**Министерство образования Российской Федерации**

# МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э. БАУМАНА

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

# ТЕОРИЯ ИГР И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ

**Лабораторная работа №1 на тему:**

«Аналитический и численный методы решения антагонистической игры в смешанных стратегиях»

Вариант 9

|  |  |
| --- | --- |
| Преподаватель: | Коннова Н.С. |
| Студент: | Киселев В.А. |
| Группа: | ИУ8-104 |

Москва, 2025

# Цель работы

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

# Постановка задачи

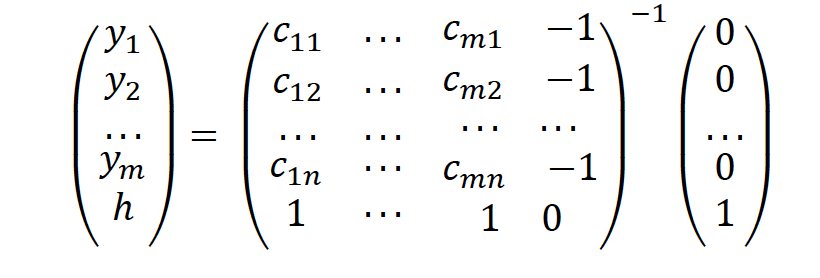
Найдите цену игры и оптимальные стратегии обоих игроков методами обратной матрицы и Брауна Робинсон. Сравните полученные результаты.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ход работы** |  | | |
| Исходная платежная матрица: |
|  | 19 | 7 | 3 |
|  | (6 | 9 | 9 ) |
|  | 8 | 2 | 11 |

Первое, что делаем – ищем доминирующие стратегии и пытаемся упростить матрицу.

Исходная матрица не упрощается.

# Аналитический метод



Для игрока А:

19x₁ + 6x₂ + 8x₃ = v

7x₁ + 9x₂ + 2x₃ = v

3x₁ + 9x₂ + 11x₃ = v

x₁ + x₂ + x₃ = 1

Решение

x₁ = 9/57 ≈ 0.1579

x₂ = 44/57 ≈ 0.7719

x₃ = 4/57 ≈ 0.0702

v = 467/57 ≈ 8.193

Для игрока В:

19y₁ + 7y₂ + 3y₃ = v

6y₁ + 9y₂ + 9y₃ = v

8y₁ + 2y₂ + 11y₃ = v

y₁ + y₂ + y₃ = 1

Решение

y₁ = 46/171 ≈ 0.2690

y₂ = 38/171 ≈ 0.2222

y₃ = 87/171 ≈ 0.5088

v = 467/57 ≈ 8.193

Получили решение:

Смешанные стратегии игрока А: ( 0.1579 ; 0.7719 ; 0.0702)

Смешанные стратегии игрока B: ( 0.2690 ; 0.2222 ; 0.5088 )

Цена игры: 8,193

# Метод Брауна Робинсон

Решение матричной игры методом Брауна Робинсон представляет собой решение задачи динамического программирования, а именно заполнения

следующей таблицы:

Таблица 1 – метод Брауна Робинсон

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k | A | B | x1 | x2 | x3 | y1 | y2 | y3 | Верхняя | Нижняя | ε |
| 1 | x1 | y1 | 19.0000 | 6.0000 | 8.0000 | 19.0000 | 7.0000 | 3.0000 | 19.0000 | 3.0000 | 16.0000 |
| 2 | x1 | y3 | 22.0000 | 15.0000 | 19.0000 | 38.0000 | 14.0000 | 6.0000 | 11.0000 | 3.0000 | 8.0000 |
| 3 | x1 | y3 | 25.0000 | 24.0000 | 30.0000 | 57.0000 | 21.0000 | 9.0000 | 10.0000 | 3.0000 | 7.0000 |
| 4 | x3 | y3 | 28.0000 | 33.0000 | 41.0000 | 65.0000 | 23.0000 | 20.0000 | 10.2500 | 5.0000 | 5.2500 |
| 5 | x3 | y3 | 31.0000 | 42.0000 | 52.0000 | 73.0000 | 25.0000 | 31.0000 | 10.4000 | 5.0000 | 5.4000 |
| 6 | x3 | y2 | 38.0000 | 51.0000 | 54.0000 | 81.0000 | 27.0000 | 42.0000 | 9.0000 | 4.5000 | 4.5000 |
| 7 | x3 | y2 | 45.0000 | 60.0000 | 56.0000 | 89.0000 | 29.0000 | 53.0000 | 8.5700 | 4.1400 | 4.4286 |
| 8 | x2 | y2 | 52.0000 | 69.0000 | 58.0000 | 95.0000 | 38.0000 | 62.0000 | 8.6200 | 4.7500 | 3.8750 |
| 9 | x2 | y2 | 59.0000 | 78.0000 | 60.0000 | 101.0000 | 47.0000 | 71.0000 | 8.6700 | 5.2200 | 3.4444 |
| 10 | x2 | y2 | 66.0000 | 87.0000 | 62.0000 | 107.0000 | 56.0000 | 80.0000 | 8.7000 | 5.6000 | 3.1000 |
| 3161 | x2 | y3 | 24891.0000 | 26166.0000 | 25216.0000 | 25942.0000 | 25957.0000 | 25849.0000 | 8.2800 | 8.1800 | 0.1003 |
| 3162 | x2 | y3 | 24894.0000 | 26175.0000 | 25227.0000 | 25948.0000 | 25966.0000 | 25858.0000 | 8.2800 | 8.1800 | 0.1003 |
| 3163 | x2 | y3 | 24897.0000 | 26184.0000 | 25238.0000 | 25954.0000 | 25975.0000 | 25867.0000 | 8.2800 | 8.1800 | 0.1002 |
| 3164 | x2 | y3 | 24900.0000 | 26193.0000 | 25249.0000 | 25960.0000 | 25984.0000 | 25876.0000 | 8.2800 | 8.1800 | 0.1002 |
| 3165 | x2 | y3 | 24903.0000 | 26202.0000 | 25260.0000 | 25966.0000 | 25993.0000 | 25885.0000 | 8.2800 | 8.1800 | 0.1002 |
| 3166 | x2 | y3 | 24906.0000 | 26211.0000 | 25271.0000 | 25972.0000 | 26002.0000 | 25894.0000 | 8.2800 | 8.1800 | 0.1001 |
| 3167 | x2 | y3 | 24909.0000 | 26220.0000 | 25282.0000 | 25978.0000 | 26011.0000 | 25903.0000 | 8.2800 | 8.1800 | 0.1001 |
| 3168 | x2 | y3 | 24912.0000 | 26229.0000 | 25293.0000 | 25984.0000 | 26020.0000 | 25912.0000 | 8.2800 | 8.1800 | 0.1001 |
| 3169 | x2 | y3 | 24915.0000 | 26238.0000 | 25304.0000 | 25990.0000 | 26029.0000 | 25921.0000 | 8.2800 | 8.1800 | 0.1000 |
| 3170 | x2 | y3 | 24918.0000 | 26247.0000 | 25315.0000 | 25996.0000 | 26038.0000 | 25930.0000 | 8.2800 | 8.1800 | 0.1000 |

Получили решение:

Приближенная цена игры: 8.230

Погрешность: 0.1000

Стратегия игрока A: [0.15899054 0.77413249 0.06687697]

Стратегия игрока B: [0.24006309 0.25488959 0.50504732]

# Выводы

В ходе лабораторной работы были успешно освоены и применены аналитический и численный методы решения антагонистических игр в смешанных стратегиях.

Аналитический метод дает точное решение задачи, но имеет существенные недостатки: применимость только для невырожденных платежных матриц, большая вычислительная сложность *O(n3), n* – число стратегий.

Метод Брауна-Робинсон, в свою очередь, не имеет вышеуказанных недостатков. Он является более универсальным и применимым для игр с большими размерностями, где аналитические методы становятся неэффективными. Недостатками данного метода можно отметить то, что он дает приближенное решение, является немонотонным.



import numpy as np import csv

from prettytable import PrettyTable

def make\_pretty(matrix):

table= PrettyTable() table.dividers

table.field\_names = ["Strategy", "bl", "b2", "b3", "min prize A"] table.add\_rows([["al", matrix[0][0], matrix[0][1], matrix[0][2], miz\_prize[0]],

