Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

КАФЕДРА ИНФОРМАТИКИ

Отчёт по лабораторной работе No3

По теме “Начальная фаза симплекс-метода”

Выполнила: студентка гр. 053503 Зырянова М.М.

Проверил: ассистент кафедры информатики Туровец Н. О.

Минск 2023

**СОДЕРЖАНИЕ**

[1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ 3](#_Toc134101085)

[2 АЛГОРИТМ ОСНОВНОЙ ФАЗЫ СИМПЛЕКС МЕТОДА 4](#_Toc134101086)

[3 ПРИМЕР РЕАЛИЗАЦИИ ОСНОВНОЙ ФАЗЫ СИМПЛЕКС МЕТОДА 6](#_Toc134101087)

[4 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ 8](#_Toc134101088)

[ПРИЛОЖЕНИЕ А 9](#_Toc134101089)

# **1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Пусть имеется задача линейного программирования в канонической форме

c⊺x → max  
Ax = b (1)

x ⩾ 0,

где c ∈ Rn, x = (x1,x2,...,xn)⊺ ∈ Rn — вектор переменных, A ∈ Rm×n — матрица, в которой m строк и n столбцов, b = (b1, b2, . . . , bm)⊺ ∈ Rm. Требуется определить совместна ли задача (1) и, в случае положительного ответа, найти какой-нибудь базисный допустимый план (x, B).

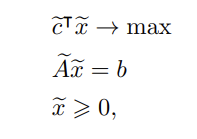
# **2 АЛГОРИТМ ОСНОВНОЙ ФАЗЫ СИМПЛЕКС МЕТОДА**

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

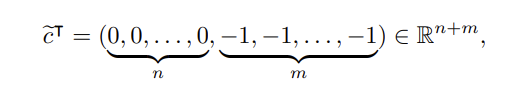
Выход: (x, B) — базисный допустимый план задачи (1) или сообщение о том, что задача (1) не имеет допустимых планов.

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

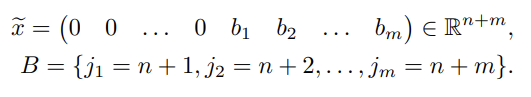


где вектор коэффициентов при переменных в целевой функции имеет вид



вектор переменных — x = (x1, x2, . . . , xn, xn+1, xn+2, . . . , xn+m)⊺ ∈ Rn+m (переменные *xn*+1, *xn*+2, . . . , *xn*+*m* называются искусственными), матрица Ã получается из матрицы A присоединением к ней справа единичной матрицы порядка m.

Шаг 3. Построим начальный базисный допустимый план (x, B e ) задачи (2)



Шаг 4. Решим вспомогательную задачу (2) основной фазой симплекс-метода и получим оптимальный план



и соответствующее ему множество базисных индексов B.

Шаг 5. Проверим условия совместности: если

,

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 *B* следующим образом.

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 *jk* = *n* + *i.*

Шаг 7. Для каждого индекса *j* ∈ {1*,*2*,...,n*} \ *B* вычислим вектор

*,*

где *Ã j* — это *j*-ый столбец матрицы *Ã*.

Шаг 8. Если найдется индекс *j* ∈ {1*,*2*,...,n*} \ *B* такой, что (*ℓ*(*j*))*k* ̸= 0, то заменим в наборе *B* значение *jk*, равное *n* + *i*, на *j*.

Шаг 9. Если для любого индекса *j* ∈ {1*,*2*,...,n*}\*B* выполняется (*ℓ*(*j*))*k* = 0, то *i*-е основное ограничение задачи (1) линейно выражается через остальные и его необходимо удалить. В этом случае удалим *i*-ую строку из матрицы *A* и *i*-ую компоненту из вектора *b*. Удалим из *B* индекс *jk* = *n* + *i*. Кроме этого, удалим *i*-ую строку из матрицы *Ã*. Переходим на Шаг 7.

# **3 ПРИМЕР РЕАЛИЗАЦИИ ОСНОВНОЙ ФАЗЫ СИМПЛЕКС МЕТОДА**

Проиллюстрируем работу метода на примере. Рассмотрим задачу (P) линейного программирования

x1 → max

x1+ x2 + x3 = 0

2x1 + 2x2 + 2x3 = 0

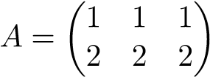
x1 ⩾ 0

x2 ⩾ 0

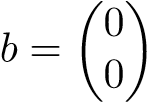
x3 ⩾ 0

Размеры задачи *m* = 2 и *n* = 3. Вектор коэффициентов при переменных в целевом функционале c⊺=(1 0 0).

