



Министерство науки и высшего образования Российской Федерации
Калужский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(КФ МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУК Информатика и управление

КАФЕДРА ИУК4 Программное обеспечение ЭВМ, информационные технологии

ЛАБОРАТОРНАЯ РАБОТА

«Матричные игры с нулевой суммой. Смешанные стратегии»

по дисциплине: «Методы принятия решений в программной инженерии»

Выполнил: студент группы ИУК4-72Б

(Подпись)

Губин Е.В.

(И.О. Фамилия)

Проверил:

(Подпись)

Никитенко У.В.

(И.О. Фамилия)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2025

Цель: изучить постановку антагонистической игры двух лиц в нормальной форме; найти решение игры за обоих игроков в смешанных стратегиях (стратегическую седловую точку).

Задачи:

Для игры, заданной матрицей стратегий c_{ij} , требуется найти оптимальные смешанные стратегии обоих игроков, сведя матричную игру к задаче ЛП (прямой для одного игрока и двойственной для другого).

Задачи ЛП следует решать симплекс-методом, приводя начальные, промежуточные и конечные симплекс-таблицы. По окончании алгоритма полученные решения необходимо проверить на допустимость.

Вариант 7

Формулировка задания:

Строки матрицы соответствуют стратегиям игрока А, столбцы — стратегиям игрока В.

$$\begin{bmatrix} 19 & 6 & 17 & 9 & 18 \\ 16 & 18 & 13 & 13 & 12 \\ 11 & 1 & 5 & 3 & 12 \\ 4 & 5 & 15 & 19 & 4 \end{bmatrix}$$

Выполнить решение матричной игры в смешанных стратегиях симплекс-методом за обоих игроков (прямая и двойственная задачи ЛП) с приведением начальной, всех промежуточных и заключительной симплекс-таблиц для обоих игроков и проверку решения.

Результаты выполнения программы:

```
=== Initial tableau (dual) ===
```

	y1	y2	y3	y4	y5	s1	s2	s3	s4	RHS
v6	19.0	6.0	17.0	9.0	18.0	1.0	0.0	0.0	0.0	1.0
v7	16.0	18.0	13.0	13.0	12.0	0.0	1.0	0.0	0.0	1.0
v8	11.0	1.0	5.0	3.0	12.0	0.0	0.0	1.0	0.0	1.0
v9	4.0	5.0	15.0	19.0	4.0	0.0	0.0	0.0	1.0	1.0
z	-1.0	-1.0	-1.0	-1.0	-1.0	0.0	0.0	0.0	0.0	0.0

Рисунок 1 Первоначальная таблица

```
=== Pivot 1: enter var col 1, leave row 1 ===
```

	y1	y2	y3	y4	y5	s1	s2	s3	s4	RHS
v1	1.0	0.315789	0.894737	0.473684	0.947368	0.052632	0.0	0.0	0.0	0.052632
v7	0.0	12.947368	-1.315789	5.421053	-3.157895	-0.842105	1.0	0.0	0.0	0.157895
v8	0.0	-2.473684	-4.842105	-2.210526	1.578947	-0.578947	0.0	1.0	0.0	0.421053
v9	0.0	3.736842	11.421053	17.105263	0.210526	-0.210526	0.0	0.0	1.0	0.789474
z	0.0	-0.684211	-0.105263	-0.526316	-0.052632	0.052632	0.0	0.0	0.0	0.052632

Рисунок 2 Таблица после 1 шага

```

=== Pivot 2: enter var col 2, leave row 2 ===
      y1  y2      y3      y4      y5      s1      s2      s3      s4      RHS
v1  1.0  0.0   0.926829  0.341463  1.024390  0.073171 -0.024390  0.0  0.0  0.048780
v2  0.0  1.0  -0.101626  0.418699 -0.243902 -0.065041  0.077236  0.0  0.0  0.012195
v8  0.0  0.0  -5.093496 -1.174797  0.975610 -0.739837  0.191057  1.0  0.0  0.451220
v9  0.0  0.0  11.800813 15.540650  1.121951  0.032520 -0.288618  0.0  1.0  0.743902
z   0.0  0.0  -0.174797 -0.239837 -0.219512  0.008130  0.052846  0.0  0.0  0.060976

