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 № 5

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

тема: **«РАЗРАБОТКА КЛАССА РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ»**

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

2019 учебный год

**ЛАБОРАТОРНАЯ РАБОТА № 5**

**РАЗРАБОТКА КЛАССА РЕШЕНИЯ НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

Цель работы:

Изучить способы перекрытия методов родительских

классов.

Задание:

Создать родительский класс решения нелинейного уравнения

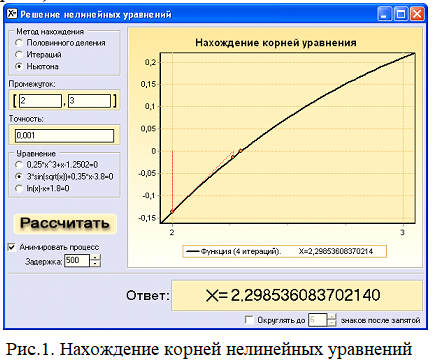
различными методами (методами деления отрезка пополам, методом

итераций и методом Ньютона). В дочерних классах предусмотреть

динамическое перекрытие нелинейных уравнений. Реализовать процедуры

графической визуализации нахождения корня уравнения для каждого

метода (см. рис.1).



Результаты выполнения задания:

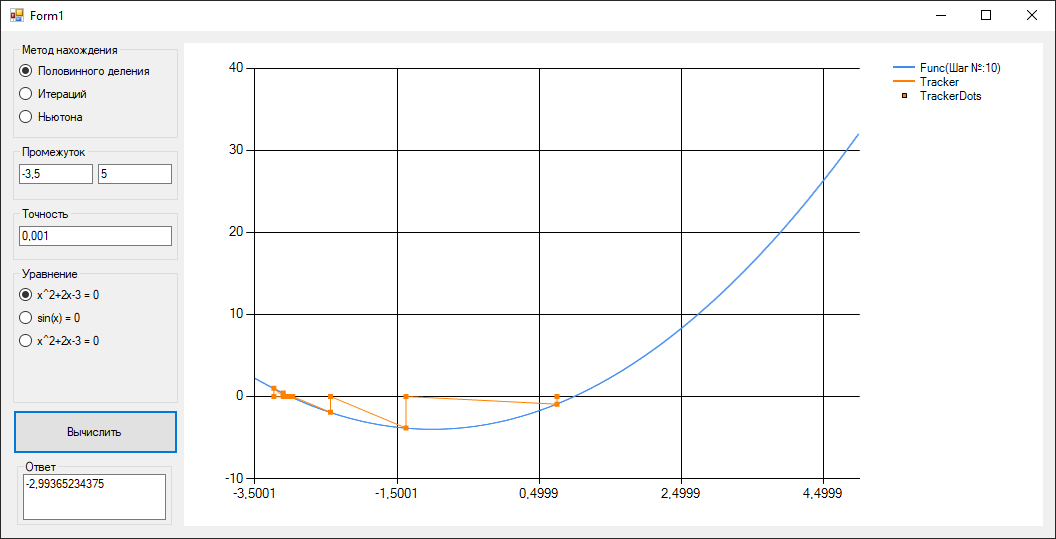


Рисунок 1 – Результат с методом половинного деления.

