

по дисциплине Практикум по программированию

Выполнил студент группы 23533/2

Руководитель

ассистент

В.Э. Ковалевский

ФГАОУ ВО «Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Высшая школа киберфизических систем и управления

Санкт-Петербург 2019

30.05.2019

А.А. Тарасов

Отчет по лабораторной работе

Графика. Построение кривых.

# Построение кривых

На практике часто стоит задача построения кривых или каких-то графиков функций (что принципиально одно и то же). Для построения графиков удобно использовать библиотеку ZedGraph на платформе .NET Framework. Приведённый ниже код написан на языке С#, по синтаксису очень похожий на С++.

Задача – построить график заданной функции на заданном промежутке, вычислить определённый интеграл на нём же квадратурной формулой Гаусса. Построить график аппроксимирующего функцию интерполяционного полинома.

Основной код:

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

{

try{

IntegralList.Clear();

GraphPane pane = zedGraphControl1.GraphPane;

pane.XAxis.Scale.MinAuto = true;

pane.XAxis.Scale.MaxAuto = true;

pane.YAxis.Scale.MinAuto = true;

pane.YAxis.Scale.MaxAuto = true;

pane.CurveList.Clear();

zedGraphControl1.AxisChange();

zedGraphControl1.Invalidate();

int n = 0;

if (Nodes\_2.Checked)

NodesQuantity = 2;

else if (Nodes\_3.Checked)

NodesQuantity = 3;

else if (Nodes\_4.Checked)

NodesQuantity = 4;

StreamReader IN = new StreamReader(Datafile);

string FileData = IN.ReadToEnd();

IN.Close();

string[] OrderData = FileData.Split(' ', '\n');

n = 0;

for (int i = 0; i < OrderData.Length - 1; i += 2)

{

n++;

IntegralList.Add(double.Parse(OrderData[i]), double.Parse(OrderData[i + 1]));

}

if (n < 2)

{

throw new ApplicationException("Степень интерполяции должна быть меньше количества точек.");

}

for (int i = 0; i < n - 1; i++)

{

for (int j = i + 1; j < n; j++)

{

if (IntegralList[i].X == IntegralList[j].X && IntegralList[i].Y != IntegralList[j].Y)

throw new ApplicationException("Зависимость многозначна");

}

}

IntegralList.TrimExcess();

IntegralList.Sort();

Plot\_Lagrange(NodesQuantity, IntegralList, pane);

LineItem curve1 = pane.AddCurve("y(x)", IntegralList, Color.Black, SymbolType.None);

curve1.Line.Width = 3;

curve1.Line.Fill = new ZedGraph.Fill(Color.CadetBlue, Color.Blue, Color.LightSkyBlue, 0);

pane.XAxis.Scale.Min = IntegralList[0].X;

pane.XAxis.Scale.Max = IntegralList[IntegralList.Count - 1].X;

pane.XAxis.Scale.BaseTic = IntegralList[0].X;

zedGraphControl1.AxisChange();

zedGraphControl1.Invalidate();

RightBorder = IntegralList[IntegralList.Count - 1].X;

LeftBorder = IntegralList[0].X;

double epsilon = 0.001, Iold = 0, Inew = 0;

// textBox1.Text = string.Format((0).ToString());

Iold = Integrate(NodesQuantity, LeftBorder, RightBorder);

double IntervalLength = Math.Abs(RightBorder - LeftBorder);

for (int i = 2; i <= 512; i+=2)

{

Inew = 0;

double step = Math.Abs(IntervalLength / i);

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

{

Inew += Integrate(NodesQuantity, LeftBorder+j\*step, LeftBorder + (1+j)\*step);

}

if (Math.Abs((Inew - Iold) / Inew) < epsilon)

{

Inew = Iold;

textBox1.Text = string.Format(((int)i / 2).ToString());

break;

}

else

Iold = Inew;

}

//IntegralValueText.Text = string.Format("{0:F6}", IntegralCalculate(IntegralList, AmountOfNodes));

