Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение вычислительной техники

и автоматизированных систем»

**Отчёт**

по лабораторной работе № 7

по дисциплине ***«РАЗРАБОТКА ПРИЛОЖЕНИЙ В ВИЗУАЛЬНЫХ СРЕДАХ»***

тема: **«СОЗДАНИЕ ПРИЛОЖЕНИЯ, РЕАЛИЗУЮЩЕГО РЕШЕНИЕ**

**ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА»**

|  |  |  |
| --- | --- | --- |
| Исполнитель: |  | студент группы 10701217  Голованов Павел Андреевич |
| Преподаватель: |  | Гурский Николай Николаевич |

2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 7**

**СОЗДАНИЕ ПРИЛОЖЕНИЯ, РЕАЛИЗУЮЩЕГО РЕШЕНИЕ**

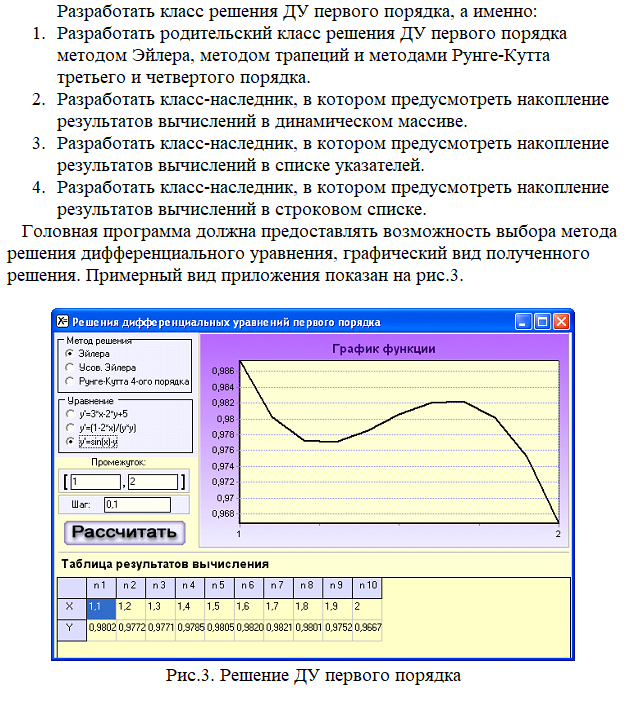
**ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ ПЕРВОГО ПОРЯДКА**

Цель работы:

Изучить принципы создания класса решения

дифференциальных уравнений (ДУ) первого порядка.

Задание:



Результаты выполнения задания:

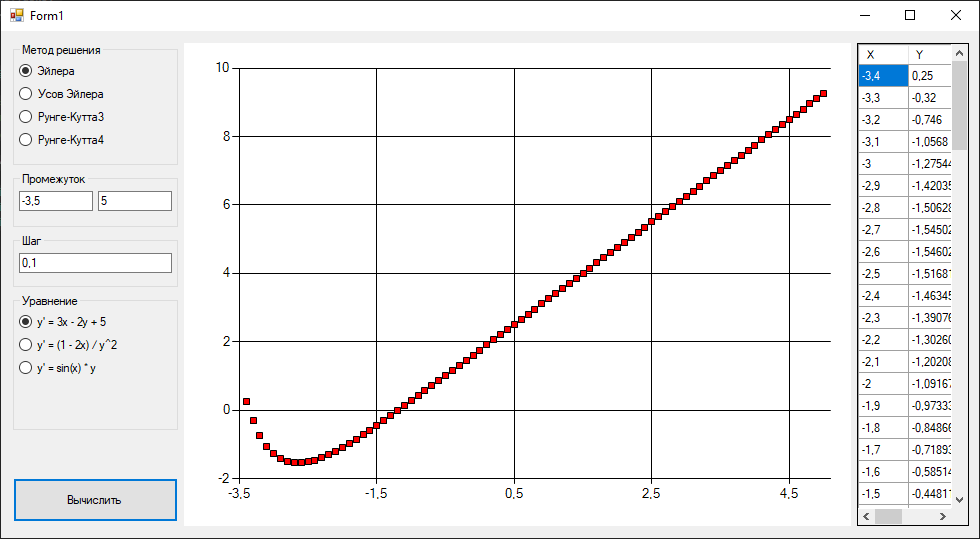


Рисунок 1 – Результат с методом эйлера.

