МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль)/специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Методы поиска равновесия Нэша в смешанных стратегиях

Студент Е. Д. Евстигнеев
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Н.А. Сосина
(ученая степень, звание, И.О. Фамилия)

Консультант Е.В. Косс
(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Аннотация

Тема бакалаврской работы: «Методы поиска равновесия Нэша в смешанных стратегиях».

Актуальность данной работы заключается в том, что с каждым годом математическое моделирование на основе теории игр растет и находит все большее применение в различных сферах: социологии, политологии, маркетинге, логистике и т.д.

Объект исследования: игровые методы оптимизации.

Предмет исследования: равновесие Нэша.

Цель работы: анализ и реализация алгоритмов нахождения равновесия Нэша в смешанных стратегиях.

Структура выпускной квалификационной работы представлена введением, тремя главами, заключением, списком литературы.

Во введении кратко описываются актуальность, цели и задачи данного исследования.

В первой главе работы рассматриваются история возникновения и задачи решаемые методом поиска равновесия Нэша.

Во второй главе описываются методы и алгоритмы поиска равновесия Нэша.

В третьей главе описывается процесс реализации и тестирование реализованного в работе метода поиска равновесия Нэша. На основе результатов тестирования формулируются выводы.

Бакалаврская работа выполнена на 47 страницах, содержит 21 рисунок и 4 таблицы.

Abstract

The title of the graduation work is «Methods for finding Nash equilibrium in mixed strategies».

The relevance of this work lies in the fact that every year the need for mathematics, as well as game theory in the fields of sociology, political science, marketing and others, is getting bigger.

The object of the study is the game optimization methods.

The subject of the study is the Nash equilibrium.

The aim of this work is the analysis and implementation of the algorithms for finding Nash equilibrium in mixed strategies.

The structure of the graduation work is represented by an introduction, three chapters, a conclusion, and a list of references.

The introduction briefly describes the relevance, goals and objectives of this study.

In the first chapter of the paper, the history of the origin and the problems solved by the Nash equilibrium search method are considered.

The second chapter describes the methods and algorithms for finding the Nash equilibrium.

The third chapter describes the process of implementing and testing the Nash equilibrium search method implemented in the work. Based on the test results, the conclusion is drawn.

The graduation work consists of an explanatory note on 46 pages, introduction, three parts including 21 illustrations, 4 tables, the list of 35 references including 5 foreign sources.

Содержание

Введение

В выпускной квалификационной работе рассматривается равновесие по Нэшу в смешанных стратегиях.

Актуальность работы обусловлена тем, что равновесие Нэша является одним из главных инструментов в маркетинге и экономике, на его основе решаются задачи по неорганизованным рынкам, теоретико-игровым моделям олигополистической конкуренции и многие другие.

Объектом исследования являются игровые методы оптимизации.

Предметом исследования является равновесие Нэша.

Целью работы является анализ и реализация равновесия Нэша в смешанных стратегиях.

Для достижения поставленных целей необходимо решить следующие задачи:

- проанализировать методы поиска равновесия Нэша в смешанных стратегиях,
- выполнить анализ технологий проектирования методов для поиска равновесия Нэша в смешанных стратегиях,
- выполнить программную реализацию метода поиска равновесия Нэша в смешанных стратегиях,
- протестировать программную реализацию методов поиска равновесия Нэша в смешанных стратегиях.

В первой главе рассматривается история возникновения методов поиска равновесия Нэша в смешанных стратегиях, а также представлено решение задач с применением данных методов.

Во второй главе рассматриваются математические методы и алгоритмы решения задач по поиску равновесия Нэша в смешанных стратегиях.

В третьей главе выполнена программная реализация методов равновесия Нэша в смешанных стратегиях. Программа протестирована на основе задач, решенных во второй главе.

1 История возникновения метода поиска равновесия Нэша и задачи, решаемые этим методом

1.1 История возникновения метода

Жизнь — это выживание, а выживание — это соревнование. Всякий раз, когда два или более объекта сталкиваются с ограничениями, такими как фиксированное количество ресурсов, в сценарии выигрыш/проигрыш возникает конкуренция. Это неизбежно и естественно. Это эволюционно, о чем свидетельствует баланс пищевых цепей в экосистемах — конечно, это врожденное поведение в равной степени распространяется и на людей. Из конкуренции, в любой из ее многочисленных форм, вытекает стратегия.

Теория игр — это раздел прикладной математики, используемый для создания оптимальной стратегии для достижения успеха в конкурентных ситуациях неопределенности и неполных знаний (как и в большинстве сценариев реальной жизни). Это математическое исследование принятия решений и моделирования в конфликтных ситуациях, которые встречаются в повседневной жизни во всех отраслях и дисциплинах [1].

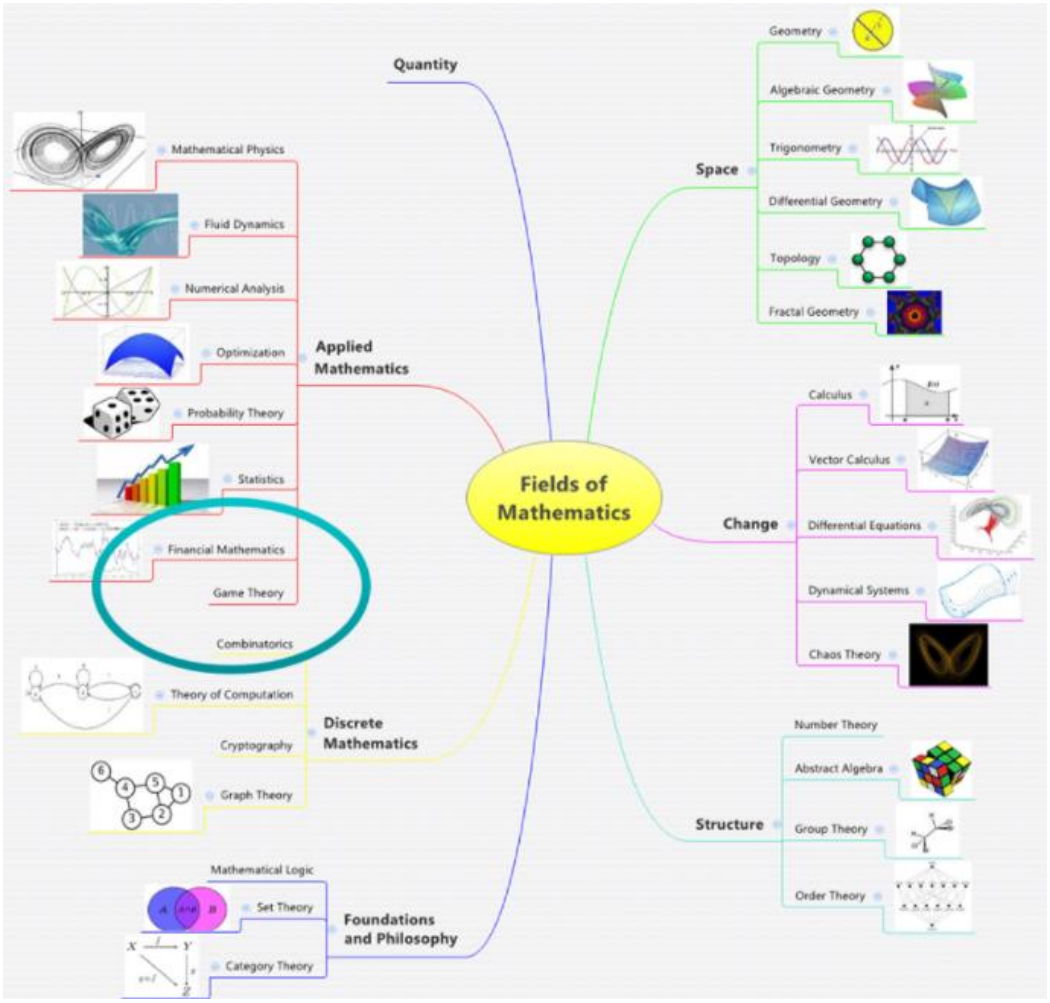

Рисунок 1 – Место Теории игр в областях математики

Страны воюют за территорию, предприятия борются за долю рынка, животные борются за ресурсы, а политические партии борются за голоса. В мире, где правят взаимозависимые агенты, стремящиеся увеличить свою «ценность» в динамических системах, теория игр чрезвычайно уместна.

В то время как академическая дисциплина и сама область были официально установлены только в 1950-х годах, идеи, подобные теории игр, ясно видны в многовековой истории.

Рассмотрим предысторию как древних дней, так и поступательное ускорение 1950-х годов [2].

1.2 Задачи, решаемые методом поиска равновесия Нэша

Идеи, лежащие в основе теории игр, неоднократно появлялись на протяжении всей истории. От военных размышлений Сунь-Цзы до эволюционных открытий Чарльза Дарвина, идеи, на которых основывается теория игр долгое время существовали как движущая сила человеческого поведения. Однако основу древней теории игр обычно приписывают популярности следующих трех конкретных работ:

- Огюстен Курно «Исследования о математических основаниях теории богатства»;
- Фрэнсис Исидро Эджворт «Математическая психология»;
- Эмиль Борель «Алгебра и исчисление вероятностей».

Каждый из этих классиков кратко заглянул в будущее теории игр; каждый из них стоял на краю знаний (своего времени) и пытался использовать математику для отображения человеческого поведения в некотором контексте (например, в богатстве). Хотя опять же, как подробно описано выше, все это было лишь шепотом дисциплины, двигавшейся в тени — эта область не получила официального признания до 1950-х годов [3].

Хотя многие авторы внесли свой вклад в историю теории игр, широко признано, что современный анализ начался с Джона фон Неймана, а методологическую основу ему в дальнейшем предоставил Джон Нэш.

Джон Форбс Нэш-младший был американским математиком, жившим с 1928 по 2015 год. Он был исследователем в Принстонском университете. Его работа была в основном в области теории игр, в которую он внес значительный важный вклад. В 1994 году он получил Нобелевскую премию по экономике за применение теории игр в экономике. Равновесие Нэша является частью всей теории равновесия, предложенной Нэшем [4].

Вполне вероятно, что теория игр не существовала до тех пор, пока Джон фон Нейман не опубликовал статью «Теория игр стратегии» (1928). Хотя это и не его opus magnum, эта статья значительно продвинула вперед

область — ее значение лучше всего можно понять, рассмотрев предложенную фундаментальную теорему теории игр, содержащуюся в ней.

Фундаментальная теорема теории игр утверждает, что в широкой категории игр для двух лиц всегда можно найти равновесие, от которого ни один из игроков не должен отклоняться в одностороннем порядке. Такое равновесие существует в любой игре двух лиц, удовлетворяющей следующим критериям:

- игра конечна,
- игра – это игра с полной информацией,
- у игры нулевая сумма.

Известная сегодня как теорема минимакса, она до сих пор чрезвычайно актуальна и является ключевой частью любого курса теории игр. Однако следующая публикация Неймана была и остается святым Граалем теории игр.

«Теория игр и экономическое поведение», опубликованная в 1944 году Джоном фон Нейманом и экономистом Оскаром Моргенштерном, считается новаторским текстом, который официально утвердил теорию игр как междисциплинарную область исследований. Фактически, в предисловии к своему 60-летию издательство Princeton University Press назвало его «классической работой, на которой основана современная теория игр» [5].

В 1980-х годах концепция смешанных стратегий подверглась резкой критике за то, что она «интуитивно проблематична» (Роберт Ауманн, «Чего пытается достичь теория игр?» в Frontiers of Economics, Oxford: Blackwell). Рандомизация, занимающая центральное место в смешанных стратегиях, не имеет поведенческой поддержки. Редко люди делают свой выбор после лотереи. Эта поведенческая проблема усугубляется когнитивной трудностью, заключающейся в том, что люди не могут генерировать случайные результаты без помощи случайного или псевдослучайного генератора.

Теоретик игр Ариэль Рубинштейн («Комментарии к интерпретации теории игр», Econometrica, Vol. 59, № 4 (июль 1991 г.), стр. 909–924) указывает на два альтернативных способа понимания этой концепции.

Один из них состоит в том, чтобы представить, что игроки представляют собой большое количество агентов. Каждый из агентов выбирает чистую стратегию, а выигрыш зависит от доли агентов, выбирающих каждую стратегию. Таким образом, смешанная стратегия представляет собой распределение чистых стратегий, выбранных каждой популяцией. Однако это не дает никаких оснований для случая, когда игроки являются индивидуальными агентами [6].

Другой, называемый очисткой, предполагает, что интерпретация смешанных стратегий просто отражает наше незнание информации агента и процесса принятия решений. По-видимому, случайный выбор тогда рассматривается как следствие неуказанных, не имеющих отношения к выигрышу экзогенных факторов, таких как «животные духи» Кейнса.

Однако неудовлетворительно иметь результаты, которые зависят от неуказанных факторов, и это исключает возможность того, что анализ смешанных стратегий имеет какую-либо предсказательную силу. Утверждение, что эти факторы являются просто убеждениями других игроков о стратегии игрока (следовательно, использование смешанной стратегии является лучшим ответом на то, что игрок использует смешанные стратегии), дает заслуживающую доверия интерпретацию [7].

С тех пор отношение экономистов к результатам, основанным на смешанных стратегиях, было двойственным. Смешанные стратегии по-прежнему широко используются из-за их способности обеспечивать равновесие Нэша в любой игре, но модель должна указывать, почему и как игрок рандомизирует свои решения.

Влияние на экономику и бизнес.

Теория игр произвела революцию в экономике, решив важнейшие проблемы предшествующих математических экономических

моделей. Например, неоклассическая экономическая теория изо всех сил пыталась понять предпринимательское ожидание и не могла справиться с несовершенной конкуренцией. Теория игр переключила внимание с установившегося равновесия на рыночный процесс.