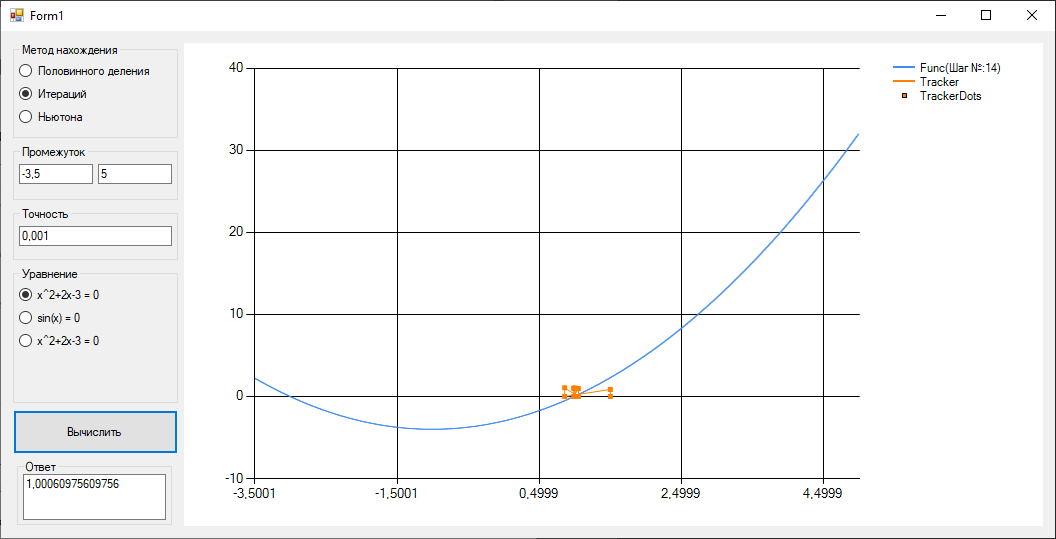


Рисунок 2 – Результат с методом итераций.

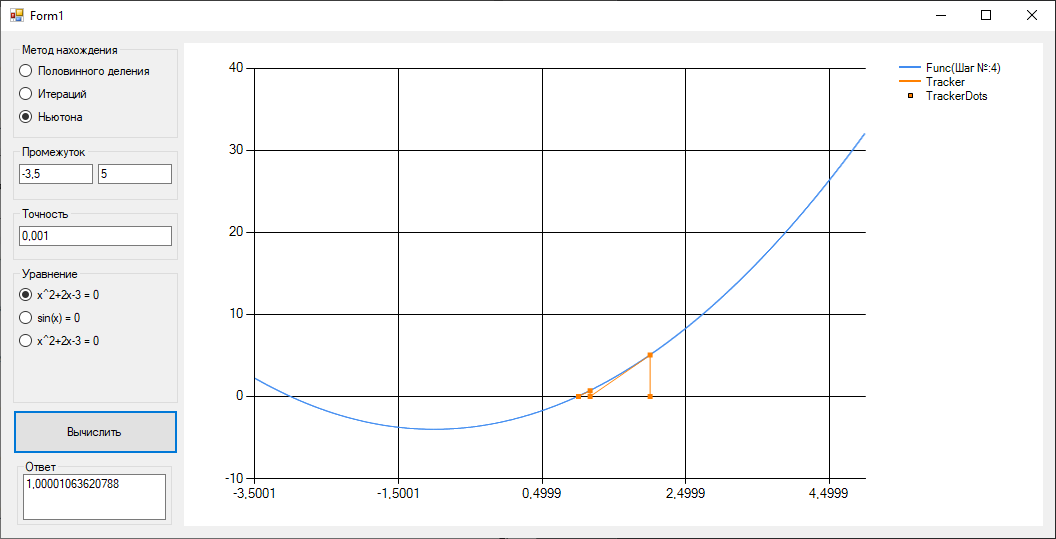


Рисунок 3 – Результат с методом ньютона.

*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\_5

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private enum Func {VAR3,SIN,VAR6}

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

{

double eps, endX, startX;

try

{

startX = Convert.ToDouble(textBox1.Text);

endX = Convert.ToDouble(textBox2.Text);

eps = Convert.ToDouble(textBox3.Text);

}

catch (Exception exception)

{

MessageBox.Show(exception.ToString());

return;

}

Father[] method = { new Var3(startX, endX, eps,ref chart1), new Sin(startX, endX, eps,ref chart1), new Var6(startX, endX, eps, ref chart1) };

Func func;

if (radioButton4.Checked)

func = Func.VAR3;

else if (radioButton5.Checked)

func = Func.SIN;

else

func = Func.VAR6;

if (radioButton1.Checked)

textBox4.Text = method[(int)func].methodMid().ToString();

else if (radioButton2.Checked)

textBox4.Text = method[(int)func].methodIter().ToString();

else

textBox4.Text = method[(int)func].methodNuton().ToString();

}

}

}

**Файл Father.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\_5

{

abstract class Father

{

private double startX; //левая граница

private double endX; //правая граница

private double eps; //точность

private Chart chart; //элемент рисования

public Father(double startX, double endX, double eps, Chart chart)

{

this.startX = startX;

this.endX = endX;

this.eps = eps;

this.chart = chart;

}

public double StartX { get => startX; set => startX = value; }

public double EndX { get => endX; set => endX = value; }

public double Eps { get => eps; set => eps = value; }

public double methodMid()

{

draw();

chart.Series["Tracker"].Points.Clear();

chart.Series["TrackerDots"].Points.Clear();

int iter = 0;

// Точность.

double accuracy = eps;

// Интервал поиска.

double min = startX;

double max = endX;

// Длина интервала.

