МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Г. ШУХОВА»
(БГТУ им. В.Г. Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №1

по дисциплине: Исследование операций
тема: "Исследование множества опорных планов системы ограничений задачи линейного программирования (задачи ЛП) в канонической форме"

Выполнил: ст. группы ВТ-231
Масленников Даниил

Проверил:
Вирченко Юрий Петрович

Белгород, 2025 г.

Лабораторная работа №1 «Исследование множества опорных планов системы ограничений задачи линейного программирования (задачи ЛП) в канонической форме.»

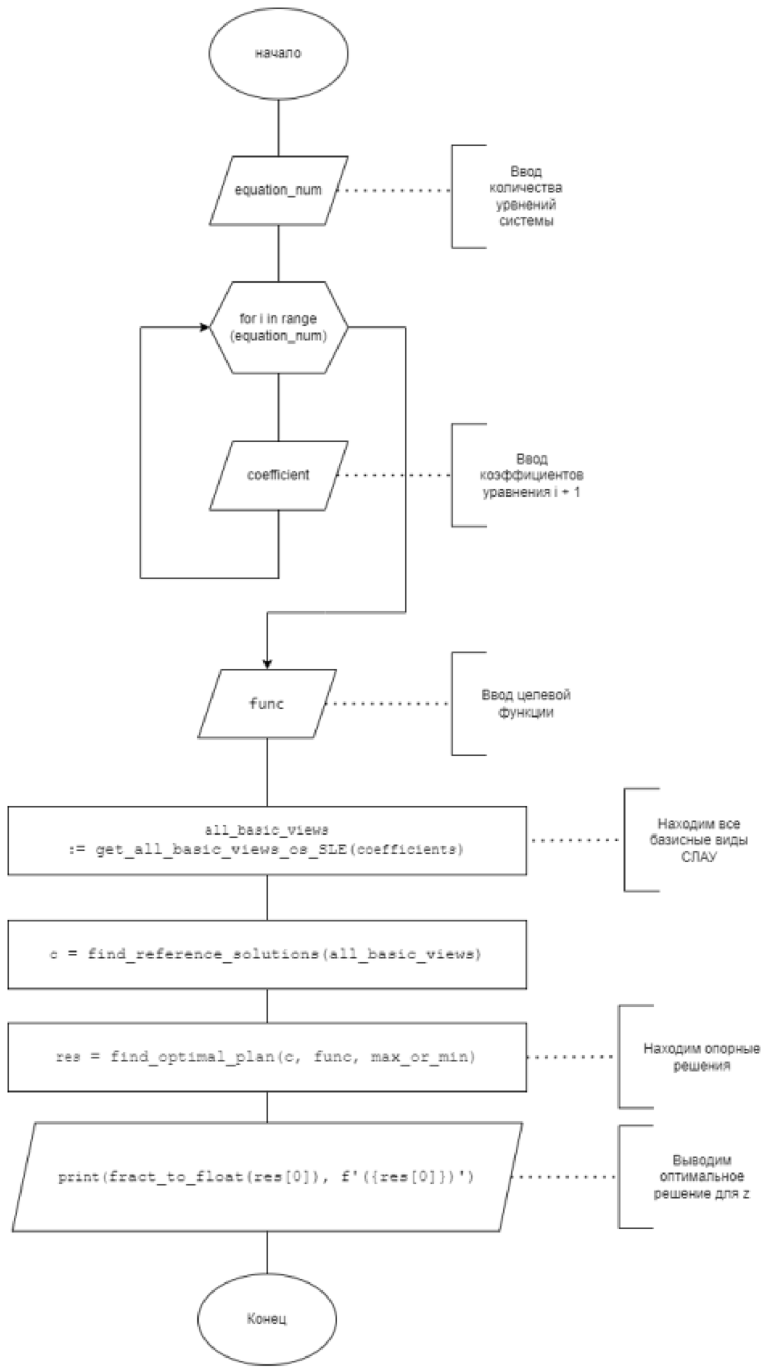
Цель работы: изучить метод Гаусса-Жордана и операцию замещения, а также освоить их применение к отысканию множества допустимых базисных видов системы линейных уравнений, и решению задачи линейного программирования простым перебором опорных решений.

