1. Упростить выражение, которое пользователь вводит в виде строки. Найти значение этого выражения в заданныx пользователем точкаx.

Результат представлен на рисунке 1.

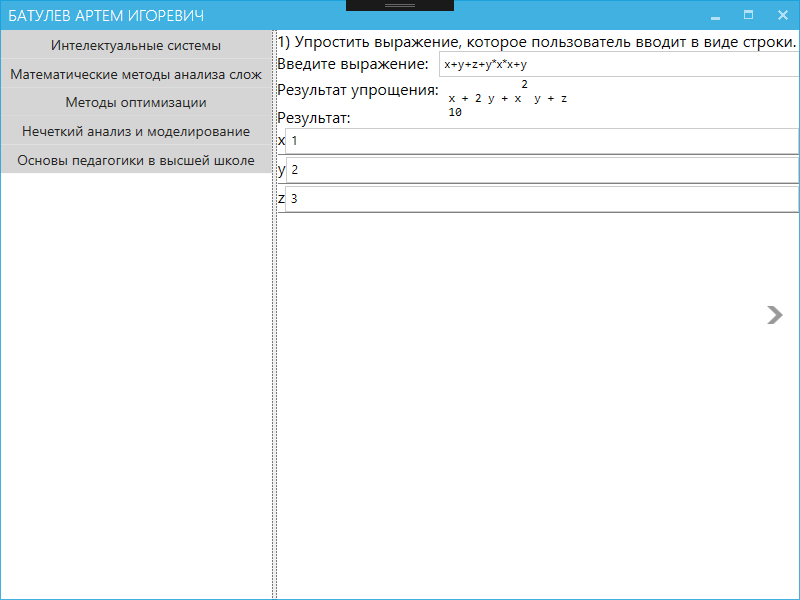


Рисунок 1 – Результат работы функции для задания 1

1. Построить график функции ошибок Erf на диапазоне заданном пользователем. График сохранить на жесткий диск в виде файла .bmp. Найти значения функции ошибок в заданных пользователем точках.

Результат представлен на рисунке 2.

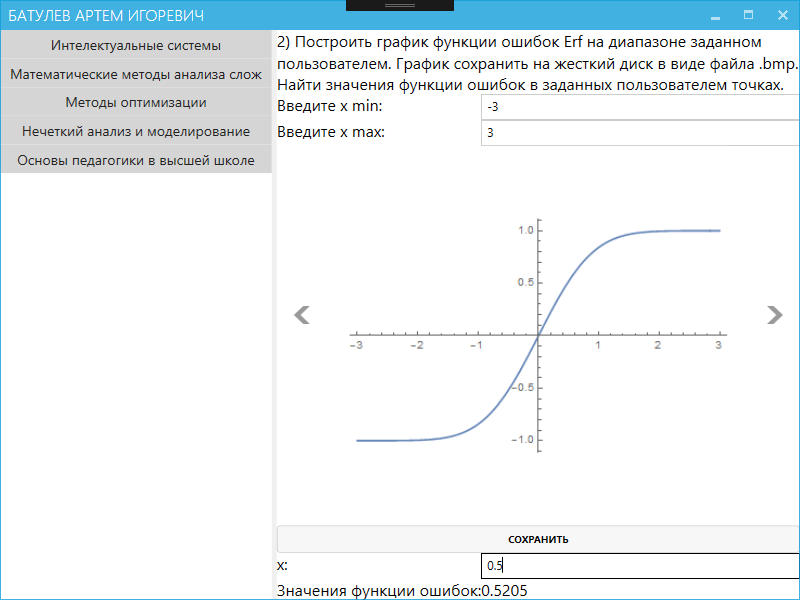


Рисунок 2 – Результат работы функции для задания 2

1. Построить график функции на диапазоне заданном пользователем. Функция вводится в виде строки. График сохранить на жесткий диск как файл .bmp.

Результат представлен на рисунке 3.

