**Практическая работа №3**

**Оглавление**

[Практическая работа № 3. Построение графиков результатов математических вычислений 3](#_Toc163812680)

[Содержание пояснительной записки 17](#_Toc163812681)

[Используемое программное обеспечение 18](#_Toc163812682)

[Список литературы 19](#_Toc163812683)

Практическая работа № 3.  
Построение графиков  
результатов математических вычислений

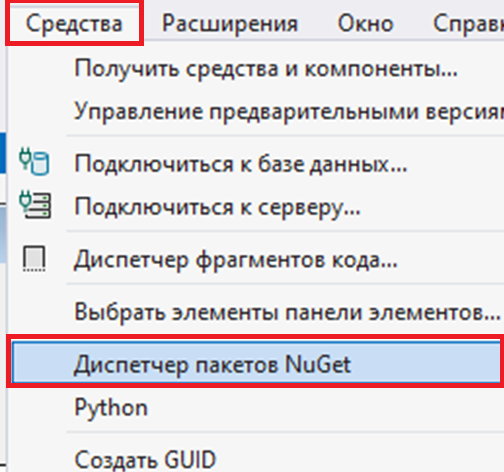
**Цель практической работы**

Закрепление теоретических знаний по построению графиков в программах с графическим интерфейсом.

**Постановка задачи**

**Построение графиков с помощью ScottPlot.**

Рассмотрим пример добавления к проекту библиотеки для построения графиков ScottPlot. Необходимо открыть диспетчер пакетов **NuGet** (рисунок 1).



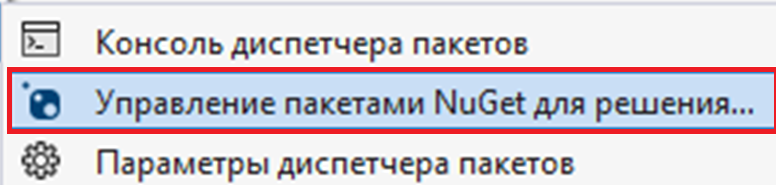


Рисунок 1 – Установка пакетов с помощью NuGet

Необходимо перейти на вкладку «Обзор» найти библиотеки ScottPlot и ScottPlot.WinForms, и нажать «Установить» (рисунок 2).

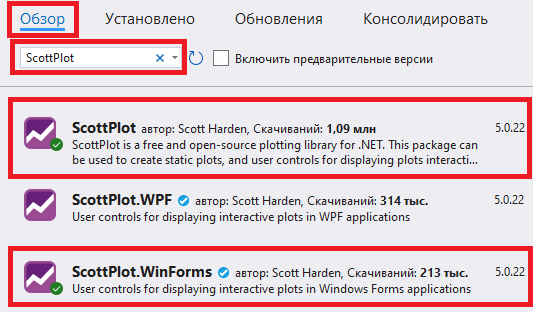


Рисунок 2 – Установка пакета ScottPlot

Элемент для построения графиков (FormsPlot) появится в панели элементов (рисунок 3). После установки пакета **возможно придется перезапустить приложение** (если появляется ошибка при добавлении элемента FormsPlot).

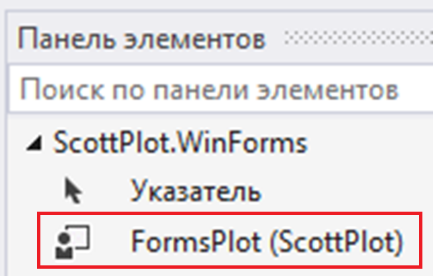


Рисунок 3 – Отображение ScottPlot в панели элементов

Добавим подписи осей, название графика и отобразим что получилось (рисунок 4).

formsPlot1.Plot.XLabel("X, м");

formsPlot1.Plot.YLabel("Y, м");

formsPlot1.Plot.Title("График");

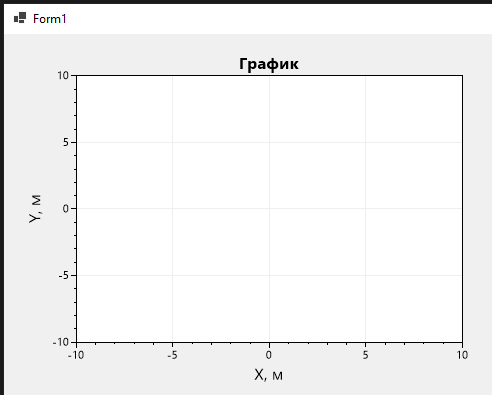


Рисунок 4 – Пустой график ScottPlot на форме

Построим график функции *f(x) = x*2 на участке от 0 до 100 (рисунок 5).

var xs = new List<double>();

var ys = new List<double>();

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

{

xs.Add(i);

ys.Add(i \* i);

}

var scatter = formsPlot1.Plot.Add.Scatter(xs, ys);

// Убираем маркеры (точки) и оставляем линию

scatter.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter.Label = "x^2"; // Подписии линии графика

formsPlot1.Plot.ShowLegend();

formsPlot1.Plot.Axes.AutoScale(); // Автоматически приблизить к области графика

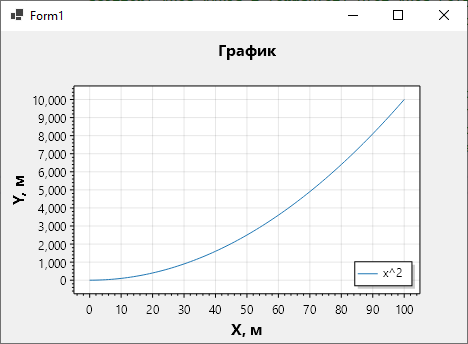


Рисунок 5 – Отображение графика функции на форме

Основное преимущество ScottPlot это **интерактивность**, график можно масштабировать и перемещаться по графику, также он сам определяет интервалы подписей у осей.

**Чтобы переместиться по графику** нужно зажать ЛКМ и двигать мышью, чтобы приблизиться (рисунок 6) или отдалиться – вращать колесико мыши, также чтобы приблизить некоторую область можно зажать колесико и выделить прямоугольник. Чтобы вернуться к исходному виду нужно нажать на колесико.

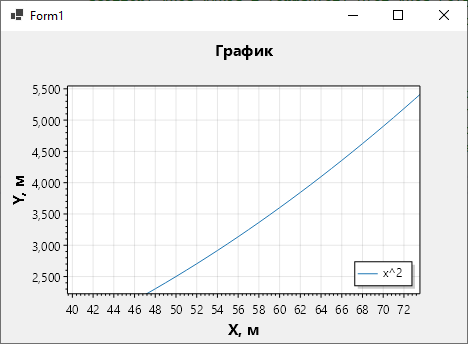


Рисунок 6 – Приближение участка графика

ScottPlot поддерживает рисование по второстепенным осям, это полезно, когда 2 набора данных сильно отличаются по значениям, но их требуется представить на одном графике (Например, давление и температура).

var xs = new List<double>();

var ys = new List<double>();

var ys2 = new List<double>();

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

{

xs.Add(i);

ys.Add(i \* i);

ys2.Add(Math.Sin(i));

}

Отобразим *x*2 на основной оси, а sin(*x*) на второстепенной оси *Oy.* Также сделаем график синуса пунктирным (рисунок 7).

var scatter1 = formsPlot1.Plot.Add.Scatter(xs, ys);

scatter1.Axes.XAxis = formsPlot1.Plot.Axes.Bottom;

scatter1.Axes.YAxis = formsPlot1.Plot.Axes.Left;

scatter1.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter1.Label = "x^2";

var scatter2 = formsPlot1.Plot.Add.Scatter(xs, ys2);

scatter2.Axes.XAxis = formsPlot1.Plot.Axes.Bottom;

scatter2.Axes.YAxis = formsPlot1.Plot.Axes.Right;

scatter2.MarkerStyle = ScottPlot.MarkerStyle.None;

scatter2.LinePattern = ScottPlot.LinePattern.Dashed;

scatter2.Label = "sin(x)";

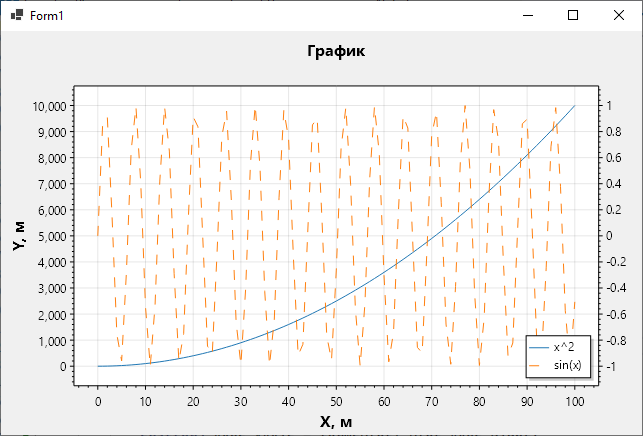


Рисунок 7 – Отображение графиков  
на главной и второстепенной осях

Рассмотрим пример построения гистограмм (рисунок 8):

var plot = formsPlot1.Plot;

double[] xs1 = { 1, 2, 3, 4 };

double[] ys1 = { 5, 10, 7, 13 };

var bars1 = plot.Add.Bars(xs1, ys1);

bars1.Label = "Alpha";

double[] xs2 = { 6, 7, 8, 9 };

double[] ys2 = { 7, 12, 9, 15 };

var bars2 = plot.Add.Bars(xs2, ys2);

bars2.Label = "Beta";

plot.ShowLegend(Alignment.UpperLeft);

// Убираем отсут от нижней границы

plot.Axes.Margins(bottom: 0);

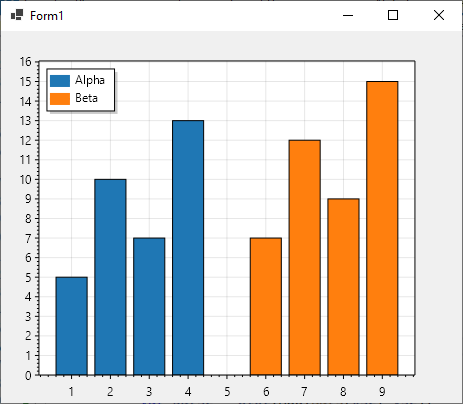


Рисунок 8 – Отображение гистограммы

Рассмотрим пример построения круговой диаграммы (рисунок 9):

var plot = formsPlot1.Plot;

List<PieSlice> slices = new()

{

new PieSlice(){ Value = 5, FillColor = Colors.Red, Label = "Red" },

new PieSlice(){ Value = 2, FillColor = Colors.Orange, Label ="Orange"},

new PieSlice(){ Value = 8, FillColor = Colors.Gold, Label ="Yellow"},

new PieSlice(){ Value = 4, FillColor = Colors.Green, Label ="Green"},

new PieSlice(){ Value = 8, FillColor = Colors.Blue, Label ="Blue" }};

var pie = plot.Add.Pie(slices);

pie.ExplodeFraction = 0.1;

plot.ShowLegend();

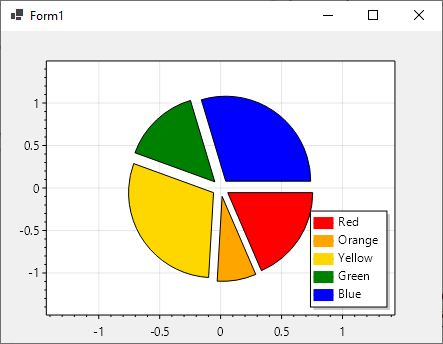


Рисунок 9 – Отображение гистограммы

**Библиотека MathNet.Numerics.**

Рассмотрим библиотеку для математических расчетов **MathNet.Numerics**. С помощью неё можно решать системы уравнений, интерполировать и экстраполировать данные, работать с комплексными числами и т.д.

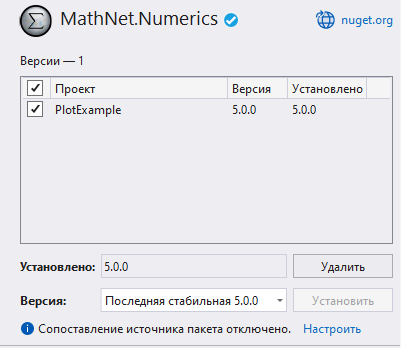


Рисунок 10 – Библиотека MathNet.Numerics

В данной библиотеке есть методы **интерполяции**, **дифференцирования**, **интегрирования** и т.д. **Интерполяция** – нахождение неизвестных промежуточных значений некоторой функции по известному дискретному набору ее значений (рисунок 11).

