) > epsa) {

x = (a+b) / 2.0;

if (func(a)\*func(x) > 0.0) {

a = x;

}

else {

b = x;

}

}

return x;

}

double iter(funct func, double a, double b, double epsa) {

double x = (a+b) / 2.0;

while(fabs(x-func(x)) > epsa) {

x = func(x);

}

return x;

}

double newton(funct func, funct der, double a, double b, double epsa) {

double previous = (a+b) / 2.0;

double next = 0.0;

while(fabs(previous-next) > epsa) {

next = previous;

previous -= func(previous)/der(previous);

}

return previous;

}

double func1(double x) {

return exp(x) + log(x) - 10.0\*x;

}

double func2(double x) {

return cos(x) - exp((-1)\*(x\*x)/2.0) + x - 1.0;

}

double func1\_iter(double x) {

return (exp(x) + log(x)) / 10.0;

}

double func2\_iter(double x) {

return exp(-(x\*x)/2) + 1 - cos(x);

}

double func1\_der(double x) {

return exp(x) + (1.0/x) - 10.0;

}

double func2\_der(double x) {

return x\*exp((-1)\*(x\*x)/2) + 1.0 - sin(x);

}

int main() {

double eps = 10000000\*Epsilon();

printf("function 1: \n");

printf("function\_of\_dichotomy| function\_of\_iterations| function\_of\_Newton \n");

double res1\_1 = dht(func1,3.0,4.0,eps);

double res1\_2 = iter(func1\_iter,3.0,4.0,eps);

double res1\_3 = newton(func1,func1\_der,3.0,4.0,eps);

printf("%.10lf | NO | %.10lf \n",res1\_1,res1\_3);

printf("\n");

printf("function 2: \n");

printf("function\_of\_dichotomy| function\_of\_iterations| function\_of\_Newton \n");

double res2\_1 = dht(func2,1.0,2.0,eps);

double res2\_2 = iter(func2\_iter,1.0,2.0,eps);

double res2\_3 = newton(func2,func2\_der,1.0,2.0,eps);

printf("%.10lf | %.10lf | %.10lf \n",res2\_1,res2\_2,res2\_3);

return 0;

}

fallfire@vb:~/kursach4$ ./kurs

function 1:

function\_of\_dichotomy| function\_of\_iterations| function\_of\_Newton

3.5264980067 | NO | 3.5264980084

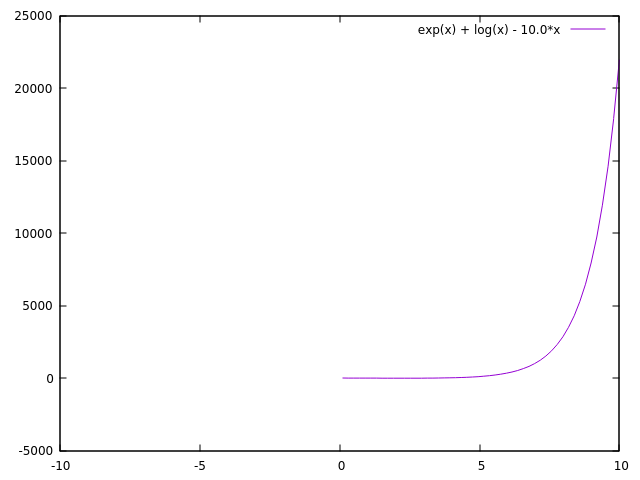
function 2:

function\_of\_dichotomy| function\_of\_iterations| function\_of\_Newton

1.0894428026 | 1.0894428022 | 1.0894428008

**Замечание**

Корень первой функции нельзя найти методом итераций, так как f1(x) имеет бесконечно много корней.



**Заключение**

Не у каждой функции можно найти корень всеми методами. Данная курсовая работа научила меня работать с функциями, вычислять корни различными методами.

Используя подпрограммы вычисления корня тремя способами, можно составить программу для вычисления этими методами любой функции.

**Список литературы**

1. Керниган, Б. Язык программирования Си: учебное пособие/ Б. Керниган, Д. Ритчи – Москва: Вильямс, 2016.
2. Подбельский, В. В. Программирование на языке Си. / В. В. Подбельский, С. С. Фомин. 2-е дополненное издание. Учебное издание. (Москва: «Финансы и статистика», 2014)
3. Метод дихотомии [Электронный ресурс].

URL: [https://en.wikipedia.org/wiki/Dichotomy](https://en.wikipedia.org/wiki/Dichotomy%20)

1. Метод итераций [Электронный ресурс].

URL: [https://en.wikipedia.org/wiki/Iterative\_method](https://en.wikipedia.org/wiki/Iterative_method%20)

1. Метод Ньютона [Электронный ресурс].

URL: [https://en.wikipedia.org/wiki/Newton%27s\_method](https://en.wikipedia.org/wiki/Newton%27s_method%20)