["a2", matrix[1][0], matrix[l][l], matrix[1][2], miz\_prize[l]],

["a3", matrix[2][0], matrix[2][1], matrix[2][2], miz\_prize[2]]]) table.add\_divider()

table.add\_row(["max loss B", max\_loss[0]. max\_loss[ll, max\_loss[2]. ""]) return table

matrix np.array([[6, 15, 16],

[15, 10, 0],

[10, 7, 10]],

dtype=int)

miz\_prize [np.min(matrix, axis=1)[0], np.min(matrix, axis=l)[l], np.min(matrix, axis=1)[2]] max\_loss = [np.max(matrix, axis=0)[0], np.max(matrix, axis=0)[1], np.max(matrix, axis=0)[2]] print(make\_pretty(matrix))

+------------+----+----+----+ +

Strategy I bl I b2 I b3 I min prize A I

+------------+----+----+----+ +

al I 6 I 15 I 16 6

a2 I 15 I 10 I 0 0

a3 I 10 I *1* I 10 7

+------------+----+----+----+ +

I max loss B I 15 I 15 I 16 I

+------------+----+----+----+ +

def remove\_duplicate\_strats(matrix):

\_, ind\_rows = np.unique(matrix, axis=0, return\_index=True) matrix= matrix[np.sort(ind\_rows)]

\_, ind\_cols = np.unique(matrix, axis=l, return\_index=True) matrix= matrix[:, np.sort(ind\_cols)J

return matrix

def remove\_dominated\_strats(matrix):

matrix= remove\_duplicate\_strats(matrix)

cols\_to\_keep = []

for i in range(matrix.shape[l]): keep= True

for j in range(matrix.shape[l]):

if i != j and np.all(matrix[:, i] <= matrix[:, j]): keep= False

break

if keep: cols\_to\_keep.append(i)

matrix= matrix[:, cols\_to\_keep] rows\_to\_keep = []

for i in range(matrix.shape[0]): keep= True

for j in range(matrix.shape[0]):

if i != j and np.all(matrix[i] >= matrix[j]): keep= False

break

if keep: rows\_to\_keep.append(i)

matrix= matrix[rows\_to\_keep] return matrix

matrix= remove\_dominated\_strats(matrix) print(make\_pretty(matrix))

+------------+----+----+----+ +

Strategy I bl I b2 I b3 I min prize A I

+------------+----+----+----+ +

al I 6 I 15 I 16 I 6

I a2

I a3

I 15 I 10 I 0 0

I 10 I *1* I 10 7

+------------+----+----+----+ +

I max loss B I 15 I 15 I 16 I

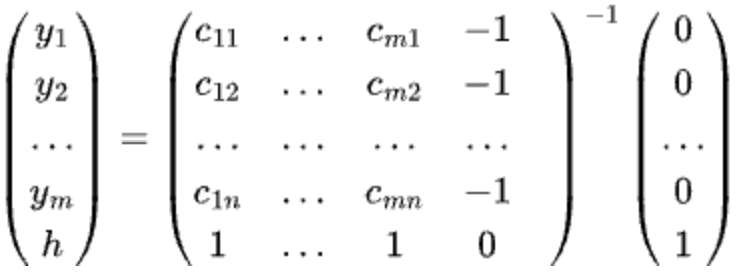
+------------+----+----+----+ +

print(f"max(min(prize A))= {max(miz\_prize)}") print(f"min(max(loss B)) {min(max\_loss)}")

**T** max(min(prize A)) = 7 min(max(loss B)) = 15

CeAnOB85l T04K8 OTCYTCTByeT

### v AHa111.1T1.11.1ecK1.1M MeTOA



def solve\_analytic(matrix):

matrix= matrix.T

m np.vstack((matrix, np.ones(matrix.shape[l])))

m = np.hstack((m, np.array([-1] \* (m.shape[0] - 1) + [0]).reshape(-1, 1))) col= np.array([0] \* (m.shape[0] - 1) + [1])

return np.linalg.solve(m, col)

def solve\_analytic\_B(matrix):

return solve\_analytic(matrix.T)

def solve\_analytic\_A(matrix):

return solve\_analytic(matrix)

print(f'CMewaHHb1e npaTerne erpoKa A: {np.round(solve\_analytic\_A(matrix)[:3], **2)}')** print(f'CMewaHHb1e npaTerna arpoKa B: {np.round(solve\_analytic\_B(matrix)[:3], 2)}') print(f'LieHa erpbi: {np.round(solve\_analytic\_A(matrix)[-1], **2)}')**

