Министерство образования Республики Беларусь

Министерство образования и науки Российской Федерации

МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Высшая математика»

**Курсовой проект**

**по дисциплине «Оптимизация проектных решений**

**на тему «Решение матричных игр методом Брауна – Робинсона»**

|  |  |  |
| --- | --- | --- |
| Выполнил: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_2022 г. | студент гр. ИСИТ-191  Шамигов М.А. |
| Руководитель: | \_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_»\_\_\_\_\_\_\_\_\_2022 г. | ст. преподаватель  Бондарев А. Н. |

Могилев 2022 г.

СОДЕРЖАНИЕ

[**Введение** 3](#_Toc122457535)

[**1.Постановка задачи** 4](#_Toc122457536)

[**2.Основные положения метода Брауна-Робинсон** 5](#_Toc122457537)

[**3.Программная реализация системы на ЭВМ** 10](#_Toc122457538)

[**3.1 Общие сведения о разработанном приложении** 10](#_Toc122457539)

[**Заключение** 12](#_Toc122457540)

[**Список использованных источников** 13](#_Toc122457541)

[**Приложение А** 14](#_Toc122457542)

# **Введение**

Курсовой проект по дисциплине «Оптимизация проектных решений» является работой, представляющей собой решение учебной прикладной задачи, основанное на применении оптимизационных моделей и методов в различных областях науки и техники.

Целью выполнения курсового проекта является систематизация, закрепление и углубление теоретических знаний и практических навыков, приобретенных при изучении дисциплины «Оптимизация проектных решений», формирование способности к самостоятельному решению прикладных задач, формирование общепрофессиональных и профессиональных компетен­ций, развитие творческих способностей.

В процессе выполнения курсового проекта решаются следующие задачи:

* создание программного обеспечения, позволяющего решать задачи, установленным заданием методом.
* самостоятельное изучение учебной и научной литературы по теме проек­та, поиск необходимой информации;
* обоснованный выбор, доработка и применение для постановки   
  и реше­ния задачи оптимизационных методов и моделей;
* выбор и применение для решения задачи аналитических и научных паке­тов прикладных программ;
* проверка адекватности моделей, анализ результатов, оценка их качества и надежности.

В качестве темы курсового проекта выбрана «Решение матричных игр методом Брауна-Робинсона».

В ходе выполнения курсового проекта необходимо познакомится с научно-исследовательской литературой, и практическими навыками решения задач, а также сформировать свое отношение к прикладной научной проблеме   
и к методам ее практического решения.

**1.Постановка задачи**

Необходимо разработать программу на языке программирования высокого уровня для решения матричных игр методом Брауна-Робинсона.

Задачей оптимизации в математике называется задача нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

Математическое программирование — это область математики, разрабатывающая теорию, численные методы решения многомерных задач   
с ограничениями.

В процессе проектирования обычно ставится задача определения наилучших, в некотором смысле, структуры или значений параметров объектов. Такая задача называется оптимизационной. Если оптимизация связана с расчётом оптимальных значений параметров при заданной структуре объекта, то она называется параметрической оптимизацией. Задача выбора оптимальной структуры является структурной оптимизацией.

# **2.Основные положения метода Брауна-Робинсон**

В методе Брауна-Робинсон рассматривается последовательность фиктивных разыгрываний с заданной платежной матрицей. Будем называть каждое такое разыгрывание *партией*. В первой партии игроки произвольно выбирают свои чистые стратегии. В последующих партиях каждый из игроков предполагает, что противник использует смешанную стратегию, которая определяется относительными частотами появления чистых стратегий на предыдущих шагах. На основе такого предположения игрок делает свой личный ход, выбирая чистую стратегию. Такой подход прост в реализации и моделирует накопление игроками опыта при повторении игровой ситуации.