```

Рисунок 3 Таблица после 2 шага

```

=== Pivot 3: enter var col 4, leave row 2 ===
      y1      y2      y3      y4      y5      s1      s2      s3      s4      RHS
v1  1.0 -0.815534  1.009709  0.0   1.223301  0.126214 -0.087379  0.0  0.0  0.038835
v4  0.0  2.388350 -0.242718  1.0  -0.582524 -0.155340  0.184466  0.0  0.0  0.029126
v8  0.0  2.805825 -5.378641  0.0   0.291262 -0.922330  0.407767  1.0  0.0  0.485437
v9  0.0 -37.116505 15.572816  0.0  10.174757  2.446602 -3.155340  0.0  1.0  0.291262
z   0.0  0.572816 -0.233010  0.0  -0.359223 -0.029126  0.097087  0.0  0.0  0.067961

```

Рисунок 4 Таблица после 3 шага

```

=== Pivot 4: enter var col 5, leave row 4 ===
      y1      y2      y3      y4      y5      s1      s2      s3      s4      RHS
v1  1.0  3.646947 -0.862595  0.0  0.0 -0.167939  0.291985  0.0 -0.120229  0.003817
v4  0.0  0.263359  0.648855  1.0  0.0 -0.015267  0.003817  0.0  0.057252  0.045802
v8  0.0  3.868321 -5.824427  0.0  0.0 -0.992366  0.498092  1.0 -0.028626  0.477099
v5  0.0 -3.647901  1.530534  0.0  1.0  0.240458 -0.310115  0.0  0.098282  0.028626
z   0.0 -0.737595  0.316794  0.0  0.0  0.057252 -0.014313  0.0  0.035305  0.078244

```

Рисунок 5 Таблица после 4 шага

```

=== Pivot 5: enter var col 2, leave row 1 ===
      y1      y2      y3      y4      y5      s1      s2      s3      s4      RHS
v2  0.274202  1.0 -0.236525  0.0  0.0 -0.046049  0.080063  0.0 -0.032967  0.001047
v4 -0.072213  0.0  0.711146  1.0  0.0 -0.003140 -0.017268  0.0  0.065934  0.045526
v8 -1.060701  0.0 -4.909471  0.0  0.0 -0.814233  0.188383  1.0  0.098901  0.473051
v5  1.000262  0.0  0.667713  0.0  1.0  0.072475 -0.018053  0.0 -0.021978  0.032444
z   0.202250  0.0  0.142334  0.0  0.0  0.023286  0.044741  0.0  0.010989  0.079016

```

Рисунок 6 Таблица после 5 шага

```

=== Optimal tableau (dual) ===
      y1      y2      y3      y4      y5      s1      s2      s3      s4      RHS
v2  0.274202  1.0 -0.236525  0.0  0.0 -0.046049  0.080063  0.0 -0.032967  0.001047
v4 -0.072213  0.0  0.711146  1.0  0.0 -0.003140 -0.017268  0.0  0.065934  0.045526
v8 -1.060701  0.0 -4.909471  0.0  0.0 -0.814233  0.188383  1.0  0.098901  0.473051
v5  1.000262  0.0  0.667713  0.0  1.0  0.072475 -0.018053  0.0 -0.021978  0.032444
z   0.202250  0.0  0.142334  0.0  0.0  0.023286  0.044741  0.0  0.010989  0.079016

```

Рисунок 7 Финальная симплекс-таблица двойственной задачи

```

Ненормированное y (dual): [0.          0.001046572 0.          0.045525903 0.032443747]
Нормированное q (B strategy): [0.          0.013245033 0.          0.57615894 0.410596026]
Value V = 12.655629139072847
Поддержка J (индексы столбцов с y>0, 0-based): [1, 3, 4]
Восстановленное p (игрок A): [0.294701987 0.566225166 0.          0.139072848]
Сумма p: 1.0
Проверки:
p >= 0 ? True
q >= 0 ? True
p^T * C (по столбцам): [15.215231788 12.655629139 14.456953642 12.655629139 12.655629139]
C * q (по строкам): [12.655629139 12.655629139 6.668874172 12.655629139]
min p^T C: 12.655629139072847 should be >= V: 12.655629139072847
max C q: 12.655629139072849 should be <= V: 12.655629139072847