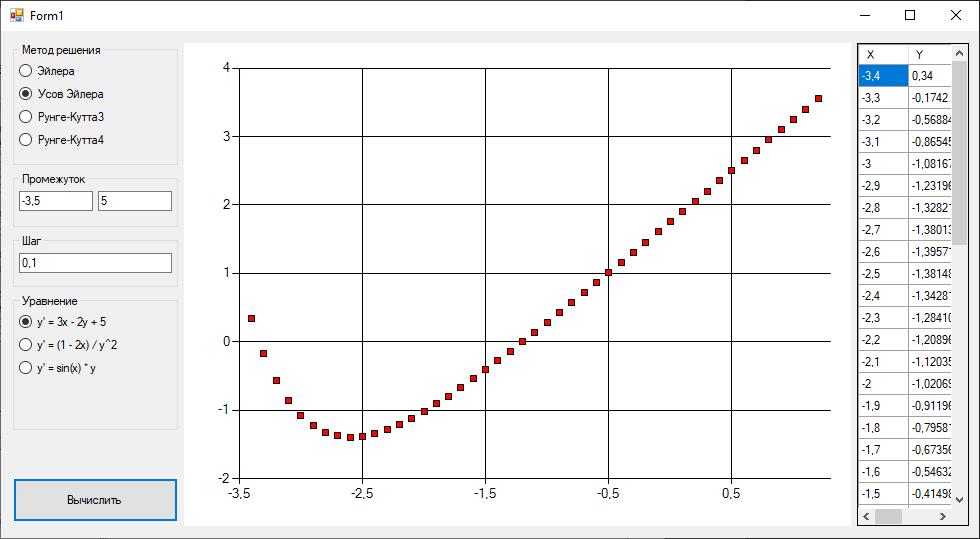


Рисунок 2 – Результат с методом усов эйлера.

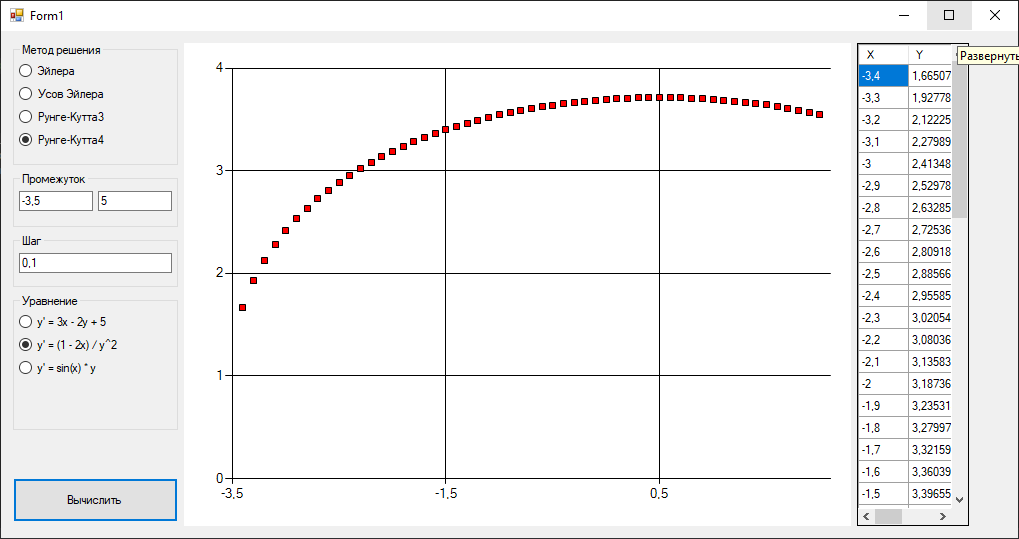


Рисунок 3 – Результат с методом усов Рунге Кутта.

