**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

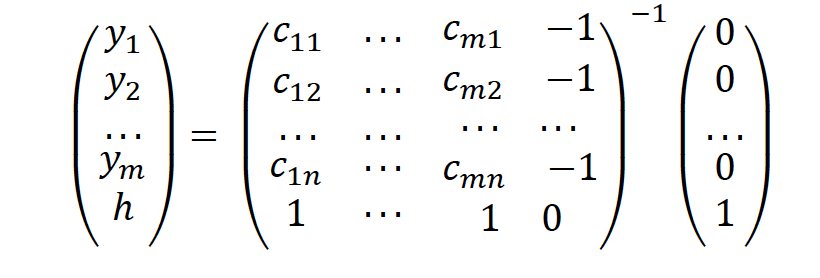
Найдите цену игры и оптимальные стратегии обоих игроков методами обратной матрицы и Брауна Робинсон. Сравните полученные результаты.

|  |  |  |  |
| --- | --- | --- | --- |
| **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 немонотонного алгоритма.

import numpy as np

import matplotlib.pyplot as plt

from tabulate import tabulate

from collections import deque

# Платежная матрица для варианта 9

C = np.array([

    [19, 7, 3],

    [6, 9, 9],

    [8, 2, 11]

])

# ========== АНАЛИТИЧЕСКИЙ МЕТОД (корректный) ==========

print("АНАЛИТИЧЕСКИЙ МЕТОД (ОБРАТНОЙ МАТРИЦЫ)")

print("=" \* 50)

C\_inv = np.linalg.inv(C)

u = np.ones(C.shape[0])

uC\_inv = u @ C\_inv

uC\_inv\_uT = u @ C\_inv @ u.T

v\_analytic = 1 / uC\_inv\_uT

x\_analytic = v\_analytic \* uC\_inv

y\_analytic = v\_analytic \* (C\_inv @ u.T)

print(f"Цена игры v = {v\_analytic:.6f}")

print(f"Оптимальная стратегия игрока A: {x\_analytic}")

print(f"Оптимальная стратегия игрока B: {y\_analytic}")

# ========== ИСПРАВЛЕННЫЙ МЕТОД БРАУНА-РОБИНСОНА ==========

print("\n" + "=" \* 50)

print("ИСПРАВЛЕННЫЙ МЕТОД БРАУНА-РОБИНСОНА")

print("=" \* 50)

# Инициализация (как в документе - частоты использования стратегий)

n = 3

x\_freq = np.zeros(n)  # Частоты стратегий игрока A (аналогично ˜x\_i[k] в документе)

y\_freq = np.zeros(n)  # Частоты стратегий игрока B (аналогично ˜y\_j[k] в документе)

# Накопленные выигрыши (для вычисления оценок)

A\_accumulated = np.zeros(n)  # Накопленные выигрыши для стратегий A

B\_accumulated = np.zeros(n)  # Накопленные проигрыши для стратегий B

# История для таблицы

first\_10 = []

last\_10 = deque(maxlen=10)

max\_iterations = 10000

for k in range(1, max\_iterations + 1):

    # Выбор стратегий (как в документе)

    if k == 1:

        i\_k, j\_k = 0, 0  # Произвольный начальный выбор

    else:

        # Игрок A выбирает стратегию, максимизирующую выигрыш против эмпирической стратегии B

        i\_k = np.argmax(A\_accumulated)

        # Игрок B выбирает стратегию, минимизирующую проигрыш против эмпирической стратегии A

        j\_k = np.argmin(B\_accumulated)

    # Обновление частот (как в документе: ˜x\_i[k] и ˜y\_j[k])

    x\_freq[i\_k] += 1

    y\_freq[j\_k] += 1

    # Обновление накопленных выигрышей (для вычисления оценок)

    for i in range(n):

        A\_accumulated[i] += C[i, j\_k]  # Выигрыш стратегии i против выбора j\_k

    for j in range(n):

        B\_accumulated[j] += C[i\_k, j]  # Проигрыш стратегии j против выбора i\_k

    # Вычисление оценок цены игры (как в документе)

    upper = np.max(A\_accumulated) / k  # ¯v[k]/k

    lower = np.min(B\_accumulated) / k  # v[k]/k

    epsilon = upper - lower

    # Сохранение истории (частоты использования стратегий, как в документе)

    record = {

        'k': k,

        'choice\_A': f'x{i\_k + 1}',

        'choice\_B': f'y{j\_k + 1}',

        'x\_freq': x\_freq.copy(),  # Частоты использования стратегий A

        'y\_freq': y\_freq.copy(),  # Частоты использования стратегий B

        'upper': upper,

        'lower': lower,

        'epsilon': epsilon

    }

    if k <= 10:

        first\_10.append(record)

    last\_10.append(record)

    if epsilon <= 0.1:

        break

# Формирование таблицы (частоты использования стратегий, как в документе)

table\_data = []

for record in first\_10 + list(last\_10):

    row = [

        record['k'],

        record['choice\_A'],

        record['choice\_B'],

        \*record['x\_freq'],  # Частоты использования стратегий A

        \*record['y\_freq'],  # Частоты использования стратегий B

        f"{record['upper']:.4f}",

        f"{record['lower']:.4f}",

        f"{record['epsilon']:.6f}"

    ]

    table\_data.append(row)

headers = ['k', 'A', 'B', 'x1', 'x2', 'x3', 'y1', 'y2', 'y3', 'Верхняя', 'Нижняя', 'ε']

print("Результаты метода Брауна-Робинсона (первые 10 и последние 10 итераций):")

print(tabulate(table\_data, headers=headers, tablefmt='grid'))

# Финальные результаты (средние стратегии)

x\_avg = x\_freq / k

y\_avg = y\_freq / k

v\_approx = (upper + lower) / 2

print(f"\nФинальные результаты после {k} итераций:")

print(f"Приближенная цена игры: {v\_approx:.6f}")

print(f"Погрешность: {epsilon:.6f}")

print(f"Стратегия игрока A: [{', '.join([f'{x:.6f}' for x in x\_avg])}]")

print(f"Стратегия игрока B: [{', '.join([f'{y:.6f}' for y in y\_avg])}]")