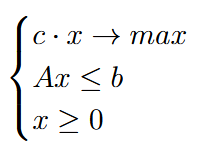
|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ | |
| Федеральное государственное автономное образовательное учреждение высшего образования | |
| **«Дальневосточный федеральный университет»** (ДВФУ) | |
| **ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ** | |
| **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  Салдаев Владимир Сергеевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись)  «\_\_» декабря 2024 г. |
|  | Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (должность, ученое звание)  кандидат физико-математических наук  \_\_\_\_Яковлев Анатолий Александрович  (подпись) (ФИО)  «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2024 г. |
| г. Владивосток  2024 | |

**Прямая задача**

Дана задача:



**Исходные данные:**

A − матрица 8x6

c − неотрицательный 6-мерный вектор

x – неотрицательный 6-мерный вектор неизвестных, который необходимо найти,

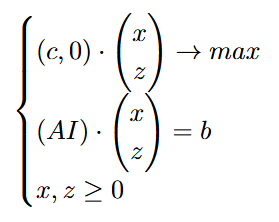
b − неотрицательный 8-мерный вектор.

**Решение:**

Решать будем симплекс-методом. Для начала приведем задачу к

каноническому виду. Введем дополнительный 8-мерный вектор переменных z = Ax – b

Тогда к вектору *c* дописываем 8 нулей и рассматриваем вектор . К матрице A справа дописываем единичную матрицу. Получаем:



Составим симплекс-таблицу. Первая строка – расширенный вектор *c*, где элементы мы запишем со знаком минус, чтобы решать задачу на минимум. Остальные строки – расширенная матрица A, последний столбик – вектор *b*, а первый элемент последнего столбца - значение целевой функции, равное 0.

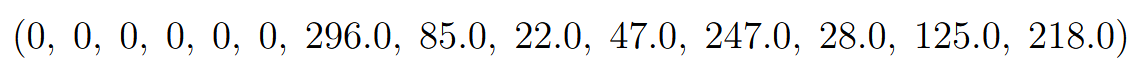
Видим, что в первой строке (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено.

Разрешающая колонка находится путем выборки такого столбца, у которого элемент строки целевой функции отрицательный. Мы будем брать отрицательный элемент, максимальный по модулю.

Разрешающей строкой будет строка, содержащая наименьшее положительное отношение свободного числа к элементу разрешающего столбца.

Элемент, расположенный на пересечении разрешающих столбца и строки, называется разрешающим элементом.

Начальное угловое решение:



Разрешающий столбец = 2

Разрешающая строка = 4

Разрешающий элемент = 303

Преобразовываем строки матрицы, то есть один из базисных столбцов станет **не** базисным, а разрешающий столбец – базисным:

1. Элементы разрешающей строки делим на разрешающий элемент, так как разрешающий элемент = 1, то строка останется прежней.
2. Преобразования остальных строк: Новая строка = Строка - элемент строки в разрешающем столбце \* элемент разрешающей строки.

В первой строке (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено

разрешающий столбец = 4

разрешающая строка = 4

разрешающий элемент = 0.40

Преобразовываем строки матрицы

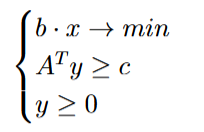
В первой строке (не включая значение целевой функции) НЕТ отрицательных элементов, а значит оптимальное решение найдено

Оптимальное решение:

Целевая функция = 93.64935064935065

**Двойственная задача**

Двойственная задача будет выглядеть так:

****

**Исходные данные:**

AT − матрица 6x8

c − неотрицательный 6-мерный вектор

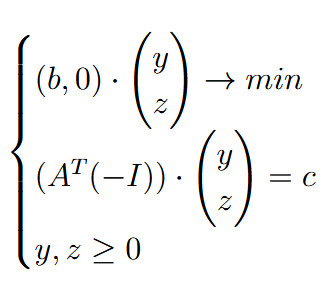
x – неотрицательный 6-мерный вектор неизвестных, который необходимо найти,

b − неотрицательный 8-мерный вектор.

AT

**Решение**

Для начала приведем задачу к каноническому виду. Введем дополнительный вектор переменных z = Ax – b

Тогда к вектору bдописываем 6 нулей и рассматриваем вектор . К матрице справа дописываем единичную матрицу со знаком минус, получаем:

Составим симплекс-таблицу для двойственной задачи. Первая строка представляет из себя расширенный вектор-строка b и значение целевой функции (его изначально указываем равным нулю). Ниже расположены -AT, единичная матрица 6x6 и вектор-столбец *-c*.

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

Разрешающая строка находится путем выборки такой строки, у которой элемент столбца свободных чисел отрицательный. Мы будем брать отрицательный элемент, максимальный по модулю.

Разрешающим столбцом будет столбец содержащий наименьшее по модулю отношение элемента строки целевой функции к элементу числа разрешающей строки.

Элемент, расположенный на пересечении разрешающих столбца и строки, называется разрешающим элементом.

