Санкт-Петербургский государственный университет
Кафедра математической теории игр и статических решений

**Черников Евгений Викторович**

**Магистерская диссертация**

**α-N-ядро в би-кооперативных играх**

Направление 01.04.02

Прикладная математика и информатика

Магистерская программа

Исследование операций и системный анализ

Научный руководитель,

кандидат физ.-мат наук,

доцент

Панкратова Я.Б.

Рецензент

Бартель Мария Владимировна

Санкт-Петербург
2021

Оглавление

[Введение 3](#_Toc72633070)

[Постановка задачи 5](#_Toc72633071)

[Обзор литературы 6](#_Toc72633072)

[Глава 1. Кооперативные игры 7](#_Toc72633073)

[Основные понятия и определения 7](#_Toc72633074)

[Пред-*N*-ядро 8](#_Toc72633075)

[α-*N*-ядро 9](#_Toc72633076)

[Глава 2. Би-кооперативные игры 11](#_Toc72633077)

[Основные понятия и определения 11](#_Toc72633078)

[SM-ядро 13](#_Toc72633079)

[α –N-ядро 14](#_Toc72633080)

[Глава 3. Программная реализация 18](#_Toc72633081)

[3.1. Описание алгоритмов и программ 18](#_Toc72633082)

[3.2. Примеры использования программы. 21](#_Toc72633083)

[Выводы 27](#_Toc72633084)

[Заключение 28](#_Toc72633085)

[Список литературы 29](#_Toc72633086)

[Приложение 30](#_Toc72633087)

Введение

Кооперативные игры — класс игр, в которых игрокам позволено заключать союзы для увеличения потенциальных выигрышей. В играх с трансферабельными полезностями данные выигрыши должны оцениваться в единицах, общих для всех игроков, и могут быть произвольно поделены между игроками. Способы распределения между всеми участниками игры общего выигрыша, полученного в результате кооперации, многочисленны и поэтому являются основным предметом изучения кооперативной теории. Эти способы распределения общего выигрыша называются решениями кооперативной игры. Наиболее изученными концепциями решения в кооперативной теории игр являются C-ядро, N-ядро и вектор Шепли . Сравнительно недавно введены новые концепции решения, такие как SM-ядро и множество α-N-ядер. Такая многочисленность решений объясняется различными наборами свойств решений, благодаря которым каждое из решений более предпочтительно в некоторых аспектах в сравнении с другими.

Данная работа посвящена изучению би-кооперативных игр и их решений. Основной особенностью и отличием би-кооперативных игр от кооперативных игр является следующее: при построении характеристической функции би-кооперативной игры рассматриваются разные типы участия игроков в формировании коалиции, которые влияют на значение характеристической функции, в отличие от ТП игр, где игрок либо вступает в коалицию, либо нет. А именно в би-кооперативных играх возможно 2 варианта участия: позитивный и негативный. Позитивный вариант участия представляет собой обычный вариант участия в распределении ресурсов между игроками, то есть игрок присоединяется к коалиции, и это приводит к увеличению характеристической функции. Отрицательный вариант участия предполагает, что при присоединении игрока к коалиции значение характеристической функции уменьшается.

Как уже было сказано выше - сравнительно недавно были введены новые решения, такие как SM-ядро и множество α-N-ядер для кооперативных игр. В данной же работе будут введены и найдены решения для α-N-ядер для би-кооперативных игр. Так же описаны некоторые свойства удовлетворяющие этим решениям.

В первой главе вводится теория для кооперативных игр, определения и теоремы которой понадобятся в следующих главах. В частности, во второй используются определения из 1 главы, чтобы переписать их на би-кооперативные игры. Так же во второй главе описан алгоритм для нахождения α-N-ядра би-кооперативной игры для N игроков и любого α ϵ R, объяснено на примере для трёх игроков. В общем случае, для большего числа N, ход решения аналогичный. В 3 главе описывается реализация этого подхода в программе в среде Matlab, а так же рассматриваются примеры, решенные с помощью этого реализованного компьютерного алгоритма.

# Постановка задачи

Целью данной работы является изучение би-кооперативных игр, некоторых свойство их решений, построение α-N-ядра для би-кооперативных игр и его нахождения с помощью программы. Для выполнения вышеописанных целей требуется сделать ряд задач, а именно:

* Разобраться в теории би-кооперативных игр
* Изучить некоторые решения би-кооперативных игр и их свойства
* Модифицировать α-N-ядро для би-кооперативных игр
* Разработать программу для нахождения α-N-ядра
* Проверить работу программы на примерах

# **Обзор литературы**

При написании данной работы была использована научная, учебно-методическая литература, а также публикации из научных изданий.

Начинал своё знакомство с теорией игр, а в частности и с кооперативными играми, с книгой авторов Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В.[3]. Отсюда были взяты основные понятия для кооперативных игр.

Основным источником для изучения би-кооперативных игр являлась публикация «A value for bi-cooperative games» [2]. Для более детального понимания би-кооперативных игр была прочитана статья «Bicooperative games» автора Bilbao J. M.[1]. Для ознакомления с возможными решениями была прочитана статья «The selectope for bicooperative games», Bilbao J. M. [4].

Для изучения алгоритма построения SM-ядра была использована публикация «The simplified modified nucleolus of a cooperative TU-game», Тарашнина С. И.[5]. На основе понятия пред-N-ядра в работах [6] были введены новые концепции решения, такие как SM-ядро и множество α-N-ядер, учитывающие разные соотношения конструктивной и блокирующей сил коалиций. В работах [5,6] были описаны алгоритмы построения SM-ядра и множество α-N-ядер для кооперативных игр, по аналогии построили новое решение α-N-ядро для би-кооперативных игр.

 Далее, в работе [7] Н. Смирнова и С. Тарашнина исследуют свойства новых решений. Для понимания о минимальных сбалансированных наборах была рассмотрена статья [8]. В статье «The nucleolus of a characteristic function game» Schmeidler D. [11] были изучены необходимые утверждения для доказательства основных свойств SM-ядра, по аналогии с которыми можно доказать аналогичные свойства и для α-N-ядра.

Глава 1. Кооперативные игры

## **Основные понятия и определения**

Для начала приведем основные понятия и определения кооперативной теории игры, которые нам пригодятся в дальнейшем. Пусть у нас имеется класс кооперативных n лиц с трансферабельными полезностями (сокращенно ТП-игры). Через *N* = {1,2,3…n} назовём конечное множество игроков, при этом не пустое. Коалицией будем называть любое подмножество S ⊆ *N*. Под характеристической функцией игры будем понимать вещественнозначную функцию υ :→ R : υ(∅) = 0. Тогда пара (*N*, υ) задает кооперативную игру с трансферабельными полезностями.

