МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«Дальневосточный федеральный университет»
(ДВФУ)

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Департамент математического и компьютерного моделирования

ЛАБОРАТОРНАЯ РАБОТА №3
По основной образовательной программе подготовки бакалавров
направлению 01.03.02 Прикладная математика и информатика
профиль «Системное программирование»

Студент группы Б9122-01.03.02сп4
Кириенко Денис Олегович_____
«_____» _____ 2024 г.

Преподаватель _____кфмн_____
(должность, ученое звание)

_____
_____Яковлев Анатолий Александрович
(подпись) (ФИО)

«_____» _____2024 г.

г. Владивосток
2024

Постановка задачи

Используя систему ограничений найти оптимальное (минимальное или максимальное) значение целевой функции Z при помощи прямого и двойственного симплекс-метода.

$$Z = c_1x_1 + c_2x_2 + ... + c_nx_n.$$

Система ограничений:

$$\begin{cases} c \cdot x \to \max \\ Ax \le b \\ x \ge 0 \end{cases}$$

Условия:
- c − неотрицательный 6-мерный вектор
- b − неотрицательный 8-мерный вектор
- x − неотрицательный b-мерный вектор неизвестных, который необходимо найти
- A − матрица 6x8

$$A = \begin{bmatrix} 15 & 115 & 106 & 290 & 232 & 167 \\ 79 & 247 & 7 & 286 & 65 & 276 \\ 219 & 125 & 174 & 42 & 114 & 202 \\ 287 & 213 & 225 & 274 & 169 & 260 \\ 202 & 124 & 211 & 200 & 174 & 183 \\ 158 & 265 & 1 & 39 & 113 & 290 \\ 175 & 196 & 170 & 270 & 187 & 178 \\ 245 & 100 & 226 & 63 & 245 & 259 \end{bmatrix}$$

$$b = \begin{bmatrix} 296 & 85 & 22 & 47 & 247 & 28 & 125 & 218 \end{bmatrix}$$

$$c = \begin{bmatrix} 173 & 299 & 240 & 120 & 249 & 86 \end{bmatrix}$$

Прямая задача

Составим симплекс-таблицу. Первая строка – расширенный вектор c. Будем решать задачу на минимум, поэтому возьмем вместо вектора c вектор -c.

Первый столбец - вектор b, далее матрица A, единичная, а последней строкой является вектор -c, слева от которого 0 - целевое значение функции, а справа - нули.

Видим, что в последней строке (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено.

Разрешающий столбец - такой столбец, у которого элемент строки целевой функции наибольший по модулю отрицательный элемент.

Разрешающая строка - строка, содержащая наименьшее положительное отношение свободного числа к элементу разрешающего столбца.

Разрешающий элемент - элемент, расположенный на пересечении разрешающих столбца и строки.

Итерационный процесс будет продолжаться, пока не исчезнет возможность найти разрешающий столбец. На каждой итерации будет искаться разрешающий элемент, а затем исключаться из таблицы путем деления разрешающей строки на разрешающий элемент - так получится единица на месте разрешающего элемента, а затем с помощью получившейся единицы будут исключаться остальные элементы разрешающего столбца путем вычитания из текущей строки ее же, умноженную на разрешающую строку. После исключения всех элементов разрешающего столбца, помимо единицы, начинается новая итерация.

Шаг 1

```
index: (5, 2)
focus func val: -299.0
focus val: 265.0
```

296	15	115	106	290	232	167	1	0	0	0	0	0	0	0
85	79	247	7	286	65	276	0	1	0	0	0	0	0	0
22	219	125	174	42	114	202	0	0	1	0	0	0	0	0
47	287	213	225	274	169	260	0	0	0	1	0	0	0	0
247	202	124	211	200	174	183	0	0	0	0	1	0	0	0
28	158	265	1	39	113	290	0	0	0	0	0	1	0	0
125	175	196	170	270	187	178	0	0	0	0	0	0	1	0
218	245	100	226	63	245	259	0	0	0	0	0	0	0	1
0	-173	-299	-240	-120	-249	-86	0	0	0	0	0	0	0	0

Шаг 2

```
index: (2, 3)
focus func val: -238.872
focus val: 173.528
```

283.849	-53.566	0	105.566	273.075	182.962	41.151	1	0	0	0	0	-0.434	0	0
58.902	-68.268	0	6.068	249.649	-40.325	5.698	0	1	0	0	0	-0.932	0	0
8.792	144.472	0	173.528	23.604	60.698	65.208	0	0	1	0	0	-0.472	0	0
24.494	160.004	0	224.196	242.653	78.174	26.906	0	0	0	1	0	-0.804	0	0
233.898	128.068	0	210.532	181.751	121.125	47.302	0	0	0	0	1	-0.468	0	0
0.106	0.596	1	0.004	0.147	0.426	1.094	0	0	0	0	0	0.004	0	0
104.291	58.14	0	169.26	241.155	103.423	-36.491	0	0	0	0	0	-0.74	1	0
207.434	185.377	0	225.623	48.283	202.358	149.566	0	0	0	0	0	-0.377	0	1
31.592	5.272	0	-238.872	-75.996	-121.502	241.208	0	0	0	0	0	1.128	0	0

