Міністерство освіти і науки України
Національний Технічний Університет України
Київський політехнічний інститут імені Ігоря Сікорського
Навчально-науковий інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Лабораторна робота №3

з дисципліни «Чисельні методи»

Тема «Розв’язання нелінійних рівнянь»

Варіант №22

Студента 2-го курсу НН ІАТЕ гр. ТР-12
Ковальова Олександра
Перевірила: к.т.н., доц. Залевська О. В.

КИЇВ 2022

Мета роботи. Навчитися розв'язувати нелінійні рівняння. Написати універсальну програму для обчислення нелінійного рівняння згідно варіанту. Попередньо виконати допрограмовий етап, результатом якого повинні бути проміжки, щодо яких проводиться уточнення. Перевірити розв'язок уточнення коренів за методами бісекції, хорд, дотичних у Mathcad. Знайти середньоквадратичну похибку обчислень.

Теоретична частина.

Знаходження коренів рівнянь за допомогою чисельних методів складається з двох етапів:

1)Відокремлення коренів: знаходження сукупності проміжків, кожен з яких містить один з коренів рівняння.

2) Уточнення коренів: знаходження приблизного значення коренів із заданою точністю ε. Розглядається рівняння:

$$f(x) = 0$$

Де $f(x) = P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 = 0$ та$a_n > 0$, для якого потрібно знайти його дійсні корені x^*

Перший етап виконується із використанням теорем, які допомагають виділити межі розташування додатніх та від'ємних коренів та знайти проміжки, що містять ці корені.

Теорема про границі усіх (комплексних) коренів рівняння.

Нехай $A = max|a_i|, i = 0, \dots, n-1; B = max|a_i|, i = 1, \dots, n$

Тоді всі (комплексні) корені рівняння (1) лежать у кільці

$$\frac{|a_0|}{B + |a_0|} \le |x^*| \le \frac{|a_n| + A}{|a_n|}$$

Теорема про верхню межу додатніх коренів

Нехай $A = \max|a_i|, a_i < 0; m = max_i, a_i < 0;$

Тоді $R = 1 + \sqrt[n-m]{\frac{A}{a}}$ – верхня межа додатніх коренів:

$$\forall x^* \le R, f(x^*) = 0$$

Цю ж теорему можна застосувати для визначення нижньої межи додатних коренів заміною $x = \frac{1}{y}$тоді $\forall x^* \ge \frac{1}{R_y}$. Заміна знаку x:= - y дозволяє обмежити від'ємні корені.

Теорема Гюа про наявність комплексних коренів

Якщо $\exists\, k: 1 < k < n \;\; a_k^2 < a_{k-1} \cdot a_{k+1}$, то рівняння має хоча б одну пару комплексно спряжених коренів.

Теорема Штурма про чередування коренів

Нехай $f(x) = P_n(x)$ поліном без кратних коренів. Утворимо послідовність многочленів:

$$f_0 = f(x);$$
$$f_1 = f'(x);$$
$$f_{i+1} = -[f_{i+1} mod f_i], i = 1, \dots$$

- кожний наступний многочлен є залишком від ділення двох попередніх многочленів, взятим з протилежним знаком. Стверджується, що кількість дійсних коренів полінома $f_0(x)$ на довільному відрізку $[a; b]$дорівнює різниці між кількістю змін знаку у цій послідовності при $x = a$ та $x = b$.

Другий етап передбачає застосування одного з нижченаведених методів до кожного з проміжків, отриманих на першому етапі.

Метод бісекції.

Дано: кінці інтервалу a та b, точність ε. На кожному кроці інтервал ділять навпіл

$$c = \frac{(a+b)}{2}$$

та залишають той підінтервал, до якого належить корінь.

Метод хорд.

