Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления» КАФЕДРА «Информационная безопасность»

**ЛАБОРАТОРНАЯ РАБОТА 1:**

**«Теория игр**

**и исследование операций»**

**Аналитический и численный методы**

**решения антагонистической игры в смешанных стратегиях**

**ВАРИАНТ: 4**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент: |  |  | Гончарова М.К.  (И.О. Фамилия) |
| Преподаватель: |  |  | Коннова Н.С.  (И.О. Фамилия) |

2020 г.

Цель работы:

Изучить аналитический (обратной матрицы) и численный (Брауна-Робинсон) методы нахождения смешанных стратегий в антагонистической игре двух лиц в нормальной форме.

Постановка задачи:

Задана матрица игры:

Найдите цену игры и оптимальные стратегии обоих игроков методами обратной матрицы и Брауна-Робинсон. Сравните полученные результаты.

Ход работы:

Решение задачи аналитическим методом приведено на рисунке 1.

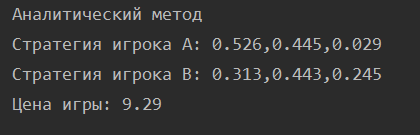
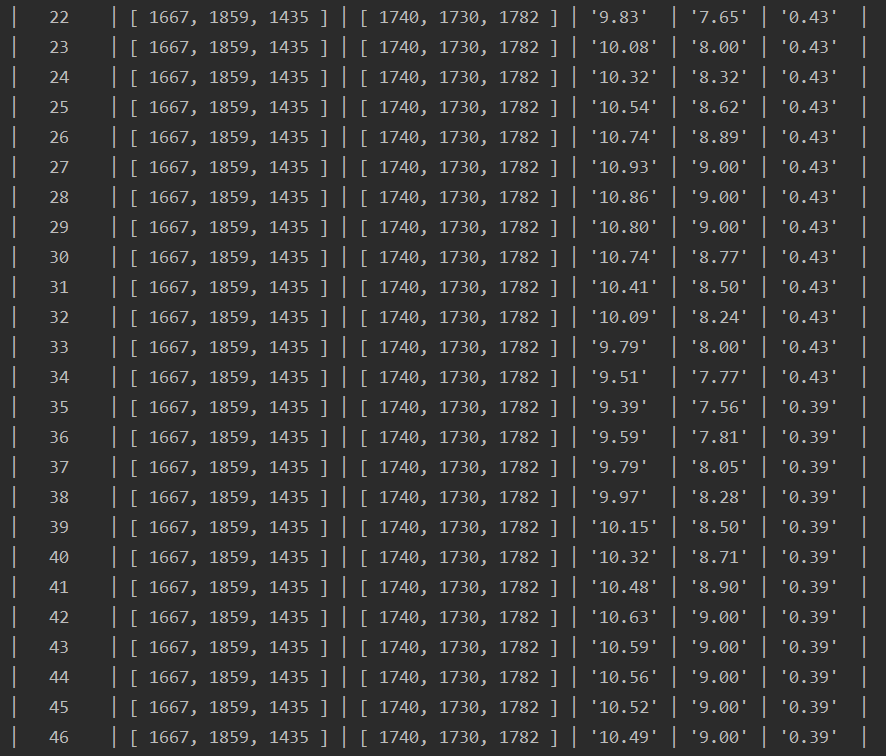
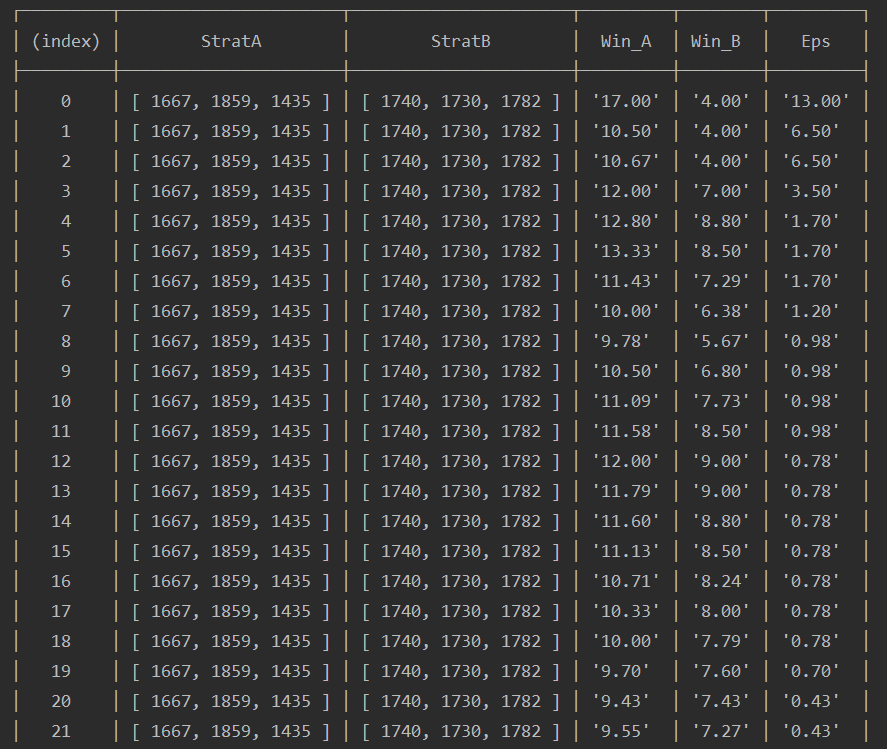