//IntegralValueText.Text = string.Format("{0:F6}", IntegralCircle(IntegralList, AmountOfNodes, LeftBorder, RightBorder));

//IntegralValueText.Text = string.Format("{0:F6}", IntegralCalculate(LagrList,4));

IntegralValueText.Text = string.Format("{0:F6}",Iold);

}

catch (FileLoadException)

{

MessageBox.Show("Файл не найдён");

}

catch (FormatException)

{

MessageBox.Show("Данные в файле некорректны");

}

catch (Exception ea)

{

MessageBox.Show(ea.Message, "Ошибка");

}

}

Выше представлена функция обработки события нажатия на определённую кнопку. При нажатии на кнопку строятся графики аппроксимирующего полинома и график исходной функции y = exp(-x) \* sin(x) \* x + x. Для 2х узлов Гаусса строится полином 3 степени, для 3х – 5й, для 4 – 7й. Интерполяционные узлы полинома равноудалены, полином построен методом Лагранжа.

Сначала объявляется объект, представляющий собой координатную плоскость. GraphPane pane = zedGraphControl1.GraphPane;

Затем настраиваются масштабирование по оси и считываются из файла данные точек, по которым будет построен график функции в список точек. Точки были вычислены и записаны в файл заранее.

После этого строится уже сам график функции, пространство под ним заливается синим цветом. Настроена толщина кривой.

LineItem curve1 = pane.AddCurve("y(x)", IntegralList, Color.Black, SymbolType.None);

curve1.Line.Width = 3;

curve1.Line.Fill = new ZedGraph.Fill(Color.CadetBlue, Color.Blue, Color.LightSkyBlue, 0);

Вызвается функция построения интерполяционного полинома.

Plot\_Lagrange(NodesQuantity, IntegralList, pane);

//ExtraLagr

Её реализация:

public void Plot\_Lagrange(int AmountOfNodes, PointPairList IntegralList, GraphPane pane)

{

//Полином лагранжа построение

PointPairList LagrList = new PointPairList();

LagrList.Clear();

PointPairList n\_list = new PointPairList(); //Тут точки по которым строится полоином

double step = (IntegralList[IntegralList.Count() - 1].X - IntegralList[0].X) / 100;

double first = IntegralList[0].X;

double last = IntegralList[IntegralList.Count()-1].X;

double x = IntegralList[0].X;

if (AmountOfNodes == 2)

{

//Полином 3 степени тогда -> 4 точки нужно

double st = (last - first) / 3.0;

for(int i=0; i<4; i++)

{

n\_list.Add(first+ st\*i, Calculate(first + st\*i));

}

//n\_list.Add(last, Calculate(last));

n\_list.Sort();

for (int i = 0; i <= 100; i++)

{

LagrList.Add(x, NLagrange(n\_list, 3, x));

x += step;

}//

}//

else if (AmountOfNodes == 3)

{

double st = (last - first) / 5.0;

for (int i = 0; i < 6; i++)

{

n\_list.Add(first + st \* i, Calculate(first + st \* i));

}

n\_list.Sort();

for (int i = 0; i <= 100; i++)

{

LagrList.Add(x, NLagrange(n\_list, 5, x));

x += step;

}//

}//

else if (AmountOfNodes == 4)

{

double st = (last - first) / 7.0;

for (int i = 0; i < 8; i++)

{

n\_list.Add(first + st \* i, Calculate(first + st \* i));

}

n\_list.Sort();

for (int i = 0; i <= 100; i++)

{

LagrList.Add(x, NLagrange(n\_list, 7, x));

x += step;