Пусть парная матричная игра с нулевой суммой задана платежной матрицей

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *В*1 | *В*2 | … | *Вn* |
| *А*1 | *a*11 | *a*12 | … | *a*1*n* |
| *А*2 | *a*21 | *a*22 | … | *a*2*n* |
| ... | … | … | … | … |
| *Аm* | *am*1 | *am*2 | … | *amn* |

Представим, что разыграны первые *k* партий. При этом первый игрок использовал свои чистые стратегии с относительными частотами, которые представлены в таблице1.

Таблица 1.

|  |  |  |  |
| --- | --- | --- | --- |
| *А*1 | *А*2 | ... | *Аm* |
| *p*1*k* | *p*2*k* | ... | *pmk* |

Второй игрок использовал свои чистые стратегии с относительными частотами, которые представлены в табл.2.

Таблица 2.

|  |  |  |  |
| --- | --- | --- | --- |
| *В*1 | *В*2 | … | *Вn* |
| *q*1*k* | *q*2*k* | ... | *qnk* |

Для относительных частот в табл.1,2 выполняются условия

Эти условия соответствуют требованиям для компонент смешанных  
стратегий.

Далее получим оценки:

Тогда в партии с номером k+1 на основании оценок первый игрок  
выбирает стратегию А, второй игрок – стратегию Вβ.

Относительные частоты, которые приведены в табл.1,2, определяют  
смешанные стратегии игроков в первых k партиях. В силу этого  
выполняется неравенство

, *(1)*

где ν – точное значение цены игры. Условие (1) приводит к оценке  
цены игры вида

В 1951 г. Дж.Робинсон доказала, что

*(2)*

то есть итерации сходятся к точному значению цены игры. Это позволяет прекратить фиктивные разыгрывания, когда величина

*(3)*

В литературе традиционно отмечается малая скорость сходимости, а  
также немонотонность последовательностей и . Также имеет  
место замедление сходимости итераций при увеличении размеров  
платежной матрицы. Однако реализация метода на современных  
компьютерах делает указанные недостатки метода Брауна-Робинсон не  
такими значительными. В то же время простота и наглядность процедуры  
фиктивного разыгрывания делают метод привлекательным, в том числе  
для учебных целей.

***Пример.*** Решить матричную игру с нулевой суммой, если платежная матрица имеет вид

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *В*1 | *В*2 | *В*3 | *В*4 | *В*5 |
| *А*1 | 3 | 7 | 1 | -2 | 2 |
| *А*2 | 2 | -5 | -4 | 0 | 2 |
| *А*3 | 1 | 6 | -3 | -5 | -1 |

Будем считать, что в первой партии игрок А воспользуется стратегией *А*1, которая является максимальной для данной матричной игры. В этом случае игрок В может получить следующие проигрыши: 3 при своей чистой стратегии *В*1, 7 – при *В*2, 1 – при *В*3, -2 – при *В*4, 2 – при *В*5. Тогда в первой партии игрок В, предполагая, что игрок А выберет стратегию *А*1, с целью минимизации своего проигрыша применит стратегию *В*4.

Приближенное значение цены игры будем находить по формуле

*(4)*

По итогам первой партии определяем относительные частоты, с которыми игроки использовали свои стратегии,

*p*11 =1, *p*21 = 0, *p*31 = 0,

*q*11 = 0, *q*21 = 0, *q*31 = 0, *q*41 =1, *q*51 = 0.

Применив формулы, находим при α = 2, β = 3 оценки ν1(1) = -2, ν2(1) = 0. Приближенная цена игры после первого разыгрывания ν∗ =−1.

В соответствии с найденными значениями параметров α и β во второй партии игрок А воспользуется стратегией *А*2, а игрок В – стратегией *В*3. Относительные частоты применения стратегий после двух партий составят

*p*12 = 0,5, *p*22 = 0,5, *p*32 = 0,

*q*12 = 0, *q*22 = 0, *q*32 = 0,5, *q*42 = 0,5, *q*52 = 0.

