МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**"МИРЭА - Российский технологический университет"**
**РТУ МИРЭА**

---

**Институт** Информационных Технологий
**Кафедра** Вычислительной Техники

# ПРАКТИЧЕСКАЯ РАБОТА №5

**по дисциплине**
**«Теория принятия решений»**
**Симплексный метод**

Студент группы: ИКБО-15-22 Оганнисян Г.А.
*(Ф.И.О. студента)*

Преподаватель Железняк Л.М.
*(Ф.И.О. преподавателя)*

Москва 2024

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

В данной работе мы рассматриваем применение симплекс-метода для решения задач линейного программирования с произвольным количеством переменных. Симплекс-метод представляет собой эффективный алгоритм, который начинает решение задачи с анализа вершин многогранника условий. Путем последовательного перехода от одной вершины к другой, улучшая значение функции цели, метод стремится к достижению оптимального решения. В случае максимизации функции цели, метод ищет вершину, где значение функции увеличивается, а при минимизации - уменьшается.

Преимущество симплекс-метода заключается в том, что он гарантирует нахождение оптимального решения или устанавливает, что задача неразрешима, за конечное число шагов. Этот метод представляет собой целенаправленный перебор опорных решений ЗЛП в многомерном пространстве переменных.

Таким образом, симплекс-метод является мощным инструментом для решения сложных задач линейного программирования, обеспечивая эффективный и надежный подход к оптимизации целевой функции при линейных ограничениях.

# 1 СИМПЛЕКСНЫЙ МЕТОД

## 1.1 Постановка задачи

Решить прямую ЗЛП с помощью симплексного метода и обратную с помощью теорем двойственности. Определить интервалы устойчивости.

***Задача.*** Фабрика может производить тарелки и кружки. На производство тарелки идет 5 единиц материала, на производство кружки – 20 единиц (керамики). Тарелка требует 10 человеко-часа, кружка – 15. На производство тарелки тратится 0,5 кВатт электроэнергии, кружки – 0,3. Расходы при производстве тарелки равны 1 рубль, а кружки – 2 рубля. Имеется 400 единиц материала и 450 человеко-часов, 25 кВатт энергии и объем накладных расходов равен 300 рублей. Эти данные представлены в таблице 1.1.

*Таблица 1.1.* Исходные данные задачи.

| Ресурс | Товар | | Объем ресурса |
|---|---|---|---|
| | Тарелки | Кружки | |
| Материал | 5 | 20 | 400 |
| Человеко-часы | 10 | 15 | 450 |
| Электроэнергия | 0,5 | 0,3 | 25 |
| Расходы | 1 | 2 | 300 |

Прибыль при производстве тарелки – 20 рублей, при производстве кружки – 50 рублей. Сколько надо сделать тарелок и кружек, чтобы получить максимальную прибыль?

## 1.2 Математическая модель задачи

Пусть x1 – тарелки, x2 – кружки. Максимальный выпуск продукции составит $20x_1 + 50x_2$.

Ограничения задачи:

$$\begin{cases} 5x_1 + 20x_2 \le 400 \\ 10x_1 + 15x_2 \le 450 \\ 0{,}5x_1 + 0{,}3x_2 \le 25 \\ x_1 + 2x_2 \le 300 \\ x_1, x_2 \ge 0 \end{cases}$$

Таким образом, переходим к задаче линейного программирования:

$$f(x) = 20x_1 + 50x_2 \to max$$

$$\begin{cases} 5x_1 + 20x_2 \le 400 \\ 10x_1 + 15x_2 \le 450 \\ 0{,}5x_1 + 0{,}3x_2 \le 25 \\ x_1 + 2x_2 \le 300 \\ x_1, x_2 \ge 0 \end{cases}$$

Приведем задачу к канонической форме. Для этого в левые части ограничений вводим дополнительные переменные: x3 ≥ 0, x4 ≥ 0, x5 ≥ 0, x6 ≥ 0. Эти переменные выбираются так, чтобы они обращали неравенства в равенства.