Матрица коэффициентов при переменных в основных ограничениях

*.*

Вектор правых частей



Так как *b* ⩾ 0, то корректировать вектор *b* не нужно.

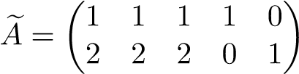
Составляем вспомогательную задачу линейного программирования (2), в которой вектор переменных

*,* 

вектор коэффициентов переменных в целевом функционале

*,*

матрица *Ã*получается из матрицы *A* приписыванием к ней справа единичной матрицы порядка два

*.*

Формируем начальный базисный допустимый план (*x,B*) задачи (2), в котором

*.*

Решаем вспомогательную задачу (2) основной фазой симплекс-метода и получаем оптимальный план:

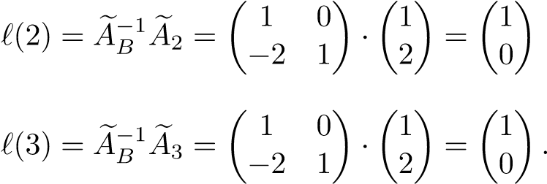


с соответствующим множеством базисных индексов *B* = {*j*1 = 1*,j*2 = 5}. В плане значения искусственных переменных *x*4 и *x*5 равны 0. Следовательно, задача (P) совместна. Значения переменных *x*1*,x*2*,x*3 в *x*e образуют допустимый план задачи (1)