*ПРИЛОЖЕНИЕ A*

Листинг исходных кодов программ

**Файл Form1.cs**

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;

namespace lab\_7

{

public partial class Form1 : Form

{

public static Calculator.DFunc F1 = (x, y) => 3 \* x - 2 \* y + 5;

public static Calculator.DFunc F2 = (x, y) => (1 - 2 \* x) / (y \* y);

public static Calculator.DFunc F3 = (x, y) => Math.Sin(x) \* y;

public static Calculator.DFunc F4 = (x, y) => Math.Sin(x) \* y;

Calculator.CalcMod currentMod = Calculator.CalcMod.EULER;

Calculator.DFunc currentFunc = F1;

Babies.ArrayListCalc arrayCalc;

Babies.ArrayListCalc arrayCalc2;

Babies.LinkListCalc listCalc1;

Babies.LinkListCalc listCalc2;

Babies.StrListCalc strCalc;

public Form1()

{

InitializeComponent();

double step, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

step = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

arrayCalc = new Babies.ArrayListCalc(ref chart1, "func", F1, startX, endX, 1, step);

arrayCalc2 = new Babies.ArrayListCalc(ref chart1, "func", F1, startX, endX, 1, step);

listCalc1 = new Babies.LinkListCalc(ref chart1, "func", F2, startX, endX, 1, step);

listCalc2 = new Babies.LinkListCalc(ref chart1, "func", F3, startX, endX, 1, step);

strCalc = new Babies.StrListCalc(ref chart1, "func", F1, startX, endX, 1, step);

}

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

{

switch (currentMod)

{

case Calculator.CalcMod.EULER:

{

arrayCalc.setEulerCalcMod();

arrayCalc.CalcDot();

int strcount = dataGridView1.Rows.Count;

dataGridView1.Rows.Add();

dataGridView1.Rows[strcount].Cells[0].Value = arrayCalc.LastPoint().XValue;

dataGridView1.Rows[strcount].Cells[1].Value = arrayCalc.LastPoint().YValues[0];

arrayCalc.DrawDot();

break;

}

case Calculator.CalcMod.TRAPEZOIDAL:

{

arrayCalc2.setYsovCalcMod();

arrayCalc2.CalcDot();

int strcount = dataGridView1.Rows.Count;

dataGridView1.Rows.Add();

dataGridView1.Rows[strcount].Cells[0].Value = arrayCalc2.LastPoint().XValue;

dataGridView1.Rows[strcount].Cells[1].Value = arrayCalc2.LastPoint().YValues[0];

arrayCalc2.DrawDot();

break;

}

case Calculator.CalcMod.RUNGE\_KHUTT3:

{

listCalc1.setRunge3CalcMod();

listCalc1.CalcDot();

int strcount = dataGridView1.Rows.Count;

dataGridView1.Rows.Add();

dataGridView1.Rows[strcount].Cells[0].Value = listCalc1.LastPoint().XValue;

dataGridView1.Rows[strcount].Cells[1].Value = listCalc1.LastPoint().YValues[0];

listCalc1.DrawDot();

break;

}

case Calculator.CalcMod.RUNGE\_KHUTT4:

{

listCalc2.setRunge4CalcMod();

listCalc2.CalcDot();

int strcount = dataGridView1.Rows.Count;

dataGridView1.Rows.Add();

dataGridView1.Rows[strcount].Cells[0].Value = listCalc2.LastPoint().XValue;

dataGridView1.Rows[strcount].Cells[1].Value = listCalc2.LastPoint().YValues[0];

listCalc2.DrawDot();

break;