Множество всех ТП-игр с фиксированным множеством игроком N обозначим через (*N*, υ) ϵ . Тогда, полагая, что игроки образовали максимальную коалицию N, рассмотрим проблему распределения величины υ(N) между игроками. Определим x(S):

**Определение 1.1**. [7] Множеством допустимых векторов выигрышей в игре (N, υ) называется множество

(υ) = {x ϵ : x(*N*) ≤ υ(*N*)}.

**Определение 1.2**. [7] Множеством эффективно-рациональных распределений в игре (N, υ) называется множество

(υ) = {x ϵ : x(*N*) = υ(*N*)}.

**Определение 1.3.** [3] Вектор х = ( называется дележём, если он удовлетворяет двум следующим условиям:

 1) υ(i), i ϵ *N*,

2) x(*N*) = υ(*N).*

Множеством всех дележей игры (*N*, υ) будем именовать I(υ)

**Определение 1.4**. Решением на множестве игр будем называть отображение f, которое каждой игре (*N*, υ) ϵ ставит в соответствие подмножество f(υ) множества (υ).

Кооперативные игры могут иметь разные варианты решений, т.е. одноточечные и многоточечные. К примеру, вектор Шепли, пред-*N*-ядро, SM-ядро,α-N-ядро являются одноточечными, а С-ядро и множество α-*N*-ядер – многоточечные.

Далее рассмотрим один из вариантов оптимального распределения дележей в ТП-игре называемый С-ядром. Но сначала дадим ещё несколько определений.

**Определение 1.5.** Делёж х доминирует делёж у по коалиции S (), если выполн­яются два следующие условия:

 1), i ϵ S,

2) x(S) ≤ υ(S).

т.е. можно интерпретировать так: первое означает, что дележ х лучше дележа y для каждого члена коалиции, второе отражает реализуемость дележа коалиций S.

**Определение 1.6**. Делёж х доминирует делёж у (), если существует такая коалиция S ⊂ N, для которой .

**Определение 1.7**. Множество недоминируемых дележей кооперативной игры (*N*, υ) называется ее C-ядром.

C-ядро игры (*N*, υ) будем обозначать C(υ).

**Определение 1.8**.Кооперативная игра (*N*, υ) называется выпуклой, если

∀ S,T ⊂ *N* верно:

υ(S) + υ(T) ≤ υ(S ∪ T) + υ(S ∩ T).

## Пред-*N*-ядро

В этой работе дальше будут рассмотрены эксцессоподобные решения кооперативной ТП-игры. Эксцесс коалиции показывает меру неудовлетворенности коалиции своим выигрышем.

**Определение 1.9**. [5] Эксцессом коалиции S ⊆ *N* при ∀ (υ**)** называется величина .

**Определение 1.10**. [5] Пред-*N*-ядром относительно множества называется множество

**Определение 1.11**. Cбалансированным набором [8] коалиций называется набор коалиций T ⊂ N, если существуют такие , S ⊂ T, что выполняется

**Определение 1.12.** [8]Сбалансированный набор коалиций называется минимальным, если не существует его подмножества, являющегося сбалансированным.

## α-*N*-ядро

**Определение 1.13**. Для фиксированного α-эксцессом коалиции S ⊆ N относительно называется величина

**Определение 1.14**. Для фиксированного α-N-ядром игры (N,) называется множество векторов

где – вектор, компоненты которого расположены в порядке невозрастания.

**Определение 1.15.** [0,1]-N-ядром игры (N,) на множестве будем называть множество всех α-N-ядер игры (N,) для и обозначать т.е.

**Определение 1.16**. Множеством α-*N*-ядер игры (*N*,) будем называть все α-*N-*ядра игры (*N,*) для и обозначать N() т.е.

Можно заметить, что при некоторых α элементы множества ) совпадают с уже известными решениями: при α = 1 получим пред-*N*-ядро, при α = 0 имеем анти-пред-*N*-ядро, при α = 0,5 получаем SM-ядро, а множество распределений векторов при α ∈ [0, 1] представляет собой [0, 1]-*N-*ядро.

# Глава 2. Би-кооперативные игры

В прошлой главе мы рассмотрели и разобрались в кооперативных играх. В этой же опишем би-кооперативные игры.

В обычных кооперативных играх игрок имеет две опции: присоединиться к коалиции или нет. Би-кооперативные игры были введены Bilbao[2] в 2000 году, как обобщение кооперативных игр с трансферабельными полезностями, в которых игрок может участвовать положительно, отрицательно или не участвовать вовсе.

## Основные понятия и определения

Пусть N – набор игроков 1, 2, … N и Q(N) = {(S,T) | S,T ⊆ N,
 S ∩ T = ∅) } – множество пар непересекающихся коалиций.

**Определение.2.1.** [2] Под би-кооперативной игрой будем понимать пару (N,v), где N – набор игроков,v – характеристическая функция игры, под которой будем понимать вещественнозначную функцию v : Q(N) → R, v(∅,∅)= 0.
v(S,T) – ценность, когда игроки из коалиции S позитивные участники, а из коалиции T – негативные, другие же вообще не участвуют.

Мы обозначим множество би-кооперативных игр с трансферабельными полезностями с конечным множеством игроков N.

**Определение 2.2**.игрок i позитивно монотонен, если

∀ (S,T) ∈ Q(N\{i}) : v(S ∪ {i}, T) ≥ v(S,T).

**Определение 2.3**.игрок i негативно монотонен, если

∀ (S,T) ∈ Q(N\{i}) : v(S , T ∪ {i}) ≤ v(S,T).

Тогда, исходя из этих двух определений, может появиться ещё одно.

**Определение 2.4.** Игрок i называется симметричным участником, если

∀ (S,T) ∈ Q(N\{i}), v(S ∪ {i}, T) = v(S , T ∪ {i}).

т.е. выбор положительного или негативного участия для игрока i не влияет на

добавленную стоимость этого игрока для данной коалиции.

Теперь сформируем вектор Шепли для би-кооперативных ТП-игр

**Определение 2.5.** [2]Вектором Шепли для би-кооперативных игр называется:

где K ⊆ (S ∪ T),

 V(K) = v(S ∩ K, T ∩ K),
k = |K|, s = |S|, t = |T|, i ϵ K.

Принимая во внимание вышеописанное равенство для V(K), получим
V(N) = v(S ∩ N, T ∩ N) = v(S,T).

**Определение 2.6.** Множеством допустимых векторов выигрышей в игре (N, v) называется множество

где

**Определение 2.7.** Множеством эффективно-рациональных векторов выигрышей в игре (N,v) называется множество

где

**Определение 2.8.** Эксцессом коалиции K ⊆ (S ∪ T), би-кооперативной игры (N,v) при будем называть величину

(2)

где

**Определение 2.9**.N-ядро (v) би-кооперативной игры (N,v) определяется как

где – вектор эксцессов, компоненты которого расположены в порядке невозрастания.