В бизнесе теория игр полезна для моделирования конкурирующего поведения между экономическими агентами. У предприятий часто есть несколько стратегических вариантов, которые влияют на их способность получать экономическую выгоду.

Например, предприятия могут столкнуться с такими дилеммами, как отказ от существующих продуктов или разработка новых, снижение цен по сравнению с конкурентами или применение новых маркетинговых стратегий. Экономисты часто используют теорию игр для понимания поведения олигополистических фирм. Это помогает предсказать вероятные результаты, когда фирмы участвуют в определенных действиях, таких как фиксирование цен и сговор [8]-[11].

2 Методы и алгоритмы поиска равновесия Нэша в смешанных стратегиях

2.1 Основные определения

Часто в экономике и маркетинге возникают такие ситуации, в которых две (или более) стороны имеют различные цели, и результат действия каждой из сторон зависит от действия партнера. Такие ситуации называют конфликтными.

Для решения и анализа конфликтных ситуаций было создано такое направление в математике, как «Теория игр».

Теория игр помогает выбрать лучшие стратегии с учётом представлений о других участниках, их ресурсах и их возможных поступках.

Игра – это математическая модель конфликтной ситуации.

В теории игр различают два вида ходов:

- случайный ход - выбор одного хода из ряда возможных, выбранный каким-либо механизмом случайного выбора (бросок монеты, кости);
- личный ход - рациональный выбор игрока одного из возможных. Например, любой ход в игре ГО можно назвать личным.

Стратегия игрока – совокупность правил, относительно которого игрок будет делать тот или иной личный ход, исходя из сложившейся ситуации в игре [12].

Игры делятся на два типа в зависимости от числа возможных стратегий:

- конечные - игры, где у игроков А и Б есть конечное количество стратегий $m_{\times}n$. Такие игры называют игрой $m_{\times}n$;

- бесконечные - игры, где у одного или у нескольких игроков есть неограниченное количество стратегий.

Платежная матрица – матрица элементами которой являются числа, выражающие выигрыши и проигрыши игроков. Выигрыши и проигрыши могут выражаться в пунктах, количестве денег или в других единицах.

Стратегии делятся на:

- чистую стратегию, которая даёт полную определённость, каким образом игрок продолжит игру. В частности, она определяет результат для каждого возможного выбора, который игроку, возможно, придётся сделать. Пространством стратегий называют множество всех чистых стратегий, доступных данному игроку;
- смешанную стратегию, которая является указанием вероятности каждой чистой стратегии. Это означает, что игрок выбирает одну из чистых стратегий в соответствии с вероятностями, заданными смешанной стратегией. Выбор осуществляется перед началом каждой игры и не меняется до её конца. Каждая чистая стратегия является частным случаем смешанной, когда вероятность одной из чистых стратегий равна единице, а остальных возможных чистых стратегий — нулю;
- игры также различают на несколько видов. Антагонистические, или игры с нулевой суммой. Игры, где выигрыш одного игрока равен проигрышу другого. Неантагонистические, или биматричные.

2.2 Биматричные игры

Предположим, что существует некая игра, в которой выигрыш игрока А не равен проигрышу игрока Б. Такие игры называют биматричными, очевидно, что название исходит из того, что для того, чтобы отобразить платежную таблицу, нам понадобится 2 матрицы, которые изображены на рисунке 2.

$$\begin{matrix} & B_1 & B_2 \\ A_1 & a_{11} & a_{12} \\ A_2 & a_{21} & a_{22} \end{matrix} = A, \qquad \begin{matrix} & B_1 & B_2 \\ A_1 & b_{11} & b_{12} \\ A_2 & b_{21} & b_{22} \end{matrix} = B.$$

Рисунок 2 – Биматричная визуализация платежных матриц

Один из самых известных примеров неантагонистические игры, это «Дилемма заключенного». В таких играх каждый игрок выбирает одну стратегию из имеющегося у него конечного числа стратегий и после этого получает платеж согласно определенным для каждого из игроков платежным матрицам. Так как подобная игра полностью определяется двумя платежными матрицами, такие игры называются биматричными [13]-[15].

Любая конечная бескоалиционная игра имеет по крайней мере одну ситуацию равновесия (теорема Нэша). В полной мере эта теорема распространяется, разумеется, и на биматричные игры.

Биматричные игры являются расширением класса матричных игр (моделируемые конфликтные ситуации гораздо разнообразнее), поэтому увеличиваются и сложности решения таких игр. В частности, в биматричной игре всегда существует хотя бы одно решение в смешанных стратегиях даже при наличии одного или нескольких равновесий в чистых стратегиях.

2.3 Теорема Нэша

Любая биматричная игра имеет одну равновесную ситуацию (точку равновесия) в смешанных стратегиях.

Формальное определение для n-мерных пространств

Пусть *(S, f)* — игра, где *S* — множество профилей стратегий, а f — множество профилей выигрышей. Пусть σ - *i* — профиль стратегии всех игроков, кроме игрока *i*. Когда каждый игрок $i \in \{1, \ldots, n\}$ выбирает стратегию, x_i в результате чего формируется профиль стратегии $x = (x_i, \ldots, x_n)$, игрок *i* получает выигрыш $f_i(x)$. Обратите внимание, что выигрыш

зависит от выбранного профиля стратегии, т.е. от стратегии, выбранной игроком, i а также от стратегий, выбранных всеми остальными игроками. Профиль стратегии $x^* \in S$ является равновесием по Нэшу (РН), если ни одно из отклонений в стратегии ни одного игрока не является прибыльным, т. е. если для всех i [15]. Формула приведена ниже:

$$f_i(x_i^*, x_{-i}^*) \geq f_i(x_i^*, x_{-i}^*) \forall i \qquad (1)$$

Наиболее важным свойством равновесия Нэша является то, что оно самоподдерживается. Это результат, к которому в конечном итоге должны прийти два рациональных игрока А и В в некооперативной игре. Игрок А достигает равновесия по Нэшу, применяя стратегию, которая является его лучшим ответом на стратегию, выбранную его противником, игроком В. Игра тяготеет к неизбежному исходу. Этот исход называется равновесием Нэша [30].

Хотя в интересах каждого игрока принять стратегию, определяемую равновесием Нэша, вовсе не обязательно, чтобы равновесие Нэша максимизировало совокупный выигрыш. Дилемма заключенных — классический пример этого явления [33]-[36].

Равновесие Нэша стабильно, потому что ни один игрок не может увеличить свой выигрыш, изменив свою стратегию.

Традиционные концепции теории игр давно и широко используются в количественных исследованиях явлений, происходящих в биологических и социальных системах. Парные взаимодействия между игроками обычно описываются с использованием формализма симметричных игр для двух игроков, в которых чистые стратегии могут представлять различные физические состояния, формы человеческого поведения или биологические виды [16].

Применимость методов, разработанных в статистической физике, к математическому рассмотрению макроскопического поведения этих систем

вызвала интерес физиков. Большая часть недавних усилий сосредоточена на подавлении социальных дилемм и исследовании последствий циклического доминирования [26]-[27].

В теории игр эгоистичные и умные игроки стремятся максимизировать собственный доход. В так называемых некооперативных играх игроки должны выбирать свою стратегию, не общаясь друг с другом. В этих системах равновесия по Нэшу, профили коллективной стратегии, от которых одностороннее отклонение невыгодно, рассматриваются как решения.

Одна из основных проблем теории игр связана с количеством равновесий по Нэшу, поскольку наличие двух и более решений вызывает затруднения у игроков при отсутствии связи. Эти особенности унаследованы эволюционными играми, взаимодействие которых обычно построено из симметричных игр для двух лиц.

Некоторые свойства эволюционных игр можно предсказать, просто основываясь на их равновесиях по Нэшу. Небольшие вариации параметров выигрыша обычно не вызывают существенных изменений в поведении игры, поэтому классификация возможных типов взаимодействия и их общие черты были предметом широкого круга систематических научных исследований. Например, среднее число равновесий по Нэшу было изучено Бергом в n - стратегических играх с выигрышными элементами, определяемыми случайными числами, а введение потенциальных игр положило начало анализу декомпозиции игр [17].

В простейшей некооперативной игре есть два эквивалентных игрока с двумя доступными стратегиями. Их доход, зависящий от стратегии, определяется матрицей выигрышей из четырех возможных элементов выигрыша. В этих играх поиск равновесий Нэша можно упростить, используя две особенности, а именно то, что ранг доходов не изменяется, если элементы матрицы умножаются на положительное число и сдвигаются на константу.

Следовательно, этот тип игры может быть определен двумя независимыми параметрами. Соответствующая двумерная карта возможных равновесий Нэша выделяет четыре характерных типа поведения, иллюстрируемых в целом традиционной «дилеммой заключенного». Вместо этого анализ обычно ограничивается n -стратегическими играми, определяемыми всего несколькими параметрами [18]-[20].

2.4 Решение игры 3×3

Начнем с основного определения случайной игры, когда игроки имеют конечные наборы стратегий $\mathbf{S_i}$:

Пусть $S_i = \{s_{i1}, s_{i1}, \ldots, s_im\}$ — конечное множество чистых стратегий игрока i. Определите ΔS_i как симплекс S_i, который является набором всех вероятностных распределений над S_i . Смешанная стратегия для игрока i — это элемент $\sigma_i \in S_i$, так что $\sigma_i=\{\sigma(s_{i1}), \sigma_i(s_{i2}), \ldots, \sigma_i(s_im)\}$ — распределение вероятностей по S_i, где $\sigma_i(s_i)$ — вероятность того, что игрок i сыграет s_i .

Теперь рассмотрим пример игры «камень-ножницы-бумага», в которой $S_i = \{R, P, S\}$ (для камня, ножниц и бумаги соответственно). Мы можем определить симплекс как:

$\Delta Si = \{(\sigma_i(R), \sigma_i(P), \sigma_i(S)) : \sigma_i(R), \sigma_i(P), \sigma_i(S) \geq 0, \sigma_i(R)+\sigma_i(P)+\sigma_i(S)=1\}$

Игрок i и его противники $-i$ выбирают смешанные действия. Это означает, что мнение игрока i о своих противниках не является фиксированным, а случайным. Таким образом, убеждение игрока i — это распределение вероятностей по стратегиям его противников [21].

Вера игрока i задается распределением вероятностей $\pi_i \in S_{-i}$ по стратегиям его противников. Обозначим через $\pi_i(s_{-i})$ вероятность, которую игрок i назначает своим противникам, играющим $s_{-i} \in S_{-i}$.

Например, в игре «камень-ножницы-бумага» убеждение игрока i представлено как $(\pi_i(R), \pi_i(P), \pi_i(S))$. Мы можем думать о σ^*_{-i} как о

убеждении игрока i о своих противниках, π_i, которое отражает идею о том, что игрок i не уверен в поведении своего противника [29]-[33].

Ожидаемая выплата

В чистой стратегии выигрыш очевиден. В смешанной стратегии для оценки выигрыша нам нужно заново ввести понятие ожидаемого выигрыша:

$$v_i(s_i, \sigma_{-i}) = \sum_{s_{-i} \in S_{-i}} \sigma_{-i}(s_i) v_i(s_i, s_{-i}) \quad (2)$$

Ожидаемый выигрыш игрока $\boldsymbol{i}$, когда он выбирает чистую стратегию $\mathbf{s_i \in S_i}$, а его противники выбирают смешанную стратегию $\boldsymbol{\sigma_{-i} \in \Delta S_{-i}}$

Обратите внимание, что чистая стратегия является частью смешанной стратегии:

$$v_i(s_i, \sigma_{-i}) = \sum_{s_{-i} \in S_{-i}} \sigma_{-i}(s_i) v_i(s_i, s_{-i}) =$$

$$= \sum_{s_{-i} \in S_{-i}} \left(\sum_{s_{-i} \in S_{-i}} \sigma_i(s_i) \sigma_{-i}(s_{-i}) v_i(s_i, s_{-i}) \right) \quad (3)$$

Когда игрок i выбирает смешанную стратегию $\sigma_i \in \Delta S_i$, а его противники выбирают смешанную стратегию $\sigma_{-i} \in \Delta S_{-i}$.

Пример матрицы представлен на рисунке 3:

		Player 2		
		R	P	S
Player 1	R	0, 0	−1, 1	1, −1
	P	1, −1	0, 0	−1, 1
	S	−1, 1	1, −1	0, 0

Рисунок 3 – Матрица игроков

Рассчитаем выигрыш в сценарии смешанной стратегии.

Предположим, что игрок 2 играет $\sigma_2(R) = 0{,}5$

$\sigma_2(P) = 0{,}5$

$\sigma_2(S) = 0$

Теперь мы можем вычислить ожидаемый выигрыш для игрока 1, если он выберет чистую стратегию.

$$V^1(R,\ \sigma^2) = 0{,}5 * (0) + 0{,}5 * (-1) + 0 * (1) = -0{,}5 \quad (4)$$

$$V^1(P,\ \sigma^2) = 0{,}5 * (1) + 0{,}5 * (0) + 0 * (-1) = 0{,}5 \quad (5)$$

$$\text{B}^1(S, \sigma^2) = 0{,}5 * (-1) + 0{,}5 * (1) + 0 * (0) = 0 \quad (6)$$

$$V^1(P, \sigma 2) > V^1(S, \sigma 2) > V^1(R, \sigma^2) \quad (7)$$

Учитывая смешанную стратегию игрока 2, мы видим лучший ответ на игрока 1, то есть действие P.

Теперь давайте разберемся, как концепция равновесного решения по Нэшу применяется к смешанным стратегиям. На самом деле это проще, чем кажется, мы просто заменяем профиль стратегии смешанным профилем стратегии.