Подсчитываем оценки: ν1(2) =−1,5, ν2(2) =−0,5. Эти оценки были получены при α=1, β= 4. По результатам двух партий приближенная цена игры равна

Переходим к третьей партии. В ней игрок А выберет стратегию *А*1, игрок В – стратегию *В*4. Пересчитаем относительные частоты применения чистых стратегий:

*p*13 = 0,667, *p*23 = 0,333, *p*33 = 0,

*q*13 = 0, *q*23 = 0, *q*33 = 0,333, *q*43 = 0,667, *q*53 = 0.

Снова определяем оценки ν1(3) =−1,333, ν2(3) =−1, при этом α=1, β= 4. Находим приближенную цену игры

Фиктивные разыгрывания продолжаем до тех пор, пока величина ∆ в формуле (3) превышает точность вычисления цены игры. В нашем примере для вычисления цены игры положим точность 0,01.

Результаты вычислений, полученных на каждой итерации при фиктивном разыгрывании, представлены в таблице 3.

Таблица 3.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер партии  *k* | Игрок А | | | | Игрок В | | | | | | ν1(*k*) | ν2(*k*) | ν\* |
| *А*α | относительные частоты  применения чистых стратегий | | | *В*β | относительные частоты  применения чистых стратегий | | | | |
| *p*1*k* | *p*2*k* | *p*3*k* | *q*1*k* | *q*2*k* | *q*3*k* | *q*4*k* | *q*5*k* |
| 1 | *А*1 | 1 | 0 | 0 | *В*4 | 0 | 0 | 0 | 1 | 0 | -2 | 0 | -1 |
| 2 | *А*2 | 0,5 | 0,5 | 0 | *В*3 | 0 | 0 | 0,5 | 0,5 | 0 | -1,5 | -0,5 | -1 |
| 3 | *А*1 | 0,667 | 0,333 | 0 | *В*4 | 0 | 0 | 0,333 | 0,667 | 0 | -1,333 | -1 | -1,167 |
| 4 | *А*1 | 0,75 | 0,25 | 0 | *В*4 | 0 | 0 | 0,25 | 0,75 | 0 | -1,5 | -1 | -1,167 |
| 5 | *А*2 | 0,6 | 0,4 | 0 | *В*4 | 0 | 0 | 0,2 | 0,8 | 0 | -1,2 | -0,8 | -1,1 |
| 6 | *А*2 | 0,5 | 0,5 | 0 | *В*3 | 0 | 0 | 0,333 | 0,667 | 0 | -1,5 | -1 | -1,1 |
| 7 | *А*1 | 0,571 | 0,428 | 0 | *В*3 | 0 | 0 | 0,429 | 0,571 | 0 | -1,143 | -0,714 | -1,071 |
| 8 | *А*1 | 0,625 | 0,375 | 0 | *В*4 | 0 | 0 | 0,375 | 0,625 | 0 | -1,25 | -0,875 | -1,071 |
| 9 | *А*1 | 0,667 | 0,333 | 0 | *В*4 | 0 | 0 | 0,333 | 0,667 | 0 | -1,333 | -1 | -1,071 |
| 10 | *А*1 | 0,7 | 0,3 | 0 | *В*4 | 0 | 0 | 0,3 | 0,7 | 0 | -1,4 | -1,1 | -1,121 |
| 11 | *А*1 | 0,727 | 0,273 | 0 | *В*4 | 0 | 0 | 0,273 | 0,727 | 0 | -1,455 | -1,091 | -1,121 |
| 12 | *А*2 | 0,667 | 0,333 | 0 | *В*4 | 0 | 0 | 0,25 | 0,75 | 0 | -1,333 | -1 | -1,121 |
| 13 | *А*2 | 0,615 | 0,385 | 0 | *В*4 | 0 | 0 | 0,231 | 0,769 | 0 | -1,231 | -0,923 | -1,121 |
| 14 | *А*2 | 0,571 | 0,429 | 0 | *В*3 | 0 | 0 | 0,286 | 0,714 | 0 | -1,143 | -1,143 | -1,143 |

Итерационный процесс завершился после 14 партий. Приближенное значение цены игры будет равно ν∗ =−1,143.

