Національний авіаційний університет

Факультет комп’ютерних наук та технологій

**ЗВІТ**

**по лабораторній роботі No 6**

**Дискримінантний аналіз**

Дисципліна: «Методи штучного інтелекту»

Кафедра: прикладної математики

ОС: бакалавр

Спеціальність: 113 «Прикладна математика»

ОПП: «Прикладне програмне забезпечення»

Виконав: здобувач вищої освіти 4 курсу. 451 групи

Архіпов Олексій Тімурович

Перевірив: Приставка Пилип Олександрович

Київ 2024

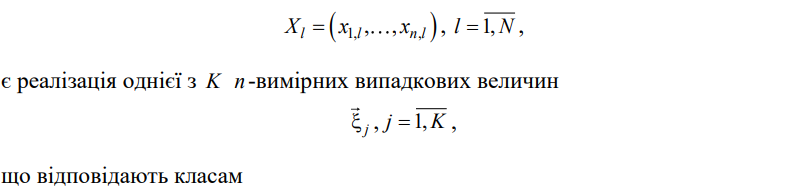
**Тема:** Дискримінантний аналіз

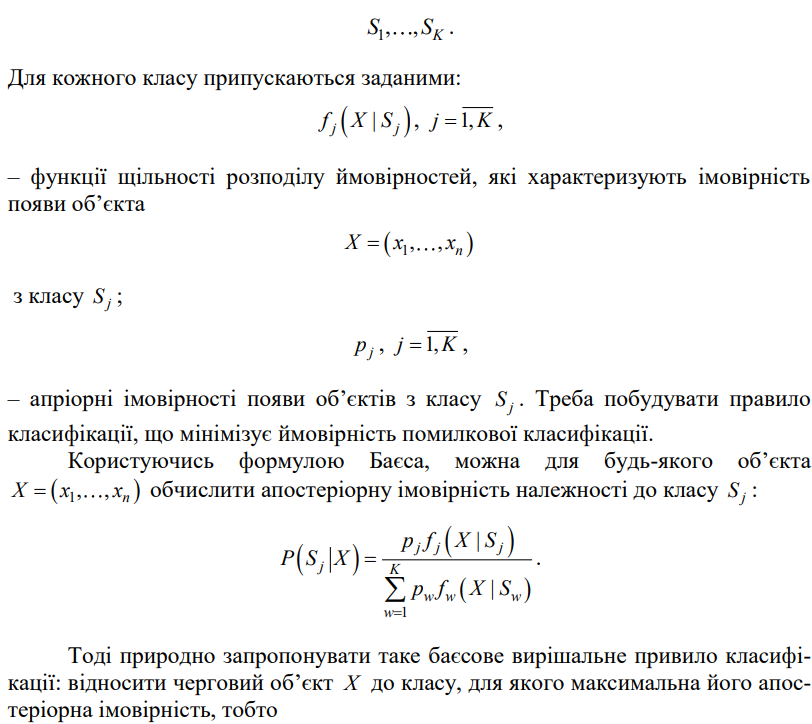
**Завдання:**

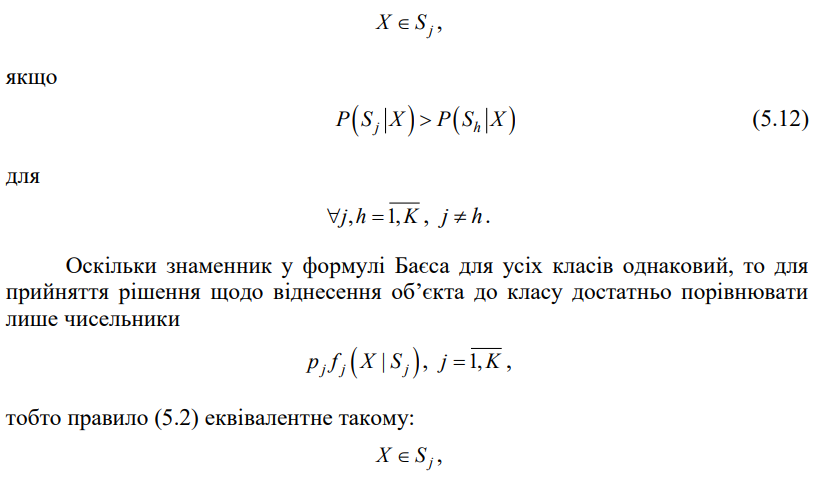
* Реалізувати дискримінантний аналіз

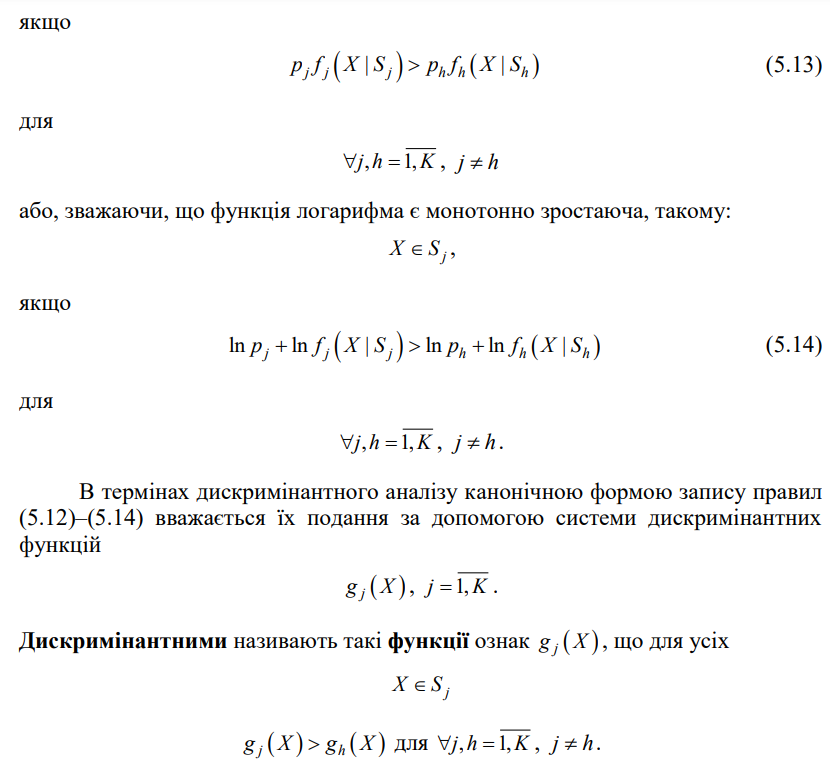
**Теоретична частина**

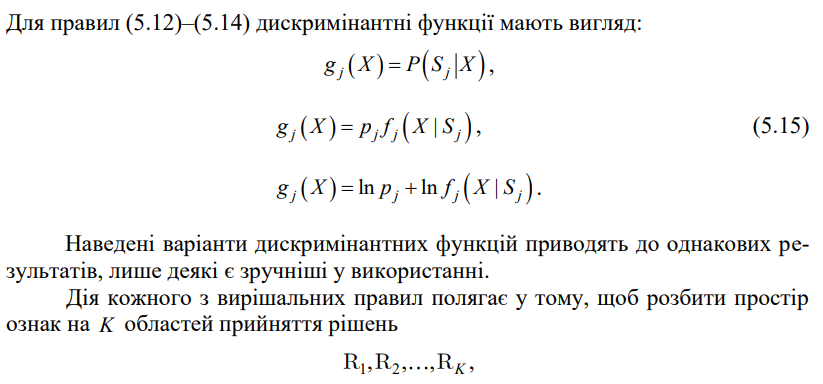
Прикладом басового підходу в задачі розпізнавання образів є дискримінантний аналіз, метою якого є не просто вирішення задачі класифікації, а й побудови дискримінантних границь (роздільних меж) між класами. В контексті дискримінантного аналізу ймовірнісна постановка задачі класифікації припускає, що кожне спостереження

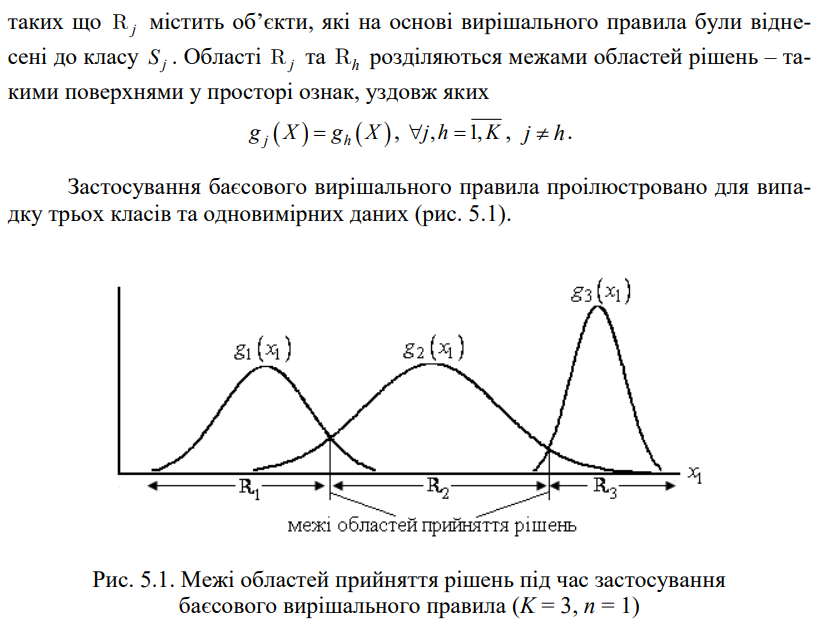


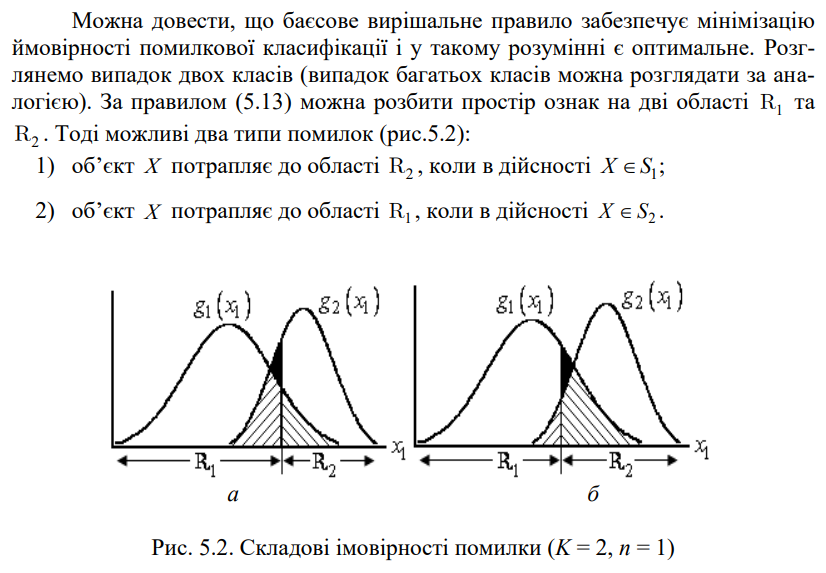


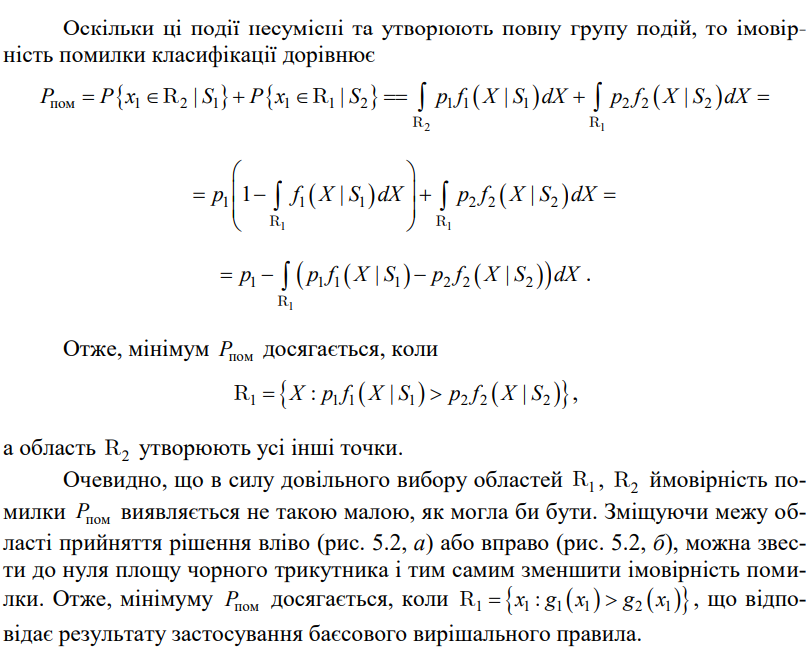


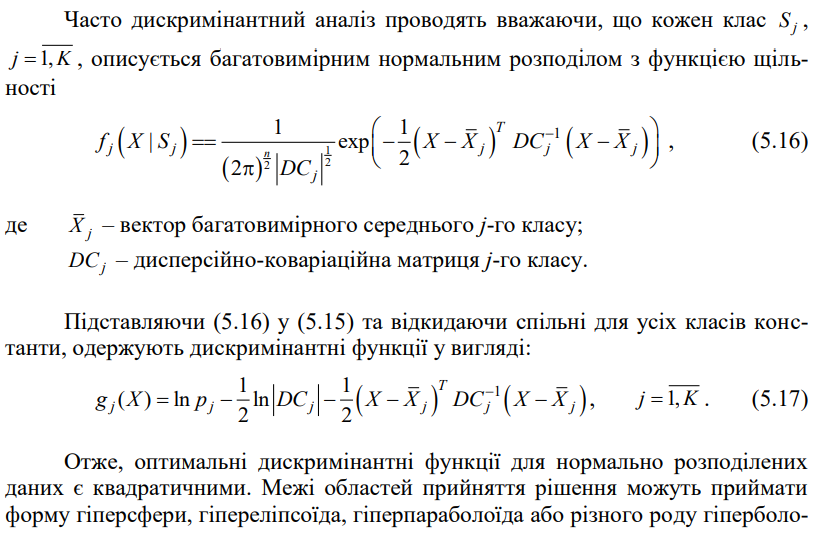


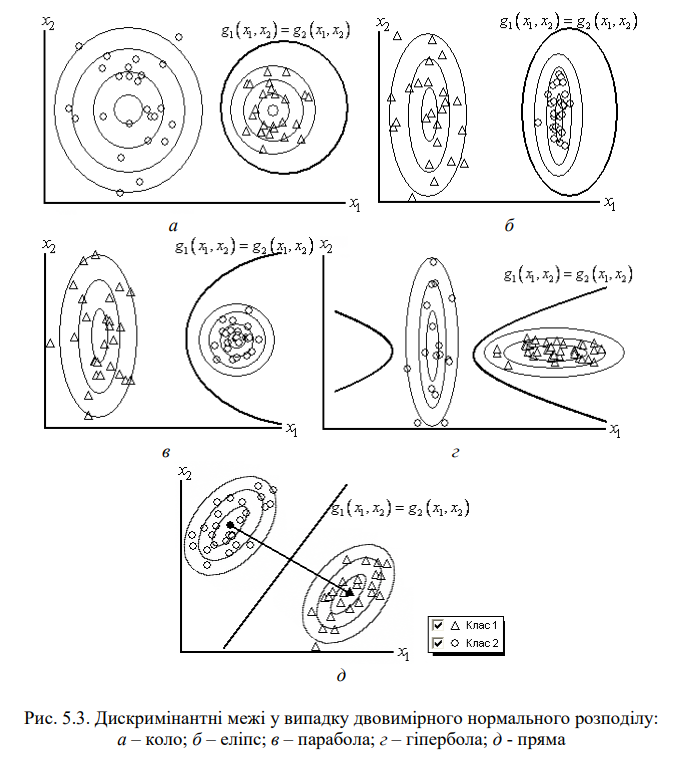


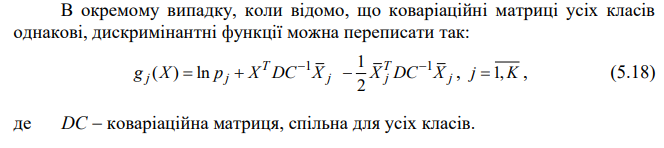


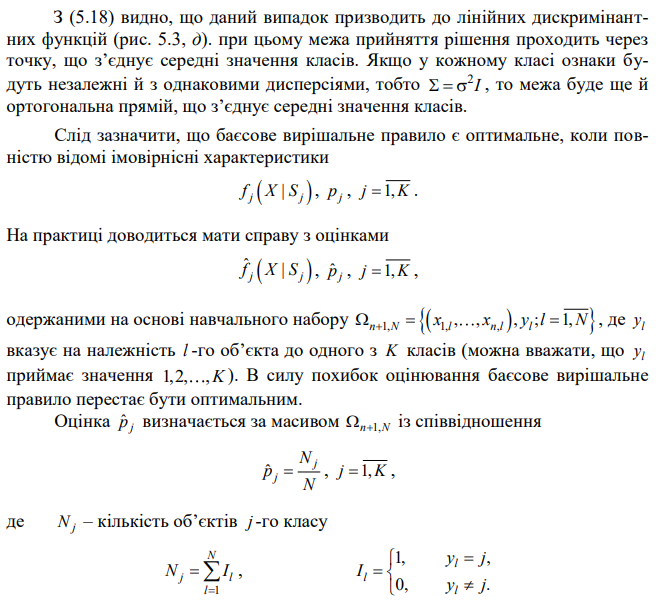


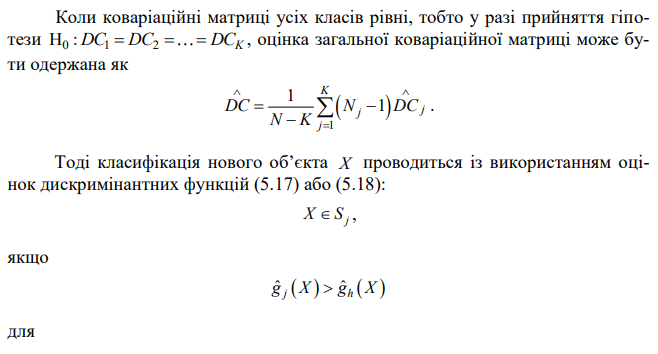


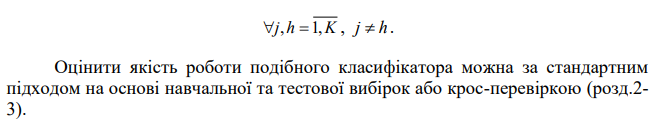












**Результати роботи програми**

При запуску програми відкриється вікно де можна обрати кількість генерування класів, генерувати дані з різними або однаковими ДК матрицями.

На рис.1 та рис.2 показані результати для різних ДК матриць.

На рис.3 та рис.4 показані результати для однакових ДК матриць.

Результатом є графік точок та замальованих областей що відносяться до конкретної множини точок, та матрицю помилок класифікації на тестовому наборі даних.

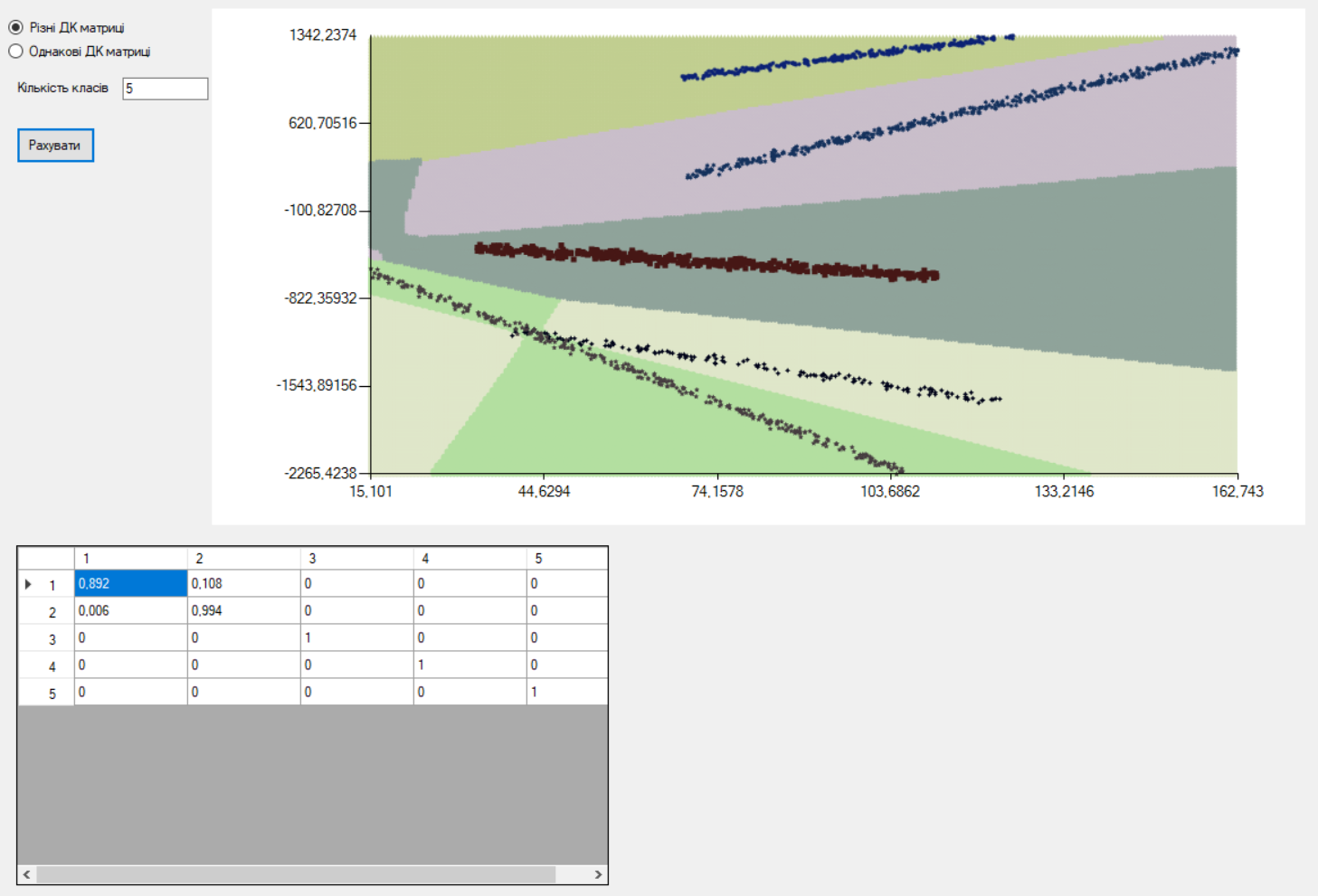


Рис.1. Приклад 1

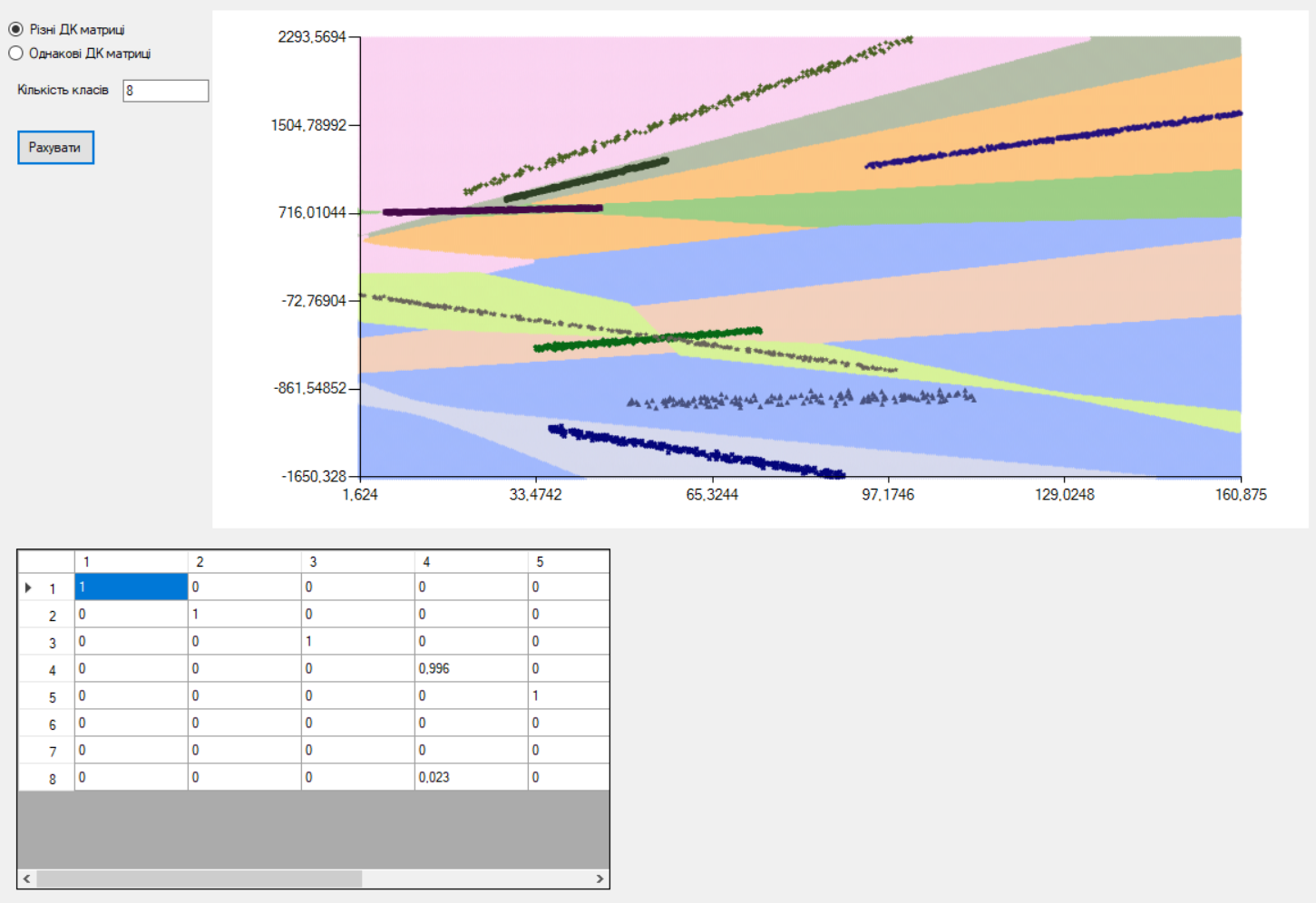


Рис.2. Приклад 2



Рис.3. Приклад 3

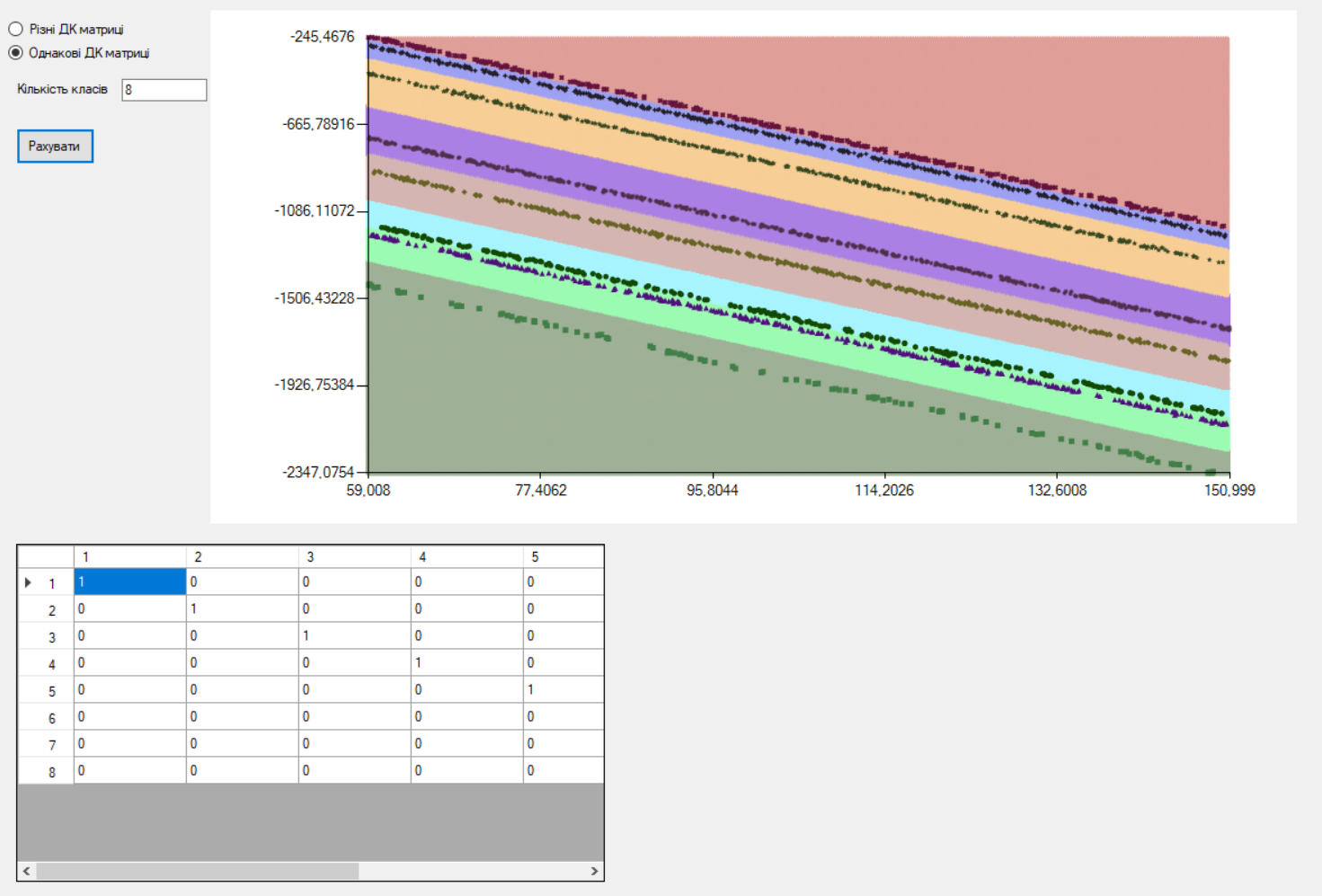
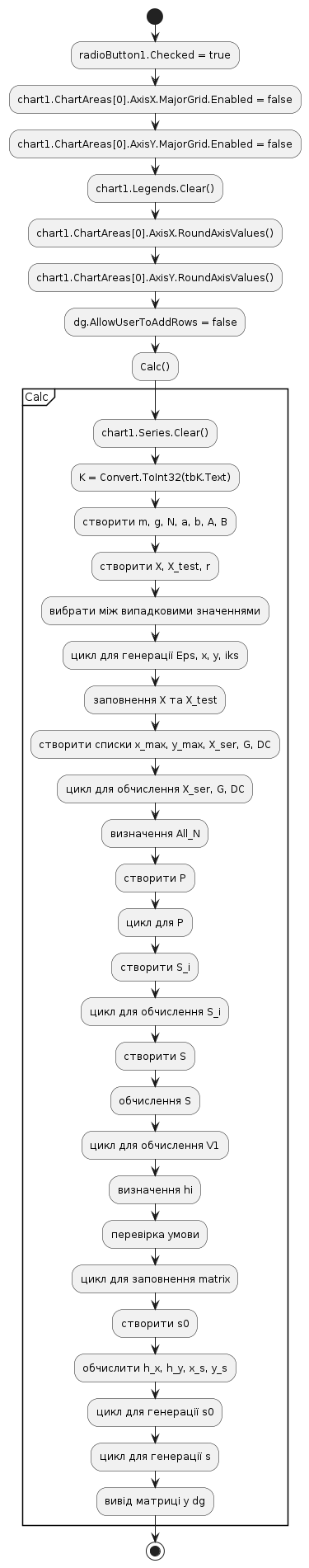


Рис.4. Приклад 4

**Блок-схема**



**Висновок**

* Я реалізував дискримінантний аналіз.

**Список використаної літератури**

* Приставка П.О. Методи штучного інтелекту / Електронний посібник. – К. Національний авіаційний університет, кафедра прикладної математики, 2022 р. – 64 с.

**Код програми**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using System.Xml.Linq;

using MathNet.Numerics.LinearAlgebra;

namespace lab3

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

radioButton1.Checked = true;

chart1.ChartAreas[0].AxisX.MajorGrid.Enabled = false;

chart1.ChartAreas[0].AxisY.MajorGrid.Enabled = false;

chart1.Legends.Clear();

chart1.ChartAreas[0].AxisX.RoundAxisValues();

chart1.ChartAreas[0].AxisY.RoundAxisValues();

dg.AllowUserToAddRows = false;

}

double R\_x\_Y(List<double> v, List<double> v2, double aver1, double aver2, double g1, double g2)

{

double ser\_xy = 0;

double Sum = v.Count();

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

{

ser\_xy = ser\_xy + (v[i] \* v2[i]);

}

ser\_xy = ser\_xy / Sum;

double r\_x\_y = (Sum / (Sum - 1)) \* (ser\_xy - aver1 \* aver2) / (g1 \* g2);

return r\_x\_y;

}

double Kvantil\_u(double a)

{

double kv\_u = 0;

if (a <= 0.0001) kv\_u = 3.69;

else if (a <= 0.001) kv\_u = 3.08;

else if (a <= 0.005) kv\_u = 2.57;

else if (a <= 0.01) kv\_u = 2.33;

else if (a <= 0.025) kv\_u = 1.96;

else if (a <= 0.05) kv\_u = 1.64;

else if (a <= 0.075) kv\_u = 1.44;

else if (a <= 0.1) kv\_u = 1.28;

else if (a <= 0.15) kv\_u = 1.03;

else if (a <= 0.2) kv\_u = 0.84;

else if (a <= 0.25) kv\_u = 0.67;

else if (a <= 0.35) kv\_u = 0.39;

else if (a <= 0.4) kv\_u = 0.25;

else if (a <= 0.45) kv\_u = 0.13;

else if (a <= 0.5) kv\_u = 0;

else if (a <= 0.55) kv\_u = 0.13;

else if (a <= 0.65) kv\_u = 0.39;

else if (a <= 0.7) kv\_u = 0.53;

else if (a <= 0.75) kv\_u = 0.67;

else if (a <= 0.8) kv\_u = 0.84;

else if (a <= 0.85) kv\_u = 1.04;

else if (a <= 0.9) kv\_u = 1.28;

else if (a <= 0.95) kv\_u = 1.65;

else if (a <= 0.97) kv\_u = 1.88;

else if (a <= 0.985) kv\_u = 2.17;

else if (a <= 0.99) kv\_u = 2.33;

else if (a <= 0.995) kv\_u = 2.57;

else kv\_u = 3.69;

return kv\_u;

}

double Kvantil\_hi(double v)

{

double a = 0.1;

a = 1 - a;

double kv\_u = Kvantil\_u(a);

double kv\_hi = v \* Math.Pow(1 - 2 / (9 \* v) + kv\_u \* Math.Sqrt(2 / (9 \* v)), 3);

return kv\_hi;

}

void Calc()

{

chart1.Series.Clear();

int K = Convert.ToInt32(tbK.Text);

List<double> m = new List<double>();

List<double> g = new List<double>();

List<double> N = new List<double>();

List<double> a = new List<double>();

List<double> b = new List<double>();

List<double> A = new List<double>();

List<double> B = new List<double>();

List<Tuple<List<double>, List<double>>> X = new List<Tuple<List<double>, List<double>>>();

List<Tuple<List<double>, List<double>>> X\_test = new List<Tuple<List<double>, List<double>>>();

Random r = new Random();

if (radioButton1.Checked)

{

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

{

m.Add(r.Next(-100, 100));

if (i != 0 && m[i] == m[i - 1]) m[i] = m[i - 1] + 30;

g.Add(r.Next(5, 40));

N.Add(r.Next(100, 800));

a.Add(r.Next(-1000, 1000));

b.Add(r.Next(-20, 20));

if (b[i] == 0) b[i] = 2;

A.Add(r.Next(0, 100));

B.Add(A[i] + r.Next(10, 100));

}

}

else if (radioButton2.Checked)

{

Random r1 = new Random();

double g1 = r1.Next(5, 20);

//double a1 = ;

double b1 = r.Next(-10, 10);

if (b1 == 0) b1 = 2;

double A1 = r1.Next(0, 100);

double B1 = A1 + r1.Next(10, 100);

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

{

m.Add(r.Next(-100, 100));

if (i != 0 && m[i] == m[i - 1]) m[i] = m[i - 1] + 10;

g.Add(g1);

N.Add(r.Next(100, 800));

a.Add(r1.Next(-1000, 1000));

b.Add(b1);

A.Add(A1);

B.Add(B1);

}

}

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

{

List<double> Eps = new List<double>();

List<double> x = new List<double>();

List<double> y = new List<double>();

double a\_lin = a[i];

double b\_lin = b[i];

List<double> iks = new List<double>();

double N2 = N[i] \* 5;

double z2 = 0;

int count2 = 0, count3 = 0;

for (int j = 0; j < N2; j++)

{

if (count2 < 5)

{

z2 += Convert.ToDouble(r.Next(Convert.ToInt32(m[i] - 3 \* g[i]) \* 1000, Convert.ToInt32((m[i] + 3 \* g[i])) \* 1000) / 1000.0);

count2++;

}

else

{

j--;

count2 = 0;

Eps.Add(z2 / 5.0);

z2 = 0;

iks.Add(Convert.ToDouble(r.Next(Convert.ToInt32(A[i]) \* 1000, Convert.ToInt32(B[i]) \* 1000) / 1000.0));

x.Add(iks[count3]);

count3++;

}

}

iks.Add(Convert.ToDouble(r.Next(Convert.ToInt32(A[i]) \* 1000, Convert.ToInt32(B[i]) \* 1000) / 1000.0));

x.Add(iks[iks.Count - 1]);

Eps.Add(z2 / 5.0);

List<double> x1 = new List<double>();

for (int j = 0; j < (N[i] \* 0.7); j++)

{

y.Add(a\_lin + b\_lin \* iks[j] + Eps[j]);

x1.Add(iks[j]);

}

X.Add(new Tuple<List<double>, List<double>>(x1, y));

List<double> x2 = new List<double>();

List<double> y2 = new List<double>();

for (int j = (int)(N[i] \* 0.7); j < N[i]; j++)

{

y2.Add(a\_lin + b\_lin \* iks[j] + Eps[j]);

x2.Add(iks[j]);

}

X\_test.Add(new Tuple<List<double>, List<double>>(x2, y2));

}

List<double> x\_max = new List<double>();

List<double> y\_max = new List<double>();

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

{

x\_max.Add(X[i].Item1.Max());

x\_max.Add(X[i].Item1.Min());

y\_max.Add(X[i].Item2.Max());

y\_max.Add(X[i].Item2.Min());

}

List<Tuple<double,double>> X\_ser = new List<Tuple<double, double>>();

List<Tuple<double, double>> G = new List<Tuple<double, double>>();

List<double[,]> DC = new List<double[,]>();

for (int i = 0; i < X.Count(); i++)

{

double aver1 = 0;

for(int j = 0; j < X[i].Item1.Count; j++)

{

aver1 += X[i].Item1[j];

}

aver1 /= X[i].Item1.Count;

double aver2 = 0;

for (int j = 0; j < X[i].Item2.Count; j++)

{

aver2 += X[i].Item2[j];

}

aver2 /= X[i].Item2.Count;

X\_ser.Add(new Tuple<double, double>(aver1, aver2));

}

for (int i = 0; i < X.Count(); i++)

{

double g1 = 0;

for (int j = 0; j < X[i].Item1.Count; j++)

{

g1 += Math.Pow(X[i].Item1[j] - X\_ser[i].Item1,2);

}

g1 /= (X[i].Item1.Count - 1);

double g2 = 0;

for (int j = 0; j < X[i].Item2.Count; j++)

{

g2 += Math.Pow(X[i].Item2[j] - X\_ser[i].Item2, 2);

}

g2 /= (X[i].Item1.Count - 1);

G.Add(new Tuple<double, double>(Math.Sqrt(g1), Math.Sqrt(g2)));

}

for (int i = 0; i < X.Count(); i++)

{

double[,] temp1 = new double[2,2];

for(int j = 0; j < 2; j++)

{

for(int k = 0; k < 2; k++)

{

if (k == j && k == 0) temp1[j, k] = Math.Pow(G[i].Item1, 2);

else if (k == j && k == 1) temp1[j, k] = (Math.Pow(G[i].Item2, 2));

else temp1[j, k] = (G[i].Item2 \* G[i].Item1 \* R\_x\_Y(X[i].Item1, X[i].Item2, X\_ser[i].Item1, X\_ser[i].Item2, G[i].Item1, G[i].Item2));

}

}

DC.Add(temp1);

}

double All\_N = 0;

for(int i = 0; i < X.Count(); i++)

{

All\_N += X[i].Item1.Count();

}

List<double> P = new List<double>();

for (int i = 0; i < X.Count(); i++)

{

P.Add(X[i].Item1.Count() / All\_N);

}

List<Matrix<double>> S\_i = new List<Matrix<double>>();

double V1 = 0;

for (int i = 0; i < DC.Count; i++)

{

Matrix<double> S\_d = Matrix<double>.Build.DenseOfArray(DC[i]);

S\_i.Add((X[i].Item2.Count - 1) \* S\_d);

}

Matrix<double> S = S\_i[0];

for (int i = 1; i < S\_i.Count; i++)

{

S += S\_i[i];

}

S = S / (All\_N - K);

for (int i = 0; i < DC.Count; i++)

{

Matrix<double> S\_d = Matrix<double>.Build.DenseOfArray(DC[i]);

double res1 = S.Determinant();

double res2 = S\_d.Determinant();

V1 += (X[i].Item2.Count - 1)/2.0 \* Math.Log(res1/res2);

}

double hi = Kvantil\_hi(6 \* (K - 1) / 2.0);

if(hi >= V1)

{

for(int i = 0; i < DC.Count; i++)

{

DC[i] = S.ToArray();

}

}

double[,] matrix = new double[K, K];

for(int i = 0; i < X.Count(); i++)

{

for (int j = 0; j < X[i].Item1.Count; j++)

{

double[,] v = new double [2,1];

v[0,0] = X[i].Item1[j];

v[1,0] = X[i].Item2[j];

Matrix<double> V = Matrix<double>.Build.DenseOfArray(v);

List<double> g\_x = new List<double>();

for(int k = 0; k < X\_ser.Count; k++)

{

double[,] v2 = new double[2,1];

v2[0,0] = X\_ser[k].Item1;

v2[1,0] = X\_ser[k].Item2;

Matrix<double> V\_ser = Matrix<double>.Build.DenseOfArray(v2);

Matrix<double> dc = Matrix<double>.Build.DenseOfArray(DC[k]);

Matrix<double> res = Math.Log(P[k]) - 0.5 \* Math.Log(dc.Determinant()) - 0.5 \* (V - V\_ser).Transpose() \* dc.Inverse() \* (V - V\_ser);

g\_x.Add(res[0, 0]);

}

int index = g\_x.IndexOf(g\_x.Max());

matrix[i, index] += 1.0 / X[i].Item1.Count;

}

}

List<Series> s0 = new List<Series>();

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

{

Series s = new Series();

s.ChartType = SeriesChartType.Point;

s.Color = GenerateLightColor();

s0.Add(s);

}

double N1 = 400.0;

double h\_x = (x\_max.Max() - x\_max.Min()) / N1;

double h\_y = (y\_max.Max() - y\_max.Min()) / N1;

double x\_s;

double y\_s = y\_max.Min();

chart1.ChartAreas[0].AxisX.Minimum = x\_max.Min();

chart1.ChartAreas[0].AxisY.Minimum = y\_max.Min();

chart1.ChartAreas[0].AxisX.Maximum = x\_max.Max();

chart1.ChartAreas[0].AxisY.Maximum = y\_max.Max();

for (int i = 0; i < N1; i++)

{

x\_s = x\_max.Min();

for (int j = 0; j < N1; j++)

{

double[,] v = new double[2, 1];

v[0, 0] = x\_s;

v[1, 0] = y\_s;

Matrix<double> V = Matrix<double>.Build.DenseOfArray(v);

List<double> g\_x = new List<double>();

for (int k = 0; k < X\_ser.Count; k++)

{

double[,] v2 = new double[2, 1];

v2[0, 0] = X\_ser[k].Item1;

v2[1, 0] = X\_ser[k].Item2;

Matrix<double> V\_ser = Matrix<double>.Build.DenseOfArray(v2);

Matrix<double> dc = Matrix<double>.Build.DenseOfArray(DC[k]);

Matrix<double> res = Math.Log(P[k]) - 0.5 \* Math.Log(dc.Determinant()) - 0.5 \* (V - V\_ser).Transpose() \* dc.Inverse() \* (V - V\_ser);

g\_x.Add(res[0, 0]);

}

int index = g\_x.IndexOf(g\_x.Max());

s0[index].Points.AddXY(x\_s, y\_s);

x\_s += h\_x;

}

y\_s += h\_y;

}

for(int i = 0; i < s0.Count; i++)

{

chart1.Series.Add(s0[i]);

}

for (int i = 0; i < X.Count(); i++)

{

Series s = new Series();

s.ChartType = SeriesChartType.Point;

s.Color = GenerateDarkColor();

for (int j = 0; j < X[i].Item1.Count; j++)

{

s.Points.AddXY(X[i].Item1[j], X[i].Item2[j]);

}

chart1.Series.Add(s);

}

Draw\_martix(matrix, K);

}

HashSet<Color> usedColors = new HashSet<Color>();

Random rand = new Random();

Color GenerateLightColor()

{

Color color;

do

{

int r = rand.Next(128, 256);

int g = rand.Next(128, 256);

int b = rand.Next(128, 256);

color = Color.FromArgb(r, g, b);

} while (usedColors.Contains(color));

usedColors.Add(color);

return color;

}

Color GenerateDarkColor()

{

Color color;

do

{

int r = rand.Next(0, 128);

int g = rand.Next(0, 128);

int b = rand.Next(0, 128);

color = Color.FromArgb(r, g, b);

} while (usedColors.Contains(color));

usedColors.Add(color);

return color;

}

void Draw\_martix(double[,] matrix, int n)

{

dg.Rows.Clear();

dg.Columns.Clear();

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

{

dg.Columns.Add((i + 1).ToString(), (i + 1).ToString());

}

dg.Rows.Add(n);

for (int i = 0; i < dg.Rows.Count; i++)

{

dg.Rows[i].HeaderCell.Value = (i + 1).ToString();

}

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

{

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

{

dg.Rows[i].Cells[j].Value = Math.Round(matrix[i,j],3).ToString();

}

}

}

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

{

Calc();

}

}

}