Министерство образования и науки Российской федерации

Государственное образовательное учреждение
Высшего профессионального образования
Санкт Петербургский государственный технологический институт
(Технический университет)

И.А. Смирнов

МЕТОДЫ ОПТИМИЗАЦИИ. БАЗОВЫЙ КУРС

Учебное пособие для студентов заочной формы обучения направления подготовки «Информатика и вычислительная техника»

Санкт-Петербург
2010

УДК 517.51

Смирнов, И.А. Методы оптимизации. Базовый курс: учебное пособие / И.А. Смирнов. – СПб.: СПбГТИ(ТУ), 2010. 102 – с.

Учебное пособие посвящено изучению основных оптимизационных методов. В пособии даются базовые понятия теории оптимизации, рассмотрены основные подходы к решению задач оптимизации.
Учебное пособие предназначено для студентов 3 курса заочной формы обучения по направлению подготовки 230100 «Информатика и вычислительная техника» и соответствуют рабочей программе дисциплины «Методы оптимизации».

Библиогр. 7 назв., илл. 28

Рецензенты: 1 Московский государственный университет прикладной биотехнологии. В. В. Митин зав. кафедрой конструирования машин, САПР и инженерной графики, д-р техн. наук, профессор.

2 В. К. Викторов, зав. кафедрой информационных систем в химической технологии Санкт-Петербургского государственного технологического института (технического университета) , д-р техн. наук, профессор.

Утверждены на заседании учебно-методической комиссии факультета информатики и управления 24.05.2010.

Рекомендованы к изданию РИСо СПбГТИ(ТУ).

Содержание

Список обозначений

$a \in A$ – элемент *a* принадлежит множеству *A*.

$a \notin A$ – элемент *a* не принадлежит множеству *A*.

$A \subset B$ – множество *A* является подмножеством множества *B*.

N – множество натуральных чисел.

Z – множество целых чисел.

R – множество действительных чисел.

R^n – (декартово) произведение *n* множеств действительных чисел или *n*-мерное евклидово арифметическое пространство.

[a,b] и *(a,b)* – отрезок и интервал с концами в точках *a* и *b*.

[a,b) и *(a,b]* –полуинтервалы с концами в точках *a* и *b*.

$f(a), f(x)\big|_{x=a}$ – значение функции $f(x)$ в точке *a*.

$D(f)$ и $R(f)$ – область определения и область значения функции $f(x)$.

$\sum_{j=1}^{n} a_k$ – сумма *n* слагаемых $a_1, a_2 ..., a_n$.

$\prod_{m=1}^{N} a_m$ – произведение *n* сомножителей.

$k = \overline{1, N}$ – число *k* принимает последовательно все значения из множества натуральных чисел от 1 до *N* включительно.

∂X – граница множества X.

$\sup_{x \in X} f(x)$ и $\inf_{x \in X} f(x)$ – точка верхняя и точка нижняя грани функции $f(x)$ на множестве *X*.

$\{x_n\}$ – *последовательность элементов* x_n.

$\lim_{n \to \infty} x_n$ – предел последовательности $\{x_n\}$ при $n \to \infty$.

$\lim_{x \to a} f(x)$ – предел функции $f(x)$ одного действительного переменного *x* в точке *a* (при $x \to a$).

$f'(a), f'(x)\big|_{x=a}$ – произвольная функция $f(x)$ одного действительного переменного в точке *a*.

AB и $|AB|$ – отрезок, соединяющий точки *A* и *B*, и его длина.

$x = (x_1 ... x_n)^T$ – вектор из R^n с координатами $x_1, ..., x_n$.

$|x|$ – длина (модуль) вектора x.

0_n – нулевой вектор из R^n.

(a,b) – скалярное произведение векторов *a* и *b*.

A^T – матрица, транспонированная к матрице *A*.

A^{-1} – матрица, обратная к матрице *A*.

I_n – единичная матрица порядка *n*.

