УДК 519.852.3

Жуков Никита Романович
Студент
Кубанский государственный аграрный университет имени И. Т. Трубилина
Фомичев Дмитрий Павлович
Студент
Кубанский государственный аграрный университет имени И. Т. Трубилина

СИМПЛЕКС-МЕТОД В ЗАДАЧАХ ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ: ТЕОРИЯ И РЕАЛИЗАЦИЯ НА PYTHON

Аннотация: Настоящая статья посвящена изучению симплекс-метода — одного из ключевых алгоритмов линейного программирования, широко применяемого в исследовании операций и методах оптимизации. Цель работы состоит в последовательном изложении теоретических основ метода и демонстрации его практической реализации средствами языка программирования Python с использованием библиотеки SciPy. В статье рассматривается геометрическая интерпретация задачи линейного программирования: область допустимых решений как выпуклый многогранник, а целевая функция как гиперплоскость, сдвигаемая до крайней допустимой вершины. Подробно описаны стандартная и каноническая формы задачи, введение переменных остатка (слабых переменных) и процедура построения симплекс-таблицы. Алгоритм пошагово проиллюстрирован на числовом примере с двумя переменными и двумя ограничениями типа «не более». Показана методика выбора входящей и выходящей переменных по правилу наибольшего коэффициента и минимального отношения. В разделе реализации приведён программный код на Python с пояснениями; сравниваются два метода решения: функция scipy.optimize.linprog и библиотека PuLP. Результаты вычислительного эксперимента подтверждают корректность теоретических выводов. В

заключении сформулированы практические рекомендации по применению метода и намечены направления дальнейшего исследования.

Abstract: This article is devoted to the study of the simplex method, one of the key algorithms in linear programming widely used in operations research and optimization. The aim of the work is to present the theoretical foundations of the method sequentially and to demonstrate its practical implementation using the Python programming language and the SciPy library. The paper examines the geometric interpretation of linear programming: the feasible region as a convex polyhedron and the objective function as a hyperplane shifted to an extreme feasible vertex. The standard and canonical forms, the introduction of slack variables, and the procedure for constructing the simplex tableau are described in detail. The algorithm is illustrated step by step with a numerical example involving two variables and two less-than-or-equal constraints. The pivot selection rules — largest coefficient for the entering variable and minimum ratio for the leaving variable — are explained. The implementation section provides annotated Python code and compares two solution approaches: scipy.optimize.linprog and the PuLP library. The results of a computational experiment confirm the theoretical conclusions. The paper concludes with practical recommendations for applying the method and outlines directions for further research.

Ключевые слова: линейное программирование; симплекс-метод; исследование операций; Python; SciPy; PuLP; оптимизация; симплекс-таблица.
Key words: linear programming; simplex method; operations research; Python; SciPy; PuLP; optimization; simplex tableau.

Введение

Задачи распределения ограниченных ресурсов — сырья, времени, финансов, производственных мощностей — занимают центральное место в управлении организациями различных отраслей. Линейное программирование (ЛП) предоставляет строгий математический аппарат для их формализации и решения: нужно найти максимум (или минимум) линейной целевой функции при

выполнении системы линейных ограничений. С момента своего появления в конце 1940-х годов метод линейного программирования прочно вошёл в арсенал аналитика, экономиста и инженера.

Симплекс-метод, разработанный Джорджем Данцигом в 1947 году, стал первым эффективным алгоритмом решения задач ЛП и по сей день остаётся наиболее востребованным на практике. Несмотря на экспоненциальную сложность в худшем случае, на реальных задачах он ведёт себя как полиномиальный алгоритм и успешно решает системы с тысячами переменных и ограничений. Параллельно с теоретическим изучением важно освоить современные программные инструменты: язык Python с его экосистемой библиотек позволяет применять симплекс-метод без написания алгоритма с нуля, сосредоточившись на постановке задачи и анализе результатов.

Цель исследования

Целью настоящей работы является систематическое изложение теоретических основ симплекс-метода — от геометрической интерпретации до пошаговой работы с симплекс-таблицей — и демонстрация его практической реализации на языке Python с использованием библиотек SciPy и PuLP на конкретном числовом примере.