Вхідні дані аналогічні тим, що використовуються методом бісекції. Проводиться січна до графіку функції. Точкою перетину її з віссю абсцис ділять інтервал:

$$c = \frac{a \cdot f(b) - b \cdot f(a)}{f(b) - f(a)}$$

та залишають той підінтервал, до якого належить корінь

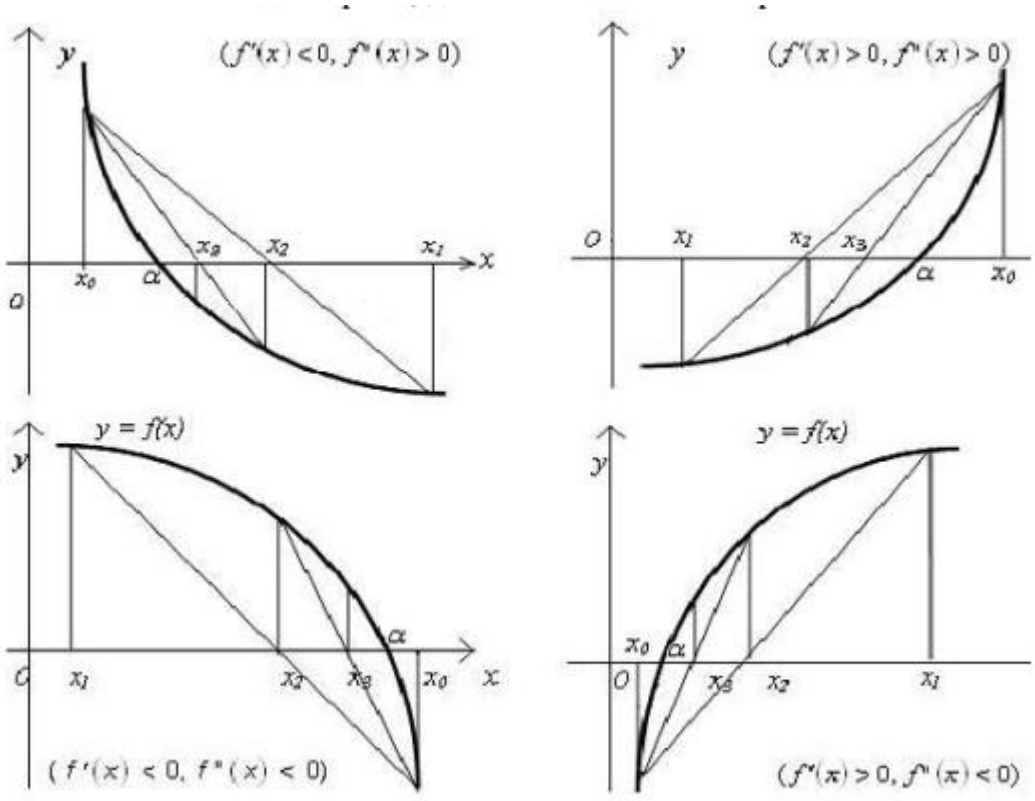


Рис.1 – Графічна інтерпретація методу хорд.

Метод Ньютона (дотичних).

Дано: початкове наближення x_0 та точність ε. Проводять дотичні до графіку функції, що дає формулу