$\int_a^b f(x)dx$ – определенный интеграл от функции $f(x)$ по отрезку *[a,b]*.

$f(x) \to min, \quad x \in \Omega$, – задача минимизации функции $f(x)$ на множестве Ω.

$f(x) \to \inf, \quad x \in \Omega$, – задача нахождения точной нижней грани функции $f(x)$ на множестве Ω.

$a \geq b, b \leq a$ – каждая координата вектора $a \in R^n$ не меньше соответствующей координаты вектора $b \in R^n$.

$Ax \leq b$ – система из m неравенств $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$, $i = \overline{1,m}$, определяемая матрицей $A = (a_{ij})$ размера *m* x *n* и столбцом $b = (b_1 ... b_n)^T$ высоты *n*.

$\exp(x), e^x$ – экспоненциальная функция.

Введение

В своей жизни человек часто сталкивается с ситуацией, когда ему из некоторой совокупности возможных вариантов своего поведения или принятия решения в какой-либо области деятельности необходимо выбрать один вариант. Наилучший вариант поведения (принятие наилучшего решения) можно выбирать по-разному. Если такой выбор предусматривает проведение количественного анализа ситуации путем сравнения различных вариантов с помощью какой-либо количественной оценки этих вариантов, то говорят о необходимости решения *задачи оптимизации* (по латыни optimus — наилучший). Ясно, что задача оптимизации имеет смысл, если есть несколько возможных вариантов се решения. Эти варианты обычно называют *альтернативами.*

По содержанию задачи оптимизации весьма разнообразны. Они могут быть связаны с проектированием технических устройств и технологических процессов, с распределением ограниченных ресурсов и планированием работы предприятий, наконец, с решением проблем, возникающих в повседневной жизни человека. Всевозможные устройства, процессы и ситуации, применительно к которым предстоит решать задачу оптимизации, объединим общим названием *объект оптимизации.*

Обычно человек хочет сделать „как лучше", но, чтобы не получить плохой результат при самых хороших намерениях, для решения задачи оптимизации нужно прежде всего найти ответы на следующие вопросы:

- Что значит "лучше"?
- Что конкретно нужно улучшить?

-За счет чего можно добиться улучшения, что можно изменить?

-В каких пределах можно производить изменения?

Отвечая на первый вопрос, необходимо сформулировать ***критерий оптимальности,*** т.е. определить те признаки и предпочтения, по которым следует провести сравнительную оценку альтернатив и выбрать среди них наилучшую с точки зрения поставленной цели оптимизации. Именно с этой точки зрения можно ответить на второй вопрос: что конкретно нужно улучшить? Это может быть повышение производительности станка или срока службы технического устройства, снижение массы конструкции летательного аппарата или затрат на его производство и т.п.

Для ответа на два последних вопроса необходимо располагать *математической моделью* объекта оптимизации. Эта модель описывает объект при помощи соотношений между величинами, характеризующими его свойства. Обычно хотя бы часть этих величин можно изменять в некоторых пределах, что и порождает множество альтернатив, среди которых и предстоит выбрать наилучшую. Изменяемые при оптимизации величины, входящие в математическую модель объекта оптимизации, называют ***параметрами оптимизации,*** а соотношения, устанавливающие пределы возможного изменения этих параметров, — ***ограничениями***. Эти ограничения могут быть

заданы в форме равенств или неравенств. Их называют соответственно ***ограничениями типа равенства или ограничениями типа неравенства***.

Если множество параметров оптимизации является подмножеством конечномерного линейного пространства, то говорят о ***конечномерной задаче оптимизации*** в отличие от бесконечномерных задач, которые рассматривают в вариационном исчислении и оптимальном управлении. При этом критерием оптимальности может быть требование достижения наибольшего или наименьшего значения одной или несколькими действительными (скалярными) функциями параметров оптимизации, выражающими количественно меру достижения цели оптимизации рассматриваемого объекта. Каждую из таких функций принято называть ***целевой***. Если целевая функция единственная, то задачу конечномерной оптимизации называют ***задачей математического программирования***, а в противном случае — задачей многокритериальной (векторной) оптимизации. В дальнейшем ограничимся рассмотрением задач математического программирования и методов их решения.

