МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №1

по дисциплине «Теория принятия решений»

Тема: Принятие решений в матричных играх

Студент гр.8382 _____ Нечепуренко Н.А.

Преподаватель _____ Попова Е.В.

Санкт-Петербург

2022

Цели работы.

Изучение различных инструментальных средств для решения задач поддержки принятия решения, а также овладение навыками принятия решения на основе матричных игр.

Основные теоретические положения.

Рассмотрим простейшую математическую модель конечной конфликтной ситуации, в которой имеется два участника и выигрыш одного равен проигрышу другого. Такая модель называется антагонистической игрой двух лиц с нулевой суммой. Игра состоит из двух ходов: игрок А выбирает одну из возможных $a_i, i = 1..m$ стратегий, а игрок Б выбирает одну из возможных стратегий $b_j, j = 1..n$. Каждый выбор производится при полном незнании выбора соперника. В результате выигрыш игроков составит соответственно a_{ij} и $-a_{ij}$. Цель игрока А – максимизировать величину a_{ij}, а игрока Б – минимизировать эту величину. Критерием принятия решения является функция, выражающая предпочтение лица принимающего решение и определяющая правило, по которому выбирается приемлемый или оптимальный вариант решения. Матрица, составленная из величин $a_{ij}, i = 1..m, j = 1..n$

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

называется платежной матрицей, или матрицей игры. Каждый элемент платежной матрицы $a_{ij}, i = 1..m, j = 1..n$, равен выигрышу А (проигрышу Б), если он выбрал стратегию $A_i, i = 1..m$, а игрок Б выбирал стратегию $B_j, j = 1..n$. Задача каждого из игроков – найти наилучшую стратегию иг-

ры, при этом предполагается, что противники одинаково разумны и каждый из них делает всё, чтобы получить наибольший доход. Найдем наилучшую стратегию первого игрока. Если игрок А выбрал стратегию $A_i, i = 1..m$, то в худшем случае (например, если его ход известен Б) он получит выигрыш $\alpha_i = \min_j a_{ij}$. Предвидя такую возможность, игрок А должен выбрать такую стратегию, чтобы максимизировать свой минимальный выигрыш.

$$\alpha = \max_i \alpha_i = \max_i \{\min_j a_{ij}\}$$

Величина α – гарантированный выигрыш игрока А называется нижней ценой игры. Стратегия $A_{\hat{i}}$, обеспечивающая получение выигрыша α , называется максиминной. Если первый игрок будет придерживаться своей максиминной стратегии, то у него есть гарантия, что он в любом случае выиграет не меньше α . Аналогично определяется наилучшая стратегия второго игрока. Игрок Б при выборе стратегии $B_j, j = 1..n$, в худшем случае получит проигрыш $\beta_j = \max_i a_{ij}$. Он выбирает стратегию $B_{\hat{j}}$, при которой его проигрыш будет минимальным и составит

$$\beta = \min_j \beta_j = \min_j \{\max_i a_{ij}\}$$

Величина β – гарантированный проигрыш игрока Б называется верхней ценой игры. Стратегия $B_{\hat{j}}$, обеспечивающая получение проигрыша β , называется минимаксной. Если второй игрок будет придерживаться своей минимаксной стратегии, то у него есть гарантия, что он в любом случае проиграет не больше β . Фактический выигрыш игрока А (проигрыш игрока Б) при разумных действиях партнеров ограничен верхней и нижней ценой игры. Для матричной игры справедливо неравенство $\alpha \leq \beta$. Если $\alpha = \beta = v$, т.е.

$$\max_i \{\min_j a_{ij}\} = \min_j \{\max_i a_{ij}\} = v$$

, то выигрыш игрока А (проигрыш игрока Б) определяется числом v . Оно называется ценой игры. Если $\alpha = \beta = v$, то такая игра называется игрой с седловой точкой. Элемент матрицы $a_{\hat{i}\hat{j}}$, соответствующий паре оптимальных стратегий $(A_{\hat{i}}, B_{\hat{j}})$, называется седловой точкой матрицы. Этот элемент является ценой игры. Седловой точке соответствуют оптимальные стратегии игроков. Их совокупность – решение игры, которое обладает свойством: если один из игроков придерживается оптимальной стратегии, то второму отклонение от своей оптимальной стратегии не может быть выгодным. Если игра имеет седловую точку, то говорят, что она решается в чистых стратегиях. Если платежная матрица не имеет седловой точки, т.е. $\alpha < \beta$ и $\alpha \leq v \leq \beta$ то поиск решения игры приводит к применению сложной стратегии, состоящей в случайном применении двух и более стратегий с определенными частотами. Сложная стратегия, состоящая в случайном применении всех стратегий с определенными частотами, называется смешанной.

