ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. О. ГОНЧАРА ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА ТЕХНОЛОГІЙ ПРОГРАМУВАННЯ

Лабораторна робота №2

на тему «Наближення функцій алгебраїчними многочленами»

з курсу «Методи обчислень»

Варіант № 7

Виконав:

студент групи ПА-19-2

Ільяшенко Єгор

Дніпро, 2021

Зміст

[Постановка задачі наближення функцій 3](#_Toc69773332)

[Основні теоретичні відомості 4](#_Toc69773333)

[Інтерполяційна формула Лагранжа. Залишок 4](#_Toc69773334)

[Поділені різниці та їх властивості 7](#_Toc69773335)

[Інтерполяційні формули Ньютона. Залишок 11](#_Toc69773336)

[Середньоквадратичне наближення функцій. Похибка 14](#_Toc69773337)

[Чисельний експеримент та аналіз результатів 20](#_Toc69773338)

[Опис програмної реалізації 21](#_Toc69773339)

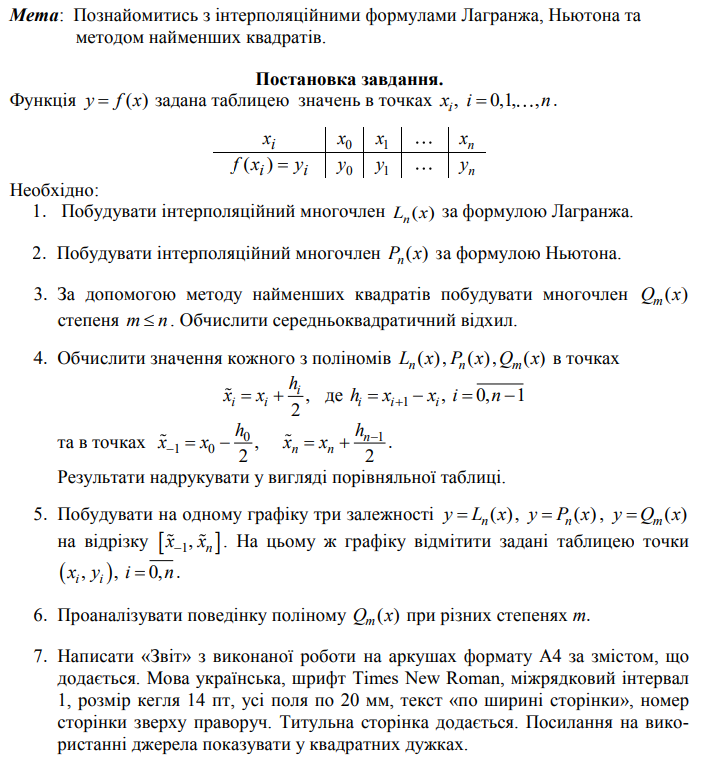
[Аналіз результатів 22](#_Toc69773340)

[Висновки 23](#_Toc69773341)

[Перелік використаних джерел 24](#_Toc69773342)

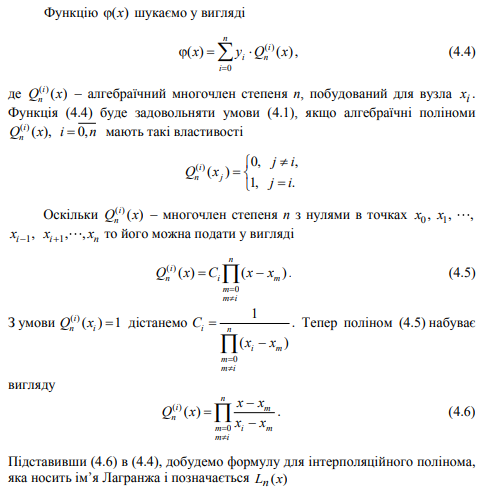
[Додаток. Код програми 25](#_Toc69773343)

# Постановка задачі наближення функцій



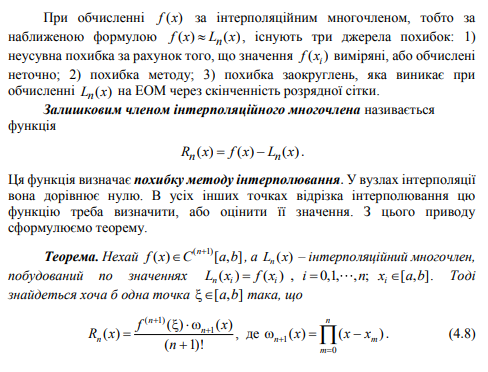
# Основні теоретичні відомості

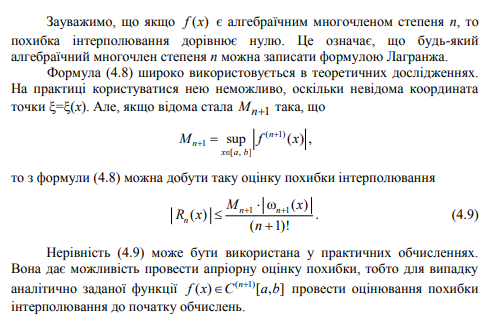
# Інтерполяційна формула Лагранжа. Залишок