$$\begin{cases} 5x_1 + 20x_2 + \ x_3 = 400 \\ 10x_1 + 15x_2 + \ x_4 = 450 \\ 0{,}5x_1 + 0{,}3x_2 + \ x_5 = 25 \\ x_1 + 2x_2 + \ x_6 = 300 \\ x_1, x_2 \ge 0 \end{cases}$$

$$f(x) = 20x_1 + 50x_2 + 0x3 \ + \ 0x4 + 0x5 + 0x6$$

Построим начальную симплекс-таблицу. Запишем систему в векторной форме:

$$\mathrm{A1x1} \ + \ \mathrm{A2x2} \ + \mathrm{A3x3} \ + \ \mathrm{A4x4} \ + \ \mathrm{A5x5} \ + \ \mathrm{A6x6} \ = \ \mathrm{A0},$$

$$\mathrm{A1} = \begin{pmatrix} 5 \\ 10 \\ 0{,}5 \\ 1 \end{pmatrix}, \mathrm{A2} = \begin{pmatrix} 20 \\ 15 \\ 0{,}3 \\ 2 \end{pmatrix}, \mathrm{A3} = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \mathrm{A4} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \mathrm{A5} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \mathrm{A6} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix},$$

$$A0 = \begin{pmatrix} 400 \\ 450 \\ 25 \\ 300 \end{pmatrix}$$

Векторы $A3$, $A4$, $A5$, $A6$ являются линейно независимыми единичными векторами 3х-мерного пространства и образуют базис этого пространства.

Поэтому за базисные переменные выбираем переменные $x3$, $x4$, $x5$, $x6$. Небазисными переменными являются $x1$, $x2$. Разложение позволяет найти первое базисное допустимое решение.

Для этого свободные переменные $x1$, $x2$ приравниваем нулю. В результате получим разложение

$$\mathrm{A3x3} + \mathrm{A4x4} + \mathrm{A5x5} + \mathrm{A6x6} = \mathrm{A0},$$

Которому соответствует первоначальный опорный план

$$\mathrm{x}^{(0)} = (\mathrm{x1}, \mathrm{x2}, \mathrm{x3}, \mathrm{x4}, \mathrm{x5}, \mathrm{x6}) = (0{,}0{,}0{,}0{,}20{,}50),$$
$$f\left(x^{(0)}\right) = 0.$$

Для проверки плана $x^{(0)}$ на оптимальность построим первую симплекс-таблицу. Введем в рассмотрение вектор коэффициентов целевой функции при базисных переменных.

$$\overline{C_B} = (c3, c4, c5, c6)^T = (0{,}0{,}0{,}0)^T.$$

В левый столбец Таблицы 1.2 запишем переменные $x3$, $x4$, $x5$, $x6$ образующие базис, в верхней строке – небазисные переменные $x1$, $x2$. В строке $c_j$ запишем коэффициенты целевой функции, соответствующие небазисным переменным с1 = 20, с2 = 50. В столбце $\overline{C_B}$ запишем коэффициенты целевой функции, соответствующие базисным переменным Столбец, определяемый переменной $x1$, состоит из коэффициентов вектора $\overline{A_1}$. Аналогично, столбец, определяемый переменной $x2$, состоит из коэффициентов вектора $\overline{A_2}$. Крайний правый столбец заполняется элементами столбца $\overline{A_0}$, в нем же в результате вычислений получаем оптимальный план.