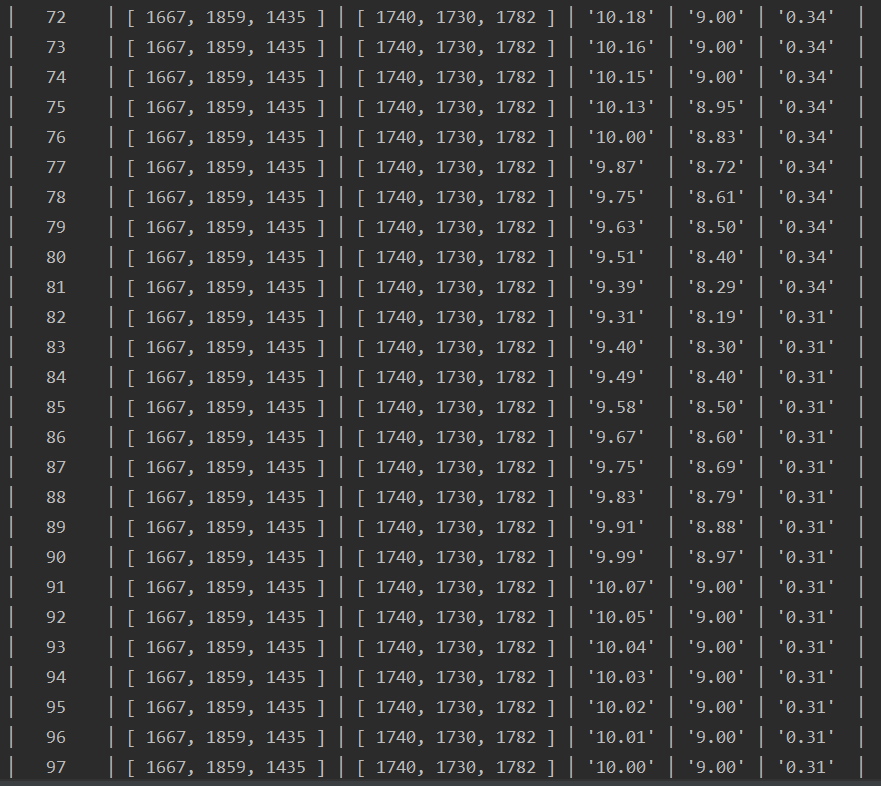
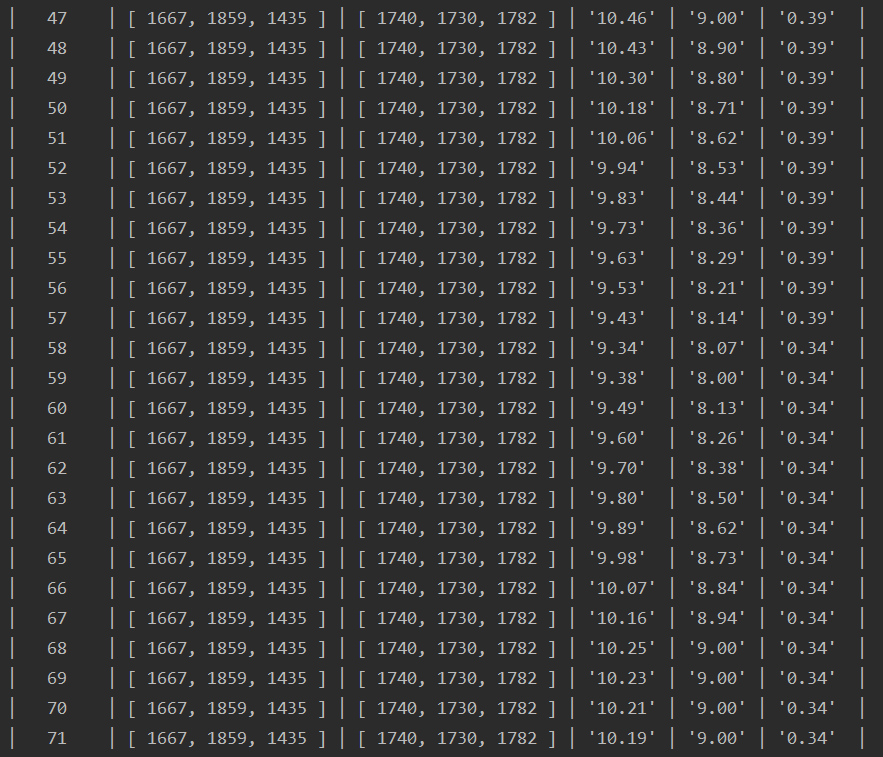
Рисунок 1 – решение задачи аналитическим методом