Вариант 14

$$\begin{cases} -x_1 + 5x_2 - 4x_3 - 6x_4 + x_6 = -9 \\ 8x_1 + x_2 - x_3 + 2x_5 + 3x_6 = 8 \\ 4x_1 + 3x_2 - 2x_3 + 9x_4 + x_5 + 7x_6 = 1 \end{cases}$$

Оглавление

Блок-схема программы



Код программы

```
from fractions import Fraction
from itertools import combinations

# Функция для проверки, является ли вектор нулевым
def is_zero_vector(vector: list) -> bool:
    return vector == [0] * len(vector)

# Функция для копирования матрицы
def clone_matrix(matrix: list) -> list:
    return [row[:] for row in matrix]

# Функция для вывода матрицы
def output_matrix(matrix: list):
    for row in matrix:
        print(*row)
    print()

# Функция для вычисления определителя матрицы
def determinant(matrix: list) -> Fraction:
    n = len(matrix)

    # Базовые случаи для матриц 1x1 и 2x2
    if n == 1:
        return matrix[0][0]
    if n == 2:
        return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0]

    det = 0

    # Рекурсивное вычисление определителя для матриц большего размера
    for j in range(n):
        minor = [row[:j] + row[j + 1:] for row in matrix[1:]]
        det += matrix[0][j] * ((-1) ** (1 + j + 1)) * determinant(minor)

    return det

# Функция для получения столбца матрицы по индексу
def get_column(matrix: list, col_index: int) -> list:
    return [row[col_index] for row in matrix]

# Функция для создания матрицы из выбранных столбцов
def create_matrix_from_cols(matrix: list, col_indices: list) -> list:
    return [get_column(matrix, i) for i in col_indices]
```

```python

# Функция для вычисления ранга матрицы
def matrix_rank(matrix: list) -> int:
    rows = len(matrix)
    cols = len(matrix[0]) if matrix else 0
    rank = 0

    # Перебор всех возможных подматриц для вычисления ранга
    for order in range(1, min(rows, cols) + 1):
        for i in range(rows - order + 1):
            for j in range(cols - order + 1):
                sub_matrix = [row[j:j + order] for row in matrix[i:i + order]]
                det = determinant(sub_matrix)
                if det != 0:
                    rank += 1
                    break
            if det != 0:
                break

    return rank

# Функция для приведения матрицы к стандартному виду (без последнего столбца)
def cut_matrix_to_standard(matrix: list) -> list:
    return [row[:-1] for row in clone_matrix(matrix)]

# Функция для выполнения метода Гаусса-Жордана
def Gauss_Jordan_eliminations(matrix: list, basic_var_indices: tuple) -> list:
    if matrix_rank(matrix) != matrix_rank(cut_matrix_to_standard(matrix)):
        return -1

    n = len(matrix)
    for i in range(n):
        if is_zero_vector(matrix[i]):
            continue

        col_num = basic_var_indices[i]
        divisor = matrix[i][col_num]

        if divisor == 0:
            exchange_row = find_exchange_row(matrix, i, col_num)
            if exchange_row is None:
                return -1
            matrix[i], matrix[exchange_row] = matrix[exchange_row], matrix[i]
            divisor = matrix[i][col_num]

        matrix[i] = [Fraction(elem, divisor) for elem in matrix[i]]

