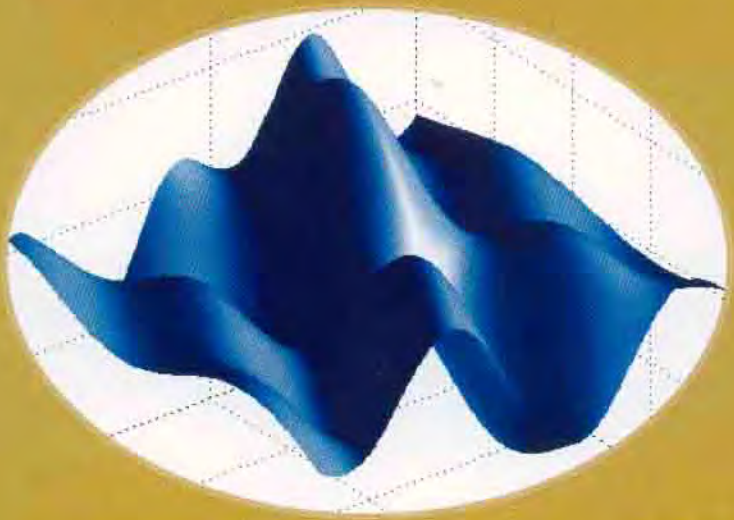
Б.А. Гладких

МЕТОДЫ ОПТИМИЗАЦИИ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ ДЛЯ БАКАЛАВРОВ ИНФОРМАТИКИ



Часть II

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ИНФОРМАТИКИ

Б. А. Гладких

МЕТОДЫ ОПТИМИЗАЦИИ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ ДЛЯ БАКАЛАВРОВ ИНФОРМАТИКИ

Часть II

Нелинейное и динамическое программирование

ТОМСК
«Издательство НТЛ»
2011

УДК 517.8
Г 522

Гладких Б. А. Методы оптимизации и исследование
Г 522 операций для бакалавров информатики. Ч. II. Нелинейное и динамическое программирование: учебное пособие. — Томск: Изд-во НТЛ, 2011. — 264 с.

ISBN 978-5-89503-483-5

Книга написана на основе лекций, в течение ряда лет читавшихся автором на факультете информатики Томского государственного университета.

Во вторую часть вошли разделы, относящиеся к нелинейному программированию (общая теория выпуклого программирования, одномерная оптимизация, многомерная оптимизация без ограничений, оптимизация с ограничениями), а также элементарное введение в динамическое программирование. Обсуждаются вопросы практической оптимизации с помощью популярных пакетов прикладных программ.

Учебное пособие соответствует Государственному образовательному стандарту по направлениям «информационные технологии» и «прикладная информатика», но может быть использовано для студентов, обучающихся по другим инженерным и экономическим направлениям.

УДК 517.8

Р е ц е н з е н т ы:

доктор техн. наук, профессор **В. В. Поддубный**,
кандидат техн. наук, доцент **А. П. Рыжаков**

ISBN 978-5-89503-483-5

Оглавление

Введение

Общая задача математического программирования имеет вид

$$f(\overrightarrow{X}) \to \min\,(\max); \tag{1}$$

$$g_i(\overrightarrow{X}) \begin{bmatrix} \geqslant \\ \leqslant \\ = \end{bmatrix} 0, \quad i = 1, \ldots, m, \tag{2}$$

где (1) — *целевая функция (objective function),* (2) — *ограничения (constrains).*

Размерность вектора переменных $\overrightarrow{X} = (x_1, \ldots, x_n)^T$ предполагается конечной, поэтому задачу математического программирования называют *конечномерной оптимизацией* в отличие от бесконечномерных задач *вариационного исчисления (variational calculus).*

В случае, когда когда f и g_i являются линейными функциями, математическое программирование превращается в *линейное (linear programming — LP),* которому мы посвятили всю первую часть нашего курса, а когда хотя бы одна из функций f и g_i нелинейна, мы имеем дело с задачей *нелинейного программирования (nonlinear programming — NLP).*

Задачи нелинейного программирования возникают в различных областях деятельности, когда линейное приближение реальных зависимостей является слишком грубым. Огромное число задач такого рода возникает при автоматизации проектирования и оптимизации технологических процессов. Например, в

[23, кн. 2, с. 273] описывается нелинейная модель оптимизации химического процесса синтеза этиленгликоля, имеющая 22 переменные, 7 ограничений-равенств, 12 ограничений-неравенств и полный набор ограничений переменных сверху и снизу.

В отличие от линейного программирования, где целевая функция представляла собой простейшую линейную форму и проблем с ее определением не было, здесь возможны различные варианты. В реальных задачах целевая функция может быть задана:

- аналитическим выражением, которое может быть сколь угодно сложным;
- алгоритмом вычисления, в том числе программой имитационного моделирования некоторого процесса;
- натурным экспериментом.

Таким образом, само вычисление значения целевой функции при заданных значениях независимых переменных, не говоря уже о ее оптимизации, может представлять определенную проблему и требовать немалых затрат труда.

Множество $D \subset \mathbf{R}^n$, определяемое совокупностью ограничений (2), называется *множеством допустимых решений (feasible decisions set)* или (в русскоязычной литературе) *множеством планов.*

Когда число переменных равно двум, задачу математического программирования легко интерпретировать геометрически. Множество допустимых решений будет представлять собой некоторую область на плоскости, а целевая функция $f(x_1, x_2)$ имеет вид неровной поверхности с холмами и впадинами. Ее можно изобразить в виде трехмерной картинки или, подобно тому, как на географических картах изображается рельеф местности, семейством линий равных значений — *изоцелевых линий* (рис. 1). Как мы увидим в дальнейшем, найти точку глобального экстремума (минимума или максимума) для таких функций бывает очень сложно, алгоритм поиска может заблудиться в сильно изрезанном рельефе.

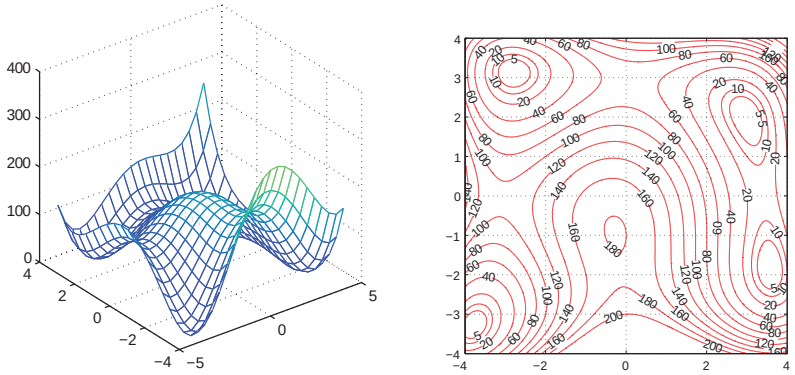


Рис. 1. Пример сложной целевой функции. Трехмерный рельеф и изоцелевые линии

Трудности, порождаемые нелинейностью

В линейном программировании, когда f и g_i были линейными функциями, мы доказали ряд фундаментальных свойств, существенно облегчающих задачу:

- множество планов представляет собой выпуклую многогранную область с конечным числом крайних точек;
- оптимальный план $\overrightarrow{X}^*$ находится в крайней точке множества планов;
- локальный экстремум всегда является глобальным.

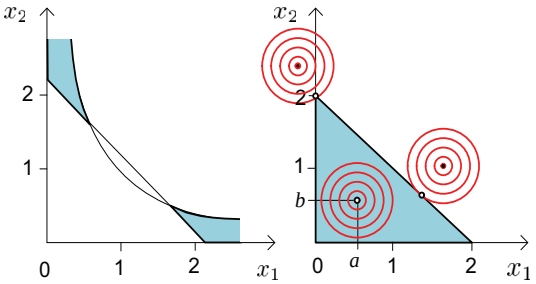
В нелинейном программировании все значительно сложнее.

Во-первых, множество планов может быть не только невыпуклым, но даже несвязным. Пример такого множества представлен на рис. 2, *а*. Оно задается следующими неравенствами[1]:

$$g_1(x_1, x_2) = x_1 x_2 \leqslant 1,$$

[1]В данной книге в численных примерах для отделения целой части числа от дробной мы будем, следуя зарубежной традиции, пользоваться не запятой, а

$$g_2(x_1, x_2) = x_1 + x_2 \geqslant 2.2,$$
$$x_1, x_2 \geqslant 0.$$



а) *б)*

Рис. 2. Трудности, порождаемые нелинейностью

Во-вторых, экстремум может находиться как на границе, так и внутри области. В примере на рис. 2, *б* целевая функция представляет собой простейшую квадратичную форму двух переменных, изоцелевые линии которой являются концентрическими окружностями с общим центром в точке $\overrightarrow{X}^* = (a,\ b)^T$:

$$f_1(x_1, x_2) = (x_1 - a)^2 + (x_2 - b)^2 \to \min,$$
$$g(x_1, x_2) = x_1 + x_2 \leqslant 1,$$
$$x_1, x_2 \geqslant 0.$$

В зависимости от значений a и b оптимальный план находится в крайней точке, на границе или внутри треугольной области планов.

В-третьих, целевая функция может иметь несколько локальных экстремумов, что демонстрирует пример на рис. 1, однако основная задача нелинейного программирования состоит в нахождении самого лучшего из них — *глобального* — оптимума.

десятичной точкой. Запятые будут использоваться для отделения чисел друг от друга. Так значительно удобнее записывать векторы с нецелыми компонентами.

Классификация задач и методов. Структура книги

В связи с указанными трудностями общих методов решения любых задач математического программирования не существует. Изобретены сотни различных методов, но ни один из них по силе и универсальности не может сравниться с симплексным алгоритмом линейного программирования. Давно замечено, что если от некоторой болезни имеется много лекарств, то это верный признак того, что данный недуг радикально неизлечим. Поэтому единственный возможный путь в этой ситуации — выделять специфические классы нелинейных задач и для каждого из них искать свои подходы (рис. 3).

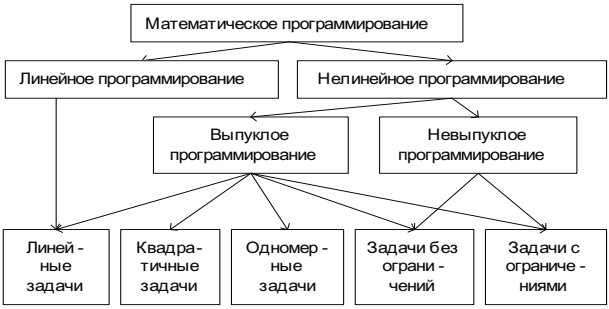


Рис. 3. Классификация задач математического программирования

К настоящему времени наиболее изученным является класс задач *выпуклого программирования (convex programming)*, в которых целевая функция и ограничения задаются выпуклыми функциями. Важнейшим свойством таких задач является единственность экстремума, то есть совпадение локального экстремума с глобальным. Для выпуклых задач удается построить общую теорию, которую мы будем изучать в гл. 9. Она обобщает классический метод множителей Лагранжа и сводит задачу нахождения экстремума при наличии ограничений к решению некоторой системы уравнений и неравенств (условий Куна — Таккера).

Как известно, линейные функции также являются выпуклыми, поэтому с теоретической точки зрения линейное программи-

рование можно рассматривать как частный случай выпуклого. Некоторые лекционные курсы, рассчитанные на чистых математиков, построены именно так, от общего к частному. Однако с практической точки зрения этот подход мало что дает, так как только специфика линейных задач позволяет сконструировать суперэффективные численные методы, подобные симплексному.

Простейшей выпуклой нелинейной задачей является задача *квадратичного программирования (quadratic programming — QP)*, в которой ищется минимум многомерной квадратичной функции при л и н е й н ы х ограничениях. Применяя условия Куна — Таккера, квадратичную задачу удается свести к линейному программированию и получить точное решение за конечное число шагов.

К сожалению, для более сложных случаев точные конечные методы нахождения решения неизвестны. Вместе с тем существует великое множество приближенных итерационных процедур, основанных на тех или иных предположениях и удачных эвристических приемах.

Традиционно задачи и методы нелинейной оптимизации изучаются от простого к сложному, от частного к общему. Как мы увидим в дальнейшем, значительная доля итерационных процедур основана на сведении многомерного поиска к последовательности шагов поиска вдоль заданного направления, то есть на методах поиска экстремума одномерной функции. Этот класс одномерных задач является далеко не тривиальным и заслуживает подробного рассмотрения. Одномерной оптимизации мы посвятим гл. 10.

Следующий по сложности класс задач относится к случаю, когда ограничения отсутствуют, то есть оптимизируемые переменные могут принимать любые значения. С одной стороны, эти задачи *безусловной оптимизации (unconstrained optimization)* являются актуальными сами по себе, когда из содержательной постановки следует, что ограничения несущественны. С другой стороны, они создают идейную основу для решения задач с ограничениями.

Проблеме оптимизации без ограничений в мировой литературе посвящены тысячи публикаций, предложены десятки различных методов. В гл. 11 мы приведем их классификацию и рассмотрим несколько наиболее типичных алгоритмов. К сожалению, все они гарантируют нахождение решения (глобального оптимума) только для выпуклых целевых функций с дополнительными условиями, которые трудно проверить на практике. Если же функция имеет сложный невыпуклый рельеф, то итеративная процедура в лучшем случае сойдется к одному из локальных экстремумов.

Невыпуклая оптимизация (nonconvex optimization), даже при отсутствии ограничений, в современной науке остается по большей части terra incognita. Тем не менее, учитывая важность проблемы, мы в конце главы рассмотрим основные подходы и некоторые практические методы решения многоэкстремальных задач.

Наличие ограничений, хоть и выпуклых, существенно усложняет задачу оптимизации. Несмотря на то, что теория Куна — Таккера указывает теоретический путь решения общей задачи выпуклого программирования, на практике решение соответствующих уравнений и неравенств может оказаться непомерно сложным. Поэтому предложен ряд обходных путей, которые в некоторых частных случаях оказываются более простыми. Основные подходы к выпуклой *оптимизации с ограничениями (constrained optimization)* мы рассмотрим в гл. 12.

Гл. 13 чисто практическая. Она будет полезна тем, кто, столкнувшись с реальной оптимизационной задачей, захочет решить ее с помощью стандартных пакетов прикладных программ.

Гл. 14 уводит изучение проблем оптимизации в другую сторону. Она посвящена оригинальному и чрезвычайно остроумному подходу к решению широкого класса задач, который получил название *динамического программирования (dynamic programming)*. Его основные принципы, элементарные на первый взгляд, но очень глубокие по сути, демонстрируются на простейших содержательных примерах.

В целом, изложение методов нелинейной оптимизации для ба-

калавров информатики имеет целью показать, как ставятся задачи и как устроены основные алгоритмы их решения. Обладая этими знаниями, можно, столкнувшись в жизни с проблемой поиска наилучшего решения, попытаться формализовать ее и предвидеть те трудности, которые возникнут при практической оптимизации. В принципе, этих знаний достаточно и для того, чтобы реализовать на компьютере простейшие алгоритмы и получить полезные результаты, но при этом нужно иметь в виду, что в реальных условиях приходится учитывать множество нюансов и тонкостей, подбирать оптимальные значения многих параметров. При оптимизации сложных целевых функций с нелинейными ограничениями простыми алгоритмами не обойтись. Поэтому на практике, как правило, пользуются промышленными пакетами прикладных программ, разработанными солидными научными организациями под руководством авторитетных ученых с учетом последних достижений в области теории оптимизации. Однако для того, чтобы пользоваться такими пакетами, нужно знать элементы теории и соответствующую терминологию, в том числе англоязычную.

Глава 9

Теория выпуклого программирования

Общепринятой канонической формы задачи выпуклого программирования нет, мы будем считать стандартной задачу м и н и м и з а ц и и выпуклой целевой функции многих переменных

$$f(x_1, \ldots, x_n) \to \min$$

при о г р а н и ч е н и я х - н е р а в е н с т в а х, заданных набором m в ы п у к л ы х функций

$$g_i(x_1, \ldots, x_n) \leqslant 0, \quad i = 1, \ldots, m$$

и дополнительных ограничениях на н е о т р и ц а т е л ь - н о с т ь переменных

$$x_1, \ldots, x_n \geqslant 0.$$

Набор переменных, как и прежде, будем считать вектором-столбцом действительных чисел:

$$\overrightarrow{X} = \begin{pmatrix} x_1 \\ \ldots \\ x_n \end{pmatrix} = (x_1, \ldots, x_n)^T,$$

с учетом этого задачу выпуклого программирования можно записать в векторном виде

$$\begin{aligned} &f(\overrightarrow{X}) \to \min, \\ &g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m, \\ &\overrightarrow{X} \geqslant 0. \end{aligned} \tag{9.1}$$

Изучая теорию линейного программирования, основанную на переборе базисных решений системы линейных уравнений, мы рассуждали в рамках понятий *арифметического* линейного пространства $\mathbf{R}^n$, то есть использовали чисто алгебраические определения линейной комбинации и линейной зависимости столбцов вещественных чисел без привлечения геометрических понятий: направлений, расстояний, углов. Несмотря на то, что элементы этого пространства назывались векторами и обозначались стрелками, они не понимались как объекты, имеющие направления.

В нелинейном программировании геометрические свойства будут играть важную роль, поэтому дальнейшие построения будут вестись в рамках значительно более сложно организованного *евклидова* пространства.

9.1. Евклидово пространство

Точечно-векторное евклидово пространство, которое мы для краткости будем называть просто *евклидовым* пространством $\mathbf{E}^n$, является многомерным обобщением «обычного» трехмерного геометрического пространства.

В школьной математике в качестве первичного геометрического понятия рассматривалась точка, далее определялись геометрические векторы как упорядоченные пары точек и изучались их алгебраические свойства. Здесь же мы поступим наоборот, взяв за основу знакомое арифметическое линейное пространство со всей его алгебраической аксиоматикой [9, с. 55]. Обогащение его гео-

метрическими категориями осуществляется в два этапа (см. [10, с. 92]):

- на первом этапе векторное $\mathbf{R}^n$ дополняется *точечным* пространством $\mathbf{A}^n$, в совокупности они образуют точечно-векторное *аффинное* пространство (от лат. *affinis* — смежный, соседний);

- на втором этапе в аффинном пространстве с помощью *скалярного произведения* вводятся метрические понятия расстояний и углов, в результате получается евклидово пространство $\mathbf{E}^n$.

Точки и векторы в аффинном пространстве

Пусть имеется арифметическое векторное пространство $\mathbf{R}^n$. Рассмотрим непустое множество $\mathbf{A}^n$ элементов, которые мы будем называть *точками*, причем каждой у п о р я д о ч е н н о й паре точек $A, B \in \mathbf{A}^n$ ставится в соответствие единственный вектор $\overrightarrow{X} = \overrightarrow{AB} \in \mathbf{R}^n$. Таким образом вектор получает геометрическую интерпретацию и ему приписывается *направление* от точки A к точке B. Заметим сразу, что обратное не верно, один и тот же вектор $\overrightarrow{X}$ может соответствовать различным парам точек, то есть векторы являются *свободными*.

Предположим выполнение следующих аксиом:

- **откладывание вектора от точки** (см. рис. 9.1, *а*). *Для каждой точки A и каждого вектора $\overrightarrow{X} \in \mathbf{R}^n$ найдется одна и только одна точка B, что $\overrightarrow{AB} = \overrightarrow{X}$*. Другими словами, определена «межвидовая» операция *сложения точки с вектором*, дающая в результате точку $B = A + \overrightarrow{X}$;

- **сложение векторов по правилу треугольника** (см. рис. 9.1, *б*). *Для любых трех точек A, B, C*

$$\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}.$$

Тогда все точки и все векторы образуют точечно-векторное (аффинное) пространство.

Точки и векторы аффинного пространства можно однозначно сопоставить друг другу. Для этого зафиксируем произвольную точку O — *начало координат.* Тогда по первой аксиоме любой точке A будет однозначно соответствовать вектор $\overrightarrow{OA} = (a_1, \ldots, a_n)^T$, отложенный в данную точку из начала координатп и называемый *радиусом-вектором.* Набор чисел $a_1, \ldots, a_n$ задает *координаты* точки A в аффинной системе координат.

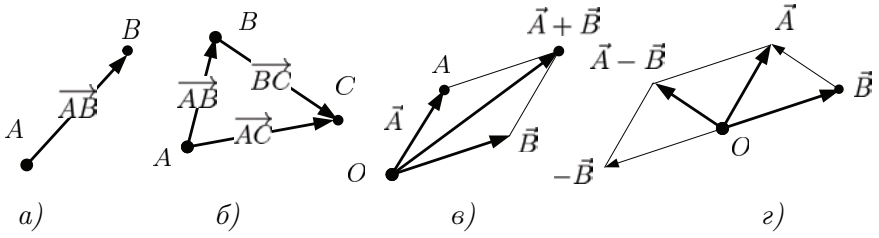


Рис. 9.1. Точки и векторы в аффинном пространстве

Если предполагать начало координат заданным и считать по умолчанию векторы отложенными от точки O, т. е. сокращенно писать $\vec{X}$ вместо $\overrightarrow{OX}$, то правило сложения векторов треугольником может быть заменено эквивалентным «правилом параллелограмма» (см. рис. 9.1, *в*), а вычитание векторов может быть интерпретировано как сложение с противоположным вектором $\vec{A} - \vec{B} = \vec{A} + (-\vec{B})$ (см. рис. 9.1, *г*). При этом из аксиомы сложения векторов следует, что $\overrightarrow{AB} = \overrightarrow{OB} - \overrightarrow{OA} = \vec{B} - \vec{A}$.

С другой стороны, каждому вектору $\vec{X}$ будет соответствовать точка $X = O + \vec{X}$. Поэтому межвидовую операцию сложения точки A с вектором X можно заменить обычным векторным сложением $\overrightarrow{OA} + \vec{X} = \vec{A} + \vec{X}$.

Итак, в аффинном пространстве можно оперировать двумя видами объектов: точками и векторами. Из точек можно строить разнообразные множества (геометрические места) и называть их фигурами. Например, прямая линия, проходящая через точку A в

направлении, заданном вектором $\overrightarrow{d}$, — это геометрическое место точек $X = A + t\,\overrightarrow{d}$, $-\infty \leqslant t \leqslant \infty$.

Если интерпретировать точки как концы соответствующих радиусов-векторов при зафиксированном начале координат, то можно для точек и векторов использовать одинаковое «стрелочное» обозначение, из контекста всегда будет ясно, о чем идет речь. В частности, уравнение прямой можно записать в эквивалентном, чисто векторном виде $\overrightarrow{X} = \overrightarrow{A} + t\,\overrightarrow{d}$, именно такую форму записи мы будем использовать в дальнейшем.

Скалярное произведение

Вводить метрические понятия в аффинном пространстве можно различным образом, однако самым эффективным способом задания возможности измерений является аксиоматическое введение *скалярного произведения векторов.*

Определение. *Аффинное пространство является евклидовым, если каждой паре векторов $\overrightarrow{X}, \overrightarrow{Y}$ поставлено в соответствие вещественное число $\langle \overrightarrow{X}, \overrightarrow{Y} \rangle$, называемое их скалярным произведением, причем выполнены следующие аксиомы:*

1) коммутативность:

$$\langle \overrightarrow{X}, \overrightarrow{Y} \rangle = \langle \overrightarrow{Y}, \overrightarrow{X} \rangle;$$

2) ассоциативность относительно умножения на вещественное число α:

$$\langle \alpha \overrightarrow{X}, \overrightarrow{Y} \rangle = \alpha \langle \overrightarrow{X}, \overrightarrow{Y} \rangle;$$

3) дистрибутивность относительно сложения векторов:

$$\langle \overrightarrow{X} + \overrightarrow{Y}, \overrightarrow{Z} \rangle = \langle \overrightarrow{X}, \overrightarrow{Z} \rangle + \langle \overrightarrow{Y}, \overrightarrow{Z} \rangle;$$

4) положительность скалярного квадрата:

$$\langle \overrightarrow{X}, \overrightarrow{X} \rangle \geqslant 0 \text{ при } \overrightarrow{X} \neq \overrightarrow{0}, \quad \langle \overrightarrow{0}, \overrightarrow{0} \rangle = 0.$$

Для n-мерного арифметического пространства $\mathbf{R}^n$, элементами которого являются столбцы вещественных чисел, понимаемые как матрицы размера $n \times 1$, скалярное произведение векторов $\overrightarrow{X} = (x_1, \ldots, x_n)^T$ и $\overrightarrow{Y} = (y_1, \ldots, y_n)^T$ определяется путем матричного умножения строки на столбец:

$$\langle \overrightarrow{X}, \overrightarrow{Y} \rangle = \overrightarrow{X}^T \overrightarrow{Y} = \sum_{i=1}^{n} x_i y_i.$$

Легко убедиться, что все аксиомы скалярного произведения при этом выполняются.

Измерение длин и углов

Основываясь на базовом определении скалярного произведения, в евклидовом пространстве вводятся фундаментальные понятия длины (нормы) и расстояния.

Определение. *Длиной (нормой, модулем) вектора в евклидовом пространстве называется корень квадратный из его скалярного квадрата*[1]:

$$\|\overrightarrow{X}\| = \sqrt{\langle \overrightarrow{X}, \overrightarrow{X} \rangle} = \sqrt{\overrightarrow{X}^T \overrightarrow{X}} = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Если вектор $\overrightarrow{X}$ умножить на масштабный коэффициент, обратный норме, то длина его станет равна единице. Такая операция называется *нормированием*:

$$\overrightarrow{X}' = \frac{\overrightarrow{X}}{\|\overrightarrow{X}\|}.$$

[1] Иногда определяющим понятием евклидова пространства считается не скалярное произведение, а определенная таким образом норма.

Пользуясь понятием нормы, легко определить расстояние между точками $X(x_1, \ldots, x_n)$ и $Y(y_1, \ldots, y_n)$ в евклидовом пространстве:

$$\rho(X, Y) = \|\overrightarrow{XY}\| = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}.$$

С введением расстояния пространство становится *метрическим,* возможно определение характерных для любого метрического пространства понятий окрестности, сходимости, замкнутости, ограниченности и ряда других, основанных на измерении расстояний.

Принципиальной особенностью евклидова пространства, выделяющей его из общего класса метрических пространств, является возможность измерения не только расстояний, но и углов.

Определение. *Угол φ между (ненулевыми) векторами $\overrightarrow{X}$ и $\overrightarrow{Y}$ в евклидовом пространстве определяется равенством*

$$\cos\varphi = \frac{\langle \overrightarrow{X}, \overrightarrow{Y} \rangle}{\|\overrightarrow{X}\| \cdot \|\overrightarrow{Y}\|}.$$

Векторы, скалярное произведение которых равно нулю (угол между векторами равен $\pm\pi/2$), называются *ортогональными.* Ортогональность обобщает на многомерный случай понятия перпендикулярности, рассматриваемые в двумерном и трехмерном пространствах. Понятие ортогональности можно распространить на множества: два множества векторов являются ортогональными, если каждый вектор первого ортогонален каждому вектору второго.

Векторы, угол между которыми равен 0 или π, называются *коллинеарными.* Очевидно, векторы $\overrightarrow{X}$ и $\overrightarrow{Y}$ коллинеарны тогда и только тогда, когда они пропорциональны, т. е. $\overrightarrow{X} = \lambda\overrightarrow{Y},\ \lambda \neq 0$.

П р и м е р. Вычислим $\varphi = \angle APB$ (рис. 9.2):

$$\cos\varphi = \frac{\langle \overrightarrow{PA}, \overrightarrow{PB} \rangle}{\|\overrightarrow{PA}\| \cdot \|\overrightarrow{PB}\|} = \frac{\langle \overrightarrow{A} - \overrightarrow{P}, \overrightarrow{B} - \overrightarrow{P} \rangle}{\|\overrightarrow{A} - \overrightarrow{P}\| \cdot \|\overrightarrow{B} - \overrightarrow{P}\|} =$$

$$= \frac{(2,1)(2,0)^T}{\sqrt{2^2+1^2}\sqrt{2^2+0^2}} = \frac{4}{2\sqrt{5}}.$$

Отсюда $\varphi \approx 26^\circ$. ▲

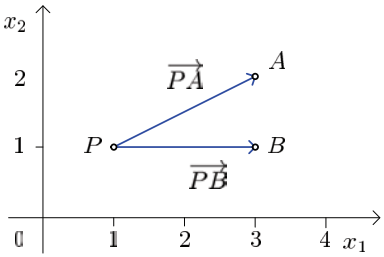


Рис. 9.2. Иллюстрация к примеру

Проекция точки (вектора) на аффинное множество

Пользуясь понятием ортогональности, гиперплоскость, проходящую чарез точку $\overrightarrow{X}_0$, можно определить как геометрическое место точек $\overrightarrow{X}$, таких, что вектор $\overrightarrow{X_0X}$ ортогонален *направляющему* вектору $\overrightarrow{a}$ (рис. 9.3, *а*).

Следовательно, уравнение гиперплоскости можно записать в виде $\overrightarrow{a}^T\overrightarrow{X_0X} = \overrightarrow{a}^T(\overrightarrow{X} - \overrightarrow{X}_0) = 0$ или $\overrightarrow{a}^T\overrightarrow{X} = b$, где обозначено $b = \overrightarrow{a}^T\overrightarrow{X}_0$.

Множество, образованное пересечением m гиперплоскостей в n-мерном пространстве ($m < n$), называется *аффинным множеством* или *линейным многообразием*. Таким образом, аффинное множество задается точкой $\overrightarrow{X}_0$ и направляющими векторами $\overrightarrow{a}_i$, $i = 1, \ldots, m$, причем текущий вектор $\overrightarrow{X_0X}$ ортогонален каждому из них:

$$\overrightarrow{a}_i^T\overrightarrow{X_0X} = \overrightarrow{a}_i^T(\overrightarrow{X} - \overrightarrow{X}_0) = 0, \text{ или } \overrightarrow{a}_i^T\overrightarrow{X} = b_i, \text{ где } b_i = \overrightarrow{a}_i^T\overrightarrow{X}_0.$$

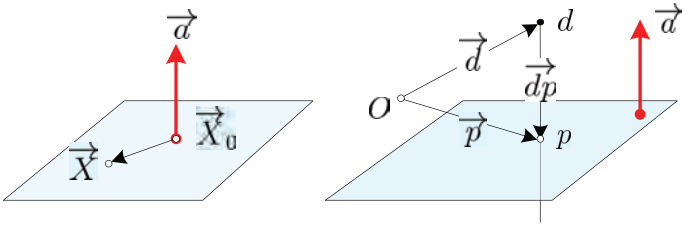


Рис. 9.3. Задание гиперплоскости направляющим вектором *(а)*; проектирование точки на гиперплоскость *(б)*

Если обозначить через $A_{m\times n}$ матрицу со строками $\vec{a}_i^T$, через $\vec{b} = (b_1, \ldots, b_m)^T$ — вектор свободных членов, то аффинное множество задается уравнением $A\vec{X} = \vec{b}$. Этот результат хорошо нам известен в алгебраической формулировке.

Как мы знаем, элементы аффинного пространства могут быть интерпретированы как точки и как векторы. Поэтому важное для последующего понятие *проекции* также может быть описано в «точечном» и «векторном» вариантах. Начнем с точечного.

Определение. *Проекцией точки $\vec{d}$ на множество D называется ближайшая к ней (в смысле введенной евклидовой метрики) точка $\vec{p}$ этого множества.*

Из определения, в частности, следует, что если проектируемая точка находится внутри множества D, то ее проекция совпадает с ней самой.

В общем случае операция проектирования сама по себе представляет непростую оптимизационную задачу, однако в простейшем варианте, когда множество D является гиперплоскостью или аффинным множеством, проекцию можно найти аналитически.

Известно, что в элементарной геометрии длина наклонной не может быть менее длины перпендикуляра. Аналогичное свойство справедливо и для произвольной евклидовой геометрии.

Для начала спроектируем произвольную точку $\vec{d}$ на одну ги-

перплоскость, заданную уравнением $\overrightarrow{a}^T\overrightarrow{X} = b$ (см. рис. 9.3, *б,* где для наглядности изображены оба способа представления проектируемого объекта и проекции: как точек d, p и как концов соответствующих радиусов-векторов из начала координат O). Для этого опустим из $\overrightarrow{d}$ на гиперплоскость перпендикуляр $\overrightarrow{dp}$, коллинеарный нормальному вектору $\overrightarrow{a}$, проекцию $\overrightarrow{p}$ будем искать в виде $\overrightarrow{p} = \overrightarrow{d} + \overrightarrow{dp} = \overrightarrow{d} + \lambda\overrightarrow{a}$. Скалярный множитель λ найдем из условия принадлежности точки $\overrightarrow{p}$ гиперплоскости: $\overrightarrow{a}^T(\overrightarrow{d} + \lambda\overrightarrow{a}) = b$. Отсюда $\lambda = (b - \overrightarrow{a}^T\overrightarrow{d})/(\overrightarrow{a}^T\overrightarrow{a})$ и

$$\overrightarrow{p} = \overrightarrow{d} + (b - \overrightarrow{a}^T\overrightarrow{d})/(\overrightarrow{a}^T\overrightarrow{a})\overrightarrow{a}. \tag{9.2}$$

Процедура проектирования (9.2) легко обобщается на многомерный случай. Пусть множество D задано пересечением m гиперплоскостей $\overrightarrow{a}_i^T\overrightarrow{X} = b_i,\ i = 1, \ldots, m;\ m < n$. Перпендикуляр, опущенный из точки $\overrightarrow{d}$ на множество D, должен быть коллинеарен нормальным векторам всех гиперплоскостей, поэтому проекцию будем искать в виде

$$\overrightarrow{p} = \overrightarrow{d} + \sum_{i=1}^{m} \overrightarrow{a}_i\lambda_i = \overrightarrow{d} + A^T\overrightarrow{\lambda},\ \text{где}\ \overrightarrow{\lambda} = (\lambda_1, \ldots, \lambda_m)^T.$$

Из условия $A\overrightarrow{p} = \overrightarrow{d}$ получаем $\overrightarrow{\lambda} = (AA^T)^{-1}(\overrightarrow{b} - A\overrightarrow{d})$ и

$$\overrightarrow{p} = \overrightarrow{d} + A^T(AA^T)^{-1}(\overrightarrow{b} - A\overrightarrow{d}) = P\overrightarrow{d} + \overrightarrow{R}, \tag{9.3}$$

где обозначены $P = \mathrm{I} - A^T(AA^T)^{-1}A,\ \overrightarrow{R} = A^T(AA^T)^{-1}\overrightarrow{b}$.

Квадратная матрица P размерности $n \times n$ называется *матрицей проектирования.* Легко убедиться, что она является симметрической и существует только тогда, когда ранг произведения $(A_{m\times n}A^T_{n\times m})$ равен m, чтобы было возможно выполнить обращение. Для этого матрица $A_{m\times n}$ должна иметь ранг, равный числу строк m (*полный ранг*). Это значит, что среди плоскостей, на которые производится проектирование, не должно быть параллельных.

Как видно, выражение (9.3) является многомерным обобщением (9.2).

Перейдем к «векторной» интерпретации проекции. Пусть имеется вектор $\overrightarrow{X}$ и аффинное множество D. Возьмем произвольную точку $\overrightarrow{X}_0 \in D$. Отложив от нее наш вектор, получим точку $\overrightarrow{d}$, проекция которой на D есть точка $\overrightarrow{p}$ (рис. 9.4). Тогда под проекцией вектора $\overrightarrow{X}$ на D мы будем понимать вектор $\overrightarrow{h} = \overrightarrow{X_0 p}$. То есть при проектировании вектора на аффинное множество D решается задача разложения произвольного вектора $\overrightarrow{X}$ на сумму двух ортогональных векторов: *проекцию* $Pr_D\overrightarrow{X} = \overrightarrow{h}$ и *ортогональную составляющую* $Ort_D\overrightarrow{X} = \overrightarrow{pd}$:

$$\overrightarrow{X} = Pr_D\overrightarrow{X} + Ort_D\overrightarrow{X}.$$

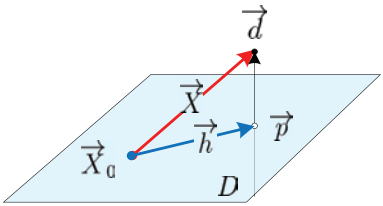


Рис. 9.4. Проектирование вектора на аффинное множество

В этом случае, согласно (9.3),

$$\overrightarrow{p} = P\overrightarrow{d} + \overrightarrow{R} = P(\overrightarrow{X}_0 + \overrightarrow{X}) + \overrightarrow{R} = (P\overrightarrow{X}_0 + \overrightarrow{R}) + P\overrightarrow{X} =$$

$$= Pr_D\overrightarrow{X}_0 + P\overrightarrow{X} = \boxed{\text{проекция точки } \overrightarrow{X}_0 \text{ совпадает с ней самой}} =$$

$$= \overrightarrow{X}_0 + P\overrightarrow{X}.$$

Отсюда

$$Pr_D\overrightarrow{X} = \overrightarrow{X_0 p} = \overrightarrow{p} - \overrightarrow{X}_0 = P\overrightarrow{X}. \qquad (9.4)$$

Видим, что проекция вектора $\overrightarrow{X}$ не зависит от выбора начальной точки $\overrightarrow{X}_0$.

Замечание 1. Понятие проекции может быть сформулировано в более общем виде, если воспользоваться определением *ортогональных подпространств.* Два подпространства евклидова пространства $\mathbf{E}^n$ *ортогональны,* если каждый вектор первого ортогонален каждому вектору второго. Тогда для каждого подпространства $\mathbf{A}$ размерности m (аффинное множество, включающее начало координат, является подпространством) существует его *ортогональное дополнение* $\mathbf{B}$ размерности $n-m$, такое, что для любого $\overrightarrow{X} \in \mathbf{E}^n$

$$\overrightarrow{X} = Pr\overrightarrow{X} + Ort\overrightarrow{X}, \text{ где } Pr\overrightarrow{X} \in \mathbf{A},\ Ort\overrightarrow{X} \in \mathbf{B}.$$

Замечание 2. Легко показать, что матрица проектирования P является симметрической и положительно определенной. Симметрия доказывается прямым транспонированием. Для доказательства положительной определенности рассмотрим произведение $\overrightarrow{X}^T P\overrightarrow{X}$ для любого неотрицательного $\overrightarrow{X}$. Тогда

$$\begin{gathered}\overrightarrow{X}^T P\overrightarrow{X} = (Pr_D\overrightarrow{X} + Ort_D\overrightarrow{X})^T Pr_D\overrightarrow{X} = \\ = Pr_D^T\overrightarrow{X} \cdot Pr_D\overrightarrow{X} + Ort_D^T\overrightarrow{X} \cdot Pr_D\overrightarrow{X} = \|Pr_D\overrightarrow{X}\|^2 + 0 > 0.\end{gathered} \tag{9.5}$$

Замечание 3. Если аффинное множество D задано неособенной квадратной матрицей $A_{n\times n}$, то оно является результатом пересечения непараллельных n гиперплоскостей в n-мерном пространстве, т. е. представляет собой точку. Тогда проекция любого вектора на точку равна нулю. Действительно, для квадратной неособенной матрицы существует обратная и $(AA^T)^{-1} = (A^T)^{-1}A^{-1}$, тогда

$$P = \mathrm{I} - A^T(AA^T)^{-1}A = \mathrm{I} - A^T(A^T)^{-1}A^{-1}A = \mathrm{I} - \mathrm{I} = 0.$$

П р и м е р. Афинное множество в двумерном евклидовом пространстве представляет собой прямую, заданную уравнением $x_1 + 2x_2 = 4$ (рис. 9.5). Найдем проекцию вектора $\overrightarrow{g} = (3,\ 0)^T$ на данную прямую.

Здесь $A = (1,\ 2)$, матрица проектирования

$$P = \mathrm{I} - A^T(AA^T)^{-1}A =$$

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 2 \end{pmatrix} \left[\begin{pmatrix} 1, & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} \right]^{-1} \begin{pmatrix} 1, & 2 \end{pmatrix} = \begin{pmatrix} 0.8 & -0.4 \\ -0.4 & 0.2 \end{pmatrix}.$$

Отсюда $\overrightarrow{h} = P\overrightarrow{g} = (2.4,\ -1.2)^T$. Отложив этот вектор, к примеру, от точки $\overrightarrow{X} = (0,\ 2)^T$, получим точку $\overrightarrow{p} = (2.4,\ 0.8)^T$. ▲

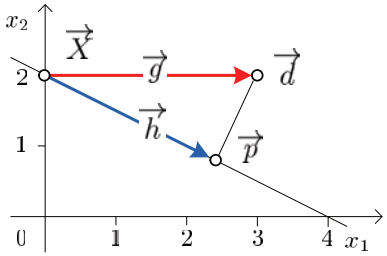


Рис. 9.5. Иллюстрация к примеру

9.2. Выпуклые функции и их свойства

Определение. *Функция многих переменных* $f(\overrightarrow{X})$ *называется выпуклой (строго выпуклой) в выпуклой области* D, *если* $\forall \overrightarrow{X}_1, \overrightarrow{X}_1 \in D, \quad 0 \leqslant \lambda \leqslant 1$:

$$f[(1-\lambda)\overrightarrow{X}_1 + \lambda\overrightarrow{X}_2] \leqslant (<)(1-\lambda)f(\overrightarrow{X}_1) + \lambda f(\overrightarrow{X}_2).$$

В одномерном случае выпуклость проявляется в том, что функция всегда лежит п о д х о р д о й, соединяющей любые две точки на ее графике (см. рис. 9.6, *а*). Непосредственно из определения также следует, что линейная функция является выпуклой, хотя и не строго выпуклой.

Замечание. Если знак неравенства в определении выпуклой функции поменять на обратный, получится определение в о г н у т о й функции. Специально рассматривать этот класс функций не имеет

смысла, так как вогнутую функцию легко превратить в выпуклую, умножив ее на -1.

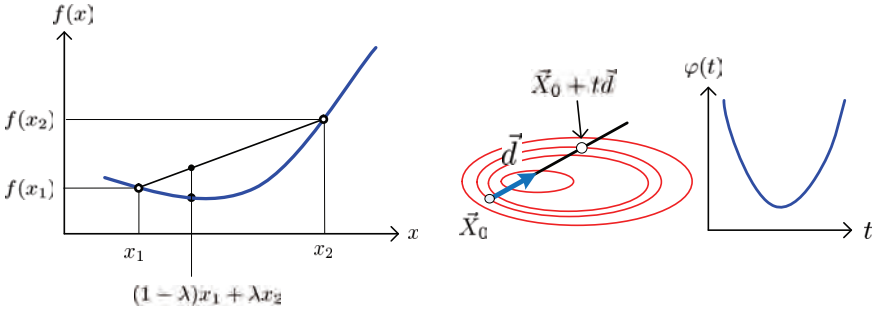
Глобальные свойства

Выпуклые функции обладают рядом полезных свойств. Сначала рассмотрим основные глобальные свойства, не связанные с существованием непрерывных частных производных.

Свойство 1 (неравенство Иенсена[2]). *Выпуклая комбинация выпуклых функций выпукла. То есть если* $f(\vec{X})$ *— выпуклая функция и* $\sum\limits_{i=1}^{m} \alpha_i = 1, \quad \alpha_i \geqslant 0$, *то*

$$f\left(\sum_{i=1}^{m} \alpha_i \vec{X}_i\right) \leqslant \sum_{i=1}^{m} \alpha_i f(\vec{X}_i).$$

Доказательство проводится индукцией по m (см., например, [17, с. 31]).



а) *б)*

Рис. 9.6. Свойства выпуклых функций

[2]Людвиг Иенсен (Jensen, Johan Ludwig; 1859–1925) — датский инженер, сотрудник Копенгагенской телефонной компании. В свободное от работы время занимался математикой. В 1906 г. опубликовал доказательство этого неравенства.

Свойство 2 (выпуклость множества, заданного выпуклой функцией). *Если $g(\overrightarrow{X})$ — выпуклая функция, то множество $D = \{\overrightarrow{X} :\ g(\overrightarrow{X}) \leqslant 0\}$ выпуклое.*

Доказательство. Пусть $\overrightarrow{x}_1 \in D, \overrightarrow{x}_2 \in D$. Это значит, что $g(\overrightarrow{X}_1) \leqslant 0,\ g(\overrightarrow{X}_2) \leqslant 0$.

Составим выпуклую комбинацию $\overrightarrow{X} = (1-\lambda)\overrightarrow{X}_1 + \lambda\overrightarrow{X}_2$:

$$g(\overrightarrow{X}) = g[(1-\lambda)\overrightarrow{X}_1 + \lambda\overrightarrow{X}_2] \leqslant$$
$$\leqslant \underbrace{(1-\lambda)}_{\geqslant 0}\underbrace{g(\overrightarrow{X}_1)}_{\leqslant 0} + \underbrace{\lambda}_{\geqslant 0}\underbrace{g(\overrightarrow{X}_2)}_{\leqslant 0} \leqslant 0. \quad (9.6)$$

Таким образом $\overrightarrow{X} \in D$. ■

Свойство 3 (выпуклость сечения). *Если $f(\overrightarrow{X})$ — выпуклая функция, то функция одной переменной $\varphi(t) = f(\overrightarrow{X}_0 + t\,\overrightarrow{d}\,)$, представляющая собой сечение функции $f(\overrightarrow{X})$ из точки $\overrightarrow{X}_0$ в направлении $\overrightarrow{d}$, является выпуклой* (см. рис. 9.6, *б*).

Доказательство. Данное свойство доказывается непосредственной проверкой:

$$\varphi[(1-\lambda)t_1 + \lambda t_2] = f(\overrightarrow{X}_0 + [(1-\lambda)t_1 + \lambda t_2]\,\overrightarrow{d}\,) =$$
$$= f(\overrightarrow{X}_0 + (1-\lambda)t_1\,\overrightarrow{d} + \lambda t_2\,\overrightarrow{d}\,) = \boxed{\text{прибавим и вычтем } \lambda\overrightarrow{X}_0} =$$
$$= f(\overrightarrow{X}_0 + (1-\lambda)t_1\,\overrightarrow{d} + \lambda t_2\,\overrightarrow{d} + \lambda\overrightarrow{X}_0 - \lambda\overrightarrow{X}_0) =$$
$$= f[(1-\lambda)(\overrightarrow{X}_0 + t_1\,\overrightarrow{d}\,) + \lambda(\overrightarrow{X}_0 + t_2\,\overrightarrow{d}\,)] \leqslant$$
$$\leqslant \boxed{\text{так как } f(\overrightarrow{X}) \text{ — выпуклая функция}} \leqslant$$
$$\leqslant (1-\lambda)f(\overrightarrow{X}_0 + t_1\,\overrightarrow{d}\,) + \lambda f(\overrightarrow{X}_0 + t_2\,\overrightarrow{d}\,) = (1-\lambda)\varphi(t_1) + \lambda\varphi(t_2).$$

Следовательно, $\varphi(t)$ — выпуклая функция. ■

На первый взгляд свойство выпуклости не связано с непрерывностью. Однако на самом деле оно оказывается столь сильным, что из него следуют важные локальные свойства.

Свойство 4 (непрерывность). *Выпуклая функция $f(\overrightarrow{X})$, определенная на выпуклом множестве D, непрерывна в каждой внутренней точке этого множества и имеет производные по любому направлению $\overrightarrow{d}$:*

$$\frac{\partial f(\overrightarrow{X})}{\partial \overrightarrow{d}} = \frac{1}{\|d\|} \lim_{t \to +0} \frac{f(\overrightarrow{X} + t\overrightarrow{d}) - f(\overrightarrow{X})}{t}.$$

Нетривиальное доказательство этого факта можно найти в [24, с. 105].

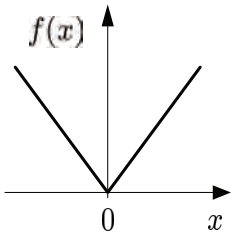


Рис. 9.7.

Замечание. Из приведенного свойства отнюдь не следует дифференцируемость функции, то есть существование непрерывных производных $f(\overrightarrow{X})$ во всех внутренних точках области определения. На рис. 9.7 в качестве примера приведена функция одной переменной $f(x) = |x|$. Эта функция с очевидностью непрерывна и выпукла на всей числовой оси, дифференцируема во всех точках, кроме нуля. В нуле она непрерывна, имеет производные по направлению слева и справа, однако эти производные не совпадают, следовательно, в данной точке функция недифференцируема.

Экстремальные свойства

Поскольку выпуклые функции интересуют нас не сами по себе, а с точки зрения оптимизации, необходимо прежде всего уточнить понятие экстремума.

Пусть задача заключается в нахождении минимума или максимума целевой функции на множестве допустимых значений D:

$$f(\overrightarrow{X}) \to \min_{\overrightarrow{X} \in D} (\max).$$

Для определенности впредь будем иметь в виду задачу минимизации, поскольку максимизация достигается сменой знака у целевой функции.

Глобальным решением или точкой *глобального минимума* называется вектор $\overrightarrow{X}^*$, такой, что

$$\forall \overrightarrow{X} \in D \quad f(\overrightarrow{X}^*) \leqslant f(\overrightarrow{X}). \tag{9.7}$$

Локальным решением или точкой *локального минимума* называется вектор $\overrightarrow{X}^*$, такой, что существует некоторая ε-окрестность точки $\overrightarrow{X}^*$, образованная пересечением шара $R_\varepsilon = \{\overrightarrow{X} : \|\overrightarrow{X} - \overrightarrow{X}^*\| \leqslant \varepsilon\}$ с допустимой областью $D_\varepsilon = R_\varepsilon \cap D$, в которой

$$\forall \overrightarrow{X} \in D_\varepsilon \quad f(\overrightarrow{X}^*) \leqslant f(\overrightarrow{X}). \tag{9.8}$$

В зависимости от вида неравенств в (9.7) и (9.8) говорят о строгих или нестрогих минимумах в глобальном или локальном смысле. Всякий глобальный минимум является локальным, но не наоборот. Для глобального решения часто используется обозначение $\overrightarrow{X}^* = \arg\min\limits_{\overrightarrow{X} \in D} f(\overrightarrow{X})$.

Свойство 5. *Любой локальный минимум выпуклой функции $f(\overrightarrow{X})$ на выпуклом множестве D является глобальным.*

Доказательство (от противного). Предположим, что точка локального оптимума $\overrightarrow{X}^* \in D$ не является глобальной. Тогда найдется $\overrightarrow{X}' \in D$, такая, что

$$f(\overrightarrow{X}') < f(\overrightarrow{X}^*).$$

Рассмотрим точки вида

$$\overrightarrow{X} = (1 - \lambda)X^* + \lambda X', \quad 0 \leqslant \lambda \leqslant 1.$$

Поскольку D — выпуклое множество, то $X \in D$. Далее, в силу выпуклости функции $f(\overrightarrow{X})$,

$$f(\overrightarrow{X}) = f[(1-\lambda)X^* + \lambda X'] \leqslant (1-\lambda)f(X^*) + \lambda f(X') <$$
$$< \boxed{\text{так как} f(\overrightarrow{X}') < f(\overrightarrow{X}^*)} < (1-\lambda)f(X^*) + \lambda f(X^*) = f(X^*).$$

Но это противоречит утверждению о том, что $\overrightarrow{X}^*$ — точка локального минимума, так как при $0 < \lambda < \varepsilon$ точка $\overrightarrow{X}$ находится в ε-окрестности точки $\overrightarrow{X}^*$. ■

Доказанное свойство имеет фундаментальное значение для выпуклой оптимизации, поскольку позволяет заменить глобальные условия минимума локальными, основанными на дифференциальных свойствах функций в предположении, что все частные производные существуют и непрерывны.

Дифференциальные свойства

Вектор частных производных функции многих переменных $f(\overrightarrow{X})$ в точке $\overrightarrow{X}$ называется *градиентом* и обозначается как

$$\overrightarrow{\nabla} f(\overrightarrow{X}) = \left(\frac{\partial f(\overrightarrow{X})}{\partial x_1}, \ldots, \frac{\partial f(\overrightarrow{X})}{\partial x_n} \right)^T.$$

Из курса математического анализа (см., например, [15, с. 19]) известно, что вектор градиента указывает направление скорейшего увеличения функции $f(\overrightarrow{X})$ в точке $\overrightarrow{X}$. Противоположный ему по направлению вектор $-\overrightarrow{\nabla} f(\overrightarrow{X})$ называется *антиградиентом*, он определяет направление скорейшего убывания. Скорость изменения $f(\overrightarrow{X})$ по направлению, задаваемому произвольным вектором $\overrightarrow{d}$, называется *производной по направлению (directional derivative)* и может быть выражена как проекция градиента на выбранное направление:

$$\frac{\partial f(\overrightarrow{X})}{\partial \overrightarrow{d}} = \lim_{t \to 0} \frac{f(\overrightarrow{X} + t\overrightarrow{d}) - f(\overrightarrow{X})}{\| t\overrightarrow{d} \|} =$$
$$= \mathrm{pr}_{\overrightarrow{d}} \overrightarrow{\nabla} f(\overrightarrow{X}) = \frac{\langle \overrightarrow{\nabla} f(\overrightarrow{X}), \overrightarrow{d} \rangle}{\| \overrightarrow{d} \|} = \frac{\overrightarrow{\nabla}^T f(\overrightarrow{X})\, \overrightarrow{d}}{\| \overrightarrow{d} \|}. \qquad (9.9)$$

Вектор градиента имеет еще одну важную геометрическую интерпретацию. Если задана функция нескольких переменных $f(\overrightarrow{X})$, то уравнение $f(\overrightarrow{X}) = \text{const}$ при различных значениях const задает семейство поверхностей равных значений (равного уровня) функции $f(\overrightarrow{X})$ в пространстве переменных $\overrightarrow{X}$. Если фиксировать точку $\overrightarrow{X}_0$, то уравнение $f(\overrightarrow{X}) = f(\overrightarrow{X}_0)$ определит поверхность равного уровня, проходящую через точку $\overrightarrow{X}_0$. Тогда $\overrightarrow{\nabla} f(\overrightarrow{X}_0)$ есть *вектор внешней нормали* к этой поверхности в точке $\overrightarrow{X}_0$, а уравнение касательной гиперплоскости к ней в точке $\overrightarrow{X}_0$ имеет вид $\overrightarrow{\nabla}^T f(\overrightarrow{X}_0)(\overrightarrow{X} - \overrightarrow{X}_0) = 0$.

Свойство 6. *Если $f(\overrightarrow{X})$ выпукла и дифференцируема на выпуклом множестве, то для любых двух точек $\overrightarrow{X}, \overrightarrow{X}_0$ этого множества*

$$f(\overrightarrow{X}) \geqslant f(\overrightarrow{X}_0) + \overrightarrow{\nabla}^T f(\overrightarrow{X}_0)(\overrightarrow{X} - \overrightarrow{X}_0). \tag{9.10}$$

В одномерном случае это свойство совершенно прозрачно: выпуклая функция располагается над касательной прямой в любой точке (рис. 9.8): $f(x_1) \geqslant f(x_0) + f'(x_0)(x - x_0)$.

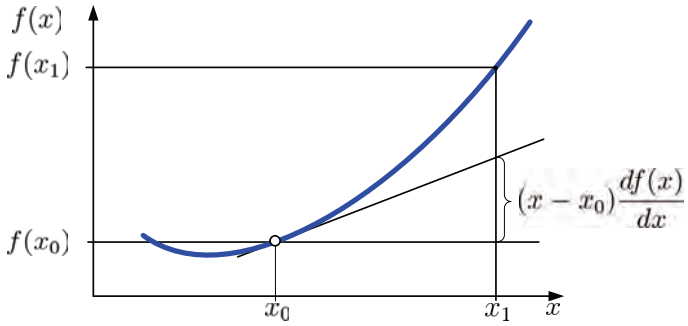


Рис. 9.8. Иллюстрация к свойству 6

Доказательство. По определению выпуклой функции для любого $0 \leqslant \lambda \leqslant 0$

$$f[(1-\lambda)\overrightarrow{X}_0 + \lambda\overrightarrow{X}] \leqslant (1-\lambda) f(\overrightarrow{X}_0) + \lambda f(\overrightarrow{X}).$$

Это неравенство может быть записано по-другому:

$$f[\overrightarrow{X}_0 + \lambda(\overrightarrow{X} - \overrightarrow{X}_0)] \leqslant f(\overrightarrow{X}_0) + \lambda[f(\overrightarrow{X}) - f(\overrightarrow{X}_0)]. \qquad (9.11)$$

Введем обозначения для направления и длины шага:

$$\overrightarrow{d} = \frac{\overrightarrow{X} - \overrightarrow{X}_0}{\|\overrightarrow{X} - \overrightarrow{X}_0\|}, \quad t = \lambda\|\overrightarrow{X} - \overrightarrow{X}_0\|.$$

С учетом этого (9.11) перепишется в виде

$$\|\overrightarrow{X} - \overrightarrow{X}_0\|\frac{f(\overrightarrow{X} + t\,\overrightarrow{d}) - f(\overrightarrow{X}_0)}{t} \leqslant f(\overrightarrow{X}) - f(\overrightarrow{X}_0).$$

Переходя к пределу при $t \to +0$ и учитывая, что $\|\,\overrightarrow{d}\,\| = 1$, слева получаем производную по направлению:

$$\|\overrightarrow{X} - \overrightarrow{X}_0\|\frac{\partial f(\overrightarrow{X}_0)}{\partial\,\overrightarrow{d}} \leqslant f(\overrightarrow{X}) - f(\overrightarrow{X}_0). \qquad (9.12)$$

Но производная по направлению равна проекции градиента на данное направление:

$$\frac{\partial f(\overrightarrow{X}_0)}{\partial\,\overrightarrow{d}} = \frac{\overrightarrow{\nabla}^T f(\overrightarrow{X}_0)\,\overrightarrow{d}}{\|d\|} = \frac{\overrightarrow{\nabla}^T f(\overrightarrow{X}_0)(\overrightarrow{X} - \overrightarrow{X}_0)}{\|\overrightarrow{X} - \overrightarrow{X}_0\|}. \qquad (9.13)$$

Подставляя (9.13) в (9.12), получаем (9.10). ■

Переходим к частным производным второго порядка. Матрица вторых частных производных (если они существуют), вычисленных в точке $\overrightarrow{X}$, называется *матрицей Гессе*[3] или *гессианом* и обозначается как

[3] Людвиг Гессе (Hesse, Ludwig Otto; 1811–1874) — немецкий математик, изучавший эту матрицу. Сам Гессе пользовался термином «функциональные детерминанты».

$$\mathrm{H}(\overrightarrow{X}) = \nabla^2 f(\overrightarrow{X}) = \frac{\partial^2 f(\overrightarrow{X})}{\partial \overrightarrow{X} \partial \overrightarrow{X}} = \begin{pmatrix} \dfrac{\partial^2 f(\overrightarrow{X})}{\partial x_1 \partial x_1} & \cdots & \dfrac{\partial^2 f(\overrightarrow{X})}{\partial x_1 \partial x_n} \\ \cdots & \cdots & \cdots \\ \dfrac{\partial^2 f(\overrightarrow{X})}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 f(\overrightarrow{X})}{\partial x_n \partial x_n} \end{pmatrix}.$$

Свойство 7. *Дважды дифференцируемая функция $f(\overrightarrow{X})$ выпукла (строго выпукла) в окрестности точки $\overrightarrow{X}$ тогда и только тогда, когда ее матрица Гессе $\mathrm{H}(\overrightarrow{X})$ неотрицательно (положительно) определена в этой точке.*

Доказательство не приводим, его можно найти в любом учебнике по математическому анализу.

Напомним, что положительная (неотрицательная) определенность матрицы $A = (a_{ij})$ означает, что для любого вектора $\overrightarrow{Z}$ квадратичная форма $\overrightarrow{Z}^T A \overrightarrow{Z} = \sum_i \sum_j a_{ij} z_i z_j$ строго положительна (неотрицательна). Для проверки матрицы на положительную определенность существует *критерий Сильвестра:* матрица является положительно (неотрицательно) определенной, если все ее угловые миноры положительны (неотрицательны).

Квадратичная функция

Исключительно важную роль в нелинейном программировании играет многомерная *квадратичная функция*, общий вид которой включает квадратичную, линейную составляющие и свободный член:

$$Q(\overrightarrow{X}) = \sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_i x_j + \sum_{i=1}^{n} c_j x_j + a = \frac{1}{2} \overrightarrow{X}^T D \overrightarrow{X} + \overrightarrow{c}^T \overrightarrow{X} + a, \quad (9.14)$$

где D — симметричная квадратная матрица (матрица квадратичной формы). Градиент и матрица Гессе для квадратичной функции равны соответственно

$$\overrightarrow{\nabla} f(\overrightarrow{X}) = D\overrightarrow{X} + \overrightarrow{c}, \quad \mathrm{H}(\overrightarrow{X}) = 2D.$$

Таким образом, выпуклость квадратичной функции целиком определяется определенностью матрицы D.