# **3.Программная реализация системы на ЭВМ**

## **3.1 Общие сведения о разработанном приложении**

Программа разработана в среде разработки Visual Studio, на высокоуровневом языке программирования С#.

С# для разработки приложения выбран как наиболее распространенный   
и популярный в наше время.

Простота и естественность языка C#, ориентация системы на разработку приложений, его высокая эффективность (большая производительность   
и относительно небольшие размеры) создаваемых с ее помощью программ сделали C# незаменимым средством разработки различного рода приложений.

Интерфейс программы является интуитивно понятным. (Рисунок 3.1)

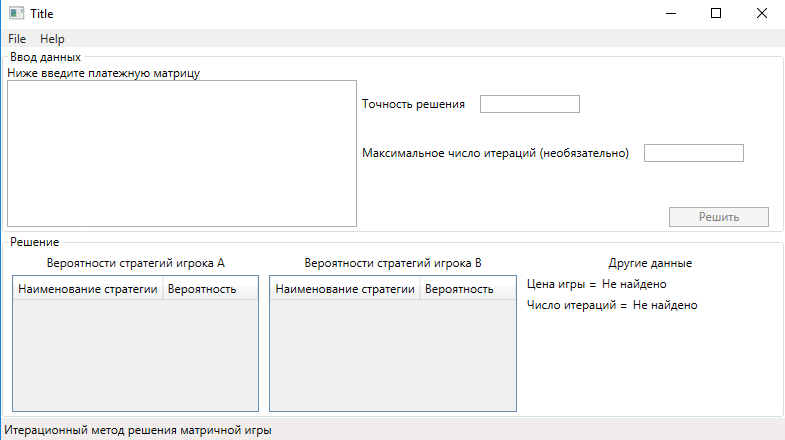


Рисунок 3.1 – Интерфейс программы

Для того, чтобы программа выполнила вычисления, необходимо ввести платежную матрицу, точность решения, и при необходимости ввести максимальное число итераций. (Рисунок 3.2)

После ввода данных необходимо нажать кнопку “Решить”. (Рисунок 3.3)