Заполнение f-строки (Таблица 1.3). Найдем относительные оценки Δ1, Δ2 и значение целевой функции $Q$.

$$\Delta_1 = (\overline{C_B} * \overline{A_1}) - c_1 = 0 * 5 + 0 * 10 + 0 * 0{,}5 + 0 * 1 - 20 = -20;$$

$$\Delta_2 = (\overline{C_B} * \overline{A_2}) - c_2 = 0 * 20 + 0 * 15 + 0 * 0{,}3 + 0 * 2 - 50 = -50;$$

$$\mathrm{Q} = (\overline{C_B} * \overline{A_0}) = 0 * 20 + 0 * 50 = 0.$$

*Таблица 1.2 – Начальная симплекс-таблица задачи*

| | $c_j$ | 20 | 50 | |
|---|---|---|---|---|
| $\overline{C_B}$ | | X1 | X2 | $\overline{A_0}$ |
| 0 | X3 | 5 | 20 | 400 |
| 0 | X4 | 10 | 15 | 450 |
| 0 | X5 | 0,5 | 0,3 | 25 |
| 0 | X6 | 1 | 2 | 300 |
| | f | | | |
| | | $\Delta_1$ | $\Delta_2$ | Q |

*Таблица 1.3 – Заполнение f-строки*

| | $c_j$ | 20 | 50 | | |
|---|---|---|---|---|---|
| $\overline{C_B}$ | | X1 | X2 | $\overline{A_0}$ | |
| 0 | X3 | 5 | 20 | 400 | 400 / 20 = 20 *min* |
| 0 | X4 | 10 | 15 | 450 | 450 / 15 = 30 |
| 0 | X5 | 0,5 | 0,3 | 25 | 25 / 0,3 = 32 |
| 0 | X6 | 1 | 2 | 300 | 300 / 2 = 150 |
| | f | -20 | -50 | 0 | |
| | | $\Delta_1$ | $\Delta_2$ | Q | |

Для оптимальности опорного решения в задаче на максимум требуется выполнение не отрицательности всех относительных оценок Δi ≥ 0. Так как оценки Δ1= −20 и Δ2= −50 в f-строке отрицательны, то это свидетельствуют о возможности улучшения полученного решения. Наибольшая по модулю отрицательная оценка Δ2= −50. В базис будет включена соответствующая ей небазисная переменная $x2$. Составим отношения свободных членов к положительным элементам разрешающего столбца. Данные отношения приведены справа от таблицы. Наименьшему частному соответствует строка с переменной $x5$. Эта переменная исключается из базиса. В Таблице 1.3 разрешающий столбец и разрешающая строка выделены. Разрешающим элементом является число $a23$ = 0.3.

Далее построим новую симплекс-таблицу. Ниже поэтапно демонстрируется процесс заполнения новой симплекс-таблицы (Таблицы 1.4, 1.5).

*Таблица 1.4 – Новая симплекс-таблица*

| | $c_j$ | 20 | 0 | |
|---|---|---|---|---|
| $\overline{C_B}$ | | X1 | X3 | $\overline{A_0}$ |
| 50 | X2 | | 0,05 | |
| 0 | X4 | | | |
| 0 | X5 | | | |
| 0 | X6 | | | |
| | f | | | |
| | | $\Delta_1$ | $\Delta_2$ | Q |

В Таблице 1.4 переменные $x2$ и $x3$ меняются местами вместе с коэффициентами $cj$. Разрешающий элемент заменяется на обратный. В Таблице 1.5 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

*Таблица 1.5 – Симплекс преобразования*

| | $c_j$ | 20 | 0 | |
|---|---|---|---|---|
| $\overline{C_B}$ | | X1 | X3 | $\overline{A_0}$ |
| 50 | X2 | 0.25 | 0.05 | 20 |
| 0 | X4 | | -0.75 | |
| 0 | X5 | | -0.02 | |
| 0 | X6 | | -0.1 | |
| | f | | 2.5 | |
| | | $\Delta_1$ | $\Delta_2$ | Q |