Можно показать (см., например, [16, с. 214]), что путем переноса начала координат в точку

$$\overrightarrow{X}^* = -\frac{1}{2}D^{-1}\overrightarrow{c}$$

можно избавиться от линейной составляющей, и в новой системе координат $\overrightarrow{X}'$ функция (9.14) превращается в простую квадратичную форму со свободным членом a', не зависящим от $\overrightarrow{X}$:

$$Q(\overrightarrow{X}') = \overrightarrow{X}'^T D\overrightarrow{X}' + a', \text{ где } a' = a - \frac{1}{4}\overrightarrow{c}^T d^{-1}\overrightarrow{c}.$$

Далее, поворотом координатных осей полученную квадратичную форму можно привести к *каноническому виду*

$$Q(\overrightarrow{X}'') = \sum_{i=1}^{n} \lambda_i (x_i'')^2 + a',$$

где λ_i — собственные числа матрицы D.

Форма поверхности равных значений целевой функции (в двумерном случае — изоцелевых линий) целиком определяется определенностью матрицы D и ее собственными числами.

Если матрица положительно определена, то $Q(\overrightarrow{X})$ выпукла, все собственные числа вещественны и положительны. Поверхности равных уровней такой функции представляют собой многомерные концентрические эллипсоиды (в двумерном случае — эллипсы) с центром в $\overrightarrow{X}^*$, являющейся точкой минимума $Q(\overrightarrow{X})$.

Размеры главных осей эллипсоидов определяются собственными числами $M = \lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n = m$. Таким образом, степень сплющенности эллипсоидов определяется отношением наибольшего и наименьшего собственных чисел матрицы D, которое называется *числом обусловленности (conditional number)* матрицы и обозначается $\mathrm{cond}(D)$. Если $\mathrm{cond}(D) = \frac{M}{m} \approx 1$, то поверхности равных уровней близки к сферическим, если же $\mathrm{cond(H)} \gg 1$,

то матрица относится к классу плохо обусловленных, вследствие чего эллипсоиды равных уровней сильно вытянуты. Как мы увидим в дальнейшем, плохая обусловленность существенно усложняет задачу минимизации квадратичной функции.

Направления главных осей эллипсоидов определяются собственными векторами, соответствующими собственным числам, при этом наибольшей полуоси соответствует наименьшее собственное число и наоборот.

П р и м е р 1. Пусть квадратичная функция двух переменных задана выражением

$$Q(x_1, x_2) = x_1^2 + 3x_2^2 + 2x_1x_2 - 4x_1 - 6x_2 + 4.5.$$

Отсюда

$$D = \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix}, \quad \overrightarrow{c} = \begin{pmatrix} -4 \\ -6 \end{pmatrix}, \quad a = 4.5.$$

Угловые миноры матрицы D положительны:

$$\|1\| > 0, \quad \begin{Vmatrix} 1 & 1 \\ 1 & 3 \end{Vmatrix} > 0,$$

следовательно, функция $Q(x_1, x_2)$ строго выпукла во всей бесконечной области определения. Для нахождения собственных чисел составим характеристическое уравнение

$$\|D - \lambda \mathrm{I}\| = \begin{Vmatrix} 1-\lambda & 1 \\ 1 & 3-\lambda \end{Vmatrix} = \lambda^2 - 4\lambda + 2 = 0.$$

Решая его, получаем $M = \lambda_1 = 3.4142$; $m = \lambda_2 = 0.5858$. Число обусловленности $\mathrm{cond}(D) \approx 5.83$, следовательно, отношение больших и малых полуосей эллипсов равно $\sqrt{\mathrm{cond}(D)} \approx 2.41$.

На рис. 9.9 изображено семейство изолиний данной квадратичной функции, при этом координаты общего центра эллипсов

находятся в точке

$$\overrightarrow{X}^* = -\frac{1}{2}D^{-1}\overrightarrow{c} = -\frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & 3\end{pmatrix}^{-1}\begin{pmatrix}-4\\ -6\end{pmatrix} = \begin{pmatrix}1.5\\ 0.5\end{pmatrix}.$$

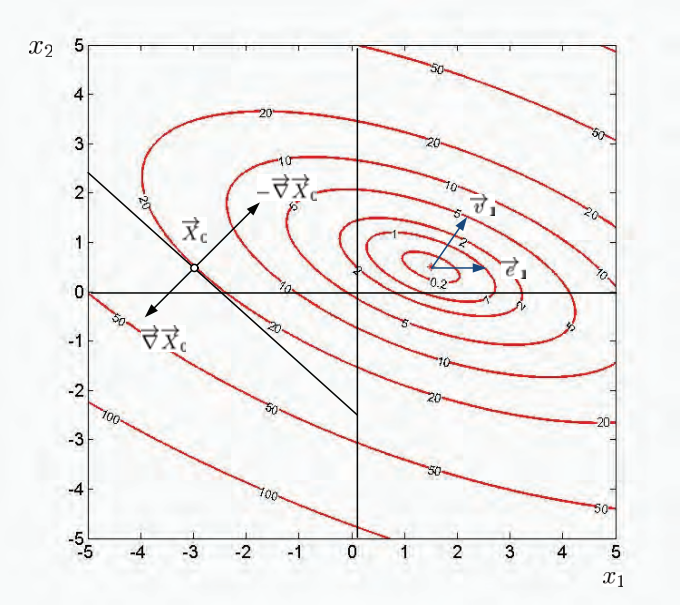


Рис. 9.9. Пример выпуклой квадратичной функции

Для нахождения направлений главных осей эллипсов определим собственные векторы матрицы D, соответствующие найденным ранее собственным числам. Воспользовавшись процедурой вычисления собственных чисел матрицы из пакета MATLAB, получаем

$$\overrightarrow{v}_1 = (0.3827,\ 0.9239)^T,\ \overrightarrow{v}_2 = (-0.9239,\ 0.3827)^T.$$

Углы между направлениями главных осей эллипсоидов и любой из координатных осей можно измерить, применив формулу для

вычисления углов между двумя векторами в евклидовом пространстве:

$$\cos\varphi = \frac{\langle \overrightarrow{X}, \overrightarrow{Y} \rangle}{\|\overrightarrow{X}\| \cdot \|\overrightarrow{Y}\|}.$$

Если положить $\overrightarrow{X} = \overrightarrow{v}_1$, $\overrightarrow{Y} = \overrightarrow{e}_1$ и учесть, что $\|\overrightarrow{v}_1\| = \|\overrightarrow{e}_1\| = 1$, получим выражение для угла между малой полуосью эллипса (ей соответствует бóльшее собственное значение λ_1) и осью абсцисс

$$\cos\varphi = \langle \overrightarrow{v}_1, \overrightarrow{e}_1 \rangle = (0.3827, 0.9239) \begin{pmatrix} 1 \\ 0 \end{pmatrix} = 0.3827.$$

Отсюда $\varphi = 67,5^\circ$ (см. рис. 9.9). Заметим, что для случая двух переменных угол наклона эллипса можно определить непосредственно по матрице D квадратичной формы: $\operatorname{tg} 2\varphi = \dfrac{2d_{12}}{d_{11} - d_{22}}$ [18, с. 66].

Продолжим пример и посмотрим, как ведет себя градиент. Возьмем точку $\overrightarrow{X}_0 = (-3,\, 0.5)^T$. Градиент функции в этой точке равен

$$\overrightarrow{\nabla} Q(\overrightarrow{X}_0) = \overrightarrow{c} + 2D\overrightarrow{X}_0 = \begin{pmatrix} -4 \\ -6 \end{pmatrix} + 2 \begin{pmatrix} 1 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} -3 \\ 0.5 \end{pmatrix} = \begin{pmatrix} -9 \\ -9 \end{pmatrix}$$

и направлен, как видно, влево и вниз. Соответственно вектор антиградиента направлен вправо и вверх (из-за недостатка места длины векторов на рисунке уменьшены).

Построим касательную к изолинии в данной точке. Ее уравнение имеет вид

$$\overrightarrow{\nabla}^T Q(X_0)(\overrightarrow{X} - \overrightarrow{X}_0) = \begin{pmatrix} -9 \\ -9 \end{pmatrix}^T \begin{pmatrix} x_1 + 3 \\ x_2 - 0.5 \end{pmatrix} = 0.$$

Выполняя скалярное произведение, получаем уравнение прямой $-9x_1 - 9x_2 - 22.5 = 0$. Если привести его к виду уравнения в

отрезках, получим $\dfrac{x_1}{-2.5}+\dfrac{x_2}{-2.5}=1$. То есть касательная отсекает на осях координат x_1, x_2 отрезки -2.5 и -2.5, что мы и видим на рис. 9.9. ▲

Пример 2. В качестве контрпримера рассмотрим квадратичную функцию $Q(x_1,x_2)=x_1^2+3x_2^2+4x_1x_2$, заданную матрицей

$$D=\begin{pmatrix}1 & 2\\ 2 & 3\end{pmatrix}.$$

Данная матрица не удовлетворяет критерию Сильвестра и имеет собственные числа $\lambda_1=-0.24$, $\lambda_2=4.23$. Как следствие, функция невыпукла, ее изоцелевые линии представляют семейство гипербол с общим центром в начале координат (рис. 9.10). Точка $\overrightarrow{X}^*=(0,\ 0)^T$ является *седловой точкой (saddle point)*, так как в ней функция $Q(x_1,x_2)$ имеет минимум по одному направлению и максимум по другому. ▲

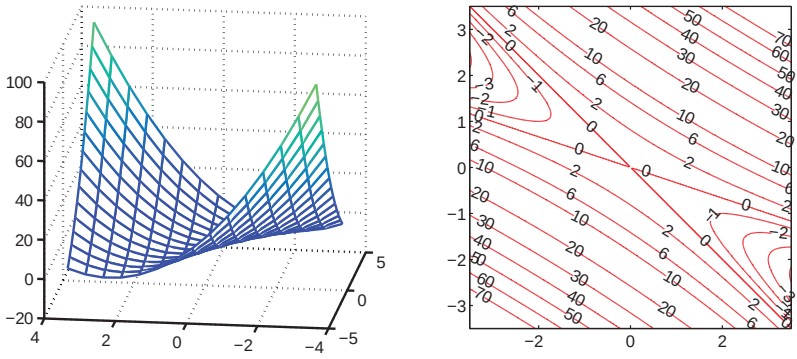


Рис. 9.10. Трехмерный рельеф и изолинии невыпуклой квадратичной функции двух переменных

9.3. Классические задачи оптимизации

Прежде чем исследовать общую задачу выпуклого программирования (9.1), рассмотрим ее упрощенные варианты, изучаемые в классическом курсе математического анализа. Все участвующие в постановке задач функции предполагаются при этом г л а д к и м и, т. е. непрерывно дифференцируемыми.

Оптимизация без ограничений

В простейшем случае ограничения на оптимизируемые переменные отсутствуют, и мы имеем задачу минимизации функции многих переменных в неограниченной области

$$f(\overrightarrow{X}) = f(x_1, \ldots, x_n) \to \min,$$
$$0 < x_j < \infty, \quad j = 1, \ldots, n.$$

В курсе математического анализа доказывается [15, с. 31], что н е о б х о д и м ы м условием локального минимума является равенство нулю всех частных производных (теорема Ферма):

$$\frac{\partial f(\overrightarrow{X})}{\partial x_j} = 0, \quad j = 1, \ldots, n. \tag{9.15}$$

Это равенство является *условием первого порядка (first order condition)* и может быть записано в компактной векторной форме

$$\overrightarrow{\nabla} f(\overrightarrow{X}) = 0. \tag{9.16}$$

Точки $\overrightarrow{X}^*$, удовлетворяющие условию первого порядка, называются *стационарными*. Однако стационарность не обязательно связана с минимумом, стационарными являются точки как минимума, так и максимума, а также точки перегиба одномерных функций или седловые точки в многомерном случае.

Д о с т а т о ч н ы м условием локального минимума является положительная определенность матрицы Гессе $\mathrm{H}(\overrightarrow{X})$ в стационарной точке. Этот факт представляется достаточно очевидным, так

как в окрестности этой точки функцию $f(\overrightarrow{X})$ можно аппроксимировать квадратичной функцией

$$Q(\overrightarrow{X}) = f(\overrightarrow{X}^*) + \overrightarrow{\nabla}^T f(\overrightarrow{X}^*)(\overrightarrow{X} - \overrightarrow{X}^*) + \frac{1}{2}(\overrightarrow{X} - \overrightarrow{X}^*)^T \mathrm{H}(\overrightarrow{X}^*)(\overrightarrow{X} - \overrightarrow{X}^*).$$

Если $\mathrm{H}(\overrightarrow{X}^*)$ положительно определена, то $Q(\overrightarrow{X})$ является выпуклой, и точка $\overrightarrow{X}^*$ доставляет ей минимум. Таким образом, для выпуклых функций условие первого порядка является необходимым и достаточным.

Ограничения на неотрицательность

Следующий случай чуть сложнее. Он похож на предыдущий, но в базовом курсе матанализа рассматривается реже, поэтому его можно отнести к «полуклассическим». Единственным видом ограничений является неотрицательность переменных, задача имеет вид

$$f(\overrightarrow{X}) = f(x_1, \ldots, x_n) \to \min,$$
$$0 \leqslant x_j < \infty, \quad j = 1, \ldots, n.$$

Легко показать, что необходимое условие минимума (9.15) для каждой переменной в этом случае имеет вид

$$\begin{aligned} &\text{а)} \quad \frac{\partial f(\overrightarrow{X})}{\partial x_j} \geqslant 0, \\ &\text{б)} \quad x_j \frac{\partial f(\overrightarrow{X})}{\partial x_j} = 0, \\ &\text{в)} \quad x_j \geqslant 0, \\ &\qquad j = 1, \ldots, n. \end{aligned} \tag{9.17}$$

Чтобы убедиться в этом, рассмотрим рис. 9.11. На нем изображены два варианта сечения функции $f(\overrightarrow{X})$ по одной из координат x_j. В левом варианте безусловный минимум функции по этой переменной находится в области отрицательных значений.

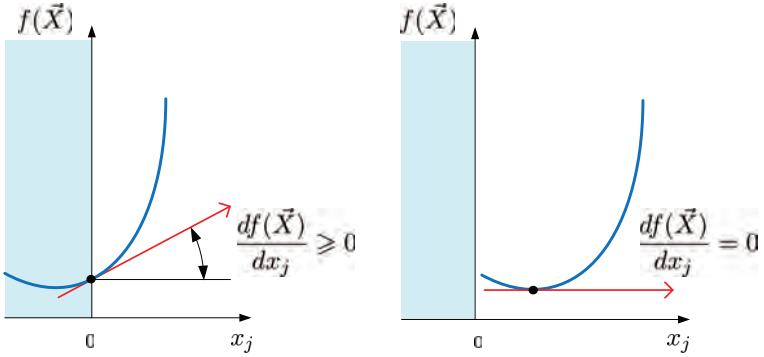


Рис. 9.11. Условная оптимизация по одной переменной с ограничением на ее неотрицательность

Поскольку одномерная функция сечения выпукла и справа от минимума возрастает, то ограничение на неотрицательность является в данном случае активным. Минимум достигается в нуле, а производная в этой точке неотрицательна.

Справа иллюстрируется случай, когда точка безусловного минимума по x_j находится на положительной полуоси и ограничение на неотрицательность неактивно. В этом случае необходимым условием минимума является обычное равенство нулю первой производной.

Условия (9.17) являются алгебраическим воплощением данных геометрических построений. Если $x_j > 0$, то условие «б» превращается в $\dfrac{\partial f(\overrightarrow{X})}{\partial x_j} = 0$, условие «а» ему не противоречит, если же $x_j = 0$, то работает условие «а», а условие «б» при этом не работает.

В векторной форме условия (9.17) приобретают вид

$$\begin{aligned} &\text{а)} \quad \overrightarrow{\nabla} f(\overrightarrow{X}) \geqslant 0, \\ &\text{б)} \quad \overrightarrow{X}^T \overrightarrow{\nabla} f(\overrightarrow{X}) = 0, \\ &\text{в)} \quad \overrightarrow{X} \geqslant 0. \end{aligned}$$

Если оптимизируемая функция выпукла, эти условия являются не только необходимыми, но и достаточными.

Ограничения-равенства

Еще один классический случай имеет место, когда ограничения задаются совокупностью уравнений, в этом случае условная экстремальная задача приобретает вид

$$f(\overrightarrow{X}) = f(x_1, \ldots, x_n) \to \min,$$
$$g_i(\overrightarrow{X}) = g_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m.$$

Стандартный подход к решению этой задачи [15, с. 35] основан на использовании *функции Лагранжа (лагранжиана)*

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i g_i(\overrightarrow{X}), \tag{9.18}$$

зависящей не только от оптимизируемых переменных $\overrightarrow{X} = (x_1, \ldots, x_n)^T$, но и от дополнительных переменных $\overrightarrow{Y} = (y_1, \ldots, y_m)^T$ — так называемых *множителей Лагранжа (Lagrange multipliers),* число которых равно количеству ограничений.

Необходимые (и достаточные в случае выпуклости всех участвующих функций) условия минимума имеют вид

$$\frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_j} = 0, \quad j = 1, \ldots, n; \tag{9.19}$$

$$\frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial y_i} = 0, \quad i = 1, \ldots, m. \tag{9.20}$$

Несмотря на всю неочевидность этого утверждения, оно объясняется довольно просто. Для начала предположим, что имеется задача минимизации при единственном ограничении. На рис. 9.12, представляющем двумерный случай, жирной линией изображено множество допустимых точек, удовлетворяющих уравнению $g(\overrightarrow{X}) = 0$ на фоне семейства изолиний целевой функции $f(\overrightarrow{X})$.

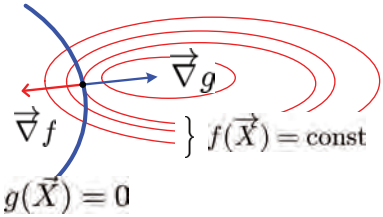


Рис. 9.12. Условная оптимизация с одним ограничением-равенством

Задача состоит в том, чтобы найти такую допустимую точку, которой соответствует наименьшее значение целевой функции. Двигаясь по допустимой кривой, например сверху вниз, мы будем пересекать изоцелевые линии, уменьшая значение целевой функции до тех пор, пока не достигнем точки касания с одной из них. Это и есть искомая точка оптимума $\vec{X}^*$. В ней касательные прямые к допустимой кривой и изоцелевой линии совпадают, следовательно, вектор нормали к изоцелевой линии, определяемый градиентом целевой функции $\vec{\nabla} f(\vec{X}^*)$, должен быть коллинеарен вектору нормали к допустимой кривой $\vec{\nabla} g(\vec{X}^*)$, т. е.

$$\vec{\nabla} f(\vec{X}^*) = \lambda \vec{\nabla} g(\vec{X}^*),$$

где λ — произвольный множитель (множитель Лагранжа). На рисунке нормали направлены в разные стороны, поэтому множитель отрицателен.

В многомерном случае линии превращаются в поверхности, однако условие коллинеарности векторов нормалей остается в силе. Таким образом, необходимым условием экстремума является одновременное выполнение двух соотношений:

$\vec{\nabla} f(\vec{X}) - \lambda \vec{\nabla} g(\vec{X}) = 0$ — касание с изоцелевой поверхностью;

$g(\vec{X}) = 0$ — нахождение на допустимой поверхности.

Если ограничивающих уравнений не одно, а много, то данные

соотношения должны выполняться для каждого из них:

$$\overrightarrow{\nabla} f(\overrightarrow{X}) - \lambda_i \overrightarrow{\nabla} g_i(\overrightarrow{X}) = 0, \quad i = 1, \ldots, m, \tag{9.21}$$
$$g_i(\overrightarrow{X}) = 0, \quad i = 1, \ldots, m.$$

Складывая равенства (9.21) и переобозначая неопределенные множители через $y_i = -\dfrac{\lambda_i}{m}$, получаем необходимые условия экстремума

$$\overrightarrow{\nabla} f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i \overrightarrow{\nabla} g_i(\overrightarrow{X}) = 0; \tag{9.22}$$

$$g_i(\overrightarrow{X}) = 0, \quad i = 1, \ldots, m. \tag{9.23}$$

Вспоминая определение функции Лагранжа (9.18), замечаем, что выражение (9.22) может быть переписано в виде

$$\overrightarrow{\nabla} f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i \overrightarrow{\nabla} g_i(\overrightarrow{X}) = \overrightarrow{\nabla} \left(f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i g_i(\overrightarrow{X}) \right) =$$
$$= \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) = \left(\frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_1}, \ldots, \frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_n} \right)^T = 0,$$

что эквивалентно (9.19).

Точно так же в силу линейности функции Лагранжа по y_i выражение (9.23) может быть представлено как

$$g_i(\overrightarrow{X}) = \frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial y_i} = 0, \quad i = 1, \ldots, m,$$

что эквивалентно (9.20).

Пример. На рис. 9.13 представлена задача минимизации функции двух переменных $f(x_1, x_2) = x_1^2 + (x_2 + 1)^2 \to \min$ при условии $2x_1 + x_2 = 2$.

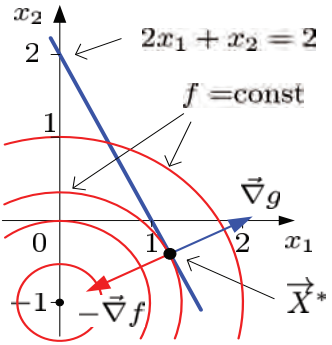


Рис. 9.13. Пример условной минимизации с ограничением в виде равенства

Функция Лагранжа для этой задачи

$$L(x_1, x_2, y) = x_1^2 + (x_2 + 1)^2 + y(2x_1 + x_2 - 2).$$

Необходимые условия экстремума:

$$1) \quad \frac{\partial L}{\partial x_1} = 2x_1 + 2y = 0,$$
$$2) \quad \frac{\partial L}{\partial x_2} = 2x_2 + 2 + y = 0,$$
$$3) \quad \frac{\partial L}{\partial y} = 2x_1 + x_2 - 2 = 0.$$

Решая систему уравнений, получаем $x_1 = 1.2$, $x_2 = -0.4$, $y = -1.2$.

Поскольку функции $f(x_1, x_2)$ и $g(x_1, x_2)$ выпуклые, то приведенные выше необходимые условия являются достаточными для того, чтобы найденная стационарная точка действительно была точкой условного минимума целевой функции. ▲

9.4. Теорема Куна — Таккера

Теорема Куна — Таккера играет ключевую роль в теории выпуклого программирования. Развивая классическую теорию оптимизации с ограничениями в виде равенств, она устанавливает необходимые и достаточные условия оптимальности для общей задачи (9.1):

$$\begin{aligned} &f(\overrightarrow{X}) \to \min, \\ &g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m, \\ &\overrightarrow{X} \geqslant 0. \end{aligned}$$

Как и в классической теории, условия оптимальности формулируются с помощью функции Лагранжа (9.18)

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i g_i(\overrightarrow{X}).$$

Для доказательства теоремы нам понадобится *условие регулярности (условие Слейтера)*, которое заключается в том, что допустимая область имеет хотя бы одну внутреннюю точку, т. е.

$$\exists \overrightarrow{X} \quad g_i(\overrightarrow{X}) < 0, \quad i = 1, \ldots, m.$$

Теорема *(Куна — Таккера). При соблюдении условия Слейтера вектор* $\overrightarrow{X}^* \geqslant 0$ *является оптимальным планом задачи выпуклого программирования тогда и только тогда, когда существует вектор* $\overrightarrow{Y}^* \geqslant 0$*, такой, что точка* $(\overrightarrow{X}^* \overrightarrow{Y}^*)$ *является седловой точкой функции Лагранжа в неотрицательном октанте, т. е.*

$$\forall \overrightarrow{X} \geqslant 0, \overrightarrow{Y} \geqslant 0 \quad L(\overrightarrow{X}^*, \overrightarrow{Y}) \leqslant L(\overrightarrow{X}^*, \overrightarrow{Y}^*) \leqslant L(\overrightarrow{X}, \overrightarrow{Y}^*).$$

Доказательство необходимости. Пусть $\overrightarrow{X}^*$ — оптимальный план. Докажем, что существует вектор $\overrightarrow{Y}^* \geqslant 0$, такой, что для $(\overrightarrow{X}, \overrightarrow{Y})$ выполняется условие седловой точки.

Построим $(m+1)$-мерное $(i = 1, \ldots, m)$ евклидово пространство векторов $\overrightarrow{Z} = (z_0, z_1, \ldots, z_m)^T$ и определим в нем два множества:

$$K_1 = \{\overrightarrow{Z} \mid \exists \overrightarrow{X} \geqslant 0 : \ z_0 \geqslant f(\overrightarrow{X}), \ z_i \geqslant g_i(\overrightarrow{X})\},$$
$$K_2 = \{\overrightarrow{Z} \mid z_0 < f(\overrightarrow{X}^*), \ z_i < 0\}.$$

По способу построения множества K_1 и K_2 не пересекаются.

Множества K_1 и K_2 выпуклы. Для K_2 это очевидно, так как оно является пересечением конечного числа полупространств, а выпуклость K_1 доказывается следующим образом. Пусть $\overrightarrow{A} = (a_0, a_1, \ldots, a_m)^T$ и $\overrightarrow{B} = (b_0, b_1, \ldots, b_m)^T$ — два произвольных вектора из K_1. Тогда по способу построения этого множества найдутся два вектора $\overrightarrow{X}_A \geqslant 0$, $\overrightarrow{X}_B \geqslant 0$, что

$$a_0 \geqslant f(\overrightarrow{X}_A), \ a_i \geqslant g_i(\overrightarrow{X}_A),$$
$$b_0 \geqslant f(\overrightarrow{X}_B), \ b_i \geqslant g_1(\overrightarrow{X}_B).$$

Образуем произвольную выпуклую комбинацию

$$\overrightarrow{C} = (1-\lambda)\overrightarrow{A} + \lambda\overrightarrow{B} =$$
$$= \big((1-\lambda)a_0 + \lambda b_0, \ldots, (1-\lambda)a_m + \lambda b_m\big)^T = (c_0, c_1, \ldots, c_m)^T$$

и покажем, что $C \in K_1$. Для этого достаточно показать, что существует $\overrightarrow{X}_C \geqslant 0$, для которого

$$c_0 \geqslant f(\overrightarrow{X}_C), \ c_i \geqslant g_i(\overrightarrow{X}_C).$$

Возьмем в качестве $\overrightarrow{X}_C = (1-\lambda)\overrightarrow{X}_A + \lambda\overrightarrow{X}_B$. Этот вектор неотрицателен и для него в силу выпуклости $f(\overrightarrow{X})$ и $g_i(\overrightarrow{X})$

$$f(\overrightarrow{X}_C) \leqslant (1-\lambda)f(\overrightarrow{X}_A) + \lambda f(\overrightarrow{X}_B) \leqslant (1-\lambda)a_0 + \lambda b_0 = c_0,$$
$$g_i(\overrightarrow{X}_C) \leqslant (1-\lambda)g_i(\overrightarrow{X}_A) + \lambda g_i(\overrightarrow{X}_B) \leqslant (1-\lambda)a_i + \lambda b_i = c_i.$$

Таким образом, $\overrightarrow{C} \in K_1$ и выпуклость этого множества доказана.

По теореме об отделимости выпуклых множеств существует разделяющая их гиперплоскость, т. е.

$$\exists \overrightarrow{\alpha} \quad \forall \overrightarrow{Z}_1 \in K_1, \overrightarrow{Z}_2 \in K_2 \quad \overrightarrow{\alpha}^T \overrightarrow{Z}_1 \geqslant \alpha^T \overrightarrow{Z}_2. \tag{9.24}$$

Это неравенство остается справедливым и тогда, когда $\overrightarrow{Z}_2$ лежит на границе K_2. При этом, так как координаты векторов в K_2 могут принимать сколь угодно большие отрицательные значения, $\overrightarrow{\alpha} \geqslant 0$.

Возьмем в качестве $\overrightarrow{Z}_1$ и $\overrightarrow{Z}_2$ граничные точки

$$\overrightarrow{Z}_1 = \Big(f(\overrightarrow{X}), g_1(\overrightarrow{X}), \ldots, g_m(\overrightarrow{X})\Big)^T, \quad \overrightarrow{Z}_2 = \Big(f(\overrightarrow{X}^*), 0, \ldots, 0\Big)^T$$

и подставим их координаты в (9.24):

$$\alpha_0 f(\overrightarrow{X}) + \sum_{i=1}^{m} \alpha_i g_i(\overrightarrow{X}) \geqslant \alpha_0 f(\overrightarrow{X}^*). \tag{9.25}$$

Легко убедиться, что $\alpha_0 > 0$. В противном случае, положив в (9.25) $\alpha_0 = 0$, получаем

$$\forall \overrightarrow{X} \quad \sum_{i=1}^{m} \alpha_i g_i(\overrightarrow{X}) \geqslant 0,$$

что противоречит условию Слейтера.

Поделим (9.25) на α_0 и обозначим $y_i^* = \dfrac{\alpha_i}{\alpha_0}$. Таким образом, убедились, что существует вектор $\overrightarrow{Y}^* = (y_1^*, \ldots, y_m^*)^T \geqslant 0$. Покажем, что для него выполняется условие седловой точки.

Неравенство (9.25) должно выполняться для любых $\overrightarrow{X}$. В частности, положив $\overrightarrow{X} = \overrightarrow{X}^*$, получаем

$$f(\overrightarrow{X}^*) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*) \geqslant f(\overrightarrow{X}^*) \quad \text{или, сокращая,} \quad \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*) \geqslant 0.$$

Но так как $\overrightarrow{X}^*$ – план, для которого $g_i(\overrightarrow{X}^*) \leqslant 0$, то

$$\sum_{i=1}^{m} \underbrace{y_i^*}_{\geqslant 0} \underbrace{g_i(\overrightarrow{X}^*)}_{\leqslant 0} \leqslant 0.$$

Отсюда

$$\sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*) = 0. \tag{9.26}$$

Прибавим (9.26) к правой части (9.25):

$$f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}) \geqslant f(\overrightarrow{X}^*) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*).$$

Получили правую часть условия седловой точки.

Теперь докажем левую часть условия седловой точки

$$L(\overrightarrow{X}^*, Y) \leqslant L(\overrightarrow{X}^* \overrightarrow{Y}^*).$$

Действительно, расписав функцию Лагранжа, имеем

$$f(\overrightarrow{X}^*) + \sum_{i=1}^{m} \underbrace{y_i}_{\geqslant 0} \underbrace{g_i(\overrightarrow{X}^*)}_{\leqslant 0} \leqslant f(\overrightarrow{X}^*) + \underbrace{\sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*)}_{0}.$$

Сокращая $f(\overrightarrow{X}^*)$, получаем очевидное неравенство. ■

Доказательство достаточности. Пусть $\overrightarrow{X}^* \geqslant 0, \overrightarrow{Y}^* \geqslant 0$ — седловая точка функции Лагранжа на неотрицательном октанте. Покажем, что $\overrightarrow{X}^*$ — оптимальный план.

Распишем условие седловой точки: $\forall \overrightarrow{X} \geqslant 0, \overrightarrow{Y} \geqslant 0$,

$$f(\overrightarrow{X}^*) + \sum_{i=1}^{m} y_i g_i(\overrightarrow{X}^*) \leqslant f(\overrightarrow{X}^*) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*) \leqslant f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}).$$

Возьмем левое неравенство и сократим его на $f(\overrightarrow{X}^*)$. Получим

$$\sum_{i=1}^{m} y_i g_i(\overrightarrow{X}^*) \leqslant \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*). \tag{9.27}$$

Так как $y_i^* \geqslant 0$, а неравенство должно выполняться для всех $y_i \geqslant 0$, то

$$g_i(\overrightarrow{X}^*) \leqslant 0,$$

т. е. $\overrightarrow{X}^*$ — план задачи. В частности, (9.27) должно выполняться для $y_i = 0$, т. е.

$$0 \leqslant \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*).$$

Но

$$\sum_{i=1}^{m} \underbrace{y_i^*}_{\geqslant 0} \underbrace{g_i(\overrightarrow{X}^*)}_{\leqslant 0} \leqslant 0.$$

Отсюда немедленно следует, что

$$\sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*) = 0.$$

Покажем, что $\overrightarrow{X}^*$ — оптимальный план. Возьмем правое неравенство:

$$\forall \overrightarrow{X} \geqslant 0 \quad f(\overrightarrow{X}^*) + \underbrace{\sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}^*)}_{0} \leqslant f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i^* g_i(\overrightarrow{X}).$$

Отсюда, если $\overrightarrow{X} \geqslant 0$ — план, то $g_i(\overrightarrow{X}) \leqslant 0$ и

$$\sum_{i=1}^{m} \underbrace{y_i^*}_{\geqslant 0} \underbrace{g_i(\overrightarrow{X})}_{\leqslant 0} \leqslant 0.$$

То есть для любого плана $\overrightarrow{X}$ $f(\overrightarrow{X}^*) \leqslant f(\overrightarrow{X})$, следовательно, план $\overrightarrow{X}^*$ оптимальный. ■

Замечание. Более подробное исследование, которое мы опускаем, показывает, что условие Слейтера является необходимым только для н е л и н е й н ы х ограничений. Для линейных ограничений теорема справедлива без этого условия.

9.5. Дифференциальные условия Куна — Таккера и их геометрическая интерпретация

При доказательстве теоремы Куна — Таккера мы нигде не использовали свойства дифференцируемости участвующих функций, требовалась только их выпуклость. Однако если предположить, что они являются гладкими, т. е. непрерывно дифференцируемыми, то условие седловой точки функции Лагранжа $L(\overrightarrow{X}, \overrightarrow{Y})$ на неотрицательном октанте можно представить в дифференциальной форме.

Минимизация L по переменным x_j при $x_j \geqslant 0$ приводит, как мы видели, рассматривая «полуклассический» случай оптимизации, к трем условиям для каждого $j = 1, \ldots, n$. Аналогичные условия получаются для каждого $i = 1, \ldots, m$ при максимизации L по множителям Лагранжа y_i при условии $y_i \geqslant 0$, однако знаки производных при переходе от минимизации к максимизации следует сменить на обратные.

В совокупности получится шесть симметричных дифференциальных условий Куна — Таккера, которые являются необходимыми и достаточными условиями минимума для общей задачи выпуклого программирования (9.1):

$$\begin{aligned}
&\text{а)}\quad \frac{\partial L(\overrightarrow{X},\overrightarrow{Y})}{\partial x_j}\geqslant 0, && \text{г)}\quad \frac{\partial L(\overrightarrow{X},\overrightarrow{Y})}{\partial y_i}\leqslant 0,\\
&\text{б)}\quad x_j\frac{\partial L(\overrightarrow{X},\overrightarrow{Y})}{\partial x_j}=0, && \text{д)}\quad y_i\frac{\partial L(\overrightarrow{X},\overrightarrow{Y})}{\partial y_i}=0,\\
&\text{в)}\quad x_j\geqslant 0, && \text{е)}\quad y_i\geqslant 0,\\
&\qquad j=1,\ldots,n, && \qquad i=1,\ldots,m.
\end{aligned}\tag{9.28}$$

Равенства «б» и «д» обычно называют *условиями дополняющей нежесткости (complementary slackness conditions).*

В векторной форме условия «а»–«в» приобретают компактный вид

$$\begin{aligned}
&\text{а)}\quad \overrightarrow{\nabla} L(\overrightarrow{X},\overrightarrow{Y})\geqslant 0,\\
&\text{б)}\quad \overrightarrow{X}^T\overrightarrow{\nabla} L(\overrightarrow{X},\overrightarrow{Y})=0,\\
&\text{в)}\quad \overrightarrow{X}\geqslant 0,
\end{aligned}\tag{9.28′}$$

а условия «г»–«е», учитывая линейность функции Лагранжа по переменным y_i, иногда удобнее использовать в равнозначной покоординатной записи:

$$\begin{aligned}
&\text{г)}\quad g_i(\overrightarrow{X})\leqslant 0,\\
&\text{д)}\quad y_i g_i(\overrightarrow{X})=0,\\
&\text{е)}\quad y_i\geqslant 0,\\
&\qquad i=1,\ldots,m.
\end{aligned}\tag{9.28″}$$

Для того чтобы компактно записать условия «г»–«е», введем обозначение *вектор-функции*:

$$\overrightarrow{G}(\overrightarrow{X})=\begin{pmatrix} g_1(\overrightarrow{X})\\ \vdots\\ g_m(\overrightarrow{X})\end{pmatrix}.$$

Тогда (9.28″) перепишется в эквивалентном векторном виде

$$\begin{aligned} &\text{г)} \quad \overrightarrow{G}(\overrightarrow{X}) \leqslant 0, \\ &\text{д)} \quad \overrightarrow{Y}^T \overrightarrow{G}(\overrightarrow{X}) = 0, \\ &\text{е)} \quad \overrightarrow{Y} \geqslant 0. \end{aligned} \tag{9.29}$$

Геометрическая интерпретация

Условия Куна — Таккера имеют простую и наглядную геометрическую интерпретацию.

Напомним еще раз общую задачу выпуклого программирования (9.1)

$$\begin{aligned} &f(\overrightarrow{X}) \to \min, \\ &g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m, \\ &\overrightarrow{X} \geqslant 0, \end{aligned}$$

и функцию Лагранжа для нее (9.18)

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \sum_{i=1}^{m} y_i g_i(\overrightarrow{X}).$$

Итак, пусть $\overrightarrow{X}^* = (x_1^*, \ldots, x_n^*)^T$ — оптимальный план, $\overrightarrow{Y}^* = (y_1^*, \ldots, y_m^*)^T$ — соответствующий ему вектор оптимальных множителей Лагранжа.

Представим градиент функции Лагранжа в точке $\overrightarrow{X}^*$ в виде разложения по координатным векторам:

$$\overrightarrow{\nabla} L(\overrightarrow{X}^*, \overrightarrow{Y}^*) = \overrightarrow{\nabla} f(\overrightarrow{X}^*) + \sum_{i=1}^{m} y_i^* \overrightarrow{\nabla} g_i(\overrightarrow{X}^*) = \sum_{j=1}^{n} v_j^* \overrightarrow{e}_j.$$

Здесь v_j^* — координата градиента, а $\overrightarrow{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)^T$, где единица стоит на j-м месте, — единичный координатный вектор (орт).

Выразим отсюда антиградиент ц е л е в о й функции в точке $\overrightarrow{X}^*$:

$$-\overrightarrow{\nabla} f(\overrightarrow{X}^*) = \sum_{i=1}^{m} y_i^* \overrightarrow{\nabla} g_i(\overrightarrow{X}^*) + \sum_{j=1}^{n} v_j^* (-\overrightarrow{e}_j). \qquad (9.30)$$

С учетом введенных обозначений условия Куна — Таккера (9.28)–(9.28″) приобретают вид

$$\begin{array}{ll} \text{а)} \quad v_j^* \geqslant 0, & \text{г)} \quad g_i(\overrightarrow{X}^*) \leqslant 0, \\ \text{б)} \quad x_j^* v_j^* = 0, & \text{д)} \quad y_i^* g_i(\overrightarrow{X}^*) = 0, \\ \text{в)} \quad x_j^* \geqslant 0, & \text{е)} \quad y_i^* \geqslant 0, \\ j = 1, \ldots, n, & i = 1, \ldots, m. \end{array}$$

Рассматривая их, прежде всего заметим, что условия «в» и «г» очевидны: они требуют, чтобы оптимальный план $\overrightarrow{X}^*$ удовлетворял всем ограничениям задачи, т. е. находился в допустимой области.

Для интерпретации остальных условий напомним, что некоторое ограничение является активным для конкретного плана, если соответствующее нестрогое неравенство превращается в равенство, то есть если данный план попадает на границу допустимой области.

В нашей задаче план $\overrightarrow{X}$ может активизировать ограничения по двум причинам: либо он попадает на границу, заданную одной или несколькими функциями $g_i(\overrightarrow{X}) = 0$, либо он попадает на границу неотрицательного октанта, при этом одна или несколько координат x_j нулевые. В связи с этим для оптимального плана $\overrightarrow{X}^*$ введем два множества индексов а к т и в н ы х ограничений

$$M(\overrightarrow{X}^*) = \{i \mid g_i(\overrightarrow{X}^*) = 0\},$$
$$N(\overrightarrow{X}^*) = \{j \mid x_j^* = 0\}.$$

Условия дополняющей нежесткости «д» и «б» означают, что для н е а к т и в н ы х (пассивных) ограничений соответствующие

им y_i^* и v_j^* равны нулю:

$$\forall i \notin M(\overrightarrow{X}^*) \ \rightarrow \ g_i(\overrightarrow{X}^*) < 0 \ \rightarrow \ y_i^* = 0; \tag{9.31}$$

$$\forall j \notin N(\overrightarrow{X}^*) \ \rightarrow \ x_j^* > 0 \ \rightarrow \ v_j^* = 0. \tag{9.32}$$

С учетом этого в выражении (9.30) суммирование по всем индексам можно заменить суммированием по индексам активных ограничений, так как остальные слагаемые, соответствующие неактивным ограничениям, равны нулю. Таким образом, условия «д» и «б» эквиваленты следующему выражению:

$$-\overrightarrow{\nabla} f(\overrightarrow{X}^*) = \sum_{i \in M(\overrightarrow{X}^*)} y_i^* \overrightarrow{\nabla} g_i(\overrightarrow{X}^*) + \sum_{j \in N(\overrightarrow{X}^*)} v_j^* (-\overrightarrow{e}_j). \tag{9.33}$$

Оставшиеся два условия «а» и «е» требуют неотрицательности входящих в разложение коэффициентов v_j^* и y_i^*.

Для того чтобы увидеть в выражении (9.33) наглядный геометрический смысл, заметим, что вектор $\overrightarrow{\nabla} g_i(\overrightarrow{X}^*)$ представляет собой внешнюю нормаль к ограничению $g_i(\overrightarrow{X}) = 0$ в точке $\overrightarrow{X}^*$, а вектор $-\overrightarrow{e}_j$ также можно понимать как внешнюю нормаль к координатной прямой $x_j = 0$.

Подводя итог, получаем следующую интерпретацию условий Куна — Таккера для задачи выпуклого программирования.

Геометрический критерий оптимальности. *В оптимальной точке антиградиент целевой функции представляет собой неотрицательную линейную комбинацию векторов внешних нормалей всех активных ограничений, т. е. антиградиент целевой функции должен попасть внутрь конуса, натянутого на векторы внешних нормалей к активным ограничениям.*

Из этой интерпретации следует и геометрический смысл множителей Лагранжа y_i: они представляют собой коэффициенты разложения антиградиента по векторам внешних нормалей к активным ограничениям. Если некоторое ограничение неактивно,

то, как следует из (9.31), соответствующий ему множитель равен нулю.

П р и м е р 1. Для иллюстрации рассмотрим простую задачу выпуклого программирования с двумя переменными :

$$f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min,$$
$$x_1^2 + x_2^2 \leqslant 4,$$
$$x_2 \leqslant x_1,$$
$$x_2 \leqslant 1,$$
$$x_1, x_2 \geqslant 0.$$

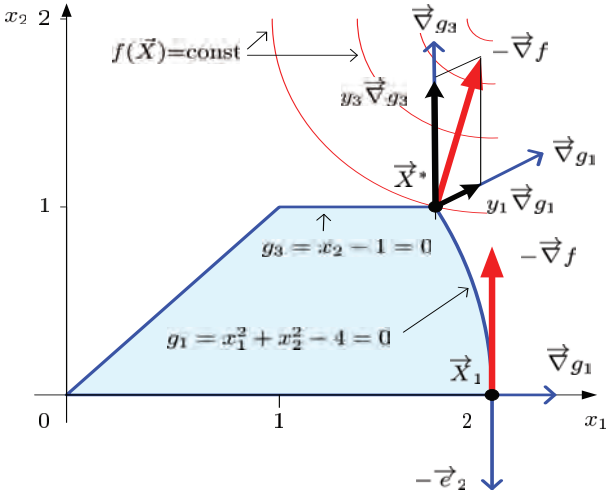


Рис. 9.14. Минимизация квадратичной функции двух неотрицательных переменных с тремя ограничениями-неравенствами

Как видно из рис. 9.14, допустимое множество (множество планов) в этой задаче представляет собой криволинейную трапецию, оптимальный план $\vec{X}_1$ находится в правом верхнем ее углу.

Координаты этой точки находятся из решения системы уравнений

$$\begin{cases} x_1^2 + x_2^2 = 4, \\ x_2 = 1, \end{cases}$$

откуда $x_1^* = \sqrt{3},\ x_2^* = 1,\ f(x_1^*, x_2^*) \approx 1.0718$.

Для записи условий Куна — Таккера приведем ограничения к каноническому виду:

$$\begin{aligned} g_1(x_1, x_2) &= x_1^2 + x_2^2 - 4 \leqslant 0, \\ g_2(x_1, x_2) &= -x_1 + x_2 \leqslant 0, \\ g_3(x_1, x_2) &= x_2 - 1 \leqslant 0. \end{aligned}$$

Активные ограничения исследуемой точки: $M(\overrightarrow{X}^*) = \{1, 3\}$, $N(\overrightarrow{X}^*) = \emptyset$. Градиенты целевой и ограничивающих функций в этой точке:

$$-\overrightarrow{\nabla} f(\overrightarrow{X}) = \begin{pmatrix} -2(x_1 - 2) \\ -2(x_2 - 2) \end{pmatrix}, \quad -\overrightarrow{\nabla} f(\overrightarrow{X}^*) = \begin{pmatrix} 4 - 2\sqrt{3} \\ 2 \end{pmatrix},$$

$$\overrightarrow{\nabla} g_1(\overrightarrow{X}) = \begin{pmatrix} 2x_1 \\ 2x_2 \end{pmatrix}, \quad \overrightarrow{\nabla} g_1(\overrightarrow{X}^*) = \begin{pmatrix} 2\sqrt{3} \\ 2 \end{pmatrix},$$

$$\overrightarrow{\nabla} g_3(\overrightarrow{X}) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \overrightarrow{\nabla} g_3(\overrightarrow{X}^*) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Коэффициенты разложения антиградиента по нормалям к активным ограничениям находятся из системы уравнений

$$y_1 \begin{pmatrix} 2\sqrt{3} \\ 2 \end{pmatrix} + y_3 \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 - 2\sqrt{3} \\ 2 \end{pmatrix},$$

откуда

$$y_1 = \frac{2\sqrt{3} - 3}{3} = 0.1547 > 0, \quad y_3 = \frac{12 - 4\sqrt{3}}{6} = 1.6906 > 0.$$

Они неотрицательны, на рис. 9.14 это хорошо видно, следовательно, план $\overrightarrow{X}^*$ действительно оптимальный. ▲

Пример 2. В качестве контрпримера рассмотрим заведомо неоптимальную точку $\overrightarrow{X}_1 = (2,0)^T$, расположенную в правом нижнем углу множества планов.

Множества активных ограничений для нее: $M(\overrightarrow{X}_1) = \{1\}$, $N(\overrightarrow{X}_1) = \{2\}$. Антиградиент и нормали в данной точке:

$$-\overrightarrow{\nabla} f(\overrightarrow{X}_1) = \begin{pmatrix} 0 \\ 4 \end{pmatrix}, \quad \overrightarrow{\nabla} g_1(\overrightarrow{X}_1) = \begin{pmatrix} 4 \\ 0 \end{pmatrix}, \quad -\overrightarrow{e}_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

Коэффициенты разложения по нормалям к активным ограничениям находятся из системы уравнений

$$y_1 \begin{pmatrix} 4 \\ 0 \end{pmatrix} + v_2 \begin{pmatrix} 0 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \end{pmatrix},$$

откуда $y_1 = 0$, $v_2 = -4$. Коэффициенты разложения неположительны, план неоптимальный. ▲

Замечание. Кроме геометрической, условия оптимальности Куна — Таккера имеют простую и наглядную физическую интерпретацию. Представим себе шарик (рис. 9.15), скатывающийся под действием силы $-\overrightarrow{\nabla} f(\overrightarrow{X})$, которую мы интерпретируем как силу тяжести, по поверхности, образованной гладкими ограничениями $g_i(\overrightarrow{X})$. Согласно Третьему закону Ньютона, на шарик действуют силы реакции поверхностей, которых он данный момент касается (т. е. активных ограничений). Эти силы ортогональны поверхностям $g_i(\overrightarrow{X})$ и направлены внутрь допустимой области, стало быть, противоположны внешним нормалям $\overrightarrow{\nabla} g_i(\overrightarrow{X})$.

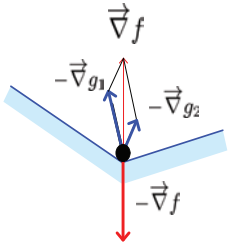


Рис. 9.15.

Шарик достигнет минимума и остановится в стационарной точке, где сумма всех сил, действующих на него, равна нулю:

$$-\overrightarrow{\nabla} f(\overrightarrow{X}^*) - y_1^* \overrightarrow{\nabla} g_1(\overrightarrow{X}^*) - y_2^* \overrightarrow{\nabla} g_2(\overrightarrow{X}^*) = 0,$$

т. е. там, где вектор антиградиента будет уравновешен равнодействующей сил реакции активных ограничений. При этом коэффициенты разложения $y_1^*, y_2^* \geqslant 0$.

9.6. Частные случаи

Если задача выпуклого программирования имеет общий вид

$$\begin{aligned} &f(x_1, \ldots, x_n) \to \min, \\ &g_i(x_1, \ldots, x_n) \leqslant 0, \quad i = 1, \ldots, m, \\ &x_1, \ldots x_n \geqslant 0, \quad j = 1, \ldots, n, \end{aligned}$$

то для нее требуются все шесть, точнее $3m + 3n$ условий Куна—Таккера (9.28):

$$\begin{array}{ll} \text{а)} \quad \dfrac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_j} \geqslant 0, & \text{г)} \quad \dfrac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial y_i} \leqslant 0, \\ \text{б)} \quad x_j \dfrac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_j} = 0, & \text{д)} \quad y_i \dfrac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial y_i} = 0, \\ \text{в)} \quad x_j \geqslant 0, & \text{е)} \quad y_i \geqslant 0, \\ \quad j = 1, \ldots, n, & \quad i = 1, \ldots, m. \end{array}$$

Если же в задаче имеются упрощения, то некоторые условия оказываются излишними, происходит естественный переход к классическим постановкам задач на условный экстремум.

1. Неограниченность переменных по знаку. Пусть некоторые переменные x_j неограничены по знаку: $-\infty < x_j < \infty$. В таком случае для этих переменных условия «а», «б», «в» минимума функции Лагранжа по x_j в неотрицательном октанте заменяются на одно классическое условие минимума на всей числовой оси

$$\frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial x_j} = 0, \quad j = 1, \ldots, n.$$

Если все переменные неограничены по знаку, то условия «а», «б», «в» можно записать в краткой векторной форме[4]

$$\overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) = 0. \tag{9.34}$$

2. Ограничения-равенства. Если некоторые ограничения записаны в виде классических равенств $g_i(\overrightarrow{X}) = 0$, то соответствующие им условия также могут быть упрощены. Действительно, каждое такое равенство эквивалентно двум неравенствам

$$g_i(\overrightarrow{X}) \leqslant 0,$$
$$-g_i(\overrightarrow{X}) \leqslant 0.$$

Обозначив соответствующие этим ограничениям множители Лагранжа через y_i' и y_i'', получаем функцию Лагранжа в виде

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \sum (y_i' - y_i'') g_i(\overrightarrow{X}).$$

То есть для этих ограничений каждую пару неотрицательных переменных y_i', y_i'' можно заменить одним множителем $y_i = y_i' - y_i''$, неограниченным по знаку. Соответственно условия «г», «д», «е» максимума функции Лагранжа по y_i в неотрицательном октанте заменяются на одно классическое условие максимума на всей числовой оси

$$\frac{\partial L(\overrightarrow{X}, \overrightarrow{Y})}{\partial y_i} = g_i(\overrightarrow{X}) = 0.$$

Если все ограничения имеют вид строгого равенства, то, пользуясь обозначением вектор-функции, это можно записать в виде одного равенства $\overrightarrow{G}(\overrightarrow{X}) = 0$.

3. Классическая условная задача. Объединяя два предыдущих случая, можно записать в компактной векторной форме

[4]Хотя функция Лагранжа зависит от двух векторных переменных, знаком $\overrightarrow{\nabla}$ мы будем обозначать вектор производных по $\overrightarrow{X}$.

необходимые условия оптимальности (уравнения Куна — Таккера) для классической задачи минимизации с неограниченными по знаку переменными и ограничениями-равенствами:

$$\begin{aligned} \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) &= 0, \\ \overrightarrow{G}(\overrightarrow{X}) &= 0. \end{aligned} \tag{9.35}$$

П р и м е р. Вернемся к примеру на с. 45 с минимизацией функции $f(x_1, x_2) = x_1^2 + (x_2+1)^2 \to \min$ при условии $2x_1 + x_2 = 2$ и добавим еще одно «полуклассическое» ограничение $x_2 > 0$, так что допустимая область теперь представляет собой полупрямую (рис. 9.16).

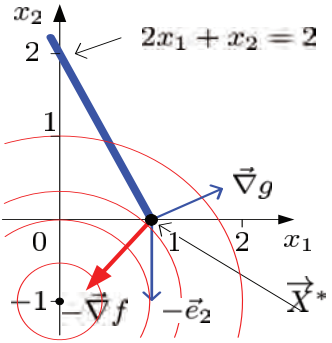


Рис. 9.16. Пример условной минимизации с полуклассическими ограничениями

Прежде чем сформулировать необходимые условия экстремума, заметим, что, хотя условие Слейтера для данной задачи не выполняется (прямая не имеет внутренних точек), ограничивающая функция является линейной, поэтому теорема Куна — Таккера и следующие из нее дифференциальные условия сохраняют силу.

Функция Лагранжа для этой задачи остается прежней:

$$L(x_1, x_2, y) = x_1^2 + (x_2 + 1)^2 + y(2x_1 + x_2 - 2).$$

Начнем выписывать необходимые условия экстремума. Поскольку переменная x_1 неограничена по знаку, условия «а», «б», «в» для нее превращаются в одно условие равенства нулю частной производной. Для переменной x_2 потребуются все три условия, в итоге получаем четыре соотношения:

$$\begin{aligned}
&1) \quad \frac{\partial L}{\partial x_1} = 2x_1 + 2y = 0,\\
&2) \quad \frac{\partial L}{\partial x_2} = 2x_2 + 2 + y \geqslant 0,\\
&3) \quad x_2 \frac{\partial L}{\partial x_2} = x_2(2x_2 + 2 + y) = 0,\\
&4) \quad x_2 \geqslant 0.
\end{aligned}$$

Так как единственное ограничение имеет вид равенства, соответствующие ему условия «г», «д», «е» превращаются в одно:

$$5) \quad \frac{\partial L}{\partial y} = 2x_1 + x_2 - 2 = 0.$$

Поскольку функции $f(x_1, x_2)$ и $g(x_1, x_2)$ выпуклые, эти необходимые условия условного минимума являются и достаточными.

Для нахождения решения следует решить систему нелинейных уравнений «1», «3», «5» и отобрать те решения, которые удовлетворяют неравенствам «2» и «4».

Эта система эквивалентна двум линейным системам:

$$\text{а)} \begin{cases} 2x_1 + 2y = 0, \\ x_2 = 0, \\ 2x_1 + x_2 = 2; \end{cases} \qquad \text{б)} \begin{cases} 2x_1 + 2y = 0, \\ 2x_2 + 2 + y = 0, \\ 2x_1 + x_2 = 2. \end{cases}$$

Решая «а», получаем $x_1 = 1$, $x_2 = 0$, $y = -1$. Легко видеть, что это решение удовлетворяет неравенствам «2» и «4», следовательно, точка $\overrightarrow{X}^* = (1, 0)^T$ есть искомое решение условной экстремальной задачи (см. рис. 9.16).

Система «б» дает решение $x_1 = \dfrac{6}{5}$, $x_2 = -\dfrac{2}{5}$, $y = -\dfrac{6}{5}$, которое не удовлетворяет неравенствам «2» и «4». ▲

Линейное программирование

Поскольку линейные функции являются выпуклыми, то с формальной точки зрения задача линейного программирования является частным случаем общей задачи выпуклого программирования. Интересно посмотреть, во что превращаются в данном случае дифференциальные условия Куна — Таккера.

Пусть задача линейного программирования задана в канонической форме (в развернутой и матричной записи):

$$\begin{aligned} &f(x_1,\ldots,x_n) = \sum_{j=1}^{n} c_j x_j \to \min, \\ &\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1,\ldots,m, \\ &x_j \geqslant 0, \quad j = 1,\ldots,n, \end{aligned} \qquad \text{или} \qquad \begin{aligned} &f(\overrightarrow{X}) = \overrightarrow{c}^T \overrightarrow{X} \to \min, \\ &A\overrightarrow{X} = \overrightarrow{b}, \\ &\overrightarrow{X} \geqslant 0. \end{aligned}$$

Функция Лагранжа для нее в развернутой записи имеет вид

$$L(\overrightarrow{X}, \overrightarrow{Y}) = \sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{m} y_i \left[-\sum_{j=1}^{n} a_{ij} x_j + b_i \right], \tag{9.36}$$

что можно представить более компактно, используя матричные обозначения:

$$L(\overrightarrow{X}, \overrightarrow{Y}) = \overrightarrow{c}^T \overrightarrow{X} + \overrightarrow{Y}^T(-A\overrightarrow{X} + \overrightarrow{b}). \tag{9.37}$$

Компоненты градиента функции Лагранжа получаются дифференцированием (9.36):

$$\frac{\partial L}{\partial x_j} = c_j - \sum_{i=1}^{m} y_i a_{ij},$$

откуда

$$\overrightarrow{\nabla} L = \left(\frac{\partial L}{\partial x_1}, \ldots, \frac{\partial L}{\partial x_n} \right)^T = \overrightarrow{c} - A^T \overrightarrow{Y}.$$

Поскольку все переменные x_j неотрицательны, необходимы все три первых условия Куна — Таккера, которые в матричной записи (9.28′) приобретают вид

$$\begin{aligned} &\text{а)} \quad \overrightarrow{\nabla} L = \overrightarrow{c} - A^T \overrightarrow{Y} \geqslant 0, \\ &\text{б)} \quad \overrightarrow{X}^T \overrightarrow{\nabla} L = \overrightarrow{X}^T \left(\overrightarrow{c} - A^T \overrightarrow{Y} \right) = 0, \\ &\text{в)} \quad \overrightarrow{X} \geqslant 0. \end{aligned} \tag{9.38}$$

Так как ограничения задачи имеют вид равенств, множители Лагранжа по знаку неограничены, при этом условия «г», «д», «е» из (9.28″) превращаются в одно:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i, \quad i = 1, \ldots, m, \qquad \text{или} \qquad A\overrightarrow{X} = \overrightarrow{b}. \tag{9.39}$$

Анализируя набор условий, приходим к выводу, что вектор $\overrightarrow{X}$ является оптимальным планом задачи линейного программирования тогда и только тогда, когда он является планом (условия (9.38, в) и (9.39)), и существует неограниченный по знаку вектор $\overrightarrow{Y}$, такой, что

$$A^T \overrightarrow{Y} \leqslant \overrightarrow{c} \quad \text{(условие (9.38, а))}; \tag{9.40}$$

$$\overrightarrow{X}^T \overrightarrow{c} = \overrightarrow{X}^T A^T \overrightarrow{Y} \quad \text{(условие (9.38, б))}. \tag{9.41}$$

Здесь нужно повторить основы теории двойственности в линейном программировании и вспомнить [9, с. 113], что двойственной по отношению к исходной является задача максимизации целевой функции m переменных $\overrightarrow{Y} = (y_1, \ldots, y_m)^T$

$$M(\overrightarrow{Y}) = \overrightarrow{b}^T \overrightarrow{Y} \to \max$$

при условиях

$$A^T\overrightarrow{Y} \leqslant \overrightarrow{c}, \quad \overrightarrow{Y} — \text{любое}.$$

С учетом этого требование (9.40) означает, что удовлетворяющий условиям Куна — Таккера вектор $\overrightarrow{Y}$ должен быть планом двойственной задачи линейного программирования, для которого выполняется условие (9.41), т. е.

$$f(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} = \boxed{\text{транспонируем скаляр}} = \overrightarrow{X}^T\overrightarrow{c} = \overrightarrow{X}^T A^T\overrightarrow{Y} =$$
$$= \boxed{\text{транспонируем скаляр}} = \overrightarrow{Y}^T A\overrightarrow{X} = \boxed{\text{так как } \overrightarrow{X} — \text{план}} =$$
$$= \overrightarrow{Y}^T\overrightarrow{b} = \boxed{\text{транспонируем скаляр}} = \overrightarrow{b}^T\overrightarrow{Y} = M(\overrightarrow{Y}).$$

Таким образом, мы показали, что в случае линейного программирования теорема Куна — Таккера эквивалентна фундаментальной Первой теореме двойственности, а двойственные переменные — это множители Лагранжа.