Профиль смешанной стратегии $\sigma^* = (\sigma^*_1, \sigma^*_2, ..., \sigma^*n)$ является равновесием Нэша, если для каждого игрока σ^*_i является лучшим ответом на σ^*_{-i} . То есть для всех $i \in N$ $v_i(\sigma^*_i, \sigma^*_{-i}) \geq v_i(\sigma_i, \sigma^*_{-i})$. $\forall\ \sigma_i \in S_i$.

Каждая смешанная стратегия в равновесии Нэша является наилучшим ответом на все другие смешанные стратегии в этом равновесии [22]-[24].

2.5 Равновесие Нэша в игре 2×2

В своей выпускной квалификационной работе я буду реализовывать Программу для матричных игр 2×2. Для этого выведем формулы и решим пару задач.

Составим 2 платежные матрицы для биматричной игры, Припишем стратегиям A_1, A_2 (B_1, B_2) вероятности *p, 1 - p (q, 1 - q)* соответственно.

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = A\,,\ \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = B \tag{8}$$

Средний выигрыш игрока А будет равен:

$$H_A(p,q) = a_{11}pq + a_{12}P * (1-q) + \\ +a_{21}q * (1-p) + a_{22} * (1-p) * (1-q) \tag{9}$$

Средний выигрыш игрока В равен:

$$H_B(p,q) = b_{11}pq + b_{12}P * (1-q) + \\ +b_{21}q * (1-p) + b_{22} * (1-p) * (1-q) \tag{10}$$

Запишем средние выигрыши игроков *A* и *B* в более удобной форме:

$$H_A(p,q) = (a_{11} - a_{12} - a_{21} + a_{22})pq + \\ +(a_{12} - a_{22})p + (a_{21} - a_{22})q + \ a_{22} \tag{11}$$

$$H_B(p,q) = (b_{11} - b_{12} - b_{21} + b_{22})pq +$$

$$+(b_{12}-b_{22})p+(b_{21}-b_{22})q+b_{22} \quad (12)$$

В формуле положим $p = 1$:

$$H_A(1,q)=(a_{11}-a_{12}-a_{21}+a_{22})q+a_{12}+(a_{21}-a_{22})q. \quad (13)$$

В формуле положим $p = 0$:

$$H_A(0,q)=(a_{21}-a_{22})q+a_{22}. \quad (14)$$

Рассмотрим разности:

$$H_A(1,q)-H_A(1,q)=(p-1)(Cq-a) \quad (15)$$

$$H_A(1,q)-H_A(0,q)=p(Cq-a) \quad (15)$$

$$C=a_{11}+a_{22}-(a_{21}+a_{12}),\ a=a_{22}-a_{12} \quad (16)$$

В случае, если пара*(p,q)* определяет точку разности, эти разности неотрицательны:

$$H_A(p,q)-H_A(1,q)\geq 0; \quad (17)$$

$$H_A(p,q)-H_A(0,q)\geq 0. \quad (18)$$

Пара чисел (p_0, q_0) определяет равновесную ситуацию, если $H_A(p,q_0)\leq H_A(p_0,q_0)$, для всех $0\leq p\leq 1$, то есть отклонение от оптимальной стратегии одного из игроков при условии, что другой игрок придерживается своей оптимальной стратегии, уменьшает средний выигрыш игрока, отклонившегося от оптимальной стратегии. Поэтому:

$$\begin{cases}(p-1)(Cq-\alpha)\geq 0\\ p(Cq-\alpha)\geq 0\end{cases} \quad (19)$$

Из формулы № имеем:

$$H_B(p,1) = (b_{11} - b_{12} - b_{21} + b_{22})\,p + (b_{12} - b_{22})\,p + b_{21} = (q-1)(Dp - \beta) \quad (20)$$

$$H_A(p,0) = (a_{21} - a_{22})\,p + b_{22} = q((Dp - \beta) \quad (21)$$

$$\text{где } D = b_{11} + b_{22} - (b_{21} + b_{12}),\ \beta = b_{22} - b_{21} \quad (22)$$

Пара чисел (p_0, q_0) определяет равновесную ситуацию, если $H_B(p_0,q) \le H_B(p_0,q_0)$ для всех $0 \le q \le 1$, то есть отклонение от оптимальной стратегии одного из игроков при условии, что другой игрок придерживается своей оптимальной стратегии, уменьшает средний выигрыш игрока, отклонившегося от оптимальной стратегии [26]-[29]. Поэтому:

$$\begin{cases} (q-1)(Dp - \beta) \ge 0 \\ q(Dp - \beta) \ge 0 \end{cases} \quad (23)$$

По матрице *A* находим числа C = $a_{11} + a_{22} - (a_{21} + a_{12})$, a = $a_{22} + a_{12}$ и решаем систему:

$$\begin{cases} (p-1)(Cq - \alpha) \ge 0 \\ p(Cq - \alpha) \ge 0 \end{cases} \quad (24)$$

По матрице *B* находим числа D = $b_{11} + b_{22} - (b_{21} + b_{12})$, ß = $b_{22} - b_{21}$ и решаем систему:

$$\begin{cases} (q-1)(Dp - \beta) \ge 0 \\ q(Dp - \beta) \ge 0 \end{cases} \quad (25)$$

Изобразим, полученные кривые в координатах *(p, q)*. Точки пересечения этих кривых, лежащие в квадрате $0 \le p \le 1,\ 0 \le q \le 1$ определяют

равновесные ситуации. Для каждой равновесной ситуации находят средние выигрыши $H_A(p,q)$ и $H_B(p,q)$.

2.6 Примеры биматричной игры 2×2

Решим 2×2 биматричную игру. На рисунке 4 приведены матрица А и матрица В.

$$\begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix} = A, \begin{pmatrix} 2 & 6 \\ 8 & 2 \end{pmatrix} = B.$$

Рисунок 4 – Матрица А и матрица В для решения биматричной игры

Воспользуемся алгоритмом нахождения равновесной ситуации:

$$C = 6 + 4 - (2 + 2) = 6,\ a = 4 - 2 = 2 \quad (26)$$

Решаем систему:

$$\begin{cases} (p-1)(6q-2) \geq 0 \\ p(6q-2) \geq 0 \end{cases} \quad (27)$$

Случай 1. p=1. Тогда q ≥ 1/3.

Случай 2. p=0. Тогда q ≤ 1/3.

Случай 3. 0 < p < 1. Тогда q = 1/3.

Наглядно это решение представлено на рисунке 5.

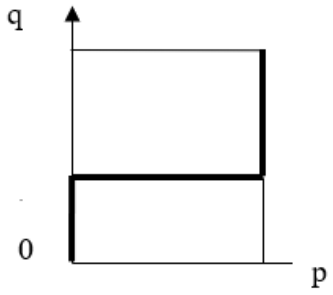

Рисунок 5 – первая часть решения биматричной игры

Продолжим решение:

$$D = 2 + 2 - (8 + 6) = -10,\ \beta = 2 - 8 = -6. \tag{28}$$

Решаем систему:

$$\begin{cases} (q-1)(-10p+6) \geq 0 \\ q(-10p+6) \geq 0 \end{cases} \tag{29}$$

Случай 1. q=1. Тогда p ≤ 3/5.

Случай 2. q=0. Тогда p ≥ 3/5.

Случай 3. 0 < q < 1. Тогда p = 3/5.

Это решение представлено на рисунке 6.

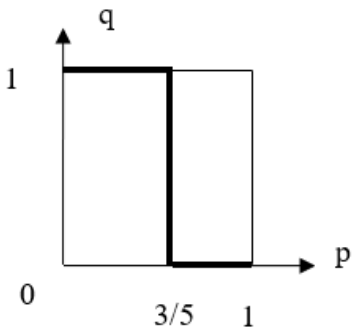

Рисунок 6 – вторая часть решения биматричной игры

Совместим оба графика для того, чтобы получить решение всей задачи на рисунке 7.

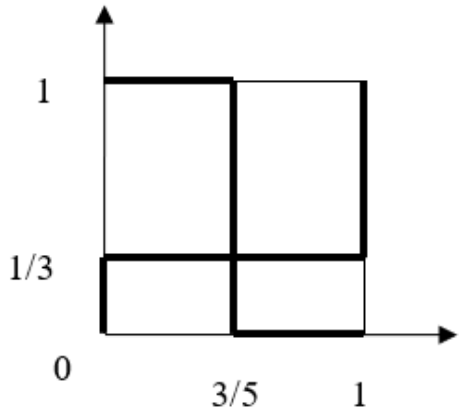

Рисунок 7 – Совмещение двух графиков для решения биматричной задачи

Получилась одна точка пересечения: p = 3/5, q = 1/3, то есть одна равновесная ситуация. 1 – p = 2/5, 1 – q = 2/3. Запись в виде матриц приведена на рисунке 8.

$$\begin{matrix} & \begin{matrix} 1/3 & 2/3 \end{matrix} \\ \begin{matrix} 3/5 \\ 2/5 \end{matrix} & \begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix} \end{matrix}, \qquad \begin{matrix} & \begin{matrix} 1/3 & 2/3 \end{matrix} \\ \begin{matrix} 3/5 \\ 2/5 \end{matrix} & \begin{pmatrix} 2 & 6 \\ 8 & 2 \end{pmatrix} \end{matrix},$$

Рисунок 8 – Равновесная ситуация

Средний выигрыш игрока A равен:

$$H_A\left(\frac{3}{5},\frac{1}{3}\right) = 6 * \frac{3}{5} * \frac{1}{3} + 2 * \frac{3}{5} * \frac{2}{3} + 2 * \frac{2}{5} * \frac{1}{3} + 4 * \frac{2}{5} * \frac{2}{3} = \frac{10}{3} \qquad (30)$$

Средний выигрыш игрока B равен:

$$H_B\left(\frac{3}{5},\frac{1}{3}\right)=2*\frac{3}{5}*\frac{1}{3}+6*\frac{3}{5}*\frac{2}{3}+8*\frac{2}{5}*\frac{1}{3}+2*\frac{2}{5}*\frac{2}{3}=\frac{66}{15} \quad (31)$$

Решим еще одну биматричную игровую задачу. Она носит название «Дилемма узника».

Два узника находятся в предварительном заключении по подозрению в совершении преступления. При отсутствии прямых улик возможность их осуждения в большей степени зависит от того, заговорят они или будут молчать. Если оба будут молчать, то наказанием будет лишь срок предварительного заключения 1 год.

Если оба сознаются, то получат срок, учитывающий признание как смягчающее обстоятельство, оба узник получат по 6 лет. Если заговорит только один из узников, а другой будет молчать, то заговоривший будет выпущен на свободу, а сохранивший молчание получит максимальный срок – 9 лет [25].

Конфликтная ситуация приводит к биматричной игре, в которой каждый из игроков имеет по две стратегии – молчать и говорить.

Таблица 1 – Платёжная матрица игрока А

	B_M	$B_Г$
A_M	-1	-9
$A_Г$	0	-6

Также укажем матрицу игрока В.

Таблица 2 – Платёжная матрица игрока В

	B_M	$B_Г$
A_M	-1	0

$A_Г$	-9	-6

Воспользуемся алгоритмом нахождения равновесной ситуации:

$$C = 2, D = 2, a = 3, ß = 3 \quad (32)$$

Подставим значения:

$$\begin{cases}(p-1)(2q-3)\geq 0\\ p(2q-3)\geq 0\end{cases} \quad \begin{cases}(q-1)(2p-3)\geq 0\\ q(2p-3)\geq 0\end{cases}. \quad (33)$$

Получим:

$$p=1, q\geq 3/2; \qquad p=0, q\leq 3/2; \qquad 0\leq p\leq 1, q=3/2.$$

(34)

$$q=1, p\geq 3/2; \qquad q=0, p\leq 3/2; \qquad 0\leq q\leq 1, p=3/2.$$

(35)

График, соответствующий решению, изображен на рисунке 9.

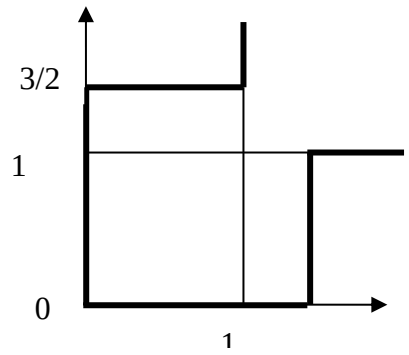

Рисунок 9 – Решение задачи Дилемма узника

Единственная равновесная ситуация – (0;0). Это ситуация, в которой каждый из игроков выбирает вторую чистую стратегию – сознаться и его потери при этом составят 6 лет.

Эта задача еще раз демонстрирует, что отклонение от ситуации равновесия одного из игроков не дает ему никаких преимуществ. Но при

одновременном отклонении обоих, каждый из игроков может получить больший выигрыш, нежели в равновесной ситуации. В данной задаче в ситуации, когда оба игрока молчат, выигрыш у каждого из игроков выше, чем в равновесной ситуации – потерять один год. Ситуация, когда оба игрока молчат неустойчива. По условию задачи сговор (создание коалиций) по условию задачи не допустим и любой из узников, изменяя стратегию молчать при условии, что другой молчит, может избежать наказания [26]-[28].

Практическое применение этого очевидно: придумывая значения соответствующим образом, власти могут сделать так, чтобы подозреваемый признался, а не сотрудничал со своими сообщниками.

Упомянутое выше сокращение количества релевантных параметров можно интерпретировать как своего рода эквивалентность между играми. Эту концепцию можно расширить, признав, что две игры могут быть преобразованы друг в друга путем переименования стратегий. Этот тип изоморфизма сужает диапазон возможного поведения.

Игрок может выбрать рандомизацию между несколькими своими чистыми стратегиями. Есть много интересных применений такого поведения, когда игрок выбирает действия стохастически (т.е. вместо выбора одной стратегии игрок выбирает распределение стратегий). Вероятность выбора любой из чистых стратегий неотрицательна, и сумма вероятностей выбора любых событий всех чистых стратегий должна равняться единице. Мы также внимательно проследим применимость равновесия Нэша к этим смешанным стратегиям [29].