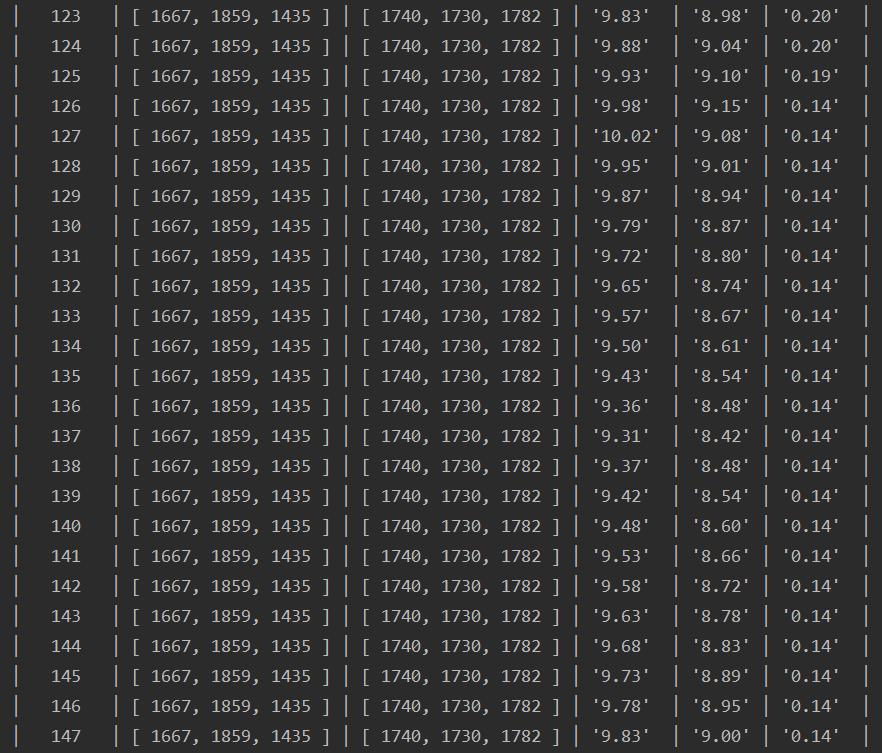
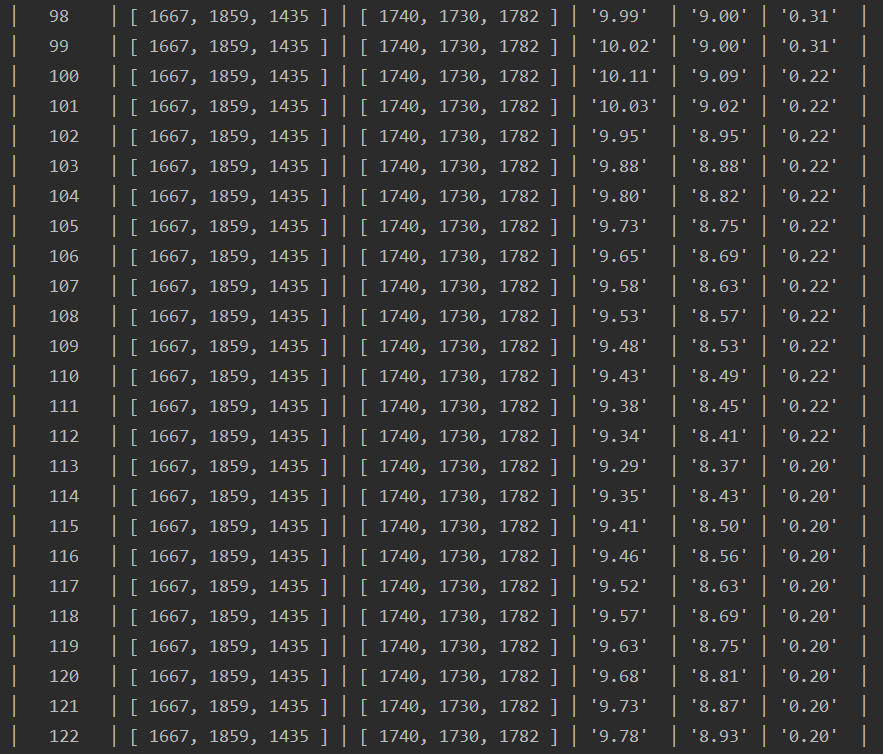
Данное решение было посчитано по формулам