9.7. Квадратичное программирование. Метод Вулфа

С точки зрения чистой математики теория Куна — Таккера исчерпывает проблему выпуклого программирования. Для любой конкретной задачи следует составить условия Куна — Таккера и решить получающуюся систему уравнений и неравенств. К сожалению, на практике все обстоит значительно сложнее. Эта система является нелинейной, ее решение представляет не менее сложную задачу, чем непосредственный поиск экстремума.

Счастливым исключением является задача *квадратичного программирования,* которую, применяя теорию Куна — Таккера, ценой существенного повышения размерности удается свести к задаче линейного программирования и найти т о ч н о е решение за конечное число шагов.

Стандартной формой задачи квадратичного программирования будем считать задачу минимизации целевой функции, состоящей из линейной и квадратичной составляющих:

$$f(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} + \overrightarrow{X}^T D\overrightarrow{X} = \\ = \sum_{i=1}^{n} c_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}x_i x_j \to \min, \tag{9.42}$$

при линейных ограничениях-неравенствах

$$A\overrightarrow{X} \leqslant \overrightarrow{b}, \\ \overrightarrow{X} \geqslant 0, \tag{9.43}$$

где $D = (d_{ij})_{n\times n}$ — симметричная неотрицательно определенная матрица.

Запишем функцию Лагранжа для нашей задачи

$$L(\overrightarrow{X}, \overrightarrow{Y}) = \overrightarrow{c}^T\overrightarrow{X} + \overrightarrow{X}^T D\overrightarrow{X} + \overrightarrow{Y}^T(A\overrightarrow{X} - \overrightarrow{b})$$

и введем обозначения

$$\overrightarrow{v} = \left(\frac{\partial L}{\partial x_1}, \ldots, \frac{\partial L}{\partial x_n}\right)^T = \overrightarrow{\nabla} L = \overrightarrow{c} + 2D\overrightarrow{X} + A^T\overrightarrow{Y}; \tag{9.44}$$

$$-\overrightarrow{u} = \left(\frac{\partial L}{\partial y_1}, \ldots, \frac{\partial L}{\partial y_m}\right)^T = A\overrightarrow{X} - \overrightarrow{b}. \tag{9.45}$$

Поскольку ограничения (9.43) заданы в форме неравенств и переменные неотрицательны, потребуются все шесть условий Куна — Таккера (9.28′) и (9.28″). С учетом введенных обозначений они запишутся в виде

$$\begin{array}{lllll} \text{а)} & \overrightarrow{\nabla} L \geqslant 0, & \to & \overrightarrow{v} \geqslant 0, \\ \text{б)} & \overrightarrow{X}^T\overrightarrow{\nabla} L = 0 & \to & \overrightarrow{X}^T\overrightarrow{v} = 0, \\ \text{в)} & \overrightarrow{X} \geqslant 0, & & \\ \text{г)} & g_i(\overrightarrow{X}) \leqslant 0 & \to & \overrightarrow{u} \geqslant 0, \\ \text{д)} & y_i g_i(\overrightarrow{X}) = 0 & \to & \overrightarrow{Y}^T\overrightarrow{u} = 0, \\ \text{е)} & y_i \geqslant 0 & \to & \overrightarrow{Y} \geqslant 0. \end{array} \tag{9.46}$$

Перепишем (9.44) — (9.46) в виде

$$2D\overrightarrow{X} + A^T\overrightarrow{Y} - \overrightarrow{v} = -\overrightarrow{c}; \tag{9.47}$$

$$A\overrightarrow{X} + \overrightarrow{u} = \overrightarrow{b}; \tag{9.48}$$

$$\overrightarrow{X} \geqslant 0, \overrightarrow{Y} \geqslant 0, \overrightarrow{u} \geqslant 0, \overrightarrow{v} \geqslant 0;$$

$$\overrightarrow{X}^T\overrightarrow{v} = 0, \overrightarrow{Y}^T\overrightarrow{u} = 0. \tag{9.49}$$

Таким образом, задача поиска оптимального решения задачи квадратичного программирования сводится к нахождению н е о т р и ц а т е л ь н о г о решения системы линейных уравнений (9.47) – (9.48) относительно переменных $\overrightarrow{X}, \overrightarrow{Y}, \overrightarrow{u}, \overrightarrow{v}$, при этом должно соблюдаться дополнительное комбинаторное условие (9.49).

Размерности векторов $\overrightarrow{X}$ и $\overrightarrow{v}$ равны n, а векторов $\overrightarrow{Y}$ и $\overrightarrow{u}$ — m. Следовательно, система имеет $m+n$ уравнений и $2n+2m$ переменных, при этом комбинаторное условие (9.49) говорит о том, что $m + n$ переменных равны нулю. Следовательно, нужно найти такое неотрицательное частное решение системы $m + n$ линейных уравнений, в котором имеется ровно $m + n$ ненулевых переменных, то есть неотрицательное б а з и с н о е решение. При этом в базисном решении не должны одновременно появляться *сопряженные переменные,* то есть переменные с одноименными индексами x_j, v_j и y_i, v_i.

Нахождение такого решения можно произвести методом искусственного базиса. Для этого введем искусственные переменные

$$\overrightarrow{W}_1 = (w_1, \ldots, w_n),$$
$$\overrightarrow{W}_2 = (w_{n+1}, \ldots, w_{n+m})$$

и поставим задачу минимизации суммы искусственных переменных

$$\sum_{i=1}^{n+m} w_i \to \min$$

при ограничениях

$$\begin{array}{lllll} 2D\overrightarrow{X}+A^T\overrightarrow{Y} & -\overrightarrow{v}+\overrightarrow{W}_1 & = & -\overrightarrow{c}, \\ A\overrightarrow{X} & +\overrightarrow{u} \qquad +\overrightarrow{W}_2 & = & \overrightarrow{b}, \end{array}$$

$$\overrightarrow{X} \geqslant 0, \overrightarrow{Y} \geqslant 0, \overrightarrow{u} \geqslant 0, \overrightarrow{v} \geqslant 0.$$

Решаем эту задачу обычным симплексным методом, с тем лишь ограничением, что в базисе не должны одновременно появляться сопряженные переменные x_j, v_j и y_i, u_i.

П р и м е р. Имеется простейшая задача минимизации квадратичной функции при линейных ограничениях (рис. 9.17):

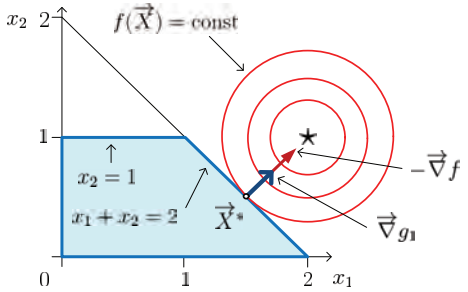


Рис. 9.17. Пример задачи квадратичного программирования

$$\begin{array}{lll} f(x_1, x_2) = & (x_1-2)^2 + (x_2-1)^2 \to \min, \\ & x_1 \quad +x_2 \quad \leqslant 2, \\ & \qquad x_2 \quad \leqslant 1, \\ & x_1, x_2 \geqslant 0. \end{array}$$

Ее решение очевидно: $x_1^* = \dfrac{3}{2}$; $x_2^* = \dfrac{1}{2}$.

Для применения метода Вулфа приведем задачу к стандартному виду 9.42, для чего раскроем скобки в целевой функции:

$$f(x_1, x_2) = (x_1-2)^2 + (x_2-1)^2 = -4x_1 - 2x_2 + x_1^2 + x_2^2 + 5 \to \min.$$

Выпишем участвующие в задаче матрицы и векторы:

$$\overrightarrow{c} = \begin{pmatrix} -4 \\ -2 \end{pmatrix}, \quad D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}, \quad \overrightarrow{b} = \begin{pmatrix} 2 \\ 1 \end{pmatrix},$$

$$\overrightarrow{X} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \quad \overrightarrow{v} = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}, \quad \overrightarrow{Y} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}, \quad \overrightarrow{u} = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

$$\overrightarrow{W}_1 = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \quad \overrightarrow{W}_2 = \begin{pmatrix} w_3 \\ w_4 \end{pmatrix}.$$

Таким образом, всего имеется $3m+3n = 12$ переменных, $m+n = 4$ уравнения.

Задача линейного программирования с искусственными переменными имеет вид

$$\begin{array}{rrrrrrrrrrrrrl}
 & & & & & & & & w_1 & +w_2 & +w_3 & +w_4 & \to \min, \\
2x_1 & & +y_1 & & & & -v_1 & & +w_1 & & & & & = 4, \\
 & 2x_2 & +y_1 & +y_2 & & & & -v_2 & & +w_2 & & & & = 2, \\
x_1 & +x_2 & & & +u_1 & & & & & & +w_3 & & & = 2, \\
 & x_2 & & & & +u_2 & & & & & & +w_4 & & = 1, \\
x_1, & x_2, & y_1, & y_2, & u_1, & u_2, & v_1, & v_2, & w_1, & w_2, & w_3, & w_4 & & \geqslant 0.
\end{array}$$

Исходная симплексная таблица приведена в табл. 9.1.

Для решения потребовалось четыре итерации симплексного метода (показаны включаемые в базис и исключаемые переменные):

$$1)\ \frac{+x_2}{-w_2}; \quad 2)\ \frac{+x_1}{-w_3}; \quad 3)\ \frac{+y_1}{-w_1}; \quad 4)\ \frac{+u_2}{-w_4}.$$

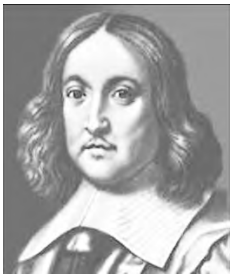
Решение: $x_1^* = \dfrac{3}{2}, \quad x_2^* = \dfrac{1}{2}; \quad y_1^* = 1; \quad u_2^* = \dfrac{1}{2}.$ ▲

Таблица 9.1. Исходная симплексная таблица для примера. Единичный искусственный базис опущен

Базис	План	x_1	x_2	y_1	y_2	u_1	u_2	v_1	v_2
w_1	4	2	0	1	0	0	0	-1	0
w_2	2	0	2	1	1	0	0	0	-1
w_3	2	1	1	0	0	1	0	0	0
w_4	1	0	1	0	0	0	1	0	0
	9	3	4	2	1	1	1	-1	-1

9.8. Исторические замечания

Классические методы нахождения экстремумов функций без ограничений были созданы в XVII веке и связаны с именами великих математиков Фермá, Лейбница, Ньютона.



Пьер Ферма

Первый аналитический способ нахождения максимумов и минимумов был найден французом **Пьером Фермá** (Fermat, Pierre; 1601–1665). По профессии он был юристом, слыл знатоком классической литературы и лингвистики. Математика всегда была для Ферма лишь увлечением, и тем не менее он заложил основы многих ее областей. Автор выдающихся трудов в области математического анализа, алгебры, геометрии, теории чисел. Вместе со своим великим современником — философом и математиком — Рене Декартом (Descartes, René; 1596–1650) разработал метод координат. Над сформулированной им в 1630 г. Великой теоремой математики бились более 350 лет, она была доказана только в 1994 г.

Во времена Ферма еще не было понятия производной, поэтому описанный им в 1636 г. метод нахождения максимумов и ми-

нимумов, использующий свойства касательной, работал только для полиномов. Общий подход, основанный на дифференцировании произвольных функций, появился несколько десятилетий спустя, когда трудами Лейбница и Ньютона, живших по обе стороны Ла-Манша, независимо друг от друга, было основано дифференциальное и интегральное исчисление, оперирующее с бесконечно малыми величинами.

Великий немецкий мыслитель **Готфрид Вильгельм Лейбниц** (Leibniz, Gottfried; 1646–1716) по основной профессии также был юристом. Он состоял на дипломатической службе у различных германских курфюрстов, в связи с этим постоянно разъезжал по Европе и в долгих путешествиях в карете предавался размышлениям на самые различные темы.

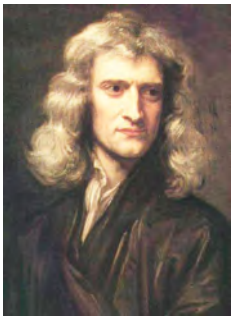
Готфрид Лейбниц

Круг его интересов поразителен. Он был выдающимся философом, психологом, историком, биологом, геологом, физиком, математиком, лингвистом. Изобрел арифмометр, выдвинул идею паровой машины и открыл двоичную систему счисления. Познакомившись с Петром I во время его путешествий в Европу в 1696 и 1711 гг., Лейбниц предложил проект научных исследований в России, что привело к созданию Академии наук в Петербурге. От российского императора Лейбниц получил титул тайного советника юстиции и пенсию в 2000 гульденов.

Будучи в 1673 г. в Лондоне, Лейбниц прослышал о работах Ньютона в области исчисления бесконечно малых и стал независимо от него развивать основы математического анализа. Вывел основные правила дифференцирования и интегрирования, предложил современную терминологию и систему обозначений. В 1684 г. вышла пятистраничная статья Лейбница «Новый метод нахождения максимумов и минимумов...», которая стала первой в мире публикацией по дифференциальному исчислению.

Эта статья чрезвычайно заинтересовала молодого швейцар-

ского математика **Якоба Бернулли**[5] (Bernoulli, Jakob; 1654–1705). Изучив ее, он стал восторженным сторонником нового научного направления, вступил в оживленную переписку с Лейбницем и привлек к работе младшего брата Иоганна (Bernoulli, Johann; 1667–1748). В течение 20 последующих лет Лейбниц и братья Бернулли разработали основные концепции математического анализа.



Исаак Ньютон

Гениальный современник Лейбница **Исаак Ньютон** (Newton, Isaac; 1643–1727) пришел к дифференциальному исчислению (он назвал его методом флюксий) еще раньше, в 1666–1667 гг., когда город Кембридж, где Ньютон работал в знаменитом университете, поразила чума, и он спасался от эпидемии в своем поместье. Однако подход к проблеме у него был не алгебраический, а физический. Новый математический аппарат был нужен Ньютону не сам по себе, а для решения главной задачи — описания фундаментальных законов механики. Основной труд Ньютона «Математические начала натуральной философии» (так в те времена называлась физика) был опубликован в 1687 г., однако в нем он почти не использовал анализ бесконечно малых, а придерживался античных геометрических приемов доказательства. Сочинения по математике увидели свет только в XVIII столетии, но в отличие от ясных рассуждений Лейбница, чисто математические идеи в трудах Ньютона излагались довольно туманно. Ньютон постоянно сомневался, стоит ли публиковать свои достижения. Его работа «Метод флюксий» вышла уже после смерти автора в 1736 г.

Отношения Ньютона с Лейбницем были сложными. Снача-

[5] Семья Николая Бернулли Старшего (1623–1708), переехавшего из неспокойной Голландии в свободный швейцарский город Базель, дала миру несколько поколений математиков. Наиболее известны его сыновья Якоб и Иоганн, а также внуки — сыновья Иоганна — Даниил и Николай.

ла они обменивались дружескими письмами, однако в начале XVIII века между ними разгорелась приоритетная война, «наиболее постыдная склока во всей истории математики», расколовшая Европу на два враждующих лагеря. Хотя исторический приоритет Ньютона был бесспорным, война не ослабевала до декабря 1716 г., когда Ньютону сообщили: «Лейбниц умер — диспут окончен».

Классические аналитические методы решения условных экстремальных задач с ограничениями в виде равенств были созданы в следующем XVIII веке лучшими математиками своего времени Лагранжем и Эйлером (об Эйлере мы будем говорить в исторических замечаниях к гл. 14).

Жозеф Луи Лагранж

Жозеф Луи Лагранж (Lagrange, Joseph Louis; 1736–1813) родился в семье банковского чиновника в Турине (Италия). Учился на адвоката, но, случайно увидев математический трактат, почувствовал свое настоящее призвание. В 19 лет был уже профессором артиллерийской школы в Турине. В 1766 г. по приглашению Фридриха II прибыл в Пруссию, заняв пост президента Берлинской Академии наук. В 1787 г. после кончины августейшего покровителя принял предложение французского монарха Людовика XVI и переехал в Париж, где был принят с королевскими почестями и стал членом Парижской Академии наук, профессором Политехнической школы. Великую французскую революцию Лагранж пережил безболезненно. Ему дали пенсию и назначили в комиссию, занимающуюся разработкой метрической системы мер и весов, а также нового календаря. Там ему удалось благополучно заблокировать революционный проект всеобщего перехода на 12-ричную систему счисления. В послереволюционное императорское время Наполеон пожаловал Лагранжу титул графа, должность сенатора и орден Почетного легиона.

Имя Лагранжа внесено в список 72 величайших ученых Фран-

ции, помещенный на первом этаже Эйфелевой башни. Ему принадлежат классические исследования по различным разделам математического анализа, алгебры, дифференциальных уравнений, механики, математической картографии, астрономии и др.

Великой заслугой Лагранжа является то, что он не просто разработал метод решения условных экстремальных задач, а возвел принцип экстремальности (*принцип наименьшего действия*) в ранг фундаментальных законов природы (см. исторические замечания к гл. 14). Вершиной научной деятельности Лагранжа является «Аналитическая механика», опубликованная в 1788 г., которую современники назвали «научной поэмой». Впервые со времен Архимеда монография по механике не содержала ни одного чертежа, чем автор особенно гордился.

Следующий прорыв — учет ограничений в виде неравенств — произошел почти 200 лет спустя, во второй половине XX века. После Второй мировой войны в Принстонском университете (США, штат Нью-Джерси) под руководством **Альберта Таккера** (Tucker, Albert William; 1905–1995), в течение 20 лет возглавлявшего факультет математики и работавшего рядом с легендарным Джоном фон Нейманом, сложилась самая авторитетная научная школа в области линейного и выпуклого программирования. В 1950 г. Таккер и его ученик **Гарольд Кун** (Kuhn, Harold W.; p. 1925) опубликовали вызвавшую большой научный резонанс статью «Нелинейное программирование», в которой было приведено доказательство их теоремы. Однако в дальнейшем выяснилось, что схожие результаты были получены ранее, и даже дважды: в 1939 г. **Вильямом Карушем** (Karush, William; 1917–1997) из Чикагского университета в его неопубликованной магистерской диссертации и в 1949 г. Фритцем Джоном (John, Fritz; 1910–1994). По этой причине необходимые условия оптимальности часто называются условиями Каруша — Куна — Таккера (ККТ).

Альберт Таккер

Глава 10

Одномерная оптимизация

Как уже отмечалось, изложенную в предыдущей главе теорию выпуклого программирования, основанную на условиях оптимальности Куна — Таккера, весьма затруднительно использовать на практике, так как в реальных условиях целевые функции являются более сложными, чем квадратичные, и свести дело к задаче линейного программирования не удается. Поэтому практические методы, как правило, основываются на других, более простых и интуитивно осязаемых идеях. Они не гарантируют получение точного решения за конечное число шагов, а предлагают итерационные процедуры, позволяющие получить приближенное решение с нужной точностью.

Как мы увидим в следующей главе, подавляющее число существующих методов нелинейного программирования относятся к числу *релаксационных с линейным поиском*, когда оптимизация многомерной функции $f(\overrightarrow{X})$ сводится к последовательности шагов поиска экстремума функции одной переменной

$$\varphi(t) = f(\overrightarrow{X}_k + t\,\overrightarrow{d}),$$

представляющей собой сечение $f(\overrightarrow{X})$ по лучу, исходящему из фиксированной точки $\overrightarrow{X}_k$ в заданном направлении $\overrightarrow{d}$. От того, насколько эффективны алгоритмы одномерной оптимизации, в су-

щественной степени зависит результативность многомерных методов. В связи с этим, а также потому, что одномерные задачи оптимизации представляют самостоятельный интерес, настоящая глава будет посвящена относительно подробному изучению одномерного (линейного) поиска экстремума — *one-dimensional search, line search.*

На первый взгляд особых проблем здесь нет. Любой первокурсник скажет, что для нахождения экстремума следует приравнять нулю первую производную, решить соответствующее уравнение и отобрать из решений то, для которого вторая производная положительна.

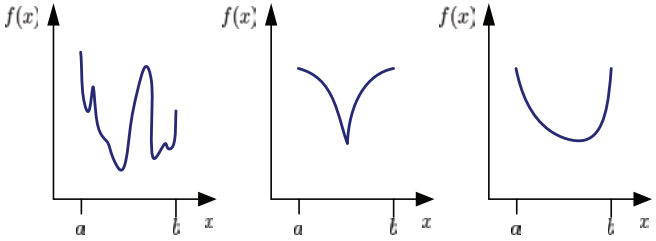
На самом деле все значительно сложнее. Во-первых, численно решить нелинейное уравнение не проще, чем найти сам минимум, а во-вторых, сама целевая функция может быть задана столь сложной формулой или даже неформульным алгоритмом вычисления, что ее производная не выражается в элементарных функциях.

В общем случае задачу приближенного нахождения экстремума (для определенности минимума) функции одной переменной можно поставить следующим образом. Пусть априорно известно, что минимум расположен на отрезке $[a_0, b_0]$. Требуется за k шагов локализовать его на отрезке $[a_k, b_k]$ длиной $b_k - a_k = \varepsilon$.

10.1. Классификация одномерных методов

Прежде чем приступать к минимизации, требуется четко уяснить, какой априорной информацией мы располагаем об оптимизируемой функции, потому что если о целевой функции неизвестно ничего, кроме того, что она непрерывна (рис. 10.1, *а*), единственным способом нахождения глобального экстремума является полный перебор ее значений, отстоящих друг от друга на расстоянии ε.

Самым простым является предположение об *унимодальности* функции, то есть о наличии у нее единственного экстремума



а) *б)* *в)*

Рис. 10.1. Целевые функции одной переменной:
а — произвольная; *б* — унимодальная;
в — выпуклая

(рис. 10.1, *б*). Унимодальная функция слева от экстремума убывает, а справа возрастает, это является основанием для применения значительно более эффективных методов минимизации (мы все время говорим о минимизации, хотя аналогичные рассуждения справедливы и для максимизации). Если же есть основания предположить, что целевая функция не только унимодальна, но и выпукла (рис. 10.1, *в*), становится доступным весь арсенал методов выпуклой минимизации (табл. 10.1).

Конкретные методы оптимизации принято группировать по способам измерения целевой функции. В случаях, когда функция задана чрезвычайно сложным выражением или алгоритмом, можно рассчитывать лишь на вычисление ее значения при конкретном значении аргумента. Методы, основанные исключительно на отсчетах целевой функции, называются *методами нулевого порядка*, в другой терминологии — *прямыми* или *поисковыми* методами.

Если функция задана не слишком сложной формулой, можно аналитически вычислить ее производную для того, чтобы в алгоритме вместе с отсчетами самой функции использовать отсчеты производной. Такие методы называются *методами первого порядка*.

Наконец, когда целевая функция дважды дифференцируема

Таблица 10.1. Классификация методов одномерной оптимизации

Что известно о $f(x)$	Метод	Группа методов
Ничего	Полный перебор	Методы нулевого порядка
Унимодальная	Дихотомия	
	Золотое сечение	
Выпуклая	Метод парабол	
Выпуклая дифференцируемая, известны $f(x), f'(x)$	Метод касательных	Методы первого порядка
Выпуклая дважды дифференцируемая, известны $f(x), f'(x), f''(x)$	Метод Ньютона	Методы второго порядка

и доступны отсчеты самой функции $f(x)$, ее первой $f'(x)$ и второй $f''(x)$ производных, можно использовать *методы второго порядка,* типичным представителем которых является классический метод Ньютона.

10.2. Грубая локализация минимума

Для поиска точки x^* минимума функции одной переменной $f(x)$ обычно требуется указать начальный промежуток $[a_0, b_0]$, в котором достоверно содержится минимум. К этому вопросу нужно относиться ответственно, непростительную ошибку делают программисты, которые «зашивают» концы промежутка в текст программы, считая, например, что значения $a_0 = -1000, b_0 = 1000$ будут приемлемыми во всех случаях.

Правильно установить эти параметры можно либо из анализа содержания задачи, либо опираясь на некоторые априорные свойства функции $f(x)$, опять-таки следующие из ее реального смысла. В частности, если функция априорно унимодальна,

то можно попытаться найти *выпуклую (удачную) тройку* точек $x_1 < x_2 < x_3$, для которой $f(x_1) > f(x_2) < f(x_3)$.

Возможны два варианта прохождения оптимизируемой функции через выпуклую тройку точек (рис. 10.2), однако в силу ее унимодальности в любом случае минимум находится на отрезке $[a_0 = x_1, b_0 = x_3]$. Для нахождения выпуклой тройки можно воспользоваться следующим алгоритмом [17, с. 133].

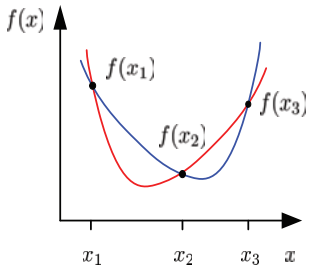


Рис. 10.2. Выпуклая тройка точек

Шаг 1 (нестандартный). Выбирают точку x_0 и определяют направление убывания функции $f(x)$.

Для этого выбирают число h и вычисляют $f(x_0 + h)$. Если $f(x_0 + h) < f(x_0)$, то $x_1 = x_0 + h$ и переходят к шагу 2 при $k = 1$. Если $f(x_0 + h) \geqslant f(x_0)$, то $h := -h$ и вычисляют $f(x_0 + h)$. Если $f(x_0 + h) < f(x_0)$, то полагают $x_1 = x_0 + h$ и переходят к шагу 2 при $k = 1$. Если $f(x_0 + h) \geqslant f(x_0)$, то полагают $h = h/2$ и повторяют процедуру предварительного шага. В результате шага 1 получают число h и точки x_0, $x_1 = x_0 + h$, такие, что $f(x_1) < f(x_0)$.

Шаг 2. Удваивают h и вычисляют $x_{k+1} = x_k + h$.

Шаг 3. Вычисляют $f(x_{k+1})$. Если $f(x_{k+1}) < f(x_k)$ то полагают $k = k + 1$ и переходят к шагу 2. Если $f(x_{k+1}) \geqslant f(x_k)$, то поиск останавливают и в качестве отрезка $[a_0, b_0]$, содержащего точку минимума, выбирают $[x_{k-1}, x_{k+1}]$.

10.3. Минимизация унимодальных функций

Дихотомия

По-видимому, самым простым методом нулевого порядка является метод половинного деления или метод дихотомии (рис. 10.3).

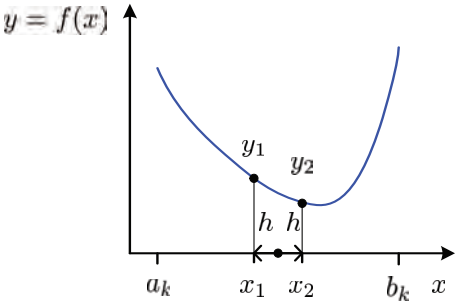


Рис. 10.3. Метод дихотомии

Работа алгоритма начинается с отрезка $[a_0, b_0]$, найденного методом грубой локализации экстремума, после чего следуют итерации ($k = 0, 1, 2, \dots$). На каждой итерации функция вычисляется в двух точках, равноотстоящих от середины текущего промежутка:

$$x_1 = \frac{a_k + b_k}{2} - h, \quad x_2 = \frac{a_k + b_k}{2} + h,$$

где h — параметр алгоритма. Далее переход к следующей итерации ($k := k+1$) с новыми границами промежутка: если $y_1 < y_2$, то $a_{k+1} := a_k, b_{k+1} := x_2$, в противном случае $a_{k+1} := x_1, b_{k+1} := b_k$.

Выход из алгоритма по условию $b_k - a_k < \varepsilon$.

Этот простейший алгоритм имеет два серьезных недостатка. Во-первых, в нем имеется предустановленный параметр h, определяющий разнос точек измерения функции, этот параметр следует согласовывать с длиной промежутка.

Во-вторых, алгоритм неоптимален с точки зрения трудоемкости. Под трудоемкостью алгоритма одномерной оптимизации обычно понимают число измерений функции, необходимое для достижения заданной точности локализации экстремума.

Если абстрагироваться от величины h, то можно считать, что на каждой итерации ценой двух измерений происходит уменьшение отрезка в два раза. Если обозначить относительную точность локализации экстремума через $\delta = (b_k - a_k)/(b_0 - a_0)$, то $\delta = 2^{-\frac{n}{2}}$, где n — число измерений, необходимое для достижения заданной относительной точности. Отсюда для дихотомии

$$n_D = -\frac{2}{\log 2} \log \delta. \tag{10.1}$$

По указанным причинам метод дихотомии на практике применять нецелесообразно, вместо него можно использовать столь же простой, но лишенный указанных недостатков *метод золотого сечения.*

Золотое сечение

Напомним, что золотым сечением отрезка называется такое его разбиение на две неравные части, при котором отношение длины меньшей части к большей равно отношению большей части к длине всего отрезка (рис. 10.4).

Чтобы найти это золотое отношение (обозначим его λ), будем считать длину всего отрезка единичной. Тогда

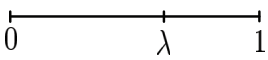


Рис. 10.4. Золотое сечение

$$\frac{1-\lambda}{\lambda} = \frac{\lambda}{1},$$

откуда $\lambda^2 = 1 - \lambda,\ \lambda = \dfrac{\sqrt{5}-1}{2} \approx 0.618$. Приведенное на рис. 10.4 сечение назовем *правым*, потому что точка сечения ближе к правому краю отрезка, однако существует симметричное ему *левое* золотое сечение.

Золотое отношение тесно связано с последовательностью чисел Фибоначчи $\{1, 1, 2, 3, 5, 8, 13, \dots\}$, которые определяются рекуррентным соотношением

$$F_0 = F_1 = 1,\ F_k = F_{k-1} + F_{k-2},\ k \geqslant 2.$$

Легко установить [6], что

$$\lim_{k\to\infty} \frac{F_{k-1}}{F_k} = \lambda.$$

Для наших целей золотое сечение представляет интерес по следующей причине. Возьмем отрезок, для простоты единичный, и поместим на него две точки x_1 и x_2, производящие его левое и правое золотые сечения (рис. 10.5).

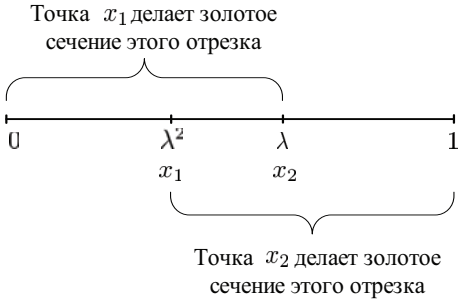


Рис. 10.5. Особенность золотого сечения

Можно показать (мы предлагаем сделать это самостоятельно), что в этом случае левая точка x_1 производит правое золотое сечение отрезка $[0, x_2]$, а правая точка x_2 — левое золотое сечение отрезка $[x_1, 1]$.

В этом и состоит изюминка метода золотого сечения. Как и при дихотомии, на k-м шаге на отрезке $[a_k, b_k]$ производятся два измерения целевой функции в точках

$$x_1 = a_k + \lambda^2(b_k - a_k),\ x_2 = \lambda(b_k - a_k).$$

Если $f(x_1) < f(x_2)$, то в качестве нового отрезка берется $[a_{k+1} = a_k, b_{k+1} = x_2]$, в противном случае $[a_{k+1} = x_1, b_{k+1} = b_k]$, причем в обоих случаях одно измерение для следующей итерации у ж е и м е е т с я.

Таким образом, хотя уменьшение длины отрезка на каждой итерации происходит не в два, а в $1/\lambda \approx 1{,}618$ раза, зато, если не считать начальную итерацию, это происходит ценой всего

одного нового измерения целевой функции. То есть относительная точность, достигаемая за n измерений, равна $\delta = \lambda^n$, откуда $\log \delta = n \log \lambda$, и число измерений целевой функции, необходимое для достижения заданной точности при золотом сечении, равно

$$n_G = \frac{\log \delta}{\log \lambda}.$$

Выигрыш в числе измерений по сравнению с дихотомией составляет

$$\frac{n_D}{n_G} = 2\frac{\log \lambda}{\log 2} \approx 1.388 \text{ раза}.$$

Возникает естественный вопрос: существуют ли еще более эффективные с точки зрения числа измерений методы нулевого порядка для унимодальных функций? Доказано (см., например, [24, с. 56]), что при заранее неизвестном, потенциально бесконечном числе итераций метод золотого сечения является наилучшим из возможных. Если же число итераций k заранее фиксировано, то теоретически возможно (хотя практически нецелесообразно) незначительное увеличение эффективности путем использования самих чисел Фибоначчи. При этом точки измерения целевой функции на первой итерации делят отрезок (если считать его единичным) в пропорции

$$x_1 = \frac{F_{k-2}}{F_k},\ x_2 = \frac{F_{k-1}}{F_k},$$

на следующей итерации k уменьшается на единицу и т. д. За счет рекуррентного свойства последовательности Фибоначчи одно измерение при переходе к следующей итерации будет сохраняться. Такой метод называется *поиском Фибоначчи.* Нетрудно видеть, что при $k \to \infty$ он превращается в золотое сечение.

10.4. Метод парабол

Если априорно известно, что целевая функция не только унимодальна, но и выпукла, эту информацию можно использовать

для ускорения поиска экстремума. Самым простым методом нулевого порядка для выпуклых функций является метод парабол.

Пусть известно, что $f(x)$ – выпуклая функция и некоторым способом (грубой локализацией экстремума) определена выпуклая тройка точек, не лежащих на одной прямой (рис. 10.6):

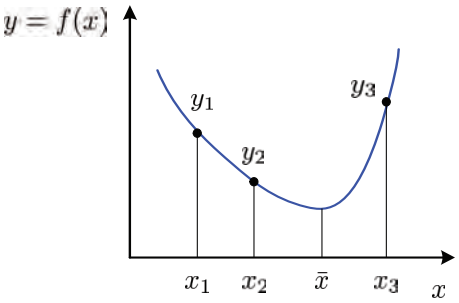


Рис. 10.6. Иллюстрация метода парабол

$$x_1 < x_2 < x_3,$$
$$y_1 \geqslant y_2 \leqslant y_3.$$

Тогда через эти три точки можно провести интерполирующую параболу

$$y = ax^2 + bx + c.$$

Точку минимума $\bar{x}$ параболы можно принять за приближенное значение экстремума, она находится из уравнения $y'(x) = 0$:

$$2ax + b = 0,$$
$$\bar{x} = -\frac{b}{2a}.$$

Коэффициенты a и b определяются из решения системы уравнений (они линейны относительно переменных a, b, c)

$$ax_1^2 + bx_1 + c = y_1,$$
$$ax_2^2 + bx_2 + c = y_2,$$
$$ax_3^2 + bx_3 + c = y_3.$$

Решая систему по правилу Крамера, получаем

$$\bar{x} = -\frac{1}{2}\frac{\begin{vmatrix} x_1^2 & y_1 & 1 \\ x_2^2 & y_2 & 1 \\ x_3^2 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} y_1 & x_1 & 1 \\ y_2 & x_2 & 1 \\ y_3 & x_3 & 1 \end{vmatrix}}.$$

Легко проверить, что $x_1 \leqslant \bar{x} \leqslant x_3$, и $y(\bar{x}) \leqslant y_2$.

Если $\bar{x} < x_2$, то в качестве следующей выпуклой тройки точек берутся $\{x_1, \bar{x}, x_2\}$, в противном случае $\{x_2, \bar{x}, x_3\}$.

10.5. Метод касательных

Все рассмотренные методы относились к методам нулевого порядка, так как не использовали информацию о производной оптимизируемой функции.

Описываемый ниже метод является методом первого порядка, т. е. предполагает знание не только самой функции $f(x)$, но также ее первой производной $f'(x)$. Идея метода состоит в аппроксимации выпуклой функции отрезками касательных.

Если $f(x)$ выпукла и дифференцируема, то уравнение касательной в точке x_k имеет вид

$$y = g_k(x) = f(x_k) + f'(x_k)(x - x_k).$$

Предположим, что минимум целевой функции грубо локализован на промежутке $[a, b]$ (см. рис. 10.7).

На начальной итерации возьмем произвольную начальную точку $x_0 \in [a, b]$ и построим касательную

$$y = g_0(x) = f(x_0) + f'(x_0)(x - x_0).$$

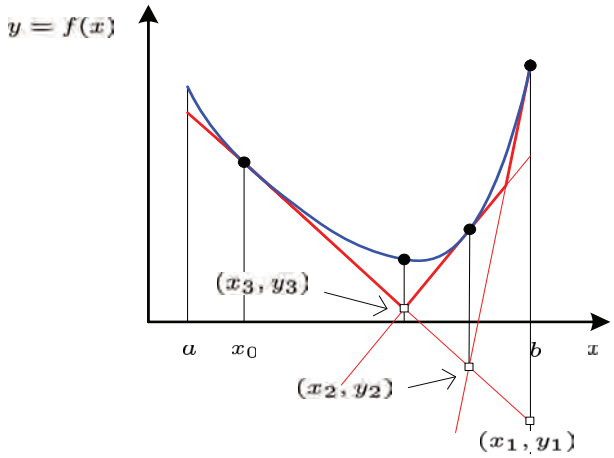


Рис. 10.7. Иллюстрация метода касательных

Далее сформулируем вспомогательную задачу линейного программирования (задачу 0) с двумя переменными x, y:

$$y \to \min,$$
$$y \geqslant g_0(x),$$
$$a \leqslant x \leqslant b.$$

Геометрически это означает, что, не выходя по абсциссе за пределы отрезка $[a, b]$, мы ищем точку с минимальной ординатой, расположенную н а д касательной к точке x_0. Разумеется, минимум достигается на одном из концов промежутка $[a, b]$, в нашем случае на правом. Координаты этой точки обозначим x_1, y_1, переходим к следующей итерации.

На п е р в о й итерации построим касательную к целевой функции в точке x_1

$$y = g_1(x) = f(x_1) + f'(x_1)(x - x_1),$$

добавим ее к ограничениям задачи 0 и построим задачу 1:

$$y \to \min,$$

$$y \geqslant g_0(x),$$
$$y \geqslant g_1(x),$$
$$a \leqslant x \leqslant b.$$

Решение этой задачи дает точку x_2, y_2. Она имеет наименьшую ординату среди точек, расположенных над обеими касательными.

На в т о р о й итерации строим касательную $y = g_2(x)$ в точке x_2 и, добавляя ограничение $y \geqslant g_2(x)$ к условиям задачи 1, получаем задачу 2. Решая ее, получаем точку x_3, y_3 и т. д.

Таким образом, на каждой итерации, добавляя новую касательную, мы уточняем аппроксимацию целевой функции отрезками касательных и приближаемся к точке ее минимума.

Разумеется, при реализации метода для решения вспомогательных задач линейного программирования нет нужды привлекать тяжелую артиллерию симплексного метода, можно придумать какой-нибудь простой алгоритм перебора точек пересечения касательных (студенты факультета информатики делают это с легкостью). Тем не менее в целом метод выглядит тяжеловесным, мы привели его не для практического использования, а для демонстрации того, как может быть использована дополнительная информация о производной.

10.6. Метод Ньютона

Метод Ньютона (см. исторические замечания в конце главы) является классическим методом второго порядка, так как предполагает знание не только самой функции $f(x)$, но также ее первой и второй производных $f'(x), f''(x)$.

В отличие от всех предыдущих, данный метод не требует грубой локализации экстремума на отрезке $[a_0, b_0]$. Пусть приближенное значение минимума на k-й итерации равно x_k (см. рис. 10.8). Идея метода состоит в аппроксимации минимизируемой функции параболой $y = g(x)$, построенной *по одной точке* x_k. Это возможно, так как известны значение самой функции, направление каса-

тельной (первая производная) и кривизна (вторая производная). Уравнение аппроксимирующей параболы в точке x_k при этом имеет вид

$$y = g(x) = f(x_k) + f'(x_k)(x - x_k) + \frac{f''(x_k)}{2}(x - x_k)^2.$$

В качестве следующего $k+1$ приближенного значения экстре-

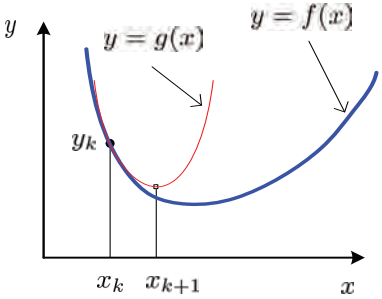


Рис. 10.8. Иллюстрация метода Ньютона. Целевая функция $f(x)$, аппроксимирующая парабола $g(x)$

мума принимаем точку минимума аппроксимирующей параболы, которая находится из уравнения $g'(x) = 0$, откуда

$$f'(x_k) + f''(x_k)(x - x_k) = 0,$$

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \tag{10.2}$$

Доказательство сходимости метода Ньютона и оценка скорости сходимости для выпуклых функций имеется в любом продвинутом учебнике по методам оптимизации, например [24, с. 219].

Замечание. Методом Ньютона в вычислительной математике обычно называют простейшую итеративную процедуру решения уравнения $f(x) = 0$, когда нелинейная функция аппроксимируется касательной, и следующее приближение для корня вычисляется по геометрически очевидной формуле (рис. 10.9)

$$x_{k+1} = x_k - \frac{f(x_k)}{\mathrm{tg}\varphi} = x_k - \frac{f(x_k)}{f'(x_k)}. \tag{10.3}$$

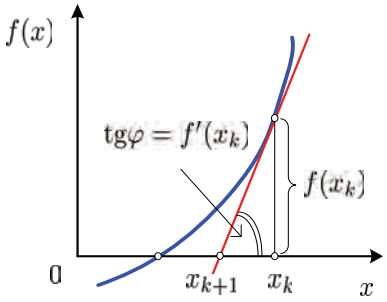


Рис. 10.9. Решение нелинейного уравнения методом Ньютона

Однако, поскольку задача нахождения экстремума функции $f(x)$ эквивалентна нахождению корня уравнения $f'(x) = 0$, итеративная процедура остается той же, только саму функцию следует заменить ее первой производной, а первую производную — второй. Получилась формула (10.2).

Подводя итог методам одномерной оптимизации, следует сказать, что с практической точки зрения наиболее удобными являются методы нулевого порядка (золотого сечения и парабол). Методы первого и второго порядков применимы лишь тогда, когда оптимизируемая функция имеет достаточно простую аналитическую форму, чтобы производные могли быть получены в явном виде и вычислялись отдельными подпрограммами. Это сделать тем более непросто, когда задача одномерной оптимизации является вспомогательной для вычисления длины шага в релаксационном алгоритме.

10.7. Исторические замечания

Золотое отношение известно с античных времен, древние математики придавали ему мистический смысл. Установлено, что пропорции, определяемые золотым отношением, наиболее приятны для глаза, они часто встречаются в классических архитектур-

Рис. 10.10. Идеальные пропорции

ных сооружениях, например в храме Парфенон в Афинах.

Леонардо да Винчи считал, что идеальные пропорции человеческого тела определяются золотым отношением длин меньшей и большей частей, приблизительно равным 38:62 (рис. 10.10).

Числа Фибоначчи названы по имени средневекового математика **Леонардо Пизанского** (1180–1240), известного под прозвищем **Фибоначчи** (от ит. «Figlio Buono Nato Ci — хороший сын родился»). В своем главном труде «Книга абака» (1202) он впервые систематически изложил для европейцев достижения арабской математики, а также ввел в рассмотрение возвратную последовательность.

Метод Ньютона — Рафсона имеет долгую историю. Первый вариант, пригодный для приближенного вычисления корней полиномов, был разработан Ньютоном в 1669 г. и впервые опубликован в 1685 г. не им самим, а в виде приложения к «Алгебре» Джона Уоллиса (Wallis, John; 1616–1703). В 1690 г. **Джозеф Рафсон** (Raphson, Joseph; 1649–1715) предложил упрощенный вариант метода, пригодный опять-таки для полиномов. Он же перевел на английский язык некоторые книги Ньютона, написанные на латыни.

Фибоначчи

Выдающийся английский математик следующего века **Томас Симпсон** (Simpson, Thomas; 1710–1761), оставивший свое имя в «формуле Симпсона» численного интегрирования, в работе, вышедшей в 1740 г., обобщил метод Ньютона — Рафсона на произвольные функции и придал ему современный вид.

Глава 11

Многомерная оптимизация без ограничений

В этой главе мы сделаем еще один шаг к решению общей задачи математического программирования. Опираясь на общую идею релаксационных методов и имея в запасе арсенал алгоритмов одномерной оптимизации, попытаемся найти экстремум функции многих переменных $f(\overrightarrow{X}) = f(x_1, \ldots, x_n)$ без учета ограничений. Проблему оптимизации с ограничениями откладываем до следующей главы.

11.1. Релаксационные методы оптимизации

Существует множество приближенных итерационных методов оптимизации без ограничений, называемых *релаксационными* (*to relax* – снижать, расслаблять) потому, что в процессе их работы, начиная с некотором образом выбранной исходной точки $\overrightarrow{X}_0$, образуется последовательность точек $\overrightarrow{X}_1, \overrightarrow{X}_2, \ldots, \overrightarrow{X}_k, \overrightarrow{X}_{k+1}, \ldots$, в которой каждая последующая имеет лучшее (меньшее) значение целевой функции: $f(\overrightarrow{X}_{k+1}) < f(\overrightarrow{X}_k)$. Если задача имеет решение и целевая функция ограничена, то в пределе эта последовательность должна сходиться к точке минимума: $\overrightarrow{X}_k \to \overrightarrow{X}^*$.

Для того чтобы прочувствовать проблемы и идеи релаксационных методов, можно воспользоваться такой аналогией. Представим себе путника, оказавшегося ночью в горной местности и желающего прийти в точку с минимальной высотой. Склоны гор ровные, нет ни камней, ни трещин, движение возможно во всех направлениях, однако из-за полной темноты путник не видит всего рельефа и не может определить истинное направление на цель, он полагается только на ощущения в ближайшей окрестности места своего нахождения[1]. Если путник находится в текущей точке $\overrightarrow{X}_k$ и хочет сделать очередной шаг для того, чтобы спуститься ниже, то он должен определить:

- направление шага $\overrightarrow{d}_k$,
- величину шага t_k в выбранном направлении,

причем сделать это таким образом, что в новой точке, определяемой по правилу

$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k \overrightarrow{d}_k,$$

значение целевой функции уменьшится.

Подавляющее большинство релаксационных методов относится к классу методов, основанных на *линейном поиске (linesearch methods).* В них первичным является выбор направления, после чего производится движение в выбранном направлении. Эти методы развивались на протяжении нескольких столетий (см. исторические замечания в конце главы), хорошо себя зарекомендовали на практике и включены во все пакеты прикладных программ оптимизации. По этой причине все внимание в дальнейшем будет уделено именно им.

[1]Некоторые авторы, например [25], предпочитают пользоваться противоположной аналогией, формулируя задачу не на минимум, а на максимум. Тогда ущелья превращаются в хребты, а цель поиска состоит в восхождении на высочайшую вершину.

Общая схема

Каждая k-я итерация спуска ($k = 0, 1, 2, \dots$) определяется следующим алгоритмом (рис. 11.1).

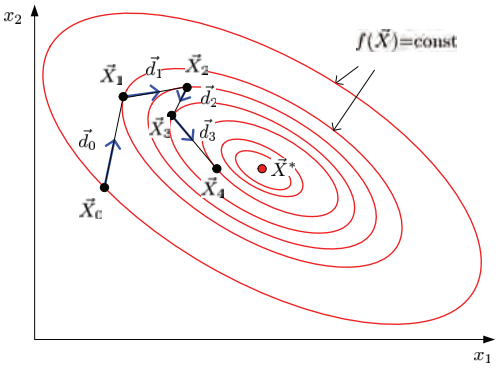


Рис. 11.1. Схема работы метода спуска на примере минимизации выпуклой функции двух переменных

1. В текущей точке $\overrightarrow{X}_k$, $k = 0, 1, 2, \dots$ определяется некоторое направление движения $\overrightarrow{d}_k$, обладающее тем свойством, что, двигаясь в данном направлении, можно хоть ненамного уменьшить значение целевой функции. Любое такое направление называется *понижающим (descending)* или *прогрессивным*. Сечение целевой функции $f(\overrightarrow{X})$ вдоль выбранного направления является одномерной функцией от величины перемещения t : $\varphi(t) = f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k)$.

2. Определяется величина шага t_k в выбранном прогрессивном направлении, такая, что $f(\overrightarrow{X}_{k+1}) < f(\overrightarrow{X}_k)$.

3. Происходит спуск в точку, являющуюся исходной для следующей итерации: $\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k\,\overrightarrow{d}_k$.

4. Проверяется критерий остановки алгоритма.

Из приведенного описания ясно, что релаксационные методы спуска могут различаться:

- по способу выбора понижающего направления,
- по способу выбора длины шага,
- по критерию остановки итерационного процесса.

Выбор направления спуска

Даже если ограничения отсутствуют, проблема выбора понижающего направления совсем не проста. С одной стороны, она определяется той априорной информацией, которой мы располагаем об оптимизируемой функции. Если, например, функция задана аналитически простой формулой, то легко вычислить все ее производные. Вполне логично в этом случае двигаться по направлению наибольшей крутизны, то есть антиградиенту. Если же информация о производных недоступна, применяются другие подходы.

С другой стороны, необходимо учитывать особенности целевой функции. Существует множество очень трудных для оптимизации, так называемых *овражных* функций, рельеф которых изрезан глубокими оврагами с крутыми стенами. Как мы увидим в дальнейшем, движение по антиградиенту в этом случае является далеко не оптимальным, приходится изыскивать иные способы выбора прогрессивного направления.

Наличие ограничений, особенно нелинейных, резко усложняет задачу оптимизации, поскольку не все направления движения из исходной точки являются *возможными (feasible)*, а только те, по которым можно хоть немного продвинуться, не выходя за пределы области допустимых значений $\overrightarrow{X}$. Продолжая аналогию с горами, можно представить, что разрешенная область ограничена забором. Если путник касается забора, то возможными направлениями будут только те, которые ведут внутрь области или хотя бы вдоль забора.

По этой причине рассмотрение соответствующих методов обычно начинают после того, как изучены методы оптимизации без учета ограничений. Для этого есть все основания, так как исходную задачу с ограничениями часто можно свести к задаче без ограничений. Мы не будем отходить от этой традиции и отложим оптимизацию с ограничениями до гл. 12.

Выбор длины шага

С точки зрения определения длины шага все методы спуска делятся на две группы. В методах *с малым (дробным) шагом* величина шага спуска в выбранном направлении является предустановленным параметром алгоритма: $t_k = h$. То есть путник, сделав шажок фиксированной длины, заново определяет направление своего следующего шага, даже если возможность уменьшения целевой функции за счет движения в прежнем направлении не исчерпана.

Следует заметить, что фиксирование длины шага, впрочем как и любых других численных параметров алгоритма, несет определенную опасность. Для одних условий величина шага может оказаться слишком малой, в результате число итераций получится непомерно большим, а для других, наоборот, она будет слишком велика, и, сделав шаг установленной длины, можно проскочить точку минимума функции $\varphi(t)$. Поэтому при конструировании рабочего алгоритма с малым шагом следует предусмотреть подстройку длины шага (см., например, [17, с. 163]).

В методах *с большим (полным) шагом* движение по лучу, исходящему из $\overrightarrow{X}_k$ в направлении d_k, осуществляется до тех пор, пока не будет достигнут минимум целевой функции в данном направлении. При этом длина шага находится из соотношения

$$t_k = \arg\min_t \varphi(t) = \arg\min_t f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k).$$

Таким образом, в полношаговых методах спуска задача оптимизации функции многих переменных сводится к последовательности задач одномерного (линейного) поиска минимума.

Критерий остановки

Подобно всем итеративным методам, релаксационные алгоритмы могут работать бесконечно, их остановка производится принудительно по достижении заданной точности вычислений. В зависимости от конкретных условий из итерационного цикла можно выходить по любому из следующих *критериев остановки (stopping criterion).*

По выполнению условий первого порядка. Как мы знаем (см. с. 39), равенство нулю градиента целевой функции является необходимым, а в случае выпуклой функции — и достаточным условием экстремума. Практически эта проверка реализуется следующим образом. В текущей точке $\overrightarrow{X}_k$ вычисляется *мера оптимальности первого порядка (first order optimality measure)*, и если она менее установленного *допуска (tolerance)* ε, например 10^{-6}, то условие считается выполненным. В качестве меры можно взять евклидову норму вектора градиента, однако чаще используется условие с так называемой *бесконечной нормой*:

$$\| \nabla f(\overrightarrow{X}_k) \|_\infty = \max_{1 \leqslant j \leqslant n} \left| \frac{\partial f(\overrightarrow{X}_k)}{\partial x_j} \right| < \varepsilon.$$

Данный критерий является самым логичным, однако он требует знания производных целевой функции, в связи с этим широко используются другие критерии, основанные на сравнении двух соседних точек в итерационном процессе.

По малости смещения текущей точки. Возможны два варианта данного критерия: по абсолютному либо относительному смещению. В первом случае оценивается евклидова норма вектора $\|\overrightarrow{X}_{k+1} - \overrightarrow{X}_k\| < \varepsilon$, во втором для измерения относительного смещения часто используют отношение

$$\frac{\|\overrightarrow{X}_{k+1} - \overrightarrow{X}_k\|}{1 + \|\overrightarrow{X}_k\|} < \varepsilon,$$

где единица в знаменателе исключает возможность деления на нуль.

По малости изменения целевой функции. Аналогично, критерием выхода по абсолютному изменению целевой функции является выполнение неравенства $|f(\overrightarrow{X}_{k+1}) - f(\overrightarrow{X}_k)| < \varepsilon$, а по относительному — соотношения

$$\frac{|f(\overrightarrow{X}_{k+1}) - f(\overrightarrow{X}_k)|}{1 + |f(\overrightarrow{X}_k)|} < \varepsilon.$$

По малости производной по направлению. Еще одним критерием может служить оценка производной по направлению в текущей точке. Используя ее определение (см. с. 30), можно записать конечно-разностное приближение

$$\frac{\partial f(\overrightarrow{X})}{\partial \overrightarrow{d}} = \lim_{t \to 0} \frac{f(\overrightarrow{X} + t\overrightarrow{d}) - f(\overrightarrow{X})}{\|t\overrightarrow{d}\|} \approx \frac{f(\overrightarrow{X}_{k+1}) - f(\overrightarrow{X}_k)}{\|\overrightarrow{X}_{k+1} - \overrightarrow{X}_k\|}.$$

Если абсолютная величина этого выражения менее установленного порога, работа алгоритма завершается.

По объему вычислений. В качестве «предохранительного клапана», защищающего итеративный процесс от зацикливания или недопустимо большого времени работы, могут служить прямые ограничения на количество итераций либо на число вычислений целевой функции.

В практических алгоритмах указанные критерии используются независимо либо в комбинации друг с другом.

Классификация релаксационных методов

Основанием для классификации является, как и в одномерном случае, доступная априорная информация о целевой функции (табл. 11.1). В литературе описано много различных методов, в таблице мы указали и в дальнейшем будем рассматривать только самые простые и типичные. Как мы отмечали выше (см. с. 94), они могут отличаться друг от друга способом выбора понижающего (прогрессивного) направления,

Таблица 11.1. Классификация релаксационных методов

Что известно о $f(\overrightarrow{X})$	Метод	Группа методов
Выпуклая	Метод покоординатного спуска (Гаусса — Зайделя)	Методы нулевого порядка
	Метод многогранника (симплексный) Нелдера — Мида	
	Поиск по образцу (метод конфигураций Хука — Дживса)	
Выпуклая, дифференцируемая, известен градиент $\overrightarrow{\nabla} f(\overrightarrow{X})$	Метод скорейшего спуска	Методы первого порядка
	Метод сопряженных градиентов (Флетчера — Ривса)	
	Квазиньютоновские методы	
Выпуклая, дважды дифференцируемая, известны градиент $\overrightarrow{\nabla} f(\overrightarrow{X})$ и матрица Гессе $\mathrm{H}(\overrightarrow{X})$	Классический метод Ньютона и его модификации	Методы второго порядка

величиной шага и критерием окончания поиска. Из этих трех параметров главным является способ выбора направления, так выбор длины шага чаще всего ведется поиском минимума функции в выбранном направлении, а критерий остановки не имеет принципиального значения.

11.2. Методы нулевого порядка

Метод покоординатного спуска

Метод циклического *покоординатного спуска (coordinatewise descend)*, носящий имя Гаусса — Зайделя[2], является классическим

[2]Зайдель (Зейдель) (Philipp Ludwig von Seidel; 1821–1896) — немецкий математик и астроном. Труды по теории рядов и других бесконечных форм. В

полношаговым методом оптимизации, не требующим знания производных.

Направления одномерной оптимизации на каждой итерации выбираются параллельными координатным осям, то есть изменяется только одна переменная. При этом для одномерной оптимизации можно использовать любой метод нулевого порядка: золотое сечение или метод парабол. После того, как найден минимум по первой переменной, осуществляется поиск по второй, и так до последней, затем цикл повторяется, пока не будет достигнута требуемая точность.

Символически покоординатный спуск описывается следующими соотношениями:

$$\begin{aligned}
&\overrightarrow{d}_k \in \{\overrightarrow{e}_1, \overrightarrow{e}_2, \ldots, \overrightarrow{e}_n\},\\
&t_k = \arg\min_t f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k),\\
&\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k\,\overrightarrow{d}_k.
\end{aligned}$$

Доказательство сходимости метода для выпуклых функций приведено в [17, с. 194].

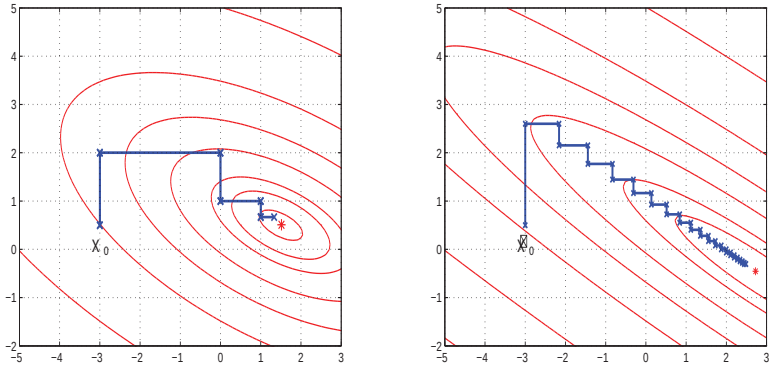
Главным достоинством метода является его предельная простота. Весь вопрос в том, какова скорость сходимости, и здесь мы сталкиваемся с проблемой, общей для всех методов многомерной оптимизации.

Проблема овражности

Рассмотрим поведение метода Гаусса — Зайделя на простом примере минимизации квадратичной функции двух переменных в двух различающихся ситуациях (см. рис. 11.2).

На рис. 11.2, *а* целевая функция взята из примера на с. 35. Матрица D квадратичной формы является хорошо обусловленной, ее число обусловленности $\mathrm{cond}(D) \approx 6$. Рельеф такой функции похож на яму, склоны которой имеют приблизительно равную крутизну во всех направлениях. В этом случае покоординатный

астрономии известен своими исследованиями яркости звезд.



а) *б)*

Рис. 11.2. Работа метода циклического покоординатного спуска на хорошо обусловленной (*а*) и плохо обусловленной (овражной) (*б*) квадратичных целевых функциях

спуск оказывается весьма эффективным, уже пять первых итераций дают хорошее приближение к минимуму.