```

```python
        for j in range(len(matrix)):
            if is_zero_vector(matrix[j]):
                continue
            if i != j:
                multiplier = matrix[j][col_num]
                matrix[j] = [elem_j - elem_i * multiplier for elem_i, elem_j in zip(matrix[i], matrix

    return matrix

# Функция для поиска строки для обмена в методе Гаусса-Жордана
def find_exchange_row(matrix: list, start_row: int, col_num: int) -> int:
    for row in range(start_row + 1, len(matrix)):
        if matrix[row][col_num] != 0:
            return row
    return None

# Функция для проверки, могут ли переменные быть базисными
def could_vars_be_basic(matrix: list, var_indices: list) -> bool:
    sub_matrix = create_matrix_from_cols(matrix, var_indices)
    det = determinant(sub_matrix)
    return det != 0

# Функция для получения всех наборов базисных переменных
def get_all_sets_of_basic_vars(matrix: list) -> list:
    sub_matrix = cut_matrix_to_standard(matrix)
    amount_of_basic_vars = matrix_rank(sub_matrix)
    all_vars = len(sub_matrix[0])

    set_of_basic_vars = list(combinations(range(all_vars), amount_of_basic_vars))

    # Фильтрация наборов базисных переменных
    for i in set_of_basic_vars:
        if not could_vars_be_basic(clone_matrix(matrix), i):
            del i
    return set_of_basic_vars

# Функция для форматированного вывода переменных
def print_vars(var_indices: list) -> str:
    return '(' + ', '.join(f'x{i + 1}' for i in var_indices) + ')'

# Функция для создания строки линейного уравнения
def make_linear_equation(coefficients: list) -> str:
    equation = ''
    for idx, coeff in enumerate(coefficients[:-1]):
        if coeff:
            if coeff > 0:
                if coeff == 1:
```

```python
                    equation += f'+ x{idx + 1} '
                else:
                    equation += f'+ {coeff}x{idx + 1} '
            else:
                if coeff == -1:
                    equation += f'-x{idx + 1} '
                else:
                    equation += f'{coeff}x{idx + 1} '
    equation += f'= {coefficients[-1]}'
    if equation[0] == '+':
        equation = equation[1:]
    return equation

# Функция для вывода системы линейных уравнений
def output_sle(matrix: list):
    output_string = '{'
    for row in matrix:
        if is_zero_vector(row):
            continue
        output_string += make_linear_equation(row) + ',\n'
    output_string = output_string[:-2] + '}'
    print(output_string)
    print()

# Функция для получения всех базисных видов системы линейных уравнений
def get_all_basic_views_of_SLE(matrix: list) -> list:
    sub_matrix = clone_matrix(matrix)
    set_of_basic_vars = get_all_sets_of_basic_vars(sub_matrix)
    list_of_basic_views = []

    for i in set_of_basic_vars:
        result = Gauss_Jordan_eliminations(clone_matrix(matrix), i)
        if result == -1:
            continue
        print(f'{set_of_basic_vars.index(i) + 1}. Базисные неизвестные:', print_vars(i))
        print('Система:')
        list_of_basic_views.append(result)
        output_sle(result)
    return list_of_basic_views

# Функция для преобразования дроби в float
def fract_to_float(x: Fraction) -> float:
    return float(x.numerator) / float(x.denominator)

# Функция для проверки, что все элементы вектора неотрицательные
def is_all_not_negative(vector: list) -> bool:
    return all(fract_to_float(x) >= 0 for x in vector)
```

```python

# Функция для нахождения опорных решений
def find_reference_solutions(list_of_basic_views: list) -> list:
    list_of_reference_solutions = []

    for matrix in list_of_basic_views:
        solution_vector = get_column(clone_matrix(matrix), -1)
        if not is_all_not_negative(solution_vector):
            continue

        solution_matrix = clone_matrix(matrix)
        for x in range(len(matrix)):
            for y in range(len(matrix[x]) - 1):
                col = get_column(matrix[:], y)
                if not (sum(col) == 1 and col.count(0) == len(col) - 1):
                    solution_matrix[x][y] = 0

        list_of_reference_solutions.append(solution_matrix)
        output_sle(solution_matrix)

    return list_of_reference_solutions

# Функция для вычисления значения целевой функции
def goal(func: list, basic_matrix: list) -> Fraction:
    result = Fraction(0, 1)  # Инициализация дробным нулем
    for i in range(len(basic_matrix)):
        for j in range(len(func)):
            if basic_matrix[i][j] == 1:
                result += func[j] * basic_matrix[i][-1]
    return result

# Функция для нахождения оптимального плана
def find_optimal_plan(list_of_solutions: list, func: list, max_or_min: str) -> tuple:
    min_val = float('inf')
    max_val = -min_val
    res_matrix_min = []
    res_matrix_max = []

    for matrix in list_of_solutions:
        curr_val = goal(func, matrix)
        if curr_val >= max_val:
            res_matrix_max = matrix
            max_val = curr_val
        if curr_val <= min_val:
            res_matrix_min = matrix
            min_val = curr_val

