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 №2

по дисциплине: Исследование операций тема: «***Симплекс-метод в чистом виде***»

Выполнил: ст. группы ВТ-231

Масленников Даниил

Проверил:

Вирченко Юрий Петрович

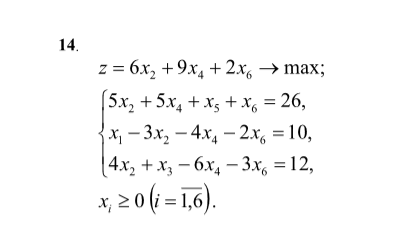
Белгород 2025 г.

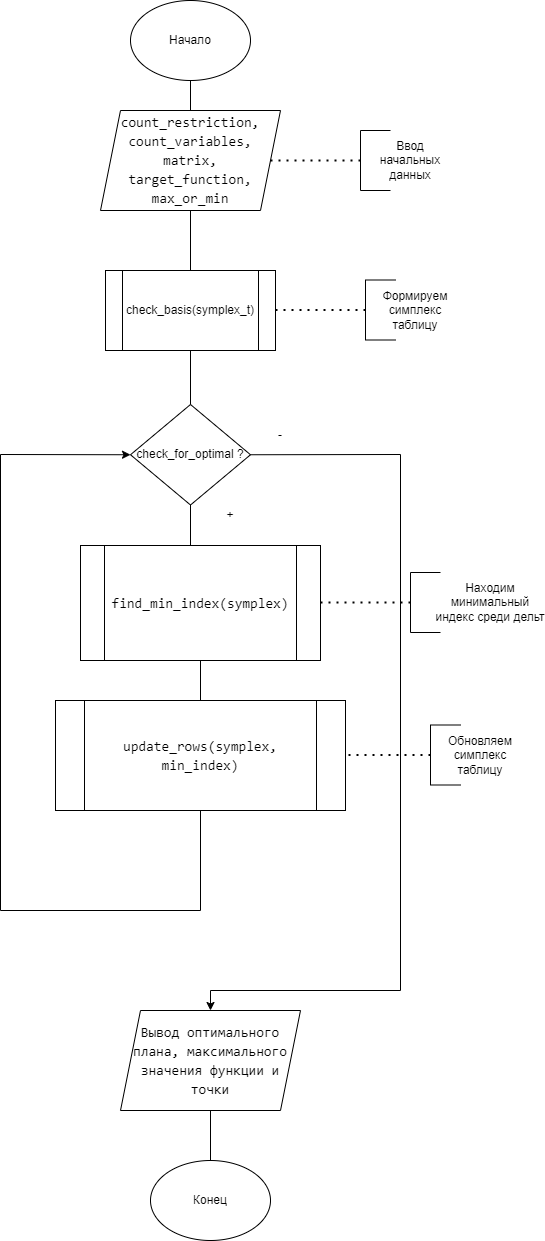
**Цель работы: изучение симплекс-метода для решения задачи линейного программирования с использованием симплекс- таблиц, получение навыков кодирования изученного алгоритма, отладки и тестирования соответствующих программ.**

# Задания для подготовки к работе

1. Выяснить: какой вид должна иметь задача ЛП, чтобы можно было применять симплекс-метод в чистом виде, а также как составляется первая симплекс-таблица?
2. Изучить алгоритм перехода от одной симплекс-таблицы к другой при решении задачи симплекс-методом.
3. Запрограммировать и отладить изученный алгоритм. В рамках подготовки тестовых данных решить вручную одну из следующих ниже задач.

# Вариант – 14

****

**Блок-схема программы**

**Код программы:**

**from** fractions **import** Fraction

**from** prettytable **import** PrettyTable

# Инициализация таблицы для симплекс-метода

symplex = PrettyTable()

# Количество ограничений и переменных

count\_restriction = 3 # Количество ограничений

count\_variables = 6 # Количество переменных

# Матрица коэффициентов ограничений

matrix = [

[0, 5, 0, 5, 1, 1, 26],

[1, -3, 0, -4, 0, -2, 10],

[0, 4, 1, -6, 0, -3, 12]

]

# Целевая функция

target\_function = [0, 6, 0, 9, 0, 2] # Целевая функция

# Заголовки таблицы симплекс-метода

symplex.field\_names = ['Базисные переменные', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'Свободные члены']

# Функция для нахождения дельт

**def** find\_delts(symplex\_t, basis):

finds\_delts = [] # Массив для дельт

# Находим дельты

**for** i in range(count\_variables + 1):

number = 0

**for** j in range(len(basis)):

number += target\_function[basis[j]] \* matrix[j][i]

finds\_delts.append(number)

finds\_delts.insert(0, 'z') # Вставляем "z" в начало

# Создаем новую таблицу и заполняем ее обновленными данными

new\_symplex = PrettyTable()

new\_symplex.field\_names = symplex\_t.field\_names # копируем все строки, кроме последней