Напомним, что двойственная игра в кооперативной игре определялась как () = (N) - (N\).

Т.к. V(N) = v(S ∩ N, T ∩ N) = v(S,T) и

V(N\K) = v(S ∩ N\K, T ∩ N\K) = v(S\K, T\K)

Тогда, исходя из этого, зададим и для би-кооперативной игры.

## SM-ядро

**Определение 2.10**.Двойственная игра (N,) к игре (N,v) задается по правилу

для всех коалиций

где, отражает конструктивную силу парной коалиции (S,T), а отражает блокирующую силу парной коалиции (S,T) соотв. K.

**Определение 2.11**.Эксцессом двойственной би-кооперативной игры (N, является (4)где

Обозначим сумму эксцессов обычной и двойственной игры как

**Определение 2.12**. The simplified modified nucleolus or SM-ядром в би-кооперативной игре называется множество

где – вектор сумм эксцессов, компоненты которого

расположены в порядке невозрастания.

**Свойства SM-ядра:**

1)Непустота (NE): .

2)Парето-оптимальность (PO): .

3)Единственность (SIVA): .

4)Обоснованность (RE): Решение удовлетворяет обоснованности, ес-

ли ∀ би-кооперативной игры (N,v) и ∀ справедливы сразу оба не

равенства: и , ∀ , где

5)Анонимность (AN): для любой перестановки π

игроков множества N кооперативной игры (N,v).

**Определение 2.13.** Игрок является болваном в би-кооперативной игре (N,v), если существует , такая, что

Константу λ можно понимать как постоянную ценность игрока при добавлении к коалиции S или T.

6)Свойство болвана (DUM): Если игрок болван в би-кооперативной игре (N,v), то тогда верно, что .

## α –N-ядро

**Определение 2.14**.α-эксцессом коалиции относительно

 для фиксированного называется величина

 (5)Здесь α – вес конструктивной силы коалиции, а (1-α) – вес блокирующей силы.

**Определение 2.15.**Для любого фиксированного , α-N ядром би-

кооперативной игры (N,v) называется множество векторов:

Напишем чем равняется α-N ядро при некоторых значениях альфа:
при α = 1 получим пред-*N*-ядро, при α = 0 имеем анти-пред-*N*-ядро,
при α = 0,5 получаем SM-ядро.

Свойства, написанные выше для SM-ядра так же верны и для α-N ядра. Доказательства очевидны, выводятся аналогичным образом, поэтому приводить их здесь не буду.

Теперь рассмотрим собственно сам алгоритм нахождения α-N ядра.

В примерах у нас задаётся характеристическая функция для би-кооперативной игры. Далее нам надо выбрать относительно какой коалиции мы будем решать. Т.е. какие игроки принимают положительное участие, а какие отрицательные. Для построения α-N ядра 3 лиц нам потребуются следующие коалиции К: {1}, {2}, {3}, {1,2}, {1,3}, {2,3}. Для каждой из них будем проводить следующий ряд действий.

По формуле (1) находим значения нашей характеристической функции, так же считаем по формуле (3) значение двойственной характеристической функции. Теперь переходим к вычислению эксцессов. Считаем эксцесс и двойственный эксцесс по формулам (2) и (4) соответственно. Далее при каком-то конкретном значение α находим значение альфа-эксцессов по формуле (5).

Теперь нужно решить системы на минимальных сбалансированных наборах. Для этого сначала выпишем как будут выглядеть это наборы для трёх игроков и для 4-х игроков (для сравнения).

Для трёх игроков:

,

,

,

,

.

Для четырёх игроков:

,

,

,

,

,

,

,

,

.

Теперь с помощью минимальных сбалансированных наборов будем решать систему. Нужно приравнять друг к другу альфа-эксцессы, входящие в этот набор коалиций – это будут одни из уравнений, а последнее уравнение - это сумма всех значений выигрышей игроков, которая приравнивается к значению V(N) = v(S, T) на выбранной коалиции (S,T). В общем виде система на минимальном сбалансированном наборе ({1}, {2}, {3}) будет представима так:

Теперь давайте рассмотрим конкретную систему на первом минимальном сбалансированном наборе ({1}, {2}, {3}) и выбранной коалиции
 (S, T) = ({1, 2, 3}, ∅) при α=0.5.

Взяв значения из примера про фермеров[2], которым надо провести воду к своим участкам, тогда система будет иметь вид:

1 - 2= 16-= 16 -

Решив эту систему, мы получим значение распределений выигрышей между игроками 1, 2, 3. Далее нам нужно прорешать оставшиеся системы, составленные на основе минимальных сбалансированных наборов.

После этого мы получим 5 решений (для трёх игроков). Затем посчитаем наши альфа эксцессы, найденные по формуле (5) с подстановкой уже найденных распределений для ,. Для каждой системы мы получим численный набор значений альфа-эксцессов, которые мы должны расположить в порядке невозрастания, а затем сравнить их лексиграфически. И выбрать лексиграфически наименьший, это и будет решением относительно нашей выбранной коалиции (S,T) и нашего конкретного альфа. Для количества игроков большего трёх решение аналогичное, однако будет больше минимальных сбалансированных наборов и систем, которые нужно будет решать для нахождения α-N-ядра.

В следующей главе это всё будет реализовано в программе. Трудность состоит в том, что в некоторых системах не хватает уравнений для единственного определения значений ,. Тогда приходится перебирать. В данный момент это простое, но действенное решение «в лоб».

# Глава 3. Программная реализация

В данной главе приводится описание алгоритма поиска α-N-ядра би-кооперативной игры 3-х лиц ∀ α. А так же основная информация по разработанным функциям и программам, которые были написаны в среде MATLAB. Исходные коды скриптов будут выложены в разделе приложение.

## 3.1. Описание алгоритмов и программ

 Будем искать α-N-ядро по уже известной нам теории, изложенной в прошлой главе. У нас имеется игра (N,v), в ней может быть сформировано коалиций так как каждый игрок может выбрать три варианта участия: позитивный, негативный или вообще не находиться в коалиции. Тогда расставим коалиции таким образом: от минимального количества игроков в ней к максимальному количеству и от наименьшего номера игрока к наибольшему. Т.е. получим вот такую последовательность коалиций (S,T) для игры трёх лиц: (∅, ∅), ({1}, ∅), (∅, {1}), ({2}, ∅), (∅, {2}), ({3}, ∅), (∅, {3}), ({1,2} ,∅), ({1}, {2}), ({2}, {1}), (∅, {1,2}), ({1,3} ,∅), ({1}, {3}), ({3}, {1}), (∅, {1,3}), ({2,3} ,∅), ({2}, {3}), ({3}, {2}), (∅, {2,3}), ({1,2,3}, ∅), ({2,3}, {1}), ({1,3}, {2}), ({1,2}, {3}), ({3}, {1,2}), ({2}, {1,3}), ({1}, {2,3}), (∅, {1,2,3}). Напомним, что S и T – это набор игроков, выбравших позитивное или негативное участие соответственно.