```

```
    if max_or_min == 'min':
        return (min_val, res_matrix_min)
    return (max_val, res_matrix_max)

# Основная часть программы
equation_num = int(input("Количество уравнений в системе: "))
a = [[] for _ in range(equation_num)]

for i in range(equation_num):
    print(f'Коэффициенты уравнения {i + 1}', end='\n')
    a[i].extend(list(map(int, input().split())))

print(f'Целевая функция ({len(a[0]) - 1} чисел)')
func = list(map(int, input().split()))
max_or_min = input('Введите "max", если значение функции стремится к максимуму, иначе "min":\n')

print('Введенная система уравнений:')
output_sle(a)

print('Все базисные виды системы:')
all_basic_views = get_all_basic_views_of_SLE(a)

print('Опорные решения системы:')
reference_solutions = find_reference_solutions(all_basic_views)

optimal_solution = find_optimal_plan(reference_solutions, func, max_or_min)
print(f'Оптимальное решение для z = {func}:')
print(fract_to_float(optimal_solution[0]), f'({optimal_solution[0]})')
output_sle(optimal_solution[1])
```

Результаты работы программы:

```
Количество уравнений в системе: 3
Коэффициенты уравнения 1
-1 5 -4 -6 1 -9
Коэффициенты уравнения 2
8 1 -1 2 3 8
Коэффициенты уравнения 3
4 3 -2 9 1 7 1
Целевая функция (5 чисел)
1 2 3 4 5 6
Введите "max", если значение функции стремится к максимуму, иначе "min":
min
Введенная система уравнений:
{-x1 + 5x2 -4x3 -6x4 + x5 = -9,
 8x1 + x2 -x3 + 2x4 + 3x5 = 8,
 4x1 + 3x2 -2x3 + 9x4 + x5 + 7x6 = 1}

Все базисные виды системы:
1. Базисные неизвестные: (x1, x2, x3)
Система:
{ x1 + 19/21x4 + 2/7x5 = 32/21,
 x2 + 111/7x4 -11/7x5 = 65/7,
 x3 + 443/21x4 -16/7x5 = 283/21}

2. Базисные неизвестные: (x1, x2, x4)
Система:
{ x1 -19/443x3 + 170/443x5 = 419/443,
 x2 -333/443x3 + 65/443x5 = -374/443,
 21/443x3 + x4 -48/443x5 = 283/443}

3. Базисные неизвестные: (x1, x2, x5)
Система:
{ x1 + 1/8x3 + 85/24x4 = 77/24,
 x2 -11/16x3 + 65/48x4 = 1/48,
-7/16x3 -443/48x4 + x5 = -283/48}

4. Базисные неизвестные: (x1, x3, x4)
Система:
{ x1 -19/333x2 + 125/333x5 = 331/333,
-443/333x2 + x3 -65/333x5 = 374/333,
 7/111x2 + x4 -11/111x5 = 65/111}
```

```
5. Базисные неизвестные: (x1, x3, x5)
Система:
{ x1 + 2/11x2 + 125/33x4 = 106/33,
-16/11x2 + x3 -65/33x4 = -1/33,
-7/11x2 -111/11x4 + x5 = -65/11}

6. Базисные неизвестные: (x1, x4, x5)
Система:
{ x1 -34/13x2 + 25/13x3 = 41/13,
 48/65x2 -33/65x3 + x4 = 1/65,
 443/65x2 -333/65x3 + x5 = -374/65}

7. Базисные неизвестные: (x2, x3, x4)
Система:
{-333/19x1 + x2 -125/19x5 = -331/19,
-443/19x1 + x3 -170/19x5 = -419/19,
 21/19x1 + x4 + 6/19x5 = 32/19}

8. Базисные неизвестные: (x2, x3, x5)
Система:
{ 11/2x1 + x2 + 125/6x4 = 53/3,
 8x1 + x3 + 85/3x4 = 77/3,
 7/2x1 + 19/6x4 + x5 = 16/3}

9. Базисные неизвестные: (x2, x4, x5)
Система:
{-13/34x1 + x2 -25/34x3 = -41/34,
 24/85x1 + 3/85x3 + x4 = 77/85,
 443/170x1 -19/170x3 + x5 = 419/170}

10. Базисные неизвестные: (x3, x4, x5)
Система:
{ 13/25x1 -34/25x2 + x3 = 41/25,
 33/125x1 + 6/125x2 + x4 = 106/125,
 333/125x1 -19/125x2 + x5 = 331/125}
```

```
Опорные решения системы:
{ x1 = 32/21,
 x2 = 65/7,
 x3 = 283/21}

{ x1 = 331/333,
 x3 = 374/333,
 x4 = 65/111}

{ x2 = 53/3,
 x3 = 77/3,
 x5 = 16/3}

{ x3 = 41/25,
 x4 = 106/125,
 x5 = 331/125}

Оптимальное решение для z = [1, 2, 3, 4, 5, 6]:
6.7057057057057055 (2233/333)
{ x1 = 331/333,
 x3 = 374/333,
 x4 = 65/111}
```