Шаг 3

```
index: (3, 4)
focus func val: -43.504
focus val: 212.157
```

278.5	-141.455	0	0	258.716	146.037	1.482	1	0	-0.608	0	0	-0.147	0	0
58.594	-73.32	0	0	248.824	-42.447	3.418	0	1	-0.035	0	0	-0.916	0	0
0.051	0.833	0	1	0.136	0.35	0.376	0	0	0.006	0	0	-0.003	0	0
13.135	-26.652	0	0	212.157	-0.248	-57.342	0	0	-1.292	1	0	-0.194	0	0
223.231	-47.211	0	0	153.114	47.483	-31.811	0	0	-1.213	0	1	0.104	0	0
0.105	0.593	1	0	0.147	0.425	1.093	0	0	-0	0	0	0.004	0	0
95.714	-82.779	0	0	218.131	44.217	-100.094	0	0	-0.975	0	0	-0.28	1	0
196.002	-2.466	0	0	17.593	123.438	64.783	0	0	-1.3	0	0	0.236	0	1
43.696	204.145	0	0	-43.504	-37.947	330.969	0	0	1.377	0	0	0.479	0	0

Шаг 4

```
index: (2, 5)
focus func val: -37.998
focus val: 0.35
```

262.483	-108.955	0	0	0	146.338	71.407	1	0	0.967	-1.219	0	0.09	0	0
43.19	-42.062	0	0	0	-42.157	70.67	0	1	1.48	-1.173	0	-0.688	0	0
0.042	0.85	0	1	0	0.35	0.413	0	0	0.007	-0.001	0	-0.003	0	0
0.062	-0.126	0	0	1	-0.001	-0.27	0	0	-0.006	0.005	0	-0.001	0	0
213.751	-27.977	0	0	0	47.662	9.573	0	0	-0.281	-0.722	1	0.245	0	0
0.096	0.612	1	0	0	0.425	1.133	0	0	0.001	-0.001	0	0.004	0	0
82.21	-55.377	0	0	0	44.472	-41.138	0	0	0.353	-1.028	0	-0.08	1	0
194.913	-0.256	0	0	0	123.459	69.538	0	0	-1.193	-0.083	0	0.252	0	1
46.389	198.68	0	0	0	-37.998	319.211	0	0	1.112	0.205	0	0.439	0	0

Шаг 5

```
result:
```

244.816	-464.252	0	-418.173	0	0	-101.105	1	0	-1.789	-0.951	0	1.175	0	0
48.279	60.291	0	120.466	0	0	120.367	0	1	2.274	-1.25	0	-1	0	0
0.121	2.428	0	2.858	0	1	1.179	0	0	0.019	-0.002	0	-0.007	0	0
0.062	-0.123	0	0.003	1	0	-0.269	0	0	-0.006	0.005	0	-0.001	0	0
207.997	-143.695	0	-136.197	0	0	-46.614	0	0	-1.179	-0.634	1	0.598	0	0
0.045	-0.421	1	-1.215	0	0	0.631	0	0	-0.007	0	0	0.007	0	0
76.841	-163.351	0	-127.082	0	0	-93.564	0	0	-0.485	-0.947	0	0.25	1	0
180.008	-300.004	0	-352.794	0	0	-76.003	0	0	-3.518	0.143	0	1.167	0	1
50.976	290.937	0	108.583	0	0	364.006	0	0	1.827	0.135	0	0.158	0	0

В последней строке (не включая значение целевой функции) больше не осталось отрицательных элементов, а значит оптимальное решение найдено. Оно находится в крайнем левом нижнем положении.

Алгоритм выдал значение целевой функции 50.976.

Двойственная задача

Двойственный симплекс-метод является по сути тем же симплекс-методом, но инвертированным. Итерационная работа остается той же, меняется только симплекс-таблица.

Можно заметить, что теперь вектора c и b транспонируются и меняются местами (с изменением знаков), а матрица A транспонирована и отрицательна.