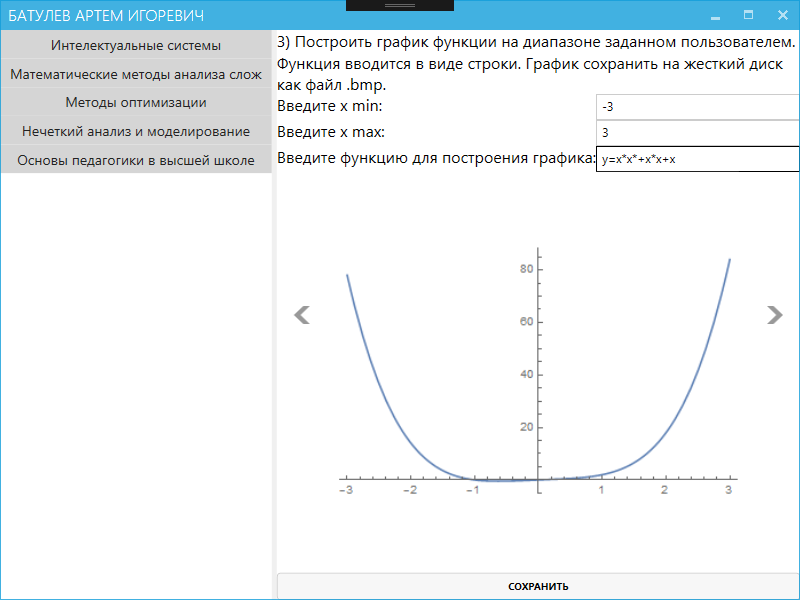


Рисунок 3 – Результат работы функции для задания 3

1. Найти решение системы уравнений



при значениях параметров a и b вводимых с клавиатуры.

Результат представлен на рисунке 4.

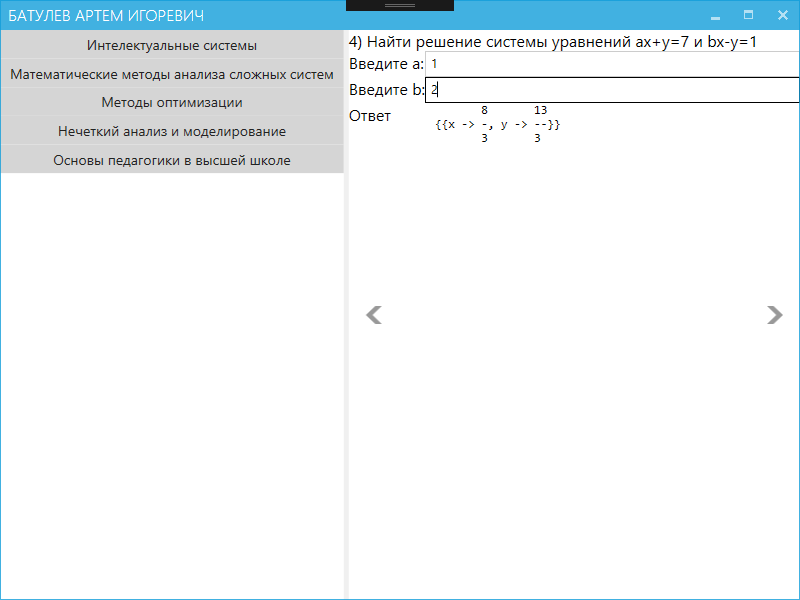


Рисунок 4 – Результат работы функции для задания 4

1. Задано уравнение с параметром *a*: *3x2-2x-a=0*. Диапазон и шаг дискретного изменения параметра задаются пользователем. Необходимо найти максимальное и минимальное решение уравнения при таком изменении *a*.

Результат представлен на рисунке 5.