В примере на рис. 11.2, *б* матрица квадратичной формы равна $D = \begin{pmatrix} 1 & 1.6 \\ 1.6 & 3 \end{pmatrix}$ с числом обусловленности около 34. Поэтому рельеф этой целевой функции похож на овраг с резко различающейся крутизной склонов, причем направление дна оврага не совпадает ни с одной из координатных осей. В такой ситуации перемещение от итерации к итерации происходит в основном в виде зигзагообразного перескакивания с одного склона на другой, а движение в сторону минимума идет очень медленно. Для нахождения экстремума с высокой точностью могут потребоваться многие тысячи итераций.

Целевые функции, обладающие подобным свойством, называются *овражными*. Они являются камнем преткновения для любых методов оптимизации, поэтому в качестве тестовых функций,

на которых проверяется работоспособность алгоритмов, обычно используются функции со сложной структурой овражного рельефа. Хрестоматийным примером овражной функции двух переменных является *функция Розенброка* [27, с. 214]

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \qquad (11.1)$$

Точка минимума с координатами $(1, 1)^T$ расположена на дне узкого крутого ущелья, дно которого к тому же изогнуто по параболе $x_2 = x_1^2$ (рис. 11.3). Из-за своеобразного вида изолиний в окрестности нуля функцию Розенброка иногда называют *банановой функцией (banana function).*

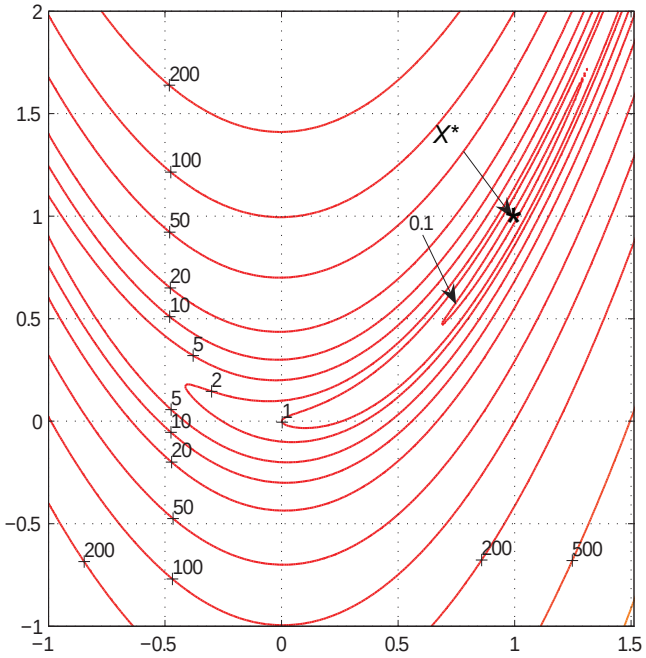


Рис. 11.3. Изолинии функции Розенброка

Задачи оптимизации с подобными функциями принято называть *плохо обусловленными (ill-conditioned problems)*. Для того, чтобы численно измерить степень обусловленности (величину овражности) целевой функции в окрестности точки $\overrightarrow{X}_k$, воспользуемся ее разложением в многомерный ряд Тейлора

$$f(\overrightarrow{X}) = f(\overrightarrow{X}_k) + \overrightarrow{\nabla}^T f(\overrightarrow{X}_k)(\overrightarrow{X} - \overrightarrow{X}_k) + \frac{1}{2}(\overrightarrow{X} - \overrightarrow{X}_k)^T \mathrm{H}(\overrightarrow{X}_k)(\overrightarrow{X} - \overrightarrow{X}_k),$$

где $\overrightarrow{\nabla}^T f(\overrightarrow{X}_k)$ — вектор градиента, а $\mathrm{H}(\overrightarrow{X}_k)$ — матрица Гессе. Таким образом, целевая функция в данной окрестности аппроксимируется многомерной квадратичной функцией, где в качестве матрицы квадратичной формы выступает матрица Гессе. Как мы знаем (см. с. 33), степень сплющенности эллипсоидов равных уровней для квадратичной функции измеряется *числом обусловленности (condition number)* $\mathrm{cond}(\mathrm{H})$. Если $\mathrm{cond}(\mathrm{H}) \approx 1$, то поверхности равных уровней близки к сферическим и свойство овражности в данной окрестности не наблюдается (см. рис. 11.2, *а*). Если же $\mathrm{cond}(\mathrm{H}) \gg 1$, то матрица относится к классу плохо обусловленных, эллипсоиды равных уровней сильно вытянуты, в результате чего целевая функция в окрестности $\overrightarrow{X}_k$ имеет выраженный овражный характер (см. рис. 11.2, *б*).

П р и м е р. Матрица Гессе для функции Розенброка имеет вид

$$\mathrm{H}(x_1, x_2) = \begin{pmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{pmatrix}.$$

Выясним поведение этой функции в окрестности точки оптимума. Подставляя координаты $x_1 = x_2 = 1$, получаем

$$\mathrm{H}(1,1) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}.$$

Собственные числа матрицы $M = \lambda_1 \approx 1000; \quad m = \lambda_2 \approx 0.4$, отсюда число обусловленности $\mathrm{cond}(\mathrm{H}) = \frac{M}{m} \approx 2500 \gg 1$. Рельеф

функции Розенброка в окрестности точки минимума представляет собой узкий овраг с такими крутыми склонами, что его трудно изобразить на графике (см. рис. 11.3). ▲

Метод многогранника

Для решения плохо обусловленных задач изобретен ряд эвристических методов нулевого порядка. Одним из наиболее простых и эффективных считается *метод (деформируемого) многогранника* Нелдера — Мида (Nelder — Mead), который в первоначальном варианте назывался *симплексным методом*. Кроме названия он не имеет ничего общего с симплексным методом в линейном программировании, а назван так потому, что в нем моделируется процесс перекатывания правильного n-мерного многогранника (симплекса) вниз по поверхности целевой функции.

Продемонстрируем симплексный метод на примере функции двух переменных (рис. 11.4). Симплекс в данном случае будет представлять собой правильный треугольник. Пусть начальное расположение его вершин есть $\overrightarrow{X}_1, \overrightarrow{X}_2, \overrightarrow{X}_3$. Вычислим целевую функцию в каждой из них и найдем вершину с наихудшим (наибольшим) значением. В нашем случае это будет вершина $\overrightarrow{X}_1$. Итерация метода состоит в том, что текущий симплекс перемещается в новое положение, при котором все вершины, кроме наихудшей, остаются на своих местах, а последняя отражается в точку, зеркальную относительно центра тяжести противоположной грани. В нашем примере вершина $\overrightarrow{X}_1$ отразится в точку $\overrightarrow{X}_4$ (это отмечено пунктирной стрелкой), и новый симплекс определится вершинами $\overrightarrow{X}_2, \overrightarrow{X}_3, \overrightarrow{X}_4$. На следующей итерации наихудшей будет вершина $\overrightarrow{X}_3$, которая отразится в точку $\overrightarrow{X}_5$, и т. д.

В описанном варианте метода существует проблема выбора размеров симплекса. Если сделать его большим, то есть опасность проскочить экстремум, а если слишком малым — то замедлится работа. В связи с этим в усовершенствованной модификации симплексного метода предусмотрена подстройка размеров граней в ходе работы, тем самым он превращается в метод деформируемо-

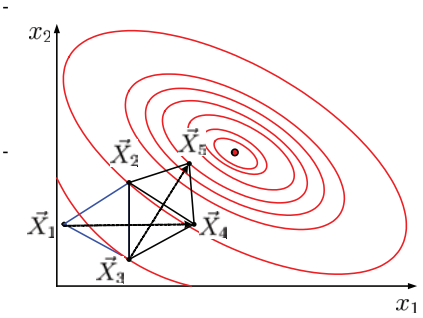


Рис. 11.4. Итерации симплексного метода

го многогранника [27, с. 163]; [21, с. 138].

В многомерном случае алгоритм метода следующий.

П о д г о т о в и т е л ь н ы й э т а п

Исходный многогранник в n-мерном пространстве, содержащий $n+1$ вершину $\overrightarrow{X}_1, \ldots, \overrightarrow{X}_{n+1}$, строится по правилу $\overrightarrow{X}_1 = \overrightarrow{X}_0$, $\overrightarrow{X}_{i+1} = \overrightarrow{X}_0 + h\overrightarrow{e}_i$, $i = 1, \ldots, n$, где $\overrightarrow{X}_0$ — исходная точка, $\overrightarrow{e}_i$ — координатный орт, h — предустановленный начальный размер симплекса.

И т е р а ц и я

Каждая итерация может завершиться одним из двух исходов. Либо включается новая вершина, которая замещает самую худшую, при этом происходит перемещение многогранника (шаг 5), либо это невозможно, тогда он сжимается со всех сторон (шаг 6). Выход из итерационного процесса происходит по достижению установленного минимального размера многогранника.

Шаг 1. Оценка. Вычисляются значения целевой функции в вершинах многогранника, затем они упорядочиваются по возрастанию. Таким образом, $f(\overrightarrow{X}_1) \leqslant \cdots \leqslant f(\overrightarrow{X}_{n+1})$, и $\overrightarrow{X}_1$ — самая лучшая, а $\overrightarrow{X}_{n+1}$ — самая худшая вершина.

Шаг 2. Отражение. Определяется центр тяжести противоположной грани $\overrightarrow{M} = \frac{1}{n}\sum_{i=1}^{n} \overrightarrow{X}_i$ (вершина $\overrightarrow{X}_{n+1}$ в суммировании не участвует), вычисляется вектор шага от худшей точки до центра тяжести $\overrightarrow{\Delta} = \overrightarrow{M} - \overrightarrow{X}_{n+1}$, и находится *отраженная (reflected)* точка $\overrightarrow{R} = \overrightarrow{M} + \overrightarrow{\Delta} = 2\overrightarrow{M} - \overrightarrow{X}_{n+1}$ (рис. 11.5).

Если $f(\overrightarrow{X}_1) \leqslant f(\overrightarrow{R}) < f(\overrightarrow{X}_n)$, то отраженная точка $\overrightarrow{R}$ включается в список вершин. Завершение итерации (шаг 5).

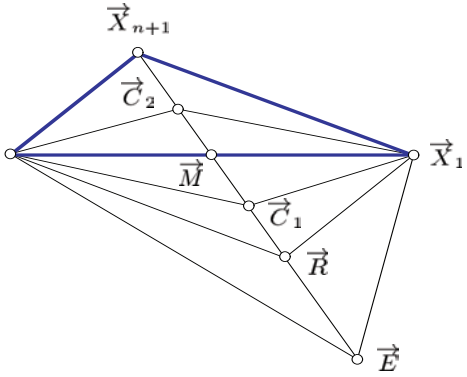


Рис. 11.5. Деформирование многогранника

Шаг 3. Растяжение. Если $f(\overrightarrow{R}) < f(\overrightarrow{X}_1)$, т. е. результат отражения лучше, чем у самой лучшей вершины, делается попытка *растянуть (expand)* многогранник в данном направлении, для чего вычисляется удаленная точка $\overrightarrow{E} = \overrightarrow{M} + 2\overrightarrow{\Delta} = 3\overrightarrow{M} - 2\overrightarrow{X}_{n+1}$ (коэффициент растяжения у нас равен двум, но возможно и другое значение).

- Если $f(\overrightarrow{E}) < f(\overrightarrow{R})$, то растяжение успешное, точка $\overrightarrow{E}$ включается в список вершин. Завершение итерации (шаг 5).

- В противном случае попытка растяжения безуспешная, в список вершин включается отраженная точка $\overrightarrow{R}$. Завершение итерации (шаг 5).

Шаг 4. Сокращение. Если $f(\overrightarrow{R}) \geqslant f(\overrightarrow{X}_n)$, то происходит *сокращение (contraction)* многогранника с одной стороны, для чего анализируются две возможности.

- Если $f(\overrightarrow{R}) < f(\overrightarrow{X}_{n+1})$, вычисляется *внешняя* точка $\overrightarrow{C}_1 = \overrightarrow{M} + \overrightarrow{\Delta}/2$ (коэффициент сокращения также установлен равным двум, хотя его можно изменить). Далее:
  - если $f(\overrightarrow{C}_1) < f(\overrightarrow{R})$, то $\overrightarrow{C}_1$ включается в список вершин, т. е. производится *внешнее сокращение (contraction outside).* Завершение итерации (шаг 5);
  - иначе сжатие многогранника (шаг 6).
- Если $f(\overrightarrow{R}) \geqslant f(\overrightarrow{X}_{n+1})$, вычисляется *внутренняя* точка $\overrightarrow{C}_2 = \overrightarrow{M} - \overrightarrow{\Delta}/2$. Затем:
  - если $f(\overrightarrow{C}_2) < f(\overrightarrow{X}_{n+1})$, то включается точка $\overrightarrow{C}_2$, т. е. производится *внутреннее сокращение (contraction inside).* Завершение итерации (шаг 5);
  - иначе сжатие многогранника (шаг 6).

Шаг 5. Перемещение. Включенная точка замещает вершину $\overrightarrow{X}_{n+1}$, тем самым многогранник переходит в новое положение. Возврат на начало итерации (шаг 1).

Шаг 6. Сжатие. Происходит *сжатие (shrink)* многогранника, например, в два раза, во всех направлениях, при этом лучшая вершина $\overrightarrow{X}_1$ остается на месте, остальные пересчитываются по формуле $\overrightarrow{X}_i = \overrightarrow{X}_1 + (\overrightarrow{X}_i - \overrightarrow{X}_1)/2$, $i = 2, \ldots, n+1$. Возврат на начало итерации (шаг 1).

На рис. 11.6 показаны первые 40 итераций поиска экстремума функции Розенброка методом деформируемого многогранника. Исходная точка $\overrightarrow{X}_0 = (-0.5,\ 0.5)^T$, $h = 0.5$, начальный симплекс выделен жирными линиями. Легко убедиться, что метод хорошо адаптируется к овражному рельефу целевой функции.

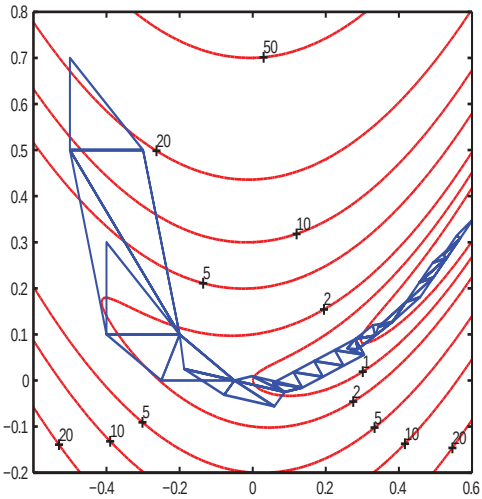


Рис. 11.6. Работа метода деформируемого многогранника на функции Розенброка

Поиск по образцу

Другой популярный метод прямого поиска был предложен Хуком и Дживсом (Hooke — Jeeves) в 1962 г., он называется *поиском по образцу (pattern search)* или *методом конфигураций* [27, с. 157]; [21, с. 130].

Алгоритм вначале обследует окрестность текущей точки, изменяя на фиксированную величину $\pm\Delta$ компоненты вектора $\overrightarrow{X}_k$. Эту процедуру можно уподобить действиям путника в темноте, который, прежде чем сделать очередной шаг, ощупывает посохом землю, пытаясь определить направление спуска по склону горы.

После того как будет найдено понижающее направление, в этом направлении производятся дробные шаги при постепенно увеличивающейся длине шага, тем самым устанавливается *конфигурация* поиска. Это продолжается до тех пор, пока поиск в направлении установленной конфигурации дает уменьшение це-

левой функции $f(\overrightarrow{X})$. Если же не удается найти точку с меньшим значением функции, размер шага уменьшается. После нескольких последовательных сокращений размера шага от принятой конфигурации отказываются и предпринимается новое обследование окрестности. Метод позволяет во время фазы обследования восстанавливать направление движения вдоль оврага в тех случаях, когда вследствие искривления его дна установленная ранее конфигурация перестает быть эффективной.

11.3. Градиентные методы

Переходим к рассмотрению типичных методов первого порядка. Они иначе называются *градиентными*, потому что предполагают, что в любой точке $\overrightarrow{X}_k$ наряду со значением целевой функции $f(\overrightarrow{X}_k)$ известно направление вектора градиента $\overrightarrow{\nabla} f(\overrightarrow{X}_k)$.

Градиентный метод с малым шагом

Поскольку ограничения отсутствуют, а градиент указывает направление скорейшего возрастания целевой функции в данной точке, то самым простым и очевидным способом выбора понижающего направления на каждом шаге является выбор направления антиградиента. Тогда итеративная процедура поиска минимума будет описываться следующими соотношениями:

$$\begin{aligned}
&\overrightarrow{d}_k = -\overrightarrow{\nabla} f(\overrightarrow{X}),\\
&t_k = \delta,\\
&\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k \overrightarrow{d}_k,
\end{aligned}$$

где δ — выбранная длина шага. В пределе длину шага δ можно считать бесконечно малой, в этом случае мы будем иметь непрерывную модель скатывания безынерционного (с нулевой массой) шарика по рельефу целевой функции. Траектория его движения $\overrightarrow{X}(t)$ описывается дифференциальным уравнением

$$\frac{d\overrightarrow{X}(t)}{dt} = -\overrightarrow{\nabla} f(\overrightarrow{X}). \tag{11.2}$$

Шарик будет катиться до тех пор, пока есть уклон, и в конце концов остановится в точке, где градиент целевой функции равен нулю, то есть в стационарной точке. Как мы знаем, для выпуклой функции стационарность является необходимым и достаточным условием экстремума.

В принципе, любой численный метод решения уравнения (11.2) даст нам метод нелинейной оптимизации без ограничений, однако на практике это была бы стрельба из пушек по воробьям, так как методы решения дифференциальных уравнений рассчитаны на вычисление формы траектории $\overrightarrow{X}(t)$, а нам нужна только ее конечная точка. Тем не менее с теоретической точки зрения модель движения шарика представляет интерес, на ее основе можно сконструировать оригинальный метод невыпуклой оптимизации, так называемый метод тяжелого шарика, который мы рассмотрим в разд. 11.5.

Метод скорейшего спуска

Более рациональным с практической точки зрения представляется метод, при котором понижающее направление выбирается по антиградиенту, однако шаг совершается на полную длину, то есть до тех пор, пока функция убывает в выбранном направлении. Это и есть классический *метод скорейшего спуска (steepest descent)*, предложенный в XIX веке французским математиком Огюстеном Луи Коши:

$$\begin{aligned}
&\overrightarrow{d}_k = -\overrightarrow{\nabla} f(\overrightarrow{X}_k),\\
&t_k = \arg\min_t f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k),\\
&\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k\,\overrightarrow{d}_k.
\end{aligned}$$

Метод скорейшего спуска очень хорошо изучен, прост в реализации, его принципиальная сходимость для выпуклых функций строго доказана. К сожалению, он, как и метод Гаусса — Зайделя, плохо работает на овражных функциях. Причина этого явления та же самая: легко показать, что направления движения на соседних шагах всегда ортогональны друг другу.

Действительно, если обозначить одномерное сечение целевой функции $\varphi(t) = f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k)$, то условие минимума этого сечения записывается в виде уравнения

$$\frac{\partial\varphi(t)}{\partial t} = \overrightarrow{\nabla}^T f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k)\,\overrightarrow{d}_k = \overrightarrow{\nabla}^T f(\overrightarrow{X}_{k+1})\,\overrightarrow{d}_k = \overrightarrow{d}_{k+1}^T\,\overrightarrow{d}_k = 0.$$

Равенство нулю скалярного произведения означает ортогональность соседних направлений.

Представьте себе лыжника с нулевой массой, который катится по склону оврага по прямой до тех пор, пока есть наклон, и умеет поворачивать лыжи только под прямым углом. Тогда вместо движения вдоль оврага он будет выписывать зигзаги, переезжая с одного его склона на другой.

П р и м е р. На рис. 11.7 показаны первые 50 итераций скорейшего спуска для функции Розенброка из начальной точки $\overrightarrow{X}_0 = (0.5,\, 0.5)^T$. Видно, что из-за ортогональности соседних направлений этому методу трудно даются повороты оврагов. ▲

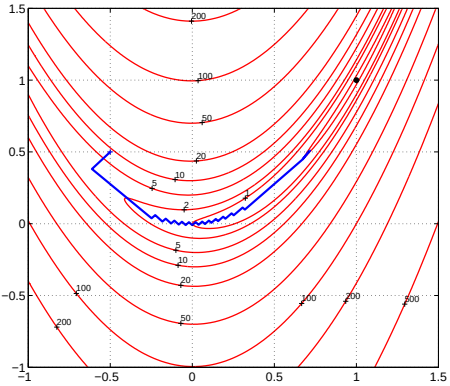


Рис. 11.7. Траектория поиска минимума функции Розенброка методом скорейшего спуска. После 50 итераций процесс все еще далек от точки минимума

Метод сопряженных градиентов

Для того чтобы ускорить движение к минимуму, нужно научить нашего воображаемого лыжника адаптироваться к местности и поворачивать лыжи не под прямым углом, а так, чтобы следующее направление соотносилось с направлением оврага. Подсказать идею нам поможет рис. 11.8, на котором крупным планом показаны два соседних шага градиентного спуска.

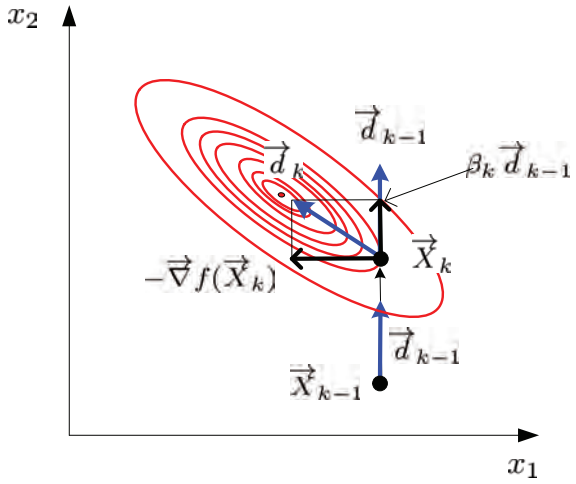


Рис. 11.8. Два соседних шага полношагового градиентного метода

Пусть в предыдущей точке $\vec{X}_{k-1}$ было выбрано некоторое направление $\vec{d}_{k-1}$. Двигаясь в этом направлении и дойдя до минимума, мы пришли в текущую точку $\vec{X}_k$. Рассмотрим в этой точке два направления: прежнее $\vec{d}_{k-1}$ и направление антиградиента $-\vec{\nabla} f(\vec{X}_k)$. Как мы установили выше, они ортогональны, движение по антиградиенту привело бы нас к длительному блужданию по склокам оврага. «Правильное» направление $\vec{d}_k$ на текущем шаге есть нечто промежуточное между старым направлением и чистым антиградиентом, его можно представить в виде линейной

комбинации

$$\overrightarrow{d}_k = -\overrightarrow{\nabla} f(\overrightarrow{X}_k) + \beta_k \overrightarrow{d}_{k-1}, \tag{11.3}$$

где β_k — некоторый весовой коэффициент. Таким образом, задача коррекции градиента на каждом шаге свелась к выбору соответствующего коэффициента.

Осталось выяснить самое главное — каково должно быть это «правильное» направление? Поскольку для произвольных целевых функций на этот вопрос ответить невозможно, авторы оптимизируют алгоритмы под некоторую известную функцию. Стандартным источником идей является к в а д р а т и ч н а я функция $f(\overrightarrow{X}) = \overrightarrow{c}^T \overrightarrow{X} + \overrightarrow{X}^T D \overrightarrow{X}$, поскольку любая выпуклая дважды дифференцируемая функция в локальной области аппроксимируется параболоидом.

Этот подход реализован в методе *сопряженных градиентов (conjugate gradients)* Флетчера — Ривса [27, с. 98]. Доказано, что если в полношаговом релаксационном методе последовательно использовать так называемые *сопряженные направления (conjugate directions)*, то точный минимум квадратичной функции будет найден не более чем за n шагов. При этом для того чтобы последующее направление было сопряжено предыдущему, весовой коэффициент в выражении (11.3) должен быть равным

$$\beta_k = \frac{\| \overrightarrow{\nabla} f(\overrightarrow{X}_k) \|^2}{\| \overrightarrow{\nabla} f(\overrightarrow{X}_{k-1}) \|^2}.$$

Если оптимизируемая функция отличается от квадратичной, то итеративная процедура может продолжаться неограниченно, однако для выпуклых функций сходимость метода сопряженных градиентов доказана.

Напомним, что два вектора $\overrightarrow{d}_1$ и $\overrightarrow{d}_2$ называются сопряженными относительно квадратичной формы D, если $\overrightarrow{d}_1^T D \overrightarrow{d}_2 = 0$, то есть понятие сопряженности является обобщением ортогональности.

П р и м е р. Оценим работоспособность метода Флетчера — Ривса на функции Розенброка (рис. 11.9). Видно, что в отличие от метода скорейшего спуска алгоритм достаточно хорошо приспосабливается к искривлению дна оврага и обеспечивает приемлемую точность нахождения минимума за относительно небольшое число шагов. ▲

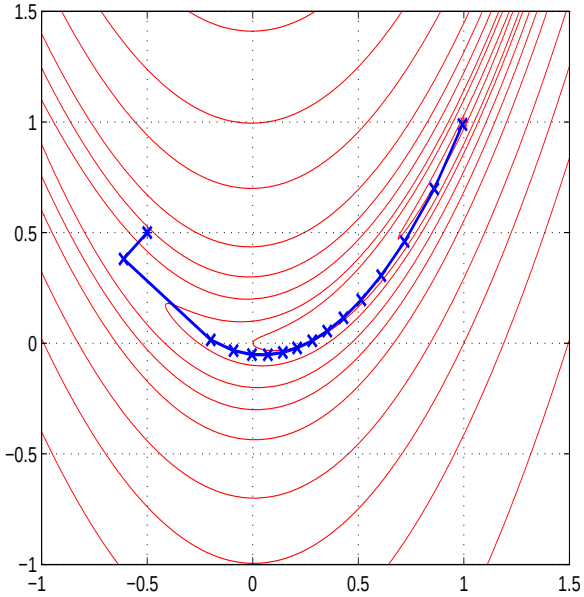


Рис. 11.9. Траектория поиска минимума функции Розенброка методом Флетчера — Ривса. Проведено 15 итераций. Начальная точка $(0.5,\ 0.5)^T$, конечная точка $(0.9939,\ 0.9883)^T$

11.4. Ньютоновские и квазиньютоновские методы

Для использования многомерных методов второго порядка необходимо иметь аналитические выражения для самой оптимизируемой функции $f(\overrightarrow{X})$, вектора градиента $\overrightarrow{\nabla} f(\overrightarrow{X})$ и матрицы Гессе вторых производных $\mathrm{H}(\overrightarrow{X})$.

Классический метод Ньютона

Данный метод является непосредственным обобщением одномерного метода Ньютона на многомерный случай.

Пусть известно приближенное значение минимума на k-й итерации $\overrightarrow{X}_k$. Если функция $f(\overrightarrow{X})$ является дважды дифференцируемой, то в окрестности точки $\overrightarrow{X}_k$ она может быть разложена в многомерный ряд Тейлора и аппроксимирована параболоидом

$$Q(\overrightarrow{X}) = f(\overrightarrow{X}_k) + \overrightarrow{\nabla}^T f(\overrightarrow{X}_k)(\overrightarrow{X} - \overrightarrow{X}_k) + \frac{1}{2}(\overrightarrow{X} - \overrightarrow{X}_k)^T \mathrm{H}(\overrightarrow{X}_k)(\overrightarrow{X} - \overrightarrow{X}_k).$$

Точка минимума этого параболоида находится из уравнения

$$\frac{dQ(\overrightarrow{X})}{d\overrightarrow{X}} = \overrightarrow{\nabla} f(\overrightarrow{X}_k) + \mathrm{H}(\overrightarrow{X}_k)(\overrightarrow{X} - \overrightarrow{X}_k) = 0.$$

Отсюда новое приближение $\overrightarrow{X}_{k+1}$:

$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k - \mathrm{H}^{-1}(\overrightarrow{X}_k)\overrightarrow{\nabla} f(\overrightarrow{X}_k). \tag{11.4}$$

В одномерном случае имеем

$$\mathrm{H}^{-1}(\overrightarrow{X}_k) = \frac{1}{f''(x_k)}, \quad \overrightarrow{\nabla} f(\overrightarrow{X}_k) = f'(x_k),$$

и выражение (11.4) естественным образом переходит в (10.2).

Замечание. Если сравнить выражение (11.4) со стандартным правилом вычисления следующего приближения в релаксационном методе $\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k \overrightarrow{d}_k$, то можно увидеть, что классический метод Ньютона

на каждой итерации определяет не только *ньютоновское направление* $\overrightarrow{d}_k = -\mathrm{H}^{-1}(\overrightarrow{X}_k)\overrightarrow{\nabla} f(\overrightarrow{X}_k)$, но и длину шага t_k, всегда равную единице.

Итерационную формулу (11.4) можно получить другим способом, не через квадратичную аппроксимацию целевой функции, а путем численного решения системы нелинейных уравнений.

Пусть система уравнений задана в виде

$$\begin{cases} F_1(\overrightarrow{X}) = 0, \\ \dots \\ F_n(\overrightarrow{X}) = 0. \end{cases}$$

С использованием понятия вектор-функции $\overrightarrow{F}(\overrightarrow{X}) = \left(F_1(\overrightarrow{X}), \dots, F_n(\overrightarrow{X})\right)^T$ эта система записывается в виде одного векторного уравнения $\overrightarrow{F}(\overrightarrow{X}) = 0$.

Если функции $F_i(\overrightarrow{X})$ дифференцируемы, можно построить квадратную функциональную матрицу частных производных, называемую *матрицей Якоби (якобианом)*[3]:

$$\mathrm{J}(\overrightarrow{X}) = \frac{\partial \overrightarrow{F}(\overrightarrow{X})}{\partial \overrightarrow{X}} = \begin{pmatrix} \frac{\partial F_1(\overrightarrow{X})}{\partial x_1} & \dots & \frac{\partial F_1(\overrightarrow{X})}{\partial x_n} \\ \dots & \dots & \dots \\ \frac{\partial F_n(\overrightarrow{X})}{\partial x_1} & \dots & \frac{\partial F_n(\overrightarrow{X})}{\partial x_n} \end{pmatrix}.$$

[3] Для вектор-функции $\overrightarrow{F}(\overrightarrow{X}) = \left(F_1(\overrightarrow{X}), \dots, F_m(\overrightarrow{X})\right)^T$ произвольной размерности m матрица Якоби имеет размерность $m \times n$, а транспонированная матрица

$$\mathrm{J}^T(\overrightarrow{X}) = \left(\frac{\partial \overrightarrow{F}(\overrightarrow{X})}{\partial \overrightarrow{X}}\right)^T = \begin{pmatrix} \frac{\partial F_1}{\partial x_1} & \dots & \frac{\partial F_m}{\partial x_1} \\ \dots & \dots & \dots \\ \frac{\partial F_1}{\partial x_n} & \dots & \frac{\partial F_m}{\partial x_n} \end{pmatrix} = \nabla \overrightarrow{F}(\overrightarrow{X})$$

может быть интерпретирована как *градиент вектор-функции* $\nabla \overrightarrow{F} = (\overrightarrow{\nabla} F_1 \dots, \overrightarrow{\nabla} F_m)$. Для того чтобы подчеркнуть, что он является матрицей, стрелка над оператором ∇ здесь не ставится. Если F — скалярная функция, то матричный градиент превращается в обычный вектор-столбец.

Тогда классический ньютоновский итеративный процесс нахождения корня уравнения запишется в виде

$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k - \mathrm{J}^{-1}(\overrightarrow{X}_k)\overrightarrow{F}(\overrightarrow{X}_k), \tag{11.5}$$

при этом определитель матрицы Якоби предполагается отличным от нуля.

Замечание. В одномерном случае, когда $\overrightarrow{F}(\overrightarrow{X}) = f(x)$, матрица Якоби превращается в простую производную, и итерационная формула (11.5) упрощается до знакомого нам элементарного выражения (10.3) $x_{k+1} = x_k - f(x_k)/f'(x_k)$.

Приведенную здесь процедуру поиска корня векторного уравнения легко приспособить для численного нахождения минимума функции многих переменных без ограничений, для чего нужно решить уравнение $\overrightarrow{\nabla} f(\overrightarrow{X}) = 0$. В этом случае $\overrightarrow{F} = \overrightarrow{\nabla} f$ и, как легко видеть, якобиан превращается в гессиан, т. е. $\nabla\overrightarrow{F} = \nabla(\overrightarrow{\nabla} f) = \nabla^2 f = \mathrm{H}$. Формула (11.5) естественным образом переходит в (11.4).

Таким образом, существует тесная связь между итеративной процедурой минимизации функции $f(\overrightarrow{X})$, аппроксимируемой параболоидом, и процессом нахождения корня векторного уравнения $\overrightarrow{\nabla} f(\overrightarrow{X}) = 0$.

П р и м е р. На рис. 11.10 показана траектория поиска минимума функции Розенброка классическим методом Ньютона. Несмотря на некоторую нерегулярность соседних точек, процесс сошелся очень быстро: для нахождения экстремума с точностью до 10^{-6} потребовалось всего 5 итераций. ▲

Классический метод Ньютона имеет два очевидных достоинства:

- он хорошо изучен теоретически, прост в реализации и быстро сходится на выпуклых функциях, а при квадратичной целевой функции дает решение всего за один шаг;

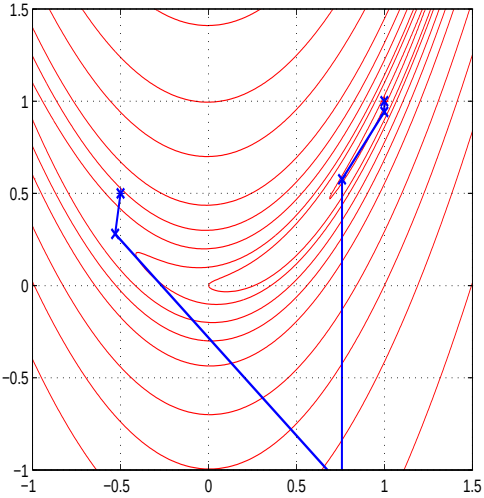


Рис. 11.10. Траектория поиска минимума функции Розенброка классическим методом Ньютона. Проведено 5 итераций. Начальная точка $(0.5,\ 0.5)^T$, конечная точка $(1,\ 1)^T$

- он не использует процедуру одномерной оптимизации вдоль сечения и поэтому не требует грубой локализации экстремума. Однако, как мы увидим далее, это достоинство метода может обернуться его недостатком.

С другой стороны, при практическом использовании метода в реальных условиях обнаруживается ряд недостатков:

- метод весьма чувствителен к сложности оптимизируемой функции. Если рельеф реальной целевой функции сильно изрезан, то квадратичное приближение в большой области становится слишком грубым. Тогда ньютоновское направление на точку минимума аппроксимирующего параболоида может оказаться далеким от направления на истинный экс-

тремум, шаг на единичную длину уводит поиск далеко в сторону;

- если целевая функция во всей области определения невыпукла, то матрица Гессе на некоторой итерации может потерять свойство положительной определенности, в результате ньютоновское направление не будет понижающим.

Указанные обстоятельства могут сделать классический ньютоновский итерационный процесс нестабильным. Иллюстрацией к сказанному является выброс после первой итерации в предыдущем примере (рис. 11.10). К счастью, на следующих итерациях поиск вернулся в область притяжения экстремума и итеративный процесс сошелся, однако на более сложной функции либо при другой начальной точке исход мог бы быть неблагоприятным.

Еще одним существенным недостатком является необходимость аналитического вычисления не только градиента, но и матрицы вторых производных, что может быть весьма затруднительно для сложной целевой функции.

По указанным причинам имеется большое число различных модификаций классического метода Ньютона, образующих обширный класс *ньютоноподобных методов (Newton-like methods)*, которые, сохраняя общую его структуру, пытаются избавиться от указанных недостатков.

Метод Ньютона с линейным поиском

Самой очевидной является модификация, при которой используется только ньютоновское направление, а длина шага определяется обычным линейным поиском вдоль выбранного направления:

$$\begin{aligned}
&\overrightarrow{d}_k = -\mathrm{H}^{-1}(\overrightarrow{X}_k)\overrightarrow{\nabla} f(\overrightarrow{X}_k),\\
&t_k = \arg\min_t f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k),\\
&\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k\,\overrightarrow{d}_k.
\end{aligned}$$

На рис. 11.11 приведена траектория поиска экстремума функции Розенброка методом Ньютона с линейным поиском. Видно, что путь поиска стал более регулярным.

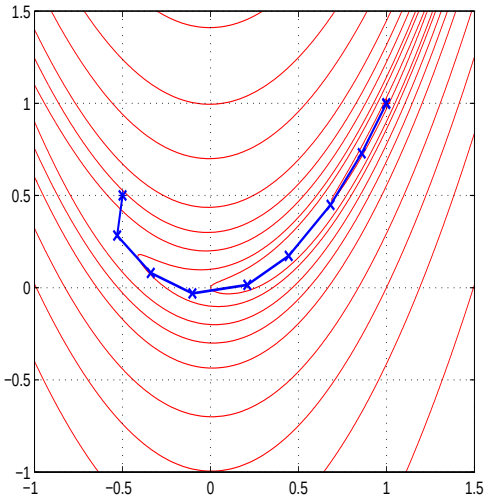


Рис. 11.11. Траектория поиска минимума функции Розенброка ньютоновским методом с линейным поиском, 9 итераций. Начальная и конечная точки те же

О методах доверительной области

Другой подход основан на принудительном ограничении ньютоновской длины шага. Он приводит к *методам с ограниченным шагом (restricted step methods)*, называемых также *методами доверительной области (trust-region methods).*

Особенность этих методов в том, что если в обычной ньютоновской процедуре аппроксимация распространяется на все пространство, то здесь аппроксимирующая (модельная) квадратичная функция $Q(\overrightarrow{X})$ должна достаточно правдоподобно отображать поведение $f(\overrightarrow{X})$ только в окрестности радиуса r_k, окружающей текущую точку $\overrightarrow{X}_k$. Эта окрестность называется *доверитель-*

ной областью (*trust region*). На каждом шаге ставится вспомогательная оптимизационная задача (*trust region subproblem*) нахождения такого шага $\overrightarrow{d}_k$, который минимизирует модельную функцию в пределах доверительной области:

$$\overrightarrow{d}_k = \arg \min_{\|\overrightarrow{d}\| \leqslant r_k} Q(\overrightarrow{X}_k + \overrightarrow{d}), \tag{11.6}$$

следовательно, определяется не только направление движения, но и длина шага, которая при этом не может превышать r_k.

Можно показать[4], что данная задача эквивалентна нахождению шага в классическом методе Ньютона, в котором исходная матрица Гессе $\mathrm{H}(\overrightarrow{X}_k)$ заменена скорректированной матрицей, полученной добавлением некоторой положительной константы λ_k к диагональным элементам:

$$\begin{aligned}
&\overrightarrow{d}_k = -[\mathrm{H}(\overrightarrow{X}_k) + \lambda_k \mathrm{I}]^{-1} \overrightarrow{\nabla} f(\overrightarrow{X}_k),\\
&t_k = 1,\\
&\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k \overrightarrow{d}_k.
\end{aligned}$$

При этом, если $\mathrm{H}(\overrightarrow{X}_k)$ была неопределена, то при подходящем выборе λ_k скорректированная матрица будет положительно определенной, поиск стабилизируется.

Полная теория методов доверительной области, разработанная в конце XX века, довольно сложна (см., например, [28, 30]) и выходит за пределы нашего курса. На каждом шаге корректирующая константа λ_k, зависящая от выбранного размера доверительной области r_k, находится путем решения некоторого нелинейного уравнения, определяемого собственными числами матрицы Гессе $\mathrm{H}(\overrightarrow{X}_k)$. Поскольку нахождение полной системы собственных чисел для матрицы большого размера является трудоемкой процедурой, разработаны приближенные алгоритмы, основанные на аппроксимации многомерной задачи двумерной.

[4]Эта фундаментальная идея была высказана в знаменитой статье Д. Марквардта, опубликованной в 1963 г. (см. исторические замечания в конце главы).

Казалось бы, такие сложные методы имеют небольшую практическую ценность. Однако на деле оказалось все наоборот. Выяснилось, что они весьма эффективны для задач с очень высокой размерностью, насчитывающих многие сотни и тысячи переменных *(large scale problems)*. По этой причине методы доверительной области в сочетании с квазиньютоновским (см. далее) подходом широко используются в современных пакетах программ нелинейной оптимизации, о которых мы будем говорить в гл. 13.

Замечание. Ограничить размер шага в ньютоновском методе можно и обратным способом, априорно задавшись на k-й итерации величиной λ_k, тогда размер доверительной области r_k определится сам. Такой упрощенный подход приводит к методу Левенберга — Марквардта (Levenberg — Marquardt) [21, с. 227], разработанному еще в 1960-е годы.

На рис. 11.12 в качестве простого примера стабилизации приведена траектория поиска классического метода Ньютона на функции Розенброка, в котором исходная матрица Гессе скорректирована априорно заданной константой $\lambda_k = 2$, не зависящей от номера шага. Сравнивая ее с траекторией на рис. 11.10, видим избавление от резких бросков в стороны.

Квазиньютоновские методы

Обременительная обязанность знать и на каждой итерации обращать матрицу Гессе привела к созданию класса *квазиньютоновских (quasi-Newton)* методов, известных также под названием *методов переменной метрики.* По своей структуре они аналогичны ньютоновским методам с линейным поиском, но не требуют априорного знания матрицы Гессе, а восстанавливают ее по отсчетам градиента на соседних итерациях. В связи с этим их иногда называют методами «полуторного» порядка.

Исторически первым представителем этого класса является популярный метод Дэвидона — Флетчера — Пауэлла (Davidon — Fletcher — Powell), сокращенно ДФП (DFP), подробно описанный во всех учебниках по нелинейной оптимизации [17, с. 179]; [27, с. 122]; [7, с. 160]; [19, с. 110]; [21, с. 207]. Идея метода опять-таки основана на предположении, что целевая функция является квад-

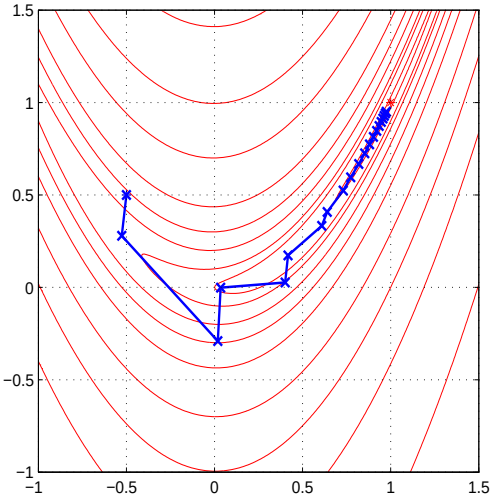


Рис. 11.12. Стабилизация траектории поиска минимума функции Розенброка методом Ньютона путем коррекции матрицы Гессе $\mathrm{H}' = \mathrm{H} + 2 \cdot \mathrm{I}$

ратичной. Алгоритм метода следующий.

П о д г о т о в и т е л ь н ы й э т а п

Выбираются точка $\overrightarrow{X}_0$ и матрица $A_0 = (a_{ij})_{n \times n}$. Обычно берут единичную матрицу $A_0 = \mathrm{I}$.

И т е р а ц и я $(k = 1, 2, \dots)$

Шаг 1. Вычисляется градиент целевой функции в точке $\overrightarrow{X}_{k-1}$ и определяется направление $\overrightarrow{d}_k = -A_{k-1} \overrightarrow{\nabla} f(\overrightarrow{X}_{k-1})$.

Шаг 2. Одномерный поиск:

$$t_k = \arg\min f(\overrightarrow{X}_{k-1} + t\,\overrightarrow{d}_k),$$

$$\overrightarrow{X}_k = \overrightarrow{X}_{k-1} + t_k \overrightarrow{d}_k.$$

Шаг 3. Производится *обновление (update)* матрицы A_k :

$$\overrightarrow{\alpha}_k = \overrightarrow{X}_k - \overrightarrow{X}_{k-1},$$
$$\overrightarrow{\beta}_k = \overrightarrow{\nabla} f(\overrightarrow{X}_k) - \nabla f(\overrightarrow{X}_{k-1}),$$
$$A_k = A_{k-1} + \frac{\overrightarrow{\alpha}_k \overrightarrow{\alpha}_k^T}{\overrightarrow{\alpha}_k^T \overrightarrow{\beta}_k} - \frac{A_{k-1} \overrightarrow{\beta}_k \overrightarrow{\beta}_k^T A_{k-1}}{\overrightarrow{\beta}_k^T A_{k-1} \overrightarrow{\beta}_k}.$$

Доказано, что для n-мерной квадратичной функции $f(\overrightarrow{X}) = \overrightarrow{c}^T \overrightarrow{X} + \overrightarrow{X}^T D \overrightarrow{X}$ метод сходится за n шагов, при этом $A_n = D^{-1}$.

Для выпуклой неквадратичной функции при $n \to \infty$ восстанавливается матрица, обратная матрице Гессе:

$$\overrightarrow{X}_k \to \overrightarrow{X}^*, \quad A_n \to \mathrm{H}^{-1}(\overrightarrow{X}^*).$$

Многочисленные экспериментальные исследования этого метода показали его хорошую сходимость и устойчивость работы на широком классе функций, поэтому он включен во многие пакеты программ прикладной оптимизации.

П р и м е р. Исследуем детально поведение метода ДФП при поиске минимума квадратичной функции, подробно рассмотренной в примере на с. 35:

$$f(x_1, x_2) = x_1^2 + 3x_2^2 + 2x_1x_2 - 4x_1 - 6x_2 + 4.5.$$

Для нее

$$\overrightarrow{\nabla} f(\overrightarrow{X}) = \begin{pmatrix} 2x_1 + 2x_2 - 4 \\ 6x_2 + 2x_1 - 6 \end{pmatrix}.$$

П о д г о т о в и т е л ь н ы й э т а п

Положим

$$\overrightarrow{X}_0 = \begin{pmatrix} -3 \\ 0,5 \end{pmatrix}, \quad A_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

И т е р а ц и я 0

Ш а г 1. Определяется направление:

$$\nabla\overrightarrow{X}_0 = \begin{pmatrix} -9 \\ -9 \end{pmatrix}, \quad \overrightarrow{d}_0 = -\begin{pmatrix} -9 \\ -9 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 9 \end{pmatrix}.$$

Ш а г 2. Одномерный поиск.
Построим одномерную функцию сечения

$$\varphi(t) = f(\overrightarrow{X}_0 + t\,\overrightarrow{d}_0) = f(-3+9t,\ 0.5+9t) = 486\,t^2 - 162\,t + 81/4.$$

Приравнивая нулю производную, получаем точку минимума:

$$\varphi'(t) = 972\,t - 162 = 0, \quad t_0 = 0.1666.$$

Отсюда

$$\overrightarrow{X}_1 = \overrightarrow{X}_0 + t_0\,\overrightarrow{d}_0 = \begin{pmatrix} -3 \\ 0.5 \end{pmatrix} + 0.1666\begin{pmatrix} 9 \\ 9 \end{pmatrix} = \begin{pmatrix} -1.5 \\ 2 \end{pmatrix}.$$

Ш а г 3. Пересчитывается матрица A:

$$\overrightarrow{\nabla} f(\overrightarrow{X}_1) = (-3,\ 3)^T,$$
$$\overrightarrow{\alpha} = \overrightarrow{X}_1 - \overrightarrow{X}_0 = (-1.5,\ 2)^T - (-3,\ 0.5)^T = (1.5,\ 1.5)^T,$$
$$\overrightarrow{\beta} = (-3,\ 3)^T - (-9,\ -9)^T = (6,\ 12)^T,$$
$$A_1 = \begin{pmatrix} 0.8833 & -0.3167 \\ -0.3167 & 0.2833 \end{pmatrix}.$$

И т е р а ц и я 1

Ш а г 1. Определяется направление:

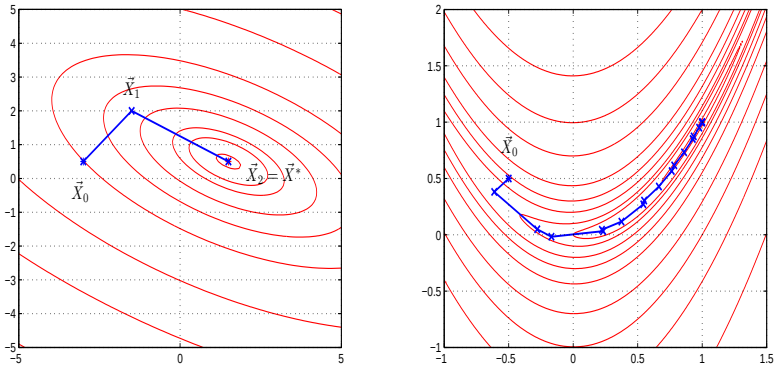
$$\overrightarrow{d}_1 = -\begin{pmatrix} -3 \\ 3 \end{pmatrix}\begin{pmatrix} 0.8833 & -0.3167 \\ -0.3167 & 0.2833 \end{pmatrix} = \begin{pmatrix} 3.6 \\ -1.8 \end{pmatrix}.$$

Ш а г 2. Одномерный поиск (далее арифметические выкладки опускаем):

$$t_1 = 0.8333, \quad \overrightarrow{X}_2 = (1.5,\ 0.5)^T.$$

Ш а г 3. Пересчитывается матрица A:

$$\overrightarrow{\nabla} f(\overrightarrow{X}_2) = (0,0)^T,$$
$$\overrightarrow{\alpha} = \overrightarrow{X}_2 - \overrightarrow{X}_1 = (3,\ -1.5)^T,$$
$$\overrightarrow{\beta} = (3,-3)^T,$$
$$A_2 = \begin{pmatrix} 0.75 & -0.25 \\ -0.35 & 0.25 \end{pmatrix}.$$



а) *б)*

Рис. 11.13. Траектория поиска минимума квадратичной функции ($а$) и функции Розенброка ($б$) методом Дэвидона — Флетчера — Пауэлла

Как видим, для точного нахождения минимума (об этом свидетельствует равенство нулю градиента в точке $\overrightarrow{X}_2$) потребовалось ровно две итерации (см. рис. 11.13, $а$). При этом точно восстановилась матрица Гессе квадратичной целевой функции $f(\overrightarrow{X}) = \overrightarrow{X}^T D \overrightarrow{X} + \overrightarrow{C}^T \overrightarrow{X} + a$:

$$A_2^{-1} = \begin{pmatrix} 2 & 2 \\ 2 & 6 \end{pmatrix} = 2D = \mathrm{H}(\overrightarrow{X}_2).$$

Идеально приспособленный для квадратичных функций, метод Дэвидона — Флетчера — Пауэлла оказался весьма эффективным и для плохо обусловленных задач минимизации со сложным овражным рельефом. На рис. 11.13, *б* представлена траектория поиска минимума функции Розенброка. Уже после 17-й итерации получено значение $\overrightarrow{X}_{18} = (0.9998,\ 0.9995)^T$, при этом

$$A_{17} = \begin{pmatrix} 0.5031 & 1.0056 \\ 1.0056 & 2.0148 \end{pmatrix} \quad A_{17}^{-1} = \begin{pmatrix} 786 & -392 \\ -392 & 196 \end{pmatrix}.$$

Видим, что матрица Гессе в оптимальной точке восстановилась неплохо: в примере на с. 102 мы получали ее точное значение

$$\mathrm{H}(\overrightarrow{X}^*) = \mathrm{H}(1,1) = \begin{pmatrix} 802 & -400 \\ -400 & 200 \end{pmatrix}. \ \blacktriangle$$

Метод ДФП является исторически первым, но не единственным представителем квазиньютоновских методов. Существует еще несколько методов, аналогичных по структуре, но несколько отличающихся формулами обновления матрицы Гессе. По мнению ряда исследователей, наилучшим из них является метод Бройдена — Флетчера — Гольдфарба — Шанно (Broyden — Fletcher — Goldfarb — Shanno), сокращенно БФГШ (BFGS) [7, с. 160]. Матрица Гессе (точнее, обратная ей матрица) для него обновляется по формуле

$$A_k = A_{k-1} + \left[1 + \frac{\overrightarrow{\beta}_k^T A_{k-1} \overrightarrow{\beta}_k}{\overrightarrow{\alpha}_k^T \overrightarrow{\beta}_k}\right] \frac{\overrightarrow{\alpha}_k \overrightarrow{\alpha}_k^T}{\overrightarrow{\alpha}_k^T \overrightarrow{\beta}_k} - \\ - \frac{\overrightarrow{\alpha}_k \overrightarrow{\beta}_k^T A_{k-1} + A_{k-1} \overrightarrow{\beta}_k \overrightarrow{\alpha}_k^T}{\overrightarrow{\alpha}_k^T \overrightarrow{\beta}_k}.$$

Метод БФГШ очень хорошо зарекомендовал себя на практике и включен во многие библиотеки стандартных программ.

Конечно-разностная аппроксимация градиента

На практике нередко приходится сталкиваться с задачами минимизации функций, которые хотя и являются гладкими, но настолько сложны, что аналитическое вычисление производных оказывается проблематичным. В этом случае можно поступать двояким образом: либо использовать методы нулевого порядка, например метод деформируемого многогранника, либо воспользоваться каким-нибудь из методов первого порядка, подменив истинный градиент $\overrightarrow{\nabla} f(\overrightarrow{X})$ его конечно-разностной аппроксимацией.

Идея аппроксимации является достаточно очевидной, но, к сожалению, нетривиальной, поскольку при численном дифференцировании возникает проблема чувствительности конечно-разностных приближений к ошибкам вычислений. Подробный анализ показывает (см., например, [7, с. 174]), что для каждой функции $f(\overrightarrow{X})$ и каждой схемы приближенных вычислений, определяемой прежде всего длиной разрядной сетки компьютера, существуют оптимальные значения приращений h_j, минимизирующих ошибки вычисления оценок составляющих градиента.

Эти оценки можно производить как по обычной формуле

$$\frac{\partial f(\overrightarrow{x})}{\partial x_j} = \frac{1}{h_j}[f(\overrightarrow{X} + h_j \overrightarrow{e}_j) - f(\overrightarrow{X})],$$

которая называется формулой *правых (прямых) разностей*, так и по формуле *центральных разностей*

$$\frac{\partial f(\overrightarrow{x})}{\partial x_j} = \frac{1}{2h_j}[f(\overrightarrow{X} + h_j \overrightarrow{e}_j) - f(\overrightarrow{X} - h_j \overrightarrow{e}_j)],$$

которая является более устойчивой к ошибкам вычислений, когда функция имеет особенности.

Для действительно гладких функций метод первого порядка с конечно-разностной оценкой градиента оказывается более эффективным, чем метод нулевого порядка, так как он использует априорную информацию о непрерывности производных. Особенно интересной является его комбинация с квазиньютоновскими методами, которые в этом случае используют конечно-разностные оценки градиента для аппроксимации матрицы Гессе[5]. Такой подход реализован в некоторых промышленных алгоритмах, о которых мы будем говорить в гл. 13.

Исследование и сравнение методов

Для любого итеративного метода оптимизации принципиально важной является его сходимость к истинному решению. Делом чести каждого математика является теоретическое исследование факта и скорости сходимости предложенного им алгоритма на некотором классе оптимизируемых функций, этому вопросу посвящено великое множество научных статей, монографий и диссертаций.

К сожалению, практическая ценность таких результатов не столь велика. Сходимость большинства релаксационных методов доказана только для выпуклых гладких функций с дополнительными условиями на производные, которые трудно проверить на деле. Еще труднее теоретически оценить скорость сходимости методов, чаще всего она рассчитывается для простейших, например квадратичных, функций. Поэтому основным способом исследования алгоритмов нелинейной оптимизации является вычислительный эксперимент. Существуют специальные подборки тестовых функций, на которых происходит сравнение различных алгоритмов. В табл. 11.2 приведено несколько тестовых функций для двух и четырех переменных, первая из них — уже упоминавшаяся классическая «банановая» функция Розенброка. Минимальные значения везде нулевые. Эти функции позаимствованы из книги Хим-

[5]Такое сочетание можно было бы назвать квази-квазиньютоновским методом или методом «половинного» порядка, но в научной литературе этот термин не используется.

мельблау [27], в которой содержится богатейший экспериментальный материал по исследованию различных методов оптимизации.

Таблица 11.2. Тестовые функции для алгоритмов оптимизации без ограничений

k	$f_k(\overrightarrow{X})$	$\overrightarrow{X}^*$
1	$100(x_2 - x_1^2)^2 + (1 - x_1)^2$	$(1, 1)^T$
2	$(x_2 - x_1^2)^2 + (1 - x_1)^2$	$(1, 1)^T$
3	$(x_2 - x_1^2)^2 + 100(1 - x_1)^2$	$(1, 1)^T$
4	$100(x_2 - x_1^3)^2 - (1 - x_1)^2$	$(1, 1)^T$
5	$[1,5 - x_1(1 - x_2)]^2 + [2,25 - x_1(1 - x_2^2)]^2 + +[2,625 - x_1(1 - x_2^3)]^2$	$(3, \frac{1}{2})^T$
6	$100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + +(1 - x_3)^3 + 10,1(x_2 - 1)^2 + (x_4 - 1)^2 + +19,8(x_2 - 1)(x_4 - 1)$	$(-3, -1, -3, -1)^T$
7	$(x_1 + 10x_2) + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + +10(x_1 - x_4)^4$	$(-3, -1, 0, 1)^T$

11.5. Оптимизация невыпуклых функций

Как мы видели в предыдущих разделах, оптимизация даже выпуклых функций в ряде случаев представляет непростую задачу. В случае невыпуклых функций эта задача многократно усложняется. Характерной особенностью невыпуклых целевых функций является сложный рельеф с наличием нескольких локальных экстремумов, поэтому задачи невыпуклого программирования называют *многоэкстремальными*. Любая релаксационная процедура выпуклой оптимизации в лучшем случае сойдется к одному из них, при этом нет гарантии, что найденный локальный экстремум является глобальным.

Общего алгоритма решения многоэкстремальных задач не существует, все предлагаемые подходы являются эвристическими, пригодными для того или иного класса целевых функций.

На рис. 11.14 приведена одна из возможных классификаций методов невыпуклой оптимизации. Первым основанием для декомпозиции выбрана степень универсальности. Некоторые методы являются узкоспециализированными, целиком опирающимися на специфику конкретной целевой функции. Решая поставленную задачу и давая полезный практический результат, они мало что дают для общего понимания проблемы, поэтому специально изучать их не имеет смысла. Вместе с тем существуют хоть и специфические, но достаточно широкие классы задач, для которых могут быть предложены эффективные подходы к решению. Примером такого подхода является *динамическое программирование,* которое мы будем рассматривать в гл. 14.

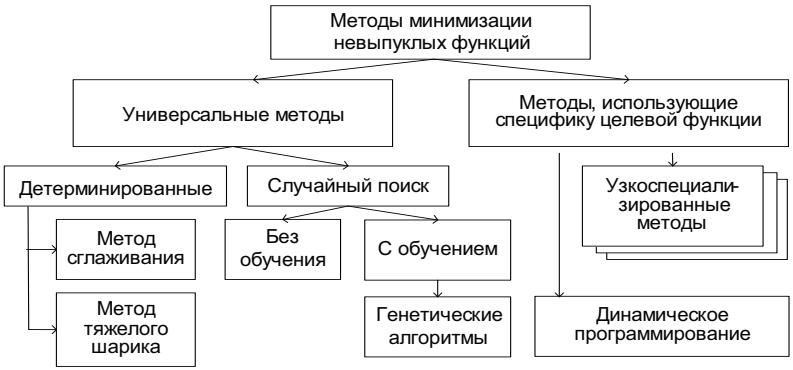


Рис. 11.14. Классификация методов невыпуклой оптимизации

Другие методы являются более или менее универсальными, пригодными для широкого класса оптимизируемых функций.

Теоретический анализ методов оптимизации в многоэкстремальном случае является еще более сложным делом, чем для выпуклых функций. Поэтому единственным доступным способом исследования является вычислительный эксперимент. Имеются спе-

циальные подборки многоэкстремальных тестовых функций [20], несколько таких функций приведено в табл. 11.3. Одна из них — функция Эккли — изображена на рис. 11.15.