2.7 Актуальность равновесий Нэша в реальном мире

Актуальность равновесий Нэша и сверхрациональности в реальном мире.

Дилемма заключенного может показаться забавным, но не слишком уместным мысленным экспериментом, но это не так: многие реальные

проблемы возникают из-за того, что люди не могут координировать свои действия и застревают в субоптимальном равновесии Нэша. Учитывайте пробки [30]-[33].

Пробки

Павел должен решить, ехать ли на работу на автобусе или на своей машине. Ехать на автобусе лучше в том смысле, что, если все сядут на автобус, пробок не будет. Но в любой конкретной ситуации ехать на машине всегда быстрее, чем на автобусе. Подумайте об этом: если все остальные сядут на машину, пробки будут всегда, вне зависимости от того, едет Павел на своей машине или нет; он все равно может попасть в пробку, так что он может взять свою машину и не тратить лишнее время на дорогу до автобусных остановок и обратно. С другой стороны, если все остальные сядут на автобус, пробок не будет, и, опять же, на машине будет быстрее. Эта ситуация равносильна дилемме заключенного, если мы заменим «Сесть на автобус» на «Молчать», «Взять машину» на «Предательство» и «Все остальные» на «Друг» (рисунок 10):

		You	
		Take the car	Take the bus
Everyone else	Take the car	1 1	0 3
	Take the bus	3 0	2 2

Рисунок 10 - Платежная матрица для задачи о пробках

Так что да, взять машину здесь — основной вариант. И если все так делают, возникают пробки, и всем становится хуже, чем если бы они все вместо этого поехали на автобусе.

Что, если бы Павел и все остальные были сверхрациональны и знали бы это друг о друге? Тогда на самом деле есть только два варианта: либо все едут на машине, либо все едут на автобусе. Последнее явно лучший вариант для Павла; поэтому в сверхрациональном случае он едет на автобусе.

Изменение климата

Аналогичная проблема возникает с борьбой с изменением климата. Павел заботится об окружающей среде и нашем будущем, но также любит жить роскошно. Эти двое не идут рука об руку. (Здесь я упрощаю, но в этом есть доля правды.) Чтобы внести свой вклад в борьбу с изменением климата, Павел должен выбрасывать меньше углекислого газа, живя менее роскошно. Но выполнение его части на самом деле не изменит ситуацию к лучшему, если никто другой не сделает свою часть; и если все остальные сделают свою часть, то плохие последствия изменения климата будут остановлены (если возможно) с поддержкой Павла или без нее. Опять же, кажется, что роскошная жизнь является доминирующей стратегией [34]-[35].

Холодная война

Во время холодной войны Соединенные Штаты и Советский Союз создавали все больше и больше ядерного оружия в рамках гонки ядерных вооружений. На каждом этапе у обеих стран в основном было два варианта: вооружиться или разоружиться.

Давайте представим, что мы находимся на определенном этапе холодной войны. Если обе страны разоружатся, разрушительная ядерная война станет менее вероятной, и обе страны потеряют меньше денег на создание и обслуживание ядерного оружия.

Но наличие большего количества оружия, чем у вашего противника, дает стратегическое преимущество — или, по крайней мере, кажется, что оно дает вам такое преимущество.

В этом смысле лучший ответ на разоружение вашего противника — это вооружиться: таким образом у вас будет больше оружия, чем у вашего противника. И если ваш противник вооружается, то вы должны вооружиться, если хотите не отставать, запись представлена на рисунке 11.

		United States	
		Arm	Disarm
Soviet Union	Arm	1 1	0 3
	Disarm	3 0	2 2

Рисунок 11 - Холодная война как дилемма заключенного

На приведенной выше диаграмме мы можем увидеть основные результаты гонки ядерных вооружений в период холодной войны. (Вы можете возразить, что разоружение также дает вам небольшую отдачу, поскольку приводит к меньшим расходам на ядерное оружие. Но здесь важны относительные отдачи, а не точные цифры. Мы могли бы добавить 1 очко к каждой отдаче, и ситуация была бы в точности одинаковый.)

Равновесие Нэша, конечно, когда обе страны вооружаются. Если подумать, это глупо. Скажем, обе страны в какой-то момент имеют по 50 ядерных боеголовок. Теперь они оба решили сделать еще 5 ядерных бомб, и в итоге у каждого по 55 ядерных бомб.

Стратегически мало что изменилось для каждой страны: у них по-прежнему одинаковое количество ядерного оружия. Они все еще могут нанести безумный урон друг другу. Но они тратят больше денег на ядерное оружие.

Конечно, если бы обе страны были сверхрациональны (и знали бы это друг о друге), они предпочли бы разоружиться. В этом сценарии на самом деле есть только два исхода: обе страны вооружаются или обе разоружаются. Из этих двух исходов каждая страна предпочитает второй.

Сверхрациональность может помочь решить большие проблемы, которые зависят от (масштабной) координации, например, изменение климата. Это ситуации, когда люди просто не могут решать и действовать в одиночку, потому что, когда так делают все, результат может быть катастрофическим.

Теперь при полученных решениях задач и при Выведенных формулах можно переходить к написанию программного кода.

3 Реализация и тестирование методов поиска равновесия Нэша в смешанных стратегиях

3.1 Описание алгоритма

Изучив необходимую теорию, приступим к реализации программы для поиска равновесия Нэша в смешанных стратегиях.

Сначала разработаем блок-схему. Она поможет нам лучше представить структуру программы и избежать недочетов. Блок-схема изображена на рисунке 12.

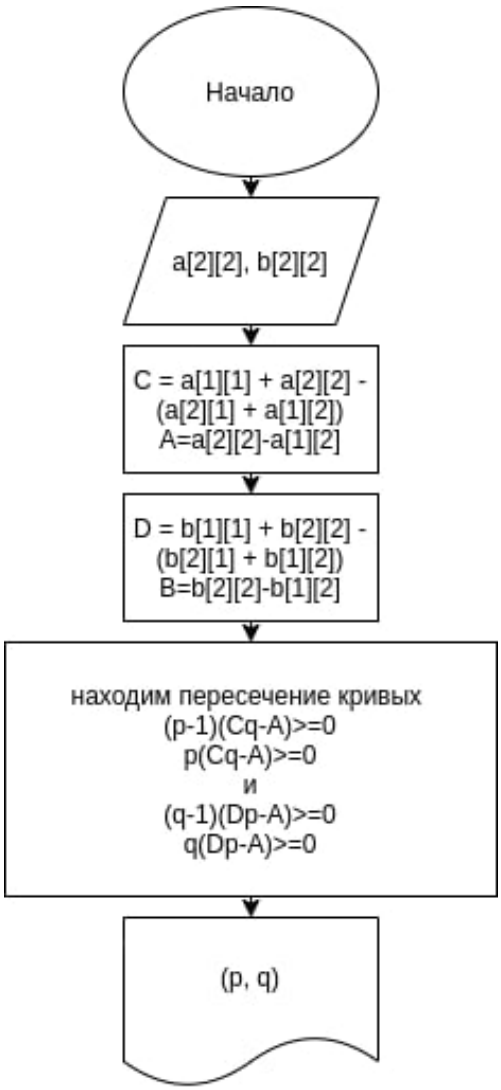

Рисунок 12 – Блок-схема алгоритма

Так мы получаем вероятности ходов для игрока А и игрока Б.

Предположим, что matrixA это матрица выплат для Игрока А, тогда для решения нам потребуется нарисовать кривые.

Пример из кода:

```
    Point[] p1_payoff_p = {new Point(0.0, matrixA[1]),
new Point(1.0, matrixA[0])};
    Point[] p1_payoff_MIN_p = {new Point(0.0,
matrixA[3]), new Point(1.0, matrixA[2])};
    t1 = new Line(p1_payoff_p[0], p1_payoff_p[1]);
    b1 = new Line(p1_payoff_MIN_p[0],
p1_payoff_MIN_p[1]);
```

Тоже самое мы делаем для второго игрока. После чего мы должны найти пересечение этих кривых. Если их кривые не пересекаются или пересекаются всего один раз, то у каждого из игроков нет доминирующей стратегии, а значит нам необходимо найти равновесие в смешанной стратегии.

Для того чтобы программа работала корректно во всех случаях, мной был добавлен функционал нахождения равновесия в чистой стратегии.

Пример кода:

```
    public      Line      whichCurveIsDominantStrat(Line
payOffCurve1, Line payOffCurve2) {
            if          (hasDominantStrat(payOffCurve1,
payOffCurve2)) {
                if      (payOffCurve1.get_y(0.5)      >
payOffCurve2.get_y(0.5)) {
                    return payOffCurve1;
                }
                return payOffCurve2;
            }
            return null;
        }
```

Функционал нахождения равновесия в чистой стратегии изображен на рисунке 13.

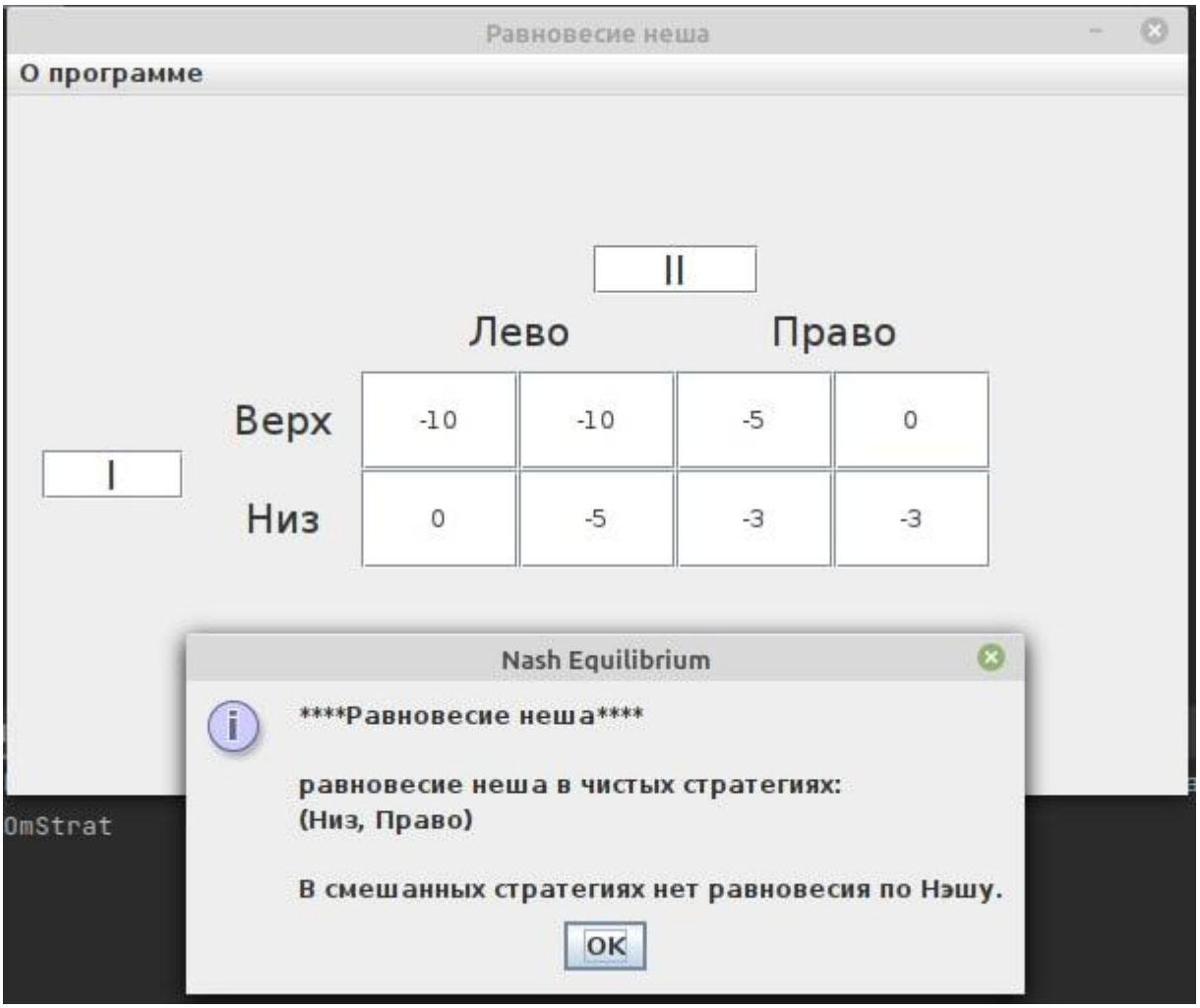

Рисунок 13 - Функционал нахождения равновесия в чистой стратегии

Так функционал работает на примере задачи «дилемма заключенного».

Дальше (на шаге 4) необходимо выполнить проверку, что существует хотя бы одна доминирующая стратегия.

Пример кода из приложения А:

```
/*
     * Определяет, происходит ли пересичение между двумя линиями
```

```
         * Выводит массив точек размера 0, если перехват
отсутствует
         * Выводит массив точек размера 1, содержащий точку
пересичения, если 1 перехват
         * Выводит массив точек размера 2, если перисечение это
линия
         */
        public Point[] intercept(Line line2) {
            if (this.getSlope() == line2.getSlope()) {

                if (this.isSameLine(line2)) {

                    Point[]     arr     =     {new     Point(0.0,
this.get_y(0.0)), new Point(1.0, this.get_y(1.0))};
                    return arr;
                } else {
                    Point[] arr = {};
                    return arr;
                }
            }
            double    X_val    =    -(this.getY_intercept()    -
line2.getY_intercept()) / (this.getSlope() - line2.getSlope());
            Point[] arr = {new Point(X_val, this.get_y(X_val))};
            return arr;
        }
```

Если мы не попадаем под условия, то тем самым мы убеждаемся, что пересечение есть только одно и доминирующих стратегий нету. После чего мы находим вышеприведенной функцией мы находим вероятности выбора стратегии хода.

