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:

**«Основная 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 6](#_Toc127292738)

[Заключение 7](#_Toc127292739)

[ПРИЛОЖЕНИЕ А Листинг кода 8](#_Toc127292740)

# ВВЕДЕНИЕ

Пусть имеется задача линейного программирования в канонической форме:

Требуется определить ограничен ли сверху целевой функционал задачи на множестве допустимых планов и, в случае положительного ответа, найти оптимальный план задачи. Это можно сделать с помощью основной фазы симплекс-метода.

# 1 Выполнение работы

Шаг 1. Строим базисную матрицу AB и находим её обратную матрицу A−1B ;

Шаг 2. Формируем вектор cB — вектор компонент вектора c, чьи индексы принадлежат множеству B;

Шаг 3. Находим вектор потенциалов u⊺ = c⊺ B\*A−1B ;

Шаг 4. Находим вектор оценок ∆⊺ = u⊺A − c⊺ ;

Шаг 5. Проверим условие оптимальности текущего плана x, а именно, если ∆ ⩾ 0, то текущий x является оптимальным планом задачи (1) и метод завершает свою работу, возвращая в качестве ответа текущий x;

Шаг 6. Находим в векторе оценок ∆ первую отрицательную компоненту и ее индекс сохраним в переменной j0;

Шаг 7. Вычислим вектор z = A −1 B Aj0 , где Aj0 — столбец матрицы A с индексом j0;

Шаг 8. Находим вектор θ⊺ = (θ1, θ2, . . . , θm) ∈ Rm по следующему правилу θi = xji /zi , если zi > 0, и ∞, если zi ⩽ 0, где ji — i-й по счету базисный индекс в упорядоченном наборе B.

Шаг 9. Вычислим θ0 = min i∈{1,2,...,m} θi (2)

Шаг 10. Проверяем условие неограниченности целевого функционала: если θ0 = ∞, то метод завершает свою работу с ответом «целевой функционал задачи не ограничен сверху на множестве допустимых планов»;

Шаг 11. Находим первый индекс k, на котором достигается минимум в (2), и сохраним в переменной j∗ k-й базисный индекс из B;

Шаг 12. В упорядоченном множестве B заменим k-й индекс j∗ на индекс j0.

Шаг 13. Обновим компоненты плана x следующим образом: xj0 := θ0 и для каждого i ∈ {1, 2, . . . , m} такого, что i != k

# 2 тестирование программного продукта

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

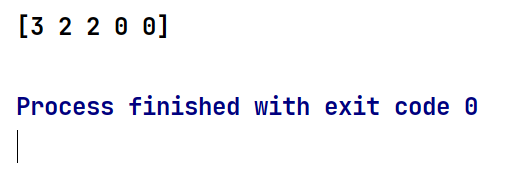


Рисунок 1. Результат вывода программного продукта

# Заключение

В результате выполнения лабораторной работы был реализован алгоритм основной 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 basic\_phase\_of\_simplex\_method(matrix\_a: np.array, vector\_c: np.array, vector\_x: np.array, vector\_b: np.array) -> (np.array, np.array):  
 a: np.array = copy.deepcopy(matrix\_a)  
 c: np.array = copy.deepcopy(vector\_c)  
 x: np.array = copy.deepcopy(vector\_x)  
 b: np.array = copy.deepcopy(vector\_b)  
  
 while True:  
 basis\_matrix: np.array = extract\_submatrix\_by\_column\_numbers(a, b)  
 inverse\_matrix: np.array = np.linalg.inv(basis\_matrix)  
  
 vector\_cb: np.array = np.array([c[index - 1] for index in b])  
  
 potential\_vector: np.array = vector\_cb.dot(inverse\_matrix) *# to return np.array, not np.matrix* grades\_vector: np.array = np.subtract(potential\_vector.dot(a), c)  
  
 j0: int = get\_index\_of\_first\_negative\_item(grades\_vector)  
  
 if j0 == -1:  
 return x, b  
  
 vector\_z: np.array = inverse\_matrix.dot(extract\_submatrix\_by\_column\_numbers(a, [j0 + 1])).flatten()  
  
 vector\_tetta = np.array([x[b[counter] - 1] / item if item > 0 else float('inf') for (counter, item) in enumerate(vector\_z)])  
  
 tetta: float | int = min(vector\_tetta)  
  
 if tetta == float('inf') or math.isinf(tetta):  
 raise Exception('Function is not limited')  
  
 replace\_index: int = vector\_tetta.tolist().index(tetta)  
  
 for (index, value) in enumerate(b):  
 x[value - 1] -= tetta \* vector\_z[index]  
  
 b[replace\_index] = j0 + 1  
  
 x[j0] = tetta**

**def extract\_submatrix\_by\_column\_numbers(source: np.array, column\_numbers: list[int]) -> np.array:  
 # create empty matrix  
 response\_columns\_count: int = len(column\_numbers)  
 response\_rows\_count: int = len(source)  
  
 response: list[list[int]] = [[0] \* response\_columns\_count for \_ in range(response\_rows\_count)]  
  
 # fill it  
 current\_filled\_column: int = 0  
 for col\_number in column\_numbers:  
 for row\_number in range(response\_rows\_count):  
 response[row\_number][current\_filled\_column] = source[row\_number][col\_number - 1]  
  
 current\_filled\_column += 1  
  
 return np.array(response)  
  
  
def get\_index\_of\_first\_negative\_item(array: list) -> int:  
 for counter, item in enumerate(array):  
 if item < 0:  
 return counter  
  
 return -1**