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 2:**

**«Теория игр**

**и исследование операций»**

**Аналитический и численный методы**

**решения непрерывной выпукло-вогнутой антагонистической игры**

**ВАРИАНТ: 4**

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

2020 г.

Цель работы:

Найти оптимальные стратегии непрерывной выпукло-вогнутой антагонистической игры аналитическим и численным методами.

Постановка задачи:

Задана игра со следующими параметрами:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a | b | c | d | e |
| -15 |  | 40 | -12 | -24 |

Функция ядра имеет вид:

Найдите цену игры и оптимальные чистые стратегии обоих игроков аналитическим и численным методами. Сравните полученные результаты.

Ход работы:

Аналитическое решение:

Проверка выполнимости условий для принадлежности игры к классу выпукло-вогнутых:

Найдём производные функции ядра по каждой переменной:

После приравнивания производных к нулю получим:

Учитывая, что х и y должны быть неотрицательными, для оптимальных стратегий соответственно имеем:

Найдём общее решение этих уравнений. Для этого подставим в выражение для y выражение для х:

Подставим получившееся значение у в выражение для х:

Получившееся значение х подставим в выражение для y, чтобы выполнить проверку:

Найдём седловую точку игры, подставив получившиеся значения х и у в функцию ядра:

Далее найдем оптимальные чистые стратегии и цену игры численным методом. Воспользуемся методом аппроксимации функции выигрышей на сетке. Параметр разбиения . С помощью программы найдем решения при различном шаге сетки для исходной задачи. Ни рисунках 1-5 представлены результаты вычислений.

