**Министерство образования Российской Федерации**

**МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**им. Н.Э. БАУМАНА**

Факультет: Информатика и системы управления Кафедра: Информационная безопасность (ИУ8)

**Методы оптимизации Лабораторная работа №1 на тему:**

«Постановка задачи линейного программирования»

Вариант 22

**Преподаватель:**

Коннова Н.С.

**Студент**:  
Спиридонов О. Ю.

**Группа:**

ИУ8-34

Москва 2025

# Цель работы

Изучение симплекс-метода решения задачи линейного программирования (ЛП).

# Постановка задачи

Требуется найти решение следующей задачи линейного программирования (ЛП):

Здесь x = [x1, x2, …, xn]T – искомый вектор решения;

с = [c1, c2, …, cn] – вектор коэффициентов целевой функции (ЦФ) F;

A = – матрица системы ограничений;

b = [b1, b2, …, bm]T – вектор правой части системы ограничений.

# Ход работы

Решим следующую задачу ЛП:

Используя фиктивные переменные x4, x5, x6 приведем исходную задачу к каноническому виду:

Пусть x4, x5, x6 – базисные переменные, x1, x2 – свободные переменные.

Тогда решение будет:

| b | x1 | x2 | x3

-------------------------------------

x4 | 8.00 | 2.00 | 1.00 | 1.00

x5 | 4.00 | 1.00 | 3.00 | 0.00

x6 | 3.00 | 0.00 | 0.50 | 4.00

F | 0.00 | 4.00 | 3.00 | 8.00

Найден разрешающий элемент: [4, 2, 2]

| b | x1 | x2 | x6

-----------------------------------------------

x4 | 7.25 | 2.00 | 0.88 | -0.25

x5 | 4.00 | 1.00 | 3.00 | 0.00

x3 | 0.75 | 0.00 | 0.12 | 0.25

F | -6.00 | 4.00 | 2.00 | -2.00

Найден разрешающий элемент: [2.0, 0, 0]

| b | x4 | x2 | x6

----------------------------------------------------

x1 | 3.62 | 0.50 | 0.44 | -0.12

x5 | 0.38 | -0.50 | 2.56 | 0.12

x3 | 0.75 | 0.00 | 0.12 | 0.25

F | -20.50 | -2.00 | 0.25 | -1.50

Найден разрешающий элемент: [2.56, 1, 1]

Найдено оптимальное решение:

| b | x4 | x5 | x6

----------------------------------------------------

x1 | 3.56 | 0.59 | -0.17 | -0.15

x2 | 0.15 | -0.20 | 0.39 | 0.05

x3 | 0.73 | 0.02 | -0.05 | 0.24

F | -20.54 | -1.95 | -0.10 | -1.51

Функция максимизирована.

Решение: 20.54

Выполним проверку:

Все неравенства верны, тогда итоговый ответ будет:

x = [3.56, 0.15, 0.73],

F = 20,54

# Вывод

В ходе работы была исследована теоретическая и практическая сторона применения симплекс-метода для решения задач линейного программирования. Этот метод обеспечивает эффективный поиск оптимальных решений задач ЛП, при этом учитываются ограничения, заданные в виде линейных неравенств. Программа работы охватывает ключевые шаги алгоритма симплекс-метода, включая преобразование задачи в каноническую форму, создание симплекс-таблиц и выполнение итераций для выявления оптимального решения. Полученные данные подтверждают, что симплекс-метод является эффективным инструментом для решения задач ЛП.

# Приложение.

Файл “main.py”:

from simplex\_method import execute\_simplex

def main():

minimize = False

c = [4, 3, 8]

A = [[2, 1, 1],

[1, 3, 0],

[0, 0.5, 4]]

b = [8, 4, 3]

f = 0

print("Решение:", execute\_simplex(c, A, b, f, minimize))

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

main()

Файл “simplex\_method.py”:

columns = []

rows = []

def init\_headers(c, A, b):

num\_columns = len(c)

num\_rows = len(b)

global columns

global rows

columns = ['b'] + [f'x{i + 1}' for i in range(num\_columns)]

rows = []

for i in range(num\_rows):

temp\_row = [f'x{i + num\_columns + 1}']

rows.append(temp\_row)

rows.append('F')

def validate\_simplex\_input(c, A, b):

len\_A = len(A[0])

return (all(len(row) == len\_A for row in A) and

len(c) == len\_A and

len(b) == len(A))

def check\_solution\_existence(c, A, b):

if all(x == 0 for x in c):

return False

return all(b[row] >= 0 or min(A[row]) < 0 for row in range(len(b)))

def construct\_simplex\_table(c, A, b, f):

table = []

for i in range(len(A)):

table.append([b[i]] + A[i])

table.append([f] + c)

return table

def format\_header\_value(value):