*Таблица 1.6 – Итерация 1*

| | $c_j$ | 20 | 0 | | |
|---|---|---|---|---|---|
| $\overline{C_B}$ | | X1 | X3 | $\overline{A_0}$ | |
| 50 | X2 | 0.25 | 0.05 | 20 | 20 / 0.25 = *80* |
| 0 | X4 | 6.25 | -0.75 | 150 | 150 / 6.25 = 24 *min* |
| 0 | X5 | 0.42 | -0.02 | 19 | 19 / 0.42 = 45.23 |
| 0 | X6 | 0.5 | -0.1 | 260 | 260 / 0.5 = 520 |
| | f | -7.5 | 2.5 | 1000 | |
| | | $\Delta_1$ | $\Delta_2$ | Q | |

Остальные элементы (Таблица 1.6) рассчитываются по «правилу прямоугольника».

$$a_{21} = \frac{(20 * 10) - (5 * 15)}{20} = 6.25; \; a_{31} = \frac{(20 * 0.5) - (5 * 0.3)}{20} = 0.42;$$

$$a_{41} = \frac{(20 * 1) - (5 * 2)}{20} = 0.5; \; \Delta_1 = \frac{(20 * (-20)) - (5 * (-50))}{20} = -7.5;$$

$$a_{23} = \frac{(20 * 450) - (400 * 15)}{20} = 150; \; a_{33} = \frac{(20 * 25) - (400 * 0.3)}{20} = 19;$$

$$a_{43} = \frac{(20 * 300) - (400 * 2)}{20} = 260;$$

Базисное решение, которое дает последняя таблица

$$x^{(1)} = (x1, x2, x3, x4, x5, x6) = (0, 150, 20, 0, 19, 260),$$

$$f\left(x^{(1)}\right) = (\overline{C_B} * \overline{A_0}) = 50 * 20 + 0 * 150 + 0 * 19 + \ 0 * 260 = 1000.$$

Это решение не является оптимальным, так как в f-строке имеется отрицательная оценка Δ1.

В Таблице 1.6 разрешающий столбец и разрешающая строка выделены. Разрешающим элементом является число $a21$ = 6.25.

Поэтому, построим новую симплекс-таблицу. Ниже поэтапно демонстрируется процесс заполнения новой симплекс-таблицы (Таблицы 1.7, 1.8).

*Таблица 1.7 – Новая симплекс-таблица*

| | $c_j$ | 0 | 0 | |
|---|---|---|---|---|
| $\overline{C_B}$ | | X4 | X3 | $\overline{A_0}$ |
| 50 | X1 | | | |
| 20 | X2 | 0.16 | | |
| 0 | X5 | | | |
| 0 | X6 | | | |
| | f | | | |
| | | $\Delta_1$ | $\Delta_2$ | Q |

В Таблице 1.4 переменные $x1$ и $x3$ меняются местами вместе с коэффициентами $cj$. Разрешающий элемент заменяется на обратный. В Таблице 1.5 элементы разрешающей строки делятся на разрешающий элемент. Элементы разрешающего столбца делятся на разрешающий элемент и меняют знак.

*Таблица 1.8 – Симплекс преобразования*

| $\overline{C_B}$ | $c_j$ | 0 | 0 | |
|---|---|---|---|---|
| | | X4 | X3 | $\overline{A_0}$ |
| 50 | X1 | -0.04 | | |
| 20 | X2 | 0.16 | -0.12 | 24 |
| 0 | X5 | -0.07 | | |
| 0 | X6 | -0.08 | | |
| | f | 1.2 | | |
| | | $\Delta_1$ | $\Delta_2$ | Q |

*Таблица 1.9 – Итерация 2*

| $\overline{C_B}$ | $c_j$ | 0 | 0 | |
|---|---|---|---|---|
| | | X4 | X3 | $\overline{A_0}$ |
| 50 | X1 | -0.04 | 0.08 | 14 |
| 20 | X2 | 0.16 | -0.12 | 24 |
| 0 | X5 | -0.07 | 0.04 | 8.8 |
| 0 | X6 | -0.08 | -0.04 | 248 |
| | f | 1.2 | 1.6 | 1180 |
| | | $\Delta_1$ | $\Delta_2$ | Q |