var length = max - min;

// Начальная ошибка.

var err = length;

// Корень.

double x = 0;

while (err > accuracy && func(x) != 0)

{

// Вычисляем середину интервала.

x = (min + max) / 2;

// Найдём новый интервал, в котором функция меняет знак.

if (func(min) \* func(x) < 0)

{

max = x;

}

else if (func(x) \* func(max) < 0)

{

min = x;

}

// Вычисляем новую ошибку.

err = (max - min) / length;

chart.Series["Tracker"].Points.AddXY(x, 0);

chart.Series["TrackerDots"].Points.AddXY(x, 0);

chart.Series["Tracker"].Points.AddXY(x, func(x));

chart.Series["TrackerDots"].Points.AddXY(x, func(x));

iter++;

chart.Series["Func"].LegendText = $"Func(Шаг №:{iter})";

chart.Update();

System.Threading.Thread.Sleep(500);

}

return x;

}

public double methodIter()

{

draw();

chart.Series["Tracker"].Points.Clear();

chart.Series["TrackerDots"].Points.Clear();

double x0 = 0, x = 0;

int iter = 0;

bool error = false;

do

{

x = iterFunc(x0);

iter++;

if (Math.Abs(x - x0) >= eps && iter == 1000)

{

error = true;

break;

}

chart.Series["Tracker"].Points.AddXY(x, 0);

chart.Series["TrackerDots"].Points.AddXY(x, 0);

chart.Series["Tracker"].Points.AddXY(x, iterFunc(x));

chart.Series["TrackerDots"].Points.AddXY(x, iterFunc(x));

iter++;

chart.Series["Func"].LegendText = $"Func(Шаг №:{iter})";

chart.Update();

System.Threading.Thread.Sleep(10);

Console.WriteLine(x);

x0 = x;

} while (Math.Abs(x0 - iterFunc(x0)) > eps);

if (error)

{

Console.WriteLine("Не найдено");

}

else

{

Console.WriteLine("Ответ: X = {0} ", Math.Round(x, 3));

Console.WriteLine("Итераций пройдено: {0}", iter);

}

return x;

}

public double methodNuton()

{

draw();

chart.Series["Tracker"].Points.Clear();

chart.Series["TrackerDots"].Points.Clear();

double x = (endX - startX)/2;

int iter = 0;

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

{

x -= func(x) / diffFunc(x);

chart.Series["Tracker"].Points.AddXY(x,0);

chart.Series["TrackerDots"].Points.AddXY(x,0);

chart.Series["Tracker"].Points.AddXY(x,func(x));

chart.Series["TrackerDots"].Points.AddXY(x,func(x));

iter++;

chart.Series["Func"].LegendText = $"Func(Шаг №:{iter})";

chart.Update();

System.Threading.Thread.Sleep(1000);

//x += dx;

}

return x;

}

public void draw()

{

chart.Series["Func"].Points.Clear();

double dx = 0.0001;

for (double x = startX; x < endX; x += dx)

{

chart.Series["Func"].Points.AddXY(x, func(x));

}

}

abstract public double func(double x);

abstract public double diffFunc(double x);

abstract public double iterFunc(double x);

}

}

**Файл sin.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\_5

{

class Sin : Father

{

public Sin(double startX, double endX, double eps,ref Chart chart) : base(startX, endX, eps, chart)

{

}

public override double func(double x)

{

return Math.Sin(x);

}

public override double iterFunc(double x)

{

return Math.Asin(x);

}

public override double diffFunc(double x)

{

return Math.Cos(x);

}

}

}

**Файл var3.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\_5

{

class Var3 : Father

{

public Var3(double startX, double endX, double eps,ref Chart chart) : base(startX, endX, eps, chart)

{

}

public override double func(double x)

{

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

}

public override double iterFunc(double x)

{

return 3/(x+2);

}

public override double diffFunc(double x)

{

return 2 \* x + 2;

}

}

}

**Файл var6.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\_5

{

class Var6 : Father

{

public Var6(double startX, double endX, double eps,ref Chart chart) : base(startX, endX, eps, chart)

{

}

public override double func(double x)

{

return Math.Pow(x, 3) - 2 \* Math.Pow(x, 2) - x + 2;

}

public override double iterFunc(double x)

{

return -2 / (Math.Pow(x, 2) - 2 \* x - 1);

}

public override double diffFunc(double x)

{

return 3 \* Math.Pow(x, 2) - 4 \* x - 1;

}

}

}