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

Пусть имеется задача линейного программирования в канонической форме:

$$c^T x \to max$$
$$Ax = b$$
$$x \geq 0$$

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

1 ВЫПОЛНЕНИЕ РАБОТЫ

Шаг 1. Строим базисную матрицу A_B и находим её обратную матрицу $A^{-1}{}_B$;

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

Шаг 3. Находим вектор потенциалов $u^T = c^T{}_B * A^{-1}{}_B$;

Шаг 4. Находим вектор оценок $\Delta^T = u^T A - c^T$;

Шаг 5. Проверим условие оптимальности текущего плана x, а именно, если $\Delta \geqslant 0$, 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 Δ первую отрицательную компоненту и ее индекс сохраним в переменной j_0;

Шаг 7. Вычислим вектор z = A −1 B Aj0 , где A_{j0} — столбец матрицы A с индексом j_0;

Шаг 8. Находим вектор $\theta^T = (\theta_1, \theta_2, \ldots, \theta_m) \in R_m$ по следующему правилу $\theta_i = x_{ji} / z_i$, если $z_i > 0$, и ∞, если $z_i \leqslant 0$, где ji — i-й по счету базисный индекс в упорядоченном наборе B.

Шаг 9. Вычислим $\theta_0 = \min_{i \in \{1,2,...,m\}} \theta_i$ (2)

Шаг 10. Проверяем условие неограниченности целевого функционала: если $\theta_0 = \infty$, 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 j_0.

Шаг 13. Обновим компоненты плана x следующим образом: $x_{j0} := \theta_0$ и для каждого $i \in \{1, 2, \ldots, m\}$ такого, что i != k

2 ТЕСТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

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

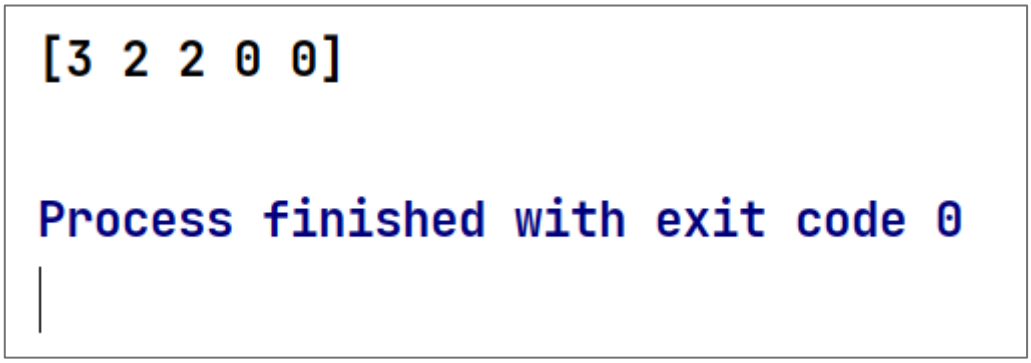

Рисунок 1. Результат вывода программного продукта

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
```