**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по лабораторной работе №3**

**по дисциплине «Теория принятия решений»**

**Тема: Принятие решений в матричных играх**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1384 |  | Усачева Д.В. |
| Преподаватель |  | Попова Е.В. |

Санкт-Петербург

2025

**Цель работы.**

Найти решение задач в матричных играх.

**Задание.**

Графически и аналитически решить матричную игру 2 × N для матрицы C2.

Графически и аналитически решить матричную игру M × 2 для матрицы C3.

С помощью инструментальных средств решить матричную игру M × N для матрицы C4.

Подсчитать относительную погрешность результатов.

Биматричная игра задана матрицами А и В. Найти решение игры графическим методом. Подсчитать выигрыши.

Решить эту игру как матричную антагонистическую с нулевой суммой: решить графически и аналитически игру 2×2 для матрицы А, а затем для матрицы В. Сравнить результаты с биматричными.

Вариант 17.

**Основные теоретические положения.**

Теория игр– это математическая теория конфликтных ситуаций.

Игра – модель конфликтной ситуации.

Градация игр по неопределенности исходов:

* комбинаторные;
* азартные;
* стратегические.

Игра с нулевой суммой, если выигрыш одного игрока равен проигрышу

другого.

Матричная игра– это игра с нулевой суммой, в которой задается выигрыш

игрока 1 в виде матрицы.

Игра называется конечной, если у каждого игрока существует конечное

число возможных стратегий.

Решить задачупо теории игр – это найти оптимальные стратегии игроков

(чистые или смешанные) и выигрыш (результат игры).

Антагонистические игры

***Определение***: Пара называется седловой точкой функции , если

или эквивалентно

Антагонистическая игра задается набором

***Определение***: Антагонистическая игра имеет решение, если имеет седловую точку.

Если – седловая точка,

то

***Лемма***: Если - 2 седловые точки, то

***Доказательство***: Если - седловая точка, то

***Определение***: Антагонистическая игра называется матричной, если множества стратегий игроков конечны

– стратегия первого игрока, – стратегия второго игрока.

***Определение***: Наилучший гарантированный результат для 1 игрока

Стратегия называется максиминной, если

Наилучший гарантированный результат для 2 игрока

Стратегия называется минимаксной, если

***Лемма***: В любой антагонистической игре справедливо

***Определение***: Если в строке матрицы A элементы строки не меньше соответствующих элементов строки , а по крайней мере один строго больше, то строка называется доминирующей, а строка доминируемой.

При решении игры можно уменьшить размеры матрицы путем удаления из неё доминирующих столбцов и доминируемых строк.

***Т1***: в матричных играх существует равновесие по Нэшу в классе смешанных стратегий.

***Т2:*** для того, чтобы ситуация была равновесием по Нэшу, необходимо и достаточно, чтобы для любых чистых стратегий и , выполнялись условия:

***Теорема фон Неймана***: всякая матричная игра имеет решение в смешанных стратегиях.

**Выполнение работы.**

Игра 2xN

Можно уменьшить размерность матрицы путём удаления доминирующих столбцов. Здесь столбец доминирует над , поэтому его можно убрать.

Теперь можем провести графическое решение:

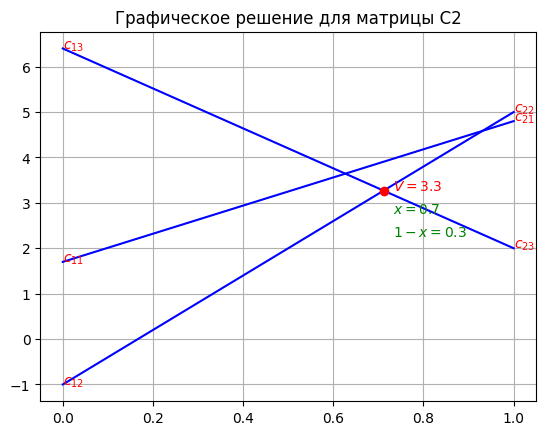


Рисунок 1 – Графическое решение матричной игры

По формулам вычислим следующие значения:

Игра Mx2

Можно уменьшить размерность матрицы путём удаления доминируемых строк. и доминируемы строкой , а строка доминируема строкой .

Теперь можем провести графическое решение:

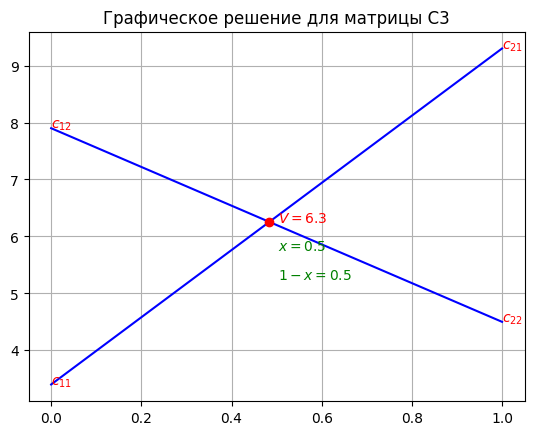


Рисунок 2 – Графическое решение матричной игры

Решение для второго игрока:

Игра MxN

Размерность матрицы уменьшить не получится. Решим задачу симплекс методом.