Теперь перейдём непосредственно к самим программам. Далее кратко опишем для чего нужна каждая программа.

**Reshenie.m**

Это основная программа, с помощью которой мы будем находить α-N-ядра би-кооперативной игры 3-х лиц ∀ α. Сначала просим пользователя ввести количество игроков. В данном случае три, но уже разработана программа и для четырёх игроков. Она просто «закомментирована», т.к. ещё до конца не проверена правильность её работы на конкретных практических примерах и поэтому не внесена в диплом. Далее требуется ввести значение характеристической функции V, соответствующей нашим расставленным коалициям. Т.е. V(1) = v(∅, ∅), V(2)= ({1}, ∅) ...V(27) = v(θ, {1,2,3}). Так как решение α-N-ядра би-кооперативной игры 3-х лиц строится относительно какой-то коалиции (S,T). Поэтому человека, использующего эту программу, попросят выбрать тип участия игроков 1,2,3. Нужно будет ввести три буквы в одинарных кавычках. Где бука «p» означает положительный тип участия игрока, а «o» - отрицательный. Например ‘opp’будет означать, что игрока 1 – выбрал положительный тип, а игроки 2 и 3 отрицательный. После этого нужно будет ввести значение для α(если же пользователь захочет вычислить для нескольких альфа, то придётся вручную задать(поменять) период для α и с каким шагом это будет делаться). Далее, исходя из выбранного человеком типа участия, мы считаем в программе эксцессы и двойственную функцию. Затем, используя другие написанные вспомогательные функции, мы считаем двойственные эксцессы и альфа-эксцессы. Следующим шагом программа решает систему на минимальных сбалансированных наборах, находит лексиграфически минимальный вектор эксцессов и выводит соответствующее ему решение. Т.к. лексиграфически минимальные векторы могут совпадать, то здесь выводится последнее найденное решение.

**DvExs.m**

Считает двойственные эксцессы, исходя из заданных ранее двойственных функций.

**alfaExs.m**

Считает альфа эксцессы, исходя из ранее заданных эксцессов и двойственных эксцессов.

**Solver3xT1.m и Solver3xT2.m**

Это две идентичные программы. Они решают систему на минимальных сбалансированных наборах и . Здесь получается найти единственное решение.

**Solver3xT3, Solver3xT4 и Solver3xT5**

Эти три программы тоже весьма идентичны. Они решают систему на минимальных сбалансированных наборах , и . Похожесть состоит в том, что у всех них не хватает уравнений для единственного решения. Т.е. точно можно найти только одно значение для какого-то конкретного игрока(зависит от минимального сбалансированного набора),а два другие игрока дают в сумме некоторый постоянный выигрыш. Следовательно, стоит решать такую задачу перебором вариантов. Найдём сумму оставшихся двух значений. Далее в цикле будем перебирать варианты распределения выигрыша между игроками, и искать лексиграфически минимальный вектор среди всех таких всевозможных распределений. Напомним, что компоненты вектора должны быть расположены в порядке невозрастания. Т.е. либо оставаться такими же, либо убывать. В данной программе установлен шаг перебора значений i=0.05. Это достаточно точный интервал, т.к. при уменьшение значения шага, не было заметно различий в конечных решениях. На самом деле, всё зависит от мощности компьютера и скорости его вычислений. У меня довольно старый, поэтому на сильно точные вычисления рассчитывать не приходится. Чтобы учесть возможность получения отрицательных значений выигрыша для игроков, было сделано расширение границ перебор от минус суммы, умноженной на два, до суммы, умноженной на два. Где под суммой подразумевается значение характеристической функции V(N)=v(S∩N, T∩N) для трёх игроков. По желанию можно расширить границы, но это только затянет вычисления и в большинстве случаев не приведёт к новым решениям. В этих трёх функциях ещё встроена другая, которую опишем ниже.

**PoiskExs3x.m**

Данный скрипт находит лексиграфически минимальный вектор из всех, которые были предложены путём перебора вариантов решений, и выводит соответствующее решение. Если таких оказалось несколько, то выводит последнее найденное. Как это всё происходит. Изначально задается численное решение, найденной функцией «eqn» в матлаб. Она решает систему уравнений символьно, потом переводим из символьных в численные переменные. Далее подаётся вектор решений для трёх игроков, благодаря нему он строит альфа-эксцессы. Затем он упорядочивает их, сравнивает поэлементно и из двух таких векторов и выбирает наименьший. И так далее.

**Solver3x.m**

Эта функция сначала запускает предыдущее Solver3x. Таким образом, на выходе из них мы имеем 5 решений, у которых вектора эксцессов упорядочены по не возрастанию, и они являются лексиграфически минимальными в своих минимальных сбалансированных наборах. Далее Solver3x.m выбирает из них лексиграфически минимальный и выдаёт его за ответ. Так же здесь запрограммировано вычисление эксцессов, при желании их тоже можно вывести на экран. Затем программа выводит на экран решение в таком виде:

 ans =

для альфа равного -2 распределение между игроками равно:

[ 13/2, 11/2, 8]

Если же мы вычисляли для конкретного одного альфа, то на этом программа завершит работу, иначе будет продолжаться цикл по другим значениям переменной альфа, и будут выводиться аналогичные результаты на экран. Можно было сделать вывод в файл или в Exsel, но у меня не так быстро считает, поэтому все результаты за время вычисления можно переносить в таблицу самостоятельно.

## 3.2. Примеры использования программы.

Рассмотрим пример из [1]. Имеются три поля и скважина. Поле игрока 1 имеет размеры 10 единиц в высоту и 5 единиц в ширину. Поля 2 и 3 являются квадратами с длинами сторон равными 5. Источником воды является скважина, расположенная посередине на границе поля 1. Целью 2 и 3 игроков является проведение труб до своих полей.

Для построения альфа-N-ядра нам нужно сначала задать нашу характеристическую функцию. В описание задачи она имеет такой вид:

v(∅, ∅) = 1,

v({1}, ∅) = v(∅, {1}) = 1,

v({2}, ∅) = v({3}, ∅) = v({1,2}, ∅) = v({1,3}, ∅) = 11,

v({2,3}, ∅) = v({1,2,3}, ∅) = 16,

v({2}, {1}) = v({3}, {1}) = v({2,3}, {1}) = 6.