где

Решение задачи методом Брауна – Робинсон представлено на рисунке 2.







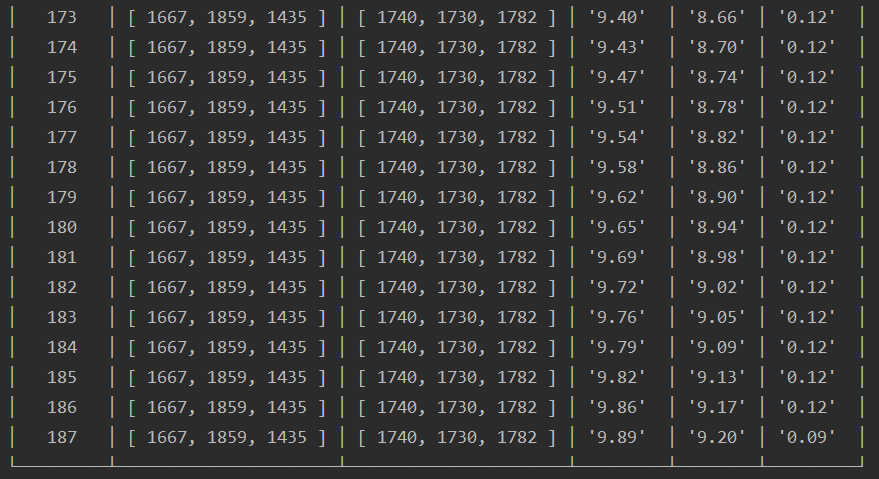
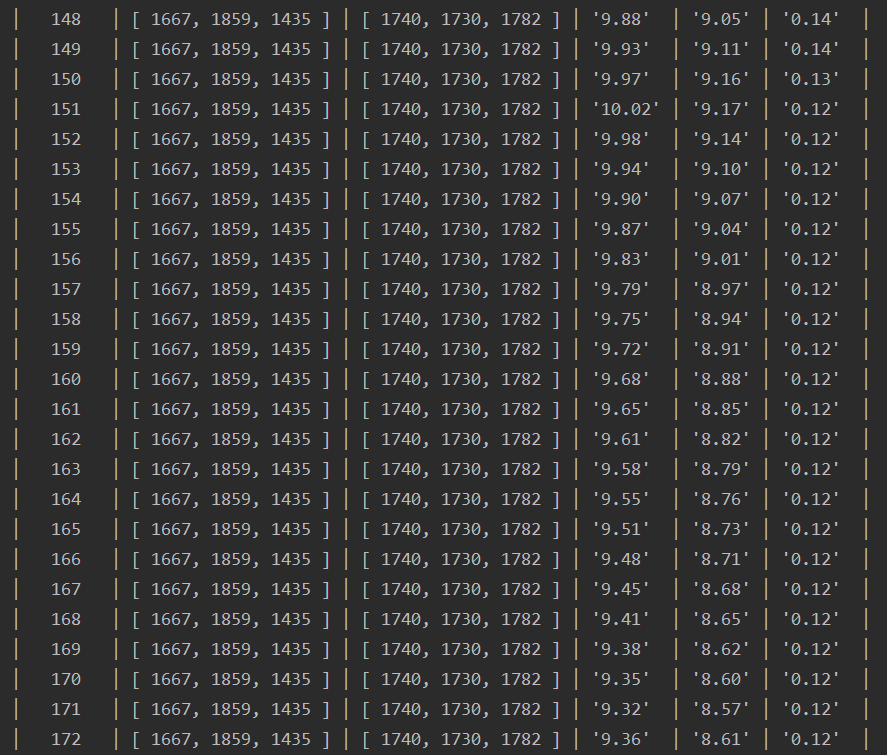


Рисунок 2 - Решение задачи методом Брауна – Робинсон

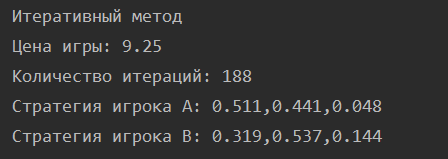
Получившиеся результаты представлены на рисунке 3. 

