**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Курсовой проект**

**по курсу**

**«Фундаментальная информатика»**

**I семестр**

**Задание 4**

| Студент:  Группа:  Руководитель:  Оценка:  Дата: | Астахова А. С.  М8О-113Б-21  Никулин Н. С. |
| --- | --- |

**Москва**

**2022г.**

**Задача**

Составить программу на языке Си с процедурами решения трансцендентных алгебраических уравнений различными численными методами (итераций, Ньютона и 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 вариантом с заданным номером. Если метод не применим дать математическое обоснование и графическую иллюстрацию.

Вариант 1

; отрезок, содержащий корень [3; 4]

Вариант 8

; отрезок, содержащий корень [1; 2]

**Решение**

Описание методов:

1. Метод половинного деления (дихотомии)

Очевидно, что если на отрезке существует корень уравнения, то значения функции на концах отрезка имеют разные знаки . Метод заключается в делении отрезка пополам и его сужения 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 . Далее, вычисления проводятся по формулам: , если ; или по формулам: , , если . Процесс повторяется до тех пор, пока не будет выполнено условие окончания . Приближенное значение корня к моменту окончания итерационного процесса получается следующим образом .

Функция непрерывна на отрезке и на концах его принимает значения разных знаков: ;

Функция = непрерывна на отрезке и на концах его принимает значения разных знаков: ;

1. Метод итераций

Идея метода заключается в замене исходного уравнения уравнением вида . Достаточное условие сходимости метода: Это условие необходимо проверить перед началом решения задачи, так как функция может быть выбрана неоднозначно, причём в случае неверного выбора указанной функции метод расходится.

Начальное приближение корня: (середина исходного отрезка).

Итерационный процесс: .

Условие окончания: .

Приближённое значение корня: .

Для преобразования в используется уравнение вида , где – некоторая постоянная, знак которой совпадает со знаком производной в некоторой окрестности корня.

Условие сходимости метода итераций:

1. Метод Ньютона

Метод Ньютона — это итерационный численный метод нахождения корня(нуля) заданной функции. Метод был впервые предложен Исааком Ньютоном. Поиск решения осуществляется путём построения последовательных. Метод обладает квадратичной сходимостью и является частным случаем метода итераций. Условие сходимости метода: на отрезке . Итерационный процесс: .

**Программный код**

#include <stdio.h>

#include <math.h>

#include <locale.h>

#include <stdbool.h>

#define \_CRT\_SECURE\_NO\_WARNINGS

double f1(double x) {

return exp(x) + log(x) - 10 \* x;

}

double f2(double x) {

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

}

double f1\_iter(double x) {

return x - 1.0/20.0 \* f1(x);

}

double f2\_iter(double x) {

return x - 1.0 \* f2(x);

}

double derivative1(double x) {

return exp(x) + 1.0 / x - 10;

}

double derivative2(double x) {

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

}

double dichotomy(double a, double b, double(\*form)(double x), double eps)

{

int count = 0;

double x = 0.0;

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

count += 1;

x = (a + b) / 2.0;

if (form(a) \* form(x) > 0) {

a = x;

}

else {

b = x;

}

if (count == 150) {

break;

}

}

return x;

}

double newton(double a, double b, double(\*form)(double x), double(\*derivative)(double x), double eps)

{

int count = 0;

double x\_now = (a + b) / 2.0;

double x\_last = 0;

while (fabs(x\_now - x\_last) > eps) {

count += 1;

x\_last = x\_now;

x\_now -= form(x\_now) / derivative(x\_now);

if (count == 150) {

break;

}

}

return x\_now;

}

double iteration(double a, double b, double(\*iter\_form)(double x), double eps)

{

int count = 0;

double x\_now = (a + b) / 2.0;

double x\_last = 0.0;

while (fabs(x\_now - x\_last) > eps) {

count += 1;

x\_last = x\_now;

x\_now = iter\_form(x\_now);

if (count == 150) {

break;

}

}

return x\_now;

}

int main()

{

setlocale(LC\_ALL, "Rus");

double EPS = 1.0;

while (1.0 + EPS / 2 > 1) {

EPS /= 2;

}

printf("\t|--------------------------------------------------------------------------------------|\n");

printf("\t|%20s| %20s| %20s| %21s\n", "Метод", "Дихотомии ", "Итераций ", "Ньютона |");

printf("\t|--------------------------------------------------------------------------------------|\n");

printf("\t|%20s| %20.16f| %20.16f| %20.16f|\n", "Вариант 1", dichotomy(3.0, 4.0, f1, EPS), iteration(3.0, 4.0, f1\_iter, EPS), newton(3.0, 4.0, f1, derivative1, EPS));

printf("\t|--------------------------------------------------------------------------------------|\n");

printf("\t|%20s| %20.16f| %20.16f| %20.16f|\n", "Вариант 2", dichotomy(1.0, 2.0, f2, EPS), iteration(1.0, 2.0, f2\_iter, EPS), newton(1.0, 2.0, f2, derivative2, EPS));

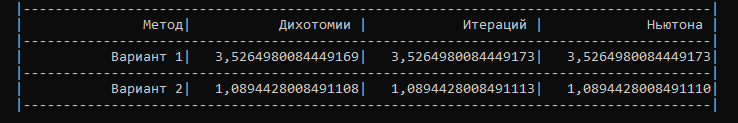
printf("\t|--------------------------------------------------------------------------------------|\n");

//printf("%f\n", EPS);

return 0;

}

**Результат работы программы**



**Заключение**

После генерации таблицы значений заданной функции можно увидеть, что значения корней уравнений, вычисленных разными способами, совпадают до 14-15 знака после запятой. Из-за того, что методы вычисления корней имеют различную «скорость» приближения к точному значению корня на данном отрезке, и один алгоритм достигает точки остановки быстрее другого, а также из-за того, что вещественные числа имеют диапазон представления в памяти компьютера, это неизбежно приводит к тому, что в вычислениях в окрестности границ данного диапазона возникают погрешности. Отсюда и небольшие различия в значениях корней.

Понятно, что 3 приведенных метода поиска корня функции не очень эффективны.В первую очередь из-за того, что нам нужно знать отрезок, на котором находится корень.Также метод итераций, например, может и не сработать для произвольной функции.Я научилась использовать функции в качестве параметров, освоила навык решения уравнений с помощью ЭВМ.