Таблица 11.3. Тестовые функции для невыпуклой оптимизации

$f(\overrightarrow{X})$	$\overrightarrow{X}^*$
Функция Эккли (Ackley's function) $f(\overrightarrow{X}) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) -$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	$(0,\ldots,0)^T$ $f^* = 0$
Функция Растригина (Rastrigin's function) $f(\overrightarrow{X}) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$	$(0,\ldots,0)^T$ $f^* = 0$
Функция Голдстейна (Goldstein–Price function): $f(x_1,x_2) = [1+(1+x_1+x_2)^2 \cdot (19-14x_1+3x_1^2 -$ $-14x_2+6x_1x_2+3x_2^2)]\cdot[30+(2x_1-3x_2)^2(18-32x_1+$ $+12x_1^2+48x_2-36x_1x_2+27x_2^2)],\quad -2 \leqslant x_i \leqslant 2$	$(0,-1)^T$ $f^* = 3$

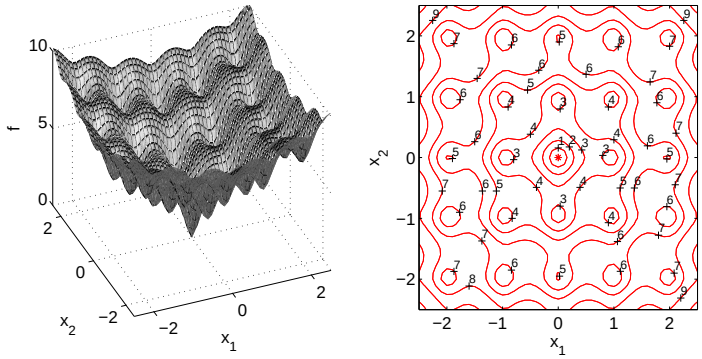


Рис. 11.15. Функция Эккли двух переменных

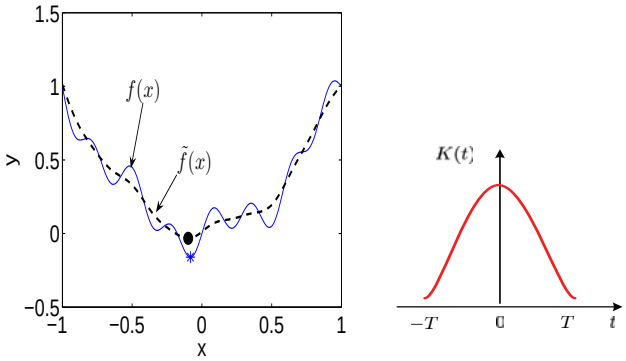
Методы сглаживания

Если целевая функция имеет ярко выраженный глобальный минимум, искаженный локальными неровностями рельефа (см. одномерный случай на рис. 11.16, *а*), то эти неровности можно нивелировать, приме-

няя процедуру скользящего сглаживания. В одномерном случае сглаженная целевая функция получается в виде

$$\tilde{f}(x) = \int_{-T}^{T} K(t) f(x+t) dt,$$

где $K(t)$ — сглаживающая (весовая, ядерная) функция, имеющая симметричную колоколообразную форму, подобную приведенной на рис. 11.16, *б*. Для этой цели часто используется гауссоида $K(t) = c \cdot \exp(-\alpha t^2)$, в которой α — параметр крутизны спада кривой, а c — нормирующий множитель.



а) *б)*

Рис. 11.16. Сглаживание невыпуклой одномерной функции

При удачно подобранных параметрах $\tilde{f}(x)$ становится унимодальной, а смещение глобального минимума x^* в точку $\tilde{x}$ незначительно. Как сказано в Писании: «Всякий дол да наполнится, всякая гора и холм да понизятся, кривизны выправятся, и неровные пути сделаются гладкими» [Исаия, гл. 40, ст. 4].

К сглаженной функции может быть применен любой метод выпуклой оптимизации, приводящий к $\tilde{x}$. Полученный минимум можно уточнить, запустив из этой точки алгоритм оптимизации исходной (несглаженной) целевой функцией $f(x)$. Если $\tilde{x}$ находил-

ся в области сходимости глобального экстремума x^*, то последний будет найден с допустимой погрешностью.

Поскольку большой точности вычисления сглаженной функции на требуется, интегрирование в практических вычислениях заменяют взвешенным суммированием в $2N+1$ равноотстоящих точках:

$$\tilde{f}(x) = \sum_{i=-N}^{N} K_i f(x + i\Delta), \ \sum_{i=-N}^{N} K_i = 1, \ K_i > 0.$$

В многомерном случае сглаживание происходит по соответствующей окрестности, например для функции двух переменных

$$\tilde{f}(x_1, x_2) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} K_{ij} f(x_1 + i\Delta, x_2 + j\Delta).$$

Замечание. Подобная процедура широко используется в машинной графике, когда требуется сгладить двумерную функцию яркости изображения с целью устранения царапин или других мелких деталей.

П р и м е р. Применим процедуру сглаживания к функции Эккли, установив следующие параметры: $N = 2, \Delta = 0.2$,

$$K_{ij} = \begin{pmatrix} 0.0084 & 0.0172 & 0.0228 & 0.0172 & 0.0084 \\ 0.0172 & 0.0460 & 0.0756 & 0.0460 & 0.0172 \\ 0.0228 & 0.0756 & 0.2512 & 0.0756 & 0.0228 \\ 0.0172 & 0.0460 & 0.0756 & 0.0460 & 0.0172 \\ 0.0084 & 0.0172 & 0.0228 & 0.0172 & 0.0084 \end{pmatrix}.$$

Результат сглаживания приведен на рис. 11.17. ▲

Метод тяжелого шарика

Рассматривая градиентные методы, мы пользовались аналогией с невесомым шариком, скатывающимся по рельефу целевой функции. Из-за отсутствия инерции он немедленно останавливается, как только крутизна склона, отражающая градиент целевой

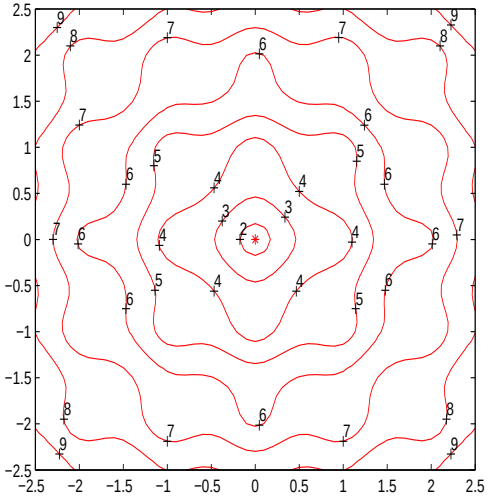


Рис. 11.17. Сглаженная функция Эккли

функции, становится равной нулю. По этой причине шарик не может преодолеть самое незначительное углубление на рельефе. Для того, чтобы шарик мог с ходу проскакивать небольшие локальные экстремумы и двигаться дальше, ему нужно придать массу.

Построим модель движения тяжелого шарика массой m в одномерном случае (см. рис. 11.18). Движущую силу приравняем производной целевой функции, взятой с обратным знаком (она направлена в сторону убывания). Кроме того, чтобы шарик в конце концов остановился, введем силу трения, пропорциональную скорости. Тогда траектория движения $x(t)$ шарика описывается дифференциальным уравнением, отображающим Второй закон Ньютона:

$$m\frac{d^2x(t)}{dt^2} + k\frac{dx(t)}{dt} = -f'(x)$$

с начальными условиями $x(0) = x_0, x'(0) = 0$, где k — коэффициент трения.

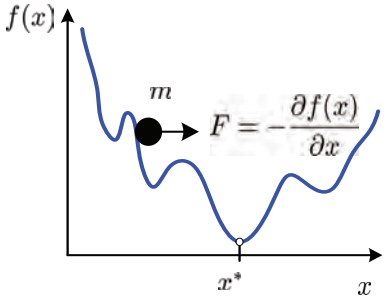


Рис. 11.18. Метод тяжелого шарика

В многомерном случае уравнение движения получается аналогично:

$$m\frac{d^2\overrightarrow{X}(t)}{dt^2} + k\frac{d\overrightarrow{X}(t)}{dt} = -\overrightarrow{\nabla} f(\overrightarrow{X}).$$

Уравнение движения можно решить любым численным методом и, устремляя $t \to \infty$, найти точку, в которой шарик остановится. Если параметры $m,\ k,\ X_0$ подобраны удачно, шарик скатится в точку глобального экстремума.

Для надежности поиск глобального экстремума следует повторить несколько раз с различными параметрами, а затем выбрать ту конечную точку, где значение целевой функции меньше.

П р и м е р. Для того чтобы описать движение тяжелого шарика по поверхности функции двух переменных, необходимо составить систему двух дифференциальных уравнений второго порядка:

$$m\frac{d^2x_1(t)}{dt^2} + k\frac{dx_1(t)}{dt} = -\frac{\partial f(x_1, x_2)}{\partial x_1},$$
$$m\frac{d^2x_2(t)}{dt^2} + k\frac{dx_2(t)}{dt} = -\frac{\partial f(x_1, x_2)}{\partial x_2}.$$

Если обозначить $y_1 = x_1(t),\ y_2 = x_2(t),\ y_3 = x_1'(t),\ y_4 = x_1'(t),\ g_1 = \frac{\partial f}{\partial x_1},\ g_2 = \frac{\partial f}{\partial x_2}$, то ее можно представить в виде экви-

валентной системы четырех дифференциальных уравнений первого порядка

$$\begin{cases} y_1' = y_3, \\ y_2' = y_4, \\ y_3' = \frac{1}{m}(-ky_3 - g_1), \\ y_4' = \frac{1}{m}(-ky_4 - g_2). \end{cases}$$

Для численного решения данной системы необходимо задать массу шарика m, коэффициент трения k, зафиксировать начальные условия (исходную точку и начальные скорости), а также установить верхний предел времени интегрирования.

На рис. 11.19 приведена одна из удачных траекторий движения тяжелого шарика по рельефу функции Эккли из начальной точки $\overrightarrow{X}_0 = (-1.5,\ 2)^T$, полученная при следующих условиях: $m = 7$, $k = 0.4$, $T = 80$. Видно, как шарик, благополучно миновав несколько локальных экстремумов, скатился во впадину, окружающую глобальный минимум, и продолжает крутиться вокруг него с убывающей скоростью. ▲

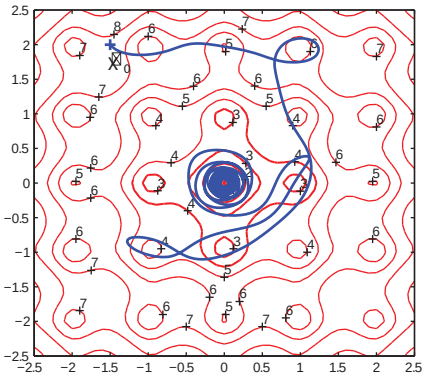


Рис. 11.19. Траектория движения тяжелого шарика по рельефу функции Эккли

Случайный поиск

Рассмотренные выше методы оказываются работоспособными в тех случаях, когда рельеф многоэкстремальной целевой функции имеет ярко выраженный глобальный минимум, незначительно искаженный локальными экстремумами, которые преодолеваются сглаживающим фильтром или инерцией тяжелого шарика. Если же отдельные локальные минимумы сравнимы по своей глубине, то алгоритм может застрять в одном из них и глобальный экстремум не будет достигнут.

Непредсказуемый характер рельефа таких функций наводит на мысль о том, что со сложностью можно бороться случайностью. Методы, использующие генераторы случайных чисел для решения различных задач вычислительной математики, обычно называют *методами Монте-Карло* (см. исторические замечания в конце главы). Существует множество разнообразных методов случайного поиска, которые, согласно образному определению одного из авторов, можно подразделить на «сеющие» (глобальные) и «ползающие» (локальные).

Слепой поиск. Простейший глобальный алгоритм случайного поиска состоит в том, что область поиска, которая считается известной, с помощью генератора случайных чисел равномерно засевается случайными точками $\overrightarrow{X}_i,\ i = 1, \ldots, N$. В качестве решения берется точка с наилучшим значением целевой функции:

$$\overrightarrow{X}^* = \arg\min_{\overrightarrow{X}_i} f(\overrightarrow{X}_i).$$

Если число координат равно n, а относительную точность локализации экстремума по одной координате обозначить ε, то вероятность того, что хотя бы одна из N случайно сгенерированных точек попадет в ε-окрестность экстремума, равна

$$P = 1 - (1 - \varepsilon^n)^N.$$

Отсюда определится необходимое число испытаний при заданной

вероятности P:

$$N = \frac{\ln(1-P)}{\ln(1-\varepsilon^n)} \approx -\ln(1-P)\left(\frac{1}{\varepsilon}\right)^n.$$

Пусть заданная надежность поиска $P = 0.95$, а локализация экстремума должна проводиться с точностью 1%, т. е. $1/\varepsilon = 100$. Тогда $N \approx 3{\cdot}(100)^n$, т. е. в одномерном случае потребуется 300 случайных измерений целевой функции, в двумерном 30 000 и т. д. С увеличением размерности трудоемкость поиска резко возрастает, этот факт образно называют «проклятьем размерности».

Мультистарт. Значительно более эффективной является комбинация слепого поиска с любым регулярным методом выпуклой оптимизации, которая в работе [13] названа *мультистартом*. Здесь случайная точка $\overrightarrow{X}_i$, брошенная наудачу в область поиска, является исходной для релаксационного спуска к локальному минимуму. В результате нескольких попыток определяется наименьший из достигнутых минимумов. В данном методе для нахождения глобального экстремума достаточно попасть в область его притяжения.

Если в предыдущем примере предположить, что область притяжения по каждой из координат составляет 10%, то при той же надежности потребуется только $N \approx 3 \cdot (10)^n$ испытаний.

Локальный поиск без обучения. Самый простой «ползающий» случайный поиск состоит в следующем. Из текущей точки $\overrightarrow{X}_k$ делается несколько случайных пробных шагов $\overrightarrow{\xi}_i$, $i = 1, \ldots, N$. Распределение пробных шагов выбирается таким, чтобы более-менее равномерно охватывалась близлежащая область, но при этом должна быть достаточной вероятность и более далекого шага, дающего выскочить из локального минимума, если поиск в него привел. С этой целью можно использовать многомерное нормальное распределение $\overrightarrow{\xi}_i \sim N(\overrightarrow{a}, \Sigma)$, где $\overrightarrow{a}$ — некоторый вектор средних значений, возможно нулевой, а Σ — ковариационная матрица. Далее делается рабочий шаг в наиболее удачную

пробную точку:

$$\overrightarrow{d}_k = \arg\min_{\xi_i} f(\overrightarrow{X}_k + \overrightarrow{\xi}_i),$$
$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + \overrightarrow{d}_k.$$

Если после заданного числа N пробных шагов улучшения целевой функции не происходит, то поиск минимума считается завершенным.

П р и м е р. На рис. 11.20 показана одна из реализаций траектории случайного поиска глобального минимума функции Эккли. Параметры поиска: $N = 20,\ a = (0,0)^T, \Sigma = \begin{pmatrix} 0.64 & 0 \\ 0 & 0.64 \end{pmatrix}$. ▲

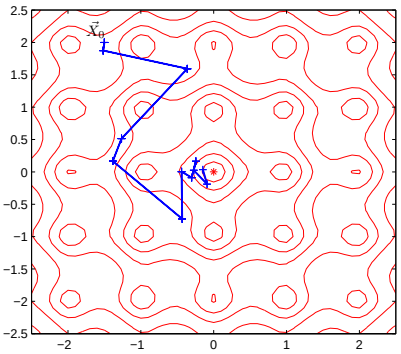


Рис. 11.20. Траектория случайного поиска минимума функции Эккли. Локальный поиск без обучения

Локальный поиск с обучением. Существенным преимуществом случайного поиска является возможность ввести механизмы самообучения. Например, в рассмотренном выше алгоритме можно после каждого рабочего шага корректировать вектор средних значений:

$$\overrightarrow{a}_{k+1} = \lambda \overrightarrow{d}_k + (1-\lambda)\overrightarrow{a}_k.$$

Тогда направление пробных шагов будет иметь тенденцию смещаться в сторону удачных рабочих шагов.

Генетические алгоритмы

Особый класс самообучающихся методов случайного поиска составляют *генетические алгоритмы,* которые моделируют процесс эволюции в живой природе. Первая формальная схема генетического алгоритма была предложена в 1975 г. Дж. Холландом, впоследствии это направление стало бурно развиваться [12]. О потенциальных возможностях генетических алгоритмов говорит сам факт появления высших разумных существ за относительно короткий исторический период.

Основной идеей генетических алгоритмов является организация «борьбы за выживание» среди случайных пробных точек $\overrightarrow{X}_i$. Пользуясь биологической терминологией, каждую такую точку будем называть *особью*, а их совокупность $\{\overrightarrow{X}_i,\ i = 1, \ldots, N\}$ — *популяцией.*

Процесс эволюции начинается с некоторой случайным образом созданной начальной совокупности пробных точек, затем имитируется искусственная жизнь популяции путем моделирования процессов размножения и гибели особей (рис. 11.21).

Первая фаза размножения — выбор родителей. Самый простой способ — *панмиксия*, когда родительские пары формируются случайным образом из всей популяции. В более сложных алгоритмах возможно моделирование брачных турниров и других тонкостей, взятых из реальной жизни.

Выбранная родительская пара $(\overrightarrow{X}_1, \overrightarrow{X}_2)$ производит на свет новую особь $\overrightarrow{X}$, при этом в алгоритме моделируются две основных движущих силы эволюции — наследственность и изменчивость.

Элементарной единицей наследственности является *ген,* несущий информацию о некотором свойстве особи, набор генов для популяции фиксирован. В зависимости от вида наследуемого свойства гены могут быть либо бинарными, отражающими в виде кодовой строки качественные характеристики, либо число-

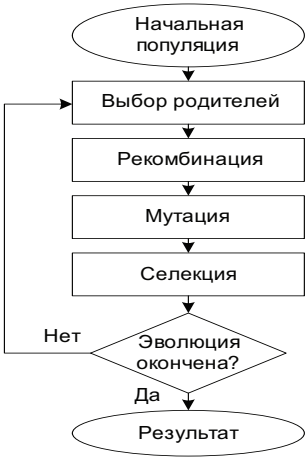


Рис. 11.21. Классический генетический алгоритм

выми, измеряющими количественные свойства. В нашей простейшей модели, когда особь представляет собой набор координат $\overrightarrow{X} = (x_1, \ldots, x_n)$, геном является одно вещественное число x_i.

Набор генов потомка формируется из одноименных генов родителей, этот процесс называется *рекомбинацией.* Принципиальной особенностью рекомбинации является наличие случайных факторов и, как следствие, непредсказуемость результата, таким образом природа обеспечивает биологическое разнообразие потомства, необходимое для успешной эволюции.

Существует множество моделей рекомбинации. Некоторые достаточно точно воспроизводят процессы, происходящие в живой природе. Подобно хромосомам две кодовых строки бинарных генов, разорвавшись в случайном месте, могут обменяться «хвостами». Такая рекомбинация называется *кроссинговером (crossingover).* Для вещественных генов есть свои модели рекомбинации. Например, при *промежуточной рекомбинации* вещественные гены потомка представляют собой случайную выпуклую ком-

бинацию одноименных генов родителей:

$$x_i = (1 - \xi_i)x_i^{(1)} + \xi_i x_i^{(2)}, \quad i = 1, \ldots, n,$$

где ξ_i — случайные числа, равномерно распределенные в $[0, 1]$.

Дополнительное разнообразие в популяцию вносит механизм *мутации* — случайное, не обусловленное наследственностью изменение гена. В природе мутация вызывается многими внешними факторами — радиоактивным излучением, воздействием химических веществ и др. Хотя для отдельной особи мутация грозит быть губительной, для популяции в целом она благоприятна, так как может привести к образованию особей, существенно отличающихся от родителей и выводящих популяцию за пределы локального экстремума. Для бинарного гена мутация может заключаться в случайной замене некоторой единицы на ноль или наоборот, для числового гена мутацию можно промоделировать, изменив его значение на случайную величину:

$$x_i = x_i + \xi.$$

Итак, в результате рекомбинации и возможной мутации образовалась новая особь, которая вливается в популяцию. Но так как размер популяции фиксирован, то некоторая особь должна покинуть ее, то есть погибнуть. Открытый Дарвином принцип естественного отбора заключается в том, что в конкурентной борьбе выживают наиболее приспособленные. В качестве меры *приспособленности (fitness)* особи $\overrightarrow{X}_i$ естественно использовать значение целевой функции $f(\overrightarrow{X}_i)$: чем она меньше, тем лучше. Процесс отбора особи для сохранения в популяции называется *селекцией.* Алгоритм селекции может быть детерминированным, когда популяцию покидает наименее приспособленная особь, либо стохастическим, когда она выбирается случайным механизмом, причем вероятность гибели устанавливается тем выше, чем менее степень приспособленности.

В результате нескольких циклов размножения-гибели формируется новая, в среднем более приспособленная популяция. Про-

цесс эволюции можно считать завершенным, когда после заданного числа итераций улучшения целевой функции не происходит. В качестве окончательного ответа из популяции выбирается особь с наименьшим значением целевой функции.

Генетические алгоритмы отличаются большим разнообразием и обладают многими достоинствами:

- они не критичны к локальным свойствам целевой функции (непрерывность, выпуклость, дифференцируемость, наличие помех);
- они пригодны для комбинаторных задач, в которых оптимизируемые переменные могут принимать нечисловые значения. Выбрав подходящий способ кодирования генов и механизм рекомбинации, можно получить достаточно эффективные алгоритмы для классических задач коммивояжера, составления расписаний или раскладки [12];
- они хорошо распараллеливаются и могут быть реализованы на многопроцессорных кластерах.

П р и м е р. Воспользовавшись пакетом генетических алгоритмов из системы MATLAB (см. пример на с. 227), рассмотрим процесс поиска глобального минимума функции Растригина

$$f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2).$$

На рис. 11.22 эта функция изображена в трехмерном представлении и в виде набора изолиний. По сравнению с тестовой функцией Эккли рельеф функции Растригина является более сложным для поиска: расположенные в узлах целочисленной решетки $\overrightarrow{X}_{ij} = (i, j)^T$, $i, j = \pm 1, 2, \ldots$, локальные минимумы со значениями $f(\overrightarrow{X}_{ij}) = 20 + i^2 + j^2$ почти не уступают глобальному в точке $\overrightarrow{X}^* = (0, 0)^T$.

Установим следующие основные параметры генетического алгоритма: численность популяции — 20; длительность эволюции

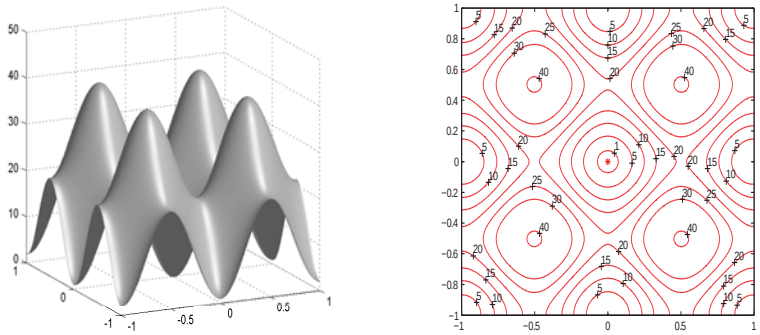


Рис. 11.22. Функция Растригина

(число поколений популяции) — 50; набор генов — два числовых гена, соответствующих координатам точки; способ выбора родителей — панмикси́я; тип рекомбинации — промежуточная; тип селекции — стохастическая.

Для демонстрации возможностей алгоритма начальная популяция сгенерирована как совокупность точек, равномерно распределенных в единичном квадрате $x_1, x_2 \in [1, 2]$, смещенном относительно точки глобального минимума. Поэтому промежуточная рекомбинация сама по себе не сможет вывести популяцию из начального региона. Для того чтобы вырваться из него, включена возможность мутации, при мутации к гену добавляется случайная величина, распределенная по нормальному закону с нулевым средним и единичной дисперсией.

На рис. 11.23 графически представлен процесс эволюции с заданными параметрами алгоритма. Показаны среднее и наилучшее значение *функции приспособленности (fitness function)*, то есть целевой функции для текущей популяции в зависимости от номера поколения.

Хорошо видно, что эволюция, подобно тому, как это происходит в живой природе, идет толчками. Первые четыре поколения обитали в области притяжения локального экстремума, попавше-

го в исходный регион, затем в результате мутации образовалась особь, потомки которой переместились в более глубокий локальный минимум (поколения 4–14) и т. д. В итоге к 50-му поколению популяция расположилась вблизи глобального экстремума, наиболее приспособленная особь имеет координаты $(0.013, -0.001)^T$. Такой точности нахождения глобального минимума более чем достаточно для того, чтобы из данной точки запустить любой локальный алгоритм. ▲

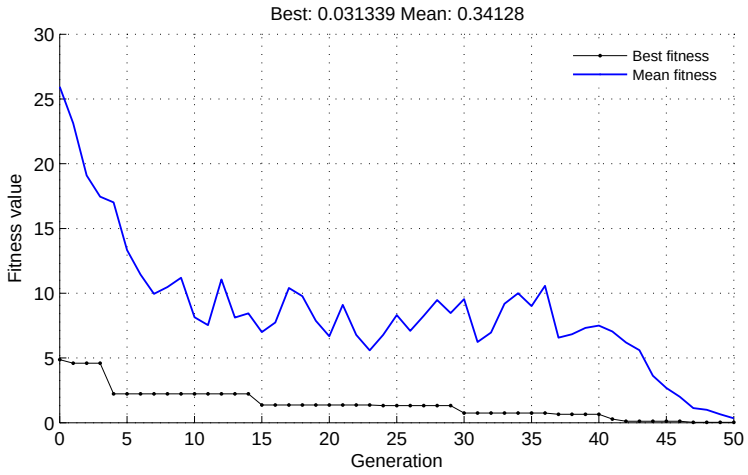


Рис. 11.23. Динамика работы генетического алгоритма поиска глобального минимума функции Растригина

Другие алгоритмы случайного поиска

Кроме генетических существуют еще немало оригинальных алгоритмов случайного поиска, многие из которых моделируют реальные явления живой и даже неживой природы. В качестве примера можно привести метод *имитации о́тжига (simulated annealing)*, который остроумно подражает физическим процессам, происходящим в металлическом изделии в процессе нагревания и медленного охлаждения. При этом происходит упорядочение кристаллической решетки и уменьшение

внутренней энергии вещества, отождествляемой с целевой функцией. Этот метод, изобретенный в середине 1980-х годов, развивает идеи Николаса Митрополиса (см. исторические замечания) по моделированию термодинамических систем, высказанные в 1953 г. Метод имитации отжига показал высокую эффективность и включен в некоторые пакеты оптимизации, в частности в последние версии MATLAB.

Изобретательность авторов новых эвристических методов невыпуклой оптимизации не знает границ. Известны алгоритмы, моделирующие поведение колонии муравьев — *ant colony optimization*, пчел — *bees algorithm,* процесс настройки музыкальных инструментов в оркестре — *harmony search* и т. д.

11.6. Исторические замечания

Мы рассматривали численные методы оптимизации в восходящем порядке — от нулевого ко второму, однако в истории математики все было наоборот.

Как уже отмечалось в замечаниях к предыдущей главе, первый численный метод, пригодный для нахождения корней функций и экстремумов, был разработан в конце XVII — начале XVIII века Исаакам Ньютоном, Джозефом Рафсоном и Томасом Симпсоном. Он основан на приближенном решении системы нелинейных уравнений и требует знания вторых производных.

Огюстен Луи Коши

Идея метода скорейшего спуска, использующего только градиент, была предложена сто лет спустя в 1847 г. великим французским математиком **Огюстеном Луи Коши** (Cauchy, Augustin Louis; 1789–1857). Он родился в семье чиновника, учился в Политехнической школе и парижской Школе мостов и дорог. По окончании стал инженером путей сообщения в Шербуре. Здесь начал самостоятельные математи-

ческие исследования, которые продолжил после того, как вернулся в столицу и стал членом Парижской академии.

Коши написал свыше 800 работ, полное собрание его сочинений содержит 27 томов. Он впервые дал строгое определение основным понятиям математического анализа: пределу, непрерывности, производной, дифференциалу, интегралу, сходимости ряда и т. д. Заложил основы теории функции комплексного переменного, получил фундаментальные результаты в теории дифференциальных уравнений в частных производных (задача Коши). Подобно Лагранжу, Коши удостоен чести быть включенным в список 72 величайших ученых Франции.

В начальном варианте метод скорейшего спуска был предложен Коши только для решения системы уравнений, в дальнейшем он был распространен на другие задачи. В первой половине XX века приложениями метода к разнообразным разделам вычислительной математики занимался будущий нобелевский лауреат Л. В. Канторович, известный нам как родоначальник линейного программирования.

Несмотря на отдельные достижения, в целом уровень развития численных методов оптимизации до второй половины XX века был невысоким, потому что задачи с несколькими переменными требовали очень большого объема вычислительной работы, который невозможно выполнить на механических калькуляторах.

Появление электронных вычислительных машин произвело настоящую революцию в прикладной математике, благодаря им появилась реальная возможность решения многомерных экстремальных задач. Такие задачи возникали в самых различных областях — физике, химии, технике, они требовали построения новых математических методов, способных справиться с их постоянно возрастающей сложностью.

В первые послевоенные годы (1945–1960) были созданы и реализованы на электронных вычислительных машинах методы линейного программирования, справлявшиеся с многотысячной размерностью. Что же касается нелинейных задач, то хотя в это вре-

мя были разработаны самые разнообразные численные методы оптимизации, основанные на идеях градиентного спуска по рельефу целевой функции (или «карабкания» в другой терминологии), их эффективность долгое время оставалась невысокой. На компьютерах тех лет реализованные алгоритмы давали возможность за разумное время решить задачу только с дюжиной переменных.

Прорыв произошел после создания квазиньютоновских методов, в десятки раз увеличивших размерность решаемых задач. Это открытие состоялось в начале 1960-х годов, оно было сделано несколькими американскими и английскими математиками, чьи имена запечатлены в названиях самых известных алгоритмов. Именно в этих странах в послевоенные годы сформировались крупные исследовательские центры, тесно сотрудничавшие с ведущими университетами и оснащенные передовой вычислительной техникой.

Вильям Дэвидон (Davidon, William; p. 1927) получил степень доктора философии по физике в Университете Чикаго. Работал в расположенной поблизости знаменитой Argonne National Laboratory, внесшей исключительно большой вклад в развитие атомной физики и технологии в США. В 1959 г. в техническом отчете лаборатории он предложил идею «метода переменной метрики». Копия неопубликованного отчета пересекла океан и попала в главный английский ядерный центр Atomic Energy Research Establishment в городе Harwell, недалеко от Лондона. Там она заинтересовала двух молодых ученых Роджера Флетчера и Майкла Пауэлла, которые сумели понять потенциал нового подхода. Благодаря их знаменитой статье, вышедшей в 1963 г., первый квазиньютоновский метод DFP стал известен научной общественности (оригинальный текст Дэвидона был опубликован только в 1991 г.).

Майкл Пауэлл (Powell, Michael; p. 1936) изучал математику в Кембриджском университете. После его окончания в 1959–1976 гг. работал в вычислительной лаборатории ядерного центра Harwell, разрабатывал библиотеку оптимизационных программ на автокоде и Фортране, затем вернулся в родной университет, где

преподавал до выхода на пенсию в 1996 г. Автор многих работ в области оптимизации и аппроксимации, кроме DFP его имя присутствует в названиях нескольких методов (например, метод сопряженных направлений Пауэлла).

Майкл Пауэлл

Роджер Флетчер

Роджер Флетчер (Fletcher, Roger; p. 1939) внес исключительно большой вклад в теорию нелинейной оптимизации, с его именем связана буква «F» в аббревиатурах DFP и BFGS, а также название метода сопряженных градиентов. Получил степень бакалавра по теоретической физике в 1960 г. в Кембриджском университете, степень доктора философии в 1963 г. в Университете города Лидс (Leeds), где в это время была образована одна из первых в Англии компьютерных лабораторий на базе ЭВМ Pegasus. Руководителем его диссертационной работы был Колин Ривс (Reeves, Colin M.), реализовывавший в то время проект создания библиотеки программ для решения задач квантовой химии. Флетчер 17 лет проработал в центре Harwell, затем перешел на работу в Университет Данди (Dundee) в Шотландии, где продолжил научную работу в области оптимизации. Опубликовал более 120 работ, его знаменитый двухтомник «*Practical Methods of Optimization*» [28] оказал большое влияние на развитие этой отрасли прикладной математики. Лауреат ряда престижных научных премий.

После выхода статьи Флетчера и Пауэлла квазиньютоновские методы стали интенсивно исследоваться и вскоре вытеснили другие релаксационные методы. В 1965 г. англичанин **Бройден** (Broyden, C. G.) из Университета Эссекс показал, что DFP —

только один из класса аналогичных методов. В 1970 г. Бройден, Флетчер, а также американцы **Гольдфарб** (Goldfarb, D.) из Нью-Йорка и **Шанно** (Shanno, D. F.) из Чикаго независимо друг от друга предложили метод, который впоследствии стал именоваться BFGS. Он по всем параметрам превзошел DFP и стал самым популярным квазиньютоновским методом. Уже в середине 1960-х годов с его помощью удавалось решать задачи с несколькими сотнями переменных.

Между тем практика требовала создания методов нелинейной оптимизации, способных справиться с еще большей размерностью. Например, такая задача возникает в математической статистике при сглаживании экспериментальных данных теоретическими кривыми по методу наименьших квадратов. Разработка соответствующих методов вылилась в отдельную ветвь теории оптимизации, основанную на новых идеях. Эпохальным событием в данном направлении стала знаменитая статья «An algorithm for least-squares estimation of nonlinear parameters» американского статистика **Дональда Марквардта** (Marquardt, Donald; 1929–1997), работавшего в американской корпорации DuPont, опубликованная в 1963 г. и занявшая 92-е место в рейтинге самых цитируемых научных публикаций за 1945–1988 гг. В этой работе, опираясь на результаты другого американского математика **Кеннета Левенберга,** опубликованные в 1944 г., был предложен способ стабилизации классического метода Ньютона, развившийся в дальнейшем в современные методы доверительной области. На сегодняшний день эти методы и основанные на них алгоритмы являются самыми употребительными для решения задач сверхбольшой размерности.

Не следует удивляться, что выдающийся математический результат был получен в химической компании. Дело в том, что повышение эффективности предприятий большой химии хотя бы на несколько процентов сулит чрезвычайно большой экономический эффект. В связи с этим крупнейшие компании всячески поддерживают научные разработки в области оптимизации.

Идея случайной оптимизации впервые была высказана одним из пионеров кибернетики **Уильямом Эшби** (Ashby, William Ross; 1903–1972). Эшби окончил медицинский факультет Кембриджского университета, работал практикующим психиатром в Англии. С 1960 г. был профессором кибернетики и психиатрии Иллинойского университета. Вместе с «отцом кибернетики» **Норбертом Винером** (Wiener, Norbert; 1894–1964) Эшби предложил использовать единый подход к изучению *сложных систем,* как живых, так и искусственных, в 1948 г. изобрел *гомеостат* — простую электромеханическую модель самоадаптирующегося организма[6].

Уильям Эшби (второй слева и Норберт Винер (крайний справа)

Практические вычислительные методы, использующие случайные числа, были созданы в конце 1940-х годов в сверхсекретной Los Alamos National Laboratory, где лучшие ученые многих стран, бежавшие из нацистской Европы, создавали американскую атомную бомбу[7]. Там в рамках «Манхэттенского проекта» работал интернациональный коллектив выдающихся физиков и математиков: венгры Джон (Янош) фон Нейман (von Neumann, John 1903–1957) и Эдвард Теллер (Teller, Edward; 1908–2003), итальянец Энрико Ферми, поляк Станислав Улам (Ulam, Stanislaw Marcin; 1909–1984), грек Николас Метрополис (Metropolis, Nicholas Constantine; 1915–1999) и др.

Как пишет Метрополис в своих воспоминаниях[8], общая схема

[6] *Эшби У.* Конструкция мозга. — М.: ИЛ, 1962.

[7] О захватывающей истории развития атомной физики и техники советую прочитать книгу: *Р. Юнг.* Ярче тысячи солнц. Повествование об ученых-атомниках. — М.: Госатомиздат, 1961. — 279 с.
http://publ.lib.ru/ARCHIVES/YU/YUNG_Robert_Yung_R..html

[8] *Nicholas Metropolis.* The beginning of the Monte Carlo method // Los Alamos Science, Special Issue dedicated to Stanislaw Ulam, 1987. — P. 125–130. — http://library.lanl.gov/la-pubs/00326866.pdf

метода статистических испытаний для решения задачи переноса радиации была предложена гениальным математиком **Джоном фон Нейманом** (и в эту область науки им внесен существенный вклад!), при этом он опирался на неопубликованную работу не менее гениального физика Энрико Ферми (Fermi, Enrico; 1901–1954), выполненную еще в 1930-х годах в Риме. Фон Нейман изобрел алгоритм получения равномерных случайных чисел (способ середины квадрата), алгоритм получения чисел с заданным законом распределения (способ обратной функции) и т. д. Название метода статистических испытаний предложил Метрополис в честь дяди Улама, который был азартным игроком и занимал деньги для поездки в Монте-Карло. В 1949 г. вышла в свет статья Метрополиса и Улама «Метод Монте-Карло», с тех пор этот метод стал широко применяться не только для решения разнообразных физических задач, но и в далеких от физики областях науки, чему способствовало бурное развитие средств вычислительной техники.

В отечественном научном сообществе идея использовать случайные числа для оптимизации, по-видимому, наиболее ярко и последовательно пропагандировалась выдающимся латвийским ученым российского происхождения, блестящим преподавателем и популяризатором науки **Леонардом Андреевичем Растригиным** (1929–1998). На эту тему он написал множество статей и монографий [22], кроме того предложил оригинальный кибернетический подход к разработке адаптивных систем управления и проектирования, к моделированию процессов познания. Особенно известен Растригин своими популярными книгами, в которых рассказывал о кибернетике, случайных процессах, вычислительных машинах и сетях. Его имя носит одна из многоэкстремальных тестовых функций, которую мы приводили выше.

Л. А. Растригин

Работы по моделированию эволюции выполнялись мно-

гими учеными в разных странах, однако они стали особенно популярны в связи с работами **Джона Холланда** (Holland, John Henry; p. 1929). Холланд изучал физику в Массачусетсском технологическом институте, затем математику в Мичиганском университете, первый получил в этом университете ученую степень по компьютерным наукам. Предложил формальную модель генетического алгоритма (схема Холланда). Его книга «Adaptation in Natural and Artificial Systems» (1975) стала мировым научным бестселлером.

Дж. Холланд

В середине 1980-х годов в Питтсбурге, штат Пенсильвания, состоялась первая международная конференция по генетическим алгоритмам. В конце 1980-х компания «General Electric» выпустила в продажу первый коммерческий пакет программ эволюционного моделирования, рассчитанный на мейнфреймы, вслед за этим появились аналогичные пакеты для персональных компьютеров.

Глава 12

Многомерная оптимизация с ограничениями

Как мы отмечали в предыдущей главе, наличие ограничений существенным образом усложняет задачу поиска экстремума. Если они к тому же невыпуклые, то никакой известный алгоритм не может гарантировать нахождения решения. По этой причине впредь целевая функция и ограничения предполагаются выпуклыми.

Изучая теорию выпуклого программирования (гл. 9), мы рассматривали задачу оптимизации в стандартном виде

$$\begin{aligned} &f(x_1, \ldots, x_n) \to \min, \\ &g_i(x_1, \ldots, x_n) \leqslant 0, \quad i = 1, \ldots, m, \\ &x_1, \ldots, x_n \geqslant 0 \end{aligned}$$

и показали, что любую задачу можно привести к такому виду, однако для практических целей иногда бывает удобно выделить отдельно ограничения-неравенства *(inequality constraints),* ограничения-равенства *(equality constraints)* и *простые ограничения* двух видов: односторонние $x_i \geqslant 0$ *(bound constraints)* или двусторонние $a_i \leqslant x_i \leqslant b_i$ *(box constraints).* Для некоторых методов бывает полезно отдельно учитывать линейные ограничения.

Поскольку оптимизация с ограничениями существенно разнообразнее безусловной минимизации, в основу классификации будет положен не порядок производных, а общий подход к учету ограничений. С этой точки зрения великое множество существующих методов можно условно разделить на три группы (рис. 12.1), в каждой из которых приведены примеры самых типичных.

Рис. 12.1. Классификация методов оптимизации с ограничениями

К первой группе относятся методы, основанные на преобразовании целевой функции. Она видоизменяется таким образом, что ее безусловный экстремум совпадает с условным экстремумом исходной задачи. В результате представляется возможность применить весь арсенал методов оптимизации без ограничений, часть которых мы рассматривали в предыдущей главе.

Вторую группу составляют методы поиска экстремума, приспосабливающие известные релаксационные методы оптимизации к ситуации, когда из-за наличия ограничений не все понижающие (прогрессивные) направления являются возможными. При этом учет ограничений производится либо непосредственно (такой подход был исторически первым, но его применение оказалось целесообразным только для простых линейных ограничений), либо косвенно через функцию Лагранжа.

Третью группу образуют методы, основанные на аппроксимации нелинейной задачи линейным приближением и последующем использовании эффективных методов линейного программирования. Для определенных классов задач такой подход оказывается весьма плодотворным.

Прежде чем рассматривать конкретные подходы к оптимизации с ограничениями, следует заметить, что подавляющее большинство существующих методов исходит из предположения, что целевая функция и ограничения являются *гладкими*, т. е. предполагается существование и непрерывность первых производных. Как только входящие в постановку задачи функции становятся недифференцируемыми, теряется возможность применять классический аппарат исчисления бесконечно малых. И хотя известны попытки обобщить понятия производной на негладкие функции и построить соответствующий теоретический базис (см., например, [11]), на практике чаще всего используются эвристические методы, приспосабливающие хорошо зарекомендовавшие себя алгоритмы деформируемого многогранника или поиска по образцу к ситуации с ограничениями. Их подробное изучение выходит за рамки нашего курса.

12.1. Сведение к задаче без ограничений

К данной группе относятся методы, позволяющие тем или иным способом преобразовать задачу с ограничениями в задачу без ограничений, соответственно усложнив целевую функцию.

Замена переменных

Избавиться от простого ограничения на некоторую переменную x_i можно подходящей заменой $x_i = \varphi(z)$. В табл. 12.1 приведены некоторые варианты замены переменной для учета одностороннего или двустороннего ограничения.

К сожалению, такая замена переменных сопряжена с определенными опасностями:

Таблица 12.1. Возможные замены переменных

Ограничение	Преобразование
$x_i \geqslant 0$	$x_i = z_i^2$
$x_i \geqslant a_i$	$x_i = a_i + z_i^2$
$x_i > a_i$	$x_i = a_i + e^{z_i}$
$a_i \leqslant x_i \leqslant b_i$	$x_i = b_i + (a_i - b_i)\sin^2 z_i$
$a_i < x_i < b_i$	$x_i = b_i + (a_i - b_i)\frac{1}{\pi}\operatorname{arctg} z_i$

- существенно возрастает степень нелинейности целевой функции, у новой целевой функции появляются особенности, которых не было у исходной, теряется гладкость;
- матрица Гессе получается вырожденной или плохо обусловленной, возникают дополнительные стационарные и локально экстремальные точки;
- зависимость целевой функции от новых переменных оказывается периодической.

В связи с этим авторы, имеющие большой опыт практической оптимизации, не рекомендуют идти по этому пути [7, с. 364], предлагая учитывать простые ограничения наряду со всем другими.

Методы штрафных функций

Идея *штрафных функций — penalty functions* состоит в замене «запретительных» ограничений «экономическими». Пусть исходная задача задана в виде (возможные ограничения на неотрицательность включены в общий список)

$$f(\overrightarrow{X}) \to \min,$$
$$g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m.$$

Сформулируем новую задачу

$$P(\overrightarrow{X}) = f(\overrightarrow{X}) + h(\overrightarrow{X}) \to \min,$$
$$-\infty < x_j < \infty, \quad j = 1, \ldots, n,$$

в которой модифицированная (так называемая штрафная) целевая функция $P(\overrightarrow{X})$ содержит дополнительное слагаемое $h(\overrightarrow{X})$, накладывающее штраф за несоблюдение ограничений исходной задачи. Если величина штрафа достаточно велика, то это заставит любой алгоритм минимизации считаться с ограничениями.

Поскольку новая задача не содержит ограничений, к ней может быть применен любой из рассмотренных в предыдущей главе метод безусловной оптимизации. Реализовать общую идею можно по-разному. Практические методы штрафных функций делятся на две группы: методы *внешней точки (exterior point)* и методы *внутренней точки (interior point)*.

Методы внешней точки исходят из предположения, что штраф внутри допустимой области отсутствует, а за ее пределами монотонно возрастает при удалении от границы. В методах внешней точки начальная точка $\overrightarrow{X}_0$ и все последующие располагаются за пределами допустимой области. Любой релаксационный метод безусловной оптимизации, стремясь минимизировать штраф, постарается прижать точку экстремума к допустимой области с внешней стороны.

Примером штрафной функции для ограничений-неравенств может служить функция с квадратичным внешним штрафом

$$P(\overrightarrow{X}) = f(\overrightarrow{X}) + R\sum_{i=1}^{m} \mathrm{cut}^2[g_i(\overrightarrow{X})], \tag{12.1}$$

где $\mathrm{cut}[x]$ обозначает так называемую *функцию срезки* (см. рис. 12.2):

$$\mathrm{cut}[x] = \max[0, x] = \begin{cases} 0, & x \leqslant 0, \\ x, & x \geqslant 0. \end{cases}$$

Для ограничений-равенств вида $g_i(\overrightarrow{X}) = 0$ соответствующие штрафные слагаемые при квадратичном штрафе записываются в форме $[g_i(\overrightarrow{X})]^2$.

Легко видеть, что для точек, расположенных внутри допустимой области, $g_i(\overrightarrow{X}) \leqslant 0$, при этом штраф равен нулю. Масштабный параметр R определяет строгость штрафа. Для того чтобы повысить крутизну штрафной функции за пределами допустимой области, нужно увеличить штрафной параметр. Фиакко и Мак-Кормик, опубликовавшие глубокое исследование методов штрафных функций [26], предлагают делать это постепенно. Задавшись исходной точкой $\overrightarrow{X}_0$, на первой итерации полагают $R_0 = 1$ и, пользуясь наиболее либеральной штрафной функцией, получают точку минимума $\overrightarrow{X}_1^*$. На каждой следующей итерации штрафной параметр увеличивают: $R_{k+1} = R_k + \Delta R$, при этом найденная на предыдущей итерации точка минимума становится исходной для следующей. Такой метод назван ими *последовательной безусловной минимизацией*. В [26] доказана принципиальная сходимость метода для выпуклых функций.

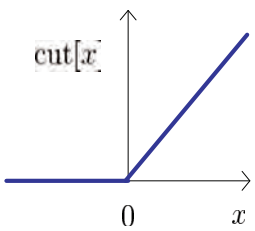


Рис. 12.2.
Функция срезки

П р и м е р. Вернемся к задаче на с. 56 и включим условия неотрицательности переменных в общий набор ограничений:

$$f(\overrightarrow{X}) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min,$$

$$g_1(\overrightarrow{X}) = x_1^2 + x_2^2 - 4 \leqslant 0,$$
$$g_2(\overrightarrow{X}) = -x_1 + x_2 \leqslant 0,$$
$$g_3(\overrightarrow{X}) = x_2 - 1 \leqslant 0,$$
$$g_4(\overrightarrow{X}) = -x_1 \leqslant 0,$$
$$g_5(\overrightarrow{X}) = -x_2 \leqslant 0.$$

Тогда внешняя квадратичная штрафная функция запишется в виде

$$P(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 2)^2 + \\ + R\left\{\mathrm{cut}^2[x_1^2 + x_2^2 - 4] + \mathrm{cut}^2[-x_1 + x_2] + \right. \\ \left. + \mathrm{cut}^2[x_2 - 1] + \mathrm{cut}^2[-x_1] + \mathrm{cut}^2[-x_2]\right\}. \quad (12.2)$$

На рис. 12.3 сплошными линиями показаны изолинии штрафной функции $P(x_1, x_2)$ для данной задачи при значении штрафного параметра $R = 2$. На этом же рисунке построена допустимая область, а также штриховыми линиями изображены изолинии исходной целевой функции $f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 2)^2$. Внутри допустимой области оба семейства изолиний, как и следовало ожидать, совпадают.

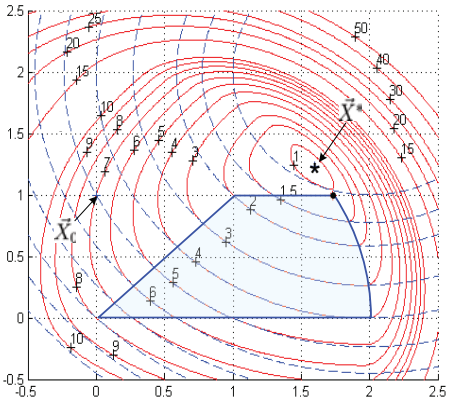


Рис. 12.3. Изолинии исходной целевой (штриховые линии) и внешней штрафной (сплошные линии) функций. Параметр штрафа R=2

Поскольку $f(\vec{X})$ и $h(\vec{X})$ выпуклы, то их сумма $P(\vec{X})$ также выпукла, поэтому любой метод выпуклой оптимизации даст точку глобального минимума.

Применив алгоритм `fminsearch` из пакета MATLAB (см. гл. 13), реализующий метод деформируемого многогранника со стартом из $\overrightarrow{X}_0 = (0,\, 1)^T$, получаем точку минимума $\overrightarrow{X}^* = (1.5970,\ 1.2299)^T$, она отмечена на рис. 12.3 звездочкой. Существенное отклонение от истинного оптимального значения $(\sqrt{3},\, 1)^T \approx (1.73205,\, 1)^T$, вычисленного в примере на с. 56 (она отмечена черным кружком), произошло из-за того, что штрафная функция оказалась не слишком крутой вблизи границы, позволив точке минимума заметно нарушить ограничения.

В табл. 12.2 (левая колонка) приведены результаты нескольких итераций метода последовательной безусловной минимизации при внешнем штрафе. Исходная точка $\overrightarrow{X}_0 = (0,\, 1)^T$. ▲

Таблица 12.2. Сходимость последовательной безусловной оптимизации. Слева — метод внешней точки, справа — метод внутренней точки

R	x_1^*	x_2^*	R	x_1^*	x_2^*
1	1.5557	1.2299	1	1.5809	0.5391
2	1.5970	1.2299	2	1.6095	0.6011
5	1.6578	1.1279	5	1.6423	0.7065
10	1.6905	1.0730	10	1.6619	0.7806
20	1.7100	1.0392	20	1.6781	0.8400
50	1.7229	1.0164	50	1.6951	0.8962
100	1.7275	1.0083	100	1.7049	0.9257

Методы внутренней точки

Вся работа методов внутренней точки происходит внутри допустимой области, но за само приближение к границе взимается штраф. Его, к примеру, можно определить по формуле

$$P(\overrightarrow{X}) = f(\overrightarrow{X}) - \frac{1}{R}\sum_{i=1}^{m}\frac{1}{g_i(\overrightarrow{X})}. \qquad (12.3)$$

Чем ближе точка $\overrightarrow{X}$ к любому из ограничений, тем ближе к нулю соответствующий знаменатель, в итоге величина штрафа растет неограниченно. Поэтому внутренняя штрафная функция иначе называется *барьерной (barrier function)*. Часто вместо $-1/g_i(\overrightarrow{X})$ используют логарифмическую функцию $-\ln[-g_i(\overrightarrow{X})]$, также неограниченно возрастающую при приближении аргумента к нулю.

Замечание. Методы внутренней точки неприменимы к ограничениям-равенствам, так как в этом случае допустимая область вообще не содержит внутренних точек.

П р и м е р. Применим метод внутренней точки к задаче из предыдущего примера. На рис 12.4 приведены изолинии внутренней штрафной функции вида (12.3) при разных значениях штрафного параметра R. При увеличении R (для метода внутренней

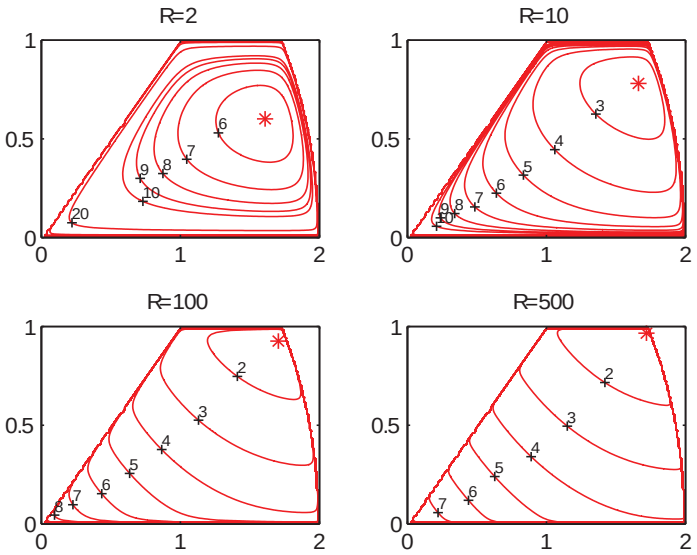


Рис. 12.4. Изолинии внутренней штрафной функции при разных значениях параметра R. Точки минимума отмечены звездочками

точки он стоит в знаменателе) доля штрафа в модифицированной целевой функции уменьшается, он начинает сказываться ближе к границе, но зато крутизна барьера увеличивается. В правой колонке табл. 12.2 приведены соответствующие результаты безусловной оптимизации. Исходная точка $\overrightarrow{X}_0 = (1,\, 0.5)^T$. ▲

Благодаря своей простоте методы штрафных функций были популярны в 1960-х годах, однако скоро выявились присущие им недостатки. Если искомый минимум находится на границе допустимой области, то для достижения высокой точности внешняя или внутренняя штрафные функции должны быть очень крутыми, при этом задача оптимизации становится плохо обусловленной, начинают сказываться ошибки вычислений, работа алгоритмов становится неустойчивой. Поэтому им на смену пришли релаксационные методы, учитывающие ограничения иным способом — либо непосредственно, либо через функцию Лагранжа.

12.2. Общая схема релаксационных методов, учитывающих ограничения

Так же как и при оптимизации без ограничений, все релаксационные методы с ограничениями задаются стандартной итерационной процедурой

$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k \overrightarrow{d}_k.$$

Следовательно, каждый конкретный метод характеризуется:

- способом выбора направления движения $\overrightarrow{d}_k$ на данном шаге;
- способом выбора длины шага t_k;
- критерием остановки итерационного процесса.

Когда ограничения отсутствовали, любые направления были возможными, длина шага также ничем не лимитировалась. При наличии ограничений процедуры выбора направления и длины шага необходимо модифицировать.

Выбор направления

Пусть задача выпуклого программирования задана в виде

$$f(\overrightarrow{X}) \to \min, \quad g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m,$$

то есть ограничения на неотрицательность всех или части переменных, если они существуют, уже заложены в условия $g_i(\overrightarrow{X}) \leqslant 0$.

Предположим, что на некоторой итерации поиск пришел в точку $\overrightarrow{X}_k$. Обозначим множество индексов активных ограничений в данной точке через $M(\overrightarrow{X}_k) = \{i \mid g_i(\overrightarrow{X}_k) = 0\}$. Пусть $\overrightarrow{d}$ – некоторое направление в точке $\overrightarrow{X}_k$. Как мы уже говорили, это направление называется *возможным (feasible)*, если по нему можно продвинуться на некоторое расстояние $t > 0$, не протыкая область ограничений, т. е. $g_i(\overrightarrow{X}_k + t\overrightarrow{d}) \leqslant 0$. Если взять t бесконечно малым и разложить $g_i(\overrightarrow{X})$ в ряд, ограничившись линейным приближением, получим

$$g_i(\overrightarrow{X}_k + t\overrightarrow{d}) = g_i(\overrightarrow{X}_k) + t\overrightarrow{d}^T\overrightarrow{\nabla} g_i(\overrightarrow{X}_k).$$

Отсюда следует дифференциальное условие возможности направления $\overrightarrow{d}$:

$$\overrightarrow{d}^T\overrightarrow{\nabla} g_i(\overrightarrow{X}_k) \leqslant 0, \quad i \in M(\overrightarrow{X}_k).$$

Направление $\overrightarrow{d}$ мы называли ранее *понижающим* или *прогрессивным,* если по нему можно продвинуться на некоторое расстояние $t > 0$, улучшая значение целевой функции, т. е. $f(\overrightarrow{X}_k + t\overrightarrow{d}) < f(\overrightarrow{X}_k)$. Из определения производной по направлению (см. с. 30) следует дифференциальное условие прогрессивности направления через скалярное произведение

$$\overrightarrow{d}^T\overrightarrow{\nabla} f_i(\overrightarrow{X}_k) < 0.$$

На рис. 12.5 приведена иллюстрация для одного нелинейного ограничения $g(\overrightarrow{X}) \leqslant 0$ в двумерном случае. Показаны вектор внешней нормали $\overrightarrow{\nabla} g(X_k)$, перпендикулярный к касательной в точке $\overrightarrow{X}_k$, а также вектор антиградиента $-\overrightarrow{\nabla} f(\overrightarrow{X}_k)$.

Вспоминая определение скалярного произведения векторов (см. с. 19), замечаем, что оно неотрицательно, если $\cos\varphi \geqslant 0$, что соответствует $-90° \leqslant \varphi \leqslant 90°$, т. е. угол между векторами острый или прямой. И наоборот, оно отрицательно, если угол тупой. Таким образом, направление $\overrightarrow{d}$ является возможным, если угол β между ним и вектором внешней нормали прямой или тупой, и является прогрессивным, если угол α между ним и антиградиентом — острый.

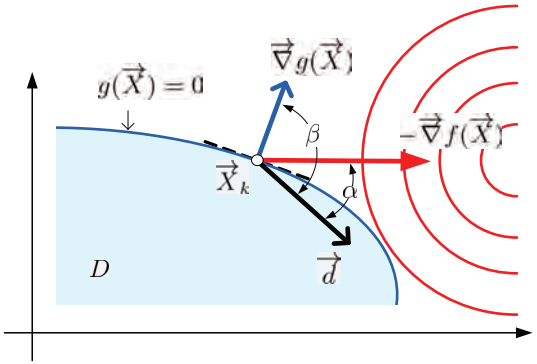


Рис. 12.5. Возможные и прогрессивные направления для одного активного ограничения в двумерном случае

Для того чтобы итеративный шаг был успешным, направление движения должно быть одновременно и возможным и понижающим (прогрессивным). В этой терминологии дифференциальные условия оптимальности Куна — Таккера могут быть сформулированы следующим образом: *полученный на k-м шаге план является оптимальным, если все возможные направления из $\overrightarrow{X}_k$ не являются прогрессивными.*

Существует множество способов выбора такого направления, в основном оно и определяет многообразие релаксационных мето-

дов оптимизации с ограничениями. Этим вопросом мы займемся позже, а пока посмотрим, каким образом наличие ограничений влияет на процедуру выбора длины шага.

Выбор длины шага

После того как направление $\overrightarrow{d}_k$ выбрано, следует сделать шаг в данном направлении. В полношаговых релаксационных методах без ограничений длина шага t_k выбиралась поиском безусловного минимума целевой функции вдоль заданного направления:

$$t_k = \arg \min_{0 \leqslant t < \infty} f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k).$$

При наличии ограничений так поступать нельзя, потому что полученное значение t_k может оказаться слишком большим и вывести поиск за область ограничений.

Если обозначить через t^* значение, полученное в результате безусловной минимизации, а через T — то значение t, при котором луч в направлении $\overrightarrow{d}$ протыкает область ограничений, то длину шага следует выбирать из условия

$$t_k = \min(t^*, T).$$

В то время как величину t^* можно найти обычным линейным поиском, с определением предельной длины шага T проблем значительно больше. Для этого следует решить нелинейные уравнения $g_i(\overrightarrow{X}_k + t\,\overrightarrow{d}_k) = 0$ относительно переменной t для всех неактивных ограничений и выбрать наименьший положительный корень. Он будет соответствовать тому ограничению, которое луч $\overrightarrow{X}_k + t\,\overrightarrow{d}_k$ проткнет в первую очередь.