Остальные элементы (Таблица 1.6) рассчитываются по «правилу прямоугольника».

$$\text{a}_{12} = \frac{(6.25 * 0.05) - \big(0.25 * (-0.75)\big)}{6.25} = 0.08;$$

$$\text{a}_{13} = \frac{(6.25 * 20) - (150 * 0.25))}{6.25} = 14;$$

$$\text{a}_{32} = \frac{\big(6.25 * (-0.02)\big) - \big(0.42 * (-0.75)\big)}{6.25} = 0.04;$$

$$\text{a}_{33} = \frac{(6.25 * 19) - (0.42 * 150)}{6.25} = 8.8;$$

$$\text{a}_{42} = \frac{(6.25 * (-0.1)) - \big(0.5 * (-0.75)\big)}{6.25} = -0.04;\ \text{a}_{43} = \frac{(6.25 * 260) - (0.5 * 150)}{6.25} = 248;$$

$$\Delta_2 = \frac{(6.25 * 2.5) - (-7.5 * (-0.75))}{6.25} = 2.5;$$

Если в последней таблице f-строке не содержит отрицательных оценок, то это свидетельствует об оптимальности полученного решения:

Подставляем базисное решение, которое дает последняя таблица

$$\mathrm{x}^{(2)} = (\mathrm{x}1, \mathrm{x}2, \mathrm{x}3, \mathrm{x}4, \mathrm{x}5, \mathrm{x}6) = (14, 24, 0{,}0, 8.8, 248),$$

Где n – количество итераций, n – степень у x.

$f\left(x^{(2)}\right) = (\overline{C_B} * \overline{A_0}) = 14 * 50 + 24 * 20 + 0 * 8.8 + 0 * 248 = 1180.$

Проверим решение по «правилу прямоугольника».

$$f_{max} = Q = (\overline{C_B} * \overline{A_0}) = \frac{(6.25 * 1000) - (-7.5 * 150)}{6.25} = 1180.$$

Таким образом, предприятие должно выпускать в течении недели $x1$ = 24 шт. тарелок и $x2$ = 14 шт. кружек. Тогда предприятие получит программу по максимальному доходу - 1180 [шт.].

# ЗАКЛЮЧЕНИЕ

В ходе работы мы изучили и применили симплекс-метод для решения задач линейного программирования с произвольным количеством переменных. Симплекс-метод представляет собой эффективный алгоритм, основанный на последовательном переходе от одной вершины многогранника условий к другой с целью улучшения значения целевой функции.

Плюсы симплекс-метода:

1. Гарантированное нахождение оптимального решения или определение неразрешимости задачи в конечное число шагов.
2. Эффективность в решении сложных задач оптимизации при линейных ограничениях.
3. Относительная простота реализации и понимания алгоритма.
4. Возможность применения к широкому спектру задач линейного программирования.

Минусы симплекс-метода:

1. Неэффективность в случае большого количества переменных или огромных размеров задачи.
2. Возможность зацикливания в некоторых случаях, что приводит к невозможности нахождения оптимального решения.
3. Не всегда обеспечивает оптимальное решение на практике из-за ограничений математической модели или особенностей конкретной задачи.

Тем не менее, несмотря на некоторые ограничения, симплекс-метод остается мощным инструментом для решения множества задач линейного программирования, обеспечивая надежное и точное определение оптимальных решений.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ


1. Болотова Л. С. Многокритериальная оптимизация. Болотова Л. С., Сорокин А. Б. [Электронный ресурс] / Метод. указания по вып. курсовой работы — М.: МИРЭА, 2015.
2. Сорокин А. Б. Методы оптимизации: гибридные генетические алгоритмы. Сорокин А. Б. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2016.
3. Сорокин А. Б. Линейное программирование: практикум. Сорокин А. Б., Бражникова Е. В., Платонова О. В. [Электронный ресурс] / Учебно-метод. пособие — М.: МИРЭА, 2017.