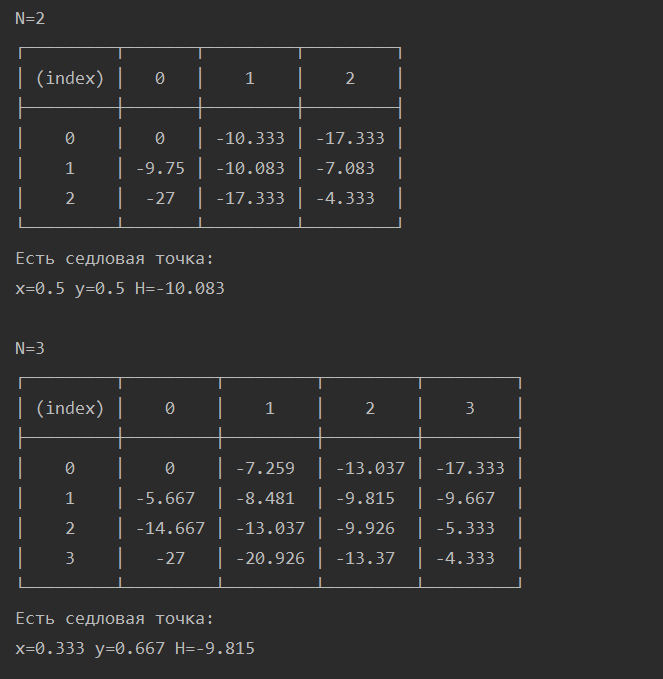


Рисунок 1 – первые 2 итерации численного метода

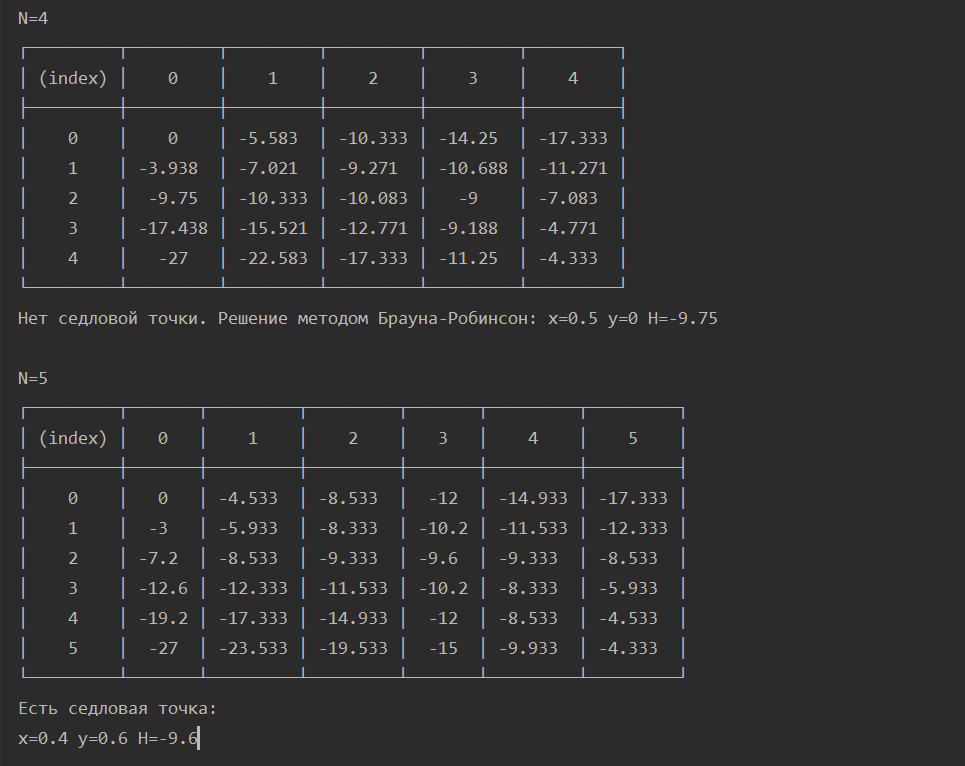


Рисунок 2 – вторые 2 итерации численного метода



Рисунок 3 – 5я итерация численного метода

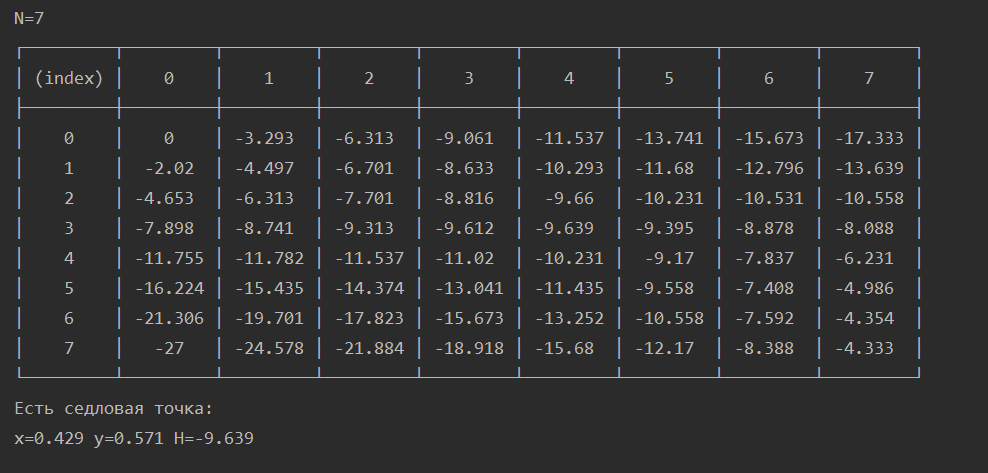


Рисунок 4 – 6я итерация численного метода

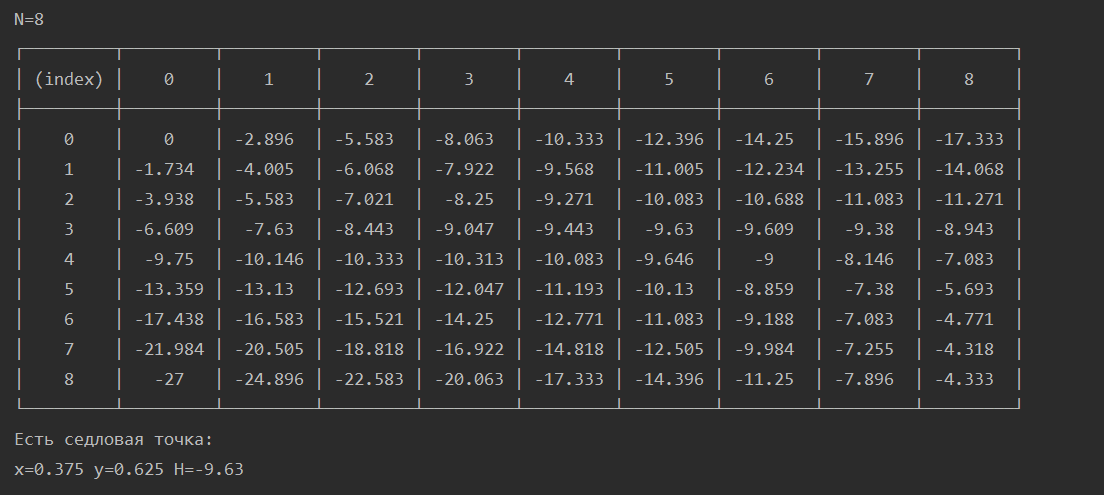


Рисунок 5 – 7я итерация численного метода

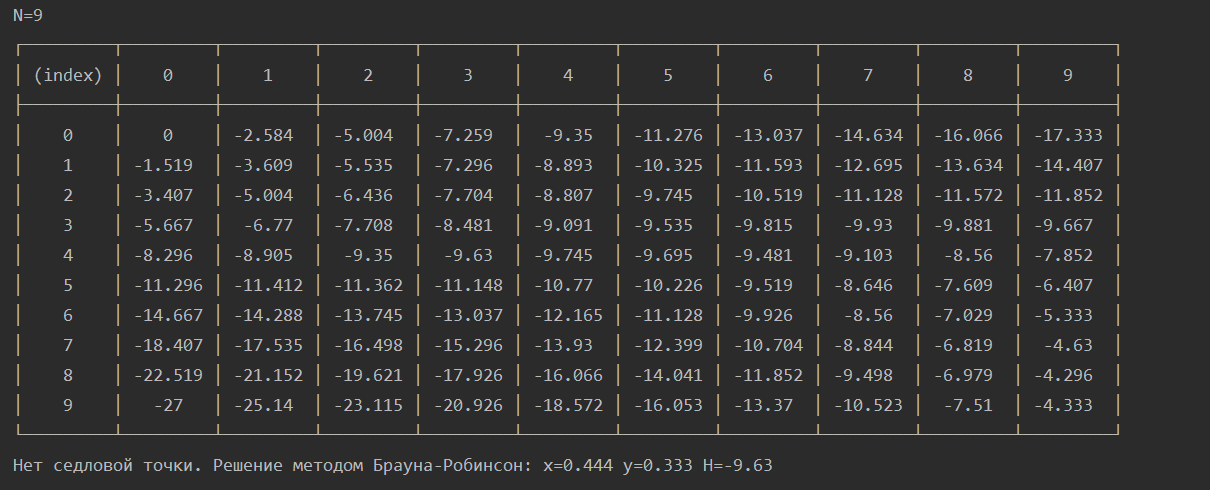


Рисунок 6 – 8я итерация численного метода

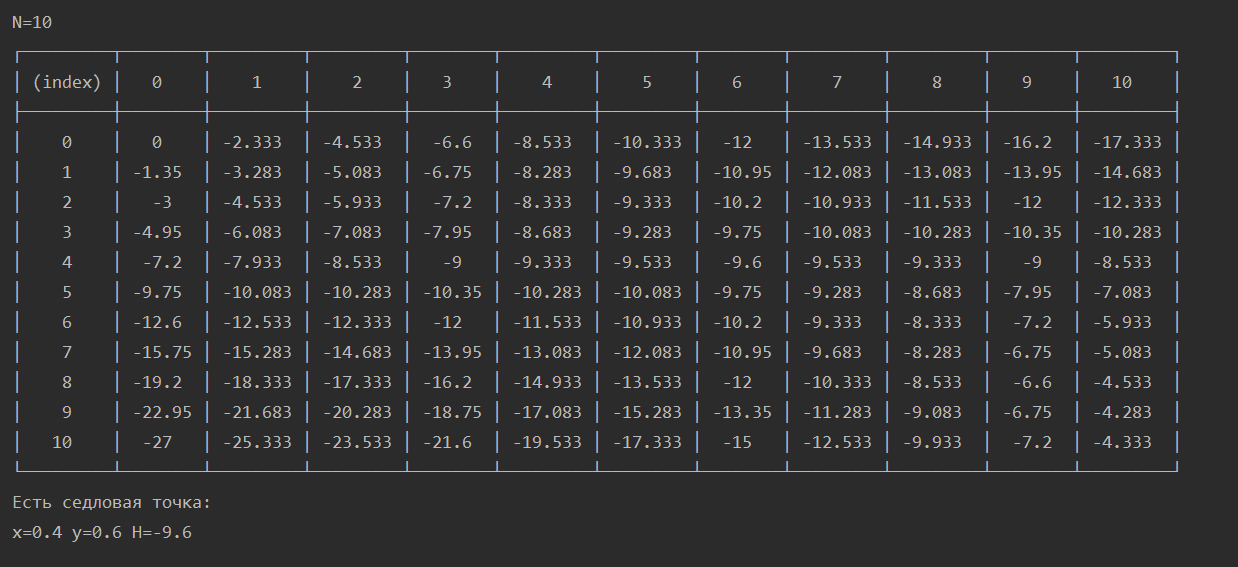


Рисунок 7 – 9я итерация численного метода

Таким образом, в результате численного метода мы получили следующие оптимальные чистые стратегии и цену игры:

Решение аналитическим методом и численным методом совпали. В программе было задано округление до третьего значения после запятой. Погрешность составляет .

Вывод:

В результате выполнения лабораторной работы были найдены оптимальные чистые стратегии и цена игры для непрерывной выпукло-вогнутой антагонистической игры аналитическим и численным методами. Результаты двух методов совпали, при условии, что в программе все значения округлялись до третьего знака после запятой.