Если функции ограничений простые, например линейные, то эта задача не составляет никакого труда, корни находятся аналитически. Для общего нелинейного случая часто используется другой подход к нахождению длины шага t_k без вычисления в отдельности t^* и T, он основан на изложенной в предыдущем параграфе идее штрафных функций. Строится одномерная функция $\psi(t)$, которая в англоязычной литературе называется *merit*

function, что можно перевести как *функция выигрыша (выгоды, полезности, качества).* Она представляет собой одномерное сечение штрафной целевой функции в выбранном направлении $\overrightarrow{d}_k$

$$\psi(t) = P(\overrightarrow{X}_k + t\,\overrightarrow{d}_k) = f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k) + h(\overrightarrow{X}_k + t\,\overrightarrow{d}_k),$$

где штрафное слагаемое $h(\overrightarrow{X})$ можно определять по-разному. Проще всего использовать внешний квадратичный штраф (12.1), однако возможны и другие варианты.

Поскольку функция качества уже учитывает ограничения, длина шага определяется из условия ее безусловного минимума:

$$t_k = \arg \min_{0 \leqslant t < \infty} \psi(t).$$

Критерий остановки

Как и в случае оптимизации без ограничений (см. с. 95), могут использоваться различные условия остановки итерационного процесса. Критерии по малости смещения текущей точки, по малости приращения целевой функции и по объему вычислений при этом не изменяются, однако условия первого порядка модифицируются с учетом ограничений.

Как следует из теории выпуклого программирования (см. с. 60), необходимыми условиями минимума для задачи в частном случае неограниченности переменных по знаку являются условия Куна — Таккера

$$\begin{aligned}
&\text{а–в)} \quad \overrightarrow{\nabla} L(\overrightarrow{X}) = 0, \\
&\text{д)} \quad y_i \frac{\partial L}{\partial y_i} = 0, \ \text{т. е.} \ \ y_i g_i(\overrightarrow{X}) = 0, \\
&\text{е)} \quad y_i \geqslant 0, \\
&\qquad i = 1, \ldots, m.
\end{aligned}$$

Что касается условия «г», имеющего вид $\dfrac{\partial L}{\partial y_i} = g_i(\overrightarrow{X}) \leqslant 0$, то оно представляет собой требование быть допустимой точкой.

В приближенном итерационном процессе оно формулируется как требование малости *нарушения ограничений (constraint violations)* для текущей точки $\overrightarrow{X}_k$ и контролируется соотношением

$$\max_i \operatorname{cut}[g_i(X_k)] = \max_i\ [0,\ g_i(\overrightarrow{X}_k)] < \varepsilon,$$

где ε — установленный допуск *(tolerance)* по ограничениям, а $\operatorname{cut}[x] = \max[0,\ x]$ — введенная ранее функция срезки (см. с. 159).

Для условий «а»–«в» в качестве меры оптимальности первого порядка используется бесконечная норма градиента лагранжиана, для условий «д» аналогичной мерой является $\max_i |\, y_i g_i(\overrightarrow{X}_k)|$, а в качестве комбинированной меры оптимальности первого порядка вычисляется наибольшее из этих двух значений. Таким образом, условие первого порядка считается выполненным, если

$$\max\left(\max_j \left|\frac{\partial L(\overrightarrow{X}_k)}{\partial x_j}\right|,\ \max_i |\, y_i g_i(\overrightarrow{X}_k)|\right) < \varepsilon.$$

Разумеется, проверка условия первого порядка по данному критерию производится только в тех алгоритмах, которые делают оценки множителей Лагранжа.

Обзор практических методов

Рассмотрев общую схему релаксационных методов, учитывающих ограничения, вернемся к проблеме выбора конкретного направления движения $\overrightarrow{d}_k$ на каждой итерации. Разумеется, эта проблема возникает только тогда, когда текущая точка $\overrightarrow{X}_k$ находится на границе допустимой области, потому что внутри области наилучшим направлением всегда является направление антиградиента.

Практические приемы выбора прогрессивного направления можно условно разбить на две группы. К первой относятся методы, явно учитывающие ограничения $g_i(\overrightarrow{X})$. Они стараются приспособить градиентный спуск к ситуации, когда движение прямо по антиградиенту невозможно. Существует обширный класс таких методов, созданных в основном в 1960–1970-х годах, из них

наиболее известен и широко используется вплоть до настоящего времени *метод проектирования градиента (метод активного набора),* который мы рассмотрим в следующем параграфе.

Вторая группа методов, интенсивно разрабатываемая в последние десятилетия, учитывает ограничения косвенным путем, через функцию Лагранжа. Наиболее передовым и эффективным представителем этой группы является метод *последовательного квадратичного программирования — sequential quadratic programming (SQP),* включенный во все современные библиотеки прикладных программ.

12.3. Метод проектирования градиента

Идею метода проиллюстрируем на простом примере (см. рис. 12.6). Пусть на k-й итерации релаксационного градиентного

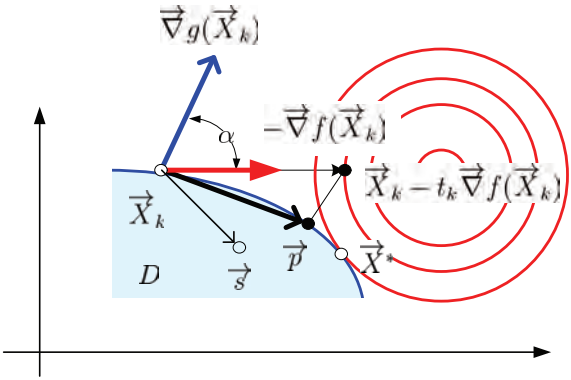


Рис. 12.6. Проекция антиградиента на допустимую область D

метода поиск пришел в такую точку $\overrightarrow{X}_k$ на границе допустимой области D, где направление антиградиента $-\overrightarrow{\nabla} f(\overrightarrow{X}_k)$ не является возможным, поскольку угол α между ним и нормалью $\overrightarrow{\nabla} g(\overrightarrow{X}_k)$ к активному ограничению острый. Это значит, что для любого $t_k > 0$ следующая точка $\overrightarrow{X}_{k+1} = \overrightarrow{X}_k - t_k \overrightarrow{\nabla} f(\overrightarrow{X}_k)$ будет лежать за пределами допустимой области.

В этих условиях направление спуска должно быть направлено не по антиградиенту, а на некоторую точку $\overrightarrow{s}$, лежащую в н у т р и (или на границе) допустимой области. Разумно потребовать, чтобы она была как можно ближе к точке $\overrightarrow{X}_k - t_k \overrightarrow{\nabla} f(\overrightarrow{X}_k)$, т. е. представляла собой п р о е к ц и ю этой точки на область D (см. определение проекции точки на множество на с. 21). В нашем примере это точка $\overrightarrow{p}$. Соответственно вектор движения $\overrightarrow{X_k p}$ есть проекция $-t_k \overrightarrow{\nabla} f(\overrightarrow{X}_k)$ на D.

Хотя метод проектирования градиента, точнее говоря, антиградиента в принципе может быть применим к любым задачам выпуклой оптимизации, на практике он оказался эффективным только для задач с линейными ограничениями, поскольку присутствующая в нем подзадача проектирования успешно решается только для аффинных множеств (линейных многообразий).

Линейные ограничения-равенства

Изложение метода начнем с более простого случая, когда ограничения заданы в форме линейных р а в е н с т в без условий неотрицательности переменных, то есть когда множество допустимых решений (планов) представляет собой аффинное множество (см. определение на с. 20). В этом случае оптимизационная задача имеет вид

$$\begin{gathered} f(\overrightarrow{X}) \to \min, \\ A\overrightarrow{X} = \overrightarrow{b}. \end{gathered} \tag{12.4}$$

Здесь матрица A составлена из строк $\overrightarrow{a}_i^T$, где $\overrightarrow{a}_i$ — направляющие векторы (векторы нормалей) гиперплоскостей, определяющих аффинное множество.

Функция Лагранжа для поставленной задачи равна

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \overrightarrow{Y}^T (A\overrightarrow{X} - \overrightarrow{b}).$$

Дифференциальные условия Куна — Таккера в частном случае классической задачи с ограничениями-равенствами согласно (9.35) запишутся в сокращенном виде

$$\text{а, б, в)} \quad \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) = \overrightarrow{\nabla} f(\overrightarrow{X}) + A^T \overrightarrow{Y} = 0; \tag{12.5}$$

$$\text{г, д, е)} \quad \overrightarrow{G}(\overrightarrow{X}) = A\overrightarrow{X} - \overrightarrow{b} = 0. \tag{12.6}$$

Геометрический смысл этих условий вполне прозрачен: выражение (12.6) означает, что оптимальный план должен принадлежать допустимому множеству, а равенство (12.5), если его переписать в виде $-\overrightarrow{\nabla} f(\overrightarrow{X}) = A^T \overrightarrow{Y} = \sum_{i=1}^{m} y_i \overrightarrow{a}_i$, требует, чтобы в точке оптимума антиградиент целевой функции был равен линейной комбинации векторов нормалей $\overrightarrow{a}_i$ к гиперплоскостям ограничений. Коэффициенты этой линейной комбинации есть множители Лагранжа, в данном случае они могут быть любыми, не обязательно неотрицательными.

Метод описывается следующей релаксационной процедурой. Пусть $\overrightarrow{X}_k$ — исходная точка на k-м шаге, $\overrightarrow{\nabla} f(\overrightarrow{X}_k)$ — градиент целевой функции в данной точке. В качестве направления движения $\overrightarrow{d}_k$ берется проекция антиградиента на допустимое множество, которая равна $\overrightarrow{d}_k = -P\,\overrightarrow{\nabla} f(\overrightarrow{X}_k)$, где матрица проектирования P определяется выражением (9.4)

$$P = \mathrm{I} - A^T (AA^T)^{-1} A.$$

Следующая точка определяется по обычному правилу $\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t\,\overrightarrow{d}_k$, где $t > 0$ — некоторым образом выбранная длина шага.

Данная рекуррентная процедура обладает рядом полезных свойств.

1. Точка $\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t\,\overrightarrow{d}_k$ при любом t принадлежит допустимому множеству, т. е. направление $\overrightarrow{d}$ является возможным (feasible). Для доказательства вычислим

$$A\overrightarrow{X}_{k+1} = A(\overrightarrow{X}_k + t\,\overrightarrow{d}_k) = A(\overrightarrow{X}_k - tP\overrightarrow{\nabla} f) =$$

$$= \overrightarrow{b} - tA[\mathrm{I} - A^T(AA^T)^{-1}A] = \overrightarrow{b} - t[A - (AA^T)(AA^T)^{-1}A]\overrightarrow{\nabla} f =$$
$$= \overrightarrow{b} - t(A - \mathrm{I}A)\overrightarrow{\nabla} f = \overrightarrow{b} - t(A - A)\overrightarrow{\nabla} f = \overrightarrow{b}.$$

2. Направление $\overrightarrow{d}_k$ является понижающим (прогрессивным). Рассмотрим одномерную функцию сечения

$$\varphi(t) = f(\overrightarrow{X}_k + t\overrightarrow{d}_k) \tag{12.7}$$

и вычислим ее производную при $t = 0$. Дифференцируя сложную функцию и учитывая положительную определенность матрицы проектирования, имеем

$$\left.\frac{\partial\varphi(t)}{\partial t}\right|_{t=0} = \overrightarrow{\nabla}^T f(\overrightarrow{X}_k)\overrightarrow{d}_k = -\overrightarrow{\nabla}^T f(\overrightarrow{X}_k)P\overrightarrow{\nabla} f(\overrightarrow{X}_k) \leqslant 0.$$

Критерий окончания поиска устанавливается следующим утверждением.

3. Если на некотором шаге

$$\overrightarrow{d}_k = P\,\overrightarrow{\nabla} f(\overrightarrow{X}_k) = 0, \tag{12.8}$$

то $\overrightarrow{X}_k$ — точка оптимума. Действительно, подставив сюда матрицу проектирования, имеем

$$P\,\overrightarrow{\nabla} f(\overrightarrow{X}_k) = \left[\mathrm{I} - A^T(AA^T)^{-1}A\right]\overrightarrow{\nabla} f(\overrightarrow{X}_k) =$$
$$= \overrightarrow{\nabla} f(\overrightarrow{X}_k) + A^T\left[-(AA^T)^{-1}A\overrightarrow{\nabla} f(\overrightarrow{X}_k)\right] = 0. \tag{12.9}$$

Сравнивая (12.9) с (12.5), видим, что это есть необходимое условие экстремума, в котором вектор множителей Лагранжа равен

$$\overrightarrow{Y} = -(AA^T)^{-1}A\overrightarrow{\nabla} f(\overrightarrow{X}_k). \tag{12.10}$$

Доказанные свойства приспосабливают градиентный метод к наличию ограничений в форме линейных равенств. Если длину шага выбирать по правилу $t_k = \arg\min\varphi(t) = f(\overrightarrow{X}_k + t\overrightarrow{d}_k)$, то мы получим метод поиска минимума, аналогичный методу наискорейшего спуска в безусловной оптимизации.

Метод активного набора

Перейдем к случаю, когда ограничения задачи сформулированы в виде неравенств (ограничения на неотрицательность включены в общий список):

$$\begin{aligned} &f(\overrightarrow{X}) \to \min, \\ &A\overrightarrow{X} \leqslant \overrightarrow{b}, \end{aligned} \quad \text{или} \quad \begin{aligned} &f(\overrightarrow{X}) \to \min, \\ &\overrightarrow{a}_i^T \overrightarrow{X} \leqslant b_i, \\ &i = 1, \ldots, m, \end{aligned} \tag{12.11}$$

где $\overrightarrow{a}_i^T$ обозначает i-ю строку матрицы условий A.

В этом частном случае (ограничения на неотрицательность переменных отсутствуют) условия оптимальности Куна — Таккера имеют вид

$$\text{а, б, в)} \quad \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) = \overrightarrow{\nabla} f(\overrightarrow{X}) + A^T \overrightarrow{Y} = 0; \tag{12.12}$$

$$\text{г)} \quad \overrightarrow{G}(\overrightarrow{X}) = A\overrightarrow{X} - \overrightarrow{b} \leqslant 0; \tag{12.13}$$

$$\text{д)} \quad \overrightarrow{Y}(A\overrightarrow{X} - \overrightarrow{b}) = 0; \tag{12.14}$$

$$\text{е)} \quad \overrightarrow{Y} \geqslant 0. \tag{12.15}$$

Здесь выражение (12.13) по-прежнему означает допустимость оптимальной точки, а равенство (12.14) есть условие дополняющей нежесткости, которое требует, чтобы множители Лагранжа, соответствующие неактивным ограничениям, обращались в нуль. Оно становится очевидным, если его переписать в виде

$$\text{д)} \quad \sum_{i=1}^{m} y_i(\overrightarrow{a}_i^T \overrightarrow{X} - b_i) = 0.$$

Поскольку оба сомножителя в сумме неотрицательны, строгая положительность одного влечет равенство нулю другого.

Таким образом, при вычислении множителей Лагранжа и проверке на оптимальность по критерию (12.12) можно учитывать только активные ограничения, считая их строгими равенствами. Задача частично свелась к оптимизации с линейными ограничениями-равенствами, рассмотренной выше.

С учетом сказанного метод проекции градиента для ограничений в виде линейных неравенств модифицируется следующим образом.

П о д г о т о в и т е л ь н ы й э т а п

Выбирается любая допустимая начальная точка $\overrightarrow{X}_0$. Так как ограничения задачи линейны, то для нахождения точки, принадлежащей допустимой области, можно воспользоваться методом искусственного базиса в линейном программировании [9, с. 100]. Если $\overrightarrow{b} \geqslant 0$, то в качестве начальной может быть взята нулевая точка.

И т е р а ц и я

Шаг 1. Проверка на оптимальность. Для текущей допустимой точки $\overrightarrow{X}_k$ определяется список, насчитывающий q активных ограничений *(активный набор)*[1]. Без ограничения общности их можно считать первыми, т. е.

$$\overrightarrow{a}_i^T \overrightarrow{X}_k = b_i,\ i = 1, \ldots, q,$$
$$\overrightarrow{a}_i^T \overrightarrow{X}_k < b_i,\ i = q+1, \ldots, m.$$

Если активные ограничения действительно имеются ($q \geqslant 1$), то составляется матрица A_q размера $q \times n$, состоящая только из активных строк $\overrightarrow{a}_i^T$, $i = 1, \ldots, q$, после чего на ее основе вычисляется матрица проектирования

$$P_q = \mathrm{I} - A_q^T (A_q A_q^T)^{-1} A_q.$$

Текущая точка $\overrightarrow{X}_k$ является оптимальной, если выполняются два условия:

[1] В некоторых публикациях активный набор ограничений называют *рабочим списком.*

1) антиградиент представляет собой линейную комбинацию внешних нормалей к активным ограничениям. Это соответствует условию (12.12), которое, как следует из (12.9), превращается в

$$P_q \overrightarrow{\nabla} f(\overrightarrow{X}_k) = 0; \tag{12.16}$$

2) множители Лагранжа, вычисляемые по формуле (12.10), неотрицательны:

$$\overrightarrow{Y} = (y_1, \ldots, y_q)^T = -(A_q A_q^T)^{-1} A_q \overrightarrow{\nabla} f(\overrightarrow{X}_k) \geqslant 0. \tag{12.17}$$

Если же активный набор пуст ($q = 0$), т. е. точка $\overrightarrow{X}_k$ находится внутри допустимой области, то критерий оптимальности имеет более простой вид

$$\overrightarrow{\nabla} f(\overrightarrow{X}_k) = 0.$$

Шаг 2. Определение направления поиска. Если условия оптимальности не выполняются, нужно идти к следующей точке $\overrightarrow{X}_{k+1}$ с меньшим значением целевой функции. Здесь возможны различные варианты.

В а р и а н т А. В простейшем варианте точка $\overrightarrow{X}_k$ лежит внутри допустимой области и активные ограничения отсутствуют ($q = 0$). Тогда направление поиска совпадает с антиградиентом:

$$\overrightarrow{d}_k = -\overrightarrow{\nabla} f(\overrightarrow{X}_k).$$

В а р и а н т Б. $P_q \overrightarrow{\nabla} f(\overrightarrow{X}_k) \neq 0$. Этот вариант соответствует ситуации, когда $1 \leqslant q < n$, и точка $\overrightarrow{X}_k$ лежит на границе допустимого многогранника, представляющей собой аффинное множество D_q, образованное пересечением q гиперплоскостей, но не в его вершине. Тогда в качестве направления поиска выбирается проекция антиградиента на это аффинное множество:

$$\overrightarrow{d}_k = -P_q \overrightarrow{\nabla} f(\overrightarrow{X}_k).$$

В а р и а н т В. $P_q \overrightarrow{\nabla} f(\overrightarrow{X}_k) = 0$, но хотя бы один из множителей Лагранжа $y_1, \ldots, y_q$ отрицателен. Этот вариант соответствует

ситуации, когда число гиперплоскостей в активном наборе совпадает с размерностью пространства ($q = n$), следовательно, аффинное множество, на которое происходит проектирование, имеет размерность 0 и представляет собой точку — вершину допустимого многогранника (см. замечание 3 на с. 24).

Для того чтобы проектирование стало возможным, одну из гиперплоскостей из числа тех, для которых $y_i < 0$, следует удалить из активного набора. Например, это может быть гиперплоскость с минимальным значением $\|\overrightarrow{a}_i\| y_i$. Соответствующая строчка удаляется из матрицы A_q, получается матрица A_{q-1}, определяющая аффинное множество D_{q-1} размерности 1 (ребро многогранника). Пересчитывается матрица проектирования

$$P_{q-1} = \mathrm{I} - A_{q-1}^T (A_{q-1} A_{q-1}^T)^{-1} A_{q-1},$$

и направление поиска определяется проектированием антиградиента на D_{q-1}:

$$\overrightarrow{d}_k = P_{q-1} \overrightarrow{\nabla} f(\overrightarrow{X}_k).$$

Шаг 3. Определение предельной длины поиска. Когда направление поиска $\overrightarrow{d}_k$ задано, предстоит определить длину шага в данном направлении. Как отмечалось выше (см. с. 166), она ограничена величиной T — значением t, при котором луч $\overrightarrow{X}_k + t\overrightarrow{d}_k$, $t > 0$, протыкает изнутри область ограничений.

Для линейных ограничений величина T находится очень просто. Если i-е ограничение неактивно, т. е. $i = q+1, \cdots < m$, то момент T_i пересечения прямой $\overrightarrow{X}_k + t\overrightarrow{d}_k$ с гиперплоскостью $\overrightarrow{a}_i^T \overrightarrow{X} = b_i$ определяется решением уравнения $\overrightarrow{a}_i^T (\overrightarrow{X}_k + t\overrightarrow{d}_k) = b_i$, откуда

$$T_i = \frac{b_i - \overrightarrow{a}_i^T \overrightarrow{X}_k}{\overrightarrow{a}_i^T \overrightarrow{d}_k}, \tag{12.18}$$

а момент T, когда луч в положительном направлении впервые проткнет неактивные ограничения, равен

$$T = \min_i \max\,(0,\, T_i), \quad i = q+1, \ldots, m. \tag{12.19}$$

Шаг 4. Линейный поиск. Длина шага t_k на данной итерации определяется поиском минимума одномерной функции сечения в промежутке $[0,\, T]$:

$$t_k = \arg \min_{0 \leqslant t \leqslant T} f(\overrightarrow{X}_k + t\,\overrightarrow{d}_k).$$

Для сложных целевых функций одномерный поиск осуществляется одним из численных методов, описанных в гл. 10, для самых простых возможно аналитическое решение.

Важным частным случаем является квадратичная целевая функция

$$f(\overrightarrow{X}) = \overrightarrow{C}^T \overrightarrow{X} + \overrightarrow{X}^T D \overrightarrow{X},$$

минимум которой по лучу $\overrightarrow{X}_k + t\,\overrightarrow{d}_k$ может быть найден простым дифференцированием одномерной функции сечения:

$$\frac{d\varphi(t)}{dt} = \frac{df(\overrightarrow{X}_k + t\,\overrightarrow{d}_k)}{dt} = \overrightarrow{d}_k^T \overrightarrow{c} + 2\,\overrightarrow{d}_k^T D(\overrightarrow{X}_k + t\,\overrightarrow{d}_k) = 0,$$

откуда

$$t^* = -\frac{\overrightarrow{d}_k^T(\overrightarrow{c} + 2D\overrightarrow{X}_k)}{2\,\overrightarrow{d}_k^T D\,\overrightarrow{d}_k}. \tag{12.20}$$

Для того чтобы не выйти за пределы ограничений, длина шага устанавливается равной $t_k = \min(t^*,\, T)$, где T — предельная длина поиска, вычисленная по формуле (12.19).

Шаг 5. Итерационный шаг. Производится переход к новой точке по правилу

$$\overrightarrow{X}_{k+1} = \overrightarrow{X}_k + t_k\,\overrightarrow{d}_k,$$

затем осуществляется возврат на начало итерации.

Замечание. Для простоты изложения мы опустили некоторые детали, которые следует учесть при реализации рабочего алгоритма для повышения его эффективности. В частности, не рассматривался случай, когда через одну вершину проходит более чем n гиперплоскостей (проблема вырождения). Другая особая ситуация возникает, когда вектор антиградиента в некоторой вершине направлен внутрь допустимой

области, при этом нет необходимости идти по ребру, а можно сразу двигаться внутрь области по антиградиенту (эта ситуация встретится далее в примере).

П р и м е р. Возьмем знакомую задачу квадратичного программирования, которую мы ранее решали методом Вулфа (см. с. 68), однако сейчас мы включим условия неотрицательности переменных в общий список линейных ограничений:

$$\begin{aligned} f(x_1, x_2) = \quad & (x_1 - 2)^2 + (x_2 - 1)^2 \to \min, \\ x_1 \quad & +x_2 \quad \leqslant 2, \\ & x_2 \quad \leqslant 1, \\ -x_1 \quad & \quad \leqslant 0, \\ & -x_2 \quad \leqslant 0. \end{aligned}$$

В данном примере

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \overrightarrow{c} = \begin{pmatrix} -4 \\ -2 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{pmatrix}, \quad \overrightarrow{b} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}.$$

Геометрическая иллюстрация процесса решения приведена на рис. 12.7.

П о д г о т о в и т е л ь н ы й э т а п

Поскольку правая часть условий $\overrightarrow{b}$ неотрицательна, в качестве начальной точки возьмем $\overrightarrow{X}_0 = (0,\, 0)^T$.

И т е р а ц и я 1

Ш а г 1. Проверим, какие ограничения являются для точки $\overrightarrow{X}_0$ активными. Для этого вычислим $A\overrightarrow{X}_0 - \overrightarrow{b} = (-2,\, -1,\, 0,\, 0)^T$. Нулевые компоненты стоят в позициях 3 и 4, т. е. активны 3-е и 4-е ограничения, их число $q = 2$. Матрица A_q на данном шаге

составлена из 3-й и 4-й строк матрицы A и равна

$$A_q = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Матрица проектирования $P_q = \mathrm{I} - A_q^T(A_qA_q^T)^{-1}A_q$, как и следовало ожидать, получилась нулевой[2], потому что A_q — квадратная матрица (см. замечание 3 на с. 24).

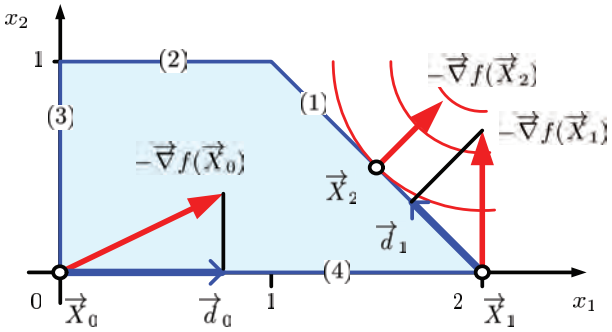


Рис. 12.7. Иллюстрация к примеру. Ограничения пронумерованы цифрами в скобках

Определим множители Лагранжа. Поскольку матрица A_q квадратная, то их можно вычислить по упрощенной формуле

$$\vec{Y} = -(A_qA_q^T)^{-1}A_q\vec{\nabla}f(\vec{X}_k) = -(A_q^T)^{-1}A_q^{-1}A_q\vec{\nabla}f(\vec{X}_k) = \\ = -(A_q^T)^{-1}\vec{\nabla}f(\vec{X}_k).$$

Подставив наши значения, получаем $\vec{Y} = (y_1, y_2)^T = (-4, -2)^T$. Оба множителя отрицательны, нулевой план неоптимальный.

Ш а г 2. Формально имеет место вариант «В», однако здесь мы сталкиваемся с ситуацией, описанной в замечании к алгоритму (см. с. 177). Вектор антиградиента в этой точке равен

[2]Подробные численные выкладки опущены ввиду их громоздкости. Предполагается, что их следует делать не вручную, а с помощью какого-либо математического пакета, лучше всего MATLAB.

$-\overrightarrow{\nabla} f(\overrightarrow{X}_0) = \overrightarrow{c} + 2D\overrightarrow{X}_0 = (4, 2)^T$, он направлен внутрь допустимой области (как мы знаем, множители Лагранжа есть коэффициенты разложения вектора антиградиента по внешним нормалям активных ограничений). Поэтому мы имеем полное право отправиться внутрь области ограничений. Тем не менее мы этим правом пользоваться не будем, а точно будем следовать алгоритму и пойдем по ребру.

Определим гиперплоскость (в нашем двумерном случае прямую), которая удаляется из активного набора. Сравнив $\|\overrightarrow{a}_1\|y_1 = 1 \cdot (-4)$ и $\|\overrightarrow{a}_2\|y_2 = 1 \cdot (-2)$, выбираем первое значение, соответствующее первой строке матрицы A_q, т. е. 3-му ограничению в исходной нумерации.

Удалив из A_q первую строку, получаем матрицу из одной строки $A_{q-1} = (0, -1)$, таким образом мы оставляем только 4-е ограничение и будем проектировать на него антиградиент. Вычислим матрицу проектирования

$$A_{q-1} = \mathrm{I} - A_{q-1}^T (A_{q-1} A_{q-1}^T)^{-1} A_{q-1} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix},$$

откуда направление движения на данном шаге

$$d_0 = -P_{q-1} \overrightarrow{\nabla} f(\overrightarrow{X}_0) = (4, 0)^T,$$

т. е. мы будем двигаться в положительном направлении по оси абсцисс.

Ш а г 3. По формуле (12.18) определяем моменты пересечения неактивных ограничений (1-го и 2-го). $T_1 = 0.5$, $T_2 = \infty$ (вектор $\overrightarrow{d}_1$ параллелен прямой $x_2 = 1$). Следовательно, предельная длина поиска $T = 0.5$.

Ш а г 4. По формуле (12.20) вычисляем точку оптимума для квадратичной функции сечения, получаем $t^* = 0.5$. Следовательно, длина шага на этой итерации $t_0 = \min(T, t^*) = 0.5$.

Ш а г 5. Делаем шаг и приходим в новую точку

$$\overrightarrow{X}_1 = \overrightarrow{X}_0 + t_0 \overrightarrow{d}_0 = (0, 0)^T + 0.5\,(4, 0)^T = (2, 0)^T.$$

И т е р а ц и я 2

Ш а г 1. Определяем активный набор ограничений: $A\overrightarrow{X}_0 - \overrightarrow{b} = (0, -1, -2, 0)^T$, следовательно, $q = 2$, активны 1-е и 4-е. Матрица, составленная из строк активного набора:

$$A_q = \begin{pmatrix} 1 & 1 \\ 0 & -1 \end{pmatrix}.$$

Действуем по образцу первой итерации. Матрицу проектирования не вычисляем, так как она заведомо нулевая. Определяем множители Лагранжа

$$\overrightarrow{Y} = -(A_q^T)^{-1}\overrightarrow{\nabla} f(\overrightarrow{X}_k) = (0, -2)^T.$$

План не оптимальный.

Ш а г 2. Опять имеем вариант «В». Из активного набора выводим вторую гиперплоскость (в исходной нумерации она 4-я). Составляем матрицу из одной строки $P_{q-1} = \overrightarrow{a}_1^T = (1, 1)$ и на ее основе вычисляем матрицу проектирования

$$A_{q-1} = \mathrm{I} - A_{q-1}^T(A_{q-1}A_{q-1}^T)^{-1}A_{q-1} = \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix},$$

затем проектируем градиент на 1-е ограничение:

$$\overrightarrow{d}_1 = -P_{q-1}\overrightarrow{\nabla} f(\overrightarrow{X}_1) = (-1, 1)^T.$$

Ш а г 3. По формуле (12.18) определяем моменты пересечения неактивных ограничений (2-го и 3-го в исходной нумерации). $T_2 = 1,\ T_3 = 2$. Следовательно, предельная длина поиска $T = 1$

Ш а г 4. По формуле (12.20) вычисляем точку оптимума для квадратичной функции сечения, получаем $t^* = 0.5$. Следовательно, длина шага на этой итерации $t_0 = \min(T, t^*) = 0.5$.

Ш а г 5. Делаем шаг и приходим в новую точку

$$\overrightarrow{X}_2 = \overrightarrow{X}_0 + t_0\overrightarrow{d}_0 = (2, 0)^T + 0.5\,(-1, 1)^T = (1.5, 0.5)^T.$$

И т е р а ц и я 3

Ш а г 1. Проверяем точку $\overrightarrow{X}_2$ на оптимальность. Вычисляя $A\overrightarrow{X}_2 - \overrightarrow{b} = (0,\ -0.5,\ -1.5,\ -0.5)^T$, видим, что имеется одно активное ограничение за номером 1, $q = 1$. Матрица активного набора состоит из одной строки $A_q = (1,\ 1)^T$, матрица проектирования (она та же, что и на шаге 2 в предыдущей итерации)

$$P_q = \mathrm{I} - A_q^T(A_qA_q^T)^{-1}A_q = \begin{pmatrix} 0.5 & -0.5 \\ -0.5 & 0.5 \end{pmatrix}.$$

По формулам (12.16) и (12.17) вычисляем проекцию антиградиента и множители Лагранжа:

$$P_q\overrightarrow{\nabla} f(\overrightarrow{X}_2) = (0,\ 0)^T,$$

$$\overrightarrow{Y} = (y_1, \ldots, y_q)^T = -(A_qA_q^T)^{-1}A_q\overrightarrow{\nabla} f(\overrightarrow{X}_2) = 1.$$

Условия Куна — Таккера выполнены, $\overrightarrow{X}^* = \overrightarrow{X}_2 = (1.5,\ 0.5)^T$. ▲

Замечание 1. Метод проектирования (метод активного набора) оказался особенно эффективным для задач, в которых целевая функция квадратична, а ограничения линейны. Подобно методу Вулфа (см. с. 65) он гарантирует нахождение т о ч н о г о решения за конечное число шагов, но при этом менее трудоемок. По этой причине он является стандартным методом квадратичного программирования (QP) в промышленных пакетах прикладных программ, о чем мы будем говорить в гл. 13.

Замечание 2. Если целевую функцию предположить линейной, то задача выпуклого программирования упрощается до задачи линейного программирования. В этом случае метод проектирования по своей сути превращается в симплексный метод, в котором переход от точки к точке осуществляется по ребрам допустимого многогранника.

12.4. Последовательное квадратичное программирование

Если метод проекции градиента можно рассматривать как адаптацию стандартных градиентных методов (т. е. методов первого порядка) к ситуации с ограничениями, то последовательное квадратичное программирование представляет собой попытку приспособить к такой же ситуации методы второго порядка (ньютоноподобные методы). Вместо того, чтобы явно учитывать ограничения, строится процедура численного решения уравнений Куна — Таккера.

Нахождение корня методом Ньютона

Проще всего продемонстрировать предлагаемый подход в классическом случае, когда ограничения задачи имеют вид строгих равенств, заданных вектор-функцией $\overrightarrow{G}(\overrightarrow{X}) = \big(g_1(\overrightarrow{X}), \ldots, g_m(\overrightarrow{X})\big)^T$:

$$\begin{aligned} f(\overrightarrow{X}) &\to \min, \\ \overrightarrow{G}(\overrightarrow{X}) &= 0. \end{aligned} \tag{12.21}$$

Функция Лагранжа для этой задачи равна

$$L(\overrightarrow{X}, \overrightarrow{Y}) = f(\overrightarrow{X}) + \overrightarrow{Y}^T \overrightarrow{G}(\overrightarrow{X}),$$

а необходимые условия оптимальности определяются (см. формулу (9.35) на с. 61) уравнениями Куна — Таккера относительно переменных $\overrightarrow{X}, \overrightarrow{Y}$:

$$\begin{cases} \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) = 0, \\ \overrightarrow{G}(\overrightarrow{X}) = 0. \end{cases} \tag{12.22}$$

Будем решать эту систему классическим методом Ньютона (см. с. 115). Если ввести обозначения для составного вектора переменных и составной вектор-функции

$$\overrightarrow{Z} = \begin{pmatrix} \overrightarrow{X} \\ \overrightarrow{Y} \end{pmatrix}, \quad \overrightarrow{F}(\overrightarrow{Z}) = \begin{pmatrix} \overrightarrow{F}_1 \\ \overrightarrow{F}_2 \end{pmatrix} = \begin{pmatrix} \overrightarrow{\nabla} L(\overrightarrow{X}, \overrightarrow{Y}) \\ \overrightarrow{G}(\overrightarrow{X}) \end{pmatrix},$$

то система (12.22) запишется в виде одного векторного уравнения $\overrightarrow{F}(\overrightarrow{Z})=0$. Ньютоновская итеративная процедура нахождения корня будет иметь вид

$$\overrightarrow{Z}_{k+1} = \overrightarrow{Z}_k - \mathrm{J}^{-1}(\overrightarrow{Z}_k)\overrightarrow{F}(\overrightarrow{Z}_k).$$

Входящая в формулу матрица Якоби J, учитывая составной характер функции $\overrightarrow{F}$ и ее аргумента $\overrightarrow{Z}$, может быть представлена в виде блочной матрицы

$$\mathrm{J}(\overrightarrow{Z}) = \frac{\partial \overrightarrow{F}(\overrightarrow{Z})}{\partial \overrightarrow{Z}} = \begin{pmatrix} \frac{\partial \overrightarrow{F}_1}{\partial \overrightarrow{X}} & \frac{\partial \overrightarrow{F}_1}{\partial \overrightarrow{Y}} \\ \frac{\partial \overrightarrow{F}_2}{\partial \overrightarrow{X}} & \frac{\partial \overrightarrow{F}_2}{\partial \overrightarrow{Y}} \end{pmatrix} = \begin{pmatrix} \mathrm{A}_{n\times n} & \mathrm{B}_{n\times m} \\ \mathrm{C}_{m\times n} & \mathrm{D}_{m\times m} \end{pmatrix},$$

элементы которой определяются следующими выражениями (см. сноску на с. 115):

$$\mathrm{A}_{n\times n} = \frac{\partial \overrightarrow{F}_1}{\partial \overrightarrow{X}} = \frac{\partial}{\partial \overrightarrow{X}} \overrightarrow{\nabla} L = \frac{\partial}{\partial \overrightarrow{X}} \frac{\partial L}{\partial \overrightarrow{X}} = \nabla^2 L,$$

$$\mathrm{B}_{n\times m} = \frac{\partial \overrightarrow{F}_1}{\partial \overrightarrow{Y}} = \frac{\partial}{\partial \overrightarrow{Y}} \overrightarrow{\nabla} L = \frac{\partial}{\partial \overrightarrow{Y}} \left(\overrightarrow{\nabla} f(\overrightarrow{X}) + \overrightarrow{Y}^T \nabla \overrightarrow{G}(\overrightarrow{X}) \right) = \nabla \overrightarrow{G},$$

$$\mathrm{C}_{m\times n} = \frac{\partial \overrightarrow{F}_2}{\partial \overrightarrow{X}} = \frac{\partial}{\partial \overrightarrow{X}} \overrightarrow{G}(\overrightarrow{X}) = \nabla^T \overrightarrow{G},$$

$$\mathrm{D}_{m\times m} = \frac{\partial \overrightarrow{F}_2}{\partial \overrightarrow{Y}} = \frac{\partial}{\partial \overrightarrow{Y}} \overrightarrow{G}(\overrightarrow{X}) = 0.$$

Следовательно, A представляет собой матрицу Гессе функции Лагранжа, а $\mathrm{C} = \mathrm{B}^T$ — матрицу Якоби (транспонированный матричный градиент) вектор-функции ограничений. Возвращаясь к первоначальным обозначениям, получаем рекуррентную формулу

$$\begin{pmatrix} \overrightarrow{X}_{k+1} \\ \overrightarrow{Y}_{k+1} \end{pmatrix} = \begin{pmatrix} \overrightarrow{X}_k \\ \overrightarrow{Y}_k \end{pmatrix} + \begin{pmatrix} \Delta \overrightarrow{X}_k \\ \Delta \overrightarrow{Y}_k \end{pmatrix},$$

где величина итеративного шага по переменным определяется выражением

$$\begin{pmatrix}\Delta\overrightarrow{X}_k\\ \Delta\overrightarrow{Y}_k\end{pmatrix} = -\begin{pmatrix}\nabla^2 L & \nabla\overrightarrow{G}\\ \nabla^T\overrightarrow{G} & 0\end{pmatrix}^{-1}\begin{pmatrix}\overrightarrow{\nabla}L\\ \overrightarrow{G}\end{pmatrix}. \qquad (12.23)$$

Условная квадратичная минимизация

Наряду с изложенной выше, возможна еще одна интерпретация классического ньютоновского процесса. Изучая оптимизацию без ограничений, мы замечали (см. с. 115), что существует тесная связь между ньютоновским процессом нахождения корня векторного уравнения $\overrightarrow{\nabla} f(\overrightarrow{X}) = 0$ и итеративной процедурой минимизации целевой функции $f(\overrightarrow{X})$, аппроксимируемой параболоидом. Оказывается, подобная связь имеет место и при наличии ограничений, однако при этом мы будем иметь дело не с самой целевой функцией, а с функцией Лагранжа. Сейчас мы покажем, что каждая итерация классического ньютоновского метода решения уравнений Куна — Таккера эквивалентна решению некоторой вспомогательной оптимизационной задачи.

Пусть на k-й итерации ньютоновского процесса решения системы нелинейных уравнений Куна — Таккера (12.22) получены оценки $(\overrightarrow{X}_k,\ \overrightarrow{Y}_k)$. Аппроксимируем функцию Лагранжа в окрестности этой точки модельной к в а д р а т и ч н о й функцией от шага $\overrightarrow{d}$:

$$Q(\overrightarrow{d}) = L(\overrightarrow{X}_k + \overrightarrow{d}, \overrightarrow{Y}_k) = L + \overrightarrow{\nabla}^T L\,\overrightarrow{d} + \frac{1}{2}\overrightarrow{d}^T\nabla^2 L\,\overrightarrow{d}, \qquad (12.24)$$

а для ограничений $\overrightarrow{G}(\overrightarrow{X}_k) = 0$ воспользуемся л и н е й н ы м приближением

$$\overrightarrow{P}(\overrightarrow{d}) = \overrightarrow{G}(\overrightarrow{X}_k + \overrightarrow{d}) = \overrightarrow{G} + \nabla^T\overrightarrow{G}\,\overrightarrow{d} = 0. \qquad (12.25)$$

Там, где аргумент у функций $L, \overrightarrow{\nabla}L, \overrightarrow{\nabla}^2 L, \overrightarrow{G}, \nabla\overrightarrow{G}$ опущен, они считаются вычисленными при фиксированных $\overrightarrow{X}_k,\ \overrightarrow{Y}_k$.

Поставим вспомогательную задачу нахождения такого шага $\overrightarrow{d}$, который минимизирует квадратичную аппроксимацию функции Лагранжа (12.24) при линейных ограничениях (12.25). Для ее решения, пока ограничения заданы равенствами, нам не потребуется применять численные методы. Решение легко найти аналитически, если применить условия оптимальности первого порядка.

Составим функцию Лагранжа для вспомогательной задачи (обозначим ее каллиграфической буквой $\mathcal{L}$, чтобы не путать с функцией Лагранжа L основной задачи), множители Лагранжа вспомогательной задачи обозначим $\overrightarrow{\lambda}$:

$$\mathcal{L}(\overrightarrow{d}, \overrightarrow{\lambda}) = Q(\overrightarrow{d}) + \overrightarrow{\lambda}^T[\overrightarrow{P}(\overrightarrow{d})] =$$

$$= L + \overrightarrow{\nabla}^T L\,\overrightarrow{d} + \frac{1}{2}\overrightarrow{d}^T\nabla^2 L\,\overrightarrow{d} + \overrightarrow{\lambda}^T[\overrightarrow{G} + \nabla^T\overrightarrow{G}\,\overrightarrow{d}].$$

Условия оптимальности первого порядка для ограничений-равенств имеют вид

$$\frac{\partial\mathcal{L}(\overrightarrow{d}, \overrightarrow{\lambda})}{\partial\overrightarrow{d}} = \overrightarrow{\nabla} L + \nabla^2 L\,\overrightarrow{d} + \nabla\overrightarrow{G}\,\overrightarrow{\lambda} = 0,$$

$$\frac{\partial\mathcal{L}(\overrightarrow{d}, \overrightarrow{\lambda})}{\partial\overrightarrow{\lambda}} = \overrightarrow{G} + \nabla^T\overrightarrow{G}\,\overrightarrow{d} = 0.$$

Получилась система л и н е й н ы х уравнений относительно переменных $\overrightarrow{d}$, $\overrightarrow{\lambda}$, которую можно переписать в виде

$$\begin{pmatrix} \nabla^2 L\,\overrightarrow{d} + \nabla\overrightarrow{G}\,\overrightarrow{\lambda} \\ \nabla^T\overrightarrow{G}\,\overrightarrow{d} \end{pmatrix} = \begin{pmatrix} \nabla^2 L & \nabla\overrightarrow{G} \\ \nabla^T\overrightarrow{G} & 0 \end{pmatrix} \begin{pmatrix} \overrightarrow{d} \\ \overrightarrow{\lambda} \end{pmatrix} = \begin{pmatrix} -\overrightarrow{\nabla} L \\ -\overrightarrow{G} \end{pmatrix}.$$

Отсюда, если матрица системы невырождена,

$$\begin{pmatrix} \overrightarrow{d} \\ \overrightarrow{\lambda} \end{pmatrix} = -\begin{pmatrix} \nabla^2 L & \nabla\overrightarrow{G} \\ \nabla^T\overrightarrow{G} & 0 \end{pmatrix}^{-1} \begin{pmatrix} \overrightarrow{\nabla} L \\ \overrightarrow{G} \end{pmatrix}. \qquad (12.26)$$

Видим, что выражение (12.26) совершенно идентично (12.23): $\Delta\overrightarrow{X}_k = \overrightarrow{d}$, $\Delta\overrightarrow{Y}_k = \lambda$.

Эта интерпретация дает возможность представить процесс оптимизации многомерной функции с ограничениями-равенствами в виде последовательной условной минимизации, когда на каждой итерации решается подзадача нахождения минимума квадратичной аппроксимации функции Лагранжа с линеаризованными ограничениями. При положительной определенности матрицы системы уравнений (12.26) итерационный процесс сойдется к оптимальным значениям $\overrightarrow{X}^*,\ \overrightarrow{Y}^*$.

П р и м е р. Пусть дана задача минимизации с двумя переменными и одним ограничением-равенством (рис. 12.8):

$$f(x_1, x_2) = -x_1^2 + (x_2 - 2)^2 \to \min,$$
$$g(x_1, x_2) = 4x_1^2 + x_2^2 - 1 = 0,$$

имеющая очевидное решение $\overrightarrow{X}^* = (0,\ 1)^T$.

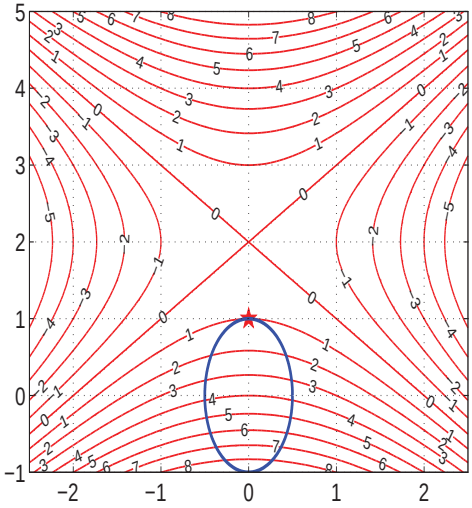


Рис. 12.8. Изолинии целевой функции и одно нелинейное ограничение-равенство. Оптимальный план отмечен звездочкой

Функция Лагранжа, ее градиент и гессиан, а также градиент нелинейного ограничения равны соответственно

$$L(x_1, x_2, y) = -x_1^2 + (x_2 - 2)^2 + y(4x_1^2 + x_2^2 - 1),$$

$$\overrightarrow{\nabla} L = \begin{pmatrix} -2x_1 + 8x_1 y \\ 2x_2 - 4 + 2x_2 y \end{pmatrix}, \nabla^2 L = \begin{pmatrix} -2 + 8y & 0 \\ 0 & 2 + 2y \end{pmatrix}, \overrightarrow{\nabla} g = \begin{pmatrix} 8x_1 \\ 2x_2 \end{pmatrix}.$$

Возьмем в качестве начальных значений $\overrightarrow{X} = (2,\ 4)^T,\ y = 0.5$ и разберем подробно первую итерацию. В исходной точке

$$\overrightarrow{\nabla} L = \begin{pmatrix} 4 \\ 8 \end{pmatrix},\ \nabla^2 L = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix},\ g = 31,\ \overrightarrow{\nabla} g = \begin{pmatrix} 16 \\ 8 \end{pmatrix},$$

и вспомогательная квадратичная подзадача (12.24)–(12.25) ставится в виде

$$Q(d_1, d_2) = 4d_1 + 8d_2 + 2d_1^2 + 3d_2^2 \to \min,$$
$$P(d_1, d_2) = 31 + 16d_1 + 8d_2 = 0.$$

Ее решение по формуле (12.26) даст величину шага по переменным и множителю Лагранжа

$$\begin{pmatrix} d_1 \\ d_2 \\ \lambda \end{pmatrix} = -\left(\begin{array}{cc|c} 2 & 0 & 16 \\ 0 & 3 & 8 \\ \hline 16 & 8 & 0 \end{array}\right)^{-1} \begin{pmatrix} 4 \\ 8 \\ 31 \end{pmatrix} = \begin{pmatrix} -0.8036 \\ -2.2679 \\ -0.1496 \end{pmatrix}.$$

Таким образом, следующее приближение дает результат $\overrightarrow{X}_1 = \overrightarrow{X}_0 + \overrightarrow{d} = (2,\ 4)^T - (0.8036,\ 2.2679)^T = (1.1964,\ 1.7321)^T$, $y_1 = y_0 + \lambda = 0.5 - 0.1496 = 0.3504$.

На рис 12.9 изображены первые четыре итерации условной квадратичной минимизации для данной задачи. Для каждой итерации на фоне изолиний соответствующей функции Лагранжа изображены нелинейное ограничение, его линейная аппроксимация, а также приближение $\overrightarrow{X}_k$ на данной итерации, которое является точкой касания аппроксимирующей прямой с одной из изолиний. Как видим, в отличие от целевой функции, лагранжиан

в данном примере является всюду выпуклой функцией, причем от итерации к итерации он меняется незначительно. Итерационный процесс быстро сходится к точке оптимального решения $\overrightarrow{X}^* = (0,\ 1)^T,\ y^* = 1.$ ▲

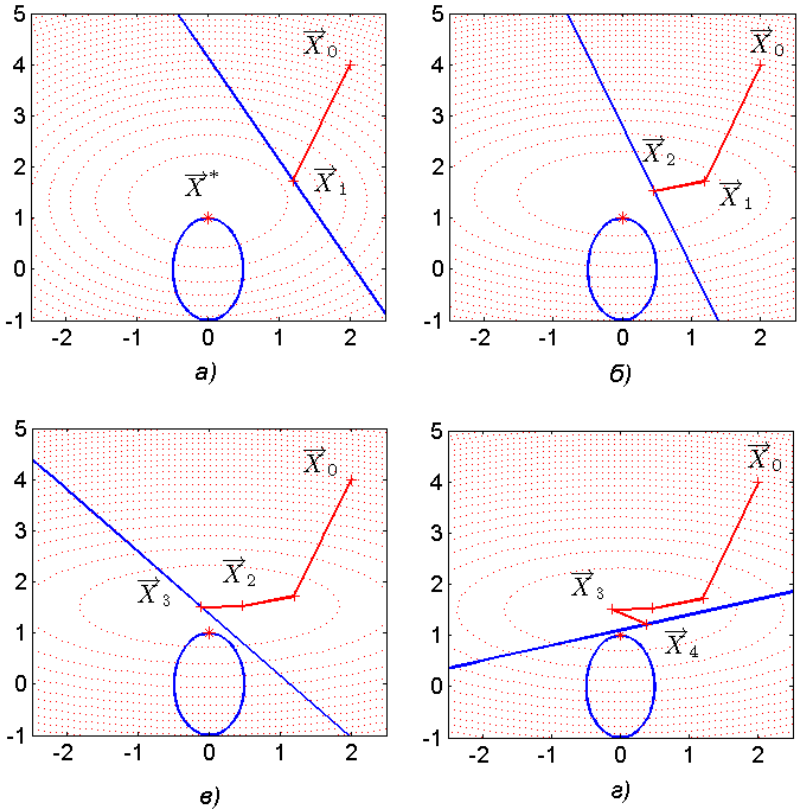


Рис. 12.9. Первые итерации условной квадратичной минимизации для задачи с одним нелинейным ограничением-равенством

Последовательное квадратичное программирование

Принципиальным преимуществом интерпретации ньютоновского процесса как последовательной квадратичной минимизации является возможность его распространения на задачи с ограничениями в форме неравенств, однако при этом необходимо учесть новые обстоятельства и ввести в алгоритм необходимые корректировки.

Подзадача квадратичного программирования. В случае, когда ограничения присутствуют в форме неравенств, линеаризованные ограничения также будут неравенствами, подзадача (12.24)–(12.25) определения ньютоновского шага приобретет вид (постоянное слагаемое L, не влияющее на решение, опущено)

$$\begin{aligned} Q(\overrightarrow{d}) &= \overrightarrow{\nabla}^T L\,\overrightarrow{d} + \tfrac{1}{2}\,\overrightarrow{d}^T \nabla^2 L\,\overrightarrow{d} \to \min, \\ \overrightarrow{P}(\overrightarrow{d}) &= \overrightarrow{G} + \nabla^T \overrightarrow{G}\,\overrightarrow{d} \leqslant 0. \end{aligned} \tag{12.27}$$

Это — задача квадратичного программирования (см. с. 65)[3], таким образом, последовательная условная квадратичная минимизация превращается в *последовательное квадратичное программирование (sequential quadratic programming — SQP)*. Из-за присутствия неравенств ее нельзя решить аналитически, однако для квадратичного программирования существует целый ряд весьма эффективных методов, например метод Вулфа, который мы рассматривали в гл. 9, или приведенный в предыдущем параграфе метод проектирования градиента (метод активного набора).

Линейный поиск при ограничениях. Решив подзадачу квадратичного программирования *(QP subproblem)* на k-й итерации, мы получим ньютоновский шаг $\overrightarrow{d}_k$; далее возникает вопрос, как к нему относиться. Поскольку целевая функция в реальных условиях может быть весьма сложной, шаг на полную (единичную) длину может оказаться не самым удачным. Когда подобная

[3] В стандартной постановке квадратичного программирования требуется неотрицательность переменных, но это обстоятельство не является принципиально важным.

проблема возникла в оптимизации без ограничений (см. с. 118), предлагалось дополнить классический ньютоновский метод линейным поиском, т. е. использовать $\overrightarrow{d}_k$ только как направление, а величину перемещения определить поиском минимума функции одномерного сечения в данном направлении. Аналогичный подход обычно используется и в последовательном квадратичном программировании, однако, как мы отмечали ранее (см. с. 166), при этом возникает дополнительная проблема ограничения длины шага, иначе можно выйти за пределы области допустимых решений. Поскольку найти значение t, при котором луч $\overrightarrow{X}_k + \overrightarrow{t}\, d_k$ протыкает область нелинейных ограничений, довольно сложно, для определения длины шага применяется описанная на с. 167 методика минимизации функции выигрыша (merit function).

Вычисление гессиана функции Лагранжа. Как следует из (12.27), для постановки подзадачи квадратичного программирования необходимо знать гессиан функции Лагранжа в текущей точке $\nabla^2 L(\overrightarrow{X}_k, \overrightarrow{Y}_k)$. Если целевая функция и ограничения представляют собой простые выражения, то гессиан вычисляется непосредственно, в этом случае излагаемый метод должен быть отнесен к методам второго порядка.

Если же вычисление гессиана затруднительно, то на помощь приходит идея, заложенная в квазиньютоновских алгоритмах безусловной оптимизации (см. с. 121). Гессиан оптимизируемой функции (в данном случае функции Лагранжа) не задается заранее, а восстанавливается по результатам предыдущих итераций. Это значит, что на подготовительном этапе гессиан аппроксимируется единичной матрицей, а на последующих итерациях происходит его *обновление* — *update* по формулам Дэвидона — Флетчера — Пауэлла (ДФП — DFP) либо Бройдена — Флетчера— Гольдфарба — Шанно (БФГШ — BFGS), причем, как показали многочисленные эксперименты, второй вариант предпочтительнее.

Обновление множителей Лагранжа. Каждая итерация метода (она называется *главной* итерацией, потому что подзада-

ча квадратичного программирования, в свою очередь, решается последовательностью собственных итераций) предполагает перевычисление не только координат текущей точки $\overrightarrow{X}_k$, но и соответствующих этой точке множителей Лагранжа $\overrightarrow{Y}_k$. Эти обновленные значения могут быть получены в процессе решения подзадачи квадратичного программирования. Как метод Вулфа, так и метод активного набора наряду с оптимальным решением $\overrightarrow{X}^*$ дают соответствующие значения $\overrightarrow{Y}^*$.

Замечание 1. Из приведенного схематического описания видно, что последовательное квадратичное программирование образует целый класс схожих методов оптимизации. Общим в нем является только принцип определения направления шага, другие составляющие (определение длины шага, механизм вычисления гессиана, начальные значения и способ обновления множителей Лагранжа и т. д.) могут быть воплощены самыми различными способами.

Замечание 2. Мы рассмотрели только общую идею подхода, в практической реализации метода необходимо учитывать многие детали и тонкости, влияющие на устойчивость и скорость сходимости. В итоге полные алгоритмы последовательного квадратичного программирования получаются довольно сложными, однако эта сложность окупается проверенной на практике эффективностью. Как утверждают разработчики промышленных пакетов оптимизации[4], последовательное квадратичное программирование представляет собой *state of the art* (достигнутый к настоящему времени уровень знаний в науке или технике) в оптимизации с ограничениями.

П р и м е р. Пусть в предыдущем примере (см. с. 187) ограничение задано неравенством

$$f(x_1, x_2) = -x_1^2 + (x_2 - 2)^2 \to \min,$$
$$g(x_1, x_2) = 4x_1^2 + x_2^2 - 1 \leqslant 0.$$

[4]См., например, документацию по пакету *Optimization Toolbox* в системе MATLAB версии 2010 г. на сайте MathWorks Inc.

Разумеется, точка оптимума будет той же самой, однако теперь для ее нахождения потребуется применить в полном объеме метод последовательного квадратичного программирования.

Ниже в таблице приведены промежуточные результаты главных итераций. Исходная точка $\overrightarrow{X}_0 = (1.5,\ 1.5)^T, y_0 = 0.5$. В линейном поиске задействована функция выигрыша, включающая штрафное слагаемое

$$\psi(t) = f(\overrightarrow{x}_k + t\,\overrightarrow{d}_k) + R\max[0,\ g(\overrightarrow{x}_k + t\,\overrightarrow{d}_k)],$$

в которой штрафной параметр $R = 1$. Обновление гессиана функции Лагранжа проводилось по методике BFGS.

Условный экстремум $\overrightarrow{X}^* = (0,\ 1)^T$ найден за 5 итераций.

k	x_1^k	x_2^k	y_k
0	1.5000	1.5000	0.5000
1	−0.0273	1.2455	0.5000
2	0.0561	1.0330	0.6894
3	−0.0006	1.0315	0.9410
4	0.0005	1.0000	0.9839
5	−0.0000	1.0000	1.0000

Сравнивая восстановленный гессиан функции Лагранжа с истинным значением

$$H_5(\overrightarrow{X}_5, y_5) = \begin{pmatrix} 5.8717 & -0.0025 \\ -0.0025 & 3.9999 \end{pmatrix}, \quad H(\overrightarrow{X}^*, y^*) = \begin{pmatrix} 6 & 0 \\ 0 & 4 \end{pmatrix},$$

видим вполне удовлетворительное совпадение. ▲

12.5. Метод линейной аппроксимации

Группа аппроксимационных методов основана на приближении нелинейных задач линейными с последующим решением задач линейного программирования.

Общая идея

Пусть имеется задача выпуклого программирования в стандартном виде

$$f(\overrightarrow{X}) = \min,$$
$$g_i(\overrightarrow{X}) \leqslant 0, \quad i = 1, \ldots, m,$$
$$\overrightarrow{X} \geqslant 0.$$

Так как f и g_i выпуклые, то область планов является выпуклой областью, каждая точка этой области может быть представлена в виде выпуклой комбинации других (опорных) точек $\overrightarrow{X}_1, \ldots, \overrightarrow{X}_s$:

$$\overrightarrow{X} = \sum_{k=1}^{s} y_k \overrightarrow{X}_k, \quad 0 \leqslant y_k \leqslant 1, \quad \sum_{k=1}^{s} y_k = 1.$$

При этом в силу выпуклости

$$f(\overrightarrow{X}) = f\Big(\sum_{k=1}^{s} y_k \overrightarrow{X}_k\Big) \leqslant \sum_{k=1}^{s} y_k f(\overrightarrow{X}_k),$$
$$g_i(\overrightarrow{X}) = g_i\Big(\sum_{k=1}^{s} y_k \overrightarrow{X}_k\Big) \leqslant \sum_{k=1}^{s} y_k g_i(\overrightarrow{X}_k), \quad i = 1, \ldots, m. \qquad (12.28)$$

Основываясь на этих неравенствах, исходную нелинейную задачу можно заменить на мажорирующую ее линеаризованную:

$$\begin{aligned} L(\overrightarrow{Y}) = \ & \sum_{k=1}^{s} y_k f(\overrightarrow{X}_k) \to \min, \\ & \sum_{k=1}^{s} y_k g_i(\overrightarrow{X}_k) \leqslant 0, \quad i = 1, \ldots, m, \\ & \sum_{k=1}^{s} y_k = 1 \\ & y_k \geqslant 0, \quad k = 1, \ldots, s. \end{aligned} \qquad (12.29)$$

Таким образом, исходную нелинейную задачу аппрксимировали задачей линейного программирования относительно переменных $y_1, \ldots, y_s$. Решая ее, получаем оптимальные значения y_k^* и по

ним восстанавливаем $\overrightarrow{X}^*$:

$$\overrightarrow{X}^* = \sum_{k=1}^{s} y_k^* \overrightarrow{X}_k.$$

Сепарабельное программирование

Сложность практического применения метода заключается в выборе опорных точек $\overrightarrow{X}_k$. Для общей задачи выпуклого программирования это сделать не удается, однако эта задача решается легко и эффективно для одного важного частного случая, когда целевая функция и ограничения являются *сепарабельными*.