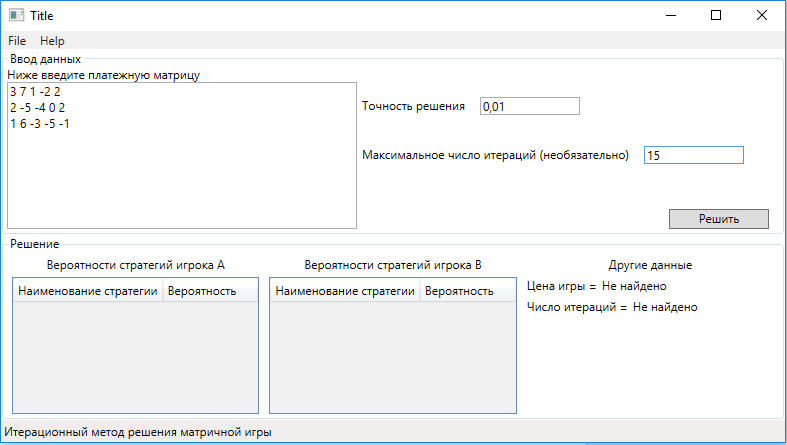


Рисунок 3.2 – Ввод данных

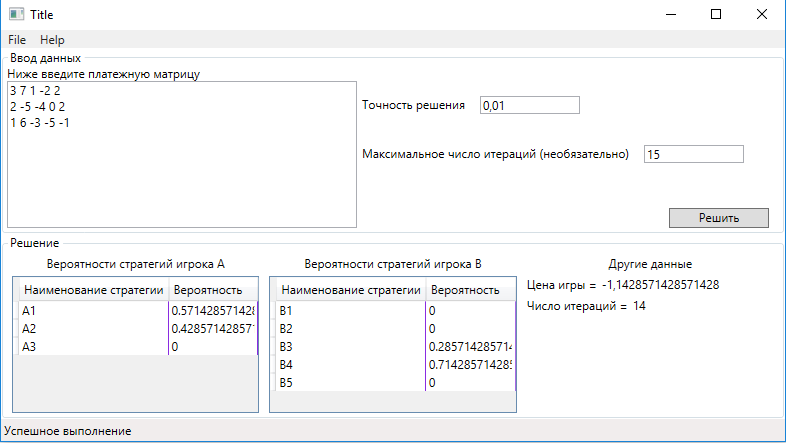


Рисунок 3.3 – Результат вычислений

Приложение выполняет подсчеты быстро, без зависаний. При этом не было обнаружено никаких ошибок в ходе работы программы.

# **Заключение**

В ходе написания курсового проекта, были проанализированы научно-литературные источники, часть из которых – позволила автоматизировать решение матричных игр методом Брауна-Робинсона, для оперативного решения поставленных задач.

Комплекс полученных знаний, удалось реализовать в виде программного обеспечения, имеющего интуитивно понятный интерфейс. Приложение ориентированного на абсолютно любого пользователя.

Оптимизация в широком смысле слова находит применение в науке, технике и в любой другой области человеческой деятельности.

Программное обеспечение соответствует изначально поставленным целям, и выполняет все условия задачи. Может быть применено как отдельное приложения для поиска экстремума функции, так и в составе более сложных программных комплексов.

# **Список использованных источников**

1 **Петросян Л.А.** Теория игр: Учебник / Л.А.Петросян, Н.А.Зенкевич, Е.В.Шевкопляс, 2012. – 704 с.

2 **Поляк, Б.Т.** Введение в оптимизацию. М.: Наука. 2003. С. – 301 с.

3 **Сухарев, А.Г.** Курс методов оптимизации. М: Физмалит. 2005. – 45 с.

4 **Боровков, А. А.** Математическая статистика: учебник для вузов / А. А. Боровков. – Санкт-Петербург : Лань, 2021. – 476 с.

# **Приложение А**

**Код программы**

**MainWindowViewModel.cs**

using MatrixGameSolver.Infrastructure.Commands;

using MatrixGameSolver.Model;

using MatrixGameSolver.Model.Data;

using MatrixGameSolver.ViewModels.Base;

using System;

using System.Collections.ObjectModel;

using System.Windows.Input;

using System.Windows.Markup;

namespace MatrixGameSolver.ViewModels

{

[MarkupExtensionReturnType(typeof(MainWindowViewModel))]

public class MainWindowViewModel : ViewModel

{

public MainWindowViewModel()

{

GetSolutionCommand = new LambdaCommand(OnGetSolutionCommandExecuted, CanGetSolutionCommandExecute);

}

#region Properties

private string \_title = "Title";

public string Title { get => \_title; set => Set(ref \_title, value); }

private string \_status = "Итерационный метод решения матричной игры";

public string Status { get => \_status; set => Set(ref \_status, value); }

private string \_paymentMatrix;

public string PaymentMatrix { get => \_paymentMatrix; set => Set(ref \_paymentMatrix, value); }

private string \_precision;

public string Precision { get => \_precision; set => Set(ref \_precision, value); }

private string \_maxStepsCount;

public string MaxStepsCount { get => \_maxStepsCount; set => Set(ref \_maxStepsCount, value); }

public ObservableCollection<StrategyLikelihood> TableA { get; set; } = new ObservableCollection<StrategyLikelihood>();

public ObservableCollection<StrategyLikelihood> TableB { get; set; } = new ObservableCollection<StrategyLikelihood>();

private string \_gamePrice = "Не найдено";

public string GamePrice { get => \_gamePrice; set => Set(ref \_gamePrice, value); }

private string \_stepsCount = "Не найдено";

public string StepsCount { get => \_stepsCount; set => Set(ref \_stepsCount, value); }