$$x_{k-1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

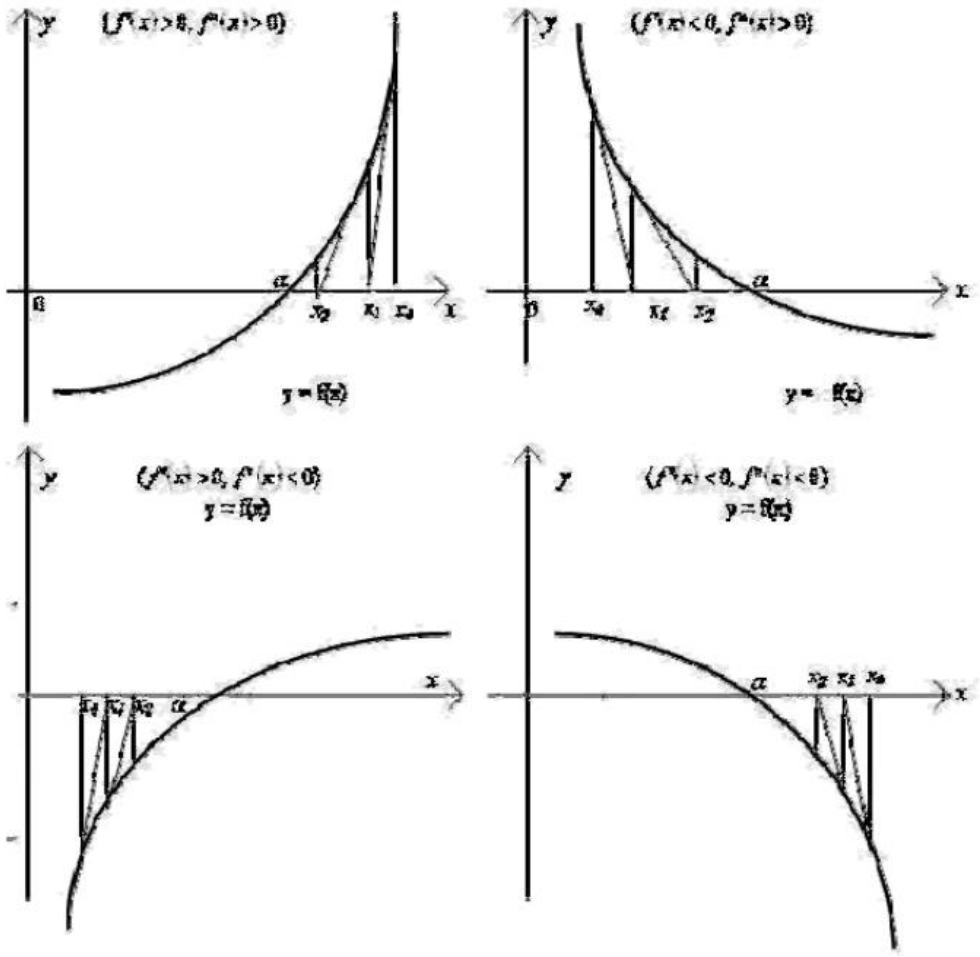


Рис.2 – Графічна інтерпретація методу дотичних.

Перевірка існування кореня на відрізку $[a; b]$здійснюється так: корінь належить відрізку, якщо $f(a) \cdot f(b) < 0$

якщо $f(a) \cdot f(b) > 0$, то відрізок не містить коренів.

Завдання.

1. Допрограмовий етап: визначити кількість дійсних коренів рівняння, відокремити корені рівняння (письмово) (див. теореми про верхню та нижню границі, Гюа, метод поліномів Штурма). Результатом є висновок: перший корінь належить проміжку […], другий корінь належить проміжку […] і т.д.

2. Програмний етап: уточнити корені рівняння:

2.1. Методом бісекції.

2.2. Методом хорд.

2.3. Методом Ньютона (дотичних).

Критерієм закінчення мають бути нерівності для методу бісекції (інтервальний метод; a та b - кінці інтервалу)

$$|b - a| < \varepsilon \text{ та } |f(x_k)| < \varepsilon$$

для методів хорд та дотичних

$$|x_k - x_{k-1}| < \varepsilon \text{ та } |f(x_k)| < \varepsilon$$

3. Порівняти отримані результати, зробити висновки, який метод приводить до меншої кількості ітерацій і чим це зумовлено.

Загальний вигляд рівняння

$$a_5(1+\propto)x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0k = 0$$

Номер варіанту - це молодша цифра Вашого номера у заліковці. Параметр k - це молодша цифра номера Вашої групи. Параметр α - старша цифра Вашого номеру у заліковці

Індивідуальне завдання:

Номер варіанту: 2

Параметр k: 2

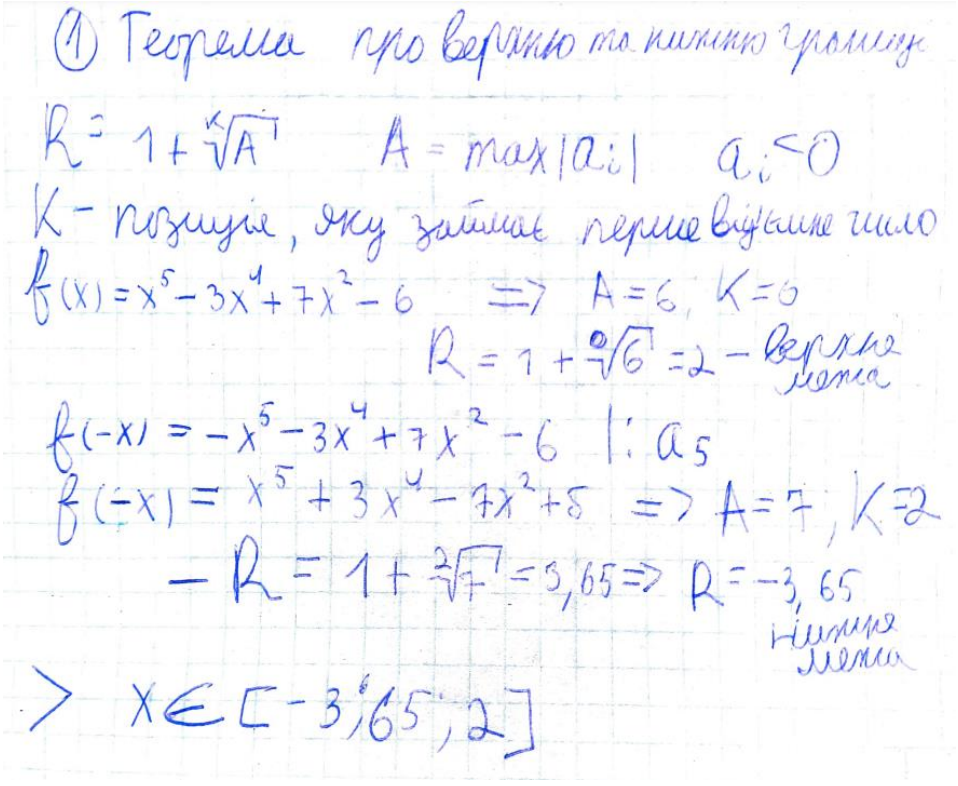
Параметр α: 0

Коефіцієнти поліному					
A5	A4	A3	A2	A1	A0
1	−3	0	7	0	−6

Отримане рівняння: $x^5 - 3x^4 + 7x^2 - 6 = 0$

Хід роботи

1. Допрограмовий етап.



② Теорема про границі дійс. коренів рівняння

$A = |\max a_i|;\ i = \overline{0, n-1} \Rightarrow A|7| = 7$

$B = |\max a_i|;\ i = \overline{1, n} \Rightarrow B = |7| = 7$

$$\frac{|a_0|}{B + |a_0|} \le |x^*| \le \frac{|a_n| + A}{|a_n|}$$

$$\frac{6}{13} \le |x| \le 8$$

$\Rightarrow x \in [0,46; 8]$

③ За теоремою Гюа про наявність комплексних коренів

$\exists k;\ 1 < k < n \qquad a_k^2 < x_{k-1} \cdot x_{k+1}$

$k = 4 \Rightarrow 9 < 10$

$k = 3 \Rightarrow 0 < (-3) \cdot 7$

$k = 2 \Rightarrow 49 < 0 \cdot 0$

$k = 1 \Rightarrow 0 < 7 \cdot (-6)$

$\Rightarrow$ Комплексних коренів немає

① Теорема Штурма про чередування коренів

$b_0 = B(x) = x^5 - 3x^4 + 7x^2 - 6$

$b_1 = B'(x) = 5x^4 - 12x^3 + 14x$

$b_2 = -(b_0 \bmod b_1) = -(-1,44x^3 + 4,2x^2 + 1,68x - 6) =$
$= 1,44x^3 - 4,2x^2 - 1,68x + 6$

$b_3 = -(b_1 \bmod b_2) = -(13,37x^2 - 3,82x - 10,76) =$
$= -13,37x^2 + 3,82x + 10,76$

$b_4 = -(b_2 \bmod b_3) = -(-1,6x + 2,95) = 1,6x - 2,95$

$b_5 = -(b_3 \bmod b_4) = -(-27,47) = 24,97$

X			
0	−	+	К-ть змін знаку на інтервалі: 5
1	+	+	$\Rightarrow$ К-ть дійсних коренів: 1
2	−	−	
3	−	−	
4	−	+	
5	+	+	$x \in [-3,65; 2]$
N	3	2	

Розглянемо таблицю, щоб з’ясувати, на якому інтервалі знаходиться дійсний корінь.

Многочлен	-∞	-2	-3/2	-1	-1/2	0	1/2	1	3/2	2	+∞
$x^5 - 3x^4 + 7x^2 - 6$	-	-	-	-	-	-	-	-	+	+	+
$5x^4 - 12x^3 + 14x$	+	+	+	+	-	0	+	+	+	+	+
$\frac{36}{25}x^3 - \frac{21}{5}x^2 - \frac{42}{25}x + 6$	-	-	-	+	+	+	+	+	-	-	+
$-\frac{1925}{144}x^2 + \frac{275}{72}x + \frac{775}{72}$	-	-	-	-	+	+	+	+	-	-	-
$\frac{864}{539}x - \frac{39744}{13475}$	-	-	-	-	-	-	-	-	-	+	+
$\frac{49441}{1800}$	+	+	+	+	+	+	+	+	+	+	+
	3	3	3	3	3	3	3	3	2	2	2

Отже, це інтервал $x \in [1; 1.5]$.