Материал и методы исследования

Теоретическую основу работы составляют классические труды по исследованию операций: учебники Дж. Данцига, Хэдли Дж., а также отечественные учебные пособия по линейному программированию. Практическая часть выполнена на языке Python 3.11; для решения задачи ЛП использовались:

— библиотека SciPy 1.11 (функция scipy.optimize.linprog, метод HiGHS — современная реализация симплекс- и барьерного методов);

— библиотека PuLP 2.7 — высокоуровневый интерфейс для формулировки задач математического программирования.

Числовой пример — задача максимизации прибыли предприятия, производящего два вида продукции при ограничениях на два ресурса, — выбран как минимально достаточный для иллюстрации всех шагов алгоритма и одновременно допускающий геометрическую интерпретацию на плоскости. Для визуализации области допустимых решений и траектории симплекс-метода применялась библиотека Matplotlib 3.7.

Результаты исследования и их обсуждение

Постановка задачи линейного программирования.

Общая форма задачи максимизации:

$$\text{максимизировать } z = c_1x_1 + c_2x_2 + \ldots + c_nx_n$$

при ограничениях a_{i1}x_1 + a_{i2}x_2 + … + a_{in}$x_n \le b_i$ (i = 1, …, m), $x_j \ge 0$.

Для числового примера примем:

$$\text{максимизировать } z = 3x_1 + 2x_2$$

при ограничениях: $x_1 + x_2 \le 4$; $x_1 + 3x_2 \le 6$; $x_1, x_2 \ge 0$.

Здесь x_1 и x_2 — объёмы выпуска двух видов продукции (в условных единицах), коэффициенты целевой функции — удельная прибыль, правые части — запасы ресурсов.

Геометрическая интерпретация.

Каждое неравенство-ограничение задаёт полуплоскость; их пересечение с условием неотрицательности образует область допустимых решений (ОДР) — выпуклый многоугольник с вершинами (0, 0), (4, 0), (3, 1) и (0, 2). Ключевая теорема линейного программирования утверждает: если задача имеет конечный оптимум, то он достигается хотя бы в одной вершине ОДР. Симплекс-метод эксплуатирует это свойство, перемещаясь по вершинам в направлении возрастания целевой функции.

Каноническая форма и симплекс-таблица.

Для перехода к канонической форме вводятся переменные остатка (слабые переменные) x_3 и x_4, переводящие неравенства в равенства:

$x_1 + x_2 + x_3 = 4$; $x_1 + 3x_2 + x_4 = 6$; $x_1, x_2, x_3, x_4 \geq 0$.

Начальный допустимый базис — $\{x_3, x_4\}$, что соответствует вершине (0, 0). Симплекс-таблица фиксирует коэффициенты системы и строку целевой функции. На каждой итерации выполняются два шага:

1) выбор входящей переменной — небазисная переменная с наибольшим (по модулю) отрицательным коэффициентом в строке z (правило Данцига);

2) выбор выходящей переменной — базисная переменная с минимальным неотрицательным отношением свободного члена к коэффициенту входящей переменной (правило минимального отношения).

Элементарные преобразования строк (приведение к единичному столбцу) пересчитывают таблицу. Процедура повторяется до тех пор, пока все коэффициенты в строке z не станут неотрицательными — это признак оптимума.

Таблица 1. Итерации симплекс-метода для числового примера

Ит.	**Базис**	**Текущая вершина**	**$z = 3x_1 + 2x_2$**	**Действие**
0	x_3, x_4	(0, 0)	0	Начало — ввести x_1 (наибольший коэф.)
1	x_1, x_4	(4, 0)	12	Проверить соседей, нет улучшений
2	x_1, x_4	(4, 0)	12	Оптимум достигнут, стоп

Источник: составлено авторами

Из таблицы 1 видно, что оптимум достигается уже после одной итерации: $x_1 = 4$, $x_2 = 0$, $z^* = 12$. Вершина (4, 0) является единственным оптимальным решением, поскольку ни одна соседняя вершина не даёт большего значения целевой функции.

Реализация на Python.