#endregion

#region Commands

public ICommand GetSolutionCommand { get; }

private void OnGetSolutionCommandExecuted(object p)

{

try

{

TableA.Clear();

TableB.Clear();

GamePrice = "Не найдено";

StepsCount = "Не найдено";

double precision = Convert.ToDouble(Precision);

int maxStepsCount = int.MaxValue;

if (!string.IsNullOrWhiteSpace(MaxStepsCount))

{

maxStepsCount = Convert.ToInt32(MaxStepsCount);

}

double[][] paymentMatrix = GetEquivalentMatrix(PaymentMatrix);

IterativeMethod method = new IterativeMethod(paymentMatrix, precision, maxStepsCount);

var answer = method.Solve();

GamePrice = answer.GamePrice.ToString();

StepsCount = answer.IterationsCount.ToString();

for (int i = 0; i < answer.LikelihoodsA.Count; i++)

{

TableA.Add(answer.LikelihoodsA[i]);

}

for (int i = 0; i < answer.LikelihoodsB.Count; i++)

{

TableB.Add(answer.LikelihoodsB[i]);

}

Status = "Успешное выполнение";

}

catch (Exception e)

{

Status = $"Операция не удалась. Причина {e.Message}";

}

}

private bool CanGetSolutionCommandExecute(object p) => !(string.IsNullOrWhiteSpace(PaymentMatrix) || string.IsNullOrWhiteSpace(Precision));

#endregion

private double[][] GetEquivalentMatrix(string matrStr)

{

double[][] matrix;

string[] rows = matrStr.Split("\r\n", StringSplitOptions.RemoveEmptyEntries);

string[] numbers;

matrix = new double[rows.Length][];

for (int i = 0; i < rows.Length; i++)

{

numbers = rows[i].Split(" ", StringSplitOptions.RemoveEmptyEntries);

matrix[i] = new double[numbers.Length];

for (int j = 0; j < numbers.Length; j++)

{

matrix[i][j] = Convert.ToDouble(numbers[j]);

}

}

return matrix;

}

}

}

**IterativeMethod.cs**

using MatrixGameSolver.Model.Data;

using System.Collections.Generic;

using System.Linq;

namespace MatrixGameSolver.Model

{

public class IterativeMethod

{

private int \_maxStepsCount;

private double \_precision;

private double[][] \_matrix;

public IterativeMethod(double[][] matrix, double precision, int maxStepsCount = int.MaxValue)

{

\_precision = precision;

\_maxStepsCount = maxStepsCount;

\_matrix = matrix;

}

public IterativeMethodAnswer Solve()

{

int currentIteration = 0;

int currentAStrategy = FindRowWithMaxElement();

int[] playerAStrategiesUsed = new int[\_matrix.GetLength(0)];

double[] accumulatedWinningsOfPlayerA = new double[\_matrix[0].Length]; // выигрыши игрока А при конкрентной стратегии, размерность - число столбцов в матрице (стратки В)

int currentBStrategy;

int[] playerBStrategiesUsed = new int[\_matrix[0].Length];

double[] accumulatedLossOfPlayerB = new double[\_matrix.GetLength(0)]; // проигрыши игрока В при конкретной стратегии, размерность - число строк в матрице (стратки А)

double minGamePrice = 0, maxGamePrice = 0;

double currentGamePrice = double.MaxValue;

while (!IsPrecisionAchieved(minGamePrice, maxGamePrice, currentGamePrice) && (currentIteration < \_maxStepsCount))