return " ".join(str(v) for v in value) if isinstance(value, list) else str(value)

def display\_simplex\_table(simplex\_table):

print()

formatted\_columns = [format\_header\_value(col) for col in columns]

formatted\_rows = [format\_header\_value(row) for row in rows]

max\_width = max(len(str(float(j))) for row in simplex\_table for j in row if isinstance(j, (int, float))) + 2

full\_headers = [""] + formatted\_columns # Добавляем 'b' перед заголовками столбцов

print(" | ".join(f"{header:>{max\_width}}" for header in full\_headers))

print("-" \* (max\_width \* len(full\_headers) + 3 \* (len(full\_headers) - 1)))

for i, row in enumerate(simplex\_table):

row\_values = [formatted\_rows[i]] + list(row) # Добавляем заголовок для строки 'b'

print(" | ".join(

f"{float(j):>{max\_width}.2f}" if isinstance(j, (int, float)) else str(j).ljust(max\_width) for j in

row\_values))

def locate\_resolving\_element(c, A, b):

if not check\_solution\_existence(c, A, b):

return ["not\_er"]

for row in range(len(b)):

if b[row] < 0:

for col in range(len(A[0])):

if A[row][col] < 0:

try:

return calculate\_min\_ratio(A, b, col)

except:

return ["inf\_er"]

if max(c) < 0:

return ["not\_er"]

c\_max\_index = c.index(max(c))

return calculate\_min\_ratio(A, b, c\_max\_index)

def calculate\_min\_ratio(A, b, ratio\_col):

min\_ratio = float("inf")

min\_ratio\_row = -1

for row in range(len(A)):

if A[row][ratio\_col] == 0:

continue

ratio = b[row] / A[row][ratio\_col]

if 0 < ratio < min\_ratio:

min\_ratio = ratio

min\_ratio\_row = row

if min\_ratio\_row == -1:

raise ValueError("No valid resolving element found.")

return [A[min\_ratio\_row][ratio\_col], min\_ratio\_row, ratio\_col]

def perform\_simplex\_iteration(c, A, b, f, res\_el):

global columns, rows

new\_resolving\_element = 1 / res\_el[0]

new\_b = [0] \* len(b)

new\_A = [[0] \* len(A[0]) for \_ in A]

for i in range(len(A)):

new\_A[i][res\_el[2]] = (

new\_resolving\_element if i == res\_el[1]

else A[i][res\_el[2]] / res\_el[0] \* -1

)

new\_c = [0] \* len(c)

new\_c[res\_el[2]] = c[res\_el[2]] / res\_el[0] \* -1

for i in range(len(A[0])):

if i != res\_el[2]:

new\_A[res\_el[1]][i] = A[res\_el[1]][i] / res\_el[0]

new\_b[res\_el[1]] = b[res\_el[1]] / res\_el[0]

for i in range(len(c)):

if i == res\_el[2]:

continue

new\_c[i] = c[i] - (A[res\_el[1]][i] \* c[res\_el[2]]) / (res\_el[0])

for i in range(len(b)):

if i == res\_el[1]:

continue

new\_b[i] = b[i] - ((A[i][res\_el[2]] \* b[res\_el[1]]) / res\_el[0])

for i in range(len(A)):

for j in range(len(A[0])):

if (i == res\_el[1]) or (j == res\_el[2]):

continue

new\_A[i][j] = A[i][j] - ((A[i][res\_el[2]] \* A[res\_el[1]][j]) / res\_el[0])

new\_f = f - ((c[res\_el[2]] \* b[res\_el[1]]) / res\_el[0])

temp\_row = rows[res\_el[1]]

rows[res\_el[1]] = columns[1+res\_el[2]]

columns[1+res\_el[2]] = temp\_row

return new\_c, new\_A, new\_b, new\_f

def execute\_simplex(c, A, b, f, minimize):

if validate\_simplex\_input(c, A, b):

print("Input Validation: Passed")

init\_headers(c, A, b)

if minimize:

c = [-x for x in c]

while (max(c) > 0) or (min(b) < 0):

simplex\_table = construct\_simplex\_table(c, A, b, f)

display\_simplex\_table(simplex\_table)

resolving\_element = locate\_resolving\_element(c, A, b)

if resolving\_element == ["not"]:

print("No feasible solution exists.")

return 1

if resolving\_element == ["inf"]:

print("Infinite solutions detected.")

return 1

print("Resolving element located:", resolving\_element)

c, A, b, f = perform\_simplex\_iteration(c, A, b, f, resolving\_element)

print("\nOptimal Solution Found:")

simplex\_table = construct\_simplex\_table(c, A, b, f)

display\_simplex\_table(simplex\_table)

else:

print("Input Validation: Failed")

return 1

if minimize:

print("The function is minimized.")

return f

print("The function is maximized.")

return f \* -1