Аналитическое решение

$$\begin{cases} -x_1 + 5x_2 - 4x_3 - 6x_4 + x_6 = -9 \\ 8x_1 + x_2 - x_3 + 2x_5 + 3x_6 = 8 \\ 4x_1 + 3x_2 - 2x_3 + 9x_4 + x_5 + 7x_6 = 1 \end{cases}$$

- Так как n $>$ m (переменных больше чем уравнений, то система имеет бесконечно много решений)
- Расширенная матрица:

$$\begin{pmatrix} -1 & 5 & -4 & -6 & 0 & 1 & | & -9 \\ 8 & 1 & -1 & 0 & 2 & 3 & | & 8 \\ 4 & 3 & -2 & 9 & 1 & 7 & | & 1 \end{pmatrix}$$

- Приведение матрицы:

1. a_{11} = -1
   - Делим первую строку на -1:

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & | & 9 \\ 8 & 1 & -1 & 0 & 2 & 3 & | & 8 \\ 4 & 3 & -2 & 9 & 1 & 7 & | & 1 \end{pmatrix}$$

   - Умножаем первую строку на 8 и вычитаем из второй:

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & | & 9 \\ 0 & 41 & -33 & -48 & 2 & 11 & | & -64 \\ 4 & 3 & -2 & 9 & 1 & 7 & | & 1 \end{pmatrix}$$

   - Умножаем первую строку на 4 и вычитаем из третьей:

$$\begin{pmatrix} 1 & -5 & 4 & 6 & 0 & -1 & | & 9 \\ 0 & 41 & -33 & -48 & 2 & 11 & | & -64 \\ 0 & 23 & -18 & -15 & 1 & 11 & | & -35 \end{pmatrix}$$

2. $a_{22} = 41$
   - Делим вторую строку на 41:

$$\left(\begin{array}{cccccc|c} 1 & -5 & 4 & 6 & 0 & -1 & 9 \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 23 & -18 & -15 & 1 & 11 & -35 \end{array}\right)$$

   - Умножаем вторую строку на -5 и вычитаем из первой:

$$\left(\begin{array}{cccccc|c} 1 & 0 & -\frac{1}{41} & \frac{6}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 23 & -18 & -15 & 1 & 11 & -35 \end{array}\right)$$

   - Умножаем вторую строку на 23 и вычитаем из третьей:

$$\left(\begin{array}{cccccc|c} 1 & 0 & -\frac{1}{41} & \frac{6}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & \frac{21}{41} & \frac{489}{41} & -\frac{5}{41} & \frac{198}{41} & \frac{37}{41} \end{array}\right)$$

3. $a_{33} = \frac{21}{41}$
   - Делим третью строку на $\frac{21}{41}$:

$$\left(\begin{array}{cccccc|c} 1 & 0 & -\frac{1}{41} & \frac{6}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & -\frac{33}{41} & -\frac{48}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & 1 & \frac{163}{7} & -\frac{5}{21} & \frac{66}{7} & \frac{37}{21} \end{array}\right)$$

   - Умножаем третью строку на $-\frac{33}{41}$ и вычитаем из второй:

$$\left(\begin{array}{cccccc|c} 1 & 0 & -\frac{1}{41} & \frac{6}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & 0 & -\frac{123}{7} & -\frac{1}{7} & \frac{55}{7} & -\frac{1}{7} \\ 0 & 0 & 1 & \frac{163}{7} & -\frac{5}{21} & \frac{66}{7} & \frac{37}{21} \end{array}\right)$$

   - Умножаем третью строку на $-\frac{1}{41}$ и вычитаем из первой:

$$\left(\begin{array}{cccccc|c} 1 & 0 & 0 & \frac{5}{7} & \frac{5}{21} & \frac{4}{7} & \frac{26}{21} \\ 0 & 1 & 0 & -\frac{123}{7} & -\frac{1}{7} & \frac{55}{7} & -\frac{1}{7} \\ 0 & 0 & 1 & \frac{163}{7} & -\frac{5}{21} & \frac{66}{7} & \frac{37}{21} \end{array}\right)$$