На этом алгоритмическая часть разработки заканчивается. Для наглядности был организован UI на основе библиотеки Swing.

Мной был выбран эта библиотека, так как она является дстаточо популярной уже много лет, и является основой для более высокоуровневых библиотек, таких как Java fx, swt и других.

Несмотря на то, что java swing является достаточно устаревшей и низкоуровневой, она достаточно поста в использовании. Так например создание окон в моем приложении выполняется всего парой строк

```
new Runnable() {
```

```
            public void run() {
                try {
                    NashSolverGUI window = new
NashSolverGUI();
                    window.frame.setVisible(true);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
      }
```

Для моей программы потребовалось всего пара программных форм, таких как TextField:

```
p2tl = getTextField(player1MovesTop);
```

Таким образом создается поле для ввода платежа в платёжную матрицу.

Наша программа готова, теперь можем приступить к тестированию. Так же использовались JLabel:

```
lblLeftLabel = new JLabel("Лево");
```

Для позиционирования всех элементов их надо складывать в контейнеры:

```
    player2MoveChoices = new JPanel();
     player2MoveChoices.setBounds(180, 100, 320, 40);
     frame.getContentPane().add(player2MoveChoices);
     player2MoveChoices.setLayout(new GridLayout(1, 0,
0, 0));
     lblLeftLabel.setHorizontalAlignment(SwingConstants
.CENTER);
     lblLeftLabel.setFont(new Font("Tahoma",
Font.PLAIN, 20));
     player2MoveChoices.add(lblLeftLabel);
```

На этом основные элементы, использованные в программе, закончились. Как можно заметить несмотря на приличный возраст данной

библиотеки, видно, что UI для программы написан очень легко. В связи со своей легкостью и с распространённостью, Swing был добавлен в стандартный пакет JRE[30].

3.2 Тестирование программы

Запустим программу.

Из первого окна можно открыть вкладку «Помощь». Она была реализована для того, чтобы было проще разобраться с элементами управления программы. Вкладка помощь изображена на рисунке 14.

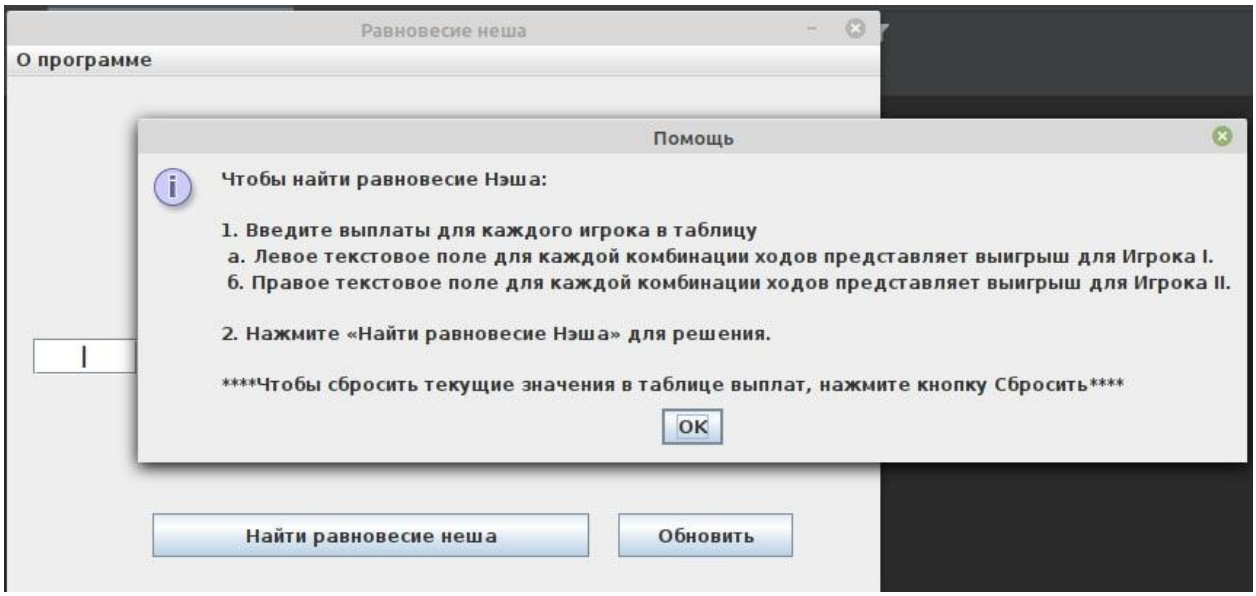

Рисунок 14 – Запуск программы. Кнопка «Помощь»

После ознакомления с элементами управления перед нами появляется окно для ввода матрицы платежей. С возможностью ввести имена игроков. Окно для ввода матрицы платежей изображено на рисунке 15.

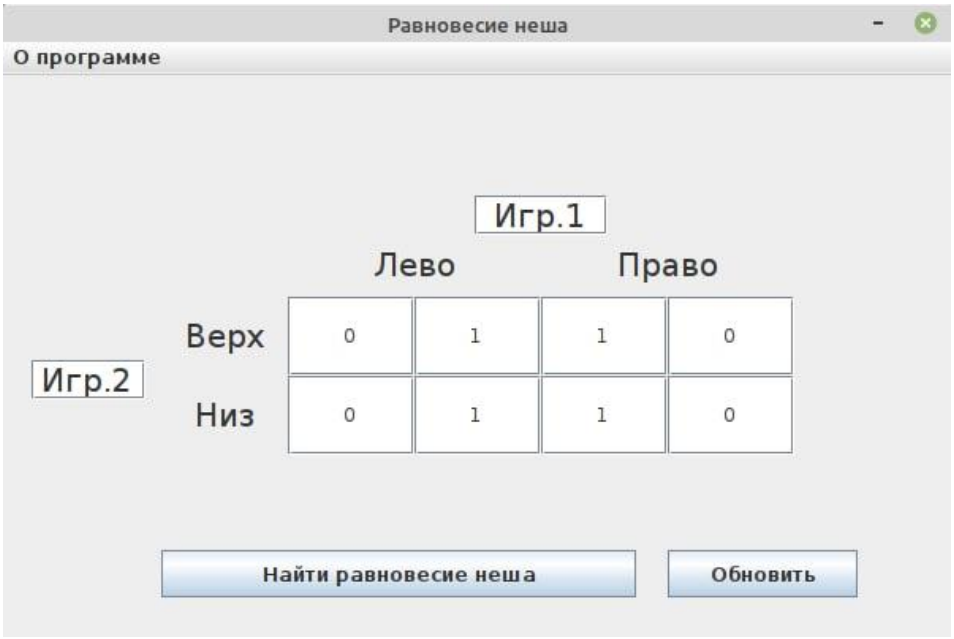

Рисунок 15 - Окно для ввода матрицы платежей

Для того чтобы убедиться, что программа работает корректно введем следующие значения. Ввод значений изображен на рисунке 16.

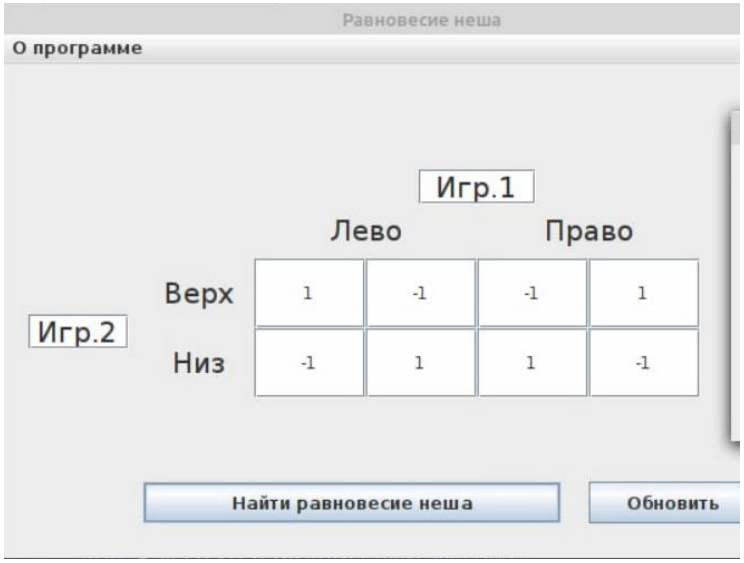

Рисунок 16 – Ввод значений

Данные значения мы взяли из задачи теории «орел или решка». Их часто используют для объяснения того, как работают смешанные стратегии.

Нажав на кнопку «Найти равновесие Нэша» получаем следующий результат, изображенный на рисунке 17:

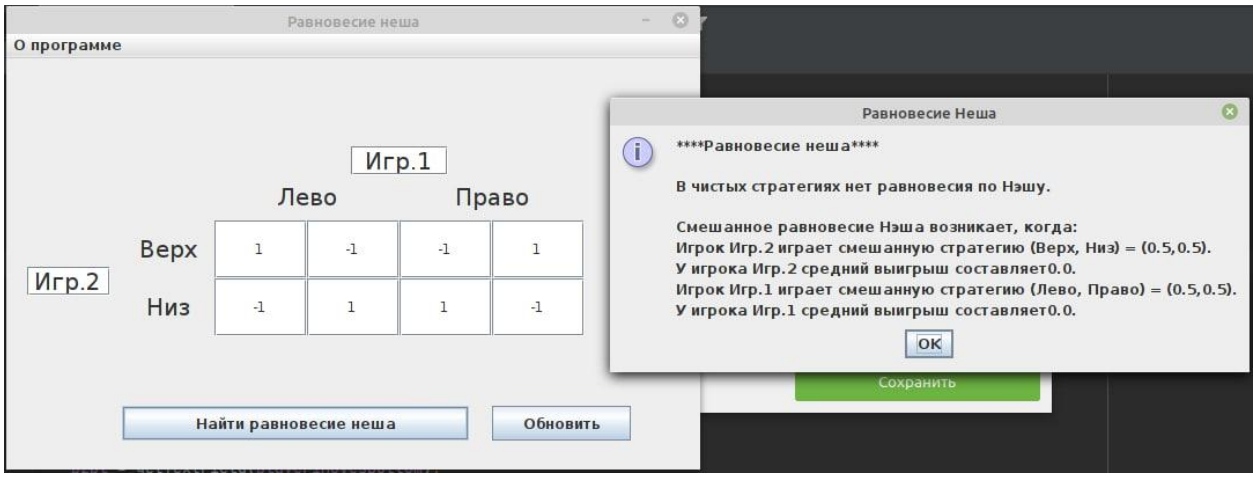

Рисунок 17 – Результат программы

Для проверки корректности выполнения алгоритма программы мы можем воспользоваться заранее вычисленными результатами с главы 2, где мы вручную произвели вычисления.

Программное решение первой задачи.

Условия задачи: $\begin{pmatrix} 6 & 2 \\ 2 & 4 \end{pmatrix} = \mathrm{A}, \begin{pmatrix} 2 & 6 \\ 8 & 2 \end{pmatrix} = B.$

На рисунке 18 вводим начальные данные в таблицу.

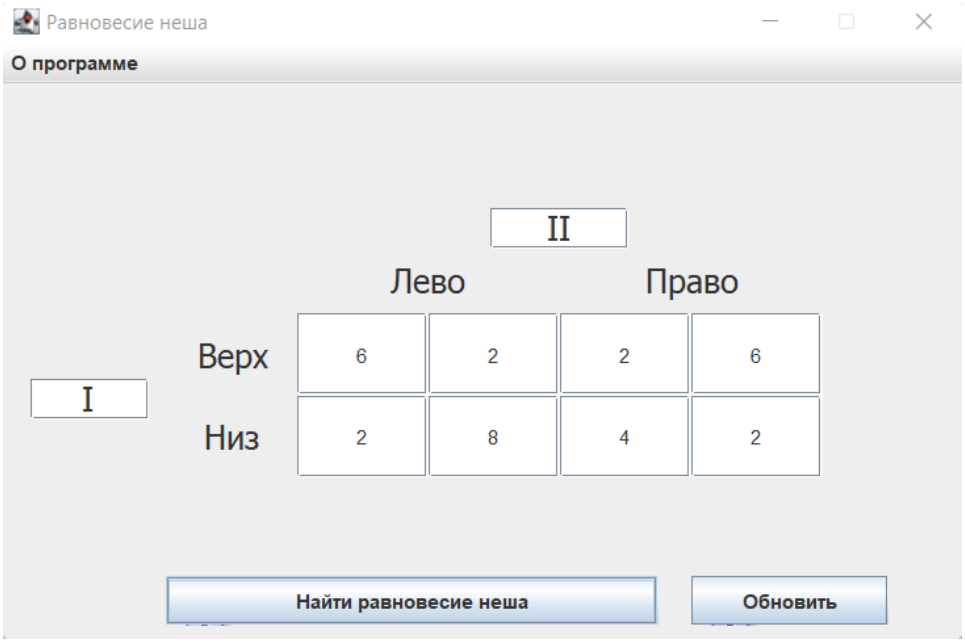

Рисунок 18 – Программа с введенными входными данными для первой задачи

Получаем результат выполнения на рисунке 19:

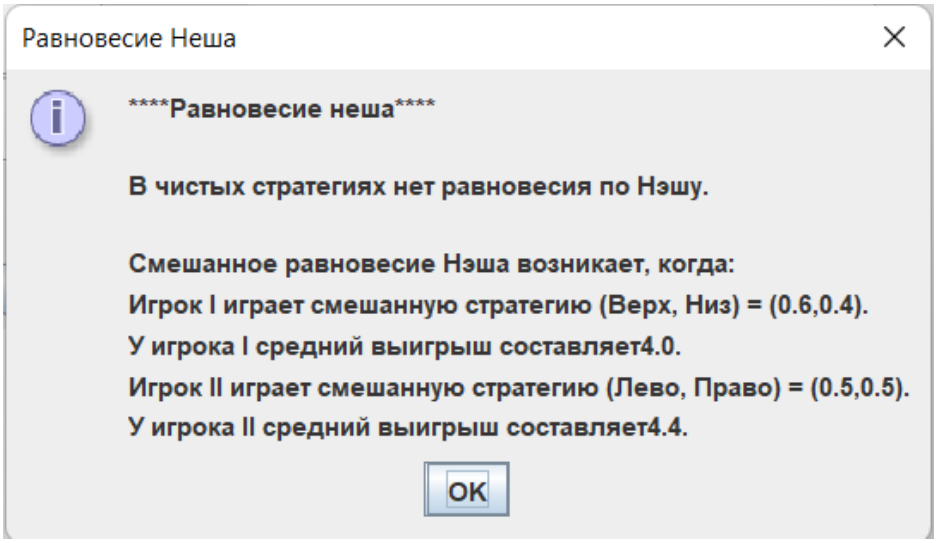

Рисунок 19 – Результат работы программы по первой задаче

Результат, учитывая машинное округление сходится с ручным вычислением.