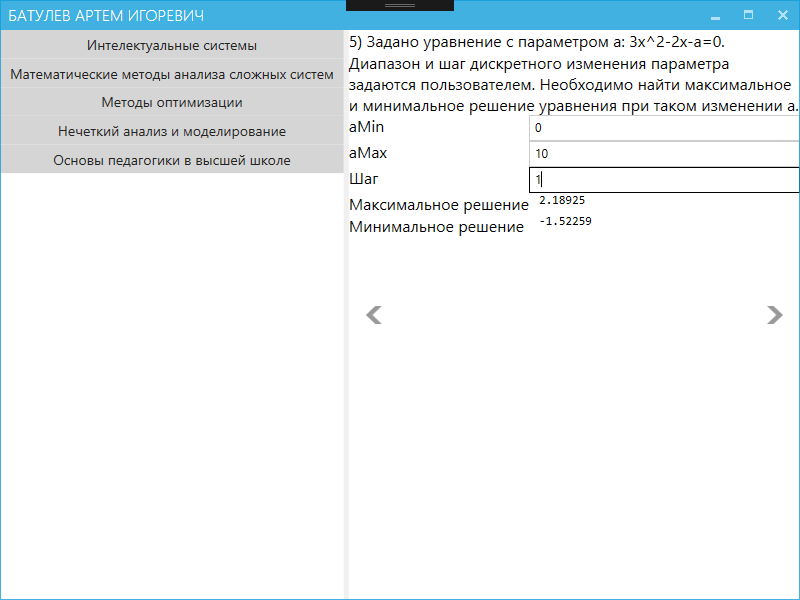


Рисунок 5 – Результат работы функции для задания 5

1. Найти решения уравнения *x2+a\*x+b=0* при значениях параметров *a* и *b* вводимых с клавиатуры.

Результат представлен на рисунке 6.

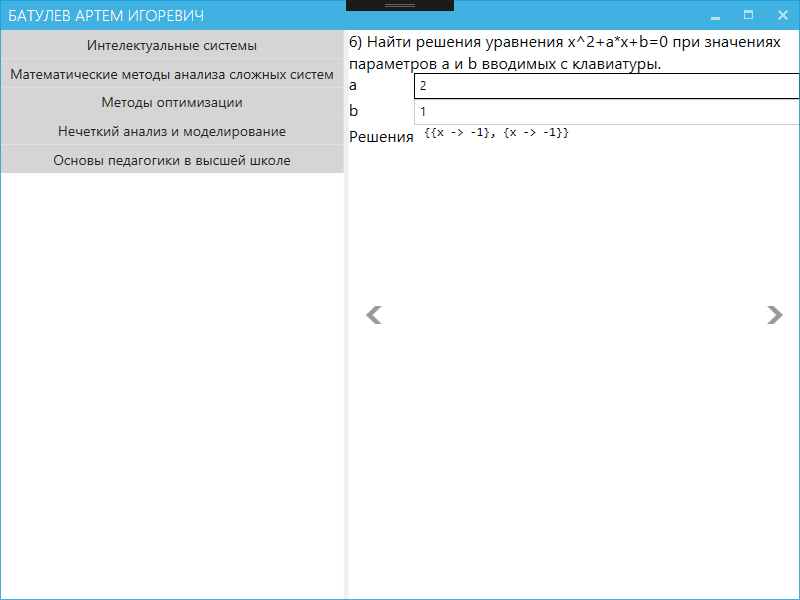


Рисунок 6 – Результат работы функции для задания 6

1. Найти максимальное и минимальное значение функции *y=3\*x3-2\*x2+7* при дискретном изменении *х*. Диапазон и шаг изменения вводится пользователем. Построить график функции на заданном диапазоне и сохранить его на диск в виде .bmp файла.

Результат представлен на рисунке 7.

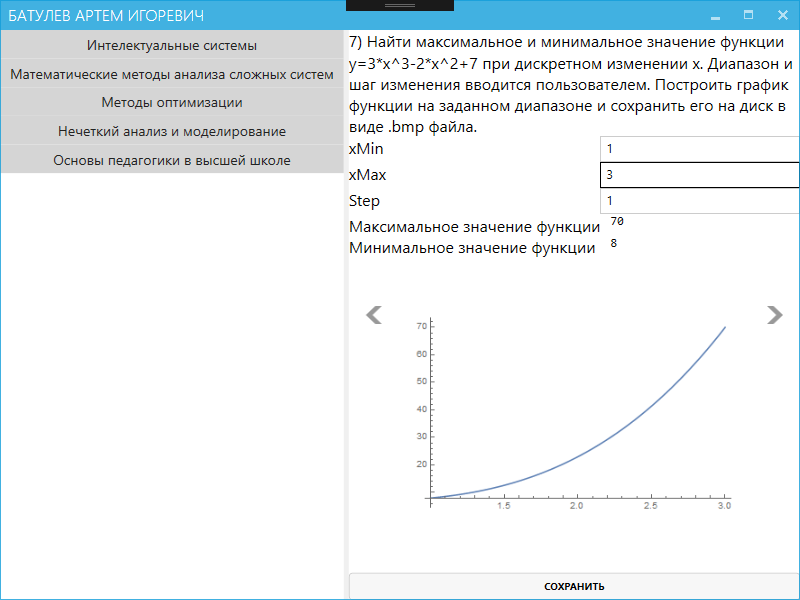


Рисунок 7 – Результат работы функции для задания 7

1. Решить задачу линейного программирования.

