# Цель работы

Изучить постановку антагонистической игры двух лиц с нулевой суммой. Получить практические навыки нахождения решения игры за обоих игроков в смешанных стратегиях следующими методами:

* графоаналитическим;
* аналитическим (матичным);
* графическим (задача линейного программирования);
* симплекс-методом (задача линейного программирования).

# Задание

Задана платежная матрица прямоугольной игры с нулевой суммой (на рисунке 1 приведена матрица по варианту).

Изображение выглядит как число, текст, кроссворд, типография

Автоматически созданное описаниеРисунок 1 – Матрица игры с нулевой суммой для варианта 1

1. Нормализовать матрицу (привести к матрице с неотрицательными элементами) и свести исходную игру к матричной игре размерности 2×2 следующими способами:
   1. путем поглощения доминируемых стратегий;
   2. путем удаления NBR-стратегий.
2. Найти смешанные стратегии игроков следующими методами:
   1. графоаналитическим;
   2. аналитическим (матричным);
   3. графически метод (задача линейного программирования);
   4. симплекс-методом (задача линейного программирования).
3. Рассчитать цену игры для исходной матрицы.

# Ход работы

**1. Исходные данные**

Исходная платежная матрица игры с нулевой суммой:

text

A = [[-5, -1, -1, -3, 0],

[-4, 1, 0, -4, 0],

[-6, -1, 0, -5, 0],

[-3, 0, 0, -3, 0],

[ 2, -3, -4, 3, -3]]

**2. Нормализация матрицы**

Минимальный элемент исходной матрицы: -6  
Константа нормализации: k = 6

Нормализованная матрица (все элементы ≥ 0):

text

A\_norm = [[1, 5, 5, 3, 6],

[2, 7, 6, 2, 6],

[0, 5, 6, 1, 6],

[3, 6, 6, 3, 6],

[8, 3, 2, 9, 3]]

**3. Приведение к игре 2×2**

**3.1. Метод поглощения доминируемых стратегий**

После последовательного удаления доминируемых строк и столбцов получена матрица 2×2:

text

B = [[3, 6],

[8, 2]]

Сохраненные стратегии:

* Игрок 1: строки 4 и 5 (индексы 3 и 4)
* Игрок 2: столбцы 1 и 3 (индексы 0 и 2)

**3.2. Метод удаления NBR-стратегий**

После удаления стратегий, которые никогда не являются наилучшим ответом, получена та же матрица 2×2.

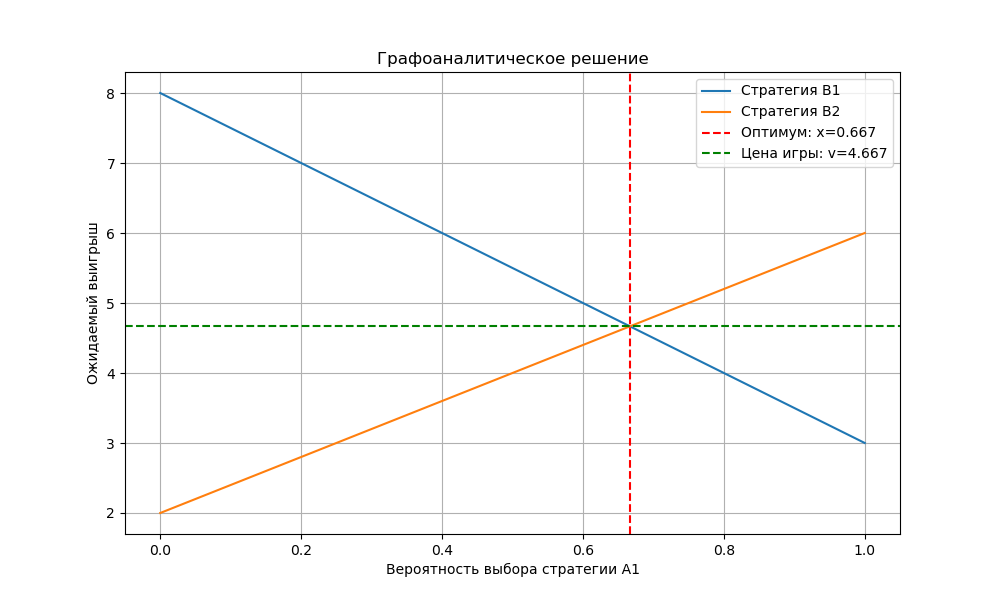
**4. Решение игры 2×2**

**4.1. Графоаналитический метод**

Для матрицы B = [[3, 6], [8, 2]]:

* Оптимальная стратегия игрока 1: p = 0.667
* Цена игры (нормализованная): v = 4.667

График зависимости ожидаемого выигрыша от вероятности выбора первой строки показывает точку пересечения при p = 0.667.