Программное решение второй задачи.

Второй задачей была одна из разновидностей дилеммы заключенного. В таблицах 3-4 приведены данные.

Таблица 3 - Платёжная матрица второй задачи 1-ого игрока

	B_M	$B_Г$
A_M	-1	-9
$A_Г$	0	-6

Таблица 4 - Платёжная матрица второй задачи 2-ого игрока

	B_M	$B_Г$
A_M	-1	0
$A_Г$	-9	-6

На рисунке 20 вводим начальные данные в таблицу для второй задачи.

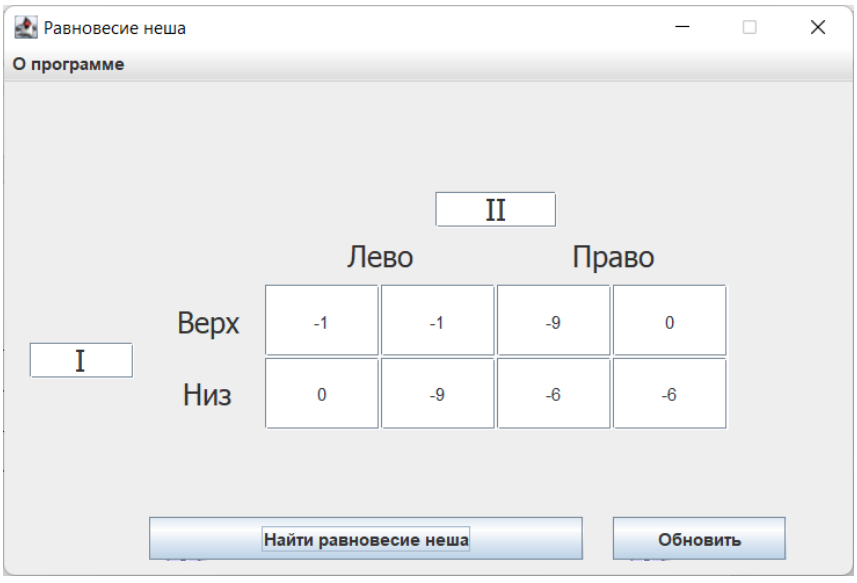

Рисунок 20 - Программа с введенными входными данными для второй задачи

По введенным входным данным запускаем программу и получаем результат на рисунке 21.

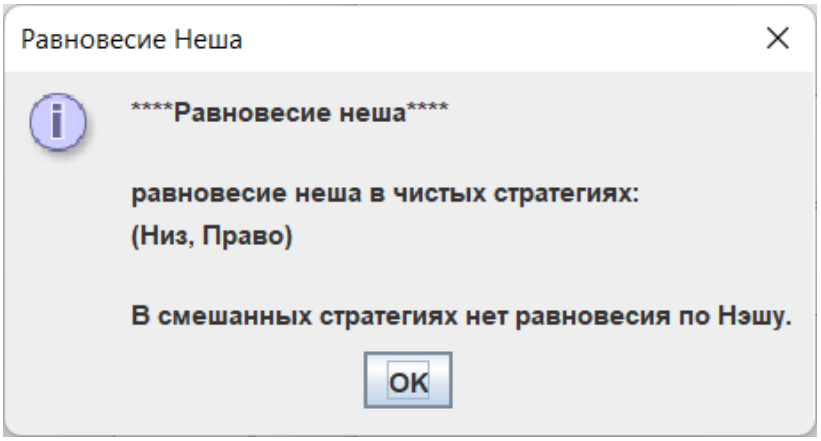

Рисунок 21 - Результат работы программы по второй задаче

Как мы можем видеть – программа работает. Сравним полученный результат с результатом задачи «орел и решка» и убедимся в том, что программа работает корректно и эффективно.

Цель выпускной квалификационной работы достигнута.

Заключение

Во время выполнения выпускной квалификационной работы на тему «Методы поиска равновесия Нэша в смешанных стратегиях», рассматривались игровые методы оптимизации, которые повсеместно используются в экономике, политологии, маркетинга и во многих других отраслях.

Теоретические основы теории игр, математическая модель равновесия Нэша, а также методы его нахождения на протяжении многих лет не теряют свою актуальность и продолжают распространяться на новые отрасли.

Цель работы заключалась в реализации алгоритма нахождения равновесия Нэша в смешанных стратегиях.

Для достижения поставленной цели были рассмотрены: история возникновения равновесия Нэша, его применение в разных сферах жизни и ситуациях, основные понятия. Также в работе много внимания уделялось решению биматричных игр и выводу формул для поиска равновесия Нэша в игре. Так были выведены формулы для игры $2 \times 2, 3 \times 3,$ и обобщенная формула $n \times n$.

С помощью существующей математической модели был реализован алгоритм поиска равновесия Нэша в смешанных стратегиях для игр 2×2. Этот алгоритм был разработан на языке программирования Java с применением Swing в современной и удобной среде для создания приложений IntelliJ IDEA. Для работы с реализованной программой и анализа ее эффективности использовалась операционная система Windows. Так же ввиду мультиплатформенности языка программирования java и его jvm программа была протестирована на операционной системе Linux mint 22.

Стоит отметить, что в реализованной программе можно не только искать коэффициенты для решения равновесия Нэша в смешанных стратегиях, но также программа показывает на оптимальную стратегию в чистых стратегиях и высчитывает средний выигрыш для обоих случаев

Список используемых источников

1. Алескеров Ф.Т. Индексы влияния, учитывающие предпочтения участников по созданию коалиций − Доклады Российской академии наук. 2007. Т. 414. № 5. С. 594-597.
2. Блекуэл Д., Гиршик М. Теория игр и статистических решений. − М.: Иностранная литература, 1958.
3. Васин А.А., Морозов В.В. Теория игр и модели математической экономики:Учебное пособие. – М.: Макс-Пресс, 2005. – 272 с.
4. Воробьёв Н. Н. Теория игр для экономистов-кибернетиков. — М.: Наука, 1985
5. Гермейер Ю.Б. Введение в теорию исследования операций. − М.: Наука, 1971.
6. Гермейер Ю.Б. Об играх двух лиц с фиксированной последовательностью ходов. ДАН, 1971, v. 198, т. 5, с. 1001-1004.
7. Горелик В.А. Теория игр и исследование операций. − М: Издво МИНГП, 1978.
8. Губко М.В., Новиков Д.А. Теория игр в управлении организационными системами: Учебное пособие. – М.: Синтег, 2005 – 138 с.
9. Давыдов Э.Г. Исследование операций. − М.: Высшая школа, 1990.
10. Захаров, А. В. Теория игр в общественных науках: учебник для вузов — М.:НИУ ВШЭ, 2015
11. Мазалов В. В. Математическая теория игр и приложения. — Изд во Лань, 2010, 446 с.
12. Меньшиков И.С. Лекции по теории игр и экономическому моделированию. – М., МЗ Пресс, 2006. – 208 с.
13. Оуэн Г. Теория игр. - М.: Вузовская книга, 2007. – 216 с.
14. Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. Теория игр. — СПб: БХВ-Петербург, 2012, 432 с.

15. Петросян С.Н., Зенкевич Н.А., Сёмина Е.А. Теория игр. М.: Высшая школа, 1998. – 304 с.
16. Протасов И.Д. Теория игр и исследование операций: Учебное пособие. _М.: Гелиос АРВ, 2003. - 368 с.
17. Романьков, В А Введение В Теорию Игр: Учебное Пособие / Романьков В А. - Москва: РГГУ, 2014. - 699 с.
18. Самаров К.Л. Математика. Учебно-методическое пособие по разделу «Элементы теории игр», ООО «Резольвента»,2011.-211с.
19. Смольяков, Э. Р. Теория антагонизмов и дифференциальные игры / Э.Р. Смольяков. - М.: Едиториал УРСС, 2016. - 160 с.
20. Сушко, Софья Стратегическое управление и теория игр в индустрии моды / Софья Сушко. - М.: LAP Lambert Academic Publishing, 2016. - 116 с.
21. Теория игр и экономическое поведение. Нейман фон Дж., Моргенштерн О. Издательство: Москва, Наука, 1970 г.
22. Теория игр. - М.: Наука, 2017. - 224 с.
23. Шагин, В.Л. Теория игр: Учебник и практикум / В.Л. Шагин. - Люберцы: Юрайт, 2016. - 223 с.
24. Amsterdam Proc., 1909, v. 11, continued in 1910, v. 12,13.
25. B. D. Bernheim; B. Peleg; M. D. Whinston (1987), "CoalitionProof Equilibria I. Concepts", Journal of Economic Theory, 42: 1–12, doi:10.1016/0022-0531(87)90099-8.
26. Bogomolnaia A., Le Breton M., Savvateev A., Weber S. Stability of jurisdiction structures under the equal share and median rules – Economic Theory. 2008. V. 34(3). P. 525-543.
27. Bogomolnaia A., Le Breton M., Savvateev A., Weber S. Stability under unanimous consent, free mobility and core – International Journal of Game Theory. 2007. V. 35(2). P. 185-204.
28. Borel E. Econometrica, 1953, v. 21, №1, p. 97-117.
29. Brouwer L.E.J. On continuous vector distributions of surfaces.

30. Dreze J., Le Breton M., Savvateev A., Weber S. "Almost" subsidy-free spatial pricing in a multi-dimensional setting – Journal of Economic Theory. 2008. V. 143. P. 275-291.
31. Dreze J., Le Breton M., Savvateev A., Weber S. A Problem of Football Bars: Vertically and Horizontally Differentiated Public Goods – X Международная научная конференция по проблемам развития экономики и общества. Т. 2. М.: ИД ГУ ВШЭ. 2010. С. 8.
32. Gilles D.B. Solutions to general non-zero-sum games. Contributions to the theory of games. IV (Kuhn H.W., Tucker A.W. eds.). Ann. Math. Studies, №40, − Princeton: Princeton Univ. Press, 1959, p. 47-86.
33. Kakutani S. A generalisation of Brower's fixed-point theorem. Duke Math. J., 1941, v. 8, №3, p. 457-459.
34. Kreps, D. M. A course in microeconomic theory / D. M. Kreps. — Princeton University Press, 1990.
35. Osborn, M.J. An introduction to game theory / M. J. Osborn. — N. Y. : Oxford University Press, 2004.

Приложение А

Программный код

```
public class Point {

    private double x;
    private double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public String toString() {
        return "Point [x=" + x + ", y=" + y + "]";
    }


    public Point compareY(Point point2){
        if(this.y > point2.getY()){
            return this;
        }
        return point2;
    }

    public double getX() {
        return x;
    }

    public double getY() {
        return y;
    }
}

public class NashSolverGUI {

    private JFrame frame;
    private JMenuBar menuBar;
    private JMenu menu;
    private JMenuItem menuHelp;
    private JMenuItem menuAbout;
    private JLabel lblTop;
    private JLabel lblBottom;
    private JTextField p1tl;
    private JTextField p2tl;
    private JTextField p1tr;
    private JTextField p2tr;
    private JTextField p1bl;
    private JTextField p2bl;
    private JTextField p1br;
```