Результат представлен на рисунке 8.

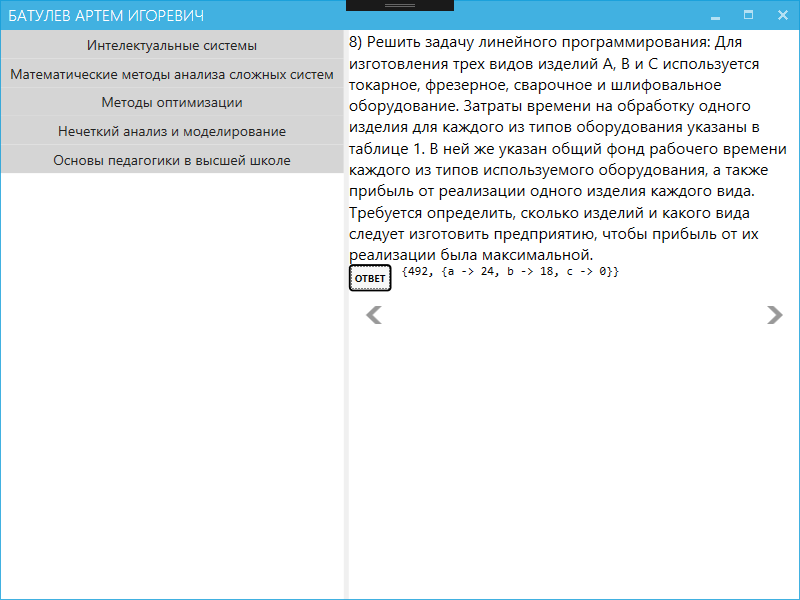


Рисунок 8 – Результат работы функции для задания 8

1. Найти решения уравнения *b\*x2+a\*x+b=0* при значениях параметров *a* и *b* вводимых с клавиатуры.

Результат представлен на рисунке 9

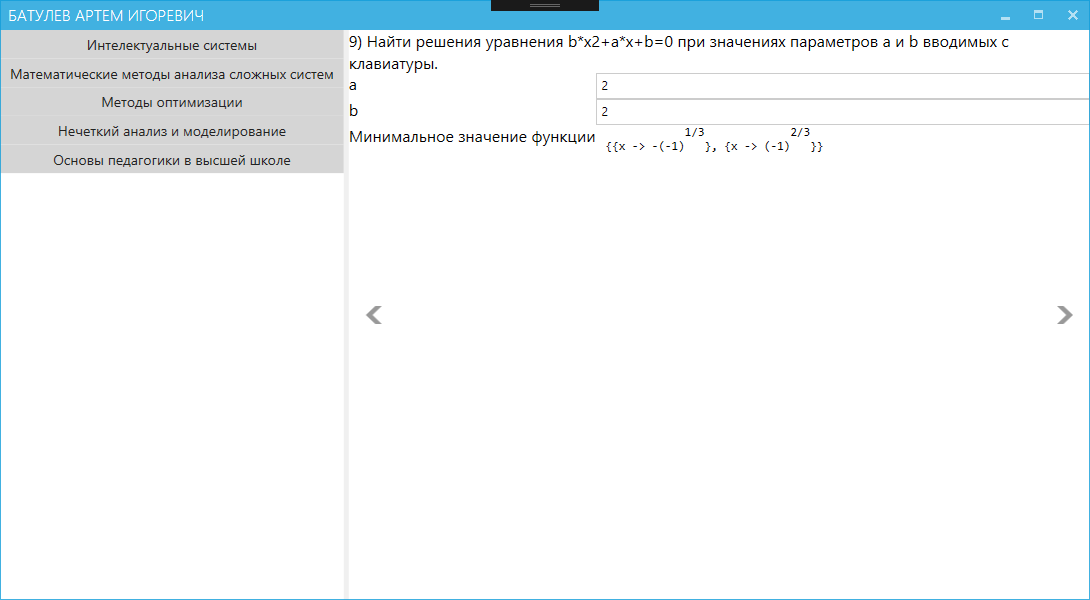


Рисунок 9 – Результат работы функции для задания 9

1. Задано уравнение с параметром *a*: . Диапазон и шаг дискретного изменения параметра задаются пользователем. Необходимо найти максимальное и минимальное решение уравнения при таком изменении *a*.