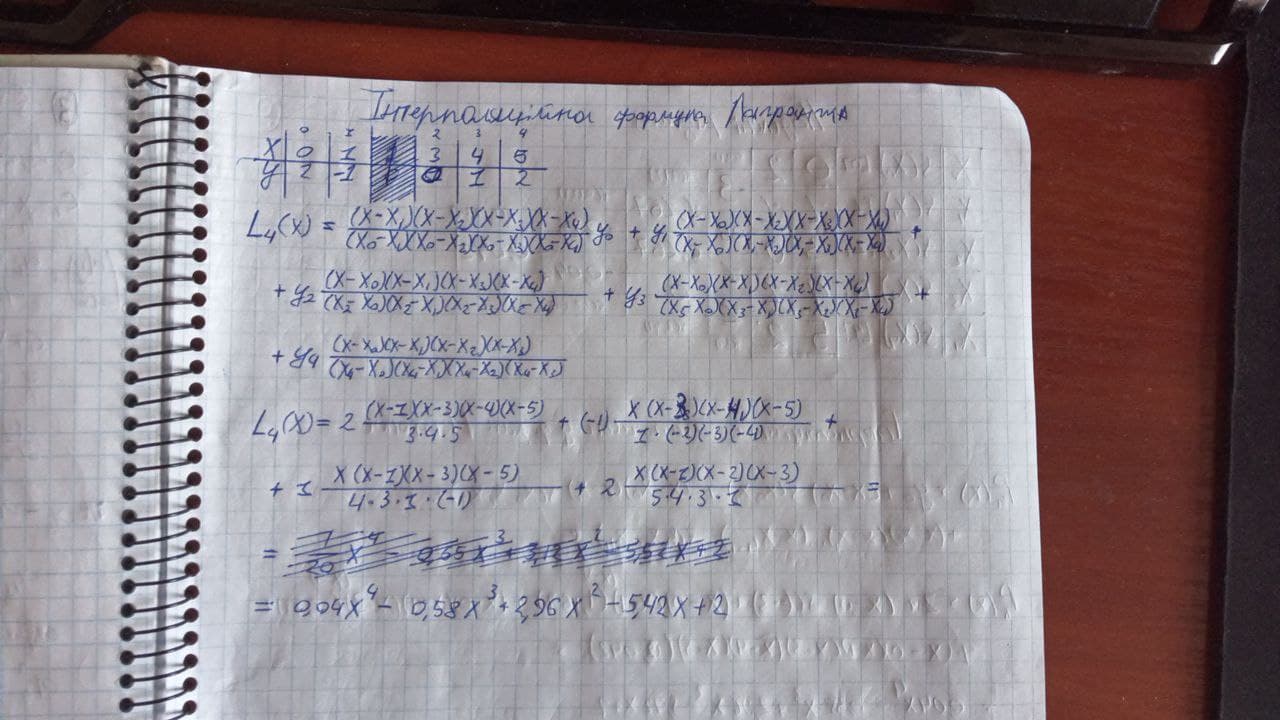




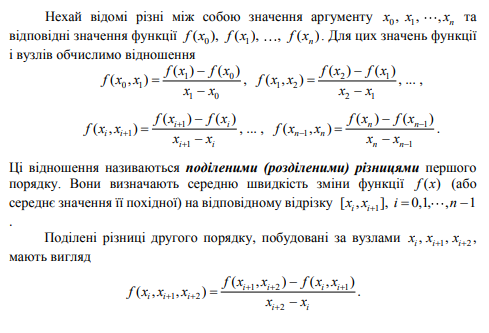


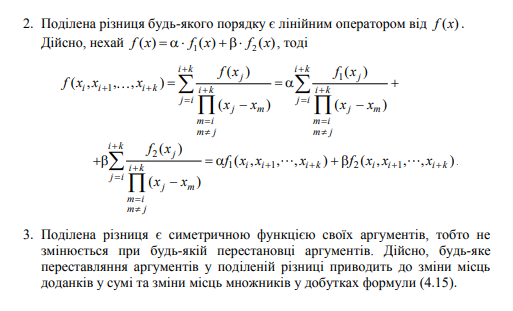
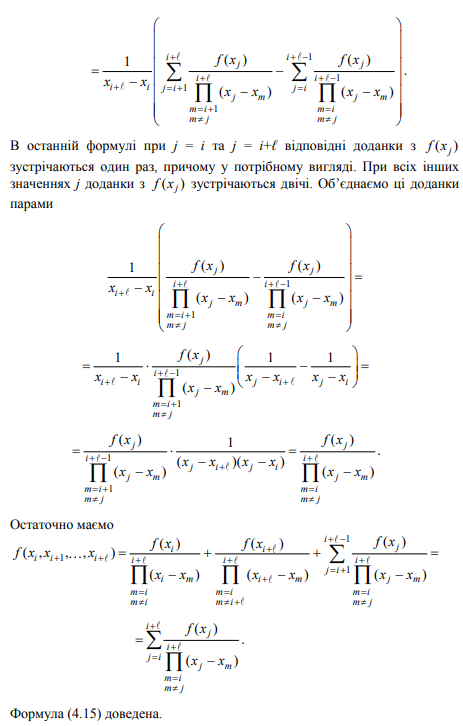