Программный код

```
         private JTextField p2br;
        private JPanel player1MovesTop;

        private JPanel player1MovesBottom;
        private JPanel player2MoveChoices;
        private JLabel lblLeftLabel;
        private JLabel lblRightLabel;
        private JTextField p1label;
        private JTextField p2label;
        private JLabel lblOfPlayers;
        private JComboBox numPlayersComboBox;
        private JComboBox dimComboBox;
        private JLabel lblDimensions;
        private JPanel optionPanel;
        private JSeparator separator;
        private JFrame playerLines;
        private JFrame p1Lines;
        private JFrame p2Lines;

        private LineComponent p1LineComp;
        private LineComponent p2LineComp;
        private double p1tlVal = 0, p1trVal = 0, p1blVal = 0,
p1brVal = 0;
        private double p2tlVal = 0, p2trVal = 0, p2blVal = 0,
p2brVal = 0;

        public static void main(String[] args) {
            EventQueue.invokeLater(new Runnable() {
                public void run() {
                    try {
                        NashSolverGUI window = new
NashSolverGUI();
                        window.frame.setVisible(true);
                    } catch (Exception e) {
                        e.printStackTrace();
                    }
                }
            });
        }

        public NashSolverGUI() {
            initialize();
        }

        private void initialize() {

            frame = new JFrame("Равновесие неша");
            frame.setBounds(100, 100, 600, 400);
```

```
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setResizable(false);
            frame.getContentPane().setLayout(null);
            menuBar = new JMenuBar();

            menu = new JMenu("О программе");
            menuBar.add(menu);
            menuHelp = new JMenuItem("Помощь");
            menuHelp.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent ae) {
                    String helpMsg = "Чтобы найти равновесие
Нэша:\n\n";
                    helpMsg += "1. Введите выплаты для каждого
игрока в таблицу\n";
                    helpMsg += "\t\t а. Левое текстовое поле для
каждой комбинации ходов представляет выигрыш для Игрока I.\n";
                    helpMsg += "\t\t б. Правое текстовое поле
для каждой комбинации ходов представляет выигрыш для Игрока
II.\n\n";
                    helpMsg += "2. Нажмите «Найти равновесие
Нэша» для решения.\n\n";
                    helpMsg += "****Чтобы сбросить текущие
значения в таблице выплат, нажмите кнопку Сбросить****";
                    JOptionPane.showMessageDialog(null, helpMsg,
"Помощь", JOptionPane.INFORMATION_MESSAGE);
                }
            });
            menu.add(menuHelp);
            menuAbout = new JMenuItem("О программе");
            menuAbout.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent ae) {
                    String aboutMsg =
                            "Добро пожаловать\n\n" +
                                    "Этот инструмент
предоставляет пользователям возможность\n" +
                                    "находить равновсие неша в
чистых и смешанных стратегиях.\n\n" +
                                    "Ввод: выплаты игрока I и
игрока II в виде десятичных чисел\n\n" +
                                    "Создано Евстинеевым Егром
Дмитриевичем\n" +
                                    "В рамках выпускной
квалификационной работы";
                    System.out.println(aboutMsg);
                    JOptionPane.showMessageDialog(null,
aboutMsg, "О программе", JOptionPane.INFORMATION_MESSAGE);
                }
            });
            menu.add(menuAbout);
            frame.setJMenuBar(menuBar);
            p2label = new JTextField();
```

```
            p2label.setText("II");

            p2label.setHorizontalAlignment(SwingConstants.CENTER
);
            p2label.setFont(new Font("Tahoma", Font.PLAIN, 20));
            p2label.setBounds(298, 76, 84, 25);
            frame.getContentPane().add(p2label);
            player2MoveChoices = new JPanel();
            player2MoveChoices.setBounds(180, 100, 320, 40);
            frame.getContentPane().add(player2MoveChoices);
            player2MoveChoices.setLayout(new GridLayout(1, 0, 0,
0));
            lblLeftLabel = new JLabel("Лево");
            lblLeftLabel.setHorizontalAlignment(SwingConstants.C
ENTER);
            lblLeftLabel.setFont(new Font("Tahoma", Font.PLAIN,
20));
            player2MoveChoices.add(lblLeftLabel);
            lblRightLabel = new JLabel("Право");
            lblRightLabel.setFont(new Font("Tahoma", Font.PLAIN,
20));
            lblRightLabel.setHorizontalAlignment(SwingConstants.
CENTER);
            player2MoveChoices.add(lblRightLabel);
            p1label = new JTextField();
            p1label.setText("I");
            p1label.setHorizontalAlignment(SwingConstants.CENTER
);
            p1label.setFont(new Font("Tahoma", Font.PLAIN, 20));
            p1label.setBounds(18, 180, 72, 25);
            frame.getContentPane().add(p1label);
            player1MovesTop = new JPanel();
            player1MovesTop.setBounds(100, 140, 400, 50);
            frame.getContentPane().add(player1MovesTop);
            player1MovesTop.setLayout(new GridLayout(1, 0, 0,
0));
            lblTop = new JLabel("Верх");
            lblTop.setFont(new Font("Tahoma", Font.PLAIN, 20));
            lblTop.setHorizontalAlignment(SwingConstants.CENTER)
;
            player1MovesTop.add(lblTop);
            p1tl = getTextField(player1MovesTop);
            p2tl = getTextField(player1MovesTop);
            p1tr = getTextField(player1MovesTop);
            p2tr = getTextField(player1MovesTop);
            player1MovesBottom = new JPanel();
            player1MovesBottom.setBounds(100, 190, 400, 50);
            frame.getContentPane().add(player1MovesBottom);
            player1MovesBottom.setLayout(new GridLayout(1, 0, 0,
0));
```

```
            lblBottom = new JLabel("Низ");

            lblBottom.setFont(new Font("Tahoma", Font.PLAIN,
20));
            lblBottom.setHorizontalAlignment(SwingConstants.CENT
ER);
            player1MovesBottom.add(lblBottom);
            p1bl = getTextField(player1MovesBottom);
            p2bl = getTextField(player1MovesBottom);
            p1br = getTextField(player1MovesBottom);
            p2br = getTextField(player1MovesBottom);
            JButton findNashButton = new JButton("Найти
равновесие неша");
            findNashButton.setBounds(100, 300, 300, 30);
            findNashButton.addActionListener(new
ActionListener() {
                public void actionPerformed(ActionEvent ae) {
                    try {
                        p1tlVal =
Double.parseDouble(p1tl.getText());
                        p1trVal =
Double.parseDouble(p1tr.getText());
                        p1blVal =
Double.parseDouble(p1bl.getText());
                        p1brVal =
Double.parseDouble(p1br.getText());

                        p2tlVal =
Double.parseDouble(p2tl.getText());
                        p2trVal =
Double.parseDouble(p2tr.getText());
                        p2blVal =
Double.parseDouble(p2bl.getText());
                        p2brVal =
Double.parseDouble(p2br.getText());
                    } catch (NumberFormatException e) {
                        String errorMsg = "Invalid Input:
\nВнесенные числа не double";
                        System.out.println(errorMsg);
                        JOptionPane.showMessageDialog(null,
errorMsg, "Invalid Input", JOptionPane.INFORMATION_MESSAGE);
                        return;
                    }
                    double[] gameMatrix = {p1tlVal, p2tlVal,
p1trVal, p2trVal, p1blVal, p2blVal, p1brVal, p2brVal};

                    MixedNashSolver solver = new
MixedNashSolver(gameMatrix);

                    String nashMsg = createNashMsg(solver);
```

```
                    JOptionPane.showMessageDialog(null, nashMsg,
"Равновесие Неша", JOptionPane.INFORMATION_MESSAGE);

                    playerLines = new JFrame(p1label.getText() +
"'s & " + p2label.getText() + "'s Lines");

                    playerLines.setBounds(100, 100, 600, 400);

                    p1LineComp = new LineComponent();
                    p2LineComp = new LineComponent();

                    playerLines.getContentPane().add(p1LineComp,
BorderLayout.CENTER);
                    playerLines.getContentPane().add(p2LineComp,
BorderLayout.CENTER);

                    Line t1 = solver.getT1();
                    Line b1 = solver.getB1();
                    Line l2 = solver.getL2();
                    Line r2 = solver.getR2();

                    p1LineComp.addLine(t1.get_y(0.0), 0.0,
t1.get_y(1.0), 1.0);
                    p1LineComp.addLine(b1.get_y(0.0), 0.0,
b1.get_y(1.0), 1.0);
                    p2LineComp.addLine(l2.get_y(0.0), 0.0,
l2.get_y(1.0), 1.0);
                    p2LineComp.addLine(r2.get_y(0.0), 0.0,
r2.get_y(1.0), 1.0);

                    playerLines.pack();
                    playerLines.setVisible(true);

                }

            });
            frame.getContentPane().add(findNashButton);

            JButton reset = new JButton("Обновить");
            reset.setBounds(420, 300, 120, 30);
            reset.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent ae) {
                    p1tl.setText("");
                    p1tr.setText("");
                    p1bl.setText("");
                    p1br.setText("");

                    p2tl.setText("");
                    p2tr.setText("");
                    p2bl.setText("");
```

```
                p2br.setText("");
            }
        });

        frame.getContentPane().add(reset);

    }

    private JTextField getTextField(JPanel panels) {
        var textField = new JTextField();
        textField.setHorizontalAlignment(SwingConstants.CENT
ER);
        panels.add(textField);
        textField.setColumns(10);
        return textField;
    }
    private String createNashMsg(MixedNashSolver n) {
        String msg = "****Равновесие неша****\n\n";
        if (n.isHasPure()) {

            msg += "равновесие неша в чистых стратегиях:
\n";
            if (n.isTopLeftIsPure()) {
                msg += "(Верх, Лево)\n";
            }
            if (n.isTopRightIsPure()) {
                msg += "(Верх, Право)\n";
            }
            if (n.isBottomLeftIsPure()) {
                msg += "(Низ, Лево)\n";
            }
            if (n.isBottomRightIsPure()) {
                msg += "(Низ, Право)\n";
            }
            msg += "\n";
        } else {
            msg += "В чистых стратегиях нет равновесия по
Нэшу.\n\n";
        }
        if (n.isHasMixed()) {
            msg += "Смешанное равновесие Нэша возникает,
когда:\n";
            if (n.isHasInfiniteMixedP1() &&
n.isHasInfiniteMixedP2()) {
                msg += "Игрок " + p1label.getText() + "
имеет бесконечные смешанные стратегии.\n";
                msg += "Игрок " + p2label.getText() + "
имеет бесконечные смешанные стратегии.\n";
            } else if (n.isHasInfiniteMixedP1()) {
                if (n.isTopRightIsPure()) {
```

```
                        msg += "Игрок " + p1label.getText() + "
имеет бесконечные смешанные стратегии.\n";
                        msg += "Игрок " + p2label.getText() + "
has a dominant strategy playing Право.";

                    } else {
                        msg += "Игрок " + p1label.getText() + "
имеет бесконечные смешанные стратегии.\n";
                        msg += "Игрок " + p2label.getText() + "
has a dominant strategy playing Лево.";
                    }

                } else if (n.isHasInfiniteMixedP2()) {
                    if (n.isTopRightIsPure()) {
                        msg += "Игрок " + p1label.getText() + "
имеет доминирующую стратегию игры.\n";
                        msg += "Игрок " + p2label.getText() + "
имеет бесконечную смешанную игру Верх.";
                    } else {
                        msg += "Игрок " + p1label.getText() + "
имеет доминирующую стратегию игры.\n";
                        msg += "Игрок " + p2label.getText() + "
имеет бесконечную смешанную игру Низ.";
                    }
                } else {
                    msg += "Игрок " + p1label.getText() + "
играет смешанную стратегию (Верх, Низ) = (" + round(n.getP()) +
"," + round((1.0 - n.getP())) + ").\n";
                    msg += "У игрока " + p1label.getText() + "
средний выигрыш составляет" + round(n.getPlayerI_avgPayoff()) +
".\n";

                    msg += "Игрок " + p2label.getText() + "
играет смешанную стратегию (Лево, Право) = (" + round(n.getQ())
+ "," + round((1.0 - n.getQ())) + ").\n";
                    msg += "У игрока " + p2label.getText() + "
средний выигрыш составляет" + round(n.getPlayerII_avgPayoff()) +
".\n";
                }
            } else {
                msg += "В смешанных стратегиях нет равновесия по
Нэшу.\n";
            }
            return msg;
        }
        private double round(double d) {
            return Math.round(d * 100.0) / 100.0;
        }
    }
    import java.util.Arrays;
```

```
    public class MixedNashSolver {

        private boolean hasPure;
        private boolean hasMixed;

        private boolean hasInfiniteMixedP1;
        private boolean hasInfiniteMixedP2;

        private boolean TopLeftIsPure = false;
        private boolean TopRightIsPure = false;
        private boolean BottomLeftIsPure = false;
        private boolean BottomRightIsPure = false;

        private double playerI_avgPayoff;
        private double playerII_avgPayoff;

        private double[] playerI_BR_p_given_q;
        private double[] playerII_BR_q_given_p;

        private double p;
        private double q;

        private Line t1;
        private Line b1;
        private Line l2;
        private Line r2;

        public String toString() {
            return "MixedNashSolver [hasPure=" + hasPure + ",
hasMixed=" + hasMixed
                    + ", hasInfiniteMixedP1=" +
hasInfiniteMixedP1 + ", hasInfiniteMixedP2=" +
hasInfiniteMixedP2
                    + ", TopLeftIsPure=" + TopLeftIsPure + ",
TopRightIsPure="
                    + TopRightIsPure + ", BottomLeftIsPure=" +
BottomLeftIsPure
                    + ", BottomRightIsPure=" + BottomRightIsPure
                    + ", playerI_avgPayoff=" + playerI_avgPayoff
                    + ", playerII_avgPayoff=" +
playerII_avgPayoff
                    + ", playerI_BR_p_given_q="
                    + Arrays.toString(playerI_BR_p_given_q)
                    + ", playerII_BR_q_given_p="
                    + Arrays.toString(playerII_BR_q_given_p) +
", p=" + p + ", q="
                    + q + "]";
        }

        public MixedNashSolver(double[] matrix) {
```

```
            /* Первый Шаг: Разделите Таблицу Выплат По
Разделам*/



            Point[] p1_payoff_p = {new Point(0.0, matrix[1]),
new Point(1.0, matrix[0])};

            Point[] p1_payoff_MIN_p = {new Point(0.0,
matrix[3]), new Point(1.0, matrix[2])};

            Point[] p2_payoff_q = {new Point(0.0, matrix[5]),
new Point(1.0, matrix[1])};

            Point[] p2_payoff_MIN_q = {new Point(0.0,
matrix[7]), new Point(1.0, matrix[3])};

            /* Второй Шаг: Создание Кривых Выигрыша */

            t1 = new Line(p1_payoff_p[0], p1_payoff_p[1]);

            b1 = new Line(p1_payoff_MIN_p[0],
p1_payoff_MIN_p[1]);

            l2 = new Line(p2_payoff_q[0], p2_payoff_q[1]);

            r2 = new Line(p2_payoff_MIN_q[0],
p2_payoff_MIN_q[1]);

            /* *** Настройка Завершена *** */

            /*
             * Третий шаг: Проверьте, существует ли уникальное