Рисунок 3 – результат численного метода

Предельная оценка цены игры была посчитана, как среднее арифметическое:

На рисунке 4 представлен график зависимости величины погрешности от номера итерации. Можно видеть, что с увеличением номера итерации погрешность уменьшается и график зависимости имеет гиперболическую форму. Итерационным методом можно приблизить величину ошибки к сколь угодно малому значению. В нашей задаче величина ошибки не превышает .

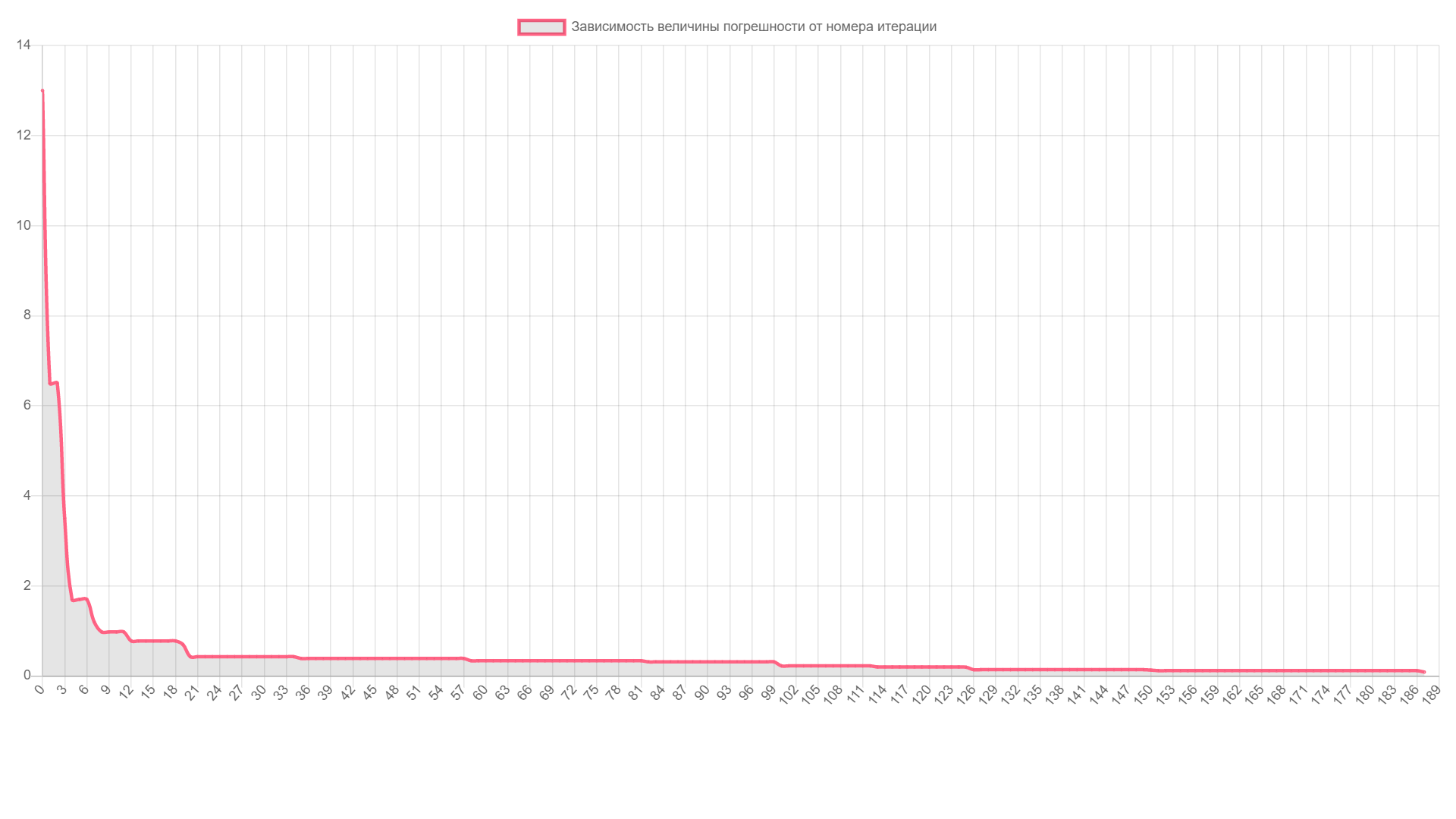


Рисунок 4 – график зависимости величины погрешности от номера итерации

Для того, чтобы убедиться, что верхняя и нижняя цены игры с каждой итерацией приближаются к одному значению, построим графики (рисунок 5) зависимости данных величин от номера итерации.