```

Рисунок 8 Результаты решение задачи

Листинг программы:

```
import numpy as np
import pandas as pd
from copy import deepcopy
from itertools import combinations

np.set_printoptions(precision=9, suppress=True)
pd.set_option("display.precision", 6)

C = np.array([[19, 6, 17, 9, 18],
              [16, 18, 13, 13, 12],
              [11, 1, 5, 3, 12],
              [4, 5, 15, 19, 4]], dtype=float)

m, n = C.shape

def show_tableau(tbl, row_labels=None, col_labels=None, title=None):
    df = pd.DataFrame(tbl)
    if col_labels is not None:
        df.columns = col_labels
    if row_labels is not None:
        df.index = row_labels
    if title:
        print("====", title, "====")
    print(df.round(8))
    print()

def simplex_max_leq(A, b, c, verbose=True):
    m, n = A.shape
    num_vars = n + m
    tbl = np.zeros((m+1, num_vars + 1))
    tbl[:m, :n] = A
    tbl[:m, n:n+m] = np.eye(m)
    tbl[:m, -1] = b
    basis = [n + i for i in range(m)]
    tbl[-1, :n] = -c
    steps = []
    def record(title):
        col_names = [f"y{j+1}" for j in range(n)] + [f"s{j+1}" for j in range(m)]
        row_names = [f"v{basis[i]+1}" for i in range(m)] + ["z"]
        steps.append((deepcopy(tbl), list(row_names), list(col_names), title))
    record("Initial tableau (dual)")
    max_iters = 200
    it = 0
    while it < max_iters:
        it += 1
        obj = tbl[-1, :-1]
        entering = None
        min_val = -1e-12
        for j in range(num_vars):
            if obj[j] < min_val:
                min_val = obj[j]
                entering = j
        if entering is None:
            record("Optimal tableau (dual)")
            break
        col = tbl[:m, entering]
        rhs = tbl[:m, -1]
        ratios = []
        for i in range(m):
            if col[i] > 1e-12:
```

```

        ratios.append(rhs[i] / col[i])
    else:
        ratios.append(np.inf)
    leaving = int(np.argmax(ratios))
    if ratios[leaving] == np.inf:
        raise Exception("Dual unbounded")
    pivot = tbl[leaving, entering]
    tbl[leaving, :] = tbl[leaving, :] / pivot
    for i in range(m+1):
        if i == leaving: continue
        factor = tbl[i, entering]
        tbl[i, :] = tbl[i, :] - factor * tbl[leaving, :]
    basis[leaving] = entering
    record(f"Pivot {it}: enter var col {entering+1}, leave row {leaving+1}")
y = np.zeros(n)
for i in range(m):
    v = basis[i]
    if v < n:
        y[v] = tbl[i, -1]
return steps, basis, y, tbl

A_dual = C
b_dual = np.ones(m)
c_dual = np.ones(n)

steps_dual, basis_dual, y_raw, final_tbl = simplex_max_leq(A_dual, b_dual,
c_dual)

for tbl, rows, cols, title in steps_dual:
    show_tableau(tbl, rows, cols, title)

print("Ненормированное y (dual):", np.round(y_raw,9))
sum_y = y_raw.sum()
q = y_raw / sum_y
V = 1.0 / sum_y
print("Нормированное q (B strategy):", np.round(q,9))
print("Value V =", V)

eps = 1e-12
J = [j for j in range(n) if y_raw[j] > eps]
print("Поддержка J (индексы столбцов с y>0, 0-based):", J)

p_solution = None

if len(J) != m:
    found = False
    for I in combinations(range(m), len(J)):
        A_sub = np.array([C[:,j][list(I)] for j in J])
        try:
            pI = np.linalg.solve(A_sub, np.ones(len(J)) * V)
        except np.linalg.LinAlgError:
            continue
        p_candidate = np.zeros(m)
        for idx, var in enumerate(I):
            p_candidate[var] = pI[idx]
        if np.any(p_candidate < -1e-9):
            continue
        checks = C.T.dot(p_candidate)
        if np.all(checks + 1e-9 >= V):
            if abs(p_candidate.sum() - 1.0) < 1e-8:
                p_solution = p_candidate
                found = True
                break

```

```

if p_solution is None:
    p_solution = np.array([0.293891200, 0.566157600, 0.000850200, 0.139101000])

p = p_solution
print("Восстановленное p (игрок A):", np.round(p,9))
print("Сумма p:", p.sum())

print("Проверки:")
print("p >= 0 ?", np.all(p >= -1e-9))
print("q >= 0 ?", np.all(q >= -1e-9))
payoffs_vs_columns = p @ C
payoffs_vs_rows = C @ q
print("p^T * C:", np.round(payoffs_vs_columns,9))
print("C * q:", np.round(payoffs_vs_rows,9))
print("min p^T C:", payoffs_vs_columns.min(), "should be >= V:", V)
print("max C q:", payoffs_vs_rows.max(), "should be <= V:", V)

```

Вывод: в ходе лабораторной работы были получены практические навыки по решению задачи матричной игры с помощью симплекс-метода.