Результат представлен на рисунке 10.

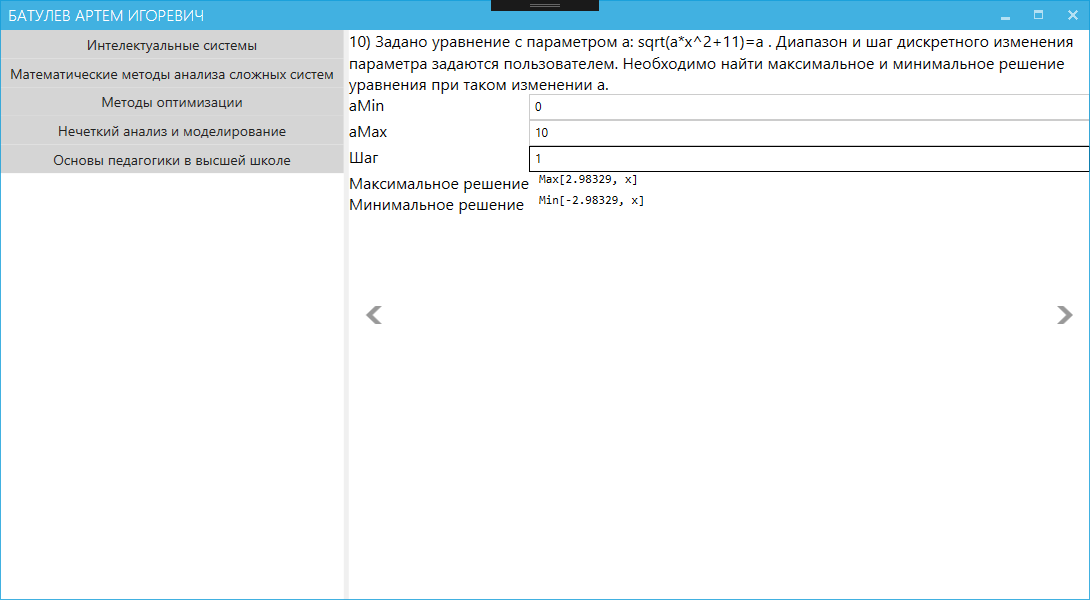


Рисунок 10 – Результат работы функции для задания 10

1. Найти максимальное и минимальное значение функции  при дискретном изменении *х*. Диапазон и шаг изменения вводится пользователем. Построить график функции на заданном диапазоне и сохранить его на диск в виде .bmp файла.

Результат представлен на рисунке 11.

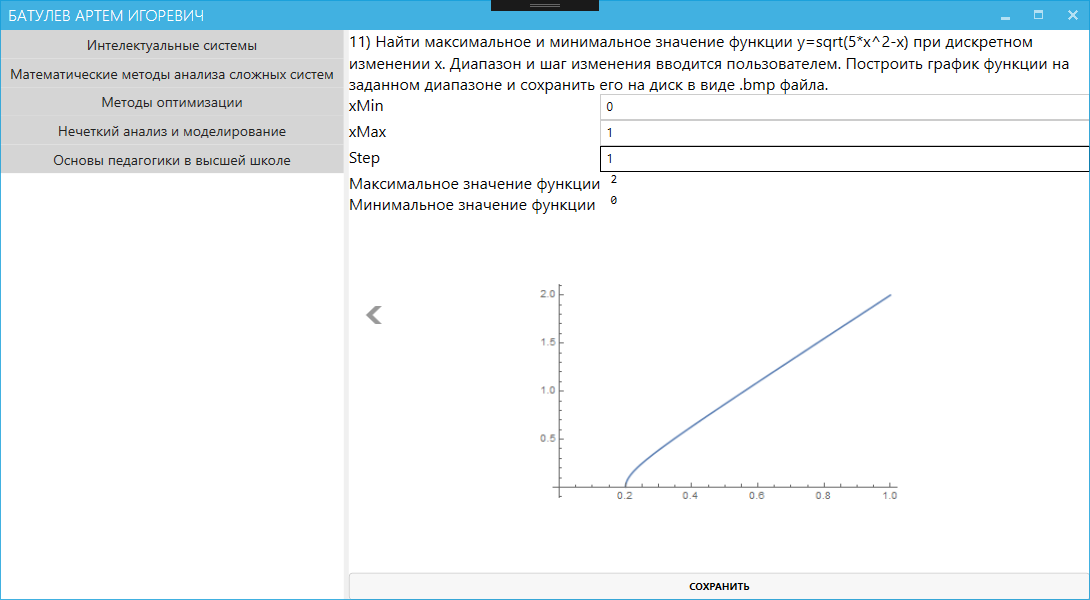


Рисунок 11 – Результат работы функции для задания 11

Приложение А

Текст программы

using System.Collections.ObjectModel;

using System.ComponentModel;

using Microsoft.Practices.ObjectBuilder2;

using Prism.Mvvm;