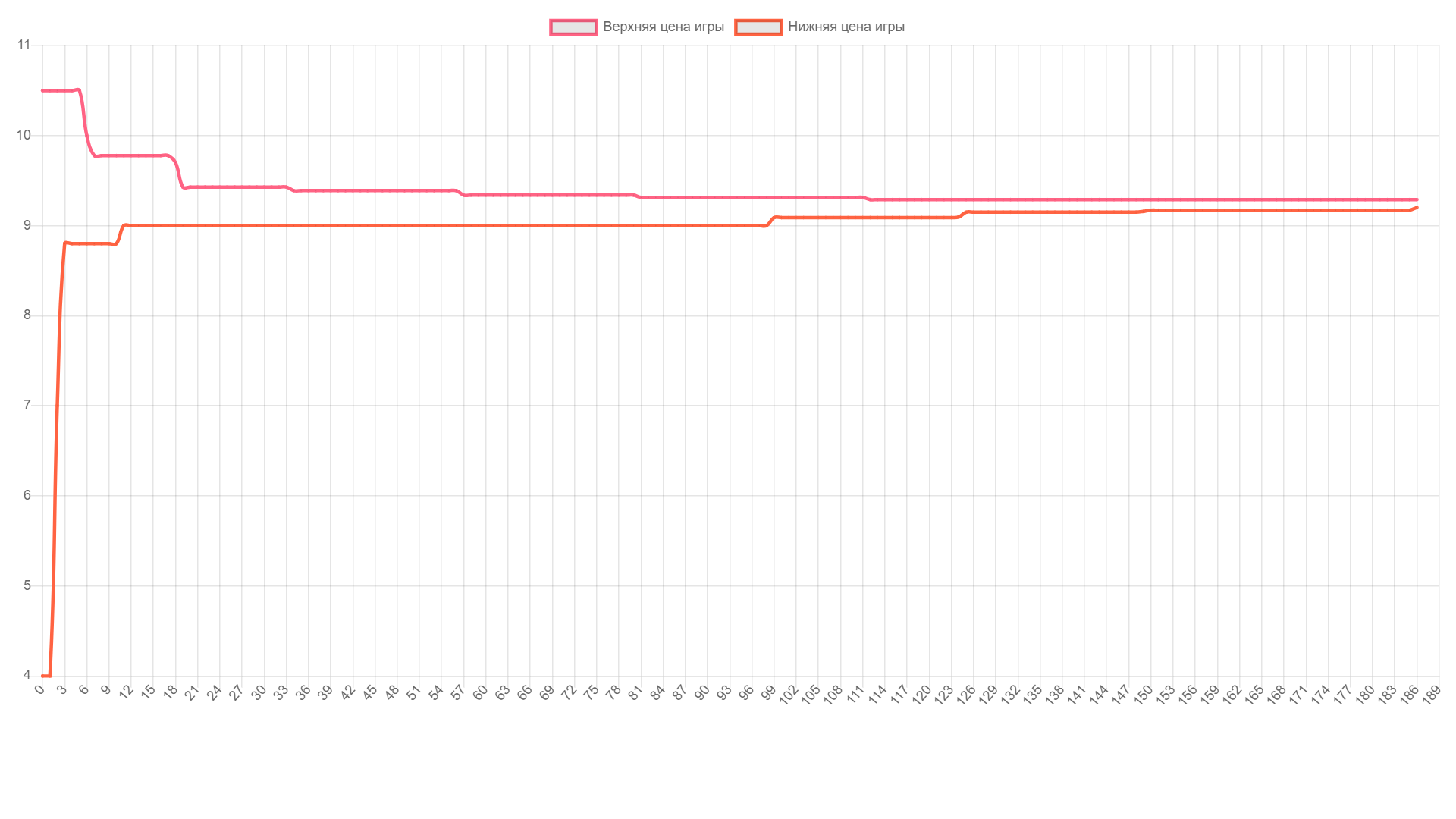


Рисунок 5 – график зависимости верхней и нижней цены игры от номера итерации

Можно заметить, что, при увеличении количества итераций в численном методе Брауна-Робинсон, разница между верхней и нижней ценой игры уменьшается, что подтверждает теорему:

Вывод:

В ходе проделанной лабораторной работы было приведено два решения (аналитическое и численное) антагонистической игры в смешанных стратегиях. Цена игры, посчитанная аналитическим методом, отличается от цены игры, которая посчитана численным методом, на 0.43%. Таким образом, несмотря на то, что метод Брауна-Робинсон является приближенным методом решения, цена игры отличается на незначительную величину. Также, преимуществом данного метода является его применимость к произвольной игре, а аналитический метод применим лишь к вполне смешанным играм.