Если целевая функция и ограничения являются линейными относительно параметров оптимизации, то говорят о ***задаче линейного программирования***. Одну из первых таких задач сформулировал и решил Л.В. Канторович. Задача Канторовича была связана с выбором оптимальной производственной программы, что и объясняет появление в названии этого класса задач слова "программирование". При нелинейной зависимости целевой функции или ограничений от параметров оптимизации говорят о ***задаче нелинейного программирования***.

1 Классы задач оптимизации

Как и в любой классификации, разделение *задач оптимизации* на отдельные классы достаточно условно. Отметим, что одна и та же прикладная задача может приводить к разным задачам оптимизации в зависимости от того, какая *математическая модель* используется при рассмотрении реального *объекта оптимизации.* Ясно, что желательно применять более простые модели, но в то же время достаточно полно отражающие свойства объекта, существенные с точки зрения поставленной цели, выраженной в *критерии оптимальности.* Поэтому при выборе либо разработке математической модели или же при обосновании ее упрощения необходимо достаточно четко представлять, к какому классу будет относиться поставленная задача оптимизации, и какие методы применимы для ее решения.

Пусть $f_0(x)$ — *целевая функция,* количественно выражающая некоторый критерий оптимальности и зависящая от координат x_j, $j=\overline{1,n}$, точки $x \in R^n$. Эти координаты являются *параметрами оптимизации* (иногда их называют также переменными задачи оптимизации или просто ***переменными задачи)***.

При математической формулировке задачи оптимизации целевую функцию выбирают с таким знаком, чтобы решение задачи соответствовало поиску минимума этой функции. Поэтому формулировку ***общей задачи математического программирования*** обычно записывают так:

$$f_0(x) \to min, \qquad x \in \Omega, \tag{1.1}$$

где $\Omega \subset R^n$ — множество возможных *альтернатив,* рассматриваемых при поиске решения задачи.

Любую точку $x \in \Omega$ называют ***допустимым решением*** задачи математического программирования, а само множество — множеством допустимых решений или, короче, ***допустимым множеством.*** Точку $x^* \in \Omega$, в которой функция $f_0(x)$ достигает своего наименьшего значения, называют ***оптимальным решением*** задачи.

При отсутствии ограничений множество Ω совпадает с областью определения $D(f_0) \subset R^n$ целевой функции. Если же рассматриваемые альтернативы должны удовлетворять некоторым ограничениям, то множество допустимых решений сужается.

Задачу (1.1) в дальнейшем будем называть ***задачей минимизации*** целевой функции на множестве Ω, понимая под этим нахождение наименьшего значения функции $f_0(x)$ на Ω, и точек $x \in \Omega$, в которых оно достигается. Но целевая функция может и не достигать на Ω наименьшего значения. Тогда говорят о точной нижней грани $\inf\limits_{x \in \Omega} f_0(x)$ функции $f_0(x)$ на этом множестве и вместо (1.1) используют запись

$$f_0(x) \to \inf, \quad x \in \Omega \tag{1.2}$$

Отличие (1.1) от (1.2) в том, что в первом случае предполагают существование точки $x^* \in \Omega$, в которой целевая функция достигает своего наименьшего значения на множестве Ω, а во втором случае такая точка может и не существовать. Поэтому решение общей задачи математического программирования состоит в том, чтобы в первом случае найти точные (или с некоторой заданной точностью) значения координат x_j, $j = \overline{1,n}$, точки $x \in \Omega$ и значение целевой функции $f_0(x^*) = \min\limits_{x \in \Omega} f_0(x)$, а во втором случае построить такую последовательность $\{x_n\}$ точек $x_n \in \Omega$, которой бы соответствовала последовательность $\{f_0(x_n)\}$, сходящаяся к значению $\inf\limits_{x \in \Omega} f_0(x)$, и вычислить это значение с заданной точностью. Отметим, что в большинстве прикладных задач имеет место первый случай, поэтому использование записи вида (1.2) будем оговаривать особо.