Преобразовываем строки матрицы, то есть один из базисных столбцов станет **не** базисным, а разрешающий столбец – базисным:

1. Элементы разрешающей строки делим на разрешающий элемент, так как разрешающий элемент = 1, то строка останется прежней.
2. Преобразования остальных строк: Новая строка = Строка - элемент строки в разрешающем столбце \* элемент разрешающей строки.

В последнем столбце (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено

разрешающий столбец = 3

разрешающая строка = 3

разрешающий элемент = -303.0

Преобразовываем строки матрицы

В последнем столбце (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено

разрешающий столбец = 1

разрешающая строка = 7

разрешающий элемент = -182.90

Преобразовываем строки матрицы

В последнем столбце (не включая значение целевой функции) есть отрицательные элементы, а значит оптимальное решение еще не найдено

разрешающий столбец = 10

разрешающая строка = 5

разрешающий элемент = -0.42

Преобразовываем строки матрицы

В последнем столбце (не включая значение целевой функции) НЕТ отрицательных элементов, а значит оптимальное решение найдено

Оптимальное решение:

Целевая функция = 93.64935064935065

**Приложения**

Код программы на языке Python:

import numpy as np

M, N = 8, 6

def straight(A, b, c):

    table = np.block([

        [-c, np.zeros(M), 0],

        [A, np.eye(M), b]

    ])

    rows, cols = table.shape

    while True:

        r\_col = -1

        min\_val = float('inf')

        for col in range(N):

            if table[0, col] < 0 and table[0, col] < min\_val:

                r\_col = col

                min\_val = table[0, col]

        if r\_col == -1:

            break

        min\_ratio = float('inf')

        r\_row = -1

        for row in range(1, rows):

            ratio = table[row, cols - 1] / table[row, r\_col]

            if ratio > 0 and ratio < min\_ratio:

                min\_ratio = ratio

                r\_row = row

        r\_el = table[r\_row, r\_col]

        table[r\_row] = table[r\_row] / r\_el

        for row in range(rows):

            if row != r\_row:

                table[row] = table[row] - table[row, r\_col] \* table[r\_row]

    solution = np.zeros(N)

    for col in range(N):

        if np.count\_nonzero(table[1:, col]) == 1 and np.any(table[1:, col] == 1):

            r = np.where(table[1:, col] == 1)[0][0] + 1

            solution[col] = table[r, cols - 1]

    function\_val = table[0, cols - 1]

    print(f"прямая задача \n \* оптимальное решение: {solution} \n \* значение функции: {function\_val} \n")

def duality(A, b, c):

    table = np.block([

        [b.reshape(1, -1), np.zeros(N), 0],

        [-A.T, np.eye(N), -c.reshape(-1, 1)]

    ])

    rows, cols = table.shape

    while True:

        min\_val = float('inf')

        r\_row = -1

        for row in range(1, rows):

            if table[row, cols - 1] < min\_val:

                min\_val = table[row, cols - 1]

                r\_row = row

        if r\_row == -1 or min\_val >= 0:

            break

        min\_ratio = float('inf')

        r\_col = -1

        for col in range(cols - 1):

            if table[r\_row, col] < 0:

                ratio = np.abs(table[0, col] / table[r\_row, col])

                if ratio < min\_ratio:

                    min\_ratio = ratio

                    r\_col = col

        r\_el = table[r\_row, r\_col]

        table[r\_row] = table[r\_row] / r\_el

        for row in range(rows):

            if row != r\_row:

                table[row] = table[row] - table[row, r\_col] \* table[r\_row]

    solution = np.zeros(M)

    for col in range(M):

        if np.count\_nonzero(table[1:, col]) == 1 and np.any(table[1:, col] == 1):

            r = np.where(table[1:, col] == 1)[0][0] + 1

            solution[col] = table[r, cols - 1]

    function\_val = -table[0, cols - 1]

    print(f"двойственная задача \n \* оптимальное решение: {solution} \n \* значение функции: {function\_val} \n")

def main():

    A = np.array([

        [204, 104, 62, 16, 276, 198],

        [10, 163, 314, 145, 75, 140],

        [155, 303, 327, 121, 300, 44],

        [145, 89, 165, 268, 243, 123],

        [7, 225, 259, 70, 315, 278],

        [136, 147, 300, 224, 181, 59],

        [237, 106, 206, 71, 177, 267],

        [274, 162, 9, 89, 234, 336]

    ])

    b = np.array([16, 332, 45, 128, 46, 146, 317, 160]).reshape(-1, 1)

    c = np.array([94, 338, 93, 219, 83, 313])

    dt = np.float64

    straight(np.copy(A.astype(dt)), np.copy(b.astype(dt)), np.copy(c.astype(dt)))

    duality(np.copy(A.astype(dt)), np.copy(b.astype(dt)), np.copy(c.astype(dt)))

if \_\_name\_\_ == '\_\_main\_\_':

    main()