}//

}

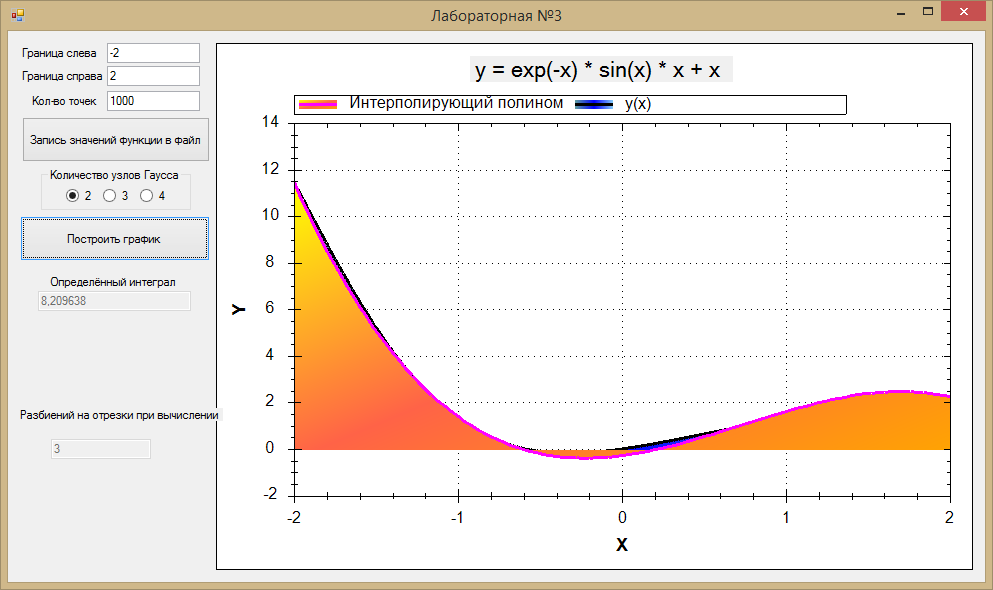
LineItem curve2 = pane.AddCurve("Интерполирующий полином", LagrList, Color.Magenta, SymbolType.None);

curve2.Line.Width = 3;

curve2.Line.Fill = new ZedGraph.Fill(Color.Yellow,Color.Tomato, Color.Orange, 45.0f);

}

Итоговый результат:



Строить графики на С# гораздо удобнее, чем использовать Visual C++ Построим теперь график какой-то функции в Winows Forms, только на С++. Использован объект Chart – стандартный объект, представляющий собой координатную плоскость в стандартной коллекции Windows Forms.

graph->Series[0]->Points->Clear(); // Почистить график

//Конвертация в double

double d\_border2, d\_step, error;

try

{

d\_border2 = System::Convert::ToDouble(this->border\_2->Text);

d\_step = System::Convert::ToDouble(this->step->Text);

if (d\_border2 > 14.5)

{

MessageBox::Show(this, "Слишком большое число введено. у -> 0");

return;

}

if (d\_border2 == 0)

{

MessageBox::Show(this, "Вам не нужно строить график функции на промежутке [0, 0]");

return;

}

else if (d\_border2 < 0)

{

MessageBox::Show(this, "Не строим график при х < 0");

return;

}

//MessageBox::Show(d\_border2.ToString());

}

catch (...) //А если будет ошибка

{

MessageBox::Show(this, "Ошибка при вводе данных. Введите корректные данные.");

return;

}

//

//Теперь конвертация в double была успешна

//

error = d\_border2 / 1000;

if (d\_step<error || d\_step>d\_border2 / 2)

{

d\_step = error;

//MessageBox::Show(this, "Ошибка при вводе данных");

//return;

}

//Полезно (int).ToString();

//double interval = d\_border2 / 10.0;

graph->ChartAreas[0]->AxisX->Minimum = 0;

graph->ChartAreas[0]->AxisX->Maximum = d\_border2+0.05\*d\_border2;

//graph->ChartAreas[0]->AxisX->IntervalAutoMode = System::Windows::Forms::DataVisualization::Charting::IntervalAutoMode::FixedCount;

//System::Windows::Forms::DataVisualization::Charting::Title;

//MessageBox::Show((graph->ChartAreas[0]->AxisX->Interval).ToString());

graph->ChartAreas[0]->AxisX->Crossing = 0;

graph->ChartAreas[0]->AxisY->Crossing = 0;

graph->ChartAreas[0]->AxisX->Interval = d\_border2/10;