Шаг 1

```
index: (1, 6)
focus func val: -299.0
focus val: -265.0
```

-173	-15	-79	-219	-287	-202	-158	-175	-245	1	0	0	0	0	0
-299	-115	-247	-125	-213	-124	-265	-196	-100	0	1	0	0	0	0
-240	-106	-7	-174	-225	-211	-1	-170	-226	0	0	1	0	0	0
-120	-290	-286	-42	-274	-200	-39	-270	-63	0	0	0	1	0	0
-249	-232	-65	-114	-169	-174	-113	-187	-245	0	0	0	0	1	0
-86	-167	-276	-202	-260	-183	-290	-178	-259	0	0	0	0	0	1
0	-296	-85	-22	-47	-247	-28	-125	-218	0	0	0	0	0	0

Шаг 2

```
index: (2, 3)
focus func val: -238.872
focus val: -173.528
```

5.272	53.566	68.268	-144.472	-160.004	-128.068	0	-58.14	-185.377	1	-0.596	0	0	0	0
1.128	0.434	0.932	0.472	0.804	0.468	1	0.74	0.377	-0	-0.004	-0	-0	-0	-0
-238.872	-105.566	-6.068	-173.528	-224.196	-210.532	0	-169.26	-225.623	0	-0.004	1	0	0	0
-75.996	-273.075	-249.649	-23.604	-242.653	-181.751	0	-241.155	-48.283	0	-0.147	0	1	0	0
-121.502	-182.962	40.325	-60.698	-78.174	-121.125	0	-103.423	-202.358	0	-0.426	0	0	1	0
241.208	-41.151	-5.698	-65.208	-26.906	-47.302	0	36.491	-149.566	0	-1.094	0	0	0	1
31.592	-283.849	-58.902	-8.792	-24.494	-233.898	0	-104.291	-207.434	0	-0.106	0	0	0	0

Шаг 3

```
index: (3, 4)
focus func val: -43.504
focus val: -212.157
```

204.145	141.455	73.32	0	26.652	47.211	0	82.779	2.466	1	-0.593	-0.833	0	0	0
0.479	0.147	0.916	0	0.194	-0.104	1	0.28	-0.236	0	-0.004	0.003	0	0	0
1.377	0.608	0.035	1	1.292	1.213	-0	0.975	1.3	-0	0	-0.006	-0	-0	-0
-43.504	-258.716	-248.824	0	-212.157	-153.114	0	-218.131	-17.593	0	-0.147	-0.136	1	0	0
-37.947	-146.037	42.447	0	0.248	-47.483	0	-44.217	-123.438	0	-0.425	-0.35	0	1	0
330.969	-1.482	-3.418	0	57.342	31.811	0	100.094	-64.783	0	-1.093	-0.376	0	0	1
43.696	-278.5	-58.594	0	-13.135	-223.231	0	-95.714	-196.002	0	-0.105	-0.051	0	0	0

Шаг 4

```
index: (4, 11)
focus func val: -37.998
focus val: -0.35
```

198.68	108.955	42.062	0	0	27.977	0	55.377	0.256	1	-0.612	-0.85	0.126	0	0
0.439	-0.09	0.688	0	0	-0.245	1	0.08	-0.252	0	-0.004	0.003	0.001	0	0
1.112	-0.967	-1.48	1	0	0.281	0	-0.353	1.193	0	-0.001	-0.007	0.006	0	0
0.205	1.219	1.173	-0	1	0.722	-0	1.028	0.083	-0	0.001	0.001	-0.005	-0	-0
-37.998	-146.338	42.157	0	0	-47.662	0	-44.472	-123.459	0	-0.425	-0.35	0.001	1	0
319.211	-71.407	-70.67	0	0	-9.573	0	41.138	-69.538	0	-1.133	-0.413	0.27	0	1
46.389	-262.483	-43.19	0	0	-213.751	0	-82.21	-194.913	0	-0.096	-0.042	-0.062	0	0

Шаг 5

290.937	464.252	−60.291	0	0	143.695	0	163.351	300.004	1	0.421	0	0.123	−2.428	0
0.158	−1.175	1	0	0	−0.598	1	−0.25	−1.167	0	−0.007	0	0.001	0.007	0
1.827	1.789	−2.274	1	0	1.179	0	0.485	3.518	0	0.007	0	0.006	−0.019	0
0.135	0.951	1.25	0	1	0.634	0	0.947	−0.143	0	−0	0	−0.005	0.002	0
108.583	418.173	−120.466	−0	−0	136.197	−0	127.082	352.794	−0	1.215	1	−0.003	−2.858	−0
364.006	101.105	−120.367	0	0	46.614	0	93.564	76.003	0	−0.631	0	0.269	−1.179	1
50.976	−244.816	−48.279	0	0	−207.997	0	−76.841	−180.008	0	−0.045	0	−0.062	−0.121	0

Алгоритм выдал значение целевой функции 50.976, что совпадает с решением задачи по прямой постановке. Значит код написан корректно.