Если $f_0(x)$ — линейная функция, то ее область определения совпадает с R^n. В R^n такую функцию с помощью стандартного скалярного произведения можно представить в виде $f_0(x) = (c, x)$, где $c = (c_1,...,c_n) \in R^n$ — известный вектор. Ясно, что целевая функция $f_0(x) = (c,x) = \sum\limits_{j=1}^{n} c_j x_j$ может достигать наименьшего значения $f_0(x^*)$ на множестве Ω лишь в точках границы $\partial\Omega$ этого множества. Если в задаче нет ограничений, то $\Omega = R^n$ и точная нижняя грань линейной функции равна $-\infty$. Поэтому в случае линейной целевой функции задача оптимизации

$$(c,x) \to \min, x \in R^n \ , \tag{1.3}$$

имеет смысл лишь при наличии ограничений.

В частном случае, когда заданы линейные *ограничения типа равенства*

$$Bx=d, \ x \in R^n \ , \tag{1.4}$$

где $d \in R^k$, B — матрица размера $k \times n$, а параметры оптимизации могут принимать лишь неотрицательные значения, т.е.

$$x_j \geq 0, j = \overline{1,n}, \tag{1.5}$$

соотношения (1.3) - (1.5) составляют ***стандартную задачу линейного программирования***, или задачу *линейного программирования в стандартной форме* (в литературе ее часто называют канонической задачей линейного программирования, или задачей линейного программирования в канонической форме).

Условие (1.5) можно представит в виде $x \in R_*^n$, где R_*^n — декартово произведение n множеств R, неотрицательных действительных чисел, называемое иногда ***неотрицательным ортантом*** размерности n. Если к (1.3-1.5) добавить m ограничений неравенства

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_m, \tag{1.6}$$

где $a_{ij} \in R, i = \overline{1,m}$, то соотношения (1.4 - 1.6) приведут к формулировке ***общей задачи линейного программирования***. При этом ограничения (1.5) могут относиться не ко всем n параметрам оптимизации, а лишь к некоторым из них. При отсутствии ограничений типа равенства соотношения (1.3), (1.5), (1.6) составляют формулировку ***основной задачи линейного программирования*** (иногда ее называют ***естественной задачей линейного программирования***).

Неравенства $x_j \geq a_j, x_j \leq b_j$ и $a_j \leq x_j \leq b_j, j = \overline{1,n}$, называют ***прямыми ограничениями*** на переменные задачи, причем последнее относят к ***двусторонним,*** а первые два — к ***односторонним***. Таким образом, ограничения (1.6) являются прямыми односторонними.

Если при линейных ограничениях минимизируемая целевая функция помимо линейной комбинации вида (1.3) включает положительно определенную квадратичную форму, т.е.

$$(c,x) + \frac{1}{2}(Qx,x) \to \min, \tag{1.7}$$

где Q — положительно определенная матрица порядка *n,* то говорят о ***задаче квадратичного программирования,*** а функцию вида (1.7) называют ***квадратичной***. В случае, когда целевая функция является отношением двух линейных функций, а ограничения линейны, имеем ***задачу дробно-линейного программирования***. Ее формулировка включает ограничения (1.4) - (1.6) и условие минимума функции

$$\frac{(q,x)+\alpha}{(r,x)+\beta} \to \min, \tag{1.8}$$

где $q \in R^n$, $r \in R^n$, $\alpha \in R$ и $\beta \in R$ заданы. Соотношения

$$\begin{cases} f_0(x) = \sum_{j=1}^{n} h_j(x_j) \to \min; \\ g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \leq \gamma_i, i = \overline{1,m}; \\ x_j \geq 0, j = \overline{1,n}; \end{cases} \tag{1.9}$$