ПРИЛОЖЕНИЕ А

let braunRobinson = require('../lab1/lab1');  
  
function findSaddlePoint(matrix) {  
  
 let matrixMin = [[]];  
 let matrixMax = [[]];  
 for (let i = 0; i < matrix.length; i++) {  
 matrixMin[i] = [];  
 matrixMax[i] = [];  
 }  
  
 for (let i = 0; i < matrix.length; i++) {  
  
 for (let j = 0; j < matrix.length; j++) {  
 matrixMin[i][j] = undefined;  
 matrixMax[i][j] = undefined;  
 }  
 }  
  
 let idxMin = 0;  
 let idxMax = 0;  
  
 for (let i = 0; i < matrix.length; i++) {  
 let minEl = matrix[i][0];  
 let maxEl = matrix[0][i];  
  
 for (let j = 0; j < matrix.length; j++) {  
  
  
 if (matrix[i][j] < minEl) {  
 minEl = matrix[i][j];  
 idxMin = j;  
  
 }  
 if (matrix[j][i] > maxEl) {  
 maxEl = matrix[j][i];  
 idxMax = j;  
 }  
  
 }  
 matrixMin[i][idxMin] = minEl;  
 matrixMax[idxMax][i] = maxEl;  
 }  
 for (let i = 0; i < matrix.length; i++) {  
  
 for (let j = 0; j < matrix.length; j++) {  
 if (matrixMax[i][j] && matrixMin[i][j]) {  
 return {  
 value: matrixMax[i][j],  
 x: ***Number***((i / (matrix.length - 1)).toFixed(3)),  
 y: ***Number***((j / (matrix.length - 1)).toFixed(3))  
  
 }  
 }  
  
  
 }  
 }  
 return 0;  
  
}  
  
function kernel(x, y) {  
 let a = -15;  
 let b = 20 / 3;  
 let c = 40;  
 let d = -12;  
 let e = -24;  
 return ***Number***((a \* x \* x + b \* y \* y + c \* x \* y + d \* x + e \* y).toFixed(3));  
  
}  
  
function buildMatrix(N) {  
 let matrix = [];  
  
 for (let i = 0; i < N + 1; i++) {  
 matrix[i] = [];  
 }  
  
  
 for (let i = 0; i < N + 1; i++) {  
 for (let j = 0; j < N + 1; j++) {  
 matrix[i][j] = kernel(i / N, j / N);  
 }  
 }  
 ***console***.table(matrix);  
 return matrix;  
  
}  
  
function findH(matrix, x, y) {  
 let H = 0;  
 for (let i = 0; i < x.length; i++) {  
 for (let j = 0; j < y.length; j++) {  
 H += matrix[i][j] \* x[i] \* y[j];  
 }  
 }  
 return H;  
  
}  
  
function algorithm() {  
 for (let i = 2; i < 11; i++) {  
 ***console***.log( '\n'+'N=' + i);  
 let matrix = buildMatrix(i);  
 let saddlePoint = findSaddlePoint(matrix);  
 if (saddlePoint) {  
 ***console***.log('Есть седловая точка: ' + '\n' + `x=${saddlePoint.x} ` + `y=${saddlePoint.y} ` + `H=${saddlePoint.value}`);  
 } else {  
 let obj = braunRobinson(matrix);  
 let H = findH(matrix, obj.x, obj.y);  
  
 let matrixDist = [];  
 for (let i = 0; i < matrix.length; i++) {  
 matrixDist[i] = [];  
  
 }  
  
  
 for (let i = 0; i < matrix.length; i++) {  
 for (let j = 0; j < matrix.length; j++) {  
 matrixDist[i][j] = ***Math***.abs(matrix[i][j] - H);  
 }  
 }  
  
 let minEl = ***Math***.min(...[].concat(...matrixDist));  
 let braunRobDesicion = {};  
 for (let i = 0; i < matrixDist.length; i++) {  
 for (let j = 0; j < matrixDist.length; j++) {  
 if (matrixDist[i][j] === minEl) {  
 braunRobDesicion["x"] = ***Number***((i / (matrix.length - 1)).toFixed(3));  
 braunRobDesicion["y"] = ***Number***((j / (matrix.length - 1)).toFixed(3));  
 braunRobDesicion['value'] = matrix[i][j];  
  
 }  
 }  
 }  
  
  
 ***console***.log('Нет седловой точки. Решение методом Брауна-Робинсон: ' + 'x=' + braunRobDesicion['x'] + ' y=' + braunRobDesicion['y'] + ' H=' + braunRobDesicion["value"]);  
 }  
  
 }  
  
}  
  
algorithm();