Запуск програми.

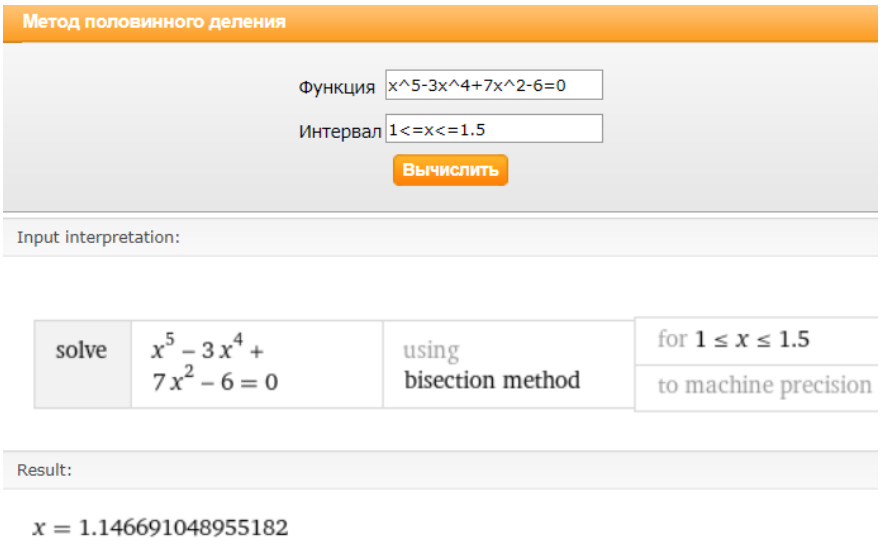
Програма для обчислень була написана мовою програмування C# у оточенні розробки JetBrains Rider. Версія мови – C# 11, платформа – .NET 7.

Спочатку вводимо максимальний степінь виразу. Потім, вводимо коефіцієнти членів. Після цього, на екран виводиться вираз для перевірки. Також, можна ввести очікувану точність та межі відрізку. Наприкінці отримуємо результати з 3 методів:

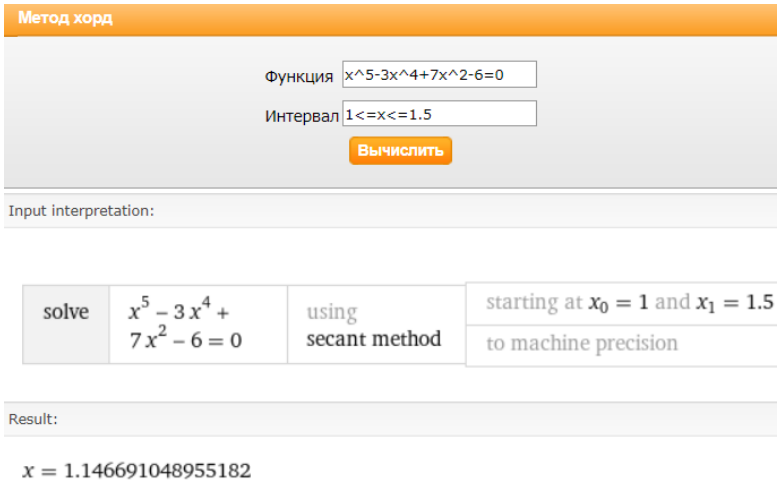
```
--- Kovalov Alex, TP-12 ---
        -- Lab 3 --

Input the degree of equation: 5

Input the coef of 0-th member of equation: -6
Input the coef of 1-th member of equation: 0
Input the coef of 2-th member of equation: 7
Input the coef of 3-th member of equation: 0
Input the coef of 4-th member of equation: -3
Input the coef of 5-th member of equation: 1

Equation:   x^5 - 3x^4 + 7x^2 - 6

Input the precision: 0,0001
Input the left border a: 1
Input the right border b: 1,5

Bisection: 1,14666748046875
Chord: 1,1467025060834752
Newtons: 1,146687526974536
```

Порахуємо результат за допомогою сервісу-калькулятора Wolfram Alpha (аналог Mathcad).