где $\gamma_i \in R$ заданы, определяют задачу ***сепарабельного программирования***. В этой задаче целевая функция $f_0(x)$ и функция $g_i(x)$ в левой части ограничений являются суммой функций, каждая из которых зависит только от одного параметра оптимизации. В этом случае функции $f_0(x)$ и $g_i(x)$ называют ***сепарабельными***. В прикладных задачах целевая функция нередко имеет вид

$$y(x) = \sum_{i=1}^{m} c_i p_i(x), \tag{1.10}$$

где $x=(x_1,...,x_n)\in R_+^n, c_i\in R_+$, а R_+^n — декартово произведение n множеств R_+ положительных действительных чисел, называемое иногда ***положительным ортантом***. При этом функции $p_i(x)$ имеют вид

$$p_i(x)=\prod_{i=1}^{n}x_j^{a_{ij}}, \tag{1.11}$$

где $a_{ij}\in R$. Требование положительности коэффициентов $c_i, i=\overline{1,m}$, послужило причиной того, что такой вид целевой функции стал называться ***позиномом*** в отличие от полинома (многочлена), в котором коэффициенты могут быть и неположительными. Кроме того в многочлене показатели степени аргументов являются целыми неотрицательными числами, а в позиноме, благодаря положительности параметров оптимизации выражение $x_j^{a_{ij}}$ определено для любых действительных чисел a_{ij}. Если целевая функция и левые части ограничений типа равенства и (или) неравенства в задаче минимизации являются позиномами, то такую задачу называют ***задачей геометрического программирования.***

В задаче нелинейного программирования ограничения могут быть заданы в неявном виде. Тогда множество Ω возможных альтернатив приходится строить путем количественного анализа математической модели объекта оптимизации. Если ограничения принадлежат к типам равенства и (или) неравенства

$$f_l=0, l=\overline{1,k};\qquad g_i\le 0, i=\overline{1,m}, \tag{1.12}$$

но хотя бы одна из функций $f_l(x)$, $g_i(x)$ или целевая функция не является линейной, то говорят об ***общей задаче нелинейного программирования.*** Ясно, что такая формулировка включает задачи квадратичного, дробно-линейного, сепарабельного и геометрического программирования.

Среди целевых функций достаточно широкий класс составляют *выпуклые функции.* Во многих прикладных задачах оптимизации область допустимых значений параметров оптимизации оказывается *выпуклым множеством.* В такой области целевая функция может сохранять одно и то же направление выпуклости, т.е. выпукла либо вниз (выпукла), либо вверх (вогнута). Например, зависимость эффективности технического устройства от параметров оптимизации является вогнутой функцией. Дело в том, что чем выше технические характеристики устройства, тем труднее добиться его дальнейшего совершенствования и существенного приращения эффективности. Наоборот, целевые функции, выражающие массу, габариты или стоимость технического устройства, по тем же причинам обычно выпуклы.

Аналогичная ситуация характерна и для функций, описывающих экономические системы. Например, рост объема выпускаемой продукции происходит не прямо пропорционально капиталовложениям или количеству используемых ресурсов, а с замедлением, причем это замедление часто тем больше, чем больше объем производства. Это приводит к вогнутости так называемых производственных функций, выражающих зависимость объема

выпускаемой продукции от израсходованных ресурсов. Наоборот, при фиксированном объеме производства дальнейшее снижение производственных затрат и стоимости единицы продукции по сравнению с достигнутым уровнем также происходит с замедлением, что приводит к выпуклости целевых функций, описывающих стоимостные характеристики производства.