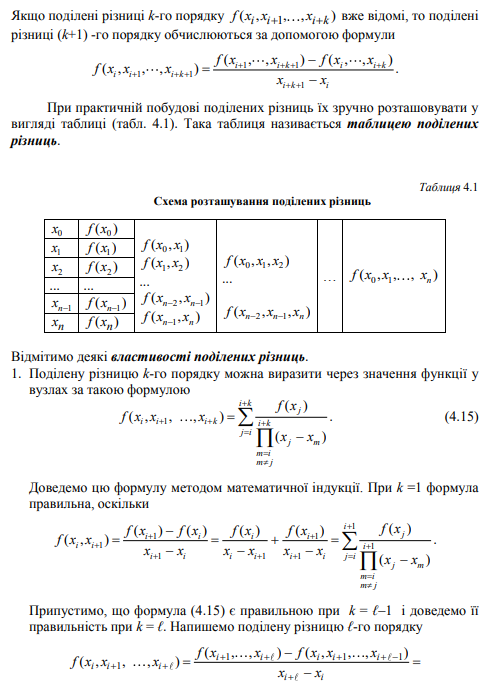






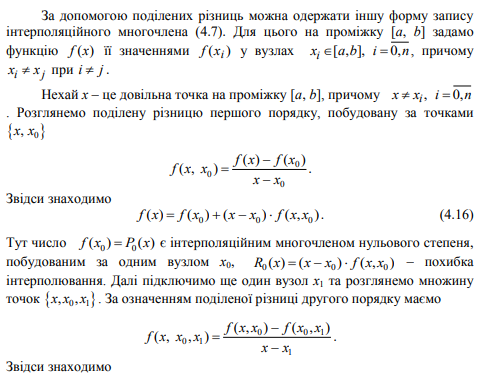
# Поділені різниці та їх властивості

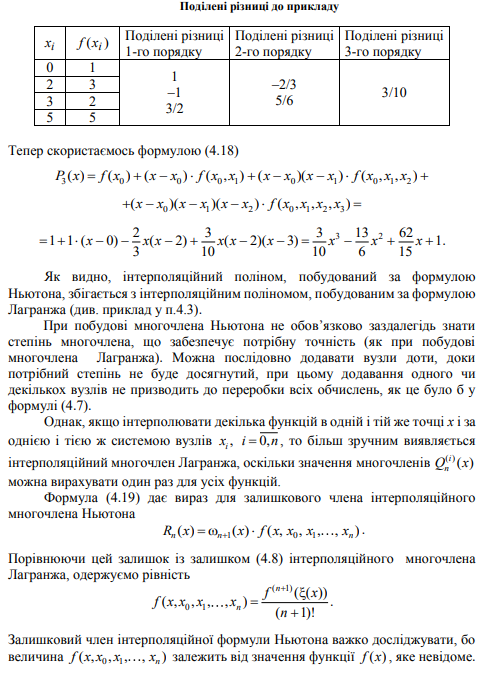
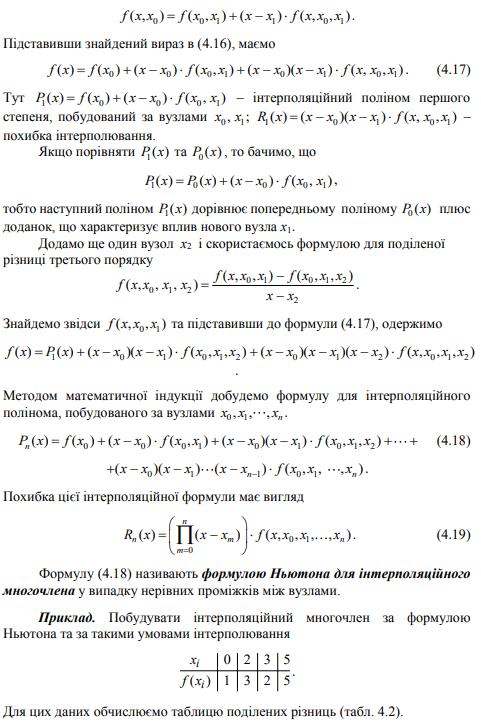