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job1 : BaseJob

{

private string \_in1;

private string \_job2;

public Job1(MathKernel mathKernel)

: base(mathKernel)

{

Unknown = new ObservableCollection<Point>();

}

public ObservableCollection<Point> Unknown { get; set; }

private void Evaluate(object obj, PropertyChangedEventArgs e)

{

\_job2 = \_in1;

foreach (var point in Unknown)

{

if (!string.IsNullOrWhiteSpace(point.Value))

{

\_job2 = \_job2.Replace(point.Name, point.Value);

}

}

OnPropertyChanged(nameof(Result2));

}

public string In1

{

get { return \_in1; }

set

{

\_in1 = value;

\_job2 = string.Empty;

Unknown.Clear();

GetUnknownVariables(\_in1).ForEach(p =>

{

var point = new Point { Name = p };

point.PropertyChanged += Evaluate;

Unknown.Add(point);

});

OnPropertyChanged(nameof(Unknown));

OnPropertyChanged(nameof(Result1));

OnPropertyChanged(nameof(Result2));

}

}

public string Result1 => Simplify(\_in1);

public string Result2 => Simplify(\_job2);

private string Simplify(string input)

{

return Compute($"Simplify[{input}]");

}

}

public class Point : BindableBase

{

private string \_value;

public string Name { get; set; }

public string Value

{

get { return \_value; }

set { SetProperty(ref \_value, value); }

}

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job2 : ImageBaseJob

{

private string \_x1;

public Job2(MathKernel mathKernel)

: base(mathKernel)

{

}

public string X1

{

get { return \_x1; }

set

{

\_x1 = value;

OnPropertyChanged(nameof(Result));

}

}

public string Result => Compute($"Erf[{\_x1}]");

protected override string GetGraphicFunc()

{

return "Erf[x]";

}

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job3 : Job2

{

private string \_graphicFunc;

public Job3(MathKernel mathKernel)

: base(mathKernel)

{

}

public string GraphicFunc

{

get { return \_graphicFunc; }

set

{

\_graphicFunc = value;

OnPropertyChanged(nameof(Img));

}

}

protected override string GetGraphicFunc()

{

return GraphicFunc;

}

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job4 : BaseJob

{

private string \_a;

private string \_b;

public Job4(MathKernel mathKernel)

: base(mathKernel)

{

}

public string A

{

get { return \_a; }

set

{

\_a = value;

OnPropertyChanged(nameof(Result));

}

}

public string B

{

get { return \_b; }

set

{

\_b = value;

OnPropertyChanged(nameof(Result));

}

}

public string Result => Compute($"Solve[{\_a} \* x + y == 7 && {\_b} \* x - y == 1, {{x,y}}]");

}

}

using System.Linq;

using System.Text;

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job5 : BaseJob

{

private string \_step;

private string \_aMin;

private string \_aMax;

public Job5(MathKernel mathKernel)

: base(mathKernel)

{

}

public string aMin

{

get { return \_aMin; }

set

{

\_aMin = value;

OnPropertyChanged(nameof(MaxResult));

OnPropertyChanged(nameof(MinResult));

}

}

public string aMax

{

get { return \_aMax; }

set

{

\_aMax = value;

OnPropertyChanged(nameof(MaxResult));

OnPropertyChanged(nameof(MinResult));

}

}

public string Step

{

get

{

return \_step;

}

set

{

\_step = value;

OnPropertyChanged(nameof(MaxResult));

OnPropertyChanged(nameof(MinResult));

}

}

public string MaxResult => Compute($"Max[{{ {GetResults()} }}]");

public string MinResult => Compute($"Min[{{ {GetResults()} }}]");

private string GetResults()