ПРИЛОЖЕНИЕ А

Файл lab1.js:

let ***matrix*** = [[17, 4, 9], [0, 16, 9], [12, 2, 19]];  
  
let ***x0*** = [17, 4, 9];  
let ***x1*** = [0, 16, 9];  
let ***x2*** = [12, 2, 19];  
  
let ***x*** = [***x0***, ***x1***, ***x2***];  
  
let ***y0*** = [17, 0, 12];  
let ***y1*** = [4, 16, 2];  
let ***y2*** = [9, 9, 19];  
  
let ***y*** = [***y0***, ***y1***, ***y2***];  
  
let ***strategyA*** = [...***y0***];//выигрыши А, при условии выбора y1  
let ***strategyB*** = [...***x0***];//проигрыши В, при условии выбора x1  
let ***wins*** = [];  
let ***looses*** = [];  
let ***eps*** = 13;  
let ***counter*** = 1;  
let ***epsilons*** = [];  
let ***priceA*** = [];  
let ***priceB*** = [];  
  
let ***strategyAVector*** = {  
 0: 1,  
 1: 0,  
 2: 0  
};  
let ***strategyBVector*** = {  
 0: 1,  
 1: 0,  
 2: 0  
};  
  
// При равенстве элементов в строке, выбираем следующую стратегию рандомно  
function getRandomInt(max) {  
 return ***Math***.floor(***Math***.random() \* ***Math***.floor(max));  
}  
  
// Выбираем максимальный выигрыш  
  
function findMaxWin(strategy) {  
 let maxWin = strategy[0];  
 let maxIdx = 0;  
 if (strategy[0] === strategy[1] && strategy[1] === strategy[2]) {  
 maxIdx = getRandomInt(3);  
  
 } else {  
 strategy.forEach(function (item, idx) {  
 if (item >= maxWin) {  
 maxWin = item;  
 maxIdx = idx;  
  
 }  
  
 })  
 }  
  
 ***wins***.push(strategy[maxIdx] / ***counter***);  
 ***strategyAVector***[maxIdx]++;  
 return maxIdx;  
  
}  
  
// Выбираем минимальный проигрыш  
function findMinLoose(strategy) {  
 let minLoose = strategy[0];  
 let minIdx = 0;  
 if (strategy[0] === strategy[1] && strategy[1] === strategy[2]) {  
 minIdx = getRandomInt(3);  
 } else {  
 strategy.forEach(function (item, idx) {  
 if (item <= minLoose) {  
 minLoose = item;  
 minIdx = idx;  
 }  
  
 })  
  
 }  
  
 ***looses***.push(strategy[minIdx] / ***counter***);  
 ***strategyBVector***[minIdx]++;  
 return minIdx;  
}  
//Вычисляем эпсилон  
function findE() {  
 return ***eps*** = ***Math***.min(...***wins***) - ***Math***.max(...***looses***);  
  
}  
  
//Суммируем стратегии  
function sum(strategy, neutrality) {  
  
 for (let i = 0; i < 3; i++) {  
 strategy[i] += neutrality[i];  
  
  
 }  
 return strategy;  
  
}  
  
let ***table*** = [];  
  
function algorithm() {  
  
 let k = findMaxWin(***strategyA***);  
 let l = findMinLoose(***strategyB***);  
 ***eps*** = findE();  
 ***epsilons***.push(***eps***);  
 ***table***.push({  
 StratA: ***strategyA***,  
 StratB: ***strategyB***,  
 Win\_A: ***wins***[***wins***.length - 1].toFixed(2),  
 Win\_B: ***looses***[***looses***.length - 1].toFixed(2),  
 Eps: ***eps***.toFixed(2)  
 });  
  
  
 while (***eps*** > 0.1) {  
 ***counter***++;  
 ***strategyA*** = sum(***strategyA***, ***y***[l]);  
 ***strategyB*** = sum(***strategyB***, ***x***[k]);  
  
 k = findMaxWin(***strategyA***);  
 l = findMinLoose(***strategyB***);  
 ***eps*** = findE();  
 ***epsilons***.push(***eps***);  
  
 let minWin = ***Math***.min(...***wins***);  
 let maxLoose = ***Math***.max(...***looses***);  
 ***priceA***.push(minWin);  
 ***priceB***.push(maxLoose);  
  