- Матрица в упрощенном ступенчатом виде:

$$\begin{pmatrix} 1 & 0 & 0 & \frac{5}{7} & \frac{5}{21} & \frac{4}{7} & | & \frac{26}{21} \\ 0 & 1 & 0 & -\frac{123}{7} & -\frac{1}{7} & \frac{55}{7} & | & -\frac{1}{7} \\ 0 & 0 & 1 & \frac{163}{7} & -\frac{5}{21} & \frac{66}{7} & | & \frac{37}{21} \end{pmatrix}$$

- После всех преобразований матрица имеет вид:

$$\begin{pmatrix} 1 & 0 & 0 & a_{14} & a_{15} & a_{16} & | & b_1 \\ 0 & 1 & 0 & a_{24} & a_{25} & a_{26} & | & b_3 \\ 0 & 0 & 1 & a_{34} & a_{35} & a_{36} & | & b_2 \end{pmatrix}$$

- Базисные переменные: x_1, x_2, x_3. Выражаем их через свободные x_4, x_5, x_6:

$$\begin{cases} x_1 = b_1 - a_{14}x_4 - a_{15}x_5 - a_{16}x_6 \\ x_2 = b_2 - a_{24}x_4 - a_{25}x_5 - a_{26}x_6 \\ x_3 = b_3 - a_{34}x_4 - a_{35}x_5 - a_{36}x_6 \end{cases}$$

- Итоговое решение:

$$\begin{cases} x_1 = \dfrac{26}{21} - \dfrac{5}{7}x_4 - \dfrac{5}{21}x_5 - \dfrac{4}{7}x_6 \\ x_2 = -\dfrac{1}{7} - \dfrac{123}{7}x_4 + \dfrac{1}{7}x_5 - \dfrac{55}{7}x_6 \\ x_3 = \dfrac{37}{21} - \dfrac{163}{7}x_4 + \dfrac{5}{21}x_5 - \dfrac{66}{7}x_6 \end{cases}$$

Найдем один из опорных планов (Всего их $C_6^3 = 20$):

- Для базисных переменных x_1, x_2, x_4
- Расширенная матрица:

$$\begin{pmatrix} -1 & 5 & -6 & -4 & 0 & 1 & | & -9 \\ 8 & 1 & 0 & -1 & 2 & 3 & | & 8 \\ 4 & 3 & 9 & -2 & 1 & 7 & | & 1 \end{pmatrix}$$

1. a_{11} = -1
   - Делим первую строку на -1:

$$\begin{pmatrix} 1 & -5 & 6 & 4 & 0 & -1 & | & 9 \\ 8 & 1 & 0 & -1 & 2 & 3 & | & 8 \\ 4 & 3 & 9 & -2 & 1 & 7 & | & 1 \end{pmatrix}$$

   - Умножаем первую строку на 8 и вычитаем из второй:

$$\begin{pmatrix} 1 & -5 & 6 & 4 & 0 & -1 & | & 9 \\ 0 & 41 & -48 & -33 & 2 & 11 & | & -64 \\ 4 & 3 & 9 & -2 & 1 & 7 & | & 1 \end{pmatrix}$$

   - Умножаем первую строку на 4 и вычитаем из третьей:

$$\begin{pmatrix} 1 & -5 & 6 & 4 & 0 & -1 & | & 9 \\ 0 & 41 & -48 & -33 & 2 & 11 & | & -64 \\ 0 & 23 & -15 & -18 & 1 & 11 & | & -35 \end{pmatrix}$$

2. a_{22} = 41
   - Делим вторую строку на 41:

$$\begin{pmatrix} 1 & -5 & 6 & 4 & 0 & -1 & | & 9 \\ 0 & 1 & -\frac{48}{41} & -\frac{33}{41} & \frac{2}{41} & \frac{11}{41} & | & -\frac{64}{41} \\ 0 & 23 & -15 & -18 & 1 & 11 & | & -35 \end{pmatrix}$$