# ПРИЛОЖЕНИЯ

Приложение А – Код реализации симплексного метода на языке Python.

## Приложение А

Код реализации симплексного метода на языке Python.

*Листинг А.1. Реализация симплексного метода.*

```
import numpy as np


class LinearModel:
    # Инициализация параметров модели
    def __init__(self, A=np.empty([0, 0]), b=np.empty([0, 0]), c=np.empty([0,
0]), minmax="MAX"):
        self.A = A  # Матрица коэффициентов ограничений
        self.b = b  # Вектор правой части ограничений
        self.c = c  # Вектор коэффициентов целевой функции
        self.x = [float(0)] * len(c)  # Начальное решение (все переменные равны
нулю)
        self.minmax = minmax  # Тип оптимизации (минимизация или максимизация)
        self.printIter = True  # Флаг для печати итераций
        self.optimalValue = None  # Оптимальное значение целевой функции
        self.transform = False  # Флаг для преобразования модели

    def addA(self, A):  # Установка матрицы коэффициентов ограничений
        self.A = A

    def addB(self, b):  # Установка вектора правой части ограничений
        self.b = b

    def addC(self, c):  # Установка вектора коэффициентов целевой функции
        self.c = c
        self.transform = False

    def setObj(self, minmax):  # Установка типа оптимизации
        self.minmax = minmax
        self.transform = False

    def setPrintIter(self, printIter):  # Установка флага для печати итераций
        self.printIter = printIter

    def printSoln(self):  # Печать решения и оптимального значения
        print(" Коэффициенты: ")
        print("", self.x)
        print("\n Оптимальное значение: ")
        print("", self.optimalValue)

    def getTableau(self):  # Создание симплекс-таблицы
        num_var = len(self.c)  # Получение количества переменных
        num_slack = len(self.A)  # Получение количества ограничений
        # Создание верхней строки таблицы
        t1 = np.hstack(([None], [0], self.c, [0] * num_slack))
```

*Продолжение листинга А.1. Реализация симплексного метода.*

```
        # Создание базисных переменных и расширение матрицы A, если необходимо
        basis = np.array([0] * num_slack)  # Создание массива для базисных 
переменных
        for i in range(0, len(basis)):
            basis[i] = num_var + i  # Установка индексов базисных переменных
        A = self.A
        if not ((num_slack + num_var) == len(self.A[0])):
            # Если матрица A не квадратная, добавляем единичную матрицу для 
расширения
            B = np.identity(num_slack)
            A = np.hstack((self.A, B))

        # Создание нижних строк таблицы
        t2 = np.hstack((np.transpose([basis]), np.transpose([self.b]), A))

        # Объединение верхней и нижней частей таблицы
        tableau = np.vstack((t1, t2))  # Слияние верхней и нижней частей 
таблицы
        tableau = np.array(tableau, dtype='float')  # Преобразование в массив 
NumPy
        return tableau  # Возвращение симплекс-таблицы

    def optimize(self): # Оптимизация симплекс-методом
        tableau = self.getTableau()  # Получение симплекс-таблицы

        if self.printIter:
            print(" Стартовая таблица:")
            self.print_table(tableau, True)  # Печать начальной симплекс-
таблицы
        optimal = False  # Флаг для проверки на оптимальность
        iter = 0  # Счетчик итераций
        while 1:
            if self.printIter:
                if iter > 0:
                    print("\n====================\n")
                    print(" Итерация :", iter)
                    self.print_table(tableau, False)  # Печать текущей 
симплекс-таблицы
            for profit in tableau[0, 2:]:
                if profit > 0:
                    optimal = False
                    break
                optimal = True
            if optimal:
                break
            n = tableau[0, 2:].tolist().index(np.amax(tableau[0, 2:])) + 2 # 
Выбор разрешающего столбца
            minimum = 99999  # Инициализация минимального значения
            r = -1  # Инициализация разрешающей строки
            for i in range(1, len(tableau)):
                if tableau[i, n] > 0:
                    val = tableau[i, 1] / tableau[i, n]
                    if val != 0 and val < minimum:
                        minimum = val  # Обновление минимального значения
                        r = i  # Обновление разрешающей строки
            pivot = tableau[r, n]  # Получение разрешающего элемента
            print("\n Разрешающий столбец:", n - 1)
            print(" Разрешающая строка:", r)
            print(" Разрешающий элемент: ", pivot)
            tableau[r, 1:] = tableau[r, 1:] / pivot  # Деление строки на 
разрешающий элемент
            for i in range(0, len(tableau)):
```

