**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. Н.Э.БАУМАНА**

**Факультет**: Информатика и системы управления  
**Кафедра**: ИУ8 - Информационная безопасность

**МЕТОДЫ ОПТИМИЗАЦИИ**

**Лабораторная работа №2**  
«Двойственность в линейном программировании»

**Вариант** 3

**Преподаватель:**   
Коннова Н.С.

**Студент:**   
Волков Г.А.

**Группа:**   
ИУ8-35

Москва, 2019

**СОДЕРЖАНИЕ**

[**ОСНОВНАЯ ЧАСТЬ** 3](#_Toc21702608)

[Цель работы 3](#_Toc21702609)

[Постановка задачи 3](#_Toc21702610)

[Ход работы 3](#_Toc21702611)

[**ЗАКЛЮЧЕНИЕ** 7](#_Toc21702612)

[**ПРИЛОЖЕНИЕ** 8](#_Toc21702613)

# **ОСНОВНАЯ ЧАСТЬ**

## Цель работы

Научиться по прямой задаче ЛП формулировать и решать соответствующую двойственную задачу.

## Постановка задачи

Требуется найти решение двойственной задачи к следующей задаче:

Необходимо найти вектор , удовлетворяющий условиям, при котором целевой функции (ЦФ) стремится к максимуму. Здесь:

c – вектор коэффициентов ЦФ,  
A – матрица системы ограничений,  
b– вектор правой части системы ограничений.

По условию:

## Ход работы

Приведём исходные данные задачи к каноническому виду:

Выразим базисные переменные в уравнениях:

Тогда исходная симплекс-таблица будет иметь вид:

Default symplic table

4.00 2.00 1.00 1.00 X4

6.00 1.00 2.00 0.00 X5

2.00 0.00 0.50 1.00 X6

0.00 2.00 8.00 3.00 F

S X1 X2 X3

Поставим двойственную задачу ЛП:

Тогда исходная симплекс-таблица примет вид:

Default symplic table

-2.00 -2.00 -1.00 0.00 Y4

-8.00 -1.00 -2.00 -0.50 Y5

-3.00 -1.00 0.00 -1.00 Y6

0.00 -4.00 -6.00 -2.00 F

S Y1 Y2 Y3

Razr stolb Y1

Razr stroka Y4

Symplic table 1

1.00 -0.50 0.50 -0.00 Y1

-7.00 -0.50 -1.50 -0.50 Y5

-2.00 -0.50 0.50 -1.00 Y6

4.00 -2.00 -4.00 -2.00 F

S Y4 Y2 Y3

Razr stolb Y4

Razr stroka Y6

Symplic table 2

3.00 -1.00 0.00 1.00 Y1

-5.00 -1.00 -2.00 0.50 Y5

4.00 -2.00 -1.00 2.00 Y4

12.00 -4.00 -6.00 2.00 F

S Y6 Y2 Y3

Razr stolb Y6

Razr stroka Y5

После нескольких итераций, получим опорное решение, в котором все значения в столбце свободных членов положительны:

8.00 -1.00 2.00 0.50 Y1

5.00 -1.00 2.00 -0.50 Y6

14.00 -2.00 3.00 1.00 Y4

32.00 -4.00 2.00 0.00 F

S Y5 Y2 Y3

Opor answer:

F = 32.000

Y1 = 8.00

Y2 = 0.00

Y3 = 0.00

Продолжим итерации до получения решения:

Razr stolb Y2

Razr stroka Y6

Symplic table 1

3.00 0.00 -1.00 1.00 Y1

2.50 -0.50 0.50 -0.25 Y2

6.50 -0.50 -1.50 1.75 Y4

27.00 -3.00 -1.00 0.50 F

S Y5 Y6 Y3

Razr stolb Y3

Razr stroka Y1

Symplic table 2

3.00 0.00 -1.00 1.00 Y3

3.25 -0.50 0.25 0.25 Y2

1.25 -0.50 0.25 -1.75 Y4

25.50 -3.00 -0.50 -0.50 F

S Y5 Y6 Y1

No more positive in F

Answer:

Fmin = 25.500

Y1 = 0.00

Y2 = 3.25

Y3 = 3.00 Проверим получившееся значение:

Так же неравенства из системы ограничений верны, значит из этого можно сделать вывод, что решение найдено верно и оно является оптимальным.