{

currentIteration++;

playerAStrategiesUsed[currentAStrategy]++;

accumulatedWinningsOfPlayerA.AddSpecifiedRow(\_matrix[currentAStrategy]);

minGamePrice = accumulatedWinningsOfPlayerA.Min();

currentBStrategy = accumulatedWinningsOfPlayerA.IndexOf(minGamePrice);

playerBStrategiesUsed[currentBStrategy]++;

accumulatedLossOfPlayerB.AddSpecifiedColumnOfMatrix(\_matrix, currentBStrategy);

maxGamePrice = accumulatedLossOfPlayerB.Max();

currentAStrategy = accumulatedLossOfPlayerB.IndexOf(maxGamePrice);

minGamePrice /= currentIteration;

maxGamePrice /= currentIteration;

currentGamePrice = (minGamePrice + maxGamePrice) / 2;

}

IterativeMethodAnswer answer = new IterativeMethodAnswer(currentIteration, currentGamePrice);

for (int i = 0; i < playerAStrategiesUsed.Length; i++)

{

answer.LikelihoodsA.Add(new StrategyLikelihood($"A{i + 1}", ((double)playerAStrategiesUsed[i]) / currentIteration));

}

for (int i = 0; i < playerBStrategiesUsed.Length; i++)

{

answer.LikelihoodsB.Add(new StrategyLikelihood($"B{i + 1}", ((double)playerBStrategiesUsed[i]) / currentIteration));

}

return answer;

}

private bool IsPrecisionAchieved(double minGamePrice, double maxGamePrice, double currentGamePrice) => ((currentGamePrice - minGamePrice) <= \_precision) && ((maxGamePrice - currentGamePrice) <= \_precision);

private int FindRowWithMaxElement()

{

int index = 0;

int currentIndexOfMaxElementInRow = FindIndexOfMaxElementInSpecifiedRow(0);

double max = \_matrix[index][currentIndexOfMaxElementInRow];

for (int i = 1; i < \_matrix.GetLength(0); i++)

{

currentIndexOfMaxElementInRow = FindIndexOfMaxElementInSpecifiedRow(i);

if (\_matrix[i][currentIndexOfMaxElementInRow] > max)

{

max = \_matrix[i][currentIndexOfMaxElementInRow];

index = i;

}

}

return index;

}

private int FindIndexOfMaxElementInSpecifiedRow(int rowIndex)

{

double max = \_matrix[rowIndex][0];

int index = 0;

for (int i = 1; i < \_matrix[rowIndex].Length; i++)

{

if (\_matrix[rowIndex][i] > max)

{

max = \_matrix[rowIndex][i];

index = i;

}

}

return index;

}

}

public class IterativeMethodAnswer

{

public int IterationsCount { get; }

public double GamePrice { get; }

public List<StrategyLikelihood> LikelihoodsA { get; set; }

public List<StrategyLikelihood> LikelihoodsB { get; set; }

public IterativeMethodAnswer(int iterationsCount, double gamePrice)

{

IterationsCount = iterationsCount;

GamePrice = gamePrice;

LikelihoodsA = new List<StrategyLikelihood>();

LikelihoodsB = new List<StrategyLikelihood>();

}

}

}

**DoubleMatrixExtensions.cs**

using System;

using System.Collections.Generic;

using System.Text;

namespace MatrixGameSolver.Model

{

public static class DoubleMatrixExtensions

{

public static int IndexOf(this double[] arr, double element)

{

for (int i = 0; i < arr.Length; i++)

{

if (arr[i] == element)

return i;

}

return -1;

}

public static void AddSpecifiedRow(this double[] arr, double[] row)

{

int size = GetMinSize(arr, row);

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

{

arr[i] += row[i];

}

}

public static void AddSpecifiedColumnOfMatrix(this double[] arr, double[][] matrix, int colIndex)

{

int size;

if (arr.Length < matrix.GetLength(0))

size = arr.Length;

else

size = matrix.GetLength(0);

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

{

arr[i] += matrix[i][colIndex];

}

}

private static int GetMinSize(double[] first, double[] second)

{

int size;

if (first.Length < second.Length)

size = first.Length;

else

size = second.Length;

return size;