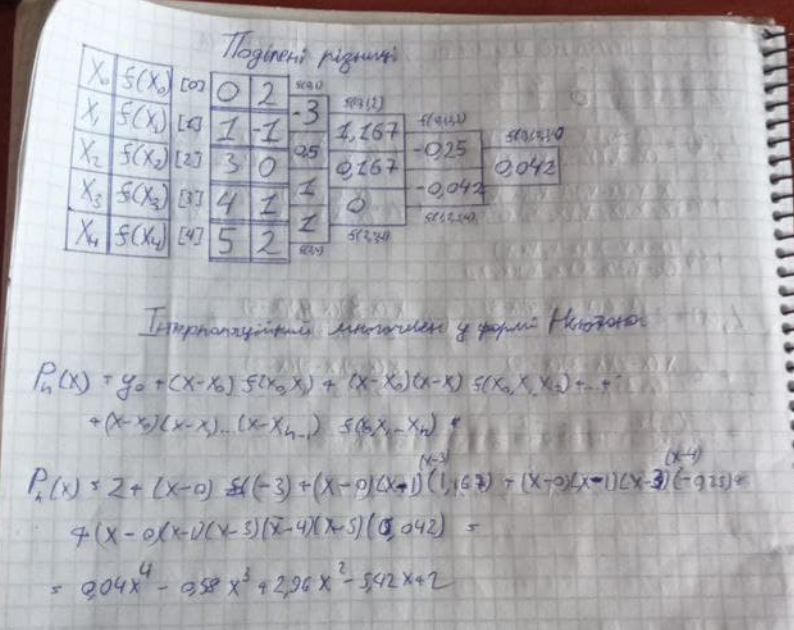




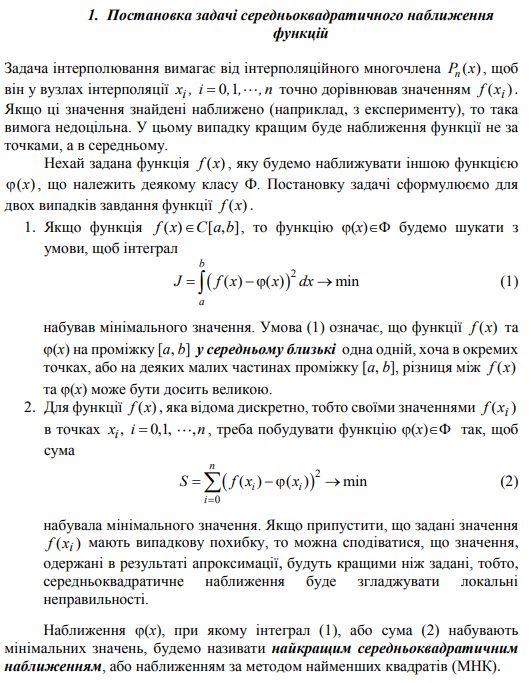
# Інтерполяційні формули Ньютона. Залишок