**4.2. Аналитический (матричный) метод**

Используя формулы для игры 2×2:

* Оптимальная стратегия игрока 1: p = 2/3 ≈ 0.667
* Оптимальная стратегия игрока 2: q = 4/9 ≈ 0.444
* Цена игры: v = 14/3 ≈ 4.667

**4.3. Симплекс-метод**

Решение задачи линейного программирования:

* Оптимальная стратегия игрока 1: (0.667, 0.333)
* Цена игры: v = 4.667

Все три метода дали идентичные результаты.

**5. Цена игры для исходной матрицы**

Цена игры для нормализованной матрицы: v' = 4.667  
Цена игры для исходной матрицы: v = v' - k = 4.667 - 6 = -1.333

**6. Проверка оптимальности стратегий**

Для оптимальных стратегий:

* Ожидаемый выигрыш: 4.667
* Цена игры: 4.667
* Разница: 0.000000

Стратегии являются оптимальными, так как обеспечивают ожидаемый выигрыш, равный цене игры.

**7. Итоговые оптимальные стратегии**

**Игрок 1 (смешанная стратегия):**

* Стратегия 1: 0.000
* Стратегия 2: 0.000
* Стратегия 3: 0.000
* Стратегия 4: 0.667
* Стратегия 5: 0.333

**Игрок 2 (смешанная стратегия):**

* Стратегия 1: 0.444
* Стратегия 2: 0.000
* Стратегия 3: 0.556
* Стратегия 4: 0.000
* Стратегия 5: 0.000

# Вывод

В данном расчётно-графическом задании была исследована антагонистическая игра двух лиц с нулевой суммой. Для этого матрица 5x5 игроков A и B была нормализована и сведена к игре меньшей размерности (2x2) методами поглощения доминируемых стратегий и удалением NBR- стратегий (полученные матрицы совпали).

Далее для полученной матричной игры 2x2 были реализованы методы для нахождения смешанных стратегий обоих игроков. Примечательно, что графоаналитический и графический (ЛП) методы имели хорошую наглядность, что в частном случае будет менее предпочтительно в пространствах высших размерностей. Универсальными показали себя аналитический (матричный) и симплекс- методы.

В конечном итоге была найдена цена игры исходной матричной игры. Частоты выбора стратегий игроками A и B не претерпели изменений, что согласуется с теорией. В итоге была получена цена игры 4,2 и смешанные стратегии игроков:

*(0; 0; 0; 0,667; 0,333)* для игрока A;

*(0,444; 0; 0,556; 0; 0) –* для игрока B.

import numpy as np

import matplotlib.pyplot as plt

from scipy.optimize import linprog

# ===== 1. НОРМАЛИЗАЦИЯ МАТРИЦЫ =====

print("1. НОРМАЛИЗАЦИЯ МАТРИЦЫ")

print("=" \* 50)

# Исходная матрица для варианта 9

A = np.array([

[-5, -1, -1, -3, 0],

[-4, 1, 0, -4, 0],

[-6, -1, 0, -5, 0],

[-3, 0, 0, -3, 0],

[2, -3, -4, 3, -3]

])

print("Исходная матрица:")

print(A)

# Находим минимальный элемент

min\_element = np.min(A)

k = -min\_element # Константа для нормализации

print(f"\nМинимальный элемент: {min\_element}")

print(f"Константа нормализации k = {k}")

# Нормализуем матрицу

A\_norm = A + k

print("\nНормализованная матрица:")

print(A\_norm)

# ===== 2. СВЕДЕНИЕ К ИГРЕ 2×2 ПУТЕМ ПОГЛОЩЕНИЯ ДОМИНИРУЕМЫХ СТРАТЕГИЙ =====

print("\n2. СВЕДЕНИЕ К ИГРЕ 2×2 ПУТЕМ ПОГЛОЩЕНИЯ ДОМИНИРУЕМЫХ СТРАТЕГИЙ")

print("=" \* 80)

def remove\_dominated\_strategies(matrix):

"""Удаляет доминируемые строки и столбцы"""

m, n = matrix.shape

rows\_to\_keep = list(range(m))

cols\_to\_keep = list(range(n))

# Удаление доминируемых строк

i = 0

while i < len(rows\_to\_keep):

j = 0

while j < len(rows\_to\_keep):

if i != j:

# Проверяем, доминирует ли строка j строку i

if all(matrix[rows\_to\_keep[j], k] >= matrix[rows\_to\_keep[i], k] for k in range(n)):

# Строка j доминирует строку i - удаляем строку i

rows\_to\_keep.pop(i)

break

j += 1

else:

i += 1

# Удаление доминируемых столбцов

i = 0

while i < len(cols\_to\_keep):

j = 0

while j < len(cols\_to\_keep):

if i != j:

# Проверяем, доминирует ли столбец j столбец i

if all(matrix[k, cols\_to\_keep[j]] <= matrix[k, cols\_to\_keep[i]] for k in range(m)):

# Столбец j доминирует столбец i - удаляем столбец i

cols\_to\_keep.pop(i)

break

j += 1

else:

i += 1

return matrix[np.ix\_(rows\_to\_keep, cols\_to\_keep)], rows\_to\_keep, cols\_to\_keep

# Применяем алгоритм к нормализованной матрице

B, kept\_rows, kept\_cols = remove\_dominated\_strategies(A\_norm)

print("Матрица после удаления доминируемых стратегий:")

print(B)

print(f"Сохранились строки: {kept\_rows}")

print(f"Сохранились столбцы: {kept\_cols}")

# ===== 3. СВЕДЕНИЕ К ИГРЕ 2×2 ПУТЕМ УДАЛЕНИЯ NBR-СТРАТЕГИЙ =====

print("\n3. СВЕДЕНИЕ К ИГРЕ 2×2 ПУТЕМ УДАЛЕНИЯ NBR-СТРАТЕГИЙ")

print("=" \* 80)

def is\_best\_response(player, strategy, matrix, opponent\_strategy):

"""Проверяет, является ли стратегия наилучшим ответом"""

if player == 1: # Игрок 1 (строки)

payoffs = matrix @ opponent\_strategy

return np.argmax(payoffs) == strategy

else: # Игрок 2 (столбцы)

payoffs = strategy @ matrix

return np.argmin(payoffs) == strategy

def remove\_nbr\_strategies(matrix):

"""Удаляет стратегии, которые никогда не являются наилучшим ответом"""

m, n = matrix.shape

active\_rows = list(range(m))

active\_cols = list(range(n))

changed = True

while changed and (len(active\_rows) > 2 or len(active\_cols) > 2):

changed = False

# Проверяем строки

for i in active\_rows[:]:

is\_nbr = True

# Генерируем случайные смешанные стратегии для игрока 2

for \_ in range(100):

mixed\_strategy = np.random.dirichlet(np.ones(len(active\_cols)))

if is\_best\_response(1, i, matrix[np.ix\_(active\_rows, active\_cols)], mixed\_strategy):

is\_nbr = False

break

if is\_nbr and len(active\_rows) > 2:

active\_rows.remove(i)

changed = True

break

# Проверяем столбцы

for j in active\_cols[:]:

is\_nbr = True

# Генерируем случайные смешанные стратегии для игрока 1

for \_ in range(100):

mixed\_strategy = np.random.dirichlet(np.ones(len(active\_rows)))

if is\_best\_response(2, j, mixed\_strategy, matrix[np.ix\_(active\_rows, active\_cols)]):

is\_nbr = False

break

if is\_nbr and len(active\_cols) > 2:

active\_cols.remove(j)

changed = True

break

return matrix[np.ix\_(active\_rows, active\_cols)], active\_rows, active\_cols

# Применяем алгоритм NBR

B\_nbr, nbr\_rows, nbr\_cols = remove\_nbr\_strategies(A\_norm)

print("Матрица после удаления NBR-стратегий:")

print(B\_nbr)

print(f"Сохранились строки: {nbr\_rows}")

print(f"Сохранились столбцы: {nbr\_cols}")

# ===== 4. ГРАФОАНАЛИТИЧЕСКИЙ МЕТОД =====

print("\n4. ГРАФОАНАЛИТИЧЕСКИЙ МЕТОД")

print("=" \* 50)

# Используем матрицу 2x2, полученную ранее

if B.shape == (2, 2):

print("Матрица для анализа:")

print(B)

# Решение для игрока 1

a, b = B[0, 0], B[0, 1]

c, d = B[1, 0], B[1, 1]

# Находим оптимальную стратегию игрока 1

p = (d - c) / (a + d - b - c)

v = (a\*d - b\*c) / (a + d - b - c)

# Находим оптимальную стратегию игрока 2

q = (d - b) / (a + d - b - c)

print(f"Оптимальная стратегия игрока 1: p = {p:.3f}")

print(f"Оптимальная стратегия игрока 2: q = {q:.3f}")

print(f"Цена игры (нормализованная): v = {v:.3f}")

# Графическое представление

p\_values = np.linspace(0, 1, 100)