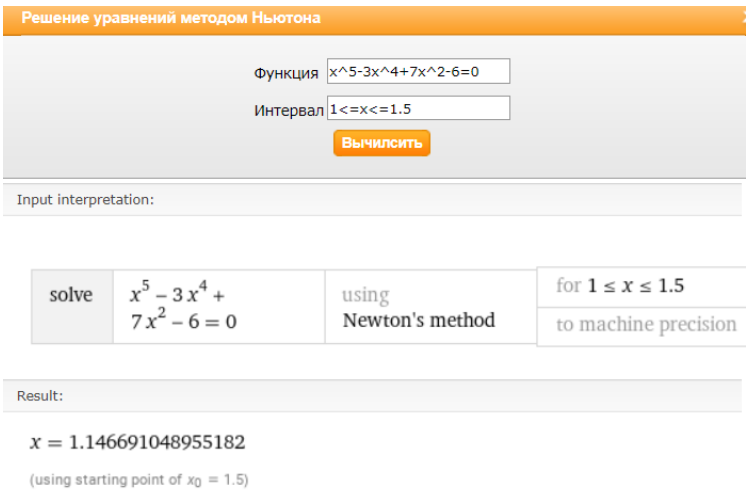
Метод бісекції:



Метод хорд:



Метод Ньютона:



Порівняємо результати калькулятора з результатами нашої програми використовуючи формулу:

$$\delta = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(x_k - x_{km})^2}$$

Для порівняння буде використовуватись Mathcad:

ТР-12 Ковальов Олександр, варіант 22

Метод бісекції:

$x := 1.14666748046875 \qquad x_m := 1.146691048955182$

$\delta := \sqrt[2]{(x - x_m)^2} = 23.57 \cdot 10^{-6}$

Метод хорд:

$x := 1.1467025060834752 \qquad x_m := 1.146691048955182$

$\delta := \sqrt[2]{(x - x_m)^2} = 11.46 \cdot 10^{-6}$

Метод Ньютона:

$x := 1.146687526974536 \qquad x_m := 1.146691048955182$

$\delta := \sqrt[2]{(x - x_m)^2} = 3.52 \cdot 10^{-6}$

Висновок. В ході виконання лабораторної роботи було розглянуто та опрацьовано на практиці розв'язання нелінійних рівнянь. Написано програму, яка знаходить корені рівняння на заданих проміжках за допомогою методів бісекції, хорд та Ньютона (дотичних). Установлено різницю між використанням кожного метода та їх ефективність. З'ясовано, що метод Ньютона (дотичних) найефективніший, бо чим більше зростає графік – тим більше кутовий коефіцієнт дотичної. Порівняно отримані результати із калькулятором.

Додатки.

Код програми (Replit): https://replit.com/join/hbsbvmbxvr-kovalevalex
Код програми:

Program.cs

```
namespace Lab3;

public class Program
{
    public static void Main(string[] args)
    {
        Console.WriteLine("--- Kovalov Alex, TP-12 ---");
        Console.WriteLine("        -- Lab 3 --      \n");

        Console.Write("Input the degree of equation: ");
        int n = int.Parse(Console.ReadLine()!);

        Console.WriteLine();

        NonLinearEquation? equation = null;
        NonLinearEquation.Create(ref equation, n);

        if (equation == null)
        {
            Console.WriteLine("No equation");
            return;
        }

        Console.WriteLine();
        Printer.Equation(equation);
        Console.WriteLine();

        Console.Write("Input the precision: ");
        var precision = double.Parse(Console.ReadLine()!);

        Console.Write("Input the left border a: ");
        var a = double.Parse(Console.ReadLine()!);

        Console.Write("Input the right border b: ");
        var b = double.Parse(Console.ReadLine()!);

        Console.WriteLine();

        Calculations.Bisection(equation, a, b, precision);
        Calculations.Chord(equation, a, b, precision);
        Calculations.Newtons(equation, a, b, precision);
    }
}
```