}

}

}

**MainWindow.xaml**

<Window x:Class="MatrixGameSolver.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:MatrixGameSolver"

mc:Ignorable="d"

Title="{Binding Title}" Height="450" Width="800"

DataContext="{Binding MainWindowViewModel, Source={StaticResource Locator}}">

<Window.Resources>

<Style TargetType="DataGrid">

<Setter Property="AutoGenerateColumns" Value="False"/>

<Setter Property="CanUserAddRows" Value="False"/>

<Setter Property="CanUserDeleteRows" Value="False"/>

<Setter Property="CanUserSortColumns" Value="False"/>

<Setter Property="VerticalGridLinesBrush" Value="BlueViolet"/>

<Setter Property="GridLinesVisibility" Value="Vertical"/>

</Style>

</Window.Resources>

<DockPanel>

<Menu DockPanel.Dock="Top">

<MenuItem Header="File"/>

<MenuItem Header="Help"/>

</Menu>

<StatusBar DockPanel.Dock="Bottom">

<StatusBarItem Content="{Binding Status}"/>

</StatusBar>

<Grid>

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<GroupBox Grid.Row="0" Header="Ввод данных">

<DockPanel>

<TextBlock Text="Ниже введите платежную матрицу" DockPanel.Dock="Top"/>

<TextBox TextWrapping="Wrap" AcceptsReturn="True" MinWidth="350" Text="{Binding PaymentMatrix, UpdateSourceTrigger=PropertyChanged}"/>

<Grid Margin="5,0">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<StackPanel Orientation="Horizontal" Grid.Row="0" VerticalAlignment="Center">

<TextBlock Text="Точность решения"/>

<TextBox Width="100" Margin="15,0" Text="{Binding Precision, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Grid.Row="1" VerticalAlignment="Center">

<TextBlock Text="Максимальное число итераций (необязательно)"/>

<TextBox Width="100" Margin="15,0" Text="{Binding MaxStepsCount, UpdateSourceTrigger=PropertyChanged}"/>

</StackPanel>

<Button Grid.Row="2" HorizontalAlignment="Right" VerticalAlignment="Bottom" Content="Решить"

Margin="5,0" Width="100" Command="{Binding GetSolutionCommand}"/>

</Grid>

</DockPanel>

</GroupBox>

<GroupBox Grid.Row="1" Header="Решение">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<DockPanel Grid.Column="0" Margin="5,0">

<TextBlock DockPanel.Dock="Top" Text="Вероятности стратегий игрока А" Margin="0,5" HorizontalAlignment="Center"/>

<DataGrid ItemsSource="{Binding TableA}">

<DataGrid.Columns>

<DataGridTextColumn Header="Наименование стратегии" Binding="{Binding Name}"/>

<DataGridTextColumn Header="Вероятность" Width="\*" Binding="{Binding Probability}"/>

</DataGrid.Columns>

</DataGrid>

</DockPanel>

<DockPanel Grid.Column="1" Margin="5,0">

<TextBlock DockPanel.Dock="Top" Text="Вероятности стратегий игрока B" Margin="0,5" HorizontalAlignment="Center"/>

<DataGrid ItemsSource="{Binding TableB}">

<DataGrid.Columns>

<DataGridTextColumn Header="Наименование стратегии" Binding="{Binding Name}"/>

</DataGrid>

</DockPanel>

<DockPanel Grid.Column="2" Margin="5,0">

<TextBlock DockPanel.Dock="Top" Text="Другие данные" Margin="0,5" HorizontalAlignment="Center"/>

<StackPanel>

<StackPanel Orientation="Horizontal">

<TextBlock Text="Цена игры ="/>

<TextBlock Margin="5,0" Text="{Binding GamePrice}"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,5">

<TextBlock Text="Число итераций ="/>

<TextBlock Margin="5,0" Text="{Binding StepsCount}"/>

</StackPanel>

</StackPanel>

</DockPanel>

</Grid>

</GroupBox>

</Grid>

</DockPanel>

</Window>