**«Поволжский государственный университет телекоммуникаций и информатики»   
Факультет информатики и вычислительной техники.**

**Математическое программирование**

Лабораторная работа№7

Вариант 8 (220153)

«*Численное решение нелинейных и трансцендентных уравнений.*»

Выполнил студент ИВТ-23  
Ермишин М.В.

Проверил:  
Чуйкова Н.А

Самара, 2025 г.

public class Lab7

{

public static void BiSectionOne()

{

static double Function(double x) => Math.Pow(x, 5) - 7 \* x + 1;

double a = -2.0;

double b = -1.5;

double tolerance = 0.000001;

double c;

while (Math.Abs(b - a) > tolerance)

{

c = (a + b) / 2;

if (Function(c) == 0.0)

break;

else if (Function(c) \* Function(a) < 0)

b = c;

else

a = c;

}

double root = (a + b) / 2;

Console.WriteLine($"Корень: {root:F6}");

}

public static void Newton()

{

static double Function(double x) => Math.Pow(x, 3) - 3 \* x - 1;

static double Derivative(double x) => 3 \* Math.Pow(x, 2) - 3;

double x0 = 1.5;

double tolerance = 0.000001;

double x = x0;

while (Math.Abs(Function(x)) > tolerance)

{

x = x - Function(x) / Derivative(x);

}

Console.WriteLine($"Корень: {x:F6}");

}

public static void ModifiedNewtonMethod()

{

static double Function(double x) => Math.Pow(x, 5) - 7 \* x + 1;

static double Derivative(double x) => 5 \* Math.Pow(x, 4) - 7;

static double FixedDerivative(double x0) => Derivative(x0); // Фиксированная производная

double x0 = 1.5;

double tolerance = 0.000001;

double x = x0;

double dfx0 = FixedDerivative(x0);

while (Math.Abs(Function(x)) > tolerance)

{

x = x - Function(x) / dfx0;

}

Console.WriteLine($"Корень: {x:F6}");

}

public static void SimpleIterationMethod()

{

static double Phi(double x) => -Math.Sqrt(12); // Итерационная функция для отрицательного корня

double x0 = -3.5;

double tolerance = 0.000001;

double x = x0;

double xNext;

do

{

xNext = Phi(x);

x = xNext;

} while (Math.Abs(xNext - x) > tolerance);

Console.WriteLine($"Корень: {x:F6}");

}

public static void BiSectionFive()

{

static double Function(double x) => 7 \* Math.Pow(x, 3) - 8 \* Math.Pow(x, 2) + 4 \* x - 3;

double a = 0.0;

double b = 2.0; // Подберите отрезок, где функция меняет знак

double tolerance = 0.000001;

double c;

while (Math.Abs(b - a) > tolerance)

{

c = (a + b) / 2;

if (Function(c) == 0.0)

break;

else if (Function(c) \* Function(a) < 0)

b = c;

else

a = c;

}

double root = (a + b) / 2;

Console.WriteLine($"Корень: {root:F6}");

}

public static void ChordAndTangentMethod()

{

static double Function(double x) => Math.Pow(x, 2) - 18 \* x - 1.7;

static double Derivative(double x) => 2 \* x - 18;

double a = 0.0;

double b = 20.0; // Подберите отрезок, где функция меняет знак

double tolerance = 0.000001;

double xChord = a;

double xTangent = b;

while (Math.Abs(xChord - xTangent) > tolerance)

{

xChord = a - Function(a) \* (b - a) / (Function(b) - Function(a));

xTangent = b - Function(b) / Derivative(b);

if (Function(xChord) \* Function(a) < 0)

b = xChord;

else

a = xChord;

b = xTangent;

}

double root = (xChord + xTangent) / 2;

Console.WriteLine($"Корень: {root:F6}");

}

}