Постановка задачи и порядок выполнения работы.

Используя инструментальные средства компьютерной алгебры решить матричные задачи.

1. Ознакомится с инструментальным средством для выполнения работы, указанный преподавателем, или выбрать его самостоятельно.
2. С помощью инструментального средства определить границы выигрыша и наличие седловой точки для матрицы C_1.
3. Графически и аналитически решить матричную игру 2×2 для матрицы C_2.
4. Графически и аналитически решить матричную игру 2×N для матрицы C_3.
5. Графически и аналитически решить матричную игру M×2 для матрицы

C_4.

6. С помощью симплекс-метода решить матричную игру M×N для матрицы C_5.

Выполнение работы.

С помощью инструментального средства определить границы выигрыша и наличие седловой точки для матрицы C1.

Матрица C_1:

$$C_1 = \begin{pmatrix} 3 & 5 & 4 & 1 \\ 4 & 6 & 3 & 5 \\ 2 & 3 & 4 & 6 \end{pmatrix}$$

Минимальные значения по строкам: $(1, 3, 2)$, максимальные по столбцам: $(4, 6, 4, 6)$. Таким образом, максиминное значение равно 3, минимаксное 4, откуда следует, что в чистых стратегиях нет седловой точки. Границы выигрыша $3 \leq V \leq 4$.

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

Матрица C_2:

$$C_2 = \begin{pmatrix} 8 & 2 \\ 1 & 5 \end{pmatrix}$$

Попробуем найти решение матричной игры аналитически, в чистых стратегиях. Максиминное значение 2, минимаксное 5, значит решения в чистых стратегиях нет.

Попробуем найти решение в смешанных стратегиях, для этого необхо-

димо найти решение системы

$$\begin{cases} 8p_1 + p_2 = V \\ 2p_1 + 5p_2 = V \\ p_1 + p_2 = 1 \end{cases}$$

для первого игрока и

$$\begin{cases} 8q_1 + 2q_2 = V \\ q_1 + 5q_2 = V \\ q_1 + q_2 = 1 \end{cases}$$

для второго.

Вычислить неизвестные можно по известным формулам:

$$p_1 = \frac{c_{22} - c_{21}}{c_{11} + c_{22} - (c_{12} + c_{21})}$$

$$p_2 = 1 - p_1$$

$$q_1 = \frac{c_{22} - c_{12}}{c_{11} + c_{22} - (c_{12} + c_{21})}$$

$$q_2 = 1 - q_1$$

$$V = \frac{c_{22}c_{11} - c_{12}c_{21}}{c_{11} + c_{22} - (c_{12} + c_{21})}$$

Получаем $p_1 = 0.4, p_2 = 0.6, q_1 = 0.3, q_2 = 0.7, V = 3.8$.

Решим задачу графически (см. рис. 1):

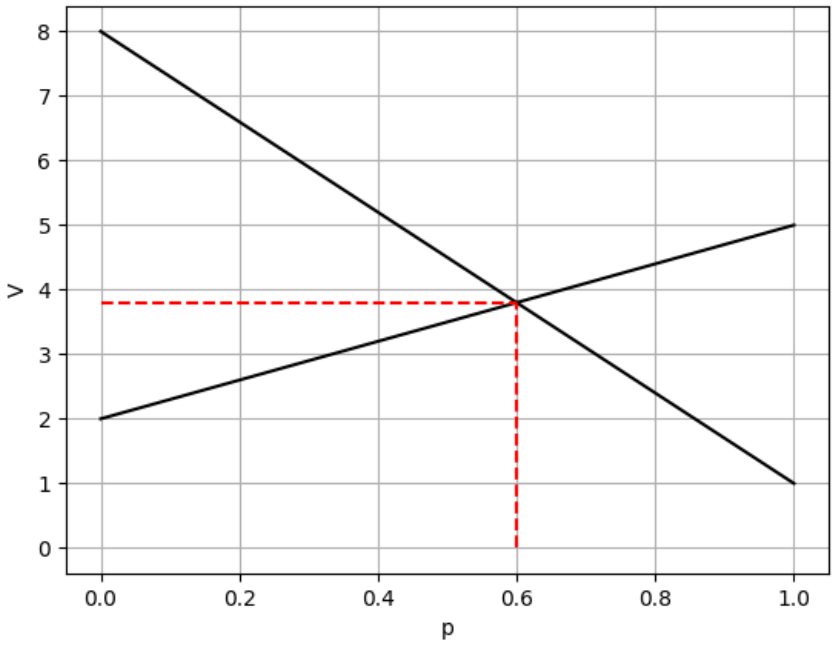

Рисунок 1 – Графическое решение игры 2x2

Значение $p_2 \approx 0.6, V \approx 0.9$. Вычислим относительную погрешность графического метода:

$$\delta(p_2) = \frac{0.6 - 0.6}{0.6} = 0\%$$
$$\delta(V) = \frac{0.9 - 0.8}{0.8} = 12.5\%$$

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

Матрица C_3:

$$C_3 = \begin{pmatrix} 4 & 3 & 2 & 5 \\ 6 & 1 & 7 & 4 \end{pmatrix}$$

Попробуем найти решение матричной игры аналитически, в чистых стратегиях. Максиминное значение 2, минимаксное 3, значит решения в чистых стратегиях нет.

Построим графическое решение (см. рис. 2):

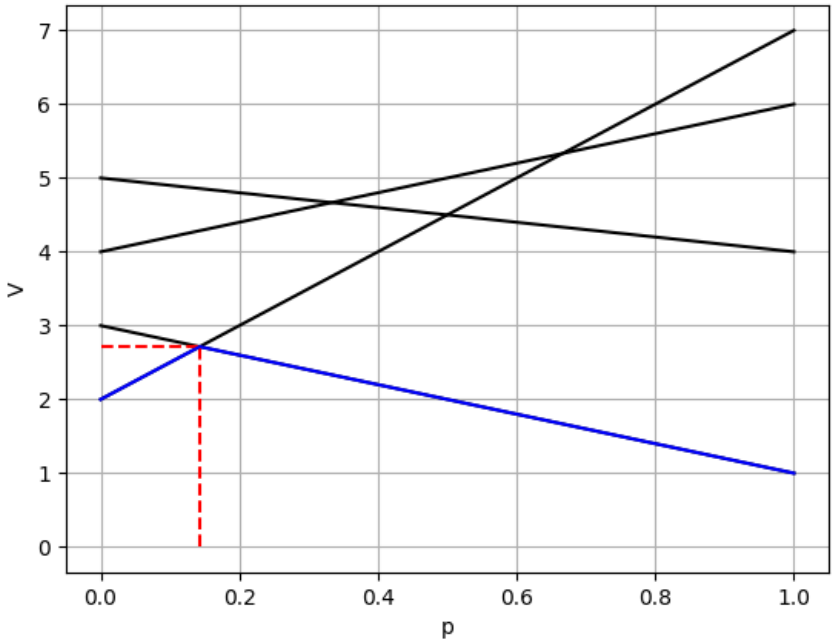

Рисунок 2 – Графическое решение игры 2xN

Чтобы решить задачу аналитически, оставим только столбцы 2 и 3, согласно рисунку, и сведем задачу к предыдущей (2x2). Получаем $p_1 \approx 0.86, p_2 \approx 0.14, q_1 \approx 0.29, q_2 \approx 0.71, V \approx 2.71$.

Графически положим $p_2 \approx 0.15, V \approx 2.75$, тогда относительные погрешности равны

$$\delta(p_2) = \frac{0.15 - 0.14}{0.14} \approx 7\%$$
$$\delta(V) = \frac{2.75 - 2.71}{2.71} \approx 1\%$$

Графически и аналитически решить матричную игру M×2 для матрицы C4.

Матрица C_4:

$$C_4 = \begin{pmatrix} 4 & 3 \\ 1 & 6 \\ 2 & 3 \\ 3 & 5 \end{pmatrix}$$

Попробуем найти решение матричной игры аналитически, в чистых стратегиях. Максиминное значение 3, минимаксное 4, значит решения в чистых стратегиях нет.