и для любой коалиции (S,T) ϵ Q({2,3}) справедливо:

v(S ∪ {1}, T) = v(S ∪ T ∪ {1}, ∅) ,

v(S, T) = v(S ∪ T, ∅) ,

v(S, T ∪ {1}) = v(S ∪ {1}, T) ,

что означает, выбор положительно или отрицательного участия игроками 2 или 3 никак не влияет на общую стоимость.

В связи с этим, будем рассматривать только два типа коалиций (S,T). Одна –где первый игрок, принимает позитивное участие, вторая – где негативное. Не умаляя общности, рассмотрим ({1,2,3}, ∅), ({2,3},{1}).

В этом примере характеристическая функция, исходя из нашей расстановки всевозможных коалиций игры, выглядит следующим образом:

V = [0 1 1 11 11 11 11 11 11 6 6 11 11 6 6 16 16 16 16 16 6 16 16 6 6 16 6].

Рассмотрим α-N-ядро относительно коалиции ({1,2,3}, ∅). В ходе тестирования было найдено, что решение изменятся в пределах [0.45, 0.6]. Дальнейшее отклонение даже на 0.00001 и увеличение шага до 0.001 не меняло значений выигрыша игроков. Для наглядности в двух таблицах ниже покажем эти результаты. В одной будет интервал для альфа [0.4, 0.65] с шагом 0.01,в другой - [-0.2, 1.2] с шагом 0.1. Как видно из таблиц 1 и 2 решение больше либо равное 0.6 всегда будет равно (0, 8, 8),а решение меньшее или равное 0.4 всегда будет равно (0.5, 7.75, 7.75). Выделим значение при альфа равном 1, 0, 0.5, которые соответственно являются пред-N-ядром, анти-N-ядром и SM-ядром.

Рассмотрим α-N-ядро относительно коалиции ({2,3}, {1}). В ходе тестирования было найдено, что решение не изменятся совсем и равно всегда () Дальнейшее отклонение даже на 0.00001 и увеличение шага до 0.001 не меняло значений выигрыша игроков при любых альфа.

|  |  |  |  |
| --- | --- | --- | --- |
|  **alfa** | **x1** | **x2** | **x3** |
| 0.40 | 0.5 | 7.75 | 7.75 |
| 0.41 | 0.5 | 7.75 | 7.75 |
| 0.42 | 0.5 | 7.75 | 7.75 |
| 0.43 | 0.5 | 7.75 | 7.75 |
| 0.44 | 0.5 | 7.75 | 7.75 |
| 0.45 | 0.5 | 7.75 | 7.75 |
| 0.46 | 0,4666 | 7.7666 | 7.7666 |
| 0.47 | 0.4333 | 7.7833 | 7.7833 |
| 0.48 | 0.4 | 7.8 | 7.8 |
| 0.49 | 0.3666 | 7.8166 | 7.8166 |
| 0.50 | 0.3333 | 7.8333 | 7.8333 |
| 0.51 | 0.3 | 7.85 | 7.85 |
| 0.52 | 0.2666 | 7.8666 | 7.8666 |
| 0.53 | 0.2333 | 7.8833 | 7.8833 |
| 0.54 | 0.2 | 7.9 | 7.9 |
| 0.55 | 0.1666 | 7.9166 | 7.9166 |
| 0.56 | 0.1333 | 7.9333 | 7.9333 |
| 0.57 | 0.1 | 7.95 | 7.95 |
| 0.58 | 0.0666 | 7.9666 | 7.9666 |
| 0.59 | 0.0333 | 7.9833 | 7.9833 |
| 0.60 | 0 | 0.8 | 0.8 |
| 0.61 | 0 | 0.8 | 0.8 |
| 0.62 | 0 | 0.8 | 0.8 |
| 0.63 | 0 | 0.8 | 0.8 |
| 0.64 | 0 | 0.8 | 0.8 |
| 0.65 | 0 | 0.8 | 0.8 |

Таблица 1. Вычисление α-N-ядра на интервале [0.40, 0.65]

|  |  |  |  |
| --- | --- | --- | --- |
| alfa | x1 | x2 | x3 |
| -0.2 | 0.5 | 7.75 | 7.75 |
| -0.1 | 0.5 | 7.75 | 7.75 |
| **0** | **0.5** | **7.75** | **7.75** |
| 0.1 | 0.5 | 7.75 | 7.75 |
| 0.2 | 0.5 | 7.75 | 7.75 |
| 0.3 | 0.5 | 7.75 | 7.75 |
| 0.4 | 0.5 | 7.75 | 7.75 |
| **0.5** | **0.3333** | **7.8333** | **7.8333** |
| 0.6 | 0 | 0.8 | 0.8 |
| 0.7 | 0 | 0.8 | 0.8 |
| 0.8 | 0 | 0.8 | 0.8 |
| 0.9 | 0 | 0.8 | 0.8 |
| **1** | **0** | **0.8** | **0.8** |
| 1.1 | 0 | 0.8 | 0.8 |
| 1.2 | 0 | 0.8 | 0.8 |

Таблица 2. Вычисление α-N-ядра на интервале [-0.2, 1.2]

Для наглядности работы программы проверим её ещё на одной абстрактной задаче. Т.к. весьма подозрительным кажется, что во втором случае при (S, T) = ({2,3},{1}) получалось одно и то же решение при любом альфа.

Зададим характеристическую функцию таким образом: V(i) = i-1, где i принимает значения 1,2,3….27. Рассмотрим α-N-ядро относительно коалиции ({2,3}, {1}). Переменную альфа будем изменять в интервале [-3.2, 2,5] с шагом 0.1. По таблице 3. можем заметить, что решения изменяются от -3 до 2.3, дальнейшие выходы за границы этих значений в меньшую и большую сторону соответственно не будут изменять итоговое распределение между игроками. Значит, программа работает корректно, и просто пример приведенный в [1] оказался не очень удачным для тестирования алгоритма.