Ясно, что любую вогнутую целевую функцию, изменив знак, можно сделать выпуклой. Задачи оптимизации, в которых необходимо найти наименьшее значение выпуклой целевой функции, рассматриваемой на выпуклом множестве, относят к ***задачам выпуклого программирования***. Частными случаями таких задач являются задачи квадратичного и линейного программирования. Задачи геометрического программирования при некоторых дополнительных условиях также являются задачами выпуклого программирования.

Если множество Ω допустимых решений оказывается конечным множеством, то мы имеем ***задачу дискретного программирования,*** а если к тому же координаты этих точек — целые числа, то — ***задачу целочисленного программирования***.

2 Необходимые и достаточные условия минимума гладких функций

Математическая формулировка оптимальной задачи часто эквивалентна задаче отыскания экстремума функции одной или многих независимых переменных. Поэтому для решения таких оптимальных задач могут быть использованы различные методы исследования функций классического анализа и главным образом методы поиска экстремума.

Из аналитических методов внимание в основном уделено методам отыскания *безусловных экстремумов.*

Методы исследования функций классического анализа в основном применяют в тех случаях, когда известен аналитический вид зависимости оптимизируемой функции R от независимых переменных x_i. Это позволяет найти также в аналитическом виде производные оптимизируемой функции, используя которые и формулируют необходимые и достаточные условия существования экстремума.

2.1 Экстремумы функций одной переменной

Необходимые условия существования экстремума к непрерывной функции *R (x)* при отсутствии ограничений на диапазон изменения переменной *x* могут быть получены из анализа первой производной $\frac{dR}{dx}$. При этом функция *R (x)* может иметь экстремальные значения

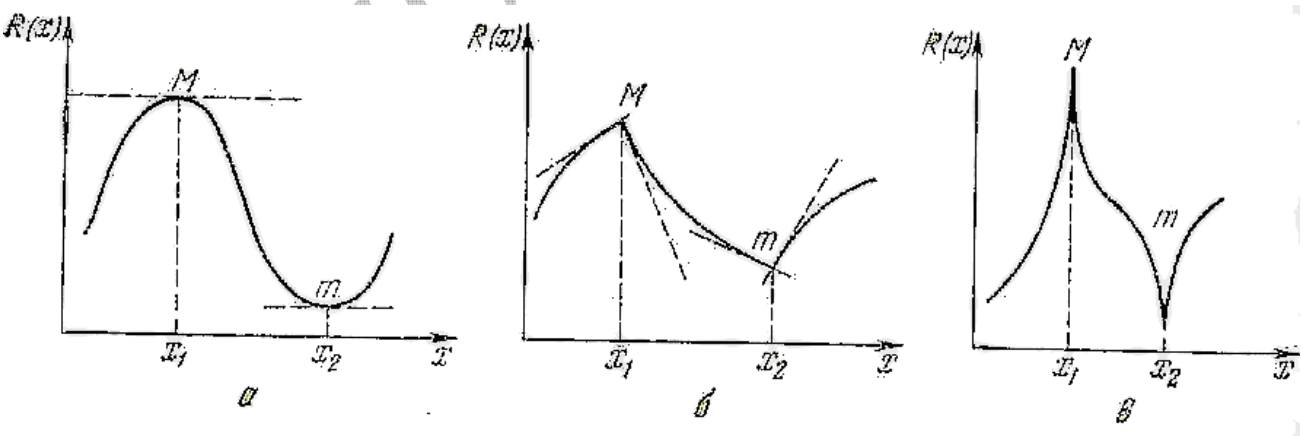


Рисунок 2.1 — Экстремальные точки функции *R (x)*

независимой переменной x или, что то - же самое, в тех точках оси x, где производная равна нулю, либо вообще не существует. Графически равенство нулю производной означает, что касательная к кривой *R (x)* в этой точке параллельна оси абсцисс (рисунок 2.1, а). На рисунке 2.1, *б* и рисунке 2.1, *в,*