Для первого игрока:

Для второго игрока:

Биматричная игра

Составим функции МО выигрышей:

Рассмотрим условие равновесия:

Следовательно:

Пусть

Пусть

При

Пусть

Пусть

При

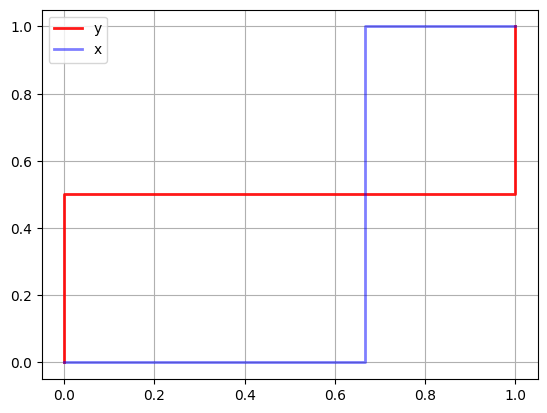


Рисунок 3 – Графическое решение биматричной игры

Области пересекаются в точке:

(x, y) = (0, 0)

(x, y) = (2/3, 1/2)

(x, y) = (1, 1)

Решим, как матричную антагонистическую с нулевой суммой

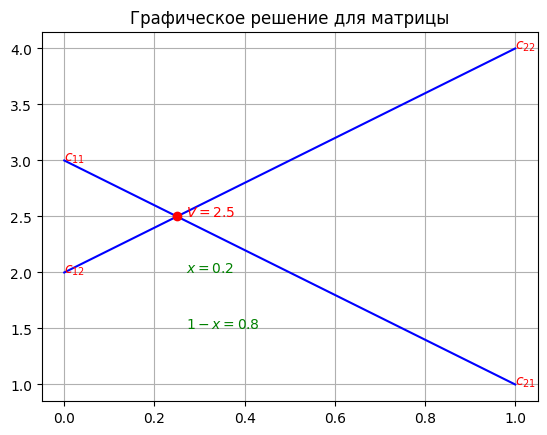
**

Рисунок 4 – Графическое решение матричной игры

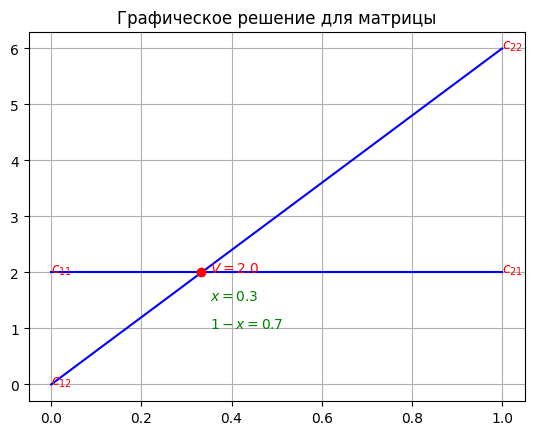
**

Рисунок 5 – Графическое решение матричной игры

Заметим, что результаты совпали с результатами биматричной игры при (x, y) = (2/3, 1/2).

**Вывод.**

В ходе выполнения данной работы были рассмотрены различные типы матричных игр, включая игры с нулевой суммой и биматричные игры, с целью нахождения оптимальных стратегий для игроков и значений игры. Для матриц C2 и C3 было установлено отсутствие седловых точек, и значения игры были определены в заданных пределах, с последующим уменьшением размерности матриц для упрощения задачи и нахождения оптимальных стратегий. Игра MxN для матрицы C4 была решена симплекс-методом, что позволило определить оптимальные стратегии и значение игры. В биматричной игре (матрицы A и B) найдено равновесие Нэша в смешанных стратегиях с выигрышами H1 = 2.5 и H2 = 2, что совпало с результатами решения игры как антагонистической с нулевой суммой для матриц A и B. Таким образом, работа подтвердила эффективность графических и аналитических методов для решения задач теории игр.