}

}

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

currentMod = Calculator.CalcMod.EULER;

dataGridView1.Rows.Clear();

arrayCalc.DrawDots();

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

currentMod = Calculator.CalcMod.TRAPEZOIDAL;

dataGridView1.Rows.Clear();

arrayCalc2.DrawDots();

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

currentMod = Calculator.CalcMod.RUNGE\_KHUTT3;

dataGridView1.Rows.Clear();

listCalc1.DrawDots();

}

private void radioButton7\_CheckedChanged(object sender, EventArgs e)

{

currentMod = Calculator.CalcMod.RUNGE\_KHUTT4;

dataGridView1.Rows.Clear();

listCalc2.DrawDots();

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

currentFunc = F1;

double step, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

step = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

arrayCalc.restart(currentFunc, startX, endX, 1, step);

arrayCalc2.restart(currentFunc, startX, endX, 1, step);

listCalc1.restart(currentFunc, startX, endX, 1, step);

listCalc2.restart(currentFunc, startX, endX, 1, step);

dataGridView1.Rows.Clear();

}

private void radioButton5\_CheckedChanged(object sender, EventArgs e)

{

currentFunc = F2;

double step, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

step = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

arrayCalc.restart(currentFunc, startX, endX, 1, step);

arrayCalc2.restart(currentFunc, startX, endX, 1, step);

listCalc1.restart(currentFunc, startX, endX, 1, step);

listCalc2.restart(currentFunc, startX, endX, 1, step);

dataGridView1.Rows.Clear();

}

private void radioButton6\_CheckedChanged(object sender, EventArgs e)

{

currentFunc = F3;

double step, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

step = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

arrayCalc.restart(currentFunc, startX, endX, 1, step);

arrayCalc2.restart(currentFunc, startX, endX, 1, step);

listCalc1.restart(currentFunc, startX, endX, 1, step);

listCalc2.restart(currentFunc, startX, endX, 1, step);

dataGridView1.Rows.Clear();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

double step, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

step = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

arrayCalc.restart(currentFunc, startX, endX, 1, step);

arrayCalc2.restart(currentFunc, startX, endX, 1, step);

listCalc1.restart(currentFunc, startX, endX, 1, step);

listCalc2.restart(currentFunc, startX, endX, 1, step);

dataGridView1.Rows.Clear();

}

}

}

**Файл Calculator.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_7

{

public abstract class Calculator

{

public enum CalcMod { EULER, TRAPEZOIDAL, RUNGE\_KHUTT3, RUNGE\_KHUTT4 }

public delegate double DFunc(double x, double y);

protected CalcMod calcMod = CalcMod.EULER;

protected Chart chart;

public double X1 { get; set; }

public double X2 { get; set; }

public double Y { get; set; } = 1;

public double Step { get; set; }

public string NameOfSeries { get; set; }

public DFunc F { get; set; }

protected Calculator(ref Chart chart, string nameOfSeries, DFunc f, double x1, double x2, double y, double step)

{

this.chart = chart;

X1 = x1;

X2 = x2;

Y = y;

Step = step;

NameOfSeries = nameOfSeries;

F = f;

}

public void setEulerCalcMod() { calcMod = CalcMod.EULER; }

public void setYsovCalcMod() { calcMod = CalcMod.TRAPEZOIDAL; }

public void setRunge3CalcMod() { calcMod = CalcMod.RUNGE\_KHUTT3; }

public void setRunge4CalcMod() { calcMod = CalcMod.RUNGE\_KHUTT4; }

public bool CalcEulerMethodStep(out DataPoint point)

{

if (X1 < X2)

{

Y += Step \* F(X1, Y);

X1 += Step;

point = new DataPoint(X1, Y);

return true;

}

point = null;

return false;

}

public bool CalcTrapezoidalMethodStep(out DataPoint point)