# **ЗАКЛЮЧЕНИЕ**

В процессе выполнения лабораторной работы мы изучили основные приёмы использования симплекс-метода. Научились ставить и решать обратную задачу линейного программирования с помощью этого метода, а так же узнали, что он является наиболее универсальным из вычислительных методов линейного программирования.

Код, реализующий симплекс-метод, написан на языке Python версии 3.\* и представлен в приложении.

# **ПРИЛОЖЕНИЕ**

**import** copy  
  
  
**def** printWithXInTable(n, tab):  
 **for** i **in** range(n + 1):  
 **for** j **in** range(n + 1):  
 print(**'{:>7.2f}'**.format(tab[i][j]), end=**""**)  
 pam = str(int(tab[i][n + 1]))  
 **if** pam == **"88"**:  
 pam = **" F"  
 elif** pam == **"99"**:  
 pam = **""  
 else**:  
 pam = **" X"** + pam  
 print(pam)  
 **for** i **in** range(n + 2):  
 pam = str(int(tab[n + 1][i]))  
 **if** pam == **"77"**:  
 pam = **" S"  
 elif** pam == **"99"**:  
 pam = **""  
 else**:  
 pam = **" X"** + pam  
 print(**'{:>7s}'**.format(pam), end=**""**)  
 print()  
  
  
**def** printAnswer(n, flag, tab):  
 **if** flag == **"max"**:  
 zn = -1  
 **else**:  
 zn = 1  
 print(**"\n"** + **"Answer:\n"** + **"F"** + str(flag) + **" = "** + **" "** + **'{:<9.3f}'**.format(tab[n][0]\*zn))  
 pam = [0] \* (n + 1)  
 **for** i **in** range(n + 1):  
 pam[i] = 0  
 **for** i **in** range(n):  
 **if** 0 < tab[i][n + 1] <= n:  
 pam[tab[i][n + 1]] = tab[i][0]  
 **for** i **in** range(1, n + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(pam[i]))  
 print()  
  
  
**def** printOpor(n, tab):  
 print(**"\n"** + **"Opor answer:\n"** + **"F"** + **" = "** + **" "** + **'{:<9.3f}'**.format(tab[n][0]))  
 pam = [0] \* (n + 1)  
 **for** i **in** range(n + 1):  
 pam[i] = 0  
 **for** i **in** range(n):  
 **if** 0 < tab[i][n + 1] <= n:  
 pam[tab[i][n + 1]] = tab[i][0]  
 **for** i **in** range(1, n + 1):  
 print(**"X"** + str(i) + **" = "** + **'{:<7.2f}'**.format(pam[i]))  
 print()  
  
**def** createSimplexTable(flag, n, a, b, c):  
 tab = [[0] \* (n + 2) **for** i **in** range(n + 2)]  
 **if** flag == **"max"**:  
 flag = 1  
 **else**:  
 flag = -1  
 **for** i **in** range(n):  
 tab[i][0] = b[i] \* flag  
 **for** j **in** range(1, n + 1):  
 tab[i][j] = a[i][j - 1] \* flag  
 tab[n][i + 1] = flag \* c[i] *# +-* tab[n][0] = 0 *# F(s)* **for** i **in** range(n + 1):  
 tab[i][n + 1] = i + n + 1  
 tab[n + 1][i] = i  
 tab[n + 1][0] = 77 *# S* tab[n][n + 1] = 88 *# F* tab[n + 1][n + 1] = 99 *# empty* **return** tab  
  