Приложение

```python
import sys

import numpy as np

from labs.funcs import print_matrix, print_matrix_latex

sys.stdout = open("./labs/output.txt", "w", encoding="utf-8")


def make_matrix(A: np.ndarray, b: np.ndarray, c: np.ndarray):
    return np.vstack(
        (
            np.hstack((np.reshape(b, (A.shape[0], 1)), A, np.eye(A.shape[0]))),
            np.hstack(((np.array([0])), c, np.zeros((A.shape[0])))),
        )
    )


def make_dual_matrix(A: np.ndarray, b: np.ndarray, c: np.ndarray):
    return np.vstack(
        (
            np.hstack((np.reshape(c, (A.T.shape[0], 1)), -A.T, np.eye(A.T.shape[0]))),
            np.hstack(((np.array([0])), -b, np.zeros((A.T.shape[0])))),
        )
    )


def simplex(simplex_matrix: np.ndarray):
    while True:
        index_of_element = simplex_matrix[-1, 1:].argmin()

        if simplex_matrix[-1, 1:][index_of_element] >= 0:
            break

        else:
            min_element = np.inf
            min_line = 0
            index_of_element += 1

            for line in range(simplex_matrix.shape[0] - 1):
                if (
                    simplex_matrix[line, index_of_element] > 0
                    and simplex_matrix[line, 0]
                    / simplex_matrix[
                        line,
                        index_of_element,
                    ]
                    < min_element
                ):
                    min_line = line
                    min_element = (
                        simplex_matrix[line, 0]
                        / simplex_matrix[
                            line,
                            index_of_element,
                        ]
                    )

            print(
                f"index: {(min_line, int(index_of_element))}\n"
                + f"focus func val: {simplex_matrix[-1, int(index_of_element)]:.3f}\n"
                + f"focus val: {simplex_matrix[min_line, int(index_of_element)]:.3f}",
            )
            print_matrix(simplex_matrix)

            simplex_matrix[min_line, :] = (
                simplex_matrix[min_line, :]
                / simplex_matrix[
                    min_line,
                    index_of_element,
                ]
            )

            for line in range(simplex_matrix.shape[0]):
                if line == min_line:
                    continue

                simplex_matrix[line, :] = (
                    simplex_matrix[line, :]
                    - simplex_matrix[min_line, :]
                    * simplex_matrix[line, index_of_element]
                )
```

```
    print("result: ")
    print_matrix(simplex_matrix)

    return simplex_matrix[-1, 0], simplex_matrix


def dual_simplex(simplex_matrix: np.ndarray):
    while True:
        index_of_element = simplex_matrix[:-1, 0].argmin()

        if simplex_matrix[:-1, 0][index_of_element] >= 0:
            break

        else:
            min_element = np.inf
            min_column = 0

            for column in range(1, simplex_matrix.shape[1]):
                if simplex_matrix[-1, column] == 0:
                    continue

                if (
                    simplex_matrix[index_of_element, column] < 0
                    and abs(
                        simplex_matrix[-1, column]
                        / simplex_matrix[index_of_element, column]
                    )
                    < min_element
                ):
                    min_column = column
                    min_element = abs(
                        simplex_matrix[-1, column]
                        / simplex_matrix[index_of_element, column]
                    )

            print(
                f"index: {(int(index_of_element), min_column)}\n"
                + f"focus func val: {simplex_matrix[:-1, 0][index_of_element]:.3f}\n"
                + f"focus val: {simplex_matrix[int(index_of_element), min_column]:.3f}",
            )
            print_matrix(simplex_matrix)

            simplex_matrix[index_of_element, :] /= simplex_matrix[
                index_of_element, min_column
            ]

            for line in range(simplex_matrix.shape[0]):
                if line == index_of_element:
                    continue

                simplex_matrix[line, :] -= (
                    simplex_matrix[index_of_element, :]
                    * simplex_matrix[line, min_column]
                )

    print("result: ")
    print_matrix(simplex_matrix)

    return simplex_matrix[-1, 0], simplex_matrix


A = np.array(
    [
        [15, 115, 106, 290, 232, 167],
        [79, 247, 7, 286, 65, 276],
        [219, 125, 174, 42, 114, 202],
        [287, 213, 225, 274, 169, 260],
        [202, 124, 211, 200, 174, 183],
        [158, 265, 1, 39, 113, 290],
        [175, 196, 170, 270, 187, 178],
        [245, 100, 226, 63, 245, 259],
    ]
)

b = np.array([296, 85, 22, 47, 247, 28, 125, 218])
c = np.array([173, 299, 240, 120, 249, 86])

print("simplex", end="\n\n")

x1 = simplex(make_matrix(A, b, -c))

print("dual simplex", end="\n\n")

x2 = dual_simplex(make_dual_matrix(A, b, -c))

print(f"simplex: {x1[0]}\ndual simplex: {x2[0]}\ndelta: {np.abs(x1[0] - x2[0])}\n")
```