Приведём программный код с использованием scipy.optimize.linprog. Функция минимизирует целевую функцию, поэтому для задачи максимизации необходимо инвертировать знаки коэффициентов:

```
from scipy.optimize import linprog
c = [-3, -2]                    # коэф. цел. функции (минус)
A = [[1, 1], [1, 3]]            # матрица ограничений
b = [4, 6]                      # правые части
bounds = [(0, None), (0, None)]
res = linprog(c, A_ub=A, b_ub=b, bounds=bounds, method='highs')
print(f'x1={res.x[0]:.2f}, x2={res.x[1]:.2f}, z*={-res.fun:.2f}')
```

Параметр method='highs' активирует решатель HiGHS, реализующий двойственный симплекс-метод. Явное указание method='highs-ds' (dual simplex) или method='highs-ipm' (метод внутренней точки) позволяет сравнивать алгоритмы на одной задаче.

Альтернативный подход — библиотека PuLP — обеспечивает более выразительный синтаксис, приближённый к математической нотации:

```
import pulp
prob = pulp.LpProblem('example', pulp.LpMaximize)
x1 = pulp.LpVariable('x1', lowBound=0)
x2 = pulp.LpVariable('x2', lowBound=0)
prob += 3*x1 + 2*x2           # целевая функция
prob += x1 + x2 <= 4          # ограничение 1
prob += x1 + 3*x2 <= 6        # ограничение 2
prob.solve(pulp.PULP_CBC_CMD(msg=0))
```

Оба подхода возвращают одинаковый результат: $x_1 = 4$, $x_2 = 0$, $z^* = 12$, что полностью совпадает с ручным расчётом. Выбор между ними определяется задачей: scipy.optimize.linprog удобен при интеграции в численные вычисления, PuLP — при необходимости читаемой формулировки задачи и быстрого прототипирования.

Анализ чувствительности.

Практически важным расширением является анализ чувствительности — исследование того, как изменение коэффициентов целевой функции или правых частей ограничений влияет на оптимальное решение. Объект res, возвращаемый linprog, содержит атрибут ineqlin.marginals — теневые цены ограничений (двойственные переменные). Теневая цена первого ограничения равна 3:

увеличение запаса первого ресурса на единицу увеличивает оптимальную прибыль на 3. Второе ограничение не является связанным (slack = 2), поэтому его теневая цена равна нулю — дополнительные единицы второго ресурса не приносят выгоды при данной структуре задачи.

Выводы

1. Симплекс-метод — алгоритм конечного обхода вершин многогранника допустимых решений, гарантирующий нахождение глобального оптимума линейной задачи. Его ключевое достоинство — экспоненциальная сложность в теории при полиномиальном поведении на практике.

2. Введение слабых переменных и построение симплекс-таблицы систематизируют вычисления: правило наибольшего коэффициента (выбор входящей переменной) и правило минимального отношения (выбор выходящей переменной) обеспечивают монотонное улучшение целевой функции на каждой итерации.

3. Язык Python предоставляет два удобных инструмента: scipy.optimize.linprog с решателем HiGHS для высокопроизводительных вычислений и библиотеку PuLP для читаемой формулировки задачи. Оба дают идентичные результаты и рекомендуются для учебного и прикладного использования.

4. Анализ чувствительности, доступный через двойственные переменные, расширяет практическую ценность метода: позволяет менеджеру оценить, насколько выгодно снять то или иное ограничение, без повторного решения задачи.

Дальнейшие исследования могут быть направлены на изучение двойственного симплекс-метода, целочисленного линейного программирования и методов внутренней точки, а также на разработку учебного симулятора пошаговой работы симплекс-таблицы на Python.

Список литературы

1. Данциг Дж. Линейное программирование, его обобщения и применения / пер. с англ. — М.: Прогресс, 1966. — 600 с.
2. Хэдли Дж. Линейное программирование / пер. с англ. — М.: Наука, 1967. — 480 с.
3. Лобанов В. С., Максимов Ю. А. Исследование операций: учеб. пособие. — М.: МИЭТ, 2009. — 256 с.
4. Воскресенский А. К. Линейное программирование. — М.: Финансы и статистика, 2012. — 208 с.
5. Scipy documentation: scipy.optimize.linprog [Электронный ресурс]. — URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html (дата обращения: 01.06.2025).
6. Mitchell S., O'Sullivan M., Dunning I. PuLP: A Linear Programming Toolkit for Python. — Auckland: University of Auckland, 2011. — 65 p.
7. Nocedal J., Wright S. J. Numerical Optimization. 2nd ed. — New York: Springer, 2006. — 664 p.