**for** row in symplex\_t.\_rows[:-1]:

new\_symplex.add\_row(row)

new\_symplex.add\_row(finds\_delts) # Добавляем дельты (всего 8 элементов)

**return** new\_symplex

# Функция для проверки базиса

**def** check\_basis(symplex\_t):

basis = []

**for** i in range(count\_restriction + 1):

row = ['0'] \* (count\_variables + 2) # Создаем строку с нужным количеством элементов

**if** i != count\_restriction:

flag = **True**

**for** index, element in enumerate(matrix[i]):

**if** flag and element == 1:

flag\_ = **True**

**for** j in range(count\_restriction):

**if** matrix[j][index] != 0 and j != i:

flag\_ = **False**

**if** flag\_:

row[0] = f'x{index + 1}' # Вставляем базисную переменную в начало строки

basis.append(index)

**else**:

flag = **False**

row[index + 1] = element # Копируем значения в соответствующие ячейки таблицы

symplex\_t.add\_row(row)

**return** symplex\_t, basis

# Функция для нахождения индекса минимальной дельты

**def** find\_min\_delta(symplex\_t):

min\_index = -1

min\_delta = float('inf')

delta = symplex\_t.rows[-1][1:-1]

**for** index, element in enumerate(delta):

**if** element < min\_delta:

min\_index = index

min\_delta = element

**return** min\_index, min\_delta

**def** calculate\_Q(symplex\_t, min\_index):

q\_values = []

**for** row in symplex\_t.rows[:-1]: # Итерируем по строкам, кроме последней (где дельты)

# Проверяем, что элемент в разрешающем столбце и свободный член являются числами

**if** isinstance(row[min\_index + 1], (int, float)) and isinstance(row[-1], (int, float)):

**if** row[min\_index + 1] != 0: # Избегаем деления на ноль

q\_value = Fraction(row[-1]) / Fraction(row[min\_index + 1])

q\_values.append(q\_value)

**else**:

q\_values.append(float('inf')) # Если знаменатель равен нулю, Q = бесконечность

**else**:

q\_values.append(float('inf')) # Если элемент не является числом, Q = бесконечность

**return** q\_values

# Функция для нахождения строки с минимальным симплекс-отношением

**def** find\_min\_Q(q\_values):

min\_Q = float('inf')

min\_index = -1

**for** index, q\_value in enumerate(q\_values):

**if** q\_value < min\_Q:

min\_Q = q\_value

min\_index = index

**return** min\_index, min\_Q

# Функция для деления элементов для нахождения базиса

**def** divide(symplex\_t, min\_index, min\_index\_string):

row = [symplex\_t.rows[min\_index\_string][0]] # Сохраняем базисную переменную (первый элемент)

permissive\_element\_in\_row = symplex\_t.rows[min\_index\_string][min\_index] # Элемент, на который будем делить

# Проверка, что элемент для деления является числом

**if** not isinstance(permissive\_element\_in\_row, (int, float)):

print(f"Ошибка: элемент для деления {permissive\_element\_in\_row} не является числом.")

**return** symplex\_t # Возвращаем таблицу без изменений, чтобы избежать дальнейших ошибок

**for** element in symplex\_t.rows[min\_index\_string][1:]:

# Проверка, что элемент является числом перед делением

**if** isinstance(element, (int, float)):

row.append(Fraction(element / permissive\_element\_in\_row)) # Делим только если это число

**else**:

row.append(element) # Если это не число (например, строка 'x1'), добавляем как есть

# Создаем новый симплекс

new\_symplex = PrettyTable()

new\_symplex.field\_names = symplex\_t.field\_names

**for** index, element in enumerate(symplex\_t.\_rows):

row\_ = list(element)

**if** index == min\_index\_string:

row\_ = row # Заменяем строку на новую, обновленную

new\_symplex.add\_row(row\_)

**return** new\_symplex

# Пример выполнения функции

symplex, basis = check\_basis(symplex)

print("Таблица после начальной проверки базиса:")

print(symplex)

# Находим дельты

symplex = find\_delts(symplex, basis)

print("Таблица после нахождения дельт:")

print(symplex)

# Шаг 1: Нахождение минимальной дельты

min\_index, min\_delta = find\_min\_delta(symplex)

print(f"Разрешающий столбец: {min\_index + 1}, Δ{min\_index + 1}: {min\_delta}")

# Шаг 2: Вычисляем симплекс-отношения Q

q\_values = calculate\_Q(symplex, min\_index)

print(f"Q: {q\_values}")