Определение. *Функция* $f(\overrightarrow{X})$ *называется сепарабельной, если*

$$f(\overrightarrow{X}) = \sum_{j=1}^{n} f_j(x_j).$$

Задача сепарабельного программирования заключается в минимизации сепарабельной целевой функции при сепарабельных ограничениях:

$$\begin{aligned} f(\overrightarrow{X}) = &\sum_{j=1}^{n} f_j(x_j) \to \min, \\ &\sum_{j=1}^{n} g_{ij}(x_j) \leqslant 0, \quad i = 1, \ldots, m, \\ &x_j \geqslant 0, \quad j = 1, \ldots, n. \end{aligned} \tag{12.30}$$

Замечание. Многие задачи приводятся к сепарабельному виду путем подходящей замены переменных. Например, произведение $x_i x_j$ может быть превращено в сумму логарифмированием.

Принципиальное упрощение по сравнению с общим случаем состоит в том, что в задаче сепарабельного программирования

каждую переменную x_j линеаризованной задачи (12.29) можно аппроксимировать в отдельности по S_j точкам в области ее определения:

$$x_j = \sum_{k=1}^{s_j} y_{jk} x_{jk},$$

где x_{jk} — числа (координаты выбранных точек), а y_{jk} — переменные. Используется двойная индексация: первый индекс — номер аппроксимируемой переменной, второй — номер точки.

С учетом этого линеаризованная задача будет иметь $S_1 + S_2 + \cdots + S_n$ переменных, $m + n$ ограничений и запишется в виде

$$\begin{aligned}
L(\overrightarrow{Y}) = &\sum_{j=1}^{n} \sum_{k=1}^{s_j} f_j(x_{jk}) y_{jk} \to \min, \\
&\sum_{j=1}^{n} \sum_{k=1}^{s_j} g_{ij}(x_{jk}) y_{jk} \leqslant 0, \quad i = 1, \ldots, m, \\
&\sum_{k=1}^{s_j} y_{jk} = 1, \quad j = 1, \ldots, n, \\
&y_{jk} \geqslant 0.
\end{aligned}$$

П р и м е р. Решим методом линеаризации задачу выпуклого программирования, которую мы исследовали аналитически (с. 56). Число переменных $n = 2$, число ограничений $m = 3$, условия на неотрицательность переменных записаны в явном виде:

$$\begin{aligned}
&f(\overrightarrow{X}) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min, \\
&g_1(\overrightarrow{X}) = x_1^2 + (x_2^2 - 4) \leqslant 0, \\
&g_2(\overrightarrow{X}) = -x_1 + x_2 \leqslant 0, \\
&g_3(\overrightarrow{X}) = x_2 - 1 \leqslant 0, \\
&x_1, x_2 \geqslant 0.
\end{aligned}$$

Целевая функция и ограничения сепарабельны, поэтому каждую из одномерных функций можно аппроксимировать независимо. Области их определения, как следует из геометрического построения (см. рис. 9.14), достаточно взять в пределах $0 \leqslant x_1 \leqslant 2$, $0 \leqslant x_2 \leqslant 1$.

Для начала сделаем очень грубое приближение. Для переменной x_1 выберем три ($S_1 = 3$) узловые точки с шагом $\Delta x_1 = 1$: $x_{11} = 0, x_{12} = 1$, $x_{13} = 2$, а для переменной x_2 — только две ($S_2 = 2$) с тем же шагом: $x_{21} = 0$, $x_{22} = 1$. Все одномерные функции и их значения в узловых точках приведены ниже в таблице.

k	1	2	3
x_{1k}	0	1	2
$f_1(x_{1k}) = (x_{1k} - 2)^2$	4	1	0
$g_{11}(x_{1k}) = x_{1k}^2$	0	1	4
$g_{21}(x_{1k}) = -x_{1k}$	0	-1	-2
$g_{31}(x_{1k}) = 0$	0	0	0

k	1	2
x_{2k}	0	1
$f_2(x_{2k}) = (x_{2k} - 2)^2$	4	1
$g_{12}(x_{2k}) = x_{2k}^2 - 4$	-4	-3
$g_{22}(x_{2k}) = x_{2k}$	0	1
$g_{32}(x_{2k}) = x_{2k} - 1$	-1	0

Аппроксимирующая задача линейного программирования имеет $S_1 + S_2 = 5$ переменных и $m + n = 5$ ограничений:

$$
\begin{aligned}
L(\overrightarrow{Y}) = \quad & 4y_{11} + y_{12} + 4y_{21} + y_{22} && \to \min, \\
& y_{12} + 4y_{13} - 4y_{21} - 3y_{22} && \leqslant 0, \\
& -y_{12} - 2y_{13} + y_{22} && \leqslant 0, \\
& -y_{21} && \leqslant 0, \\
& y_{11} + y_{12} + y_{13} && = 1, \\
& y_{21} + y_{22} && = 1, \\
& y_{11}, \ldots, y_{22} \geqslant 0.
\end{aligned}
$$

Решив эту задачу симплексным методом, получаем

$$
\begin{aligned}
& y_{11}^* = 0, \quad y_{12}^* = 0.3333, \quad y_{13}^* = 0.6667, \\
& y_{21}^* = 0, \quad y_{22}^* = 1.
\end{aligned}
$$

Восстанавливая по ним оптимальные значения переменных исходной задачи, имеем

$$x_1^* = x_{11}y_{11}^* + x_{12}y_{12}^* + x_{13}y_{13}^* = 1 \cdot 0.333 + 2 \cdot 0.6667 = 1,6667,$$
$$x_2^* = x_{21}y_{21}^* + x_{22}y_{22}^* = 1 \cdot 1 = 1,$$
$$f^* = 1.1111.$$

Напомним точные значения: $x_1^* = \sqrt{3} = 1.7320$, $x_2^* = 1$, $f^* = 1.0718$. Как видим, приближение получилось не слишком плохим, несмотря на то, что мы сделали очень грубую аппроксимацию нелинейных функций отрезками прямых с шагом $\Delta x_1 = \Delta x_2 = 1$. Если величину шага уменьшить, результат получится более точным:

Шаг	x_1^*	x_2^*	f^*
0.5	1.7143	1	1.0816
0.2	1.7294	1	1.0732
0.1	1.7314	1	1.0721
0.05	1.7319	1	1.0719
0.02	1.7320	1	1.0718

▲

Замечание 1. Метод линеаризации работает тем эффективнее, чем меньше отличаются присутствующие в задаче нелинейные функции от линейных. При этом в силу выпуклости функций g_i приближение всегда будет находиться внутри допустимой области.

Замечание 2. Описанный метод с небольшой модификацией применим для решения н е в ы п у к л ы х задач сепарабельного программирования, правда, в этом случае он гарантирует нахождение только локального минимума. Если для некоторого индекса j хотя бы одна из функций f_j, g_{ij} невыпукла, то для правильной ее аппроксимации кусочно-линейной зависимостью весовые коэффициенты y_{jk} для данного j должны не только удовлетворять обычным соотношениям $y_{jk} \geqslant 0$, $\sum_k y_{jk} = 1$, но, кроме того, их ненулевые значения допускаются только д л я с о с е д н и х индексов k. Для этого в ходе работы

симплексного алгоритма необходимо следить за тем, чтобы на очередной итерации в базис включался только вектор, соседствующий по индексу k с одним из векторов текущего базиса. Детали реализации см. в [23, с. 317].

12.6. Метод секущих плоскостей

Этот аппроксимационный метод — *cutting plane method* — был предложен Дж. Келли в 1960 г. и носит его имя [23, с. 331].

Линеаризация целевой функции

Прежде чем излагать сам метод, заметим, что любая задача минимизации нелинейной целевой функции при выпуклых ограничениях совершенно точно сводится к задаче минимизации линейной целевой функции при выпуклых же ограничениях.

Пусть задача выпуклого нелинейного программирования задана в стандартном виде

$$\begin{aligned} &f(x_1,\ldots,x_n)\to\min,\\ &g_i(x_1,\ldots,x_n)\leqslant 0,\quad i=1,\ldots,m,\\ &x_j\geqslant 0,\quad j=1,\ldots,n. \end{aligned} \tag{12.31}$$

Приведение к виду с линейной целевой функцией достигается ценой небольшого увеличения размерности. Введем дополнительную $n+1$-ю переменную, дополнительное $m+1$-е ограничение и составим задачу:

$$\begin{aligned} &x_{n+1}\to\min,\\ &\tilde g_i(x_1,\ldots,x_{n+1})\leqslant 0,\quad i=1,\ldots,m+1,\\ &x_j\geqslant 0,\quad j=1,\ldots,n+1, \end{aligned} \tag{12.32}$$

где обозначено

$$\tilde g_i(x_1,\ldots,x_{m+1})=\begin{cases} g_i(x_1,\ldots,x_n), & i=1,\ldots,m,\\ f(x_1,\ldots,x_n)-x_{n+1}, & i=m+1. \end{cases}$$

Так как f и g_i выпуклы, то выпуклы и $\tilde{g}_i$, преобразованная задача есть задача с линейной целевой функцией и выпуклыми ограничениями. Легко видеть, что новая задача эквивалентна исходной.

Теория метода

Пусть имеется задача минимизации л и н е й н о й целевой функции при выпуклых нелинейных ограничениях (если задача преобразовывалась, то рассматривается уже преобразованная задача, нумерация переменных и ограничений новая, знаки тильды опущены):

$$\begin{aligned} &L(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} \to \min, \\ &g_i(\overrightarrow{X}) \leqslant 0,\ i = 1, \ldots, m, \\ &\overrightarrow{X} \geqslant 0. \end{aligned} \tag{12.33}$$

Метод Кэлли является методом первого порядка, так как предполагает знание градиентов ограничивающих функций $\overrightarrow{\nabla} g_i(\overrightarrow{X})$. По концепции он близок методу Гомори в целочисленном программировании [9, гл. 7], также основанному на отсечениях.

Метод предполагает сведение нелинейной задачи к последовательности задач линейного программирования. Поскольку целевая функция в задаче (12.33) уже линейна, осталось линеаризовать ограничения. Это предлагается сделать следующим образом.

Пусть $\overrightarrow{X}^{(0)}$ — некоторая начальная точка. Для каждой нелинейной функции $g_i(\overrightarrow{X})$ построим линеаризованную функцию

$$G_i^{(0)}(\overrightarrow{X}) = g_i(\overrightarrow{X}^{(0)}) + \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})(\overrightarrow{X} - \overrightarrow{X}^{(0)}),$$

которая действительно является линейной функцией векторного аргумента $\overrightarrow{X}$, в чем легко убедиться, раскрыв скобки:

$$\begin{aligned} &G_i^{(0)}(\overrightarrow{X}) = g_i(\overrightarrow{X}^{(0)}) + \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})(\overrightarrow{X} - \overrightarrow{X}^{(0)}) = \\ &= \underbrace{\overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})}_{(\overrightarrow{a}_i^{(0)})^T}\overrightarrow{X} + \underbrace{g_i(\overrightarrow{X}^{(0)}) - \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})\overrightarrow{X}^{(0)}}_{b_i^{(0)}} = (\overrightarrow{a}_i^{(0)})^T\overrightarrow{X} + b_i^{(0)}. \end{aligned}$$

Тогда одно неравенство $G_i^{(0)}(\overrightarrow{X}) = (\overrightarrow{a}_i^{(0)})^T\overrightarrow{X} + b_i^{(0)} \leqslant 0$ определит полупространство, а совокупность m неравенств в сочетании с ограничениями на неотрицательность задаст выпуклое многогранное множество $M^{(0)}$.

Легко убедиться, что оно включает в себя множество планов исходной задачи (12.33), которое мы обозначим через D. Действительно, по свойству 6 выпуклых функций (с. 31)

$$g_i(\overrightarrow{X}) \geqslant g_i(\overrightarrow{X}^{(0)}) + \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})(\overrightarrow{X} - \overrightarrow{X}^{(0)}) = G_i^{(0)}(\overrightarrow{X}).$$

Отсюда следует, что если $\overrightarrow{X} \in D$, т. е. $g_i(\overrightarrow{X}) \leqslant 0$, то $G_i^{(0)}(\overrightarrow{X}) \leqslant$ $\leqslant g_i(\overrightarrow{X}) \leqslant 0$, т. е. $\overrightarrow{X} \in M^{(0)}$.

Теперь можно поставить задачу линейного программирования (задачу 0)

$$\begin{aligned} &L(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} \to \min, \\ &(\overrightarrow{a}_i^{(0)})^T\overrightarrow{X} \leqslant -b_i^{(0)},\ i = 1, \ldots, m, \\ &\overrightarrow{X} \geqslant 0, \end{aligned}$$

ее решение даст точку $\overrightarrow{X}^{(1)}$, являющуюся одной из крайних точек множества $M^{(0)}$.

Если $\overrightarrow{X}^{(1)}$ удовлетворяет условиям исходной задачи, т. е. все $g_i(\overrightarrow{X}^{(1)}) \leqslant 0$, то это и есть искомый оптимальный план $\overrightarrow{X}^*$. Если же $\overrightarrow{X}^{(1)}$ нарушает ограничения, т. е. для некоторого p имеет место $g_p(\overrightarrow{X}^{(1)}) > 0$, то строится линейное отсечение

$$G_p^{(1)}(\overrightarrow{X}) = g_p(\overrightarrow{X}^{(1)}) + \overrightarrow{\nabla}^T g_p(\overrightarrow{X}^{(1)})(\overrightarrow{X} - \overrightarrow{X}^{(1)}) = (\overrightarrow{a}_p^{(1)})^T\overrightarrow{X} + b_p^{(1)} \leqslant 0.$$

Можно показать, что это отсечение является правильным, т. е. точка $\overrightarrow{X}^{(1)}$ будет отсечена от множества D. Действительно, если подставить координаты этой точки в уравнение гиперплоскости, получим

$$G_p^{(1)}(\overrightarrow{X}^{(1)}) = g_p(\overrightarrow{X}^{(1)}) + \overrightarrow{\nabla}^T g_p(\overrightarrow{X}^{(1)})(\overrightarrow{X}^{(1)} - \overrightarrow{X}^{(1)}) = g_p(\overrightarrow{X}^{(1)}) > 0.$$

Добавляя полученное отсечение к ограничениям задачи 0, получаем задачу 1

$$\begin{aligned}
&L(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} \to \min,\\
&(\overrightarrow{a}_i^{(0)})^T\overrightarrow{X} \leqslant -b_i^{(0)},\ i = 1,\dots,m,\\
&(\overrightarrow{a}_p^{(1)})^T\overrightarrow{X} \leqslant -b_p^{(1)},\\
&\overrightarrow{X} \geqslant 0,
\end{aligned}$$

дающую решение $\overrightarrow{X}^{(2)}$ и т. д.

На каждой итерации добавляется новая отсекающая плоскость, в результате чего образуется сжимающаяся последовательность вложенных друг в друга многогранников, охватывающих нелинейную выпуклую область планов D:

$$M^{(0)} \supset M^{(1)} \supset \cdots \supset M^{(k)} \cdots \supset D.$$

Решения линейных задач при этом образуют последовательность приближений, сходящуюся к точке минимума:

$$\overrightarrow{X}^{(k)} \to \overrightarrow{X}^*.$$

В отличие от метода Гомори, метод Кэлли не гарантирует сходимости за конечное число итераций, при этом на каждой итерации точка приближенного решения остается з а п р е д е л а м и области допустимых значений.

Для того чтобы определить, какую конкретно отсекающую плоскость следует построить на k-й итерации, рекомендуется подставить текущее приближение $\overrightarrow{X}^{(k)}$ во все ограничения и найти то из них, которое нарушается в наибольшей степени. Ему соответствует индекс

$$p = \arg\max_i \operatorname{cut}[g_i(\overrightarrow{X}^{(k)}],$$

где $\operatorname{cut}[x]$ — функция срезки (с. 158). Если при этом текущее приближение оказывается достаточно близко к допустимой области,

о чем свидетельствует выполнение соотношения $g_p(\overrightarrow{X}^{(k)}) < \varepsilon$, где ε — предустановленный параметр точности, то работа алгоритма завершается.

Замечание. Если $g_i(\overrightarrow{X})$ – линейная функция, то дополнительно линеаризовать ее не нужно, на любой итерации линеаризованная функция совпадает с исходной: $G_i^{(k)}(\overrightarrow{X}) = g_i(\overrightarrow{X})$. Действительно, пусть $g_i(\overrightarrow{X}) = \overrightarrow{a}^T\overrightarrow{X} + b$. Тогда $\overrightarrow{\nabla} g_i(\overrightarrow{X}) = \overrightarrow{a}$ и

$$G_i^{(k)}(\overrightarrow{X}) = g_i(\overrightarrow{X}^{(k)}) + \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(k)})(\overrightarrow{X} - \overrightarrow{X}^{(k)}) = \\ = \overrightarrow{a}^T\overrightarrow{X}^{(k)} + b + \overrightarrow{a}^T(\overrightarrow{X} - \overrightarrow{X}^{(k)}) = \overrightarrow{a}^T\overrightarrow{X} + b = g_i(\overrightarrow{X}).$$

Практический алгоритм

П о д г о т о в и т е л ь н ы й э т а п

Если целевая функция задачи нелинейна, производится ее линеаризация, для чего вводится дополнительная переменная и дополнительное ограничение (далее функции ограничений g_i уже новые).

Н а ч а л ь н а я з а д а ч а

Шаг 1. Выбирается начальная точка $\overrightarrow{X}^{(0)}$, например $\overrightarrow{X}^{(0)} = 0$.

Шаг 2. Строится задача линейного программирования

$$L(\overrightarrow{X}) = \overrightarrow{c}^T\overrightarrow{X} \to \min,$$
$$\left(\overrightarrow{a}_i^{(0)}\right)^T\overrightarrow{X} \leqslant -b_i^{(0)},\ i = 1, \ldots, m,$$
$$\overrightarrow{X} \geqslant 0,$$

в которой матрица условий формируется из строк

$$\left(\overrightarrow{a}_i^{(0)}\right)^T = \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)}),$$

а вектор правых частей набирается из компонент

$$-b_i^{(0)} = -g_i(\overrightarrow{X}^{(0)}) + \overrightarrow{\nabla}^T g_i(\overrightarrow{X}^{(0)})\overrightarrow{X}^{(0)}.$$

Итерация $(k = 1, 2, \dots)$

Шаг 3. Любым, например симплексным, методом решается построенная задача линейного программирования, ее оптимальный план становится текущим приближением $\overrightarrow{X}^{(k)}$.

Шаг 4. Определяется индекс самого нарушаемого ограничения

$$p = \arg\max_i \operatorname{cut}[g_i(\overrightarrow{X}^{(k)}].$$

Если $\delta^{(k)} = g_p(\overrightarrow{X}^{(k)}) < \varepsilon$, то работа алгоритма завершается.

Шаг 5. Достраивается задача линейного программирования, для чего к матрице условий добавляется строка $\left(\overrightarrow{a}_p\right)^T$, а к вектору правых частей дописывается компонента b_p, где

$$\overrightarrow{a}_p = \nabla g_p(\overrightarrow{X}^{(k)}),$$
$$b_p = -g_p(\overrightarrow{X}^{(k)}) + \overrightarrow{\nabla}^T g_p(\overrightarrow{X}^{(k)})\overrightarrow{X}^{(k)}.$$

Счетчик итераций увеличивается на единицу. Возврат на начало итерации (шаг 3).

П р и м е р. Продемонстрируем работу метода отсечений на хорошо знакомой задаче (см. примеры на с. 56, 196):

$$f(\overrightarrow{X}) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min,$$
$$g_1(\overrightarrow{X}) = x_1^2 + x_2^2 - 4 \leqslant 0,$$
$$g_2(\overrightarrow{X}) = -x_1 + x_2 \leqslant 0,$$
$$g_3(\overrightarrow{X}) = x_2 - 1 \leqslant 0,$$
$$x_1, x_2 \geqslant 0.$$

П о д г о т о в и т е л ь н ы й э т а п

Введя дополнительную переменную x_3 и дополнительную ограничивающую функцию $\tilde{g}_4$, сформулируем новую задачу с линейной целевой функцией и нелинейными ограничениями. Функции $\tilde{g}_1, \ldots, \tilde{g}_4$ теперь зависят от т р е х переменных (для упрощения обозначений знаки тильды далее опускаем):

$$
\begin{aligned}
&x_3 \to \min;\\
&g_1(x_1, x_2, x_3) = x_1^2 + x_2^2 - 4 \leqslant 0,\\
&g_2(x_1, x_2, x_3) = -x_1 + x_2 \leqslant 0,\\
&g_3(x_1, x_2, x_3) = x_2 - 1 \leqslant 0,\\
&g_4(x_1, x_2, x_3) = (x_1 - 2)^2 + (x_2 - 2)^2 - x_3 \leqslant 0,\\
&x_1, x_2, x_3 \geqslant 0.
\end{aligned}
$$

Н а ч а л ь н а я з а д а ч а

Ш а г 1. Выберем исходную точку $\overrightarrow{X}^{(0)} = \overrightarrow{0}$.

Ш а г 2. Строим начальную задачу линейного программирования. Поскольку функции g_2 и g_3 линейные, то потребуется знание градиента только для g_1 и g_4:

$$
\overrightarrow{\nabla} g_1(\overrightarrow{X}) = \begin{pmatrix} 2x_1 \\ 2x_2 \\ 0 \end{pmatrix}, \quad \overrightarrow{\nabla} g_4(\overrightarrow{X}) = \begin{pmatrix} 2x_1 - 4 \\ 2x_2 - 4 \\ -1 \end{pmatrix}.
$$

Для исходной точки вычисляем

$$
\begin{aligned}
&g_1(\overrightarrow{0}) = -4, \quad \left[\overrightarrow{a}_1^{(0)}\right]^T = \overrightarrow{\nabla}^T g_1(0,0,0) = (0,0,0),\ b_1^{(0)} = 4,\\
&g_4(\overrightarrow{0}) = 8, \qquad \left[\overrightarrow{a}_4^{(0)}\right]^T = \overrightarrow{\nabla}^T g_4(\overrightarrow{0}) = (-4,-4,-1),\ b_4^{(0)} = -8.
\end{aligned}
$$

Таким образом, задача линейного программирования для нахождения первого приближения имеет вид

$$\begin{array}{rrrcr} & & x_3 & \to \min & \\ 0x_1 & 0x_2 & 0x_3 & \leqslant & 4, \\ -x_1 & +x_2 & & \leqslant & 0, \\ & x_2 & & \leqslant & 1, \\ -4x_1 & -4x_2 & -x_3 & \leqslant & -8, \\ \multicolumn{5}{c}{x_1, x_2, x_3 \geqslant 0.} \end{array}$$

Первое ограничение получилось тривиальным, но оно не нарушает корректности постановки задачи и не должно нас смущать.

И т е р а ц и я 1

Ш а г 3. Решаем данную задачу симплексным методом, получаем оптимальный план $\overrightarrow{X}^{(1)} = (68.8506,\ 0.9951,\ 0)^T$, он выходит далеко за допустимую область.

Ш а г 4. Подставляя $\overrightarrow{X}^{(1)}$ в каждое из ограничений, получаем четыре числа: $(4737.4,\ -67.9,\ 0,\ 4470.0)$. Нарушаются первое и четвертое ограничения, причем первое нарушается сильнее, поэтому индекс $p = 1$, $\delta^{(1)} = 4737.4$. На основе этого ограничения строим новое отсечение. Вычисляем

$$\overrightarrow{a}_1 = \nabla g_p(\overrightarrow{X}^{(1)}) = (137.7,\ 1.990,\ 0)^T,$$
$$b_1 = -g_1(\overrightarrow{X}^{(1)}) + \overrightarrow{\nabla}^T g_1(\overrightarrow{X}^{(1)})\overrightarrow{X}^{(1)} = 4745.4.$$

Ш а г 5. Добавляя новую строку, достраиваем задачу линейного программирования:

$$\begin{array}{rrrcr} & & x_3 & \to \min & \\ 0x_1 & 0x_2 & 0x_3 & \leqslant & 4, \\ -x_1 & +x_2 & & \leqslant & 0, \\ & x_2 & & \leqslant & 1, \\ -4x_1 & -4x_2 & -x_3 & \leqslant & -8, \\ 137.7x_1 & 1.99x_2 & & \leqslant & 4745.4, \\ \multicolumn{5}{c}{x_1, x_2, x_3 \geqslant 0.} \end{array}$$

И т е р а ц и я 2

Ш а г 3. Решаем новую задачу симплексным методом, получаем оптимальный план $\overrightarrow{X}^{(1)} = (33.6886,\ 0.5549,\ 0.0000)^T$.

Ш а г 4. $\overrightarrow{X}^{(1)}$ также выходит за допустимую область, но уже меньше, так как $\delta^{(2)} = 1131.2 < \delta^{(1)} = 4737.4$.

Ш а г 5. Достраиваем задачу линейного программирования, переходим к новой итерации и т. д.

В табл. 12.3 приведены первые 10 итераций метода секущих плоскостей. Хорошо видно, что оптимальная точка все время остается за пределами допустимой области, хотя монотонно приближается к ней. ▲

Таблица 12.3. Итерации метода секущих плоскостей для примера

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$\delta^{(k)}$
1	68.850	0.995	0.000	4737.400
2	33.688	0.554	0.000	1131.228
3	15.034	0.044	0.000	222.034
4	3.881	0.775	0.000	11.666
5	1.008	0.998	4.000	1.986
6	2.195	0.997	1.000	1.815
7	1.780	1.000	0.450	0.597
8	1.780	1.000	1.047	0.171
9	1.732	1.000	1.069	0.002
10	1.732	1.000	1.071	0.002

12.7. Исторические замечания

Численные методы нелинейной оптимизации с ограничениями в силу их вычислительной трудоемкости стали активно развиваться только в середине XX века после создания электронных вычислительных машин, хотя некоторые фундаментальные идеи были высказаны заблаговременно при решении прикладных физических и технических задач. В частности, идея штрафных функций была предложена в статье[5], опубликованной еще в 1943 г. Ее автором был выдающийся математик XX века **Рихард Курант** (Courant, Richard; 1888–1972). Курант повторил судьбу многих европейских ученых еврейского происхождения. С 1920 по 1933 г. он был профессором Гёттингенского университета в Германии, после прихода нацистов к власти и разгрома Математического института Курант эмигрировал в США. С 1936 г. работал профессором Нью-Йоркского университета, где ему было поручено создание специального математического института (с 1964 г. называется Курантовский институт математических наук).

Рихард Курант

Расцвет численных методов нелинейного программирования пришелся на 1960–1970-е годы, в эпоху общего бума исследования операций. Активные исследования велись параллельно во многих научных коллективах, в мировой математической литературе появились тысячи публикаций на эту тему, некоторые впоследствии были признаны классическими. Хотя национальная принадлежность их авторов была весьма широкой, наиболее существенные результаты были получены в США и Великобритании, где на базе ведущих университетов и крупнейших корпораций развились научные коллективы, возникшие во време-

[5] *Courant R.* Variational methods for the solution of problems of equilibrium and vibrations // Bull. Amer. Math. Soc. 1943. V. 49. P. 1–23.

на Второй мировой войны и активно поддерживаемые военным бюджетом в эпоху «холодной войны». Именно в это время были разработаны основные алгоритмы нелинейного программирования, созданы первые промышленные пакеты прикладных оптимизационных программ.

Метод штрафных функций наиболее полно и всесторонне был исследован **Энтони Фиакко** (Fiacco, Antony V.) и **Гарфом Мак-Кормиком** (McCormick, Garth P.; 1935–2008) в Research Analysis Corporation. Эта компания выросла из спонсируемого Министерством обороны США исследовательского центра знаменитого Университета Джона Гопкинса (Johns Hopkins University), расположенного в Балтиморе, штат Мэриленд. Их монография [26], вышедшая в 1968 г., вошла в классику исследования операций и получила соответствующую премию.

Метод проектирования градиента был одной из первых удачных попыток приспособить хорошо известные градиентные методы к ситуации с ограничениями. Он предложен[6] в 1960 г. американским математиком **Розеном** (Rosen, J. Benjamin; 1923–2009), работавшим в это время в качестве руководителя Департамента прикладной математики знаменитой нефтяной компании Shell (хороший пример того, как крупные американские компании поддерживают научные исследования). В дальнейшем появилось множество схожих методов, что дало возможность голландскому математику **Гусу Зойтендейку** (Zoutendijk, Guus) назвать их общим именем «методы возможных направлений» [14].

[6] *Rosen J. B.* The gradient projection method for nonlinear programming // J. Soc. Indust. Appl. Mathem. 1960. V. 8. P. 181-217

Глава 13

Практическая оптимизация с помощью пакетов прикладных программ

На рынке программного обеспечения имеется немало специализированных пакетов прикладных программ для оптимизации как коммерческих, так и свободно распространяемых. Многие прошли длительный путь эволюции и заслужили славу мощных и надежных инструментов.

Например, в 1975 г. было объявлено о начале работ над системой GAMS — General Algebraic Modeling System [29] для решения широкого круга задач линейного, нелинейного и дискретного программирования. Эти работы были поддержаны Международным банком реконструкции и развития, для консолидации усилий математиков и программистов была создана GAMS Development Corporation. С тех пор, непрерывно совершенствуясь, GAMS превратилась в высокоуровневую программную систему, реализованную на различных аппаратных платформах — от ноутбуков до суперкомпьютеров. Для формального описания задач линейного и нелинейного программирования в системе имеется специальный входной язык, который мы рассмотрим далее.

Другим примером является очень популярная и стремительно

развивающаяся система компьютерной математики MATLAB [1] (сокращение от MATrix LABoratory). Она начала свою жизнь в конце 1970-х годов в Университете Нью-Мексико. Вначале система задумывалась как программа для доступа студентов, не владеющих Фортраном, к библиотеке программ линейной алгебры и линейного программирования LINPACK. Впоследствии функции системы постоянно расширялись, к началу XXI века она превратилась в уникальную по богатству возможностей программную среду, широко используемую академическим и научным сообществом, число ее легальных пользователей превышает миллион[1].

Кроме коммерческих существует множество свободно распространяемых программ, а также программ, к которым возможен доступ через интернет. В данной главе мы в общих чертах рассмотрим три варианта решения оптимизационных задач с помощью персонального компьютера. Первый основан на использовании надстройки `Solver` в электронных таблицах Excel, второй иллюстрирует возможность использования пакета расширений `Optimization Toolbox` для системы MATLAB, а третий предполагает обращение к специализированным интернет-ресурсам.

13.1. Оптимизация в пакете Excel

Для многих миллионов пользователей персональных компьютеров электронная таблица Excel, входящая в пакет прикладных программ Microsoft Office, является единственным и незаменимым средством автоматизации вычислений. С помощью этой программы делают все — от простых расчетов «на скорую руку» до комплексных систем бухгалтерского учета и планирования. Благодаря встроенному в Office языку программирования VBA (Visual

[1] Система поставляется американской компанией The MathWorks, Inc., расположенной в пригороде г. Бостона, США (`http://www.mathworks.com`). Подробная пользовательская документация в pdf-формате с описанием использованных алгоритмов и подробными примерами доступна для скачивания на сайте компании.

Basic for Applications) имеются неограниченные возможности разработки специализированных *надстроек — Add-ons* для решения многих, в том числе достаточно сложных задач.

Одной из надстроек, имеющихся в стандартной конфигурации Excel, является программа `Поиск решения - Solver`, предназначенная для решения разнообразных задач оптимизации.

С помощью этой надстройки можно решать задачи линейного и нелинейного программирования с ограничениями в виде равенств и неравенств, с дополнительными ограничениями на целочисленность, а также задачи с булевыми переменными. Для линейного программирования применяется симплексный метод, для нелинейного — методы первого порядка (квазиньютоновский и метод сопряженных направлений) с численной оценкой градиента (см. с. 127). Для целочисленного и булевого программирования используется метод ветвей и границ.

Описание надстройки `Solver` приведено в документации Excel и в контекстной справке, поэтому мы не будем его дублировать, вместо этого на простых примерах продемонстрируем простоту и удобство ее использования.

П р и м е р 1. Возьмем совсем несложную задачу выпуклого программирования (см. пример на с. 56), которую мы решали самыми различными методами:

$$f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min,$$
$$x_1^2 + x_2^2 \leqslant 4,$$
$$-x_1 + x_2 \leqslant 0,$$
$$x_2 \leqslant 1,$$
$$x_1, x_2 \geqslant 0.$$

На рис. 13.1 представлен соответствующий рабочий лист Excel, в котором для наглядности включен режим отображения формул, а не значений, и имеются поясняющие надписи. Под переменные x_1, x_2 выделены ячейки B3 и C3 с нулевыми начальными значе-

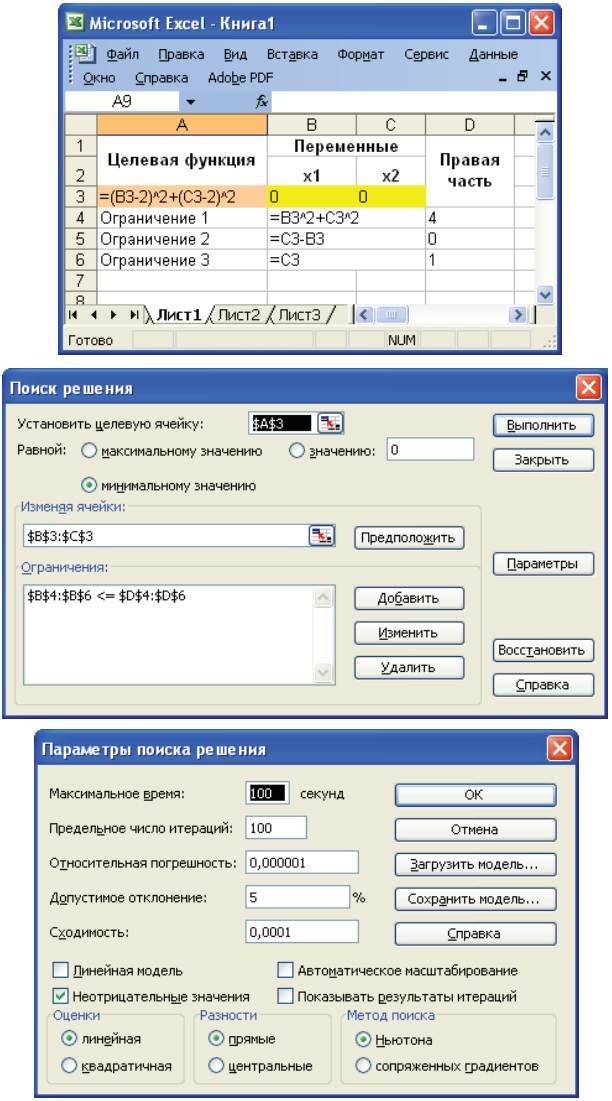


Рис. 13.1. Рабочий лист Excel и настройки Solver для задачи нелинейного программирования

ниями, формула вычисления целевой функции записана в ячейку A3, а формулы левых частей ограничений занимают диапазон ячеек B4:B6.

Для инициализации процесса оптимизации запускаем надстройку `Поиск решения (Solver)` из меню `Сервис (Tools)` и настраиваем ее на рабочий лист. Для этого указываем: 1) целевую ячейку, 2) диапазон ячеек с переменными, 3) диапазон ячеек с формулами левых частей ограничений (если они есть) и 4) диапазон ячеек с правыми частями ограничений.

Установка параметров поиска решения производится нажатием кнопки `Параметры`. В открывшемся окне видим опцию `Линейная модель`, которую оставляем выключенной, так как наша модель нелинейная, и включаем опцию `Неотрицательные значения`. Выбираем квазиньютоновский метод оптимизации с прямыми (правыми) разностями для численного дифференцирования. Оставляем значения параметров точности по умолчанию, возвращаемся в главное окно поиска решения и нажимаем кнопку `Выполнить`. В нашем примере для нахождения решения с заданной точностью потребовалось 5 итераций.

Отчеты Solver формируются на отдельных рабочих листах (рис. 13.2). В «Отчете о результатах» распечатываются начальные и достигнутые при оптимизации значения переменных и целевой функции, а также анализируются ограничения. Как видно из отчета, для найденного оптимального решения связанными (активными) являются первое и третье ограничения, что соответствует действительности.

В «Отчете по устойчивости» приводятся полученные в ходе оптимизации оценки множителей Лагранжа. Напомним, что их истинные значения, полученные при аналитическом решении в примере на с. 56, равны соответственно (0.1547, 0, 1.6906). Обратные знаки получились потому, что в данной программе, как и в некоторых учебниках, используется форма записи функции Лагранжа с минусом перед знаком суммы. ▲

Microsoft Excel 11.0 Отчет по результатам

Целевая ячейка (Минимум)

Ячейка	Имя	Исходное значение	Результат
$A$3	Целевая функция	8	1,071796696

Изменяемые ячейки

Ячейка	Имя	Исходное значение	Результат
$B$3	x1	0,0000	1,7321
$C$3	x2	0	1

Ограничения

Ячейка	Имя	Значение	Формула	Статус	Разница
$B$4	Ограничение 1 x1	4,000000475	$B$4<=$D$4	связанное	0
$B$5	Ограничение 2 x1	-0,732050945	$B$5<=$D$5	не связан.	0,732050945
$B$6	Ограничение 3 x1	1	$B$6<=$D$6	связанное	0

Microsoft Excel 11.0 Отчет по устойчивости

Изменяемые ячейки

Ячейка	Имя	Результ. значение	Нормир. градиент
$B$3	x1	1,7321	0,0000
$C$3	x2	1	0

Ограничения

Ячейка	Имя	Результ. значение	Лагранжа Множитель
$B$4	Ограничение 1 x1	4,000000475	-0,15469986
$B$5	Ограничение 2 x1	-0,732050945	0
$B$6	Ограничение 3 x1	1	-1,69059918

Рис. 13.2. Отчет Solver для задачи нелинейного программирования

П р и м е р 2. В качестве другого примера решим простейшую транспортную задачу линейного программирования с двумя складами и тремя магазинами, подробно рассмотренную в [9, с. 133]:

$$
\begin{array}{rrrrrrrrl}
L(X) = & 2x_{11} & +4x_{12} & +2x_{13} & +3x_{21} & +x_{22} & +3x_{23} & \to \min, \\
 & x_{11} & +x_{12} & +x_{13} & & & & = 5, \\
 & & & & +x_{21} & +x_{22} & +x_{23} & = 7, \\
 & x_{11} & & & +x_{21} & & & = 2, \\
 & & x_{12} & & & +x_{22} & & = 4, \\
 & & & x_{13} & & & +x_{23} & = 6,
\end{array}
$$

$$x_{11}, \ldots, x_{23} \geqslant 0.$$

Рабочий лист Excel и настройки главного окна `Solver` для нее приведены на рис. 13.3. Ячейки для оптимизируемых переменных выделены цветом и оставлены пустыми. В параметрах поиска решения следует включить опции `Линейная модель` и `Неотрицательные значения`. При этом автоматически вызовется программа симплексного метода. Хотя с точки зрения трудоемкости стандартный симплексный метод неоптимален для транспортной задачи, так как метод потенциалов гораздо экономнее, для небольших размерностей он вполне работоспособен. Отчет по результатам мы для экономии места не приводим, найденное оптимальное решение полностью соответствует тому, которое в данной задаче было получено вручную. ▲

Оба приведенных примера говорят о том, что для несложных задач оптимизации электронная таблица Excel является вполне приемлемым практическим инструментом. Несомненным удобством является то, что `Solver` легко подключается к любой существующей электронной таблице, не нужно ни импорта, ни экспорта данных. Однако размерность решаемых задач ограничена быстродействием самой системы Excel. По своей сути она является довольно медленной, так как формулы на рабочем листе за-

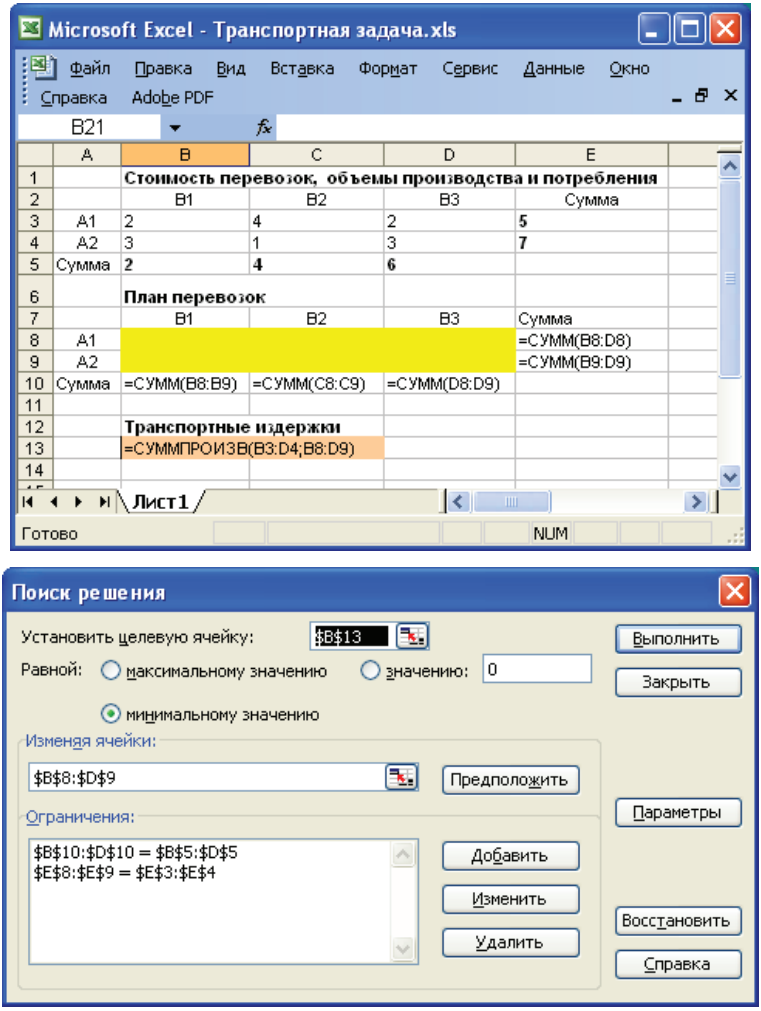


Рис. 13.3. Рабочий лист Excel и настройки Solver для транспортной задачи

ново анализируются интерпретатором системы при каждом перевычислении.

13.2. Оптимизация в пакете MATLAB

Система компьютерной математики MATLAB состоит из ядра и многочисленных *расширений (toolboxes)*, их насчитывается более 80. Существуют специализированные расширения для самых различных приложений — от символьных вычислений до обработки изображений и имитационного моделирования.

Для решения оптимизационных задач во всех версиях системы имеется `Optimization Toolbox`, в дополнение к нему, начиная с версии 7.0.1 (2004 г.), поставляется пакет расширений `Genetic Algorithm and Direct Search Toolbox`.

Optimization Toolbox

Поскольку круг оптимизационных задач весьма разнообразен и наилучшего универсального алгоритма не существует, в Optimization Toolbox имеется возможность подобрать наиболее подходящий для данной задачи *решатель (solver)* и настроить его параметры в соответствии с особенностями конкретной задачи. Основные решатели перечислены в табл. 13.1, кроме них существуют специфические решатели для подгонки кривых методом наименьших квадратов, для полубесконечной оптимизации, многоцелевой оптимизации, минимаксной оптимизации и т. д.

Каждый из решателей представляет собой достаточно сложную программу, в которой реализовано несколько алгоритмов оптимизации с различными вариантами использования. Как правило, имеются отдельные алгоритмы для задач среднего масштаба — `Medium Scale` и крупномасштабных задач — `Large Scale`. Провести четкую границу между ними нельзя, но ясно, что если промежуточные данные, например матрица Гессе, в задаче с очень большим числом переменных не помещаются в оперативной памяти компьютера, то необходимо применить такой алгоритм, который экономит память.

Таблица 13.1. Задачи оптимизации и соответствующие им решатели

Задача	Solver	Область применения
Одномерная оптимизация	`fminbnd`	Любые одномерные целевые функции
Оптимизация без ограничений	`fminunc`	Гладкие целевые функции
	`fminsearch`	Любые целевые функции
Линейное программирование	`linprog`	Линейные целевые функции с линейными ограничениями
Квадратичное программирование	`quadprog`	Квадратичные целевые функции с линейными ограничениями
Оптимизация с ограничениями	`fmincon`	Гладкие целевые функции с гладкими ограничениями
	`patternsearch`	Любые целевые функции без ограничений или с ограничениями (линейными и нелинейными)
Дискретное программирование	`bintprog`	Линейные целевые функции с булевыми ограничениями

Основные алгоритмы оптимизации, реализованные в решателях, приведены в табл. 13.2. Как видно, их набор весьма разнообразен, причем далеко не все нам знакомы. Это не удивительно, так как подобные задачи, насчитывающие многие тысячи переменных, отличаются особой сложностью, и для их решения изобретены изощренные алгоритмы, рассмотрение которых выходит далеко за пределы нашего курса.

С другой стороны, даже давно известные классические методы за те десятилетия, пока шел прогресс в области оптимизации и создавались промышленные алгоритмы, претерпели значительные изменения и продолжают совершенствоваться. Примером может служить поиск по образцу (метод конфигураций Хука — Дживса). В решателе `patternsearch`, включенном в пакет расшире-

ний `Genetic Algorithm and Direct Search Toolbox`, с 2006 г. появилась возможность учета нелинейных ограничений (ранее допускались только линейные). Это необходимо учитывать при чтении быстро стареющих пособий по системе MATLAB.

Таблица 13.2. Основные алгоритмы оптимизации в MATLAB

Solver	Масштаб	Используемые алгоритмы
`fminbnd`	-	Золотое сечение, метод парабол
`fminunc`	`medium`	Метод Ньютона и квазиньютоновские (BFGS и DFP) методы
	`large`	Метод доверительной области
`fminsearch`	любой	Метод деформируемого многогранника (Нелдера — Мида)
`lprog`	`medium`	Симплексный метод
	`large`	Метод внутренней точки в линейном программировании [9, с. 109]
`quadprog`	`medium`	Метод активного набора ограничений (метод проектирования)
	`large`	Метод доверительной области
`fmincon`	`medium`	Последовательное квадратичное программирование
	`large`	Метод доверительной области
`patternsearch`	Любой	Обобщенный алгоритм поиска по образцу
`bintprog`	Любой	Метод ветвей и границ

На перечне решателей разнообразие возможностей пакета оптимизации не кончается. Даже в пределах одного алгоритма возможны варианты задания исходных данных. Например, одному и тому же решателю `fmincon` в режиме `medium`[2], использующему квазиньютоновскую процедуру, можно передать вычисленные

[2]Названия опций и детали интерфейса время от времени меняются. Например, в версии 2008a данный режим можно применить, установив название алгоритма `Active-set`

аналитически значения первых и вторых производных целевой функции, в результате чего реализуемый метод становится классическим методом Ньютона (см. с. 114), а можно ограничиться отсчетами функции и выражениями для градиента, в этом случае он превращается по желанию пользователя либо в квазиньютоновский метод «полуторного» порядка Бройдена — Флетчера — Гольдфарба — Шанно BFGS, либо в аналогичный метод Дэвидона — Флетчера — Пауэлла DFP (с. 126). Наконец, можно поручить решателю выполнить численное дифференцирование целевой функции и сделать оценку градиента по ее отсчетам. Тогда мы получаем квази-квазиньютоноский метод «половинного» порядка (см. сноску на с. 127).

Графический интерфейс

Сформулировать задачу оптимизации и настроить алгоритм решения в `Optimization Toolbox` можно двояким образом: с помощью графического оконного интерфейса `Optimization Tool` (появился в версии 2006b) и из командной строки. Эти два способа совершенно эквивалентны по возможностям, поскольку в результате графического диалога формируется последовательность обращений к функциям, устанавливающим те или иные параметры алгоритмов, разница только в удобстве работы.

Общий вид графического окна `Optimization Tool` приведен на рис. 13.4. Как только подобран подходящий поставленной задаче решатель (в данном случае `fmincon`) и установлен алгоритм из числа реализованных в этом решателе, справа появляется окно контекстной помощи, облегчающее настройку многочисленных параметров алгоритма (в большинстве случаев достаточно принять рекомендованные предустановленные значения).

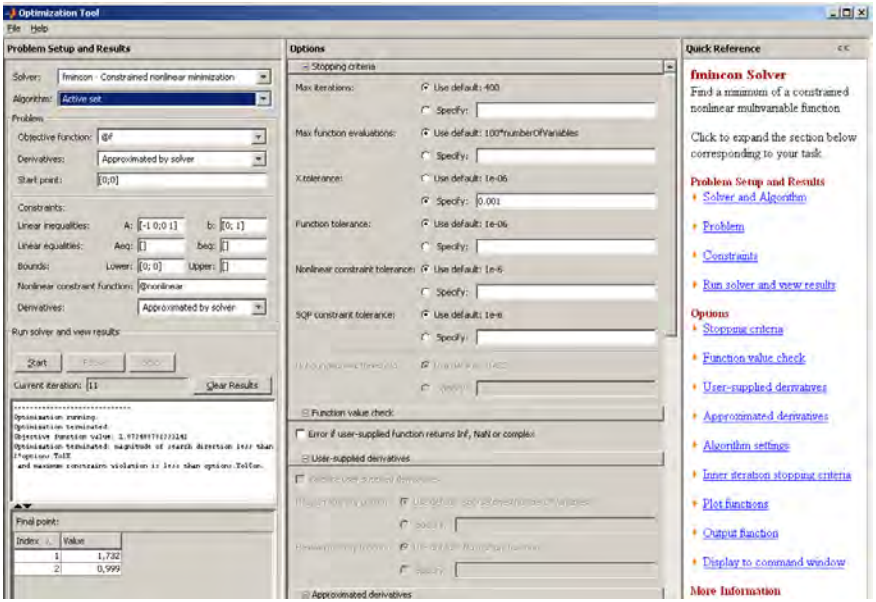


Рис. 13.4. Графический интерфейс Optimization Tool. Настройки соответствуют примеру

П р и м е р. Рассмотрим задачу нелинейного программирования, которую мы многократно решали в предыдущих примерах:

$$f(x_1, x_2) = (x_1 - 2)^2 + (x_2 - 2)^2 \to \min,$$

$$x_1^2 + x_2^2 \leqslant 4,$$

$$-x_1 + x_2 \leqslant 0,$$

$$x_2 \leqslant 1,$$

$$x_1, x_2 \geqslant 0.$$

Целевая функция квадратичная, ограничения нелинейные, гладкие. Обратившись к табл. 13.1, видим, что более всего к данной задаче подходит решатель `fmincon`, работающий в масштабе `medium`.

Как следует из инструкции к алгоритму (она есть в документации и контекстной подсказке), все ограничения задачи должны быть рассортированы на пять групп:

1) линейные ограничения-неравенства $A\overrightarrow{X} \leqslant \overrightarrow{b}$;

2) линейные ограничения-равенства $A_{eq}\overrightarrow{X} = \overrightarrow{b}_{eq}$;

3) простые ограничения $\overrightarrow{b}_l \leqslant \overrightarrow{X} \leqslant \overrightarrow{b}_u$;

4) нелинейные ограничения-неравенства $g_i(\overrightarrow{X}) \leqslant 0$;

5) нелинейные ограничения-равенства $h_i(\overrightarrow{X}) = 0$.

В нашем случае имеются линейные ограничения-неравенства, простые левосторонние ограничения на неотрицательность, а также одно нелинейное ограничение в форме неравенства:

$$A = \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}, \ \overrightarrow{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \ \overrightarrow{b}_l = \begin{pmatrix} 0 \\ 0 \end{pmatrix},$$

$$g_1(\overrightarrow{X}) = x_1^2 + x_2^2 - 4,$$

остальные ограничения отсутствуют.

Все линейные ограничения задаются непосредственно в графическом окне (см. рис. 13.4), а целевая функция и нелинейные функции ограничений (в виде равенств и неравенств) должны быть записаны на встроенном языке программирования *(М-языке)*, они подготавливаются любым редактором и хранятся как подпрограммы в виде текстовых файлов. В нашем примере М-файлы выглядят следующим образом.

Подпрограмма вычисления целевой функции:

```
function y = f(X)
y = (X(1)-2)^2 + (X(2)-2)^2;
```

Подпрограмма вычисления функций нелинейных ограничений:

```
function [g, h] = nonlinear(X);
g(1) =  X(1)^2 + X(2)^2 -4; % Одно ограничение-неравенство
h = [ ]; % Нелинейные ограничения-равенства отсутствуют
```

Исходная точка установлена в начало координат, большинство опций и параметров алгоритма оставлены стандартными, однако для сокращения числа итераций уменьшена стандартная точность вычисления минимума, для чего параметр `TolX` (сокращение от *tolerance X)*, означающий допустимое отклонение $\|\overrightarrow{X}_k - \overrightarrow{X}_{k-1}\|$, установлен равным 10^{-4} . Для вычисления производных предложено применить методы численного дифференцирования (опция `Derivatives` установлена в `Approximated by solver`). Кроме того, включена печать диагностической информации и промежуточных итераций.

Запустив решатель кнопкой `Start`, после 12 итераций получаем результат $\overrightarrow{X}^* = (1.732,\ 1)^T$ и протокол работы по итерациям (рис. 13.5). ▲

Командный интерфейс

Работа через графический интерфейс, разумеется, более приятна для неподготовленного пользователя, однако при решении практических задач удобнее пользоваться расширенными возможностями системы MATLAB, запуская из командной строки на интерпретацию заранее заготовленную главную программу, которая в нашем примере имеет следующий вид.

```
% Задание линейных ограничений
 A = [-1 0
       0 1];
 b = [0; 1];
Aeq = [ ];    % Ограничения-равенства отсутствуют
beq = [ ];    % Ограничения-равенства отсутствуют
 lb = [0; 0]; % Левые простые ограничения
                на неотрицательность
 ub = [ ];    % Правые простые ограничения отсутствуют

% Задание исходной точки для алгоритма оптимизации
X0 = [0; 0];

% (См. продолжение на с. 226)
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

   Diagnostic Information

Number of variables: 2

Functions
 Objective:                            f
 Gradient:                             finite-differencing
 Hessian:                              finite-differencing (or Quasi-Newton)
 Nonlinear constraints:                nonlinear
 Gradient of nonlinear constraints:    finite-differencing

Constraints
 Number of nonlinear inequality constraints: 1
 Number of nonlinear equality constraints:   0

 Number of linear inequality constraints:    2
 Number of linear equality constraints:      0
 Number of lower bound constraints:          2
 Number of upper bound constraints:          0

Algorithm selected
   medium-scale: SQP, Quasi-Newton, line-search


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 End diagnostic information


                                 Max     Line search  Directional  First-order
 Iter F-count        f(x)   constraint   steplength   derivative   optimality Procedure
    0       3           8            0
    1       8      4.0625        -0.25         0.25        -11.5         2.25
    2      12     2.17185       -0.375          0.5        -3.06         1.06
    3      16     1.54229      -0.1875          0.5        -1.13        0.359
    4      20     1.29174     -0.09375          0.5       -0.478        0.167
    5      24     1.17826     -0.04688          0.5       -0.222       0.0935
    6      28     1.12418     -0.02344          0.5       -0.107       0.0504
    7      32     1.09778     -0.01172          0.5      -0.0525       0.0261
    8      36     1.08474    -0.005859          0.5       -0.026       0.0133
    9      40     1.07825     -0.00293          0.5      -0.0129      0.00671  Hessian modified
   10      44     1.07502    -0.001465          0.5     -0.00646      0.00337  Hessian modified
   11      48     1.07341   -0.0007324          0.5     -0.00323      0.00169  Hessian modified
   12      52      1.0726   -0.0003662          0.5     -0.00161     0.000845  Hessian modified

Optimization terminated: first-order optimality measure
less than options.

TolFun and maximum constraint violation is less than
options.

TolCon. No active inequalities.
```

Рис. 13.5. Протокол работы решателя `fmincon` для примера. Для каждой итерации приведены: Iter — порядковый номер; f(x) — значение целевой функции; F-count — накопленное число измерений целевой функции; Max constraint — наибольшее нарушение ограничений; Directional derivative — производная по направлению; First-order optimality — мера оптимальности первого порядка (см. с. 96).

```
% (Продолжение со с. 224)
% Установка нестандартных опций и значений параметров
options = optimset('LargeScale', 'off', ...
          'Display','iter', 'TolX', 0.001);

% Запуск решателя fmincon
[X_opt,f_opt] = fmincon(@f, X0, A, b, Aeq, beq, lb, ub, ...
           @nonlinear, options)
```

Эта программа полностью соответствует описанному выше сеансу работы через графический интерфейс. Перед запуском решателя с помощью функции `optimset` изменены предустановленные значения некоторых опций и параметров. Отключена опция `LargeScale,` которая вызывает алгоритм для решения крупномасштабных задач и действует по умолчанию.

Более подробно познакомиться со всеми нюансами работы с пакетом `Optimization Toolbox`, примерами его применения и описаниями алгоритмов можно по фирменному документу Optimization Toolbox User's Guide.

Генетические алгоритмы

Пакет расширений `Genetic Algorithm and Direct Search Toolbox` включает в себя решатель `patternsearch`, который был упомянут выше, и решатель `ga - Genetic Algorithm`. Так же как и в `Optimization Toolbox`, работа в данном пакете возможна двумя способами — через графический интерфейс и в программном режиме. Так как в алгоритмах имеется очень большое число опций и настроечных параметров, использовать программный режим целесообразно только после накопления достаточного опыта.

Генетические алгоритмы в MATLAB очень быстро совершенствуются, поэтому в новых версиях системы имеется ряд дополнительных функций. В частности, в версии 2008 г. по сравнению с версией 2004 г. появилась принципиально новая возможность уче-

та ограничений, кроме того, прежний специализированный графический интерфейс `Genetic Algorithm Tool` генетических алгоритмов был унифицирован с другими алгоритмами оптимизации и включен в `Optimization Tool`.