Построим графическое решение в смешанных стратегиях (см. рис. 3):

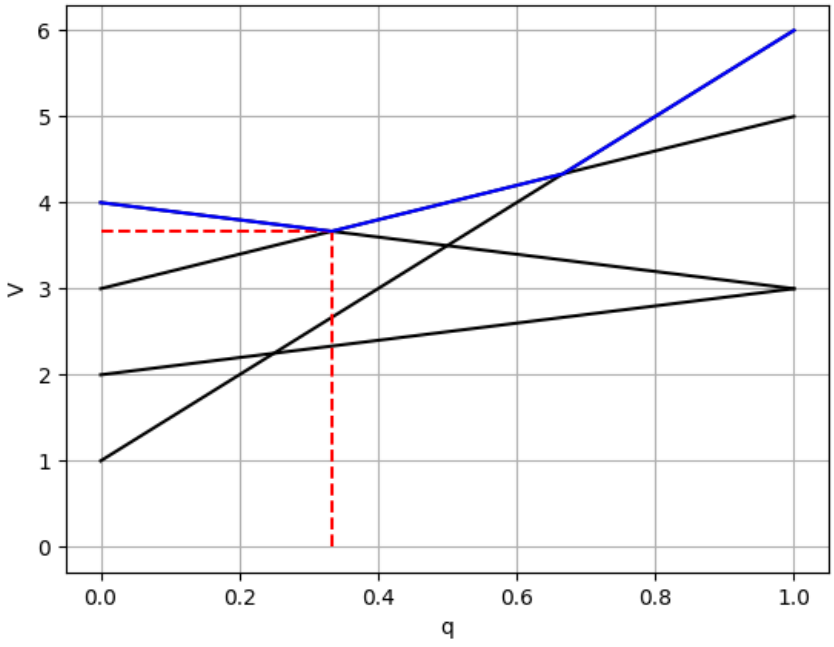

Рисунок 3 – Графическое решение игры Mx2

Чтобы решить задачу, оставим 1 и 4 строки, сведем задачу к задаче 2x2. Аналитически получаем $p_1 = \frac{2}{3}, p_2 = \frac{1}{3}, q_1 = \frac{2}{3}, q_2 = \frac{1}{3}, V = \frac{11}{3}$.

Графически положим $q_2 \approx 0.35, V \approx 3.6$, тогда относительные погрешности примерно равны:

$$\delta(q_2) = \frac{0.35 - \frac{1}{3}}{\frac{1}{3}} \approx 5\%$$
$$\delta(V) = \frac{\frac{11}{3} - 3.6}{\frac{11}{3}} \approx 2\%$$

С помощью симплекс-метода решить матричную игру M×N для матрицы C5.

Решение игры MxN в смешанных стратегиях представляется как

$$\begin{cases} c_{11}p_1 + c_{21}p_2 + \cdots + c_{m1}p_m \geq V \\ c_{12}p_1 + c_{22}p_2 + \cdots + c_{m2}p_m \geq V \\ \cdots \\ c_{1n}p_1 + c_{2n}p_2 + \cdots + c_{mn}p_m \geq V \end{cases}$$

Выполним замену вида $x_i = \frac{p_i}{V}$, переходим к системе

$$\begin{cases} c_{11}x_1 + c_{21}x_2 + \cdots + c_{m1}x_m \geq 1 \\ c_{12}x_1 + c_{22}x_2 + \cdots + c_{m2}x_m \geq 1 \\ \cdots \\ c_{1n}x_1 + c_{2n}x_2 + \cdots + c_{mn}x_m \geq 1 \end{cases}$$

Заметим, что $\sum_i x_i = \frac{1}{V} = Z$. Стратегия первого игрока должна максимизировать V, другими словами, минимизировать Z.

Таким образом, задача поиска решения матричной игры сводится к задаче линейного программирования. Решим ее с помощью метода linprog пакета scipy.

Матрица C_5:

$$\begin{pmatrix} -2 & 8 & 5 & 7 \\ -3 & 9 & 3 & 5 \\ 8 & -1 & 9 & 3 \end{pmatrix}$$

С помощью симплекс-метода получаем следующие значения:

$$p_1 = 0$$

$$p_2 = 0.429$$

$$p_3 = 0.571$$

$$q_1 = 0.476$$

$$q_2 = 0.524$$

$$q_3 = 0$$

$$q_4 = 0$$

$$V = 3.286$$

Исходный код программы приведен в Приложении А.

Выводы.

В результате выполнения работы были получены решения для матричных игр с нулевой суммой для форматов 2x2, 2xN, Mx2 аналитически и графически. Для графических решений была вычислена относительная погрешность. Для игры MxN было получено решение с использованием симплекс-метода.