# Шаг 3: Находим строку с минимальным Q

min\_row\_index, min\_Q = find\_min\_Q(q\_values)

print(f"Qmin = {min\_Q}, строка {min\_row\_index + 1}")

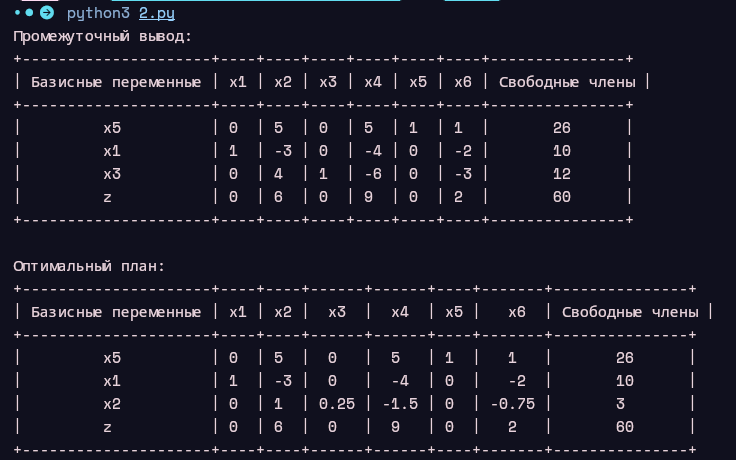
# Шаг 4: Делим строку на разрешающий элемент

symplex = divide(symplex, min\_index, min\_row\_index)

print("Таблица после деления строки на разрешающий элемент:")

print(symplex)

**Результат работы программы**



**Аналитическое решение:**

### Исходная система уравнений:

1. **x₅ = -5x₂ - 5x₄ - x₆ + 26**
2. **x₁ = 3x₂ + 4x₄ + 6x₆ + 10**
3. **x₃ = -4x₂ + 6x₄ + 3x₆ + 12**

### Целевая функция:

**F(X) = 6x₂ + 9x₄ + 2x₆ → min**

### Матрица коэффициентов:

|  | x₁ | x₂ | x₃ | x₄ | x₅ | x₆ |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 5 | 0 | 5 | 1 | 1 |
| 2 | 1 | -3 | 0 | -4 | 0 | -6 |
| 3 | 0 | 4 | 1 | -6 | 0 | -3 |

### Начальный опорный план:

**X₀ = (10, 0, 12, 0, 26, 0)**  
**F(X₀) = 0**

### Итерация №0:

1. **Текущий план неоптимален** (в индексной строке есть отрицательные коэффициенты).
2. **Ведущий столбец:** x₄ (наибольший коэффициент по модулю).
3. **Ведущая строка:** x₅ (min(26/5, —, —) = 26/5).
4. **Разрешающий элемент:** 5.

#### Новая симплекс-таблица:

| Базис | B | x₁ | x₂ | x₃ | x₄ | x₅ | x₆ |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x₄ | 26/5 | 0 | 1 | 0 | 1 | 1/5 | 1/5 |
| x₁ | 154/5 | 1 | 1 | 0 | 0 | 4/5 | -26/5 |
| x₃ | 216/5 | 0 | 10 | 1 | 0 | 6/5 | -9/5 |
| **F(X)** | 234/5 | 0 | 3 | 0 | 0 | 9/5 | -1/5 |

### Итерация №1:

1. **Текущий план неоптимален** (есть отрицательные коэффициенты).
2. **Ведущий столбец:** x₆.
3. **Ведущая строка:** x₄ (min(26, —, —) = 26).
4. **Разрешающий элемент:** 1/5.

#### Новая симплекс-таблица:

| Базис | B | x₁ | x₂ | x₃ | x₄ | x₅ | x₆ |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x₆ | 26 | 0 | 5 | 0 | 5 | 1 | 1 |
| x₁ | 166 | 1 | 27 | 0 | 26 | 6 | 0 |
| x₃ | 90 | 0 | 19 | 1 | 9 | 3 | 0 |
| **F(X)** | 52 | 0 | 4 | 0 | 1 | 2 | 0 |

### Результат:

* **Оптимальный план:**  
  **x₁ = 166**, x₂ = 0, **x₃ = 90**, x₄ = 0, x₅ = 0, **x₆ = 26**.
* **Значение целевой функции:**  
  **F(X) = 6·0 + 9·0 + 2·26 = 52**.

**Вывод: В ходе лабораторной работы освоен симплекс-метод для решения задачи линейного программирования с использованием симплек-стаблиц. Разработана программа для поиска оптимального опорного плана для выбранной целевой функции с использованием этого метода. Вычисленный программой результат совпал с собственными вычислениями. Программа успешно прошла тестовые данные. Реализация симплекс-метода подтвердила его эффективность для аналитического решения поставленных задач**