П р и м е р. На рис. 13.6 представлен вид графического окна `Optimization Tool`, настроенного на решение задачи поиска минимума функции Растригина, рассмотренной в примере на с. 143. Сравнивая его с окном на рис. 13.4, видим, что основные настройки унифицированы, отличаются только параметры, специфические для генетических алгоритмов. ▲

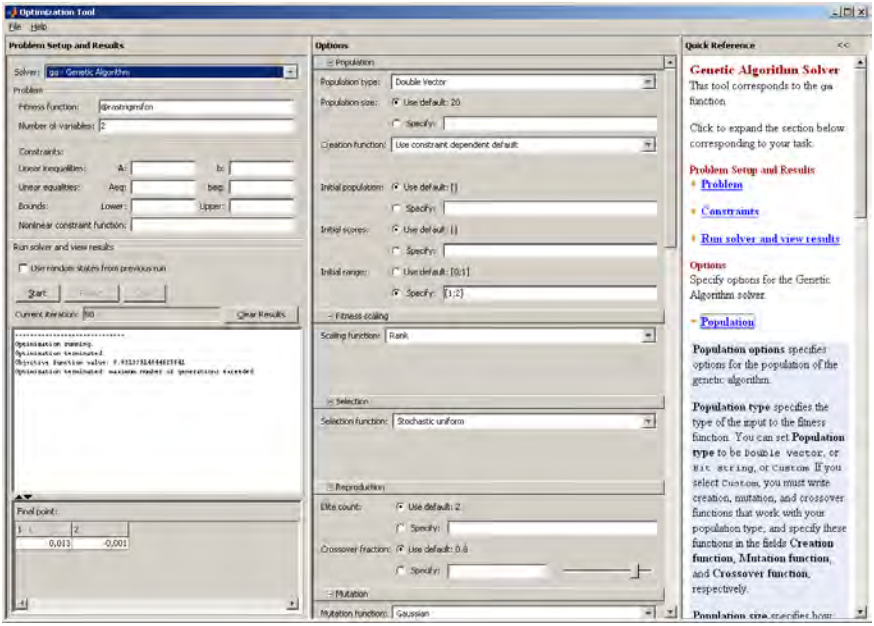


Рис. 13.6. Графический интерфейс генетического алгоритма. Задача минимизации функции Растригина без ограничений

13.3. Оптимизация в интернете

Бурное развитие компьютерных сетей привело к тому, что многие пользователи предпочитают не устанавливать на своих компьютерах узкоспециализированное программное обеспечение, а пользоваться сетевыми вычислительными ресурсами. Такой подход представляется особенно привлекательным, если соответствующие задачи возникают от случая к случаю, так что приобретение и обслуживание подходящих пакетов прикладных программ становится нерентабельным. Другая возможная причина выхода в сеть — чрезвычайно большая сложность или размерность задачи, требующая такой вычислительной мощности, которая недоступна персональному компьютеру.

Во Всемирной паутине имеется немало публичных серверов, специально предназначенных для решения оптимизационных задач. Как правило, они функционируют на базе крупных суперкомпьютерных центров и используют мощные и проверенные практикой решатели задач линейного и нелинейного программирования: CPLEX, MINOS, IPOPT, SNOPT и т. д.

Взаимодействие пользователя с решателем может происходить по-разному:

- из программ на стандартных языках программирования (Fortran, C, Java и др.) путем удаленного вызова подпрограмм;
- через графический интерфейс (GUI) — специализированную клиентскую программу, которая устанавливается у пользователя;
- через стандартный web-интерфейс или по электронной почте путем передачи условий задачи, записанной на входном языке — *языке алгебраического моделирования (Algebraic Modeling Language — AML)*;

Последний способ является наиболее удобным, так как не зависит

от аппаратно-программной платформы, на которой работает сервер. Хотя многие решатели имеют собственные входные языки, в этой области намечается определенная стандартизация. Наиболее распространенными языками описания задач математического программирования являются GAMS и AMPL.

Язык GAMS Язык GAMS был исторически первым языком алгебраического моделирования и разрабатывался для одноименной системы, о которой говорилось в начале настоящей главы. Изначально он предназначался для описания экономических и управленческих задач, и это отразилось на его синтаксисе, избегающем использования математической символики. Впоследствии границы использования языка расширились, он стал использоваться как входной язык во многих решателях сторонних производителей.

Общее представление о выразительных возможностях языка GAMS можно получить, рассмотрев простой пример, для более подробного знакомства следует обратиться к фирменным руководствам.

П р и м е р. Возьмем простейшую транспортную задачу с двумя складами и тремя магазинами, которую мы решали с помощью электронной таблицы Excel (см. с. 216). Ее описание на языке GAMS имеет следующий вид.

```
Sets
    i "склады" /Store_1, Store_2 /
    j "магазины" /Shop_1, Shop_2, Shop_3 / ;

Parameters a(i) "запас на складе i, шт."
             / Store_1 5
               Store_2 7 /
           b(j) "потребности магазина j, шт."
             / Shop_1 2
               Shop_2 4
               Shop_3 6 / ;
```

```
Table d(i,j) "расстояние в кварталах"
             Shop_1  Shop_2  Shop_3
    Store_1       2       4       2
    Store_2       3       1       3 ;

Scalar f "цена перевозки 1 шт. на расстояние квартала" /1/;

Parameter c(i,j) "матрица стоимости перевозок, у.е. за 1 шт.";
            c(i,j) = f*d(i,j) ;
Variables
         x(i,j) "величина перевозки, шт."
         L "транспортные расходы, у.е.";
Positive variable x ;

Equations
    cost        "целевая функция"
    supply(i)   "наличие на складе i"
    demand(j)   "потребность магазина j" ;
cost ..         L =e= sum((i,j), c(i,j)*x(i,j)) ;
supply(i) .. sum(j,x(i,j)) =e= a(i) ;
demand(j) .. sum(i,x(i,j)) =e= b(j) ;
Model transport /all/ ;
```

В данном примере слова на русском языке в кавычках являются комментариями, они без изменений копируются в результаты. Однако на практике их лучше избегать либо пользоваться латиницей, так как некоторые компиляторы не воспринимают кириллицу. Индексы записываются в круглых скобках, для задания множества используются не фигурные скобки $\{a, b, c\}$, а две наклонные черты $/a, b, c/$, так как авторы языка посчитали, что не на всех клавиатурах есть соответствующие символы. Сочетание знаков «=e=» означает отношение равенства (equal), в отличие от оператора присваивания «=».

Оператор `Sets` объявляет множества индексов и присваивает их элементам содержательные текстовые значения, в нашем случае складам (Store_1, Store_2) и магазинам (Shop_1, Shop_2, Shop_3). Параметры (данные) модели объявляются и получают

значения с помощью операторов `Scalar` (единственное значение), `Parameter(s)` (список значений) и `Table` (таблица), для объявления переменных модели служит оператор `Variables`. Уравнения модели задаются оператором `Equations`, в нашем примере они линейные.

Описав модель задачи, можно вызвать необходимый решатель (в нашем случае решатель линейных задач `lp`), для этого достаточно написать предложение `Solve transport using lp minimizing L.` ▲

Язык GAMS обладает достаточно мощными изобразительными возможностями, он позволяет описывать широкий класс линейных и нелинейных, в том числе дискретных, моделей и достаточно широко используется на практике[3]. Однако в нем нет четкого разделения программы на описание модели и задание данных, что является серьезным недостатком.

Язык AMPL Язык *AMPL (A Mathematical Programming Language)* был разработан в 1990-х годах в Bell Laboratories для того, чтобы описывать и решать сложные задачи оптимизации и теории расписаний. По назначению и структуре очень близок к GAMS, но он создавался позже, его синтаксис ближе к стандартному языку математики, что позволяет дать очень краткое и легкочитаемое определение задач математического программирования. Еще одним отличием AMPL является то, что описание модели в нем полностью отделено от ввода данных, это придает программе логическую стройность и гибкость[4].

[3] Самая свежая документация по языку и системе, учебный курс, а также дистрибутивы для различных аппаратно-программных платформ доступны на сайте GAMS Development Corporation по адресу `http://www.gams.com`. Без лицензии программы работают в демо-режиме, ограничивающем число переменных и ограничений.

[4] Полная информация о языке AMPL, условия лицензирования, пробная и студенческая версии программ доступны на фирменном сайте `http://www.ampl.com/`.

П р и м е р. Переведем задачу из предыдущего примера на язык AMPL. Названия и назначения операторов почти полностью соответствуют GAMS и понятны без дополнительных пояснений, ключевое слово *subject to* хорошо знакомо математикам, оно переводится *при условии*. Объявления модели и присваивания данных разнесены, они даже могут находиться в разных файлах. Комментарии на русском языке оставлены для сопоставления с программой на языке GAMS, однако при практическом использовании языка комментарии следует писать на латинице.

```
##### МОДЕЛЬ #####

### Множества индексов ###
set Stores "склады" ;
set Shops "магазины";

### Параметры ###
param a "запас на складе i, шт." {i in Stores};
param b "потребности магазина j, шт." {j in Shops};
param d  "расстояние в кварталах" {i in Stores, j in Shops};
param c "матрица стоимости перевозок, у.е. за 1 шт."
        {i in Stores, j in Shops} = f * d[i,j];
param f "цена перевозки 1 шт. на расстояние квартала";

### Переменные ###
var x "план перевозок" {Stores,Shops}>=0;

### Целевая функция ###
minimize L: sum {i in Stores, j in Shops} c[i,j] * x[i,j];

### Ограничения ###
subject to {i in Stores}: sum {j in Shops} x[i,j] = a[i];
subject to {j in Shops}: sum {i in Stores} x[i,j] = b[j];

##### ДАННЫЕ #####
data;
set Stores := Store_1 Store_2;
set Shops := Shop_1 Shop_2 Shop_3;
```

```
param a :=
    Store_1  5
    Store_2  7;

param b :=
    Shop_1  2
    Shop_2  4
    Shop_3  6;

param d:          Shop_1 Shop_2 Shop_3 :=
          Store_1    2      4       2
          Store_2    3      1       3;

param f := 1;
end;
```

Так же как и в системе GAMS, компилятор AMPL не решает задачу сам, а передает ее внешнему решателю. Если запустить приведенную выше модель на фирменном сервере AMPL, попросив подключить широко известный решатель MINOS, будет выдан следующий результат.

```
MINOS 5.51: optimal solution found.
2 iterations, objective 23
:          _varname          _var    :=
1   "x['Store_1','Shop_1']"   0
2   "x['Store_1','Shop_2']"   0
3   "x['Store_1','Shop_3']"   5
4   "x['Store_2','Shop_1']"   2
5   "x['Store_2','Shop_2']"   4
6   "x['Store_2','Shop_3']"   1
```

▲

Сетевые решатели

Доступ к сетевым серверам, обеспечивающим работу решателей, возможен с различных сайтов, как коммерческих, так и свободных. В частности, упомянутые выше фирменные сайты GAMS и AMPL предоставляют право бесплатно решать задачи небольшого размера в демонстрационном режиме.

Наиболее продвинутой сетевой оптимизационной системой является Network-Enabled Optimization System (NEOS), поддерживаемая отделением математики и компьютерных наук (MCS) знаменитой Argonne National Laboratory (ANL), о которой мы упоминали в исторических замечаниях к гл. 11 (см. с. 148). Имеющиеся в системе решатели отражают самое современное состояние науки в области оптимизации, и хотя некоторые из них являются коммерческими продуктами, онлайновый доступ к ним предоставляется неограниченным и бесплатным.

Зайдя на сайт этого проекта[5], организованный наподобие Wikipedia, можно увидеть разнообразную информацию об используемых математических алгоритмах, тестовых задачах, примерах практического использования (case studies) и т. п., а также передать исходные данные вычислительному серверу `NEOS server`, к которому подключено несколько десятков различных решателей. Пользуясь имеющимися рекомендациями, пользователь может подобрать наиболее подходящий. Далее выбирается способ взаимодействия с решателем, поскольку решатель может иметь несколько входных интерфейсов, в том числе на языках GAMS и AMPL.

Сами вычисления происходят в пакетном режиме. Сформированное задание отправляется на один из вычислительных серверов лаборатории и ставится в очередь. Пользователю сообщается идентификационный номер и пароль, по ним можно получить справку о состоянии задания (сложные задачи могут считаться долго). Результаты счета можно посмотреть непосредственно на сайте либо получить по электронной почте.

[5] `http://wiki.mcs.anl.gov/NEOS/index.php/NEOS_Wiki`

Глава 14

Динамическое программирование

В отличие от всех предыдущих методов оптимизации, динамическое программирование, созданное выдающимся американским математиком Ричардом Беллманом (см. исторические замечания в конце главы), не описывается конкретным алгоритмом, его правильнее рассматривать как специфический подход к решению задач определенного класса — многошаговых процедур принятия решений.

Излагать этот раздел исследования операций можно поразному. Некоторые авторы идут от общего к частному, формулируя общую проблему многошаговой оптимизации в пространстве состояний и конкретизируя ее в конкретных ситуациях. Мы поступим по-другому: на нескольких элементарных примерах попытаемся продемонстрировать основные принципы динамического программирования. Внешне они очень просты, однако эта простота обманчива, так как за кажущейся очевидностью скрыт глубокий смысл.

14.1. Задача о кратчайшем пути

П р и м е р. Шесть населенных пунктов, обозначенных буквами a, b, c, d, e, f, связаны сетью дорог, представленных графом на рис. 14.1. Цифры показывают расстояние между соседними пунктами (вершинами графа), движение одностороннее. Требуется найти кратчайший путь из a в f. ▲

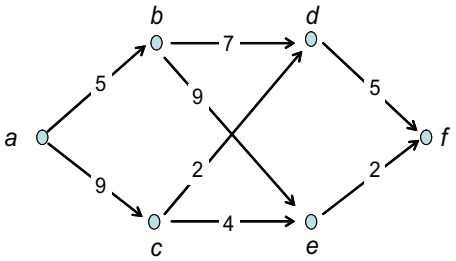


Рис. 14.1. Пример задачи о кратчайшем пути

Демонстрация основных принципов

Поставленная задача определенно вызовет снисходительную улыбку у читателя, даже не знакомого с теорией графов. Здесь имеется всего четыре варианта, их легко перебрать и получить кратчайший путь $a \to c \to e \to f$. Но не торопитесь с выводами. На этом примитивном примере мы рассмотрим фундаментальные идеи динамического программирования и сделаем далеко идущие обобщения.

Базовых принципов динамического программирования три:

- принцип многошаговости,
- принцип погружения,
- принцип оптимальности.

Принцип многошаговости *состоит в том, что исходная многомерная задача развертывается во времени и представляется как целенаправленный многошаговый процесс движения в пространстве состояний (отсюда название динамического программирования), причем на каждом шаге решается простейшая одномерная задача.*

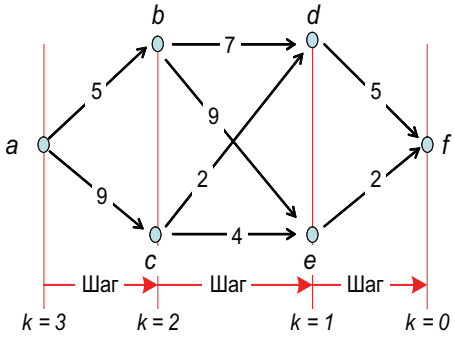


Рис. 14.2. Превращение задачи в многошаговую

В нашем примере превращение переборной задачи в многошаговую происходит естественным образом. Состояния соответствуют вершинам графа. Если образовать четыре условных рубежа, показанных на рис. 14.2, то шагом можно считать перемещение в соседнюю вершину следующего рубежа. Таким образом, наша задача теперь заключается в том, чтобы за три шага дойти из вершины a до цели, причем на каждом шаге принимается простейшее решение — свернуть налево или направо. По причине, которая станет ясной в следующем абзаце, нумерацию рубежей индексом k удобно вести с конца, от цели пути.

Принцип погружения *состоит в том, что исходная задача с конкретным числом шагов n и конкретным начальным состоянием погружается в класс задач с числом шагов $k = 0, 1, \ldots, n$ и всевозможными допустимыми начальными состояниями.*

Этот принцип способен вызвать недоумение: какой смысл со-

знательно усложнять поставленную задачу? Однако в нем заключается вся хитрость подхода, предложенного Беллманом. Весь класс задач будет решаться рекуррентно, с конца, то есть с $k = 0$, при этом при увеличении k будут использоваться результаты предыдущего решения.

В нашем случае число шагов k меняется от 0 до 3, и весь класс задач выглядит следующим образом.

- $k = 0$: за 0 шагов дойти до цели из вершины f (тривиальная задача);
- $k = 1$: за 1 шаг дойти кратчайшим путем до цели из вершин d или e;
- $k = 2$: за 2 шага дойти кратчайшим путем до цели из вершин b или c;
- $k = 3$: за 3 шага дойти кратчайшим путем до цели из вершины a.

Приступаем к решению сформулированного класса задач. Ход решения будем фиксировать, устанавливая около каждой рассмотренной вершины графа пометку — указатель в виде кружка с числом и стрелкой, где число обозначает длину к р а т ч а й ш е г о пути до цели из этой вершины, а стрелка указывает оптимальное решение в данном месте, то есть направление, соответствующее кратчайшему пути. На местности указатель можно представить себе в виде закопанного у развилки дорог столбика с соответствующей надписью.

Начнем с $k = 0$, то есть с конца. Задача состоит в том, чтобы за ноль шагов дойти из пункта f до цели. Ответ тривиален: никуда идти не нужно, при этом длина кратчайшего пути равна нулю. Ставим соответствующий указатель около вершины f (см. рис. 14.3).

Переходим к $k = 1$. Возможными начальными состояниями для одношаговой задачи являются вершины первого с конца рубежа, то есть вершины d и e. Для них решение также очевидно:

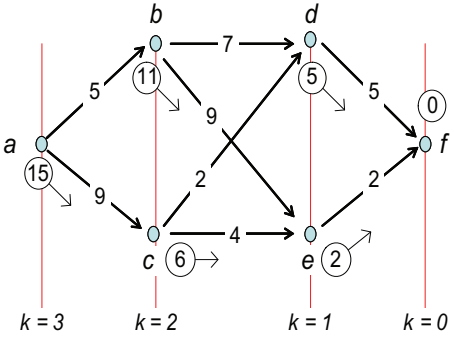


Рис. 14.3. Расстановка пометок у вершин

вариантов пути нет. Ставим соответствующие пометки у данных вершин.

Увеличиваем k до 2 и рассматриваем подкласс двухшаговых задач. Возможными начальными состояниями в этом случае являются вершины второго с конца рубежа: b и c. Здесь уже возможны варианты, выбрать оптимальное решение поможет третий фундаментальный принцип динамического программирования.

Принцип оптимальности. *Оптимальное поведение обладает тем свойством, что каковы бы ни были начальное состояние и начальное решение, последующее поведение должно быть оптимальным относительно состояния, получившегося в результате начального решения.*

Эту формулировку, данную самим Беллманом, невозможно улучшить, в ней важно каждое слово, поэтому ее следует не просто понять, а выучить наизусть.

Сделаем подробный разбор принципа оптимальности применительно к нашей задаче. *Поведение* означает последовательность решений, приводящих систему из некоторого начального в заданное конечное — целевое — состояние. Соответственно *оптимальное поведение* есть последовательность оптимальных решений.

Как говорит китайская пословица: путь длиною в тысячу ли[1] начинается с первого шага. Оптимальное поведение начинается с начального решения в начальном состоянии. Прежде чем принять это решение, нужно хорошо подумать, к чему оно приведет. Принцип оптимальности предлагает при выборе решения учитывать как непосредственный результат этого решения, так и отдаленный в предположении, что все дальнейшее поведение п о с л е начального решения б у д е т о п т и м а л ь н ы м.

Вооруженные принципом оптимальности, вернемся к нашему примеру и продолжим разметку вершин. Берем вершину b и начнем оценивать варианты решений. Их два: пойти налево или направо. Расстояние по левой дороге до соседней вершины равно 7, но в результате этого начального решения мы попадем в вершину d, откуда, как гласит расположенная там пометка, кратчайшее расстояние до цели равно 5. В сумме 12. Если же пойти по правой дороге, то, хотя расстояние до пункта e равно 9, зато оттуда до цели всего 2 ед. длины. Всего 11. Таким образом, с учетом непосредственного и отдаленного результатов, мы выбираем решение идти направо, при этом длина кратчайшего пути равна 11. Ставим соответствующую пометку у вершины b.

Аналогично помечаем вершину c. Левый поворот дает $2 + 5 = 7$, правый $4 + 2 = 6$. Оптимальное решение — направо, кратчайшее расстояние до цели 6.

Разметив вершины второго рубежа, решаем последнюю задачу: из вершины a дойти до цели за 3 шага. Применяя принцип оптимальности и учитывая, что двухшаговые задачи решены, а вершины второго рубежа уже помечены, сравниваем два варианта начального решения из вершины a. Левый поворот дает $5 + 11 = 16$, правый $9 + 6 = 15$. Оптимальное решение — направо, длина кратчайшего пути 15.

Итак, расставив пометки у вершин, мы решили класс задач нахождения кратчайшего пути из всех пунктов в целевой пункт. Идя по стрелкам, находим сам кратчайший путь $a \to c \to e \to f$.

[1] *Ли* — древняя китайская мера длины, около 0.5 км.

Замечание. Принципиальной особенностью динамического программирования является то, что в результате решения класса задач получается не просто оптимальная траектория движения из начального состояния к цели в пространстве состояний, а оптимальные решения для каждого состояния. Это предоставляет дополнительные возможности при реализации оптимального поведения. Если по каким-то причинам произошло отклонение от оптимальной траектории, то нет нужды возвращаться на нее, а следует реализовать оптимальное поведение, начинающееся в состоянии, в котором система оказалась. Другими словами: прошлого не вернуть, нужно оптимально распорядиться тем, что есть.

Например, если в нашем примере путник на первом же шаге выбрал неправильное направление и вместо пункта c попал в пункт b, то остаток пути нужно проложить по пунктам $b \to e \to f$.

Функция Беллмана

Аналитическим воплощением принципа оптимальности является *уравнение Беллмана.* Оно представляет собой специфическое функциональное уравнение относительно неизвестной *функции Беллмана.*

Функция Беллмана $B_k(s)$ зависит от двух аргументов и описывает весь класс задач, в который погружается конкретная оптимизационная задача. *Она равна оптимальному значению целевой функции, которое может быть достигнуто в результате решения k-шаговой задачи, если исходное состояние равно s.*

Для нашего примера $B_k(s)$ есть длина кратчайшего пути из вершины s до цели не более чем за k шагов.

Уравнение Беллмана

Согласно принципу оптимальности, результат k-шагового процесса, начинающегося из состояния s, складывается из двух составляющих: 1) непосредственного результата начального решения, 2) оптимального отдаленного результата от последующих $k-1$ решений.

Объединение этих результатов осуществляется для каждой задачи индивидуально, поэтому уравнение Беллмана всегда конкретно, записать его безотносительно к задаче нельзя.

Если обозначить через $\Gamma(s)$ множество вершин, доступных из

вершины s за один шаг, $l(s,x)$ — длину дуги из вершины s в вершину x, то принцип оптимальности для задачи о кратчайшем пути запишется в виде

$$B_k(s) = \min_{x\in\Gamma(s)}[l(s,x) + B_{k-1}(x)]. \tag{14.1}$$

Это и есть уравнение Беллмана для данной задачи. Начальные условия для него:

$$B_0(s) = \begin{cases} 0, \text{ если } s \text{ – целевая вершина,} \\ \infty \text{ в противном случае.} \end{cases}$$

Поскольку оптимизация в выражении (14.1) производится для каждого $k = 0, \ldots, n$ и каждого состояния s, оптимальное решение

$$x_k^*(s) = \arg\min_{x\in\Gamma(s)}[l(s,x) + B_{k-1}(x)]$$

определяется как функция от k и s. В нашей задаче $x_k^*(s)$ — это указатель вершины, в которую следует идти из вершины s при данном k.

Замечание. В динамическом программировании основная цель оптимизации — найти оптимальное поведение — как бы отодвигается на второй план. Оно находится попутно, при вычислении функции Беллмана во время нахождении оптимума.

П р и м е р. Решим нашу задачу чисто аналитическим путем, не прибегая к расстановке пометок на графе. Для этого зададим матрицу расстояний $l(x,y)$ между вершинами в виде

	a	b	c	d	e	f
a	0	5	9	∞	∞	∞
b	∞	0	∞	7	9	∞
c	∞	∞	0	2	4	∞
d	∞	∞	∞	0	∞	5
e	∞	∞	∞	∞	0	2
f	∞	∞	∞	∞	∞	0

Отсюда следует, что

$$\Gamma(a) = \{a, b, c\}; \quad \Gamma(b) = \{b, d, e\}; \quad \Gamma(c) = \{c, d, e\};$$
$$\Gamma(d) = \{d, f\}; \quad \Gamma(e) = \{e, f\}; \quad \Gamma(f) = \{f\}.$$

Начальные значения для функции Беллмана

$$B_0(a) = B_0(b) = B_0(c) = B_0(d) = B_0(e) = \infty, B_0(f) = 0.$$

Рекуррентные вычисления по формуле (14.1) делаем чисто формально. В качестве примера подробно покажем расчет $B_1(a)$ и $B_1(d)$ с попутным определением $x_1^*(a)$, $x_1^*(d)$:

$$B_1(a) = \min[l(a,a) + B_0(a); l(a,b) + B_0(b); l(a,c) + B_0(c)] =$$
$$= \min[0 + \infty; 5 + \infty; 9 + \infty] = \infty,$$
$$x_1^*(a) \text{ не определено;}$$

$$B_1(d) = \min[l(d,d) + B_0(d); l(d,f) + B_0(f)] =$$
$$= \min[0 + \infty; \underline{5 + 0}] = 5,$$
$$x_1^*(d) = f.$$

Дальнейшие вычисления представлены в табл. 14.1. Левая таблица для каждого s и k дает значения функции Беллмана $B_k(s)$, а правая — соответствующие оптимальные решения $x_k^*(s)$. Именно они были представлены на указателях, которые мы помещали около вершин при графическом решении задачи.

Теперь пришло время подставлять начальные условия. Исходная задача состояла в нахождении кратчайшего пути из a за три шага. Войдя в левую таблицу со значениями $k = 3$, $s = a$, получаем длину кратчайшего пути $B_3(a) = 15$.

Для того чтобы найти оптимальное поведение, нужно поработать с правой таблицей. В самом начале пути мы имеем трехшаговую задачу с начальным состоянием $s = a$. Войдя в таблицу со значениями $k = 3$, $s = a$, получаем указатель на следующее состояние $x_3^*(a) = c$. Далее имеем уже двухшаговую задачу с начальным состоянием c. Поднявшись в таблице на строчку выше,

определяем указатель $x_2^*(c) = e$. Теперь перед нами одношаговая задача с начальным состоянием e. В первой строке таблицы находим $x_1^*(e) = f$. Таким образом, оптимальное поведение есть последовательность решений $x_3^*(a) \to x_2^*(c) \to x_1^*(e)$, то есть $a \to c \to e \to f$. ▲

Таблица 14.1. Решение уравнения Беллмана для задачи о кратчайшем пути

Таблица $B_k(s)$

	s					
k	a	b	c	d	e	f
0	∞	∞	∞	∞	∞	0
1	∞	∞	∞	5	2	0
2	∞	11	6	5	2	0
3	15	11	6	5	2	0

Таблица $x_k^*(s)$

	s					
k	a	b	c	d	e	f
0	–	–	–	–	–	–
1	–	–	–	f	f	f
2	–	e	e	f	f	f
3	c	e	e	f	f	f

Замечание 1. В некоторых приложениях, например в сетевом планировании, требуется найти не кратчайший, а, наоборот, длиннейший, так называемый *критический путь* в графе. Эта задача может быть легко решена методом динамического программирования, для этого в уравнении Беллмана (14.1) минимизацию следует заменить максимизацией.

Замечание 2. Герой одного из рассказов, открыв учебник литературы, сильно удивился, когда узнал, что до сих пор всю жизнь говорил прозой. Все известные алгоритмы поиска экстремальных путей на графах в той или иной степени основаны на принципах динамического программирования несмотря на то, что в явном виде функция и уравнение Беллмана там не упоминаются. Хотя, возможно, все было наоборот: классическая задача о кратчайшем пути привела Беллмана к общей идее рекуррентной оптимизации.

14.2. Задача об инвестициях

Другой пример многошагового процесса принятия решений может быть сформулирован как простая задача размещения капитала.

Постановка задачи

Имеется капитал b единиц и n предприятий. Доход от помещения x_j капитала в j-е предприятие равен $c_j(x_j)$. Оптимизация портфеля инвестиций сводится к задаче сепарабельного, возможно невыпуклого программирования:

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{n} c_j(x_j) \to \max,$$
$$\sum_{j=1}^{n} x_j = b,$$
$$x_j \geqslant 0.$$

Для того чтобы применить к данной задаче аппарат динамического программирования, превращаем многомерную задачу в многошаговую: на первом шаге вкладываем капитал в предприятие 1, на втором — в предприятие 2, ... , на n-м — в предприятие n. На каждом шаге решается одномерная задача нахождения оптимального x_j^*.

Далее, следуя принципу погружения, погружаем задачу с конкретным числом предприятий n и конкретным объемом капитала b в класс задач: разместить s единиц капитала ($0 \leqslant s \leqslant b$) в k первых по счету предприятий ($0 \leqslant k \leqslant n$).

Функция и уравнение Беллмана

Функция Беллмана $B_k(s)$ для нашей задачи определяется как максимальный доход от вложения s единиц капитала в первые по счету k предприятий:

$$B_k(s) = \max_{x_j} \sum_{j=1}^{k} c_j(x_j) \quad \text{при} \quad \sum_{j=1}^{k} x_j = s, \quad x_j \geqslant 0. \tag{14.2}$$

Уравнение Беллмана записывается совершенно прозрачно: суммарный доход от k-шагового процесса инвестиций складывается из дохода на начальном шаге, то есть от инвестиции в k-е предприятие плюс оптимальный доход от оставшегося $(k-1)$-шагового процесса с учетом того, что часть капитала уже потрачена на начальном шаге:

$$B_k(s) = \max_{0 \leqslant x_k \leqslant s} [c_k(x_k) + B_{k-1}(s - x_k)]. \tag{14.3}$$

Начальные условия очевидны: если вложений нет, доход отсутствует, т. е.

$$B_0(s) \equiv 0. \tag{14.4}$$

Решая рекуррентно уравнение (14.3), можно получить значения функции Беллмана и попутно определить оптимальные решения $x_k^*(s)$, $k = 1, \ldots, n$.

П р и м е р. Некий инвестор имеет 5 у. е. (миллионов рублей или долларов) свободного капитала и желает им оптимально распорядиться.

Предположим, возможны три направления размещения денег.

1. Инвестировать капитал в торговлю. Тем самым обеспечен не слишком большой, но надежный доход, пропорциональный объему вложений.

2. Построить предприятие пищевой промышленности. Для такого бизнеса характерен достаточно быстрый возврат инвестиций при небольших вложениях, а затем наступает насыщение, вызванное ограниченной емкостью рынка.

3. Создать инновационное производство. Возврат инвестиций от такого производства наступает не сразу, а после превышения некоторого порогового уровня, однако в дальнейшем доходность предприятия быстро растет, насыщение рынка происходит при более высоких значениях прибыли.

Пример функций дохода $c_j(x)$ для каждого из трех направлений, разумеется, сильно стилизованный, приведен на рис. 14.4 в

табличном и графическом виде. Заметим, что две из трех функций являются невыпуклыми, поэтому обычные методы выпуклой оптимизации здесь неприменимы. К тому же функции $c_j(x)$ определены только на дискретном множестве точек с интервалом в 1 у. е.

Таблица $c_j(x)$

j	x					
	0	1	2	3	4	5
1	0	1	2	3	4	5
2	0	2	3	3	3	3
3	0	0	2	4	6	6

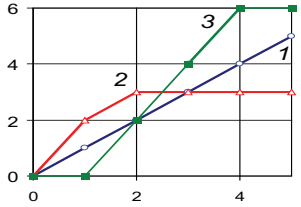


Рис. 14.4. Функции возврата инвестиций для трех направлений вложения капитала: *1* — торговля, *2* — обычное производство, *3* — инновации

Будем рекуррентно решать уравнение Беллмана (14.3). Значения функции $B_0(s)$ заданы начальными условиями (14.4), поэтому начнем с $k = 1$.

Функция Беллмана $B_1(s)$ для нашей задачи, согласно (14.2), равна

$$B_1(s) = \max_{0 \leqslant x_1 \leqslant s} c_1(x_1).$$

Для монотонной функции $c_1(x) = x$ максимум находится на правой границе области определения, отсюда $x_1^*(s) = s$, $B_1(s) = s$. Полученные значения сразу же заносим в таблицу окончательных результатов (табл. 14.2). Две из четырех строчек заполнены.

Процесс рекуррентного решения уравнения Беллмана для $k = 2$ и $k = 3$ приведен в табл. 14.3. Оптимизация по x_k осуществляется прямым перебором. Для каждого значения s (первая колонка) перебираются $x_k = 0, \ldots, s$ (вторая колонка), и в конце концов вычисляются суммы $c_k(x_k) + B_{k-1}(s - x_k)$ (последняя колонка). Наибольшее значение для каждого s, дающее значение $B_k(s)$, и соответствующее ему значение x_k обведены рамкой. Они переносятся в табл. 14.2 окончательных результатов.

Таблица 14.2. Функция Беллмана и оптимальные решения для задачи об инвестициях

$B_k(s)$	s					
k	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	2	3	4	5	6
3	0	2	3	4	6	8

$x_k^*(s)$	s					
k	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	2	3	4	5
2	0	1	1	1	1	1
3	0	0	0	0	3	4

После того как эта таблица заполнена полностью, можно интерпретировать полученные результаты, подставляя конкретные начальные условия. Исходная задача заключалась в трехшаговом процессе инвестирования с начальным капиталом $b = 5$. Войдя в левую таблицу с $k = 3$, $s = 5$, получаем $B_s(5) = 8$, что соответствует наибольшему достижимому значению целевой функции, это число обведено рамкой.

Для нахождения самих объемов инвестирования анализируем правую таблицу оптимальных решений. Войдя в нее начальными условиями $k = 3$, $s = 5$, получаем $x_3^*(5) = 4$, это значит, что в третье (инновационное) направление нужно вложить 4 у. е. капитала. После этого задача стала двухшаговой, а остаток капитала составил 1 у. е. Поднимаясь на строчку выше, находим $x_2^*(1) = 1$. Задача стала одношаговой, состояние капитала нулевое. В первой строке определяем $x_1^*(0) = 0$. Все перечисленные оптимальные решения отмечены рамкой.

Таким образом, окончательный ответ такой: $x_1^* = 0$, $x_2^* = 1$, $x_3^* = 4$, $f^* = 8$. Возвращаясь к рис. 14.4, убеждаемся в разумности такого решения. Вложения в высокие технологии соответствуют точке перегиба функции возврата инвестиций. ▲

Таблица 14.3. Рабочая таблица для решения уравнения Беллмана в задаче об инвестициях

Шаг $k = 2$

s	x_2	$s - x_2$	$c_2(x_2)$	B_1	Σ
0	**0**	0	0	0	**0**
1	0	1	0	1	1
	1	0	2	0	**2**
2	0	2	0	2	2
	1	1	2	1	**3**
	2	0	3	0	3
3	0	3	0	3	3
	1	2	2	2	**4**
	2	1	3	1	4
	3	0	3	0	3
4	0	4	0	4	4
	1	3	2	3	**5**
	2	2	3	2	5
	3	1	3	1	4
	4	0	3	0	3
5	0	5	0	5	5
	1	4	2	4	**6**
	2	3	3	3	6
	3	2	3	2	5
	4	1	3	1	4
	5	0	3	0	3

Шаг $k = 3$

s	x_3	$s - x_3$	$c_3(x_3)$	B_2	Σ
0	**0**	0	0	0	**0**
1	**0**	1	0	2	**2**
	1	0	0	0	0
2	**0**	2	0	3	**3**
	1	1	0	2	2
	2	0	2	0	2
3	**0**	3	0	4	**4**
	1	2	0	3	3
	2	1	2	2	4
	3	0	4	0	4
4	0	4	0	5	5
	1	3	0	4	4
	2	2	2	3	5
	3	1	4	2	**6**
	4	0	6	0	6
5	0	5	0	6	6
	1	4	0	5	5
	2	3	2	4	6
	3	2	4	3	7
	4	1	6	2	**8**
	5	0	6	0	6

14.3. Непрерывная оптимизация

В рассмотренных выше примерах число переменных в задаче оптимизации и, следовательно, число шагов в многошаговой динамической модели предполагалось конечным. Однако наиболее интересные результаты в динамическом программировании получаются в бесконечномерном случае. Эта область оптимизации относится к вариационному исчислению [8], подробное изучение которого выходит за рамки нашего курса. Тем не менее имеет смысл хотя бы на элементарном примере продемонстрировать связь динамического программирования с вариационным исчислением.

Основная задача вариационного исчисления

Простейшую (основную) задачу вариационного исчисления с закрепленными концами можно рассматривать как обобщение на непрерывный случай задачи о кратчайшем пути. В одномерном случае она ставится следующим образом.

На плоскости (t, x) заданы точки $A(t_0, x_0)$ и $B(t_1, x_1)$ (рис. 14.5).

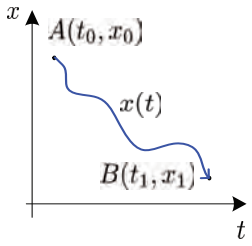


Рис. 14.5. Вариационная задача с закрепленными концами

В момент времени t_0 объект находился в точке A, в момент времени t_1 должен оказаться в точке B. Нужно найти такую траекторию движения $x(t)$, которая минимизирует интегральный функционал:

$$\Phi = \int_{t_0}^{t_1} F[t, x(t), x'(t)]dt. \qquad (14.5)$$

П р и м е р. Вариационное исчисление отсчитывает свою историю с классической *задачи о брахистохроне* (греч. *брахистос* — кратчайший, *хронос* — время), решенной еще в конце XVII века (см. исторические замечания в конце главы). Если на рис. 14.5 переменную x понимать как высоту, а t как длину, требуется найти такую форму кривой $x(t)$, по которой материальный шарик скатится из A в B за кратчайшее время.

Закон сохранения энергии гласит, что полная механическая энергия материальной точки, складывающаяся из потенциальной mgx и кинетической $mv^2/2$, должна оставаться неизменной:

$$mgx(t) + \frac{mv^2(t)}{2} = mgx(t_0) + \frac{mv^2(t_0)}{2}.$$

Отсюда, полагая $x(t_0) = x_0, v(t_0) = 0$, определим скорость скатывания шарика:

$$v(t) = \sqrt{2g(x_0 - x(t))}.$$

Элемент пути равен $ds = \sqrt{dx^2 + dt^2} = \sqrt{1 + (x')^2}dt$, элемент времени ds/v. Тогда полное время спуска

$$T = \int\limits_{x(t_0)=x_0}^{x(t_1)=x_1} \frac{ds}{v} = \int\limits_{t_0}^{t_1} \sqrt{\frac{1 + (x'(t))^2}{2g(x_0 - x(t))}}dt. \qquad (14.6)$$

В результате получилась простейшая вариационная задача с закрепленными концами типа (14.5), в которой подынтегральное выражение не зависит явно от t. ▲

Сведение к динамическому программированию

Для того чтобы описать бесконечномерную задачу (14.5) моделью динамического программирования, нужно превратить ее в многошаговую и погрузить в класс однородных задач. Рассматривая динамический процесс движения на плоскости, образуем класс задач поиска оптимального пути из п р о и з в о л ь н о й точки (t, x) в целевую точку

(t_1, x_1) и введем функцию Беллмана как минимально достижимое значение функционала:

$$B(x,t) = \min_{x(t)} \int_t^{t_1} F[t, x(t), x'(t)]dt.$$

Непрерывный процесс движения можно представить себе в виде бесконечной последовательности бесконечно малых шагов, соответствующих элементарным приращениям dt.

Для составления уравнение Беллмана предположим, что мы находимся в некотором начальном состоянии (t, x). Тогда непрерывная траектория движения к цели будет складываться из элементарного начального участка на интервале $(t, t + dt)$ и оставшейся части, которую, согласно принципу оптимальности, следует считать оптимальной (рис. 14.6).

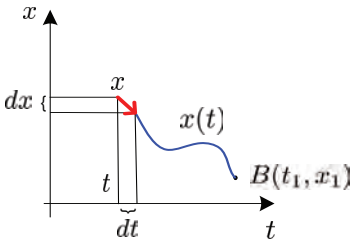


Рис. 14.6. Бесконечношаговая динамическая задача

Решение, которое принимается на начальном шаге из точки t, состоит в выборе величины dx при фиксированном dt. Иными словами — это выбор значения производной $x'(t) = dx/dt$.

Непосредственный результат от начального элементарного шага равен

$$\int_t^{t+dt} F(t, x, x')dt \approx F(t, x, x')dt.$$

После этого мы переместимся в состояние $(t+dt, x+dx)$, функционал оптимального пути из которого до цели будет равен значению

функции Беллмана $B(x + dx, t + dt)$. Воспользуемся принципом оптимальности и запишем уравнение Беллмана для бесконечношагового процесса:

$$B(x,t) = \min_{x'(t)}[F(t,x,x')dt + B(x + dx, t + dt)]. \tag{14.7}$$

Если функцию Беллмана считать непрерывной, то ее можно разложить в ряд

$$B(x + dx, t + dt) = B(x,t) + \frac{\partial B(x,t)}{\partial x}x'dt + \frac{\partial B(x,t)}{\partial t}dt. \tag{14.8}$$

Подставляем (14.8) в (14.7) и делим на dt. Получаем

$$\min_{x'(t)}\left[F(t,x,x') + \frac{\partial B(x,t)}{\partial x}x' + \frac{\partial B(x,t)}{\partial t}\right] = 0. \tag{14.9}$$

Выражение 14.9 содержит в себе два условия:

1) условие минимума по x', которое можно записать, приравнивая нулю частную производную выражения в скобках по x':

$$\frac{\partial F(t,x,x')}{\partial x'} + \frac{\partial B(x,t)}{\partial x} = 0; \tag{14.10}$$

2) равенство этого минимума нулю:

$$F(t,x,x') + \frac{\partial B(x,t)}{\partial x}x' + \frac{\partial B(x,t)}{\partial t} = 0. \tag{14.11}$$

Получились два дифференциальных уравнения первого порядка в частных производных относительно функции Беллмана. Однако нас в конечном счете интересует не функция Беллмана, а оптимальная траектория $x(t)$. Поэтому мы постараемся свести оба уравнения к одному обыкновенному дифференциальному уравнению второго порядка относительно $x(t)$, не содержащего функцию Беллмана.

Возьмем полную производную (14.10) по t:

$$\frac{d}{dt}\left[\frac{\partial F(t,x,x')}{\partial x'}+\frac{\partial B(x,t)}{\partial x}\right]= \\ =\frac{d}{dt}\frac{\partial F(t,x,x')}{\partial x'}+\frac{\partial^2 B(x,t)}{\partial x^2}\frac{dx}{dt}+\frac{\partial^2 B(x,t)}{\partial x\partial t}=0. \quad (14.12)$$

Возьмем частную производную (14.11) по x:

$$\frac{\partial}{\partial x}\left[F(t,x,x')+\frac{\partial B(x,t)}{\partial x}x'+\frac{\partial B(x,t)}{\partial t}\right]= \\ =\frac{\partial F(t,x,x')}{\partial x}+\frac{\partial^2 B(x,t)}{\partial x^2}\frac{dx}{dt}+\frac{\partial^2 B(x,t)}{\partial x\partial t}=0. \quad (14.13)$$

Вычитая из (14.13) выражение (14.12), получаем

$$\frac{\partial F(t,x,x')}{\partial x}-\frac{d}{dt}\frac{\partial F(t,x,x')}{\partial x'}=0. \quad (14.14)$$

Это — классическое уравнение Эйлера в вариационном исчислении, записанное в краткой форме. Если раскрыть полную производную

$$\frac{d}{dt}\frac{\partial F(t,x,x')}{\partial x'}=\frac{\partial^2 F}{\partial t\partial x'}+\frac{\partial^2 F}{\partial x\partial x'}x'+\frac{\partial^2 F}{\partial x'\partial x'}x'',$$

то можно убедиться, что уравнение Эйлера есть обыкновенное дифференциальное уравнение второго порядка с переменными коэффициентами

$$\frac{\partial^2 F}{\partial x'\partial x'}x''+\frac{\partial^2 F}{\partial x\partial x'}x'+\frac{\partial^2 F}{\partial t\partial x'}-\frac{\partial F}{\partial x}=0, \quad (14.15)$$

которое необходимо решать с начальными условиями $x(t_0)=x_0$, $x(t_1)=x_1$.

Замечание 1. В классическом вариационном исчислении уравнение Эйлера выводится по-другому, с использованием понятия *вариации*

функции, которое является обобщением понятия производной на бесконечномерный случай.

Замечание 2. Аналитическое решение уравнение Эйлера может быть найдено в очень редких случаях [21, с. 419]. В частности, когда подынтегральное выражение (14.6) не зависит явно от t, уравнение (14.15) приобретает вид

$$\frac{\partial^2 F}{\partial x' \partial x'} x'' + \frac{\partial^2 F}{\partial x \partial x'} x' - \frac{\partial F}{\partial x} = 0.$$

Легко показать, что оно эквивалентно:

$$\frac{d}{dt}\left(F - x' \frac{\partial F}{\partial x'}\right) = 0. \tag{14.16}$$

Действительно,

$$\begin{aligned}
&\frac{d}{dt}\left(F(x,x') - x' \frac{\partial F(x,x')}{\partial x'}\right) = \\
&= \frac{\partial F}{\partial x} x' + \frac{\partial F}{\partial x'} x'' - x'' \frac{\partial F}{\partial x'} - x' \frac{\partial^2 F}{\partial x' \partial x} x' - x' \frac{\partial^2 F}{\partial x' \partial x'} x'' = \\
&= -x' \left(\frac{\partial^2 F}{\partial x' \partial x'} x'' + \frac{\partial^2 F}{\partial x \partial x'} x' - \frac{\partial F}{\partial x}\right).
\end{aligned}$$

Уравнение (14.16) имеет очевидное решение (первый интеграл):

$$F - x' \frac{\partial F}{\partial x'} = C. \tag{14.17}$$

Таким образом, в данном частном случае искомая функция $x(t)$ (она называется *экстремалью*) может быть найдена в результате решения дифференциального уравнения первого порядка.

П р и м е р. В задаче о брахистохроне подынтегральное выражение (14.6) не зависит от t, следовательно, решение $x(t)$ дается первым интегралом. Подставляя (14.6) в (14.17), получаем

$$\frac{\sqrt{1 + x'^2}}{\sqrt{x_0 - x}} - \frac{x'^2}{\sqrt{x_0 - x}\sqrt{1 + x'^2}} = \frac{1}{\sqrt{c_1}},$$

откуда после упрощений имеем $x'^2 = (c_1 - x_0 + x)/(x_0 - x)$. Сделаем подстановку $(x_0 - x) = \dfrac{c_1}{2}(1 - \cos z)$. Для нее $x' = -\dfrac{c_1}{2} z' \sin z$. Подставляя в уравнение, последовательно получаем

$$\frac{c_1^2}{4} z'^2 (1 - \cos z)^2 = 1,$$
$$\frac{c_1}{2} z' (1 - \cos z) = \pm 1,$$
$$\frac{c_1}{2}(1 - \cos z) dz = \pm dt.$$

Отсюда

$$\begin{cases} t = \pm \dfrac{c_1}{2}(z - \sin z) + c_2, \\ x = \dfrac{c_1}{2}(1 - \cos z). \end{cases} \tag{14.18}$$

Выражения (14.18) представляют собой параметрические уравнения кривой, носящей название *циклоида*. То есть брахистохрона является циклоидой. Это — линия, которую описывает точка окружности радиуса c_1, катящейся по оси абсцисс. Параметры c_1 и c_2 определяются из начальных условий $x(t_0) = x_0,\ x(t_1) = x_1$. Название кривой дал Галлилей, впервые обративший на нее внимание. ▲

14.4. Исторические замечания

Как это не раз случалось в математике, многие непрерывные задачи были поставлены и решены ранее дискретных. Классическое вариационное исчисление появилось на 200 лет ранее динамического программирования.

Исторически первой задачей вариационного исчисления является задача о брахистохроне, над которой размышлял еще великий итальянец Галлилео Галлилей (Galilei, Galileo; 1564–1642), при этом он ошибочно полагал, что искомая кривая — дуга окружности. Для нахождения правильного решение потребовалось создание анализа бесконечно малых. В 1696 г., через 12 лет после

публикации первой статьи Лейбница о наибольших и наименьших значениях , у его швейцарского последователя Иоганна Бернулли возникла мысль исследовать брахистохрону строго математически. Получив результат, Иоганн сформулировал задачу в письме Лейбницу, который легко с ней справился и посоветовал выставить на конкурс. В ответ пришли еще три решения, все верные: от Якова Бернулли, Лопиталя[2] и анонимного автора, напечатавшего ответ в английском журнале. Прочитав неподписанный текст, Бернулли произнес: «Tamquam ex ungue leonem»[3]. Автором, несомненно, был Исаак Ньютон. Брахистохрона оказалась циклоидой.

Систематическое изучение вариационного исчисления началось полвека спустя благодаря трудам великих ученых XVIII столетия Эйлера и Лагранжа.

Леонард Эйлер

Леонард Эйлер (Euler, Leonhard; 1707–1783) — самый продуктивный математик в истории, автор более чем 800 работ по математическому анализу, дифференциальной геометрии, теории чисел, приближенным вычислениям, небесной механике, математической физике, оптике, баллистике, кораблестроению, теории музыки и др., собрание его сочинений составляет 72 тома.

Эйлер родился в Швейцарии, учился в Базельском университете у Иоганна Бернулли и был дружен с его сыновьями Николаем и Даниилом. После организации Петром I Академии наук все трое друзей — перспективных молодых ученых — были приглашены работать в Петербург, где Эйлер прожил с 1726 по 1741 г. В эпоху дворцовых переворотов, когда власть в России попала в руки регентши Анны Леопольдовны и его «положение начало

[2] Маркиз де Лопиталь (de L'Hopital, Guillaume Francois Antoine; 1661–1704) — французский математик, ученик Иоганна Бернулли, автор самого первого учебника по математическому анализу (1696). Нынешнему поколению студентов известен благодаря *правилу Лопиталя.*

[3] *Tamquat ex ungue leonem* [*тампквам экс унгэ леонэм*] — *узнаю льва по когтям.* Латинская пословица, соответствует русской «видно птицу по полету».

представляться довольно неуверенным», Эйлер принял приглашение прусского короля Фридриха II и пробыл в Берлине следующие 25 лет. В 1766 г. по настоятельной просьбе Екатерины II, создавшей ему идеальные условия для жизни и работы, вновь вернулся в Россию и оставался там до конца жизни. Вскоре после приезда Эйлер потерял зрение, но это никак не повлияло на его работоспособность, свои труды он диктовал мальчику-слуге. Хорошо знал русский язык, на нем опубликовал часть своих сочинений. Покоится на мемориальном кладбище у Александро-Невской лавры.

Книга «Метод нахождения кривых линий, обладающих свойствами максимума, либо минимума...» вышла в 1744 г., это было первое систематическое изложение вариационного исчисления. В ней и последующих трудах Эйлер в отчетливой математической постановке сформулировал *принцип наименьшего действия,* высказанный ранее французским математиком и астрономом **Мопертюи** (de Maupertuis, Pierre-Louis Moreau; 1698–1759).

Согласно рассуждениям Мопертюи, совершенство созданной Богом природы не допускает любой бесполезной работы. Естественное движение должно быть таким, чтобы сделать некоторую величину минимальной. Например, луч света, двигаясь в неоднородной среде, выбирает такую траекторию, при которой время прохождения минимально (этот основополагающий принцип геометрической оптики был сформулирован Пьером Ферма в 1666 г.); свободно подвешенная цепь принимает форму, минимизирующую ее потенциальную энергию и т д. Проблема в том, чтобы для каждой задачи правильно определить эту величину, называемую *действием.* Мопертюи ее подбирал эмпирически, а Эйлер предложил описывать действие интегральным функционалом.

Экстремальный принцип оказался удивительно плодотворным. Он был продолжен и усовершенствован следующими поколениями ученых. К ним прежде всего следует отнести **Жозефа Луи Лагранжа** (1736–1813), о котором мы говорили в исторических замечаниях к гл. 9 и к которому Эйлер относился с большой симпатией. Известен случай, когда Эйлер задержал свои публи-

кации по вариационному исчислению, чтобы молодой и никому тогда не известный Лагранж, независимо пришедший к тем же открытиям, смог опубликовать их первым.

Вершиной развития вариационного подхода к классической механике можно считать труды самого знаменитого из ирландских математиков **Уильяма Гамильтона** (Hamilton, William Rowan; 1805–1865). Введенный Гамильтоном формализм для описания механических систем стал классическим, а способ получения уравнений движения физической системы на основании принципа наименьшего (точнее, *стационарного*) действия получил названия *принципа Гамильтона*[4]. В XX веке вариационные принципы классической механики были распространены на механику сплошных сред, квантовую механику и теорию поля.

Уильям Гамильтон на ирландской почтовой марке


Ричард Беллман

Автором динамического программирования является выдающийся американский математик **Ричард Беллман** (Bellman, Richard Ernest; 1920–1984). Беллман родился в Нью-Йорке, где его отец держал небольшую бакалейную лавку в районе Бруклина. Изучал математику и электронику в Brooklyn College и University of Wisconsin. Во время Второй мировой войны некоторое время работал по проблеме атомной бомбы в теоретическом отделе Los Alamos National Laboratory.

[4]Кроме этого Гамильтон ввел понятие векторного поля и создал основы векторного анализа, предложил оператор набла ∇, создал теорию гиперкомплексных чисел — кватернионов — с тремя мнимыми единицами. В теории графов известен гамильтонов цикл.

В 1946 г. защитил диссертацию по теории дифференциальных уравнений в знаменитом Принстонском университете, где в это время сложилась выдающаяся математическая школа, включавшая известных нам Джона фон Неймана, Альберта Такера, Гарольда Куна и др.

Посетив в 1948 г. только что созданную в Калифорнии RAND Corporation (о ней мы говорили в кратком введении в историю исследования операций [9, с. 13]), Беллман был впечатлен свободной творческой атмосферой, которая царила в этой удивительной исследовательской компании, а также научным уровнем сотрудников, среди которых был, в частности, автор линейного программирования Джордж Данциг. В итоге Беллман предпочел прикладную математику академической и 13 лет проработал в RAND на штатной должности.

В 1952 г. была опубликована первая статья, а 1957 г. — знаменитая монография по динамическому программированию [4], сразу завоевавшая мировую славу. По поводу необычного названия Беллман объяснял, что поскольку RAND финансировалась Министерством обороны, «теория многошаговых процессов» для военных звучала слишком абстрактно, тогда как слово «программирование» было вполне привычным армейским термином (см. [9, с. 41]). Кроме того, в новом названии содержался намек на превосходство перед линейным программированием Данцига.

Обладая прекрасными литературными данными, Беллман проявил удивительную творческую активность. Всего он опубликовал 619 статей и 39 книг, являясь одним из самых цитируемых математиков в мире. О широте его научных интересов говорит такой факт. Когда в 1965 г. Беллман вернулся к преподавательской работе в Университете южной Калифорнии в Лос-Анджелесе, он стал там профессором математики, электротехники и медицины.

В 1973 г. в возрасте 53 лет, находясь на вершине своей научной деятельности, Беллман перенес операцию по удалению опухоли мозга, в результате осложнения потерял способность двигаться. Всю оставшуюся жизнь он был прикован к инвалидному креслу, сохранив при этом полную умственную работоспособность.

Литература

1. *Ануфриев И. Е., Смирнов А. Б., Смирнова Е. Н.* MATLAB 7. — СПб.: БХВ-Петербург, 2005. — 1104 с.

2. *Аоки М.* Введение в методы оптимизации: пер. с англ. — М.: Наука. Гл. ред. физ.-мат. лит., 1977. — 344 с.

3. *Аттетков А. В., Галкин С. В., Зарубин В. С.* Методы оптимизации: учеб. для вузов. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2003. — 440 с. (Сер. Математика в техническом вузе. Вып. XIV).

4. *Беллман Р.* Динамическое программирование: пер. с англ. — М.: ИЛ, 1960. — 400 с.

5. *Бертсекас Д.* Условная оптимизация и методы множителей Лагранжа: пер. с англ. — М.: Радио и связь, 1987. — 400 с.

6. *Воробьев Н. Н.* Числа Фибоначчи. — М.: Наука, 1978. — 144 с.

7. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация: пер. с англ. — М.: Мир, 1985. — 509 с.

8. *Гельфанд И. М., Фомин С. В.* Вариационное исчисление. — М.: Физматгиз, 1961.

9. *Гладких Б. А.* Методы оптимизации и исследование операций для бакалавров информатики. Ч. I. Введение в иссле-

дование операций. Линейное программирование: учеб. пособие. — Томск: Изд-во НТЛ, 2009. — 200 с.

10. *Головина Л. И.* Линейная алгебра и некоторые ее приложения: учеб. пособие. — 2-е изд. — М.: Наука. Гл. ред. физ.-мат. лит., 1975. — 408 с.

11. *Демьянов В. Ф., Васильев Л. В.* Недифференцируемая оптимизация. — М.: Наука. Гл. ред. физ.-мат. лит., 1981. — 384 с.

12. *Емельянов В. В., Курейчик В. В., Курейчик В. М.* Теория и практика эволюционного моделирования. — М.: Физматлит, 2003. — 432 с.

13. *Жилявский А. А., Жилинскас А. Г.* Методы поиска глобального экстремума. — М.: Наука. Гл. ред. физ.-мат. лит., 1991. — 248 с.

14. *Зойтендейк Г.* Методы возможных направлений. — М.: ИЛ, 1963. — 176 с.

15. *Змеев О. А., Терпугов А. Ф., Якупов Р. Т.* Математический анализ. Ч. III. — Томск: Изд-во НТЛ, 2007. — 152 с.

16. *Канатников А. Н., Крищенко А. П.* Линейная алгебра: учеб. для вузов. — М.: Изд-во МГТУ им. Н. Э. Баумана, 2002. — 336 с. (Сер. Математика в техническом вузе. Вып. IV).

17. *Карманов В. Г.* Математическое программирование. — 5-е изд. — М.: Физматлит, 2004. — 264 с.

18. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. — М., 1974. — 832 с.

19. *Мину М.* Математическое программирование. Теория и алгоритмы: пер. с фр. — М.: Наука. Гл. ред. физ.-мат. лит., 1990. — 488 с.

20. *Панченко Т. В.* Генетические алгоритмы: учеб.-методич. пособие. — Астрахань: Изд. дом «Астраханский университет», 2007. — 87 с.

21. *Пантелеев А. В., Летова Т. А.* Методы оптимизации в примерах и задачах: учеб. пособие. — 2-е изд., испр. — М.: Высш. шк., 2005. — 544 с.

22. *Растригин Л.А̇.* Статистические методы поиска. — М.: Наука. Гл. ред. физ.-мат. лит., 1968. — 376 с.

23. *Реклейтис Г., Рейвиндран А., Рэгсдел К.* Оптимизация в технике: в 2-х кн.: пер. с англ. — М.: Мир, 1986. — Кн. 1. — 349 с. — Кн. 2. — 320 с.

24. *Сухарев А. Г., Тимохов А. В., Федоров В. В.* Курс методов оптимизации. — 2-е изд. — М.: Физматлит, 2005. — 368 с.

25. *Уайлд Д. Дж.* Методы поиска экстремума: пер. с англ. — М.: Наука. Гл. ред. физ.-мат. лит., 1967. — 268 с.

26. *Фиакко А., Мак-Кормик Г.* Нелинейное программирование. Методы последовательной безусловной минимизации: пер. с англ. — М.: Мир., 1972. — 240 с.

27. *Химмельблау В. В.* Прикладное нелинейное программирование). — 2-е изд. — М.: Физматлит, 2005. — 534 с

28. *Fletcher R.* Practical Methods of Optimization. — Chichester–New York–Brisbane–Toronto: John Wiley & Sons. — V 1: Unconstrained optimization, 1980. — viii+120 p.; V 2: Constrained optimization, 1981. — ix+224 p.

29. *GAMS* — A User's Guide. — Washington, DC: GAMS Development Corporation, 2006. — 251 p.

30. *Conn A. R., Gould N. I. M. and Toint P. L.* Trust-Region Methods. — Philadelphia, PA: SIAM, 2000. — xx+959 p.

Учебное пособие

ГЛАДКИХ Борис Афанасьевич

МЕТОДЫ ОПТИМИЗАЦИИ И ИССЛЕДОВАНИЕ ОПЕРАЦИЙ ДЛЯ БАКАЛАВРОВ ИНФОРМАТИКИ

Часть II. Нелинейное и динамическое программирование

Редактор *Н. И. Шидловская*
Вёрстка *Б. А. Гладких*
Дизайн *Д. В. Фортеса*

Изд. лиц. ИД № 04000 от 12.02.2001. Подписано к печати 23.06.11.
Формат $60 \times 84\,{}^{1}/_{16}$**. Бумага офсетная. Печать офсетная.**
Гарнитура «Computer Modern Super». Усл. печ. л. 15,5.
Уч.-изд. л. 17,4. Тираж 200 экз.

ООО «Издательство научно-технической литературы»
634050, г. Томск, пл. Ново-Соборная, 1, тел. (3822) 533-335

Отпечатано в типографии ЗАО «М-Принт»,
г. Томск, ул. Пролетарская, 38/1