**ПРИЛОЖЕНИЕ А**

ИСХОДНЫЙ КОД ПРОГРАММЫ

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import linprog

def plot\_solution(matrix, title):

lines = []

for i in range(matrix.shape[1]):

c1i = matrix[0][i]

c2i = matrix[1][i]

plt.plot([0, 1], [c1i, c2i], color='blue')

plt.text(1, c2i, f'$c\_{{2{i+1}}}$', color='red')

plt.text(0, c1i, f'$c\_{{1{i+1}}}$', color='red')

lines.append({'a': c2i - c1i, 'b': c1i})

low\_intersection = []

for i in range(len(lines) - 1):

for j in range(i + 1, len(lines)):

a1, b1 = lines[i].values()

a2, b2 = lines[j].values()

if a1 == a2:

continue

x = (b2 - b1) / (a1 - a2)

y = a1 \* x + b1

if 0 < x < 1 and (not low\_intersection or y < low\_intersection[1]):

low\_intersection = (x, y)

if low\_intersection:

x\_inters, y\_inters = low\_intersection

plt.scatter(x\_inters, y\_inters, color='red', zorder=5)

plt.text(x\_inters + 0.02, y\_inters, f'$V={round(y\_inters, 4)}$', color='red')

plt.title(title)

plt.xlabel("Стратегии игрока")

plt.ylabel("Выигрыш")

plt.grid()

plt.show()

def solve\_task\_1():

print("=== Решение задачи 1: Матричная игра 2 × N ===")

C2 = np.array([[1.7, -1, 6.4], [4.8, 5, 2]])

print("Матрица C2:")

print(C2)

plot\_solution(C2, "Графическое решение для матрицы C2")

j, k = 1, 2

c1, c2 = C2[0], C2[1]

p1 = (c2[j] - c2[k]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

p2 = (c1[k] - c1[j]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

V = (c1[k] \* c2[j] - c1[j] \* c2[k]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

print(f"Вероятности для первого игрока: p1 = {p1:.4f}, p2 = {p2:.4f}")

print(f"Значение игры: V = {V:.8f}")

q1 = (c2[k] - c1[k]) / (c1[j] + c2[k] - (c1[k] + c2[j]))

print(f"Вероятность для второго игрока: q1 = {q1:.4f}\n")

def solve\_task\_2():

print("=== Решение задачи 2: Матричная игра M × 2 ===")

C3 = np.array([[3.4, 7.9], [9.3, 4.5]])

print("Матрица C3:")

print(C3)

plot\_solution(C3, "Графическое решение для матрицы C3")

j, k = 0, 1

q1 = (C3[j][1] - C3[k][1]) / (C3[k][0] + C3[j][1] - (C3[k][1] + C3[j][0]))

print(f"Вероятность для второго игрока: q1 = {q1:.4f}\n")

def solve\_task\_3():

print("=== Решение задачи 3: Матричная игра M × N ===")

C4 = np.array([[2, 2, -5], [1, -6, 9], [-4, 4, -11]]) + 11

print("Матрица C4:")

print(C4)

def solve\_lp(c, A\_ub, b\_ub):

res = linprog(c, A\_ub=A\_ub, b\_ub=b\_ub, method='highs')

if res.success:

Z = res.fun

V = 1 / Z

p = res.x \* V

return V, p

return None, None

V1, p = solve\_lp([1, 1, 1], -C4.T, [-1, -1, -1])

if V1 is not None:

print(f"Решение для первого игрока: V = {V1-11:.4f}, p = {p}")

V2, q = solve\_lp([-1, -1, -1], C4, [1, 1, 1])

if V2 is not None:

print(f"Решение для второго игрока: V = {V2-11:.4f}, q = {q}")

def solve\_task\_4():

print("=== Решение задачи 4: Биматричная игра ===")

A = np.array([[3, 2], [1, 4]])

B = np.array([[2, 0], [2, 6]])

print("Матрица A:")

print(A)

print("Матрица B:")

print(B)

\_x, \_y = 2/3, 1/2

plt.grid()

plt.plot([0, 0, 1, 1], [0, \_y, \_y, 1], c='red', alpha=0.9, label='y', linewidth=2)

plt.plot([0, \_x, \_x, 1], [0, 0, 1, 1], c='blue', alpha=0.5, label='x', linewidth=2)

plt.legend()

plt.show()

def solve\_task\_5(matrix):

print("=== Решение задачи 5: Матричная игра 2 × 2 ===")

plot\_solution(matrix, "Графическое решение для матрицы")

j, k = 0, 1

c1, c2 = matrix[0], matrix[1]

p1 = (c2[j] - c2[k]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

p2 = (c1[k] - c1[j]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

V = (c1[k] \* c2[j] - c1[j] \* c2[k]) / (c1[k] + c2[j] - (c1[j] + c2[k]))

q1 = (matrix[j][1] - matrix[k][1]) / (matrix[k][0] + matrix[j][1] - (matrix[k][1] + matrix[j][0]))

print(f"Вероятности для первого игрока: p1 = {p1:.4f}, p2 = {p2:.4f}")

print(f"Вероятность для второго игрока: q1 = {q1:.4f}")

print(f"Значение игры: V = {V:.8f}\n")

A = np.array([[3, 2], [1, 4]])

B = np.array([[2, 0], [2, 6]])

solve\_task\_1()

solve\_task\_2()

solve\_task\_3()

solve\_task\_4()

solve\_task\_5(A)

solve\_task\_5(B)