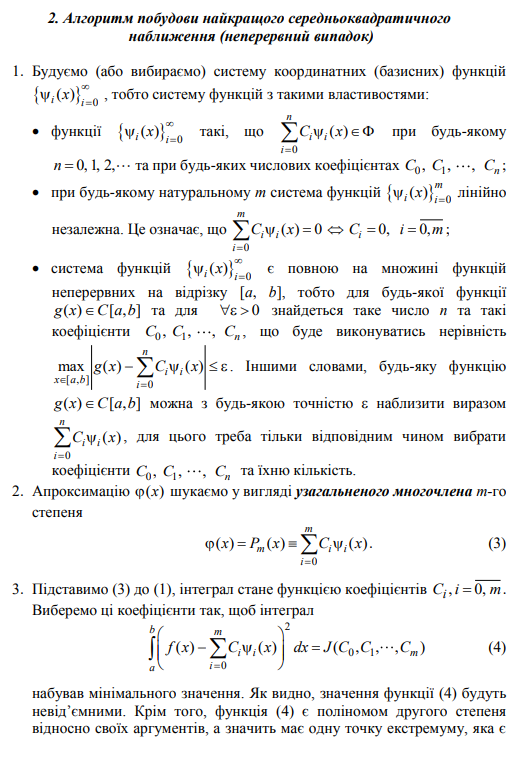


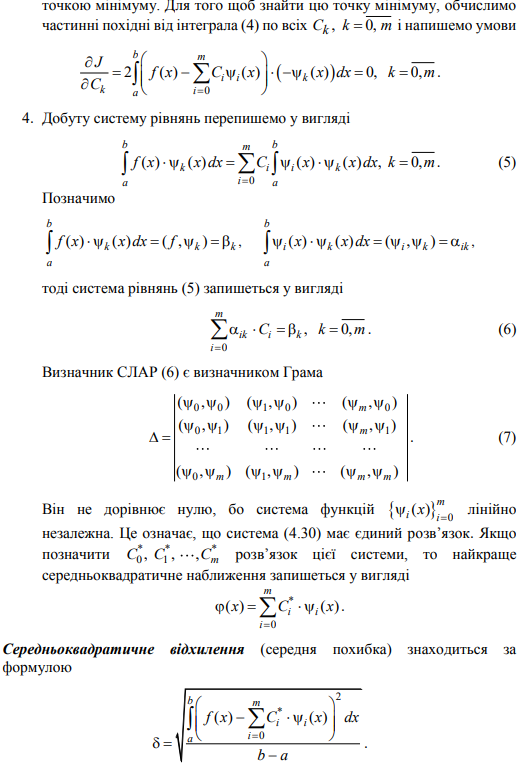


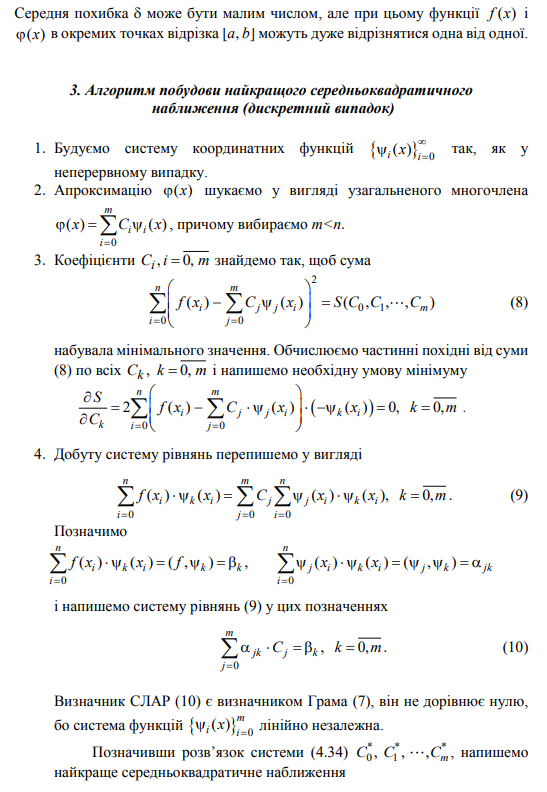


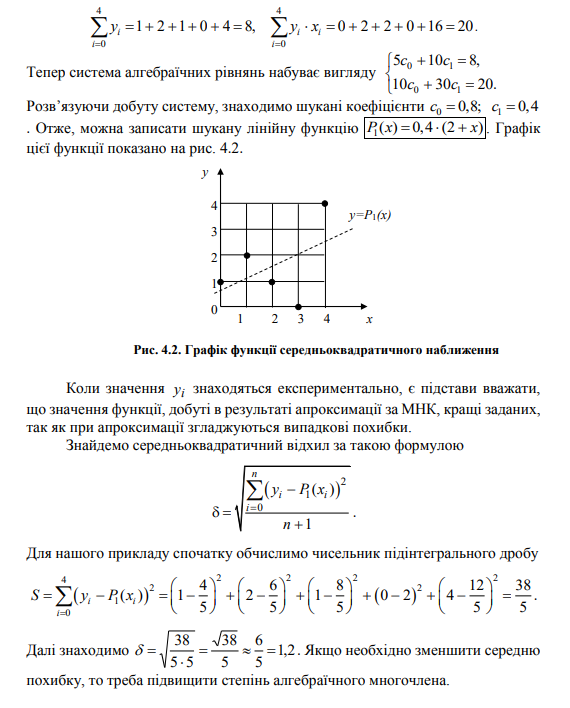
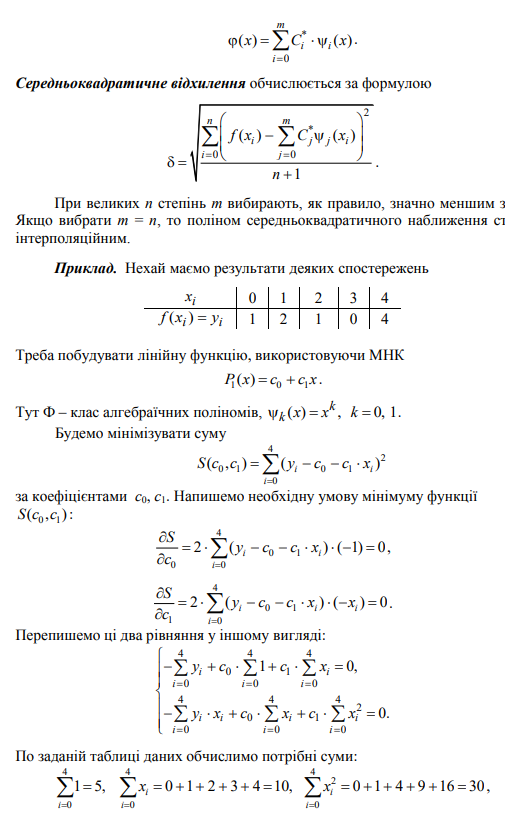
# Середньоквадратичне наближення функцій. Похибка

