Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы оптимизации и управления»

**ОТЧЁТ**

к лабораторной работе

на тему:

**«начальная фаза симплекс-метода»**

|  |  |
| --- | --- |
|  | Выполнил студент группы 053505  Слуцкий Никита Сергеевич |
|  | Проверил ассистент каф.информатики  Туровец Николай Олегович |

Минск 2023

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc127292736)

[1 Выполнение работы 5](#_Toc127292737)

[2 Тестирование программного продукта 7](#_Toc127292738)

[Заключение 8](#_Toc127292739)

[ПРИЛОЖЕНИЕ А Листинг кода 9](#_Toc127292740)

# ВВЕДЕНИЕ

Пусть имеется задача линейного программирования в канонической форме:

Требуется определить совместна ли задача и, в случае положительного ответа, найти какой-нибудь базисный допустимый план (x, B).

# 1 Выполнение работы

Начальная фаза симплекс-метода

Вход: c ∈ R n, A0 ∈ R m×n и b ∈ R m — параметры задачи (1)

Выход: (x, B) — базисный допустимый план задачи (1) или сообщение о том, что задача (1) не имеет допустимых планов.

Шаг 1. Необходимо преобразовать задачу (1) таким образом, чтобы вектор правых частей b был неотрицательным. Для этого умножим на −1 все ограничения задачи, правая часть которых отрицательна. А именно, для каждого индекса i ∈ {1, 2, . . . , m} выполним следующую операцию: если bi < 0, то умножим на −1 компоненту bi и i-ю строку матрицы A;

Шаг 2. Составим вспомогательную задачу линейного программирования

(2)

где вектор коэффициентов при переменных в целевой функции имеет вид cT = (0, 0, . . . , 0 , −1, −1, . . . , −1) ∈ Rn+m, вектор переменных — x = (x1, x2, . . . , xn, xn+1, xn+2, . . . , xn+m)⊺ ∈ Rn+m ( переменные xn+1, xn+2, . . . , xn+m называются искусственными), матрица A получается из матрицы A0 присоединением к ней справа единичной матрицы порядка m.

Шаг 3. Построим начальный базисный допустимый план (x, B ) вспомогательный задачи

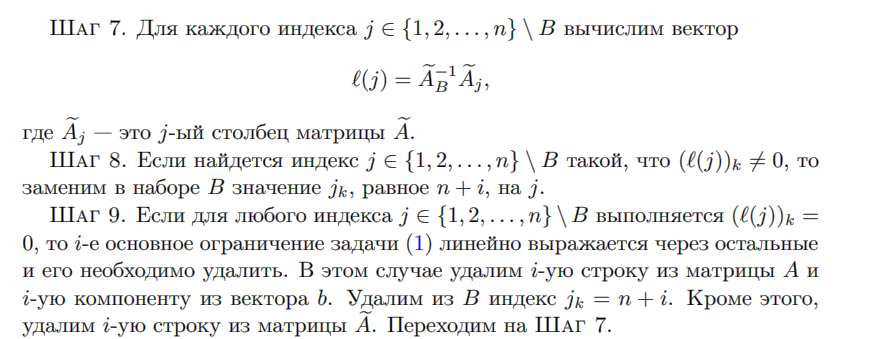
Шаг 4. Решим вспомогательную задачу основной фазой симплекс-метода и получим оптимальный план и соответствующее ему множество базисных индексов B.

Шаг 5. Проверим условия совместности: если xn+1 = xn+2 = . . . = xn+m = 0, то задача (1) совместна; в противном случае, задача (1) не совместна и метод завершает свою работу.

Шаг 6. Формируем допустимый план задачи (1). Для него необходимо подобрать множество базисных индексов. С этой целью скорректируем множество.

Шаг 7. Если B ⊆ {1, 2, . . . , n}, то метод завершает свою работу и возвращает базисный допустимый план (x, B).

Шаг 8. Выберем в наборе B максимальный индекс искусственной переменной



# 2 тестирование программного продукта

Для заданного в условии лабораторной работы примера программный продукт после отработки выдаёт корректный ответ.