{

if (X1 < X2)

{

double k1 = Step \* F(X1, Y);

double k2 = Step \* F(X1 + Step, Y + k1);

Y += (k1 + k2) / 2;

X1 += Step;

point = new DataPoint(X1, Y);

return true;

}

point = null;

return false;

}

public bool CalcRungeCutt3MethodStep(out DataPoint point)

{

if (X1 < X2)

{

double k1 = F(X1, Y) \* Step;

double k2 = F(X1 + Step / 2, Y + Step \* k1 / 2) \* Step;

double k3 = F(X1 + Step, Y + 2 \* k2 + k1) \* Step;

Y += (k1 + 4 \* k2 + k3) / 6;

X1 += Step;

point = new DataPoint(X1, Y);

return true;

}

point = null;

return false;

}

public bool CalcRungeCutt4MethodStep(out DataPoint point)

{

if (X1 < X2)

{

double k1 =F(X1, Y) \* Step;

double k2 =F(X1 + Step / 2, Y + Step \* k1 / 2) \* Step;

double k3 =F(X1 + Step / 2, Y + Step \* k2 / 2) \* Step;

double k4 =F(X1 + Step, Y + k3) \* Step;

Y += (k1 + 2 \* k2 + 2 \* k3 + k4) / 6;

X1 += Step;

point = new DataPoint(X1, Y);

return true;

}

point = null;

return false;

}

}

}

**Файл ArrayList.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_7.Babies

{

public class ArrayListCalc : Calculator

{

private List<DataPoint> dataPoints;

public List<DataPoint> DataPoints { get => dataPoints; set => dataPoints = value; }

public ArrayListCalc(ref Chart chart, string nameOfSeries, DFunc f, double x1, double x2, double y, double step) : base(ref chart, nameOfSeries, f, x1, x2, y, step)

{

dataPoints = new List<DataPoint>();

}

public void restart(DFunc f, double x1, double x2, double y, double step)

{

X1 = x1;

X2 = x2;

Y = y;

Step = step;

F = f;

dataPoints.Clear();

chart.Series[NameOfSeries].Points.Clear();

}

public DataPoint CalcDot()

{

switch (calcMod)

{

case CalcMod.EULER:

{

if (CalcEulerMethodStep(out DataPoint point))

{

dataPoints.Add(point);

return point;

}

break;

}

case CalcMod.TRAPEZOIDAL:

{

if (CalcTrapezoidalMethodStep(out DataPoint point))

{

dataPoints.Add(point);

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT3:

{

if (CalcRungeCutt3MethodStep(out DataPoint point))

{

dataPoints.Add(point);

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT4:

{

if (CalcRungeCutt4MethodStep(out DataPoint point))

{

dataPoints.Add(point);

return point;

}

break;

}

}

return null;

}

public void DrawDot()

{

chart.Series[NameOfSeries].Points.AddXY(dataPoints[dataPoints.Count - 1].XValue, dataPoints[dataPoints.Count - 1].YValues[0]);

}

public DataPoint LastPoint()

{

return dataPoints[dataPoints.Count - 1];

}

public void DrawDots()

{

chart.Series[NameOfSeries].Points.Clear();

foreach (DataPoint point in dataPoints)

{

chart.Series[NameOfSeries].Points.AddXY(point.XValue, point.YValues[0]);

}

}

}

}

**Файл LinkListCalc.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_7.Babies

{

public class LinkListCalc : Calculator

{

private LinkedList<DataPoint> dataPoints;

public LinkedList<DataPoint> DataPoints { get => dataPoints; set => dataPoints = value; }

public LinkListCalc(ref Chart chart, string nameOfSeries, DFunc f, double x1, double x2, double y, double step) : base(ref chart, nameOfSeries, f, x1, x2, y, step)

{

dataPoints = new LinkedList<DataPoint>();