Ниже, в приложении, будут указаны исходные коды всех описанных выше программ, а пока можно рассмотреть как изменяются значения выигрышей игроков относительно коалиции ({2,3}, {1}) в зависимости от альфа.

|  |  |  |  |
| --- | --- | --- | --- |
| **alfa** | **x1** | **x2** | **x3** |
| -3.2 | 7 | 5 | 8 |
| -3.1 | 7 | 5 | 8 |
| -3 | 7 | 5 | 8 |
| -2.9 | 6.95 | 5.05 | 8 |
| -2.8 | 6.90 | 5.10 | 8 |
| -2.7 | 6.85 | 5.15 | 8 |
| -2.6 | 6.80 | 5.20 | 8 |
| -2.5 | 6.75 | 5.25 | 8 |
| -2.4 | 6.70 | 5.30 | 8 |
| -2.3 | 6.65 | 5.35 | 8 |
| -2.2 | 6.60 | 5.40 | 8 |
| -2.1 | 6.55 | 5.45 | 8 |
| -2.0 | 6.50 | 5.50 | 8 |
| -1.9 | 6.45 | 5.55 | 8 |
| -1.8 | 6.40 | 5.60 | 8 |
| -1.7 | 6.35 | 5.65 | 8 |
| -1.6 | 6.30 | 5.70 | 8 |
| -1.5 | 6.25 | 5.75 | 8 |
| -1.4 | 6.20 | 5.80 | 8 |
| -1.3 | 6.15 | 5.85 | 8 |
| -1.2 | 6.10 | 5.90 | 8 |
| -1.1 | 6.05 | 5.95 | 8 |
| -1.0 | 6 | 6 | 8 |
| -0.9 | 5.95 | 6.05 | 8 |
| -0.8 | 5.9 | 6.10 | 8 |
| -0.7 | 5.85 | 6.15 | 8 |
| -0.6 | 5.80 | 6.20 | 8 |
| -0.5 | 5.75 | 6.25 | 8 |
| -0.4 | 5.70 | 6.30 | 8 |
| -0.3 | 5.65 | 6.35 | 8 |
| -0.2 | 5.6 | 6.4 | 8 |
| -0.1 | 5.4666 | 6.3666 | 8.1666 |
| 0 | 5.3333 | 6.3333 | 8.3333 |
| 0.1 | 5.2 | 6.3 | 8.5 |
| 0.2 | 5.0666 | 6.2666 | 8.6666 |
| 0.3 | 4.9333 | 6.0333 | 8.8333 |
| 0.4 | 4.8 | 6.2 | 9 |
| 0.5 | 4.6666 | 6.1666 | 9.1666 |
| 0.6 | 4.5333 | 6.1333 | 9.3333 |
| 0.7 | 4.4 | 6.1 | 9.5 |
| 0.8 | 4.2666 | 6.0666 | 9.6666 |
| 0.9 | 4.1333 | 6.0333 | 9.8333 |
| 1. | 4 | 6 | 10 |
| 1.1 | 3.8666 | 5.9666 | 10.1666 |
| 1.2 | 3.7333 | 5.9333 | 10.3333 |
| 1.3 | 3.6 | 5.9 | 10.5 |
| 1.4 | 3.5 | 5.85 | 10.65 |
| 1.5 | 3.5 | 5.75 | 10.75 |
| 1.6 | 3.5 | 5.65 | 10.85 |
| 1.7 | 3.5 | 5.55 | 10.95 |
| 1.8 | 3.5 | 5.45 | 11.05 |
| 1.9 | 3.5 | 5.35 | 11.15 |
| 2.0 | 3.5 | 5.25 | 11.25 |
| 2.1 | 3.5 | 5.15 | 10.35 |
| 2.2 | 3.5 | 5.05 | 11.45 |
| 2.3 | 3.5 | 5 | 11.5 |
| 2.4 | 3.5 | 5 | 11.5 |
| 2.5 | 3.5 | 5 | 11.5 |

Таблица 3. Вычисление α-N-ядра на интервале [-3.2, 2.5]

# Выводы

В данной работе были выполнены все поставленные задачи. Изучены основные понятия теории би-кооперативных игр, изучен алгоритм для построения α-N-ядра для кооперативных игр и модернизирован для би-кооперативных игр. Была разработана программа в среде MATLAB для поиска α-N-ядра игры трёх лиц в общем случае. Так же была проверена правильность исполнения программы на двух примерах, результаты можно посмотреть в таблице.

Полученные результаты решались аналитически, а потом уже с помощью тестирования на компьютере. Как оказалось, распределения игроков было идентично в обоих вариантах подсчёта. Если подходить критически к рассмотрению алгоритма, то он строится частично на переборе, и для большего количества игроков это может быть проблемой, т.к. на минимальных сбалансированных наборах придётся перебирать не две, три, а иногда и более переменных.

В дальнейшем исследовании в этой области можно будет изучить, доказать свойства относительно α-N-ядра и строить более быстрые и точные алгоритмы для поиска решения. Как, например, это сделано для кооперативных игр, где α-N-ядро игры (N,υ) может быть найдено как пред-N-ядро игры (N, ,где , .

# Заключение

В данной работе изучены основные понятия теории би-кооперативных игр, изучен алгоритм для построения α-N-ядра для кооперативных игр и модернизирован для би-кооперативных игр. Была разработана программа в среде MATLAB для поиска α-N-ядра игры трёх лиц в общем случае. Так же была проверена правильность исполнения программы на двух примерах.

Список литературы

1. Bilbao J. M. et al. Bicooperative games //Cooperative games on combinatorial structures. Kluwer Acad., 2000, С. 131-295.

2. Labreuche C., Grabisch M. M. A value for bi-cooperative games // Int J Game Theory, 2008, T. 37, No.3, C. 409-438.

3. Петросян Л. А., Зенкевич Н. А., Шевкопляс Е. В. Теория игр: учебник СПб.: БХВ-Петербург, 2012. С. 432.

4. Bilbao J. M., Jimenez N., Lopez J. J. The selectope for bicooperative games //European Journal of Operational Research, 2010, Т. 204, No. 3, С. 522-532.

5. Tarashnina S. I., The simplified modified nucleolus of a cooperative TUgame //Top, 2011, T.19, C. 150-166.

6. Смирнова Н. В., Тарашнина С. И. О свойствах решений кооперативных игр с трансферабельными полезностями // Известия высших учебных заведений. Математика. 2016. № 6. С. 73-85.

7. Смирнова Н. В., Тарашнина С. И., Об одном обобщении N-ядра в кооперативных играх//Дискретный анализ и исследование операций, 2011, Т.18, в.4, С. 77-93.

8. Сбалансированные игры, <http://xity.narod.ru/game/balance.pdf>.