//graph->ChartAreas[0]->AxisX->Title = "Ось X ->";

//graph->ChartAreas[0]->AxisX->LabelStyle->Format = "yyyy-MM-dd";

step->Text = d\_step.ToString();

std::ofstream log;

log.open("log.txt");

int counter = 0;

double y;

for (double i = 0; i < d\_border2 ; i += d\_step)

{

log << counter << ") \t";

y = evaluate(i);

log << counter << '\t' << i << "\t" << y << '\t';

graph->Series[0]->Points->AddXY(i, y);

log << counter++ << " OK"<< std::endl;

}

/\*double interval = graph->ChartAreas[0]->AxisX->Interval;

MessageBox::Show(interval.ToString());

//MessageBox::Show("Финал");\*/

graph->Series[0]->Points->AddXY(d\_border2, evaluate(d\_border2));

Данная функция строит график функции 0.5 \* exp(-2 \* x) \* (x + exp(x)\*(2\*x - 3)\*sin(x) - exp(x)\*(x - 2)\*x\*cos(x)) на промежутке [0;x] где х выбирается пользователем. Шаг построения функции также выбирается пользователем.

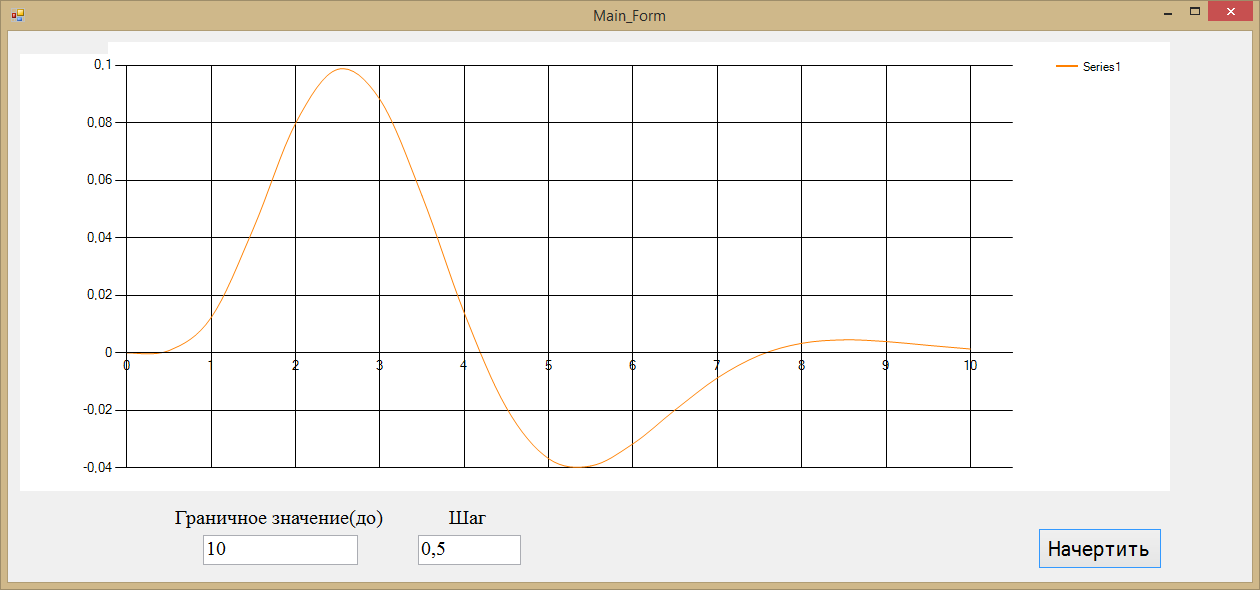


Рисунок – Результат вызова функции построения графика

# Вывод:

В данной работе были освоены навыки работы со сторонними библиотеками для построения графиков на платформе .NET Framework на языках С# и С++, а также разобраны стандартные средства построения графиков в windows forms.