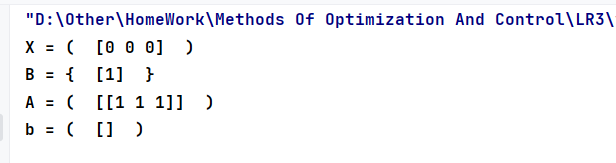


Рисунок 1. Результат вывода программного продукта

# Заключение

В результате выполнения лабораторной работы был реализован алгоритм начальной фазы симплекс-метода в соответствии с шаблоном из методического пособия. Программное средство создано на языке программирования Python с использованием библиотеки для математических вычислений NumPy.

Цели лабораторной работы можно считать достигнутыми. Работа выполнена.

# ПРИЛОЖЕНИЕ А

**Листинг кода**

**def start\_phase\_of\_simplex\_method(matrix\_a: np.array, vector\_b: np.array) -> (np.array, np.array, np.array, np.array):  
 a: np.array = copy.deepcopy(matrix\_a)  
 b: np.array = copy.deepcopy(vector\_b)  
  
 rows\_count: int = matrix\_a.shape[0]  
 columns\_count: int = matrix\_a.shape[1]  
  
 *# 1) remove negative rows (where b[i] < 0)* for row\_index in range(rows\_count):  
 if b[row\_index] < 0:  
 for col\_index in range(columns\_count):  
 a[row\_index][col\_index] \*= -1  
 b[row\_index] \*= -1  
  
 *# 2) get auxiliary task of linear programming c\_auxiliary* c\_auxiliary: np.array = np.array([0] \* columns\_count + [-1] \* rows\_count)  
 identity\_matrix: np.array = np.identity(rows\_count)  
  
 a\_auxiliary: np.array = np.append(a, identity\_matrix, axis=1) *# glue matrices horizontally  
  
 # 3)* x\_auxiliary\_initial\_plan: np.array = np.append(np.array([0] \* columns\_count), b)  
 b\_auxiliary\_initial\_plan: np.array = np.array([columns\_count + counter + 1 for counter in range(rows\_count)])  
  
 *# 4) solve auxiliary task by main(basic) phase of simplex method* auxiliary\_response = basic\_phase\_of\_simplex\_method(a\_auxiliary, c\_auxiliary, x\_auxiliary\_initial\_plan, b\_auxiliary\_initial\_plan)  
 x\_basis\_plan, b\_basis\_indexes = auxiliary\_response  
  
 *# 5) check the conditions of jointness // условия совместности* for index in range(columns\_count, a\_auxiliary.shape[0]):  
 if x\_basis\_plan[index] != 0:  
 raise Exception('The task is not joint and the method completes its work')  
  
 while max(b\_basis\_indexes) > columns\_count + 1:  
 *# 6)* j\_k: int = np.max(b\_basis\_indexes)  
 k: int = b\_basis\_indexes.tolist().index(j\_k) + 1  
 i: int = j\_k - columns\_count  
  
 j\_vector: list = [i for i in range(1, columns\_count + 1) if i <= columns\_count + 1 and i not in b\_basis\_indexes.tolist()]  
  
 a\_base\_matrix: np.array = extract\_submatrix\_by\_column\_numbers(a\_auxiliary, b\_basis\_indexes.tolist())  
 inverse\_matrix: np.array = np.linalg.inv(a\_base\_matrix)  
  
 for j in j\_vector:  
 *# print('Inverse \n', inverse\_matrix)  
 # print('submatrix \n', extract\_submatrix\_by\_column\_numbers(a\_auxiliary, [j]))* l\_j = np.matmul(inverse\_matrix, extract\_submatrix\_by\_column\_numbers(a\_auxiliary, [j]))  
 *# print(l\_j)* if l\_j[k - 1] != 0:  
 b\_basis\_indexes[k - 1] = j - 1  
 else:  
 a\_auxiliary = np.array(a\_auxiliary[:j-1])  
 a = np.array(a[:j-1])  
  
 b\_basis\_indexes = np.delete(b\_basis\_indexes, np.argwhere(b\_basis\_indexes == j\_k)) *# remove by value* b = np.delete(b, np.argwhere(b == b[j-1]))  
  
 break  
  
 return x\_basis\_plan[:columns\_count], b\_basis\_indexes, a, b**