9. Schmeidler D., The nucleolus of a characteristic function game//SIAM J, 1969, T.17, C. 1163-1170.

# Приложение

**Reshenie.m**

clc;

clear ;

% Вводим сначала число игроков, пока 3, модернизировать можно будет и дл9 4 дл9 общего случа9

n = input('Введите число игроков ');

switch n

 case 3

 for i=1:3^n

 % V(i)=i-1;

 V(i) = input('Ввведите значение дл9 характеристич. функции ' );

 end

%V=[0 1 1 11 11 11 11 11 11 6 6 11 11 6 6 16 16 16 16 16 6 16 16 6 6 16 6];

length(V);

tipych = input('Введите тип участи9 трёх игроков(в ковычках подр9д без пробелов): "p", если положительный, "o", если отрицательный ');

switch tipych

 case 'ppp'

 %alfa = input ('Введите значение альфа ');

for alfa=0.35:0.01:0.65

syms x1 x2 x3

summa=V(20);

%считаем эксцессы

e1=V(2)-x1;

e2=V(4)-x2;

e3=V(6)-x3;

e12=V(8)-x1-x2;

e13=V(12)-x1-x3;

e23=V(16)-x2-x3;

%считаем двойственную фунцию

DV1=V(20)-V(16);

DV2=V(20)-V(12);

DV3=V(20)-V(8);

DV12=V(20)-V(6);

DV13=V(20)-V(4);

DV23=V(20)-V(2);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'opp'

 % alfa = input ('Введите значение альфа ');

for alfa=-3:0.1:3

syms x1 x2 x3

summa=V(21);

%считаем эксцессы

e1=V(3)-x1;

e2=V(4)-x2;

e3=V(6)-x3;

e12=V(10)-x1-x2;

e13=V(14)-x1-x3;

e23=V(16)-x2-x3;

%считаем двойственную фунцию

DV1=V(21)-V(16);

DV2=V(21)-V(14);

DV3=V(21)-V(10);

DV12=V(21)-V(6);

DV13=V(21)-V(4);

DV23=V(21)-V(3);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсессы

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'pop'

 % alfa = input ('Введите значение альфа ');

for alfa=-10:0.5:10

syms x1 x2 x3

summa=V(22);

%считаем эксцессы

e1=V(2)-x1;

e2=V(5)-x2;

e3=V(6)-x3;

e12=V(9)-x1-x2;

e13=V(12)-x1-x3;

e23=V(18)-x2-x3;

%считаем двойственную фунцию

DV1=V(22)-V(18);

DV2=V(22)-V(12);

DV3=V(22)-V(9);

DV12=V(22)-V(6);

DV13=V(22)-V(5);

DV23=V(22)-V(2);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'ppo'

 %alfa = input ('Введите значение альфа ');

for alfa=-1.5:0.1:1.5

syms x1 x2 x3

summa=V(23);

%считаем эксцессы

e1=V(2)-x1;

e2=V(4)-x2;

e3=V(7)-x3;

e12=V(8)-x1-x2;

e13=V(13)-x1-x3;

e23=V(17)-x2-x3;

%считаем двойственную фунцию

DV1=V(23)-V(17);

DV2=V(23)-V(13);

DV3=V(23)-V(8);

DV12=V(23)-V(7);

DV13=V(23)-V(4);

DV23=V(23)-V(2);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'oop'

 %alfa = input ('Введите значение альфа ');

for alfa=-10:0.5:10

syms x1 x2 x3

summa=V(24);

%считаем эксцессы

e1=V(3)-x1;

e2=V(5)-x2;

e3=V(6)-x3;

e12=V(11)-x1-x2;

e13=V(14)-x1-x3;

e23=V(18)-x2-x3;

%считаем двойственную фунцию

DV1=V(24)-V(18);

DV2=V(24)-V(14);

DV3=V(24)-V(11);

DV12=V(24)-V(6);

DV13=V(24)-V(5);

DV23=V(24)-V(3);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'opo'

 %alfa = input ('Введите значение альфа ');

for alfa=-10:0.5:10

syms x1 x2 x3

summa=V(25);

%считаем эксцессы

e1=V(3)-x1;

e2=V(4)-x2;

e3=V(7)-x3;

e12=V(10)-x1-x2;

e13=V(15)-x1-x3;

e23=V(17)-x2-x3;

%считаем двойственную фунцию

DV1=V(25)-V(17);

DV2=V(25)-V(15);

DV3=V(25)-V(10);

DV12=V(25)-V(7);

DV13=V(25)-V(4);

DV23=V(25)-V(3);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'poo'

% alfa = input ('Введите значение альфа ');

for alfa=-10:0.5:10

syms x1 x2 x3

summa=V(26);

%считаем эксцессы

e1=V(2)-x1;

e2=V(5)-x2;

e3=V(7)-x3;

e12=V(9)-x1-x2;

e13=V(13)-x1-x3;

e23=V(19)-x2-x3;

%считаем двойственную фунцию

DV1=V(26)-V(19);

DV2=V(26)-V(13);

DV3=V(26)-V(9);

DV12=V(26)-V(7);

DV13=V(26)-V(5);

DV23=V(26)-V(2);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

 case 'ooo'

 % alfa = input ('Введите значение альфа ');

for alfa=-10:0.5:10

syms x1 x2 x3

summa=V(27);

%считаем эксцессы

e1=V(3)-x1;

e2=V(5)-x2;

e3=V(7)-x3;

e12=V(11)-x1-x2;

e13=V(15)-x1-x3;

e23=V(19)-x2-x3;

%считаем двойственную фунцию

DV1=V(27)-V(19);

DV2=V(27)-V(15);

DV3=V(27)-V(11);

DV12=V(27)-V(7);

DV13=V(27)-V(5);

DV23=V(27)-V(3);

%считаем двойстенный эксцессы

DvExs

%cчитаем альфа экцсесс

alfaExs

%на минмальных сбалансировванных наборах решаем систему, строем экцсессы

Solver3x

end

end

 case 4 %модернизаци9 на 4-х игроков

 %Reshenie4x

End

**DvExs.m**

%считаем двойственные эксцессы

De1=DV1-x1;

De2=DV2-x2;

De3=DV3-x3;

De12=DV12-x1-x2;

De13=DV13-x1-x3;

De23=DV23-x2-x3;

**alfaExs.m**

%суммируем эксцессы, т.е. считаем альфа эксцесс

E1=alfa\*e1+(1-alfa)\*De1;

E2=alfa\*e2+(1-alfa)\*De2;

E3=alfa\*e3+(1-alfa)\*De3;

E12=alfa\*e12+(1-alfa)\*De12;

E13=alfa\*e13+(1-alfa)\*De13;

E23=alfa\*e23+(1-alfa)\*De23;

**Solver3xT1.m**

clear x1 x2 x3

syms x1 x2 x3

alfaExs

eqns = [E1==E2,E2==E3,E1==E3,x1+x2+x3==summa];

Resh = solve(eqns,[x1 x2 x3]);

%преобразовываем наши х1 х2 х3 в числовые переменные и находим решение х и вектор эксцессов ему соотв.

x11=Resh.x1;

x22=Resh.x2;

x33=Resh.x3;

clear x1 x2 x3

x1=x11;

x2=x22;

x3=x33;