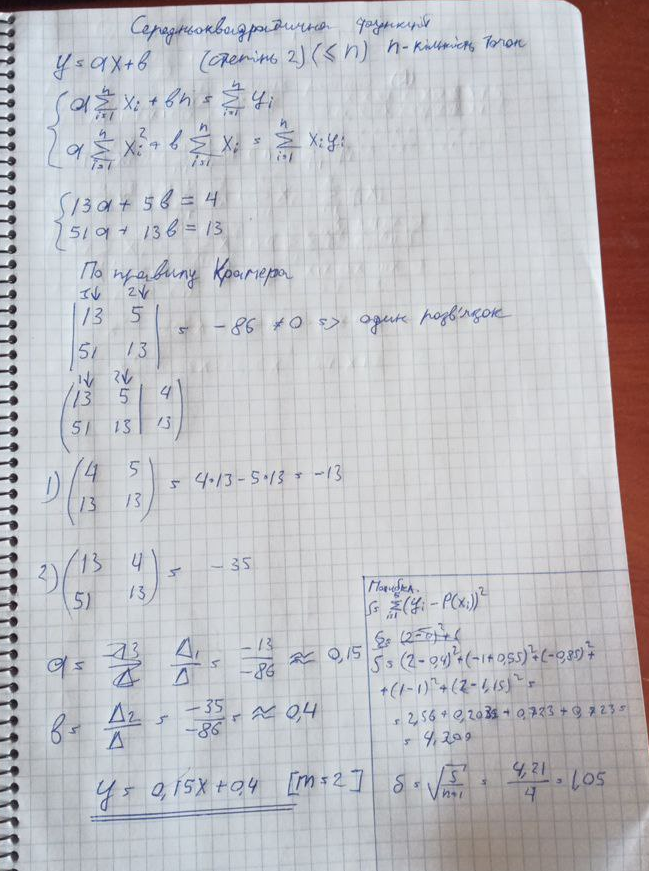




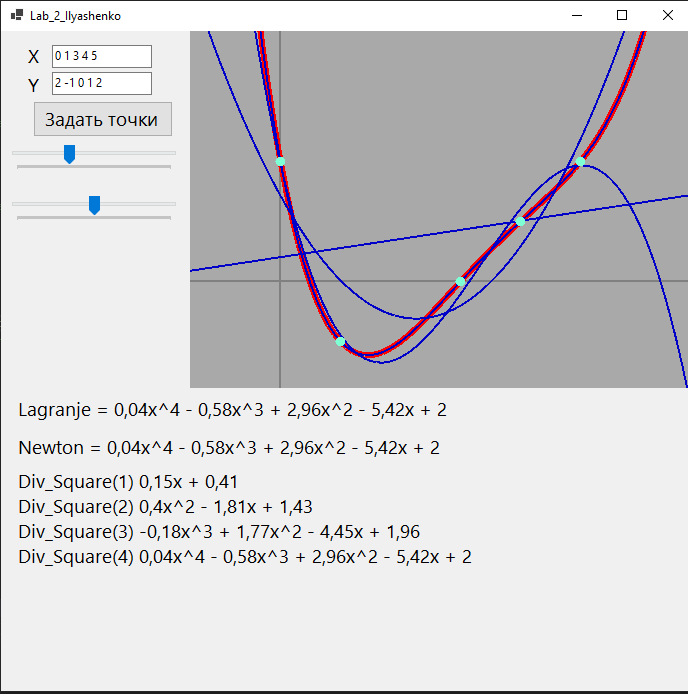








# Опис програмної реалізації



Через пробел задаются точки Х и У. Ползулками можно сдвинуть график. Красным нарисован полином по функции Лагранжа и Ньютона, а синим – полином построенный по методу разделённых квадратов в степени M. (M<=N) (M>=1)

# Аналіз результатів

При інтерполюванні функції методами Лагранжа та Ньютона, отримана функція завжди має співпадати. Більш того, функція, отримана методом розділених квадратів у степені (N-1), де N – кількість введених точок, теж повинна співпадати с двома іншими.

# Висновки

Виконуючи лабораторну роботу #2, я зрозумів, як можна будувати функцію по заданих точках (приблизно), зрозумів, які є методи інтерполювання функції, та як їх викорістовувати. Інтерполював функцію у зошиті трьома способами, використовуючи данні, надані у моєму варіанту. Також зробив программу, яка сама їх вираховувала. Зробив графік функції, на якому відобразив усі графіки, а також 5 точок, які були введені по моєму варіанту.

# Перелік використаних джерел

1. Бойко Л.Т. Основи чисельних методів: навчальний посібник. – Д.: Вид-во ДНУ, 2009. – 244 с.
2. Шахно С.М. Практикум з чисельних методів: навч. посібник [Текст] / С.М. Шахно, А.Т. Дудикевич, С.М. Левицька. – Львів: ЛНУ імені Івана Франка. 2013. – 432 с.

# Додаток. Код програми

using System;

using System.Collections.Generic;

using System.Text;

using MathNet.Numerics.LinearAlgebra;

using MathNet.Numerics;

using MathNet.Numerics.Differentiation;

namespace Lab\_2\_Methods

{

public class MyInput

{

public int[] x\_input;

public int[] y\_input;

public Polynomial lagranje;

public Polynomial newton;

public Polynomial div\_square;

public struct P\_R

{

public double[] podil\_rizn;

}

public P\_R[] pod\_r;

public MyInput()

{

}

public void AddPoints(string x, string y)

{

string[] splitted = x.Split(' ');

x\_input = new int[splitted.Length];

for (int i = 0; i < splitted.Length; i++)

x\_input[i] = int.Parse(splitted[i]);

splitted = y.Split(' ');

y\_input = new int[splitted.Length];

for (int i = 0; i < splitted.Length; i++)

y\_input[i] = int.Parse(splitted[i]);

}

public void BuildLagranjePolynome()

{

lagranje = new Polynomial(0, 0);

for (int i = 0; i < x\_input.Length; i++)

{

Polynomial buf = new Polynomial(1, 0);

buf \*= y\_input[i];

for (int j = 0; j < x\_input.Length; j++)

{

if (j == i)

continue;

buf \*= new Polynomial(-x\_input[j], 1);

buf /= (x\_input[i] - x\_input[j]);

}

lagranje += buf;

}

}

public void BuildNewtonPolynome()

{

pod\_r = new P\_R[x\_input.Length - 1];

int lader\_count = x\_input.Length - 1;

for (int i = 0; i < pod\_r.Length; i++)

{

pod\_r[i].podil\_rizn = new double[lader\_count];

lader\_count--;

}

for (int i = 0; i < pod\_r.Length; i++)

{

if (i == 0)

{

for (int j = 0; j < pod\_r[0].podil\_rizn.Length; j++)

{

pod\_r[0].podil\_rizn[j] = ((double)y\_input[j + 1] - y\_input[j]) / ((double)x\_input[j+1] - x\_input[j]);

}

}

else

{

for (int j = 0; j < pod\_r[i].podil\_rizn.Length; j++)

{

pod\_r[i].podil\_rizn[j] = (pod\_r[i-1].podil\_rizn[j + 1] - pod\_r[i-1].podil\_rizn[j]) / ((double)x\_input[j + 1 + i] - x\_input[j]);

}

}

}

newton = new Polynomial(0, 0);

Polynomial buf = new Polynomial(0, 0);

Polynomial buf2;

newton += y\_input[0];

for (int i = 0; i < x\_input.Length - 1; i++)

{

buf = new Polynomial(1, 0);

for (int j = 0; j <= i; j++)

{

buf2 = new Polynomial(-x\_input[j], 1);

buf \*= buf2;

}

buf \*= pod\_r[i].podil\_rizn[0];

newton += buf;

}

}

//public double[,] slar;

public void BuildDivSquarePolynome(int power)

{

//Polynomial line;

double[,] slar = new double[power+1, power+1];

double[] right\_part = new double[power + 1];

double[] koef = new double[power + 1];

for (int i = 0; i < power+1; i++)

{

for (int j = 0; j < power + 1; j++)

{

double sum = 0;

for (int k = 0; k < x\_input.Length; k++)

{

sum += Math.Pow(x\_input[k], power + i - j);

}

slar[i, j] = sum;

}

}

for (int i = 0; i < power + 1; i++)

{

double sum = 0;

for (int k = 0; k < x\_input.Length; k++)

{

sum += y\_input[k]\*Math.Pow(x\_input[k], i);

}

right\_part[i] = sum;

}

Matrix<double> slar\_matrix = Matrix<double>.Build.DenseOfArray(slar);

double determinant = slar\_matrix.Determinant();

double[,] buf\_slar = new double[power + 1, power + 1];

for (int i = 0; i < power + 1; i++)

{

for (int b = 0; b < power + 1; b++)

for (int g = 0; g < power + 1; g++)

{

buf\_slar[b, g] = slar[b, g];

}

for (int j = 0; j < power + 1; j++)

{

buf\_slar[j, i] = right\_part[j];

}

Matrix<double> koef\_find = Matrix<double>.Build.DenseOfArray(buf\_slar);

koef[i] = koef\_find.Determinant() / determinant;

}

double[] koef\_buf = new double[koef.Length];

for (int i = 0; i < koef.Length; i++)

{

koef\_buf[i] = koef[koef.Length - i - 1];

}

div\_square = new Polynomial(koef\_buf);

}

public string OutputNewton()

{

string output = "";

Polynomial out\_p = newton.Clone();

for (int i = 0; i < out\_p.Coefficients.Length; i++)

out\_p.Coefficients[i] = Math.Round(out\_p.Coefficients[i], 2);

output = out\_p.ToStringDescending();

return output;

}

public string OutputLagranje()

{

string output = "";