ПРИЛОЖЕНИЕ А. ИСХОДНЫЙ КОД ПРОГРАММЫ.

```
import numpy as np
from matplotlib import pyplot as plt
from scipy.optimize import linprog


def solve_2x2(arr):
    arr = np.array([
        [8, 2],
        [1, 5]
    ])
    p1 = (arr[1][1] - arr[1][0]) / (arr[0][0] + arr[1][1] - arr
       [0][1] - arr[1][0])
    p2 = 1 - p1
    q1 = (arr[1][1] - arr[0][1]) / (arr[0][0] + arr[1][1] - arr
       [0][1] - arr[1][0])
    q2 = 1 - q1
    v = (arr[0][0] * arr[1][1] - arr[0][1] * arr[1][0]) / \
        (arr[0][0] + arr[1][1] - arr[0][1] - arr[1][0])
    print('p1: ', p1, '\np2: ', p2, '\nq1: ', q1, '\nq2: ', q2,
       '\nV: ', v)
    return p1, p2, q1, q2, v


def task1():
    c_1 = np.array([
        [3, 5, 4, 1],
        [4, 6, 3, 5],
        [2, 3, 4, 6]
    ])

    print(c_1.min(axis=1))
```

```
    print(c_1.max(axis=0))
    print(max(c_1.min(axis=1)))
    print(min(c_1.max(axis=0)))


def task2():
    c_2 = np.array([
        [8, 2],
        [1, 5]
    ])
    p1, p2, q1, q2, v = solve_2x2(c_2)

    plt.figure()
    plt.grid(True)
    plt.xlabel("p")
    plt.ylabel("V")
    x = [0, 1]
    plt.plot(x, [c_2[0][0], c_2[1][0]], "k")
    plt.plot(x, [c_2[0][1], c_2[1][1]], "k")
    plt.plot([p2, p2], [0., v], 'r--')
    plt.plot([0., p2], [v, v], 'r--')
    plt.show()


def task3():
    c_3 = np.array([
        [4, 3, 2, 5],
        [6, 1, 7, 4]
    ])
    p1, p2, q1, q2, v = solve_2x2(
        np.array([
            [2, 3],
```

```
            [7, 1]
        ]))

    plt.figure()
    plt.grid(True)
    plt.xlabel("p")
    plt.ylabel("V")
    x = [0, 1]
    line = np.zeros(shape=(4, 2))
    print(c_3.shape[1])
    for i in range(0, c_3.shape[1]):
        line[i] = [c_3[0][i], c_3[1][i]]
        plt.plot(x, line[i], "k")
    plt.plot([p2, p2], [0., v], 'r--')
    plt.plot([0., p2], [v, v], 'r--')
    plt.plot([0., 1 / 7., 1.], [2., 5 / 7. + 2, 1.], "b")
    plt.show()


def task4():
    c_4 = np.array([
        [4, 3],
        [1, 6],
        [2, 3],
        [3, 5]
    ])
    p1, p2, q1, q2, v = solve_2x2(
        np.array([
            [4, 3],
            [3, 5]
        ]))
```

```
    plt.figure()
    plt.grid(True)
    plt.xlabel("q")
    plt.ylabel("V")
    x = [0, 1]
    line = np.zeros(shape=(4, 2))
    for i in range(0, c_4.shape[0]):
        line[i] = [c_4[i][0], c_4[i][1]]
        plt.plot(x, line[i], "k")
    plt.plot([q2, q2], [0., v], 'r--')
    plt.plot([0., q2], [v, v], 'r--')
    plt.plot([0., q2, 2 / 3, 1.], [4., v, 13 / 3, 6.], "b")
    plt.show()


def task5():
    c_5 = np.array([
        [-2, 8, 5, 7],
        [-3, 9, 3, 5],
        [8, -1, 9, 3]
    ])
    dim = c_5.shape[0]
    c = np.ones(dim)
    b = np.ones(c_5.shape[1]) * (-1)
    A = c_5.T * (-1)
    solution = linprog(c=c, A_ub=A, b_ub=b, method='simplex')
    v = np.reciprocal(solution.fun)
    print("V = ", "{:1.3f}".format(v))
    for i in range(0, dim):
        print('p' + str(i + 1) + ' = ' + str(np.round(solution.x[
           i] * v, 3)))
    dim = c_5.shape[1]
```

```
    c = -np.ones(dim)
    A = c_5
    b = np.ones(c_5.shape[0])
    solution = linprog(c=c, A_ub=A, b_ub=b, method='simplex')
    v = -np.reciprocal(solution.fun)
    print("V = ", "{:.3f}".format(v))
    for i in range(0, dim):
        print('q' + str(i + 1) + ' = ' + str(np.round(solution.x[
           i] * v, 3)))


if __name__ == "__main__":
    task5()
```