E1 = a \* p\_values + c \* (1 - p\_values) # Ожидаемый выигрыш против столбца 1

E2 = b \* p\_values + d \* (1 - p\_values) # Ожидаемый выигрыш против столбца 2

plt.figure(figsize=(10, 6))

plt.plot(p\_values, E1, 'b-', label='Против столбца 1')

plt.plot(p\_values, E2, 'r-', label='Против столбца 2')

plt.axhline(y=v, color='g', linestyle='--', label=f'Цена игры v = {v:.3f}')

plt.axvline(x=p, color='m', linestyle='--', label=f'Оптимальная p = {p:.3f}')

plt.xlabel('Вероятность выбора первой строки (p)')

plt.ylabel('Ожидаемый выигрыш')

plt.title('Графоаналитический метод: поиск оптимальной стратегии')

plt.legend()

plt.grid(True)

plt.show()

else:

print("Матрица не 2x2, графический метод не применим")

# ===== 5. АНАЛИТИЧЕСКИЙ (МАТРИЧНЫЙ) МЕТОД =====

print("\n5. АНАЛИТИЧЕСКИЙ (МАТРИЧНЫЙ) МЕТОД")

print("=" \* 50)

if B.shape == (2, 2):

# Формулы для матричной игры 2x2

det = a\*d - b\*c

trace = a + d - b - c

p\_matrix = (d - c) / trace

q\_matrix = (d - b) / trace

v\_matrix = det / trace

print(f"Оптимальная стратегия игрока 1: p = {p\_matrix:.3f}")

print(f"Оптимальная стратегия игрока 2: q = {q\_matrix:.3f}")

print(f"Цена игры (нормализованная): v = {v\_matrix:.3f}")

# ===== 6. СИМПЛЕКС-МЕТОД =====

print("\n6. СИМПЛЕКС-МЕТОД")

print("=" \* 50)

# Задача линейного программирования для игрока 1 (минимизация максимальных потерь)

if B.shape == (2, 2):

# Целевая функция: минимизировать v

c = [0, 0, 1] # [x1, x2, v]

# Ограничения: B^T \* x >= v, x1 + x2 = 1, x1, x2 >= 0

A\_ub = [

[-B[0, 0], -B[1, 0], 1], # -a\*x1 - c\*x2 + v <= 0

[-B[0, 1], -B[1, 1], 1] # -b\*x1 - d\*x2 + v <= 0

]

b\_ub = [0, 0]

# Ограничение равенства: x1 + x2 = 1

A\_eq = [[1, 1, 0]]

b\_eq = [1]

# Границы переменных

bounds = [(0, None), (0, None), (None, None)]

# Решаем задачу ЛП

result = linprog(c, A\_ub=A\_ub, b\_ub=b\_ub, A\_eq=A\_eq, b\_eq=b\_eq, bounds=bounds)

if result.success:

x1, x2, v\_simplex = result.x

print(f"Оптимальная стратегия игрока 1: x1 = {x1:.3f}, x2 = {x2:.3f}")

print(f"Цена игры (нормализованная): v = {v\_simplex:.3f}")

else:

print("Симплекс-метод не смог найти решение")

# ===== 7. РАСЧЕТ ЦЕНЫ ИГРЫ ДЛЯ ИСХОДНОЙ МАТРИЦЫ =====

print("\n7. РАСЧЕТ ЦЕНЫ ИГРЫ ДЛЯ ИСХОДНОЙ МАТРИЦЫ")

print("=" \* 50)

# Цена игры для исходной матрицы

v\_original = v - k

print(f"Цена игры для нормализованной матрицы: {v:.3f}")

print(f"Константа нормализации: k = {k}")

print(f"Цена игры для исходной матрицы: v = {v:.3f} - {k} = {v\_original:.3f}")

# ===== ВЫВОД ОПТИМАЛЬНЫХ СТРАТЕГИЙ =====

print("\nОПТИМАЛЬНЫЕ СМЕШАННЫЕ СТРАТЕГИИ")

print("=" \* 50)

# Восстанавливаем стратегии для исходной матрицы 5x5

if B.shape == (2, 2) and len(kept\_rows) == 2 and len(kept\_cols) == 2:

# Стратегия игрока 1

p1\_optimal = np.zeros(5)

p1\_optimal[kept\_rows[0]] = p

p1\_optimal[kept\_rows[1]] = 1 - p

# Стратегия игрока 2

p2\_optimal = np.zeros(5)

p2\_optimal[kept\_cols[0]] = q

p2\_optimal[kept\_cols[1]] = 1 - q

print("Оптимальная стратегия игрока 1:")

for i, prob in enumerate(p1\_optimal):

print(f" Стратегия {i+1}: {prob:.3f}")

print("\nОптимальная стратегия игрока 2:")

for i, prob in enumerate(p2\_optimal):

print(f" Стратегия {i+1}: {prob:.3f}")

print(f"\nИтоговая цена игры: {v\_original:.3f}")