*.*

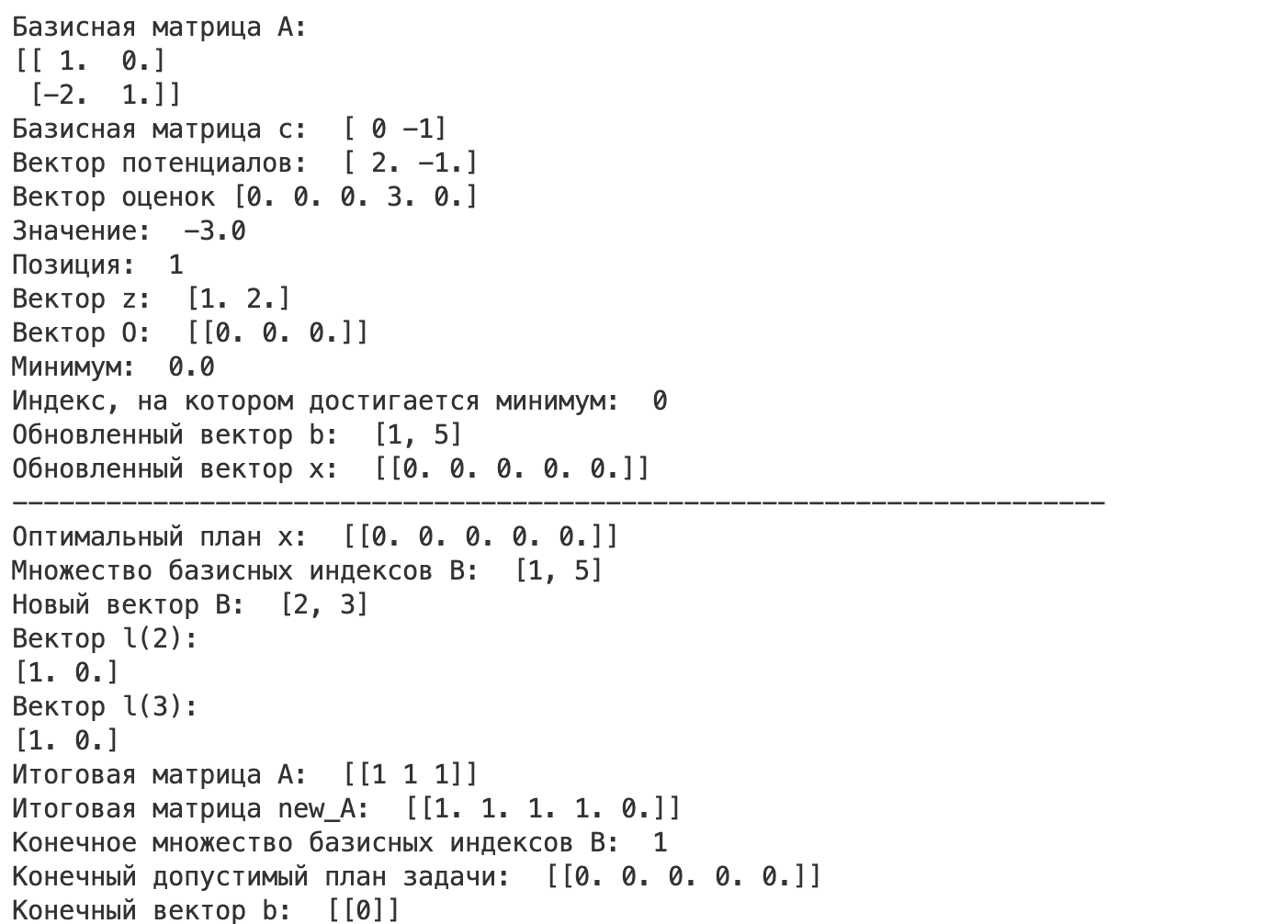
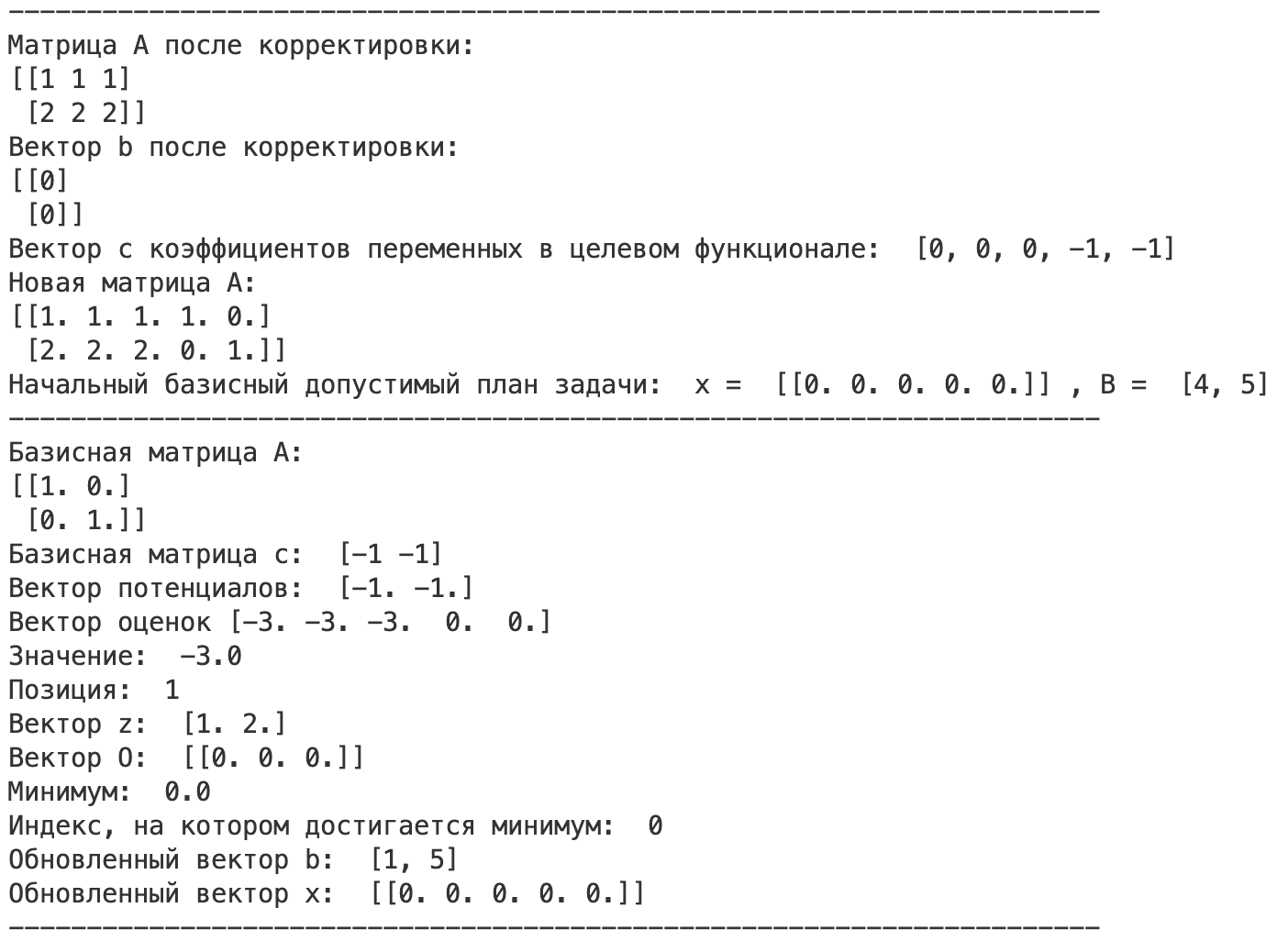
В множестве *B* базисных индексов выбираем максимальный индекс искусственной переменной *j*2 = 5 = 3+2*.*