}

public void restart(DFunc f, double x1, double x2, double y, double step)

{

X1 = x1;

X2 = x2;

Y = y;

Step = step;

F = f;

dataPoints.Clear();

chart.Series[NameOfSeries].Points.Clear();

}

public DataPoint CalcDot()

{

switch (calcMod)

{

case CalcMod.EULER:

{

if (CalcEulerMethodStep(out DataPoint point))

{

dataPoints.AddLast(point);

return point;

}

break;

}

case CalcMod.TRAPEZOIDAL:

{

if (CalcTrapezoidalMethodStep(out DataPoint point))

{

dataPoints.AddLast(point);

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT3:

{

if (CalcRungeCutt3MethodStep(out DataPoint point))

{

dataPoints.AddLast(point);

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT4:

{

if (CalcRungeCutt4MethodStep(out DataPoint point))

{

dataPoints.AddLast(point);

return point;

}

break;

}

}

return null;

}

public void DrawDot()

{

chart.Series[NameOfSeries].Points.AddXY(dataPoints.Last.Value.XValue, dataPoints.Last.Value.YValues[0]);

}

public DataPoint LastPoint()

{

return dataPoints.Last.Value;

}

public void DrawDots()

{

chart.Series[NameOfSeries].Points.Clear();

foreach (DataPoint point in dataPoints)

{

chart.Series[NameOfSeries].Points.AddXY(point.XValue, point.YValues[0]);

}

}

}

}

**Файл StrListCalc.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms.DataVisualization.Charting;

namespace lab\_7.Babies

{

public class StrListCalc : Calculator

{

private List<string[]> dataPoints;

public List<string[]> DataPoints { get => dataPoints; set => dataPoints = value; }

public StrListCalc(ref Chart chart, string nameOfSeries, DFunc f, double x1, double x2, double y, double step) : base(ref chart, nameOfSeries, f, x1, x2, y, step)

{

dataPoints = new List<string[]>();

}

public void restart(DFunc f, double x1, double x2, double y, double step)

{

X1 = x1;

X2 = x2;

Y = y;

Step = step;

F = f;

dataPoints.Clear();

chart.Series[NameOfSeries].Points.Clear();

}

public DataPoint CalcDot()

{

switch (calcMod)

{

case CalcMod.EULER:

{

if (CalcEulerMethodStep(out DataPoint point))

{

dataPoints.Add(new string[] { point.XValue.ToString(), point.YValues[0].ToString() });

return point;

}

break;

}

case CalcMod.TRAPEZOIDAL:

{

if (CalcTrapezoidalMethodStep(out DataPoint point))

{

dataPoints.Add(new string[] { point.XValue.ToString(), point.YValues[0].ToString() });

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT3:

{

if (CalcRungeCutt3MethodStep(out DataPoint point))

{

dataPoints.Add(new string[] { point.XValue.ToString(), point.YValues[0].ToString() });

return point;

}

break;

}

case CalcMod.RUNGE\_KHUTT4:

{

if (CalcRungeCutt4MethodStep(out DataPoint point))

{

dataPoints.Add(new string[] { point.XValue.ToString(), point.YValues[0].ToString() });

return point;

}

break;

}

}

return null;

}

public void DrawDot()

{

chart.Series[NameOfSeries].Points.AddXY(

Convert.ToDouble(dataPoints[dataPoints.Count - 1][0]),

Convert.ToDouble(dataPoints[dataPoints.Count - 1][1]));

}

public string[] LastPoint()

{

return dataPoints[dataPoints.Count - 1];

}

public void DrawDots()

{

chart.Series[NameOfSeries].Points.Clear();

foreach (string[] point in dataPoints)

{

chart.Series[NameOfSeries].Points.AddXY(

Convert.ToDouble(dataPoints[dataPoints.Count - 1][0]),

Convert.ToDouble(dataPoints[dataPoints.Count - 1][1]));

}

}

}

}