Calculations.cs

```
namespace Lab3;

public static class Calculations
{
     public static void Bisection(NonLinearEquation equation, double a, double
b, double precision)
     {
          double fa = Function(equation, a), fb = Function(equation, b);
          double c, fc;
          do {
               c = (a + b) / 2.0;
```

```
                        fc = Function(equation, c);
                        if (fa * fc < 0) {
                              b = c;
                              fb = fc;
                        } else if (fc * fb < 0) {
                              a = c;
                              fa = fc;
                        }
                  } while (Math.Abs(b - a) > precision && Math.Abs(fc) > precision);

                  Console.WriteLine($"Bisection: {c}");
            }

            public static void Chord(NonLinearEquation? equation, double a, double b,
double precision)
            {
                  var fa = Function(equation, a);
                  var fb = Function(equation, b);
                  double x = a, fx, buf;

                  do {
                        buf = x;
                        x = (a * fb - b * fa) / (fb - fa);
                        fx = Function(equation, x);
                        if (fa * fx < 0) {
                              b = x;
                              fb = fx;
                        } else if (fx * fb < 0) {
                              a = x;
                              fa = fx;
                        }
                  } while (Math.Abs(x - buf) > precision && Math.Abs(fx) > precision);

                  Console.WriteLine($"Chord: {x}");
            }

            public static void Newtons(NonLinearEquation equation, double a, double b,
double precision)
            {
                  NonLinearEquation? derivative = null;
                  Derivative(ref derivative, equation);

                  var fa = Function(equation, a);
                  var fb = Function(equation, b);
                  double x = a, fx = fa, buf;

                  do {
                        buf = x;

                        double dx = DerivativeValue(derivative, x);

                        x -= (fx / dx);
                        fx = Function(equation, x);
                        if (fa * fx < 0) {
                              fb = fx;
                        }
                        else if ((fx * fb) < 0) {
                              fa = fx;
                        }
                  } while (Math.Abs(x - buf) > precision && Math.Abs(fx) > precision);

                  Console.WriteLine($"Newtons: {x}");
            }
```

```
            private static double Function(NonLinearEquation? equation, double x)
            {
                  var item = equation;
                  double f = 0;

                  while (item != null)
                  {
                        f += (item.Coef * Math.Pow(x, item.Degree));
                        item = item.Next;
                  }

                  return f;
            }

            private static void Derivative(ref NonLinearEquation? derivative,
NonLinearEquation? equation)
            {
                  var item = equation;
                  while (item != null)
                  {
                        var coef = item.Coef * item.Degree;
                        if (coef != 0)
                        {
                              derivative = new NonLinearEquation(coef, item.Degree - 1,
derivative);
                        }

                        item = item.Next;
                  }
            }

            private static double DerivativeValue(NonLinearEquation? derivative, double
x)
            {
                  var item = derivative;
                  double f = 0;

                  while (item != null)
                  {
                        f += (item.Coef * Math.Pow(x, item.Degree));
                        item = item.Next;
                  }

                  return f;
            }
      }
```

NonLinearEquation.cs

```
      namespace Lab3;

      public class NonLinearEquation
      {
          public double Coef { get; }
          public int Degree { get; }
          public NonLinearEquation? Next { get; }

          public NonLinearEquation(double coef, int degree, NonLinearEquation?
equation) {
              Coef = coef;
              Degree = degree;
              Next = equation;
          }

          public static void Create(ref NonLinearEquation? equation, int degree)
```

```
    {
        for (int i = 0; i < degree + 1; i++) {
            Console.Write($"Input the coef of {i}-th member of equation: ");
            var k = double.Parse(Console.ReadLine()!);

            if (k == 0) continue;

            equation = new NonLinearEquation(k, i, equation);
        }
    }
}
```

Printer.cs

```
namespace Lab3;

public class Printer
{
    public static void Member(double coef, int degree) {
        if (coef != 1 && degree != 0)
        {
            Console.Write($"{coef}x^{degree}");
        }
        else if (degree != 0)
        {
            Console.Write($"x^{degree}");
        }
        else
        {
            Console.Write(coef);
        }
    }

    public static void Equation(NonLinearEquation? equation) {
        if (equation == null) {
            Console.WriteLine("No equation.");
            return;
        }

        Console.Write("Equation:   ");
        Member(equation.Coef, equation.Degree);
        var item = equation.Next;

        while (item != null)
        {
            Console.Write(item.Coef > 0 ? " + " : " - ");
            Member(Math.Abs(item.Coef), item.Degree);
            item = item.Next;
        }
        Console.WriteLine();
    }
}
```