Таким образом, *k* = 2 и *i* = 2. Для каждого индекса *j* ∈ {1*,*2*,*3} \ *B* = {2*,*3} найдем вектор *ℓ*(*j*)

*.*

Заметим, что для любого индекса *j* ∈ {1*,*2*,*3}\*B* компонента (*ℓ*(*j*))*k* = 0. Стало быть, *i*-ое ограничение задачи (P) линейно выражается через остальные и его необходимо удалить. Из матриц *A* и *Ã*исключим вторую строку, а из вектора *b* вторую компоненту. Из множества *B* удалим выбранный индекс искусственной переменной *j*2. Получаем множество *B* = {*j*1 = 1}. Поскольку в *B* нет индексов искусственных переменных, то метод завершает свою работу. Ответ: *x*⊺ = (0*,*0*,*0), *B* = {*j*1 = 1}, *A* = (1*,*1*,*1) и *b* = (1).

# **4 ДЕМОНСТРАЦИЯ РАБОТЫ ПРОГРАММЫ**



# **ПРИЛОЖЕНИЕ А**

(Листинг программы)

import numpy as np

from lab\_2 import (

symplex,

basis\_matrix

)

def correct\_vector(b: list, A: list):

for i in range(0, len(b)):

if b[i] < 0:

A[i] = 0

b[i] = 0

return b, A

def create\_vector(m: int, n: int):

new\_vector = []

for i in range(0, n):

new\_vector.append(0)

for i in range(0, m):

new\_vector.append(-1)

return new\_vector

def create\_matrix(m: int):

new\_vector = np.zeros((m, m))

for i in range(0, m, 1):

new\_vector[i][i] = 1

return new\_vector

def create\_x(n: int, b: list):

x = np.zeros((1, n))

b = np.transpose(b)

return np.hstack([x, b])

def create\_B(n: int, m: int):

B = []

for i in range(1, m+1):

B.append(n+i)

return B

def find\_max\_B(B: list):

\_max = -1e20

\_pos: int

for i in range(len(B)):

if B[i] > \_max:

\_max = B[i]

\_pos = i

return \_max, \_pos

def find\_l(basisA, new\_A: list, B: list, n: int):

l = []

for j in B:

l.append(np.matmul(basisA, new\_A[:, j-1]))

return l

def delete\_intersections(j: list , B: list):

intersections = []

new\_B = []

for i in j:

for b in B:

if i == b:

intersections.append(i)

for i in j:

for item in intersections:

if i != item:

new\_B.append(i)

return new\_B

if \_\_name\_\_ == "\_\_main\_\_":

A = np.array([

[1, 1, 1],

[2, 2, 2]

])

b = np.array([

[0],

[0]

])

c = np.array([1, 0, 0])

m = 2

n = 3

b, A = correct\_vector(b, A)

c\_transported = create\_vector(m, n)

new\_A = np.hstack([A, create\_matrix(m)])

x = create\_x(n, b)

B = create\_B(n, m)

print('-'\*70)

print('Матрица А после корректировки: ', A, sep="\n")

print('Вектор b после корректировки: ', b, sep="\n")

print('Вектор c коэффициентов переменных в целевом функционале: ', c\_transported)

print('Новая матрица А: ', new\_A, sep="\n")

print('Начальный базисный допустимый план задачи: ', 'x = ', x, ', B = ', B)

basisA, x, B = symplex(new\_A, B, c\_transported, x)

\_max, \_pos = find\_max\_B(B)

i = n - \_pos

j = [x for x in range(1, n+1)]

new\_B = delete\_intersections(j, B)

l = find\_l(basisA, new\_A, new\_B, n)

print('-'\*70)

print('Оптимальный план x: ', x)

print('Множество базисных индексов B: ', B)

print('Новый вектор B: ', new\_B)

k = 0

flag = False

for j in new\_B:

print(f'Вектор l({j}): ', l[k], sep="\n")

if l[k][i-1] == 0:

flag = True

k += 1

if flag == True:

A = np.delete(A, (i-1), axis=0)

new\_A = np.delete(new\_A, (i-1), axis=0)

B = B.pop(0)

b = np.delete(b, (i-1), axis=0)

print('Итоговая матрица А: ' , A)

print('Итоговая матрица new\_A: ', new\_A)

print('Конечное множество базисных индексов В: ', B)

print('Конечный допустимый план задачи: ', x[0:n])

print('Конечный вектор b: ', b)