*-:!:* CMewaHHble cTpaTerne erpoKa A:[0.28 0.19 0.53] CMewaHHwe cTpaTeree erpoKa B:[0.61 0.06 0.33]

LleHa erpbi: 9.81

### v MeTOA 6payHa-Po61.1HcoH

def brown\_robinson(payoff\_matrix, epsilon=0.1, max\_iterations=1000, filename="brown\_robinson.csv"): rows, cols= payoff\_matrix.shape

a\_sums np.zeros(rows) b\_sums = np.zeros(cols)

current\_pl 0

current\_p2 0

pl\_strategies np.zeros(cols) p2\_strategies np.zeros(cols)

min\_upper float('inf') max\_lower -float('inf')

with open(filename, 'w', newline='', encoding='utf-8') as f: writer= csv.writer(f, quoting=csv.QUOTE\_NONNUMERIC) writer.writerow([

**'k', 'A', 'B',**

**'al', 'a2',** 'a3',

'bl', 'b2', 'b3',

'BL(\,1', 'HL(\,1', 'E'

])

for iteration in range(l, max\_iterations+l): a\_sums += payoff\_matrix[:, current\_p2] b\_sums += payoff\_matrix[current\_pl, :]

pl\_strategies[current\_pl] += 1

p2\_strategies[current\_p2] += 1

upper np.max(a\_sums) / iteration lower np.min(b\_sums) / iteration

min\_upper = min(min\_upper, upper) max\_lower = max(max\_lower, lower) eps = min\_upper - max\_lower

writer.writerow([ **iteration,** f"a{current\_pl + 1}",

f"b{current\_p2 + 1}",

**\*a\_sums,**

**\*b\_sums,**

round(upper, 2),

round(lower, 2),

round(eps, 2)

])

if eps <= epsilon: break

current\_pl np.argmax(a\_sums) current\_p2 np.argmin(b\_sums)

# v1TOCOBble CTpaTerne

final\_pl\_strategy = pl\_strategies / iteration final\_p2\_strategy = p2\_strategies / iteration price= (min\_upper + max\_lower) / 2

return final\_pl\_strategy, final\_p2\_strategy, price

pl, p2, val= brown\_robinson(matrix, epsilon=0.1) print("v1Toroswe cTpaTeree:")

print(f"A: {np.round(pl, 2)}")

print(f"B:{np.round(p2, 2)}")

print(f"l.leHa erpb1: {val:.2f}")

v1TOCOBble CTpaTern,:

A: [0.27 0.15 0.58]

B: [0.63 0.03 0.34]

L(eHa arpb1: 9.86

print(f"nposepKa 6payHa-Po6eHCOH AllR erpoKa A: sum(pl)) print(f"nposepKa 6payHa-Po6eHCOH AllR erpoKa B: sum(p2))

nposepKa 6payHa-Po6eHCOH AllR erpoKa A: 1.0 nposepKa 6payHa-Po6>HCOH AllR erpoKa B: 1.0

print("AnR aHall>T>YecKoro MeTOAa:")

print(f'A: {np.round(solve\_analytic\_A(matrix)[:3], 2)}')

print(f'B: {np.round(solve\_analytic\_B(matrix)[:3], 2)}') print(f'L(eHa erpb1: {np.round(solve\_analytic\_A(matrix)[-1], 2)}')

AllR aHaJl>T>YeCKOCO MeTOAa:

A: [0.28 0.19 0.53]

B: [0.61 0.06 0.33]

L(eHa erpb1: 9.81

print(f"nposepKa aHall>T>YecKoro MeTOAa AllR erpoKa A: " print(f"nposepKa aHall>T>YeCKOrQ MeTOAa AllR erpoKa B: "

T nposepKa aHan>T>YecKoro MeTOAa AJlR erpoKa A: 1.0 nposepKa aHaJl>T>YeCKOCO MeTOAa AJlR erpoKa B: 1.0

sum(solve\_analytic\_A(matrix)[:3])) sum(solve\_analytic\_B(matrix)[:3]))