показаны случаи, когда производные в точках экстремума функции *R (x)* не существуют. Так, на рисунке 2.1, *б* изображена непрерывная функция *R (x)*, которая в точках x_1 и x_2 имеет изломы, что соответствует наличию конечного разрыва у производной $\frac{dR}{dx}$ в этих точках. При этом, очевидно, вычисляемое значение производной зависти от того, с какой стороны от экстремальной точки будет производиться вычисление. В таких случаях принято говорить о существовании различных значений производных слева и справа от точки экстремума. На рисунке 2.1, *в*, показан также вариант, когда значение производной в точках экстремума обращается в бесконечность. Здесь происходит бесконечный разрыв производной, при котором ее значение изменяется от $+\infty$ до $-\infty$ в точке x_1 и от $-\infty$ до $+\infty$ в точке x_2.

Перечисленные условия, т.е. равенство нулю или отсутствие производной, являются, как уже отмечалось выше, только необходимым условиями экстремума. Их выполнение еще не означает, что в данной точке функция имеет экстремум (рисунок 2.2).

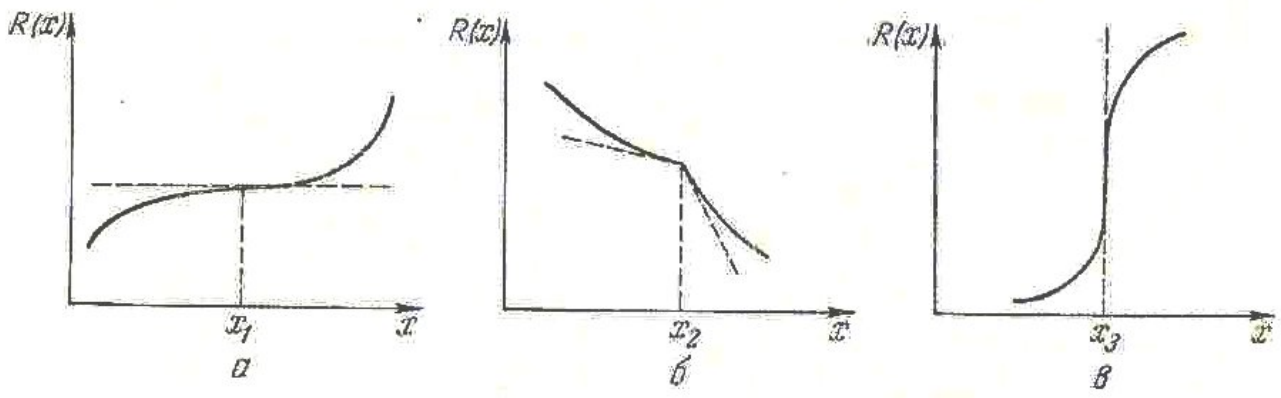


Рисунок 2.2 — Отсутствие экстремума у функции *R (x)*

Для этого чтобы определить, действительно ли в указанной точке существует экстремум или же при этом справедлив один из случаев, представленных на рисунке 2.2, необходимо провести дополнительное исследование, для чего может быть использован один из способов, приведенных ниже.

Сравнение значений функции. Этот способ сводится к тому, что со значением в точке x_k, "подозреваемой" на экстремум, сравнивают два ее значения, рассчитанные в точках, достаточно близких к исследуемой и расположенных слева и справа от нее, т.е. при значениях переменной $x_k - \varepsilon$ и $x_k + \varepsilon$, где ε – малая положительная величина (рисунок 2.3). Если при этом окажется, что оба рассчитанных значения $R(x_k - \varepsilon)$ и $R(x_k + \varepsilon)$ меньше или больше $R(x_k)$, то в точке x_k существует максимум или минимум соответственно (рисунок 2.3, *а* и рисунок 2.3, *б*). Если же $R(x_k)$ имеет промежуточное значение между $R(x_k - \varepsilon)$ и $R(x_k + \varepsilon)$, то в точке x_k функция $R(x)$ не обладает ни максимумом, ни минимумом (рисунок 2.3, *в*).

