МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ



Кафедра СМАРТ технологий

Лабораторная работа № 1

“На Использование графических возможностей C# для визуализации данных стохастических процессов”

По дисциплине «Технологии визуализации данных систем управления»

Группа: 221-327

Студент: Шурова Д. С.

Дата: 14.04.2025

Преподаватель: Идиатуллов Т. Т.

Москва, 2025

**Задача**

Разработать приложение по генерации стохастических данных с заданным профилем распределения и визуализации распределения случайных величин.

Задача 1.

Подготовить приложение на языке C# для статистической обработки и визуализации собранных наборов данных Реализовать генерацию заданного (через текстовое поле) количества случайных точек (X1, X2), где X1 и X2 – равномерно распределенные случайные величины на диапазоне [0 ÷ 1]. Подготовить функционал для настройки профиля преобразования (пересчета) двух случайных величин (Y1, Y2) из равномерно распределенных случайных величин (X1, X2). Реализовать отображение профиля преобразования как кусочно-линейных функций (по пяти точкам – первая и последняя привязаны к границам диапазона генерации равномерно-распределенных случайных величин). Реализовать возможность сохранения и чтения данных и настроек функций преобразования. Реализовать отрисовку наборов данных в виде облака точек, с возможностью выбора пар параметров, используемых как координаты точек. Должна существовать возможность отобразить точки (X1, X2), (X1, Y1), (X2, Y2), (Y1, Y2).

Задача 2.

Реализовать функции анализа данных. Реализовать расчёт плотности распределения случайных точек и выполнить фоновую окраску области отрисовки случайных точек. При подсчете плотности разделить диапазон отображения по каждой оси на 10 интервалов. Добавить в приложение расчёт статистических данных (описательной статистики) полученного распределения (среднее, средне-квадратическое отклонение, мода, медиана), а также отображения гистограммы распределения каждой из случайных величин. Расположить оси гистограмм вдоль соответствующих осей на диаграмме облака точек.

**Ход работы**

Реализована генерацию заданного (через текстовое поле) количества случайных точек (X1, X2), равномерно распределенных на диапазоне [0 ÷ 1].

Добавлены dataGrid для настройки профиля преобразования (интерполяции) двух случайных величин (Y1, Y2) из равномерно распределенных случайных величин (X1, X2) и реализовано отображение профиля отображения как кусочно-линейной функции в методе UpdateChart.

Реализована возможность сохранения и чтения данных и настроек функций преобразования в формат csv в методах buttonSave\_Click и buttonLoad\_Click.

Реализована отрисовка наборов данных в виде облака точек, с возможностью выбора пар параметров в ComboBox, используемых как координаты точек. Осуществлена возможность отобразить точки (X1, X2), (X1, Y1), (X2, Y2), (Y1, Y2) на рис. 1 (а, б, в и г соответственно)

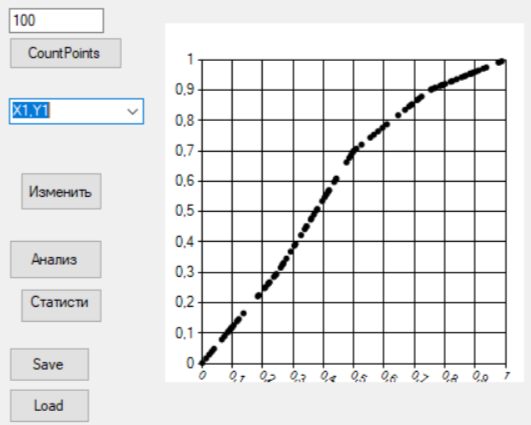
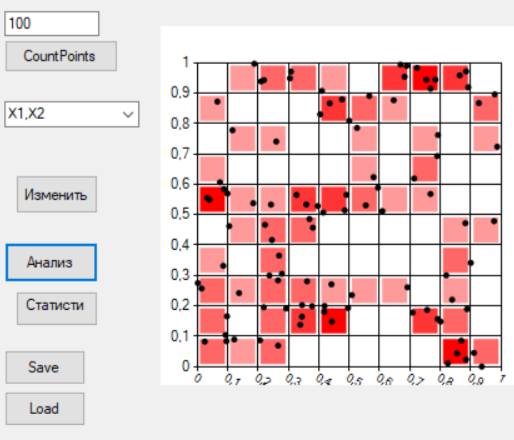


Рисунок 1 – а, б

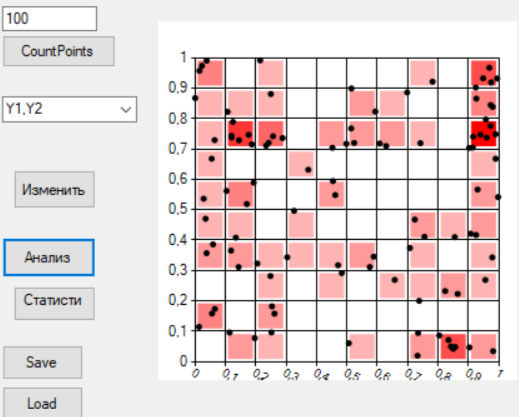
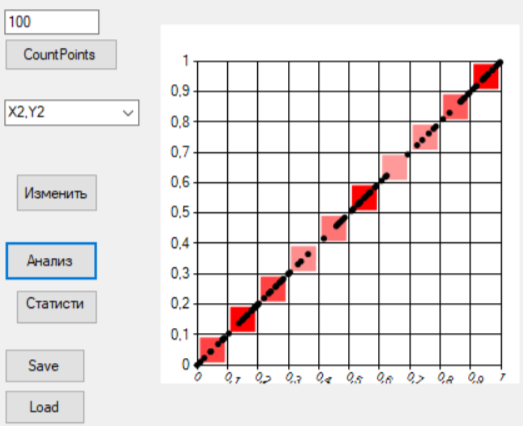


Рисунок 1 – в, г

Реализован расчёт плотности распределения случайных точек и выполнить фоновую окраску области отрисовки случайных точек в методе DrawDensityBackground(). На диаграмме берется ячейка 0,1 на 0,1. В список с количеством точек для каждой ячейки идет подсчет, находится максимальное количество точек. Далее вычисляется интенсивность цвета (intensity), как отношение количества точек в ячейке к максимальному количеству (count/maxCount).

* Цвет выбирается как полупрозрачный красный (Color.Red), где прозрачность и насыщенность зависят от intensity:
  + Минимальная прозрачность: 50 (если intensity = 0).
  + Максимальная насыщенность: 50 + 205 = 255 (если intensity = 1).
* Создается точка данных (DataPoint) с координатами центра ячейки и заданным цветом.

В приложение добавлен расчёт статистических данных (описательной статистики) полученного распределения (среднее, средне-квадратическое отклонение, мода, медиана) в методе btnStats\_Click(), а также отображения гистограммы распределения каждой из случайных величин, что вычисляется и выводится в методе AddMarginalHistograms(). Для построения гистограмм рассматривается в какой бин попадает каждая точка (x,y), увеличивая его. (Рис. 2)

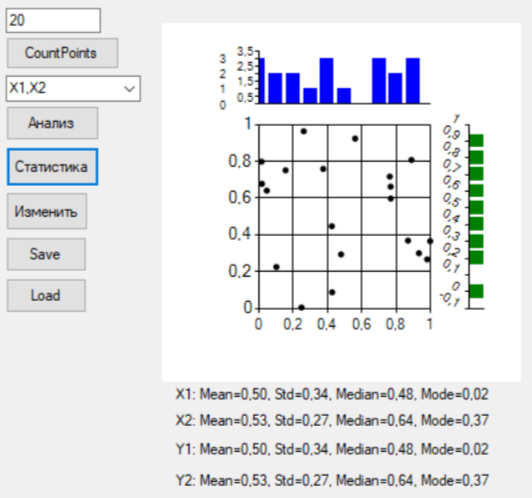


Рисунок 2 – Статистические данные

**Вывод**

Графические возможности C# подходят для визуализации данных стохастических процессов.

**Приложение А**

*Листинг 1 – Код программы*

using System;

using System.Collections.Generic;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Text.Json;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

namespace Lab1\_VD

{

public partial class Form1 : Form

{

List<double> X1 = new List<double>();

List<double> X2 = new List<double>();

List<double> Y1 = new List<double>();

List<double> Y2 = new List<double>();

List<PointF> transformationProfile1 = new List<PointF>()

{

new PointF(0f, 0f),

new PointF(0.25f, 0.25f),

new PointF(0.5f, 0.5f),

new PointF(0.75f, 0.75f),

new PointF(1f, 1f)

};

List<PointF> transformationProfile2 = new List<PointF>();

int count;

Boolean dence = false;

Boolean stats = false;

public Form1()

{

InitializeComponent();

comboBoxView.Items.AddRange(new string[] { "X1,X2", "X1,Y1", "X2,Y2", "Y1,Y2" });

comboBoxView.SelectedIndex = 0;

SetupProfileGrid();

UpdateChart();

}

private void SetupProfileGrid()

{

dataGridViewProfile1.Columns.Clear();

dataGridViewProfile2.Columns.Clear();

dataGridViewProfile1.Columns.Add("X1", "X1");

dataGridViewProfile1.Columns.Add("Y1", "Y1");

dataGridViewProfile2.Columns.Add("X2", "X2");

dataGridViewProfile2.Columns.Add("Y2", "Y2");

dataGridViewProfile1.RowCount = 5;

dataGridViewProfile2.RowCount = 5;

for (int i = 0; i < 5; i++)

{

dataGridViewProfile1.Rows[i].Cells[0].Value = transformationProfile1[i].X;

dataGridViewProfile1.Rows[i].Cells[1].Value = transformationProfile1[i].Y;

dataGridViewProfile2.Rows[i].Cells[0].Value = transformationProfile1[i].X;

dataGridViewProfile2.Rows[i].Cells[1].Value = transformationProfile1[i].Y;

}

}

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

{

if (!int.TryParse(textBoxCount.Text, out count) || count <= 0)

{

MessageBox.Show("Введите корректное количество точек.");

return;

}

Random rnd = new Random();

X1.Clear();

X2.Clear();

Y1.Clear();

Y2.Clear();

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

{

double x1 = rnd.NextDouble();

double x2 = rnd.NextDouble();

X1.Add(x1);

X2.Add(x2);

Y1.Add(Transform1(x1));

Y2.Add(Transform2(x2));

}

UpdateChart();

}

private double Transform1(double x)

{

var profile = GetProfile1();

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

{

if (x >= profile[i].X && x <= profile[i + 1].X)

{

double x0 = profile[i].X;

double x1 = profile[i + 1].X;

double y0 = profile[i].Y;

double y1 = profile[i + 1].Y;

return y0 + (x - x0) / (x1 - x0) \* (y1 - y0);

}

}

return x; // fallback

}

private double Transform2(double x)

{

var profile = GetProfile2();

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

{

if (x >= profile[i].X && x <= profile[i + 1].X)

{

double x0 = profile[i].X;

double x1 = profile[i + 1].X;

double y0 = profile[i].Y;

double y1 = profile[i + 1].Y;

return y0 + (x - x0) / (x1 - x0) \* (y1 - y0);

}

}

return x; // fallback

}

private List<PointF> GetProfile1()

{

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

for (int i = 0; i < 5; i++)

{

float x = Convert.ToSingle(dataGridViewProfile1.Rows[i].Cells[0].Value);

float y = Convert.ToSingle(dataGridViewProfile1.Rows[i].Cells[1].Value);

profile.Add(new PointF(x, y));

}

profile = profile.OrderBy(p => p.X).ToList();

return profile;

}

private List<PointF> GetProfile2()

{

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

for (int i = 0; i < 5; i++)

{

float x = Convert.ToSingle(dataGridViewProfile2.Rows[i].Cells[0].Value);

float y = Convert.ToSingle(dataGridViewProfile2.Rows[i].Cells[1].Value);

profile.Add(new PointF(x, y));

}

profile = profile.OrderBy(p => p.X).ToList();

return profile;

}

private void UpdateChart()

{

chartData.Series.Clear();

chartData.ChartAreas.Clear();

// Создаём заново основную область графика

ChartArea mainArea = new ChartArea("MainArea");

mainArea.AxisX.Minimum = 0;

mainArea.AxisX.Maximum = 1;

mainArea.AxisY.Minimum = 0;

mainArea.AxisY.Maximum = 1;

mainArea.AxisX.Interval = 0.1;

mainArea.AxisY.Interval = 0.1;

mainArea.Position = new ElementPosition(0, 0, 100, 100);

mainArea.InnerPlotPosition = new ElementPosition(10, 10, 85, 85); // красивое размещение

chartData.ChartAreas.Add(mainArea);

List<(double x, double y)> points = new List<(double, double)>();

string selected = comboBoxView.SelectedItem.ToString();

for (int i = 0; i < X1.Count; i++)

{

if (selected == "X1,X2")

points.Add((X1[i], X2[i]));

else if (selected == "X1,Y1")

points.Add((X1[i], Y1[i]));

else if (selected == "X2,Y2")

points.Add((X2[i], Y2[i]));

else if (selected == "Y1,Y2")

points.Add((Y1[i], Y2[i]));

}

if (dence) { DrawDensityBackground(points); } // <--- добавили

if (stats)

{

AddMarginalHistograms(points);

}

Series series = new Series("Points")

{

ChartType = SeriesChartType.Point,

MarkerStyle = MarkerStyle.Circle,

MarkerSize = 5,

Color = Color.Black // ← вот здесь задаётся цвет!

};

foreach (var (x, y) in points)

{

series.Points.AddXY(x, y);

}

chartData.Series.Add(series);

}

private void comboBoxView\_SelectedIndexChanged(object sender, EventArgs e)

{

UpdateChart();

}

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

{

try

{

// Check if data is available

if (count == 0 || X1 == null || X2 == null || Y1 == null || Y2 == null)

{

MessageBox.Show("Сначала сгенерируйте точки", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

using (SaveFileDialog saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "CSV files (\*.csv)|\*.csv";

saveFileDialog.Title = "Сохранить точки как CSV";

if (saveFileDialog.ShowDialog() == DialogResult.OK)

{

using (StreamWriter writer = new StreamWriter(saveFileDialog.FileName))

{

// Write first 5 rows from grid views

for (int i = 0; i < 5; i++)

{

writer.WriteLine($"{dataGridViewProfile1.Rows[i].Cells[0].Value};" +

$"{dataGridViewProfile1.Rows[i].Cells[1].Value};" +

$"{dataGridViewProfile2.Rows[i].Cells[0].Value};" +

$"{dataGridViewProfile2.Rows[i].Cells[1].Value}");

}

// Write generated points

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

{

writer.WriteLine($"{X1[i]};{X2[i]};{Y1[i]};{Y2[i]}");

}

}

MessageBox.Show("Файл успешно сохранён", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении файла: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void loadButton\_Click(object sender, EventArgs e)//загрузка файла

{

List<double[]> data = new List<double[]>();

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Text files (\*.csv)|\*.csv";

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

try

{

using (StreamReader reader = new StreamReader(openFileDialog.FileName))

{

string line;

while ((line = reader.ReadLine()) != null)

{

// Разделяем строку по ';' (можно заменить на ',')

string[] values = line.Split(';');

// Конвертируем каждое значение в double

double[] row = new double[values.Length];

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

{

if (double.TryParse(values[i], out double num))

row[i] = num;

else

row[i] = 0.0; // или можно выбросить ошибку

}

data.Add(row);

}

}

dataGridViewProfile1.Columns.Clear();

dataGridViewProfile2.Columns.Clear();

dataGridViewProfile1.Columns.Add("X1", "X1");

dataGridViewProfile1.Columns.Add("Y1", "Y1");

dataGridViewProfile2.Columns.Add("X2", "X2");

dataGridViewProfile2.Columns.Add("Y2", "Y2");

dataGridViewProfile1.RowCount = 5;

dataGridViewProfile2.RowCount = 5;

X1.Clear();

X2.Clear();

Y1.Clear();

Y2.Clear();

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

{

if (i<5)

{

dataGridViewProfile1.Rows[i].Cells[0].Value = data[i][0].ToString();

dataGridViewProfile1.Rows[i].Cells[1].Value = data[i][1].ToString();

dataGridViewProfile2.Rows[i].Cells[0].Value = data[i][2].ToString();

dataGridViewProfile2.Rows[i].Cells[1].Value = data[i][3].ToString();

}

if (i > 4)

{

X1.Add(data[i][0]);

X2.Add(data[i][1]);

Y1.Add(data[i][2]);

Y2.Add(data[i][3]);

}

}

UpdateChart();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке карты: " + ex.Message);

}

}

}

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

{

Y1.Clear();

Y2.Clear();

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

{

Y1.Add(Transform1(X1[i]));

Y2.Add(Transform2(X2[i]));

}

UpdateChart();

}

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

{

dence = true;

UpdateChart();

dence = false;

}

private void DrawDensityBackground(List<(double x, double y)> points)

{

int gridSize = 10; // 0.1 x 0.1 ячейки

double cellSize = 0.1;

int[,] counts = new int[gridSize, gridSize];

int maxCount = 0;

// Подсчёт точек в ячейках

foreach (var (x, y) in points)

{

if (x < 0 || x >= 1 || y < 0 || y >= 1) continue;

int i = (int)(x / cellSize);

int j = (int)(y / cellSize);

counts[i, j]++;

maxCount = Math.Max(maxCount, counts[i, j]);

}

// Удаляем старую серию, если была

if (chartData.Series.IndexOf("Density") >= 0)

chartData.Series.Remove(chartData.Series["Density"]);

// Создаем новую серию для прямоугольников

Series densitySeries = new Series("Density")

{

ChartType = SeriesChartType.Point,

MarkerStyle = MarkerStyle.Square,

MarkerSize = 20,

};

chartData.Series.Add(densitySeries);

// Добавляем квадратики

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

{

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

{

int count = counts[i, j];

if (count == 0) continue;

double centerX = (i + 0.5) \* cellSize;

double centerY = (j + 0.5) \* cellSize;

double intensity = (double)count / maxCount;

Color color = Color.FromArgb(50 + (int)(205 \* intensity), Color.Red);

DataPoint dp = new DataPoint(centerX, centerY)

{

Color = color

};

densitySeries.Points.Add(dp);

}

}

}

private (double mean, double std, double median, double mode) CalculateStatistics(List<double> data)

{

var sorted = data.OrderBy(x => x).ToList();

double mean = data.Average();

double std = Math.Sqrt(data.Sum(x => (x - mean) \* (x - mean)) / data.Count);

double median = sorted[data.Count / 2];

double mode = data.GroupBy(x => Math.Round(x, 2))

.OrderByDescending(g => g.Count())

.First().Key;

return (mean, std, median, mode);

}

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

{

stats = true;

UpdateChart();

var statsX1 = CalculateStatistics(X1);

var statsX2 = CalculateStatistics(X2);

var statsY1 = CalculateStatistics(Y1);

var statsY2 = CalculateStatistics(Y2);

label1.Text = $"X1: Mean={statsX1.mean:F2}, Std={statsX1.std:F2}, Median={statsX1.median:F2}, Mode={statsX1.mode:F2}\n";

label2.Text = $"X2: Mean={statsX2.mean:F2}, Std={statsX2.std:F2}, Median={statsX2.median:F2}, Mode={statsX2.mode:F2}\n";

label3.Text = $"Y1: Mean={statsY1.mean:F2}, Std={statsY1.std:F2}, Median={statsY1.median:F2}, Mode={statsY1.mode:F2}\n";

label4.Text = $"Y2: Mean={statsY2.mean:F2}, Std={statsY2.std:F2}, Median={statsY2.median:F2}, Mode={statsY2.mode:F2}";

stats = false;

}

private void AddMarginalHistograms(List<(double x, double y)> points)

{

var chart = chartData;

chart.Series.Clear();

chart.ChartAreas.Clear();

// --- Основной график (облако точек) ---

ChartArea mainArea = new ChartArea("Main");

mainArea.Position = new ElementPosition(10, 25, 65, 65); // Центральное положение

mainArea.AxisX.Minimum = 0;

mainArea.AxisX.Maximum = 1;

mainArea.AxisY.Minimum = 0;

mainArea.AxisY.Maximum = 1;

chart.ChartAreas.Add(mainArea);

Series scatterSeries = new Series("Scatter")

{

ChartType = SeriesChartType.Point,

ChartArea = "Main",

MarkerStyle = MarkerStyle.Circle,

MarkerSize = 6

};

foreach (var (x, y) in points)

scatterSeries.Points.AddXY(x, y);

chart.Series.Add(scatterSeries);

// --- Верхняя гистограмма (по X, над основной осью X) ---

ChartArea topArea = new ChartArea("TopHistogram");

topArea.Position = new ElementPosition(10, 5, 65, 20); // Над основным графиком, выровнена по X

topArea.AlignWithChartArea = "Main";

topArea.AlignmentOrientation = AreaAlignmentOrientations.Vertical;

topArea.AxisX.Minimum = 0;

topArea.AxisX.Maximum = 1;

topArea.AxisY.MajorGrid.Enabled = false;

topArea.AxisX.MajorGrid.Enabled = false;

topArea.AxisX.LabelStyle.Enabled = false;

topArea.AxisX.MajorTickMark.Enabled = false;

chart.ChartAreas.Add(topArea);

// --- Правая гистограмма (по Y, справа от основной оси Y) ---

ChartArea rightArea = new ChartArea("RightHistogram");

rightArea.Position = new ElementPosition(75, 25, 15, 65); // Справа от основного графика, выровнена по Y

rightArea.AlignWithChartArea = "Main";

rightArea.AlignmentOrientation = AreaAlignmentOrientations.Horizontal;

rightArea.AxisY.Minimum = 0;

rightArea.AxisY.Maximum = 1;

rightArea.AxisX.MajorGrid.Enabled = false;

rightArea.AxisY.MajorGrid.Enabled = false;

rightArea.AxisY.LabelStyle.Enabled = false;

rightArea.AxisY.MajorTickMark.Enabled = false;

chart.ChartAreas.Add(rightArea);

// --- Гистограмма по X ---

Series histX = new Series("HistX")

{

ChartType = SeriesChartType.Column,

ChartArea = "TopHistogram",

Color = Color.Blue

};

// --- Гистограмма по Y ---

Series histY = new Series("HistY")

{

ChartType = SeriesChartType.Bar,

ChartArea = "RightHistogram",

Color = Color.Green

};

int bins = 20;

double[] histXData = new double[bins];

double[] histYData = new double[bins];

foreach (var (x, y) in points)

{

int binX = Math.Min((int)(x \* bins), bins - 1);

int binY = Math.Min((int)(y \* bins), bins - 1);

histXData[binX]++;

histYData[binY]++;

}

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

{

double binStart = (double)i / bins;

histX.Points.AddXY(binStart, histXData[i]);

histY.Points.AddXY(histYData[i], binStart);

}

chart.Series.Add(histX);

chart.Series.Add(histY);

}

}

}