{

var sb = new StringBuilder();

double min, max, step;

double.TryParse(\_aMin, out min);

double.TryParse(\_aMax, out max);

double.TryParse(\_step, out step);

if (min < max && step > 0)

{

for (double i = min; i <= max; i = i + step)

{

sb.Append(Compute($"N[x /.Solve[{GetEquation(i)}, x, Reals]]") + ",");

}

}

var result = sb.ToString();

if (result.Any() && result.Last() == ',')

{

result = result.Remove(result.Length - 1);

}

return result;

}

protected virtual string GetEquation(double a)

{

return $"3\*x\*x-2\*x-({a})==0";

}

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job6 : BaseJob

{

private string \_a;

private string \_b;

public Job6(MathKernel mathKernel)

: base(mathKernel)

{

}

public string A

{

get { return \_a; }

set

{

\_a = value;

OnPropertyChanged(nameof(Result));

}

}

public string B

{

get { return \_b; }

set

{

\_b = value;

OnPropertyChanged(nameof(Result));

}

}

public string Result => Compute($"Solve[ x^2+{\_a}\*x+{\_b}==0 ,x]");

}

}

using System.Linq;

using System.Text;

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job7 : ImageBaseJob

{

private string \_step;

public Job7(MathKernel mathKernel)

: base(mathKernel)

{

PropertyChanged += (o, e) =>

{

if (e.PropertyName == nameof(Img))

{

UpdateView();

}

};

}

private void UpdateView()

{

OnPropertyChanged(nameof(MaxResult));

OnPropertyChanged(nameof(MinResult));

}

public string Step

{

get { return \_step; }

set

{

\_step = value;

UpdateView();

}

}

public string MaxResult => Compute($"Max[{{ {GetResults()} }}]");

public string MinResult => Compute($"Min[{{ {GetResults()} }}]");

protected override string GetGraphicFunc()

{

return "y=3\*x^3-2\*x^2+7";

}

private string GetResults()

{

var sb = new StringBuilder();

double min, max, step;

double.TryParse(\_xMin, out min);

double.TryParse(\_xMax, out max);

double.TryParse(\_step, out step);

if (min < max && step > 0)

{

for (double x = min; x <= max; x = x + step)

{

sb.Append(Compute(GetEquation(x)) + ",");

}

}

var result = sb.ToString();

if (result.Any() && result.Last() == ',')

{

result = result.Remove(result.Length - 1);

}

return result;

}

protected virtual string GetEquation(double x)

{

return $"3\*({x})^3-2\*({x})^2+7";

}

}

}

using Prism.Commands;

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job8 : BaseJob

{

public Job8(MathKernel mathKernel)

: base(mathKernel)

{

EvaluateCommand = new DelegateCommand(Evaluate);

}

public DelegateCommand EvaluateCommand { get; set; }

private void Evaluate()

{

OnPropertyChanged(nameof(Result));

}

public string Result => Compute("Maximize[ { 10\*a + 14\*b + 12\*c , 2\*a + 4\*b + 5\*c <= 120 && a + 8\*b + 6\*c <= 280 && 7\*a + 4\*b + 5\*c <= 240 && 4\*a + 6\*b + 7\*c <= 360 && a >= 0 && b >= 0 && c >= 0 }, { a, b, c }]");

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job9 : BaseJob

{

private string \_a;

private string \_b;

public Job9(MathKernel mathKernel)

: base(mathKernel)

{

}

public string A

{

get { return \_a; }

set

{

\_a = value;

OnPropertyChanged(nameof(Result));

}

}

public string B

{

get { return \_b; }

set

{

\_b = value;

OnPropertyChanged(nameof(Result));

}

}

public string Result => Compute($"Solve[{\_b}\*x^2+({\_a})\*x+({\_b})==0 ,x]");

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job10 : Job5

{

public Job10(MathKernel mathKernel)

: base(mathKernel)

{

}

protected override string GetEquation(double a)

{

return $"Sqrt[{a}\*x^2+11]=={a}";

}

}

}

using Wolfram.NETLink;

namespace OptimizationMethods.ViewModels.Lab1

{

public class Job11 : Job7

{

public Job11(MathKernel mathKernel)

: base(mathKernel)

{

}

protected override string GetGraphicFunc()

{

return "y=Sqrt[5\*x^2-x]";

}

protected override string GetEquation(double x)

{

return $"Sqrt[5\*{x}^2-{x}]";

}

}

}