   - Умножаем вторую строку на -5 и вычитаем из первой:

$$\begin{pmatrix} 1 & 0 & \frac{6}{41} & -\frac{1}{41} & \frac{10}{41} & \frac{14}{41} & | & \frac{49}{41} \\ 0 & 1 & -\frac{48}{41} & -\frac{33}{41} & \frac{2}{41} & \frac{11}{41} & | & -\frac{64}{41} \\ 0 & 23 & -15 & -18 & 1 & 11 & | & -35 \end{pmatrix}$$

   - Умножаем вторую строку на 23 и вычитаем из третьей:

$$\begin{pmatrix} 1 & 0 & \frac{6}{41} & -\frac{1}{41} & \frac{10}{41} & \frac{14}{41} & | & \frac{49}{41} \\ 0 & 1 & -\frac{48}{41} & -\frac{33}{41} & \frac{2}{41} & \frac{11}{41} & | & -\frac{64}{41} \\ 0 & 0 & \frac{489}{41} & \frac{21}{41} & -\frac{5}{41} & \frac{198}{41} & | & \frac{37}{41} \end{pmatrix}$$

3. $a_{33} = \frac{21}{41}$

   – Делим третью строку на $\frac{489}{41}$:

$$\left(\begin{array}{cccccc|c} 1 & 0 & \frac{6}{41} & -\frac{1}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & -\frac{48}{41} & -\frac{33}{41} & \frac{2}{41} & \frac{11}{41} & -\frac{64}{41} \\ 0 & 0 & 1 & \frac{7}{163} & -\frac{5}{489} & \frac{66}{163} & \frac{37}{489} \end{array}\right)$$

   – Умножаем третью строку на $-\frac{48}{41}$ и вычитаем из второй:

$$\left(\begin{array}{cccccc|c} 1 & 0 & \frac{6}{41} & -\frac{1}{41} & \frac{10}{41} & \frac{14}{41} & \frac{49}{41} \\ 0 & 1 & 0 & -\frac{123}{163} & \frac{6}{163} & \frac{121}{163} & \frac{240}{163} \\ 0 & 0 & 1 & \frac{7}{163} & -\frac{5}{489} & \frac{66}{163} & \frac{37}{489} \end{array}\right)$$

   – Умножаем третью строку на $-\frac{6}{41}$ и вычитаем из первой:

$$\left(\begin{array}{cccccc|c} 1 & 0 & 0 & \frac{1020}{6683} & -\frac{173}{6683} & \frac{2678}{6683} & \frac{8061}{6683} \\ 0 & 1 & 0 & -\frac{123}{163} & \frac{6}{163} & \frac{121}{163} & \frac{240}{163} \\ 0 & 0 & 1 & \frac{7}{163} & -\frac{5}{489} & \frac{66}{163} & \frac{37}{489} \end{array}\right)$$

- После всех преобразований матрица имеет вид:

$$\left(\begin{array}{cccccc|c} 1 & 0 & 0 & a_{14} & a_{15} & a_{16} & b_1 \\ 0 & 1 & 0 & a_{24} & a_{25} & a_{26} & b_3 \\ 0 & 0 & 1 & a_{34} & a_{35} & a_{36} & b_2 \end{array}\right)$$

- Базисные переменные: x_1, x_2, x_3. Выражаем их через свободные x_4, x_5, x_6:

$$\begin{cases} x_1 = b_1 - a_{14}x_4 - a_{15}x_5 - a_{16}x_6 \\ x_2 = b_2 - a_{24}x_4 - a_{25}x_5 - a_{26}x_6 \\ x_3 = b_3 - a_{34}x_4 - a_{35}x_5 - a_{36}x_6 \end{cases}$$

- Итоговое решение:

$$\begin{cases} x_1 = \dfrac{8061}{6683} - \dfrac{1020}{6683}x_4 + \dfrac{173}{6683}x_5 - \dfrac{2678}{6683}x_6 \\ x_2 = \dfrac{240}{163} + \dfrac{123}{163}x_4 - \dfrac{6}{163}x_5 - \dfrac{121}{163}x_6 \\ x_3 = \dfrac{37}{489} - \dfrac{7}{163}x_4 + \dfrac{5}{489}x_5 - \dfrac{66}{163}x_6 \end{cases}$$

Вывод: в ходе выполнения лабораторной работы я составил программу для отыскания всех базисных решений системы уравнений с помощью метода Гаусса-Жордана, вывод которой совпал с ответом в моем аналитическом решении.