 ***table***.push({  
 StratA: ***strategyA***,  
 StratB: ***strategyB***,  
 Win\_A: ***wins***[***wins***.length - 1].toFixed(2),  
 Win\_B: ***looses***[***looses***.length - 1].toFixed(2),  
 Eps: ***eps***.toFixed(2)  
 })  
 }  
 ***strategyAVector***[k]--;  
 ***strategyBVector***[l]--;  
  
  
 let minWin = ***Math***.min(...***wins***);  
 let maxLoose = ***Math***.max(...***looses***);  
  
  
 ***console***.table(***table***);  
 let priceAverage = (minWin + maxLoose) / 2;  
  
 let vecA = [(***strategyAVector***[0] / ***counter***).toFixed(3), (***strategyAVector***[1] / ***counter***).toFixed(3), (***strategyAVector***[2] / ***counter***).toFixed(3)];  
 let vecB = [(***strategyBVector***[0] / ***counter***).toFixed(3), (***strategyBVector***[1] / ***counter***).toFixed(3), (***strategyBVector***[2] / ***counter***).toFixed(3)];  
  
  
 ***console***.log('Итеративный метод ' + '\n'  
 + 'Цена игры: ' + priceAverage.toFixed(2) + '\n' +  
 'Количество итераций: ' + ***counter*** + '\n' +  
 'Стратегия игрока А: ' + vecA + '\n' +  
 'Стратегия игрока В: ' + vecB);  
  
  
 let cU = [120 / 3566, 170 / 3566, 94 / 3566]; //Результат умножения вектора u на матрицу, обратную к С  
 let uC = [202 / 3566, 171 / 3566, 11 / 3566]; //Результат умножения матрицы, обратной к С на вектор u  
 let uCu = [uC[0] + uC[1] + uC[2]]; //Вычисление знаменателя  
 let xAn = [(uC[0] / uCu).toFixed(3), (uC[1] / uCu).toFixed(3), (uC[2] / uCu).toFixed(3)];  
 let yAn = [(cU[0] / uCu).toFixed(3), (cU[1] / uCu).toFixed(3), (cU[2] / uCu).toFixed(3)];  
 let cAn = [(1 / uCu).toFixed(2)];  
  
  
 ***console***.log('Аналитический метод\n' +  
 'Стратегия игрока А: ' + xAn + '\n' +  
 'Стратегия игрока B: ' + yAn + '\n' +  
 'Цена игры: ' + cAn);  
  
 return ***eps***;  
  
  
}  
  
algorithm();

ПРИЛОЖЕНИЕ В

Файл graphics.html

<!DOCTYPE html>  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Graph</title>  
 <script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>  
 <script src="lab1.js"></script>  
  
</head>  
<body>  
<canvas id="myChart"></canvas>  
  
<script>  
 let ***numbers*** = [];  
 let ***i*** = 0;  
  
 while (***i*** < 190) {  
 ***numbers***.push(***i***);  
 ***i***++;  
 }  
  
 let ***ctx*** = ***document***.getElementById('myChart').getContext('2d');  
   
 //Строим график зависимости величины погрешности от номера итерации  
 let ***chart*** = new Chart(***ctx***, {  
 type: 'line',  
 data: {  
 labels: ***numbers***,  
 datasets: [{  
 label: 'Зависимость величины погрешности от номера итерации',  
  
 //backgroundColor: 'rgb(255, 99, 132)',  
 borderColor: 'rgb(255, 99, 132)',  
 data: ***epsilons***,  
 pointRadius: 1,  
  
 }]  
 },  
   
 });  
   
 //Строим графики зависимости цены игры от номера итерации  
 let ***mixedChart*** = new Chart(***ctx***, {  
 type: 'line',  
 data: {  
 datasets: [{  
 label: 'Верхняя цена игры',  
 data: ***priceA***,  
 pointRadius: 1,  
 borderColor: 'rgb(255, 99, 132)',  
 fill: false  
  
 }, {  
 label: 'Нижняя цена игры',  
 data: ***priceB***,  
 type: 'line',  
 pointRadius: 1,  
 borderColor: 'rgb(255, 99, 66)',  
 fill: false  
  
 }],  
 labels: ***numbers*** },  
  
 });  
  
</script>  
  
</body>  
</html>