Polynomial out\_p = lagranje.Clone();

for (int i = 0; i < out\_p.Coefficients.Length; i++)

out\_p.Coefficients[i] = Math.Round(out\_p.Coefficients[i], 2);

output = out\_p.ToStringDescending();

return output;

}

public string OutputDivSquare()

{

string output = "";

Polynomial out\_p = div\_square.Clone();

for (int i = 0; i < out\_p.Coefficients.Length; i++)

out\_p.Coefficients[i] = Math.Round(out\_p.Coefficients[i], 2);

output = out\_p.ToStringDescending();

return output;

}

public (double, double) LagranjePoint(double x, int accuracy)

{

return (x, Math.Round(lagranje.Evaluate(x), accuracy));

}

public (double, double) NewtonPoint(double x, int accuracy)

{

return (x, Math.Round(newton.Evaluate(x), accuracy));

}

public (double, double) DivSquarePoint(double x, int accuracy)

{

return (x, Math.Round(div\_square.Evaluate(x), accuracy));

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MathNet.Numerics;

using MathNet.Numerics.LinearAlgebra;

namespace Lab\_2\_Methods

{

public partial class Lab\_2\_Methods\_Form : Form

{

public MyInput input;

public Lab\_2\_Methods\_Form()

{

InitializeComponent();

input = new MyInput();

gr = pictureBox1.CreateGraphics();

gr.Clear(Color.DarkGray);

}

Graphics gr;

private void button1\_Click(object sender, EventArgs e)

{

gr.Clear(Color.DarkGray);

input.AddPoints(textBox\_X.Text, textBox\_Y.Text);

input.BuildLagranjePolynome();

label1.Text = "Lagranje = " + input.OutputLagranje();

input.BuildNewtonPolynome();

label2.Text = "Newton = " + input.OutputNewton();

DrawMethods();

}

void DrawMethods()

{

PointF buf = new PointF(0 + 50, (float)input.LagranjePoint(0 + 50, 3).Item2 \* 10);

Pen pen = new Pen(Color.Red, 6);

List<PointF> points = new List<PointF>();

gr.DrawLine(new Pen(Color.Gray, 2), ToMap(new PointF(-500, 0)), ToMap(new PointF(500, 0)));

gr.DrawLine(new Pen(Color.Gray, 2), ToMap(new PointF(0, 550)), ToMap(new PointF(0, -500)));

for (double i = -2; i < 10; i += 0.1)

{

points.Add(new PointF((float)i, -(float)input.LagranjePoint(i, 3).Item2));

//gr.DrawEllipse(pen, new RectangleF((float)i\*30+250, -(float)input.LagranjePoint(i, 3).Item2\*30+250, 1, 1));

}

for (int i = 0; i < points.Count - 1; i++)

{

gr.DrawLine(pen, ToMap(points[i]), ToMap(points[i + 1]));

//gr.DrawEllipse(pen, new RectangleF((float)i\*30+250, -(float)input.LagranjePoint(i, 3).Item2\*30+250, 1, 1));

}

Random rnd = new Random();

label3.Text = "";

for (int k = 1; k < input.x\_input.Length; k++)

{

pen = new Pen(Color.FromArgb(0,0,200), 2);

input.BuildDivSquarePolynome(k);

points.Clear();

for (double i = -2; i < 10; i += 0.1)

{

points.Add(new PointF((float)i, -(float)input.DivSquarePoint(i, 3).Item2));

//gr.DrawEllipse(pen, new RectangleF((float)i\*30+250, -(float)input.LagranjePoint(i, 3).Item2\*30+250, 1, 1));

}

for (int i = 0; i < points.Count - 1; i++)

{

gr.DrawLine(pen, ToMap(points[i]), ToMap(points[i + 1]));

//gr.DrawEllipse(pen, new RectangleF((float)i\*30+250, -(float)input.LagranjePoint(i, 3).Item2\*30+250, 1, 1));

}

label3.Text += "Div\_Square(" + k + ") " + input.OutputDivSquare() + "\n";

}

for (int i = 0; i < input.x\_input.Length; i++)

{

gr.FillEllipse(new SolidBrush(Color.Aquamarine), new RectangleF(new PointF(ToMap(new PointF(input.x\_input[i], -input.y\_input[i])).X-5, ToMap(new PointF(input.x\_input[i], -input.y\_input[i])).Y-5), new SizeF(10, 10)));

//gr.DrawLine(new Pen(Color.Aquamarine, 6), ToMap(new PointF(input.x\_input[i], -input.y\_input[i])), ToMap(new PointF(input.x\_input[i]+0.1f, -input.y\_input[i]+0.1f)));

}

}

PointF ToMap(PointF point)

{

point.X \*= 60;

point.Y \*= 60;

point.X += 250 + trackBar1.Value;

point.Y += 250 + trackBar2.Value;

return point;

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

gr.Clear(Color.DarkGray);

DrawMethods();

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

gr.Clear(Color.DarkGray);

DrawMethods();

}

}

}