равновесие Нэша (у каждого игрока есть доминирующая стратегия)
             * Следовательно, никакого смешанного NashE, а
только одна чистая стратегия
             */

            if (hasDominantStrat(t1, b1) && hasDominantStrat(l2,
r2)) {

                System.out.println("DOmStrat");
                this.hasMixed = false;
                this.hasPure = true;

                if (whichCurveIsDominantStrat(t1,
b1).equals(t1)) {

                    if (whichCurveIsDominantStrat(l2,
r2).equals(l2)) {
```

```
                        this.setPureNash(true, false, false,
false);

                    } else {

                        this.setPureNash(false, true, false,
false);
                    }
                } else {

                    if (whichCurveIsDominantStrat(l2,
r2).equals(l2)) {

                        this.setPureNash(false, false, true,
false);
                    } else {

                        this.setPureNash(false, false, false,
true);
                    }
                }

            } else if (t1.isSameLine(b1) && l2.isSameLine(r2)) {
                this.hasPure = true;
                this.hasMixed = true;
                this.hasInfiniteMixedP1 = true;
                this.hasInfiniteMixedP2 = true;
                this.setPureNash(true, true, true, true);

            }

            /*
             * Четвертый шаг: Проверьте, существует ли ровно
одна доминирующая стратегия
             */

            else if (hasDominantStrat(t1, b1) ||
hasDominantStrat(l2, r2)) {

                System.out.println("1 dominant strategy");
                this.hasPure = true;

                if (hasDominantStrat(t1, b1)) {
                    if (whichCurveIsDominantStrat(t1, b1) == t1)
{
                        this.p = 1.0;

                        if (getMaxPayoffLine(p, l2, r2) == null)
{
                            this.hasMixed = true;
                            this.hasInfiniteMixedP2 = true;
```

```
                            this.setPureNash(true, true, false,
false);
                        } else if (getMaxPayoffLine(p, l2, r2)
== l2) {

                            this.setPureNash(true, false, false,
false);
                        } else if (getMaxPayoffLine(p, l2, r2)
== r2) {
                            this.setPureNash(false, true, false,
false);
                        } else {
                        }

                    } else {
                        this.p = 0.0;

                        if (getMaxPayoffLine(p, l2, r2) == null)
{
                            this.hasMixed = true;
                            this.hasInfiniteMixedP2 = true;
                            this.setPureNash(false, false, true,
true);
                        } else if (getMaxPayoffLine(p, l2, r2)
== l2) {
                            this.setPureNash(false, false, true,
false);
                        } else if (getMaxPayoffLine(p, l2, r2)
== r2) {
                            this.setPureNash(false, false,
false, true);
                        } else {
                        }
                    }
                } else {

                    if (whichCurveIsDominantStrat(l2, r2) == l2)
{
                        this.q = 1.0;

                        if (getMaxPayoffLine(q, t1, b1) == null)
{
                            this.hasMixed = true;
                            this.hasInfiniteMixedP1 = true;
                            this.setPureNash(true, false, true,
false);
                        } else if (getMaxPayoffLine(q, t1, b1)
== t1) {
                            this.setPureNash(true, false, false,
false);
```

```
                        } else if (getMaxPayoffLine(q, t1, b1)
== b1) {
                            this.setPureNash(false, false, true,
false);
                        } else {

     }

                    } else {
                        this.q = 0.0;

                        if (getMaxPayoffLine(q, t1, b1) == null)
{
                            this.hasMixed = true;
                            this.hasInfiniteMixedP1 = true;
                            this.setPureNash(false, true, false,
true);
                        } else if (getMaxPayoffLine(q, t1, b1)
== t1) {
                            this.setPureNash(false, true, false,
false);
                        } else if (getMaxPayoffLine(q, t1, b1)
== b1) {
                            this.setPureNash(false, false,
false, true);
                        } else {
                        }
                    }

                }
            }

            /*
             * Пятый шаг: Тогда нет доминирующих стратегий, и мы
имеем смешанное равновесие Нэша
             * от нуля до двух чистого Нэша. Существует один
перехват, то есть некоторый X val s.t. 0 < X > 1
             * */
            else {
                System.out.println("No dominant strategies");
                this.hasMixed = true;

                this.playerI_avgPayoff =
t1.robustIntercept(b1).getY();
                this.playerII_avgPayoff =
l2.robustIntercept(r2).getY();

                this.p = l2.robustIntercept(r2).getX();
                this.q = t1.robustIntercept(b1).getX();

                if (p == 0 || q == 0 || p == 1 || q == 1) {
```

```
                    this.hasPure = true;
                    this.hasMixed = false;

                    if (p == 0 && q == 0) {
                        System.out.println(p + "||" + q);

                        this.setPureNash(false, true, true,
true);
                    }
                    if (p == 0 && q == 1) {
                        System.out.println(p + "||" + q);
                        this.setPureNash(true, false, true,
true);
                    }
                    if (p == 1 && q == 0) {
                        System.out.println(p + "||" + q);
                        this.setPureNash(true, true, false,
true);
                    }
                    if (p == 1 && q == 1) {
                        System.out.println("p = q = 1");
                        this.setPureNash(true, true, true,
false);
                    }

                } else {

                    if (t1.getX_intercept() >
b1.getY_intercept()) {
                        this.playerI_BR_p_given_q = new
double[]{1.0, this.q, 0.0};
                    } else {
                        this.playerI_BR_p_given_q = new
double[]{0.0, this.q, 1.0};
                    }
                    if (l2.getY_intercept() >
r2.getY_intercept()) {
                        this.playerII_BR_q_given_p = new
double[]{1.0, this.p, 0.0};
                    } else {
                        this.playerII_BR_q_given_p = new
double[]{0.0, this.p, 1.0};
                    }

                    /*
                     * Шестой шаг: Учитывая функции BR, мы можем
вычислить, существуют ли два чистых равновесия, если BR
                     * * значения равны, в противном случае нет
чистых нэшей
                     */
```

```
                    System.out.println(Arrays.toString(playerI_B
R_p_given_q));
                    System.out.println(Arrays.toString(playerII_
BR_q_given_p));


                    if (playerI_BR_p_given_q[0] ==
playerII_BR_q_given_p[0]) {
                        this.hasPure = true;
                        if (playerI_BR_p_given_q[0] == 1.0) {

                            this.setPureNash(false, true, true,
false);
                        } else {

                            this.setPureNash(true, false, false,
true);
                        }
                    }
                }
            }
        }

        /**
         * Определяет, есть ли у игрока доступная доминирующая
стратегия
         * путем проверки наличия пересечения
         */
        public boolean hasDominantStrat(Line payOffCurve1, Line
payOffCurve2) {

            if (payOffCurve1.intercept(payOffCurve2).length ==
0)
                return true;
            if (!(payOffCurve1.intercept(payOffCurve2).length ==
1))
                return false;

            if (payOffCurve1.intercept(payOffCurve2)[0].getX() <
0 || payOffCurve1.intercept(payOffCurve2)[0].getX() > 1) {
                return true;
            } else {
                return false;
            }

        }

        public Line whichCurveIsDominantStrat(Line payOffCurve1,
Line payOffCurve2) {
            if (hasDominantStrat(payOffCurve1, payOffCurve2)) {
```

```
                if (payOffCurve1.get_y(0.5) >
payOffCurve2.get_y(0.5)) {
                    return payOffCurve1;
                }
                return payOffCurve2;
            }

            return null;
        }

        public Line getMaxPayoffLine(double x, Line l1, Line l2)
{
            if (l1.get_y(x) > l2.get_y(x)) {
                return l1;
            } else if (l1.get_y(x) < l2.get_y(x)) {
                return l2;
            } else {
                return null;
            }
        }

        public void setPureNash(boolean tl, boolean tr, boolean
bl, boolean br) {
            this.TopLeftIsPure = tl;
            this.TopRightIsPure = tr;
            this.BottomLeftIsPure = bl;
            this.BottomRightIsPure = br;
        }

        public boolean isHasPure() {
            return hasPure;
        }

        public boolean isHasMixed() {
            return hasMixed;
        }

        public boolean isHasInfiniteMixedP1() {
            return hasInfiniteMixedP1;
        }

        public boolean isHasInfiniteMixedP2() {
            return hasInfiniteMixedP2;
        }

        public boolean isTopLeftIsPure() {
            return TopLeftIsPure;
        }

        public boolean isTopRightIsPure() {
            return TopRightIsPure;
```

```
}

    public boolean isBottomLeftIsPure() {
        return BottomLeftIsPure;
    }


    public boolean isBottomRightIsPure() {
        return BottomRightIsPure;
    }

    public double getPlayerI_avgPayoff() {
        return playerI_avgPayoff;
    }

    public double getPlayerII_avgPayoff() {
        return playerII_avgPayoff;
    }

    public double[] getPlayerI_BR_p_given_q() {
        return playerI_BR_p_given_q;
    }

    public double[] getPlayerII_BR_q_given_p() {
        return playerII_BR_q_given_p;
    }

    public double getP() {
        return p;
    }

    public double getQ() {
        return q;
    }

    public Line getT1() {
        return t1;
    }

    public Line getB1() {
        return b1;
    }

    public Line getL2() {
        return l2;
    }

    public Line getR2() {
        return r2;
    }
}
```

```
    public class Line {

        private final double slope;

        private final Point y_intercept;
        private final Point x_intercept;

        public Line(Point p1, Point p2) {
            this.slope = (p1.getY() - p2.getY()) / (p1.getX() -
p2.getX());
            this.y_intercept = new Point(0.0, (p1.getY() -
(this.slope * p1.getX())));
            this.x_intercept = new Point(-
this.y_intercept.getY() / this.slope, 0.0);
        }

        public String toString() {
            return "Line [slope=" + slope + ", y_intercept=" +
y_intercept
                    + ", x_intercept=" + x_intercept + "]";
        }

        /**
         * Определяет значение x для пересечения двух линий
         * Должен знать, что пересичение существует
         */
        public Point robustIntercept(Line line2) {
            double X_val = -(this.getY_intercept() -
line2.getY_intercept()) / (this.getSlope() - line2.getSlope());
            return new Point(X_val, this.get_y(X_val));
        }

        /**
         * Определяет, происходит ли пересичение между двумя
линиями
         * Выводит массив точек размера 0, если перехват
отсутствует
         * Выводит массив точек размера 1, содержащий точку
пересичения, если 1 перехват
         * Выводит массив точек размера 2, если перисечение это
линия
         */
        public Point[] intercept(Line line2) {
            if (this.getSlope() == line2.getSlope()) {

                if (this.isSameLine(line2)) {
```

```
                Point[] arr = {new Point(0.0, this.get_y(0.0)),
new Point(1.0, this.get_y(1.0))};
                    return arr;
                } else {
                    Point[] arr = {};
                    return arr;
                }

            }
            double X_val = -(this.getY_intercept() -
line2.getY_intercept()) / (this.getSlope() - line2.getSlope());
            Point[] arr = {new Point(X_val, this.get_y(X_val))};
            return arr;
        }

        public boolean isSameLine(Line line2) {
            return ((this.get_y(0.0) == line2.get_y(0.0)) &&
(this.get_y(1.0) == line2.get_y(1.0)));
        }

        public double get_x(double y) {
            return (y - y_intercept.getY()) / slope;
        }

        public double get_y(double x) {
            return slope * x + y_intercept.getY();
        }

        public double getSlope() {
            return slope;
        }

        public double getY_intercept() {
            return y_intercept.getY();
        }

        public double getX_intercept() {
            return x_intercept.getX();
        }
    }

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.Line2D;
import java.awt.Shape;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.LinkedList;
```

```
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class LineComponent extends JComponent{

private static class Line{
    final double x1;
    final double y1;
    final double x2;
    final double y2;
    final Color color;

    public Line(double x1, double y1, double x2, double y2,
Color color) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
        this.color = color;
    }
}

private final LinkedList<Line> lines = new LinkedList<Line>();

public void addLine(double x1, double x2, double x3, double x4)
{
    addLine(x1, x2, x3, x4, Color.black);
}

public void addLine(double x1, double x2, double x3, double x4,
Color color) {
    lines.add(new Line(x1,x2,x3,x4, color));
    repaint();
}

public void clearLines() {
    lines.clear();
    repaint();
}

@Override
protected void paintComponent(Graphics g) {
    Graphics2D g2 = (Graphics2D) g;
    for (Line line : lines) {
        g2.setColor(line.color);
        Shape l = new Line2D.Double(line.x1, line.y1, line.x2,
line.y2);
        g2.draw(l);
```

```
}
       /*
    super.paintComponent(g);
    for (Line line : lines) {
        g.setColor(line.color);
        g.drawLine(line.x1, line.y1, line.x2, line.y2);
    }

*/
}

public static void main(String[] args) {
    JFrame testFrame = new JFrame();
    testFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
    final LineComponent comp = new LineComponent();
    comp.setPreferredSize(new Dimension(320, 200));
    testFrame.getContentPane().add(comp, BorderLayout.CENTER);
    JPanel buttonsPanel = new JPanel();
    JButton newLineButton = new JButton("New Line");
    JButton clearButton = new JButton("Clear");
    buttonsPanel.add(newLineButton);
    buttonsPanel.add(clearButton);
    testFrame.getContentPane().add(buttonsPanel,
BorderLayout.SOUTH);
    newLineButton.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            double x1 = (double) (Math.random()*320);
            double x2 = (double) (Math.random()*320);
            double y1 = (double) (Math.random()*200);
            double y2 = (double) (Math.random()*200);
            Color randomColor = new Color((float)Math.random(),
(float)Math.random(), (float)Math.random());
            comp.addLine(x1, y1, x2, y2, randomColor);
        }
    });
    clearButton.addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
            comp.clearLines();
        }
    });
    testFrame.pack();
    testFrame.setVisible(true);
}

}
```