*Продолжение листинга А.1. Реализация симплексного метода.*

```
                if i != r:
                    mult = tableau[i, n] / tableau[r, n]  # Вычисление
множителя
                    tableau[i, 1:] = tableau[i, 1:] - mult * tableau[r, 1:]  #
Обновление строк
            tableau[r, 0] = n - 2  # Обновление индекса базисной переменной в
таблице
            iter += 1  # Увеличение счетчика итераций
        if self.printIter:
            print("\n----------------------------------\n")
            print(" Финальная таблица была получена за", iter, "итерации")
            self.print_table(tableau, False)  # Печать финальной симплекс-
таблицы
        else:
            print("Решено")
        self.x = np.array([0] * len(self.c), dtype=float)  # Создание массива
для решения
        for key in range(1, (len(tableau))):
            if tableau[key, 0] < len(self.c):
                self.x[int(tableau[key, 0])] = tableau[key, 1]  # Обновление
значений переменных
            self.optimalValue = -1 * tableau[0, 1]  # Установка оптимального
значения


    def print_table(self, tableau, start):  # Функция для печати симплекс-
таблицы
        print("ind A0\t\t ", end="")  # Печать заголовка столбца с индексом и
A0

        for i in range(1, len(self.c) + 1):  # Печать заголовков столбцов
переменных x
            print("x_" + str(i), end="\t ")

        for i in range(1, 5):  # Печать заголовков столбцов правой части
ограничений
            print("b_" + str(i), end="\t ")
        print()  # Переход на новую строку после печати заголовка

        for j in range(0, len(tableau)):  # Перебор строк таблицы
            for i in range(0, len(tableau[0])):  # Перебор элементов в строке
                if not np.isnan(tableau[j, i]):  # Проверка, что элемент не
NaN
                    if i == 0: # Если это первый столбец (индекс базисной
переменной)
                        print('x_' + str(int(tableau[j, i]) + 1), end="\t ")
                    else:
                        if j == 0 and start is False:  # Если это первая
строка и start равно False
                            if round(tableau[j, i], 2) == 0:
                                print(round(tableau[j, i], 2), end="\t ")  #
Если значение округленное до 2 знаков после запятой равно 0
                            else:
                                print((-1) * round(tableau[j, i], 2), end="\t
")  # В противном случае, печать отрицательного значения
              else:
                            print(round(tableau[j, i], 2), end="\t ")  # Если
не первая строка или start равно True
                else:
                    print('F', end="\t ")  # Если элемент NaN, печать символа
'F' вместо значения
            print()  # Переход на новую строку после печати строки таблицы
```

*Продолжение листинга А.1. Реализация симплексного метда*

```
if __name__ == '__main__':
    model1 = LinearModel()
    A = np.array(
        [
            [16, 12],
            [0.2, 0.4],
            [6, 5],
            [3, 4]
        ]
    )
    b = np.array(
        [1200, 30, 600, 300]
    )
    c = np.array(
        [260, 300]
    )
    model1.addA(A)
    model1.addB(b)
    model1.addC(c)
    print("\n Дано:")
    print("> A =\n", A, "\n")
    print("> A0 =\n", b, "\n")
    print("> C =\n", c, "\n\n")
    model1.optimize()
    print("\n")
    model1.printSoln()
```