Сравнение знаков производных. При этом способе исследования точки x_k, "подозреваемой" на экстремум, в точках $x_k - \varepsilon$ и $x_k + \varepsilon$ определяется знак производной $\frac{dR}{dx}$. Если знаки производной в этих точках различны, то в точке x_k имеется экстремум функции $R(x)$. Тип экстремума может быть найден по изменению знака производной при переходе от точки $x_k - \varepsilon$ к точке $x_k + \varepsilon$. Если знак производной $\frac{dR}{dx}$ при таком переходе меняется с (+) на (–), то в точке x_k – максимум (см. рисунок 2.3, *а*), если наоборот – с (–) на (+), то – минимум (см. рисунок 2.3, *б*). Если же знаки производной в точках $x_k - \varepsilon$ и $x_k + \varepsilon$ совпадают, то в точке x_k нет ни максимума, ни минимума, т. е. она не является экстремальной (см. рисунок 2.3, *в*).

Исследование знаков высших производных. Этот способ можно применять в тех случаях, когда в точке, "подозреваемой" на экстремум, существуют производные высших порядков, т.е. функция $R(x)$ не только сама непрерывна, но имеет также непрерывные производные $\frac{dR}{dx}$ и $\frac{d^2R}{dx^2}$, а в некоторых случаях и более высокого порядка. Способ сводится к следующему. В точке x_k, "подозреваемой" на экстремум, для которой справедливо

$$\left.\frac{dR(x)}{dx}\right|_{x=x_k} = 0 \qquad (2.1)$$

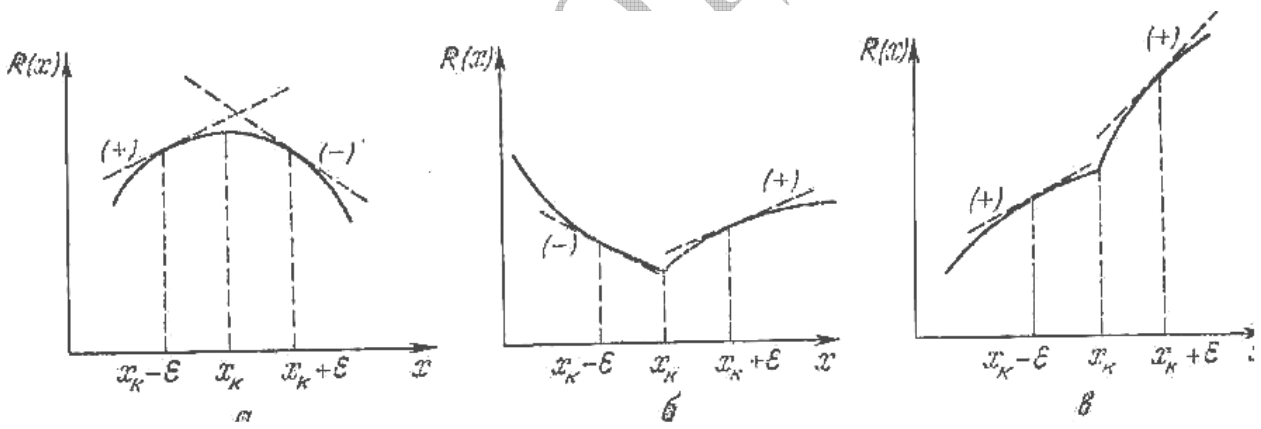


Рисунок 2.3 — Способы проверки на достаточность

вычисляется значение производной $\frac{d^2R}{dx^2}$. Если при этом $\left.d^2R/dx^2\right|_{x=x_k} < 0$, то в точке x_k – максимум, если $\left.d^2R/dx^2\right|_{x=x_k} > 0$, то–минимум. Эти два случая изображены на рисунке 2.4, *а* и 2.4, