МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

***Факультет информационных технологий и робототехники***

Кафедра программного обеспечения информационных систем и технологий

**Отчет по лабораторной работе № 5**

по дисциплине: ”Разработка приложений в визуальных средах ”

# **на тему: *”* Разработка класса решения нелинейных уравнений*”***

Вариант 1

Выполнил**:** студент группы 10701322 Бородкин Д.В.

Принял**:** доц. Гурский Н. Н.

Минск 2024

**Лабораторная работа № 5.**

**Цель лабораторной работы:** изучить способы перекрытия методов родительских классов.

**Задание:** создать родительский класс решения нелинейного уравнения различными методами (методами деления отрезка пополам, методом итераций и методом Ньютона). В дочерних классах предусмотреть динамическое перекрытие нелинейных уравнений. Реализовать процедуры графической визуализации нахождения корня уравнения для каждого метода.

**Текст программы:**

using System;

using System.Drawing;

using System.Windows.Forms.DataVisualization.Charting;

using Timer = System.Windows.Forms.Timer;

namespace Lab\_5

{

public abstract class The\_nonlinear\_equation

{

public abstract double Calculate(); // абстрактный метод

public abstract void DrawAnim(int time); // абстрактный метод

public int Iter = 0;

protected Timer timer = new Timer();

// Геттер и сеттер для начала отрезка

public double Starting\_point { get; set; }

// Геттер и сеттер для конца отрезка

public double End\_point { get; set; }

// Геттер и сеттер для шага

public double Step { get; set; }

// Геттер и сеттер для mychart

public Chart Mychart { get; set; }

//Метод для 3\*sin(2\*x) - 1.5\*x-1 = 0

public double F1(double x)

{

return 3 \* Math.Sin(2 \* x) - 1.5 \* x - 1;

}

//Метод для x \* x + 2 \* x - 3;

public double F2(double x)

{

return x \* x + 2 \* x - 3;

}

//Метод для Cos(2\*x) - 2x - 4 = 0

public double F3(double x)

{

return Math.Cos(2 \* x) - 2 \* x - 4;

}

//Делегат для функции

public delegate double myFunc(double x);

public myFunc func;

public void DrawF()

{

// Установка свойств графика (например, подписей осей)

Mychart.ChartAreas[0].AxisX.Title = "Ось X";

Mychart.ChartAreas[0].AxisY.Title = "Ось Y";

// Создание нового объекта класса Series для графика линии

Series series = new Series

{

Name = "Функция",

ChartType = SeriesChartType.Spline,

};

Mychart.ChartAreas[0].AxisX.Minimum = Starting\_point;

Mychart.ChartAreas[0].AxisX.Maximum = End\_point;

Mychart.ChartAreas[0].AxisX.Interval = Step;

int count = (int)Math.Ceiling((End\_point + 2 \* 0.2f - Starting\_point) / 0.1f) + 1; //точность (плавность) графика

double[] x = new double[count];

double[] y = new double[count];

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

{

x[i] = Math.Round(Starting\_point - 0.2f + 0.1f \* i, 2);

y[i] = Math.Round(func(x[i]), 5);

}

series.Points.DataBindXY(x, y); // Привязываем данные к серии

Mychart.Series.Add(series); // Добавляем серию на график

series.Color = Color.Red;

series.BorderWidth = 2;

Mychart.Titles["Title1"].Text = "Нахождение корней уравнения"; // Загаловок

}

}

//производный класс Половинного деления на отрезки

public class Cut\_in\_half : The\_nonlinear\_equation

{

public override double Calculate()

{

double left = Starting\_point;

double right = End\_point;

double point = 0;

while ((right - left) / 2 > Step)

{

point = (left + right) / 2;

if ((func(left) \* func(point)) > 0) left = point;

else right = point;

Iter++;

}

return point;

}

public override void DrawAnim(int time)

{

double left = Starting\_point;

double rigth = End\_point;

double x1, x2 = 0;

double c = 0;

Series series1 = new Series

{

Name = "Половинного деления",

ChartType = SeriesChartType.Column,

};

timer.Interval = time;

timer.Tick += (sender, args) =>

{

bool b = (rigth - left) / 2 < Step;

if ((rigth - left) / 2 < Step)

{

timer.Stop();

}

c = (left + rigth) / 2;

x1 = c;

if ((func(left) \* func(c)) > 0) left = c;

else rigth = c;

series1.Points.AddXY(x1, func(x1));

Iter++;

x2 = c;

};

// Добавление созданного объекта Series на график

Mychart.Series.Add(series1);

series1.Color = Color.DimGray;

// Легенда

Mychart.Legends["Legend1"].Title = "Легенда";

Mychart.Legends["Legend1"].Docking = Docking.Bottom;

timer.Start();

}

}

//производный класс Итераций

public class Iteration : The\_nonlinear\_equation

{

public override double Calculate()

{

double x = (Starting\_point + End\_point) / 2;

double eps = Step, xLast, dx = double.MaxValue;

while (Math.Abs(dx) > eps)

{

xLast = x;

x = 3 / (x + 2);

dx = x - xLast;

Iter++;

}

return x;

}

public override void DrawAnim(int time)

{

// Создание нового объекта класса Series для графика линии

Series series2 = new Series

{

Name = "Метод итераций",

ChartType = SeriesChartType.StepLine,

};

double x = (Starting\_point + End\_point) / 2;

double xLast = 0, dx = double.MaxValue;

timer.Interval = time;

timer.Tick += (sender, args) =>

{

if (Math.Abs(dx) < Step)

{

timer.Stop();

}

series2.Points.AddXY(x, func(x));

xLast = x;

x = 3 / (x + 2);

dx = x - xLast;

Iter++;

};

// Добавление созданного объекта Series на график

Mychart.Series.Add(series2);

series2.Color = Color.DarkBlue;

series2.BorderWidth = 2;

// Легенда

Mychart.Legends["Legend1"].Title = "Легенда";

Mychart.Legends["Legend1"].Docking = Docking.Bottom;

timer.Start();

}

}

//производный класс Ньютона

public class Newton : The\_nonlinear\_equation

{

public override double Calculate()

{

double x = (Starting\_point + End\_point) / 2, df;

df = (func(x + Step) - func(x)) / Step;

while (Math.Abs(func(x)) > Step)

{

x -= func(x) / df;

Iter++;

}

return x;

}

public override void DrawAnim(int time)

{

// Создание нового объекта класса Series для графика линии

Series series3 = new Series

{

Name = "Ньютона",

ChartType = SeriesChartType.Line,

};

double x = (Starting\_point + End\_point) / 2, df;

df = (func(x + Step) - func(x)) / Step;

timer.Interval = time;

timer.Tick += (sender, args) =>

{

double i = Math.Abs(func(x));

if (Math.Abs(func(x)) < Step)

{

timer.Stop();

}

series3.Points.AddXY(x, 0);

series3.Points.AddXY(x, func(x));

x -= func(x) / df;

Iter++;

};

// Добавление созданного объекта Series на график

Mychart.Series.Add(series3);

series3.Color = Color.Blue;

series3.BorderWidth = 2;

// Легенда

Mychart.Legends["Legend1"].Title = "Легенда";

Mychart.Legends["Legend1"].Docking = Docking.Bottom;

timer.Start();

}

**Результаты выполнения программы:**

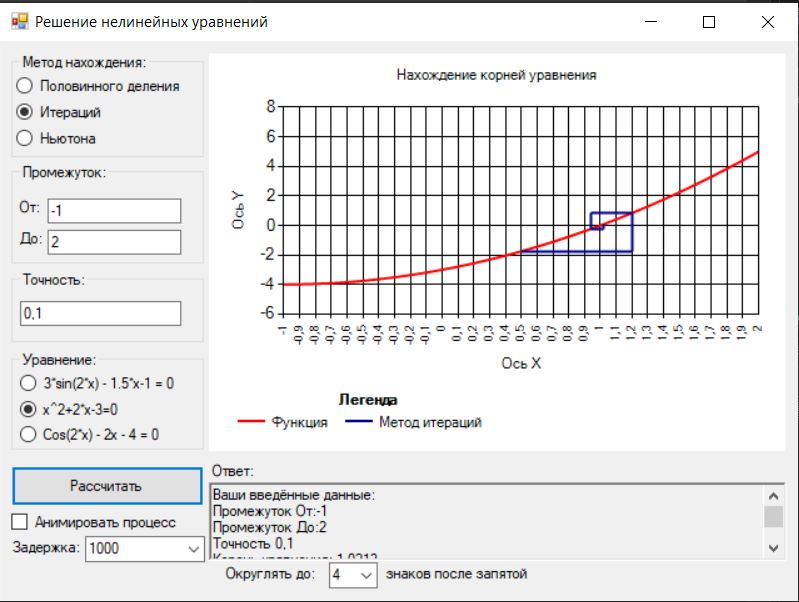


Рисунок 1. Результат работы программы.

**Вывод:** изучил способы перекрытия методов родительских классов.