var x = new double[] {1.0, 10, 25, 35, 50, 65, 80, 90};

var y = new double[] {2.0, 3.5, 6, 7.5, 9, 10, 11, 13};

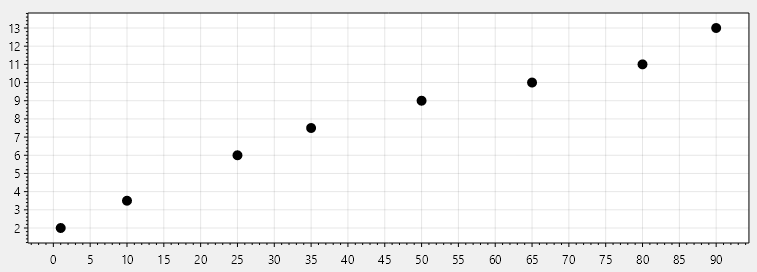


Рисунок 11 – Отображение табличных значений

Определим промежуточные значения между известными данными с помощью линейной интерполяции и построим график (рисунок 12).

var spline = LinearSpline.Interpolate(x, y);

var xSpline = new List<double>();

var ySpline = new List<double>();

for (double xi = x[0]; xi <= x[^1]; xi++)

{

var yInterpolated = spline.Interpolate(xi);

xSpline.Add(xi);

ySpline.Add(yInterpolated);

}

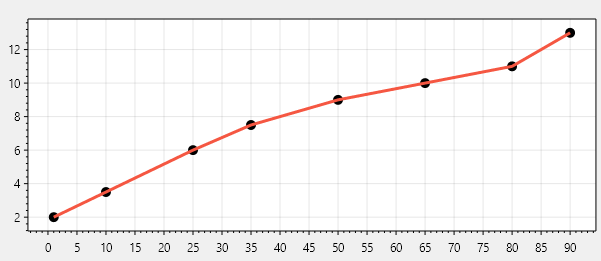


Рисунок 12 – Отображение линейно  
интерполированных значений

Также существует **кубическая** интерполяция. В данном случае точки соединяются кубическими функциями, например (рисунок 13):

var x = new double[] {1.0, 10, 25, 35, 50, 65, 80, 90};

var y = new double[] {2.0, 3.5, 6, 7.5, 9, 10, 11, 13};

var spline = CubicSpline.InterpolateNatural(x, y);

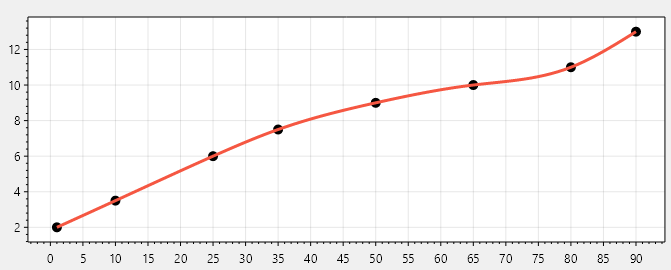


Рисунок 13 – Отображение кубически  
интерполированных значений

Из рисунка видно, что при кубической интерполяции график имеет гладкий вид.

**Задание на практическую работу**

1. Разработать приложение с графическим интерфейсом для построение графиков.
2. Варианты заданий представлены в пункте «**Варианты заданий на практическую работу».**
3. **В задании 1 необходимо** интерполировать данные и построить график функции согласно варианту.
4. **В задании 2 необходимо:**
   1. найти статистические данные по субъектам РФ согласно варианту.
   2. на одном графике на разных осях отобразить в зависимости от года: численность населения и среднюю зарплату.
5. Все оси и графики должны быть подписаны.
6. Защита работы включает демонстрацию работы программы на подготовленном примере.

**Варианты заданий на практическую работу**

**Задание 1.**

Используя метод CubicSpline.InterpolateNatural интерполировать данные и построить график функции.

В качестве данных используем таблицу стандартной атмосферы: <https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B0%D1%82%D0%BC%D0%BE%D1%81%D1%84%D0%B5%D1%80%D0%B0>

Независимая переменная (*x*) – высота, м. Зависимая переменная (*y*) – по варианту:

1. температура,
2. скорость звука,
3. давление,
4. плотность,
5. кинематическая вязкость.

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»