X1=[x1,x2,x3];

Exs11=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 **Solver3xT2.m**

clear x1 x2 x3

syms x1 x2 x3

alfaExs

eqns = [E12==E13,E13==E23,E23==E12,x1+x2+x3==summa];

Resh = solve(eqns,[x1 x2 x3]);

%преобразовываем наши х1 х2 х3 в числовые переменные и находим решение х и вектор эксцессов ему соотв.

x11=Resh.x1;

x22=Resh.x2;

x33=Resh.x3;

clear x1 x2 x3

x1=x11;

x2=x22;

x3=x33;

X2=[x1,x2,x3];

Exs22=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

**Solver3xT3.m**

clear x1 x2 x3

syms x1 x2 x3

alfaExs

eqns = [E1==E23,x1+x2+x3==summa];

Resh = solve(eqns,[x1 x2 x3]);

%преобразовываем наши х1 х2 х3 в числовые переменные и находим решение х и вектор эксцессов ему соотв.

x11=Resh.x1;

x22=Resh.x2;

x33=Resh.x3;

clear x1 x2 x3

x1=x11;

x2=x22;

x3=x33;

XX=[x1,x2,x3];

xn=x2+x3;

EX=subs([E1,E2,E3,E12,E13,E23]);

D=sort(EX,'descend');

for i=-2\*summa:0.05:2\*summa %вернуть потом на 0.01 сотую, когда дорешаю + если будет врем9

 x1=x11;

 x2=i;

 x3=xn-i;

 E1=alfa\*e1+(1-alfa)\*De1;

 E2=alfa\*e2+(1-alfa)\*De2;

 E3=alfa\*e3+(1-alfa)\*De3;

 E12=alfa\*e12+(1-alfa)\*De12;

 E13=alfa\*e13+(1-alfa)\*De13;

 E23=alfa\*e23+(1-alfa)\*De23;

 EX=subs([E1,E2,E3,E12,E13,E23]);

 B=sort(EX,'descend');

 PoiskExs3x

end

Exs33=EX2;

X3=XX;

**Solver3xT4.m**

clear x1 x2 x3

syms x1 x2 x3

alfaExs

eqns = [E2==E13,x1+x2+x3==summa];

Resh = solve(eqns,[x1 x2 x3]);

%преобразовываем наши х1 х2 х3 в числовые переменные и находим решение х и вектор эксцессов ему соотв.

x11=Resh.x1;

x22=Resh.x2;

x33=Resh.x3;

clear x1 x2 x3

x1=x11;

x2=x22;

x3=x33;

XX=[x1,x2,x3];

xn=x1+x3;

EX=subs([E1,E2,E3,E12,E13,E23]);

D=sort(EX,'descend');

for i=-2\*summa:0.05:2\*summa %вернуть потом на 0.01 сотую, когда дорешаю + если будет врем9

 x1=i;

 x2=x22;

 x3=xn-i;

 E1=alfa\*e1+(1-alfa)\*De1;

 E2=alfa\*e2+(1-alfa)\*De2;

 E3=alfa\*e3+(1-alfa)\*De3;

 E12=alfa\*e12+(1-alfa)\*De12;

 E13=alfa\*e13+(1-alfa)\*De13;

 E23=alfa\*e23+(1-alfa)\*De23;

 EX=subs([E1,E2,E3,E12,E13,E23]);

 B=sort(EX,'descend');

 PoiskExs3x

end

Exs44=EX2;

X4=XX;

**Solver3xT5.m**

clear x1 x2 x3

syms x1 x2 x3

alfaExs

eqns = [E3==E12,x1+x2+x3==summa];

Resh = solve(eqns,[x1 x2 x3]);

%преобразовываем наши х1 х2 х3 в числовые переменные и находим решение х и вектор эксцессов ему соотв.

x11=Resh.x1;

x22=Resh.x2;

x33=Resh.x3;

clear x1 x2 x3

x1=x11;

x2=x22;

x3=x33;

XX=[x1,x2,x3];

xn=x1+x2;

EX=subs([E1,E2,E3,E12,E13,E23]);

D=sort(EX,'descend');

for i=-2\*summa:0.05:2\*summa %вернуть потом на 0.01 сотую, когда дорешаю + если будет врем9

 x1=i;

 x2=xn-i;

 x3=x33;

 E1=alfa\*e1+(1-alfa)\*De1;

 E2=alfa\*e2+(1-alfa)\*De2;

 E3=alfa\*e3+(1-alfa)\*De3;

 E12=alfa\*e12+(1-alfa)\*De12;

 E13=alfa\*e13+(1-alfa)\*De13;

 E23=alfa\*e23+(1-alfa)\*De23;

 EX=subs([E1,E2,E3,E12,E13,E23]);

 B=sort(EX,'descend');

 PoiskExs3x

 end

Exs55=EX2;

X5=XX;

**PoiskExs3x.m**

if (B(1)<D(1))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

elseif (B(1)==D(1))

 if(B(2)<D(2))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 elseif (B(2)==D(2))

 if (B(3)<D(3))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 elseif (B(3)==D(3))

 if (B(4)<D(4))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 elseif (B(4)==D(4))

 if (B(5)<D(5))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 elseif (B(5)==D(5))

 if (B(6)<=D(6))

 XX=[x1,x2,x3];

 D=sort(EX,'descend');

 EX2=sort(subs([E1,E2,E3,E12,E13,E23]),'descend');

 end

 end

 end

 end

 end

end

 **Solver3x.m**

%объедин9ем все наши мин. сбалансированные наборы в один решатель.

Solver3xT1

Solver3xT2

Solver3xT3

Solver3xT4

Solver3xT5

E=[Exs11;Exs22;Exs33;Exs44;Exs55];

X=[X1;X2;X3;X4;X5];

%вычисл9ем к какому из решенных мин. сбалансированных наборов принадлежит лексиграфически минимальный эксцесс

%и выводим соответствующее ему разделение дл9 игроков.

lexmin=[inf inf inf inf inf inf];

k=1; %число лексиграфически одинаковых эксцессов, и значит возможно различных решений.т.е.т.к. пишет последнее решение в ответ

for i=1:1:5

 EEE=E(i,:);

 ind = find(EEE ~= lexmin);

 if ( isempty(ind)==1 )

 lexmin=EEE;

 Xkon=X(i,:);

 k=k+1;

 else

 if ( EEE(ind(1)) < lexmin(ind(1)) )

 lexmin=EEE;

 Xkon=X(i,:);

 k=1;

 end

 end

end

sprintf('для альфа равного %d распределение между игроками равно:',alfa)

disp(Xkon)

E;

 clear x1 x2 x3