  
**def** calculateSimplexTable(n, tab):  
 kol = 0  
 **while True**:  
 kol += 1  
 mi = 10000000  
 k = -1  
 **for** i **in** range(1, n + 1):  
 **if** tab[n][i] > 0 **and** tab[n][i] < mi:  
 mi = tab[n][i]  
 k = i  
 **if** k == -1:  
 print(**'{:>28s}'**.format(**"No more positive in F"**))  
 **break** mi = 10000000  
 r = -1  
 **for** i **in** range(0, n):  
 **if** tab[i][k] != 0:  
 **if** tab[i][0] / tab[i][k] < mi **and** tab[i][0] / tab[i][k] > 0:  
 mi = tab[i][0] / tab[i][k]  
 r = i  
 **if** r == -1:  
 print(**'{:>28s}'**.format(**"No more positive in S/ Xk"**))  
 **break** print(**'{:<28s}'**.format(**"Razr stolb "** + **"X"** + str(tab[n + 1][k])))  
 print(**'{:<28s}'**.format(**"Razr stroka "** + **"X"** + str(tab[r][n + 1])), end=**"\n\n"**)  
 tab2 = copy.deepcopy(tab)  
 tab[r][k] = 1 / tab2[r][k]  
 **for** i **in** range(n + 1):  
 **for** j **in** range(n + 1):  
 **if** j != k:  
 tab[r][j] = tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tab[i][j] = tab2[i][j] - tab2[i][k] \* tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tab[i][k] = -tab2[i][k] / tab2[r][k]  
 tab[n + 1][k], tab[r][n + 1] = tab[r][n + 1], tab[n + 1][k]  
 print(**'{:>28s}'**.format(**"Symplic table "** + str(kol)))  
 printWithXInTable(n, tab)  
 **return** tab  
  
  
**def** findOpor(n, tab):  
 kol = 0  
 **while True**:  
 kol += 1  
 ma = -1  
 k = -1  
 fl = **True  
 for** i **in** range(0, n + 1):  
 **if** tab[i][0] < 0:  
 pam = i  
 fl = **False  
 break  
 if** fl:  
 **break  
 for** i **in** range(1, n + 1):  
 **if** tab[pam][i] < 0:  
 k = i  
 **break  
 if** k == -1:  
 print(**'{:>28s}'**.format(**"Error"**))  
 **break** mi = 10000000  
 r = -1  
 **for** i **in** range(0, n):  
 **if** tab[i][k] != 0:  
 **if** tab[i][0] / tab[i][k] < mi **and** tab[i][0] / tab[i][k] > 0:  
 mi = tab[i][0] / tab[i][k]  
 r = i  
 **if** r == -1:  
 print(**'{:>28s}'**.format(**"No more positive in S/ Xk"**))  
 **break** print(**'{:<28s}'**.format(**"Razr stolb "** + **"X"** + str(tab[n + 1][k])))  
 print(**'{:<28s}'**.format(**"Razr stroka "** + **"X"** + str(tab[r][n + 1])), end=**"\n\n"**)  
 tab2 = copy.deepcopy(tab)  
 tab[r][k] = 1 / tab2[r][k]  
 **for** i **in** range(n + 1):  
 **for** j **in** range(n + 1):  
 **if** j != k:  
 tab[r][j] = tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tab[i][j] = tab2[i][j] - tab2[i][k] \* tab2[r][j] / tab2[r][k]  
 **if** i != r:  
 tab[i][k] = -tab2[i][k] / tab2[r][k]  
 tab[n + 1][k], tab[r][n + 1] = tab[r][n + 1], tab[n + 1][k]  
 print(**'{:>28s}'**.format(**"Symplic table "** + str(kol)))  
 printWithXInTable(n, tab)  
 printOpor(n, tab)  
 **return** tab  
  
  
**def** getA():  
 *# min* a = [[2, 1, 0],  
 [1, 2, 0.5],  
 [1, 0, 1]]  
 **return** a  
  
  
**def** getB():  
 *# min* b = [2, 8, 3]  
 **return** b  
  
  
**def** getC():  
 *# min* c = [4, 6, 2]  
 **return** c  
  
  
**def** main():  
 a = getA()  
 b = getB()  
 c = getC()  
 n = 3  
 flag = **"min"** tab = createSimplexTable(flag, n, a, b, c)  
 print(**'{:>28s}'**.format(**"Default symplic table"**))  
 printWithXInTable(n, tab)  
 tab = findOpor(n, tab)  
 tab = calculateSimplexTable(n, tab)  
 **if** type(tab) == list: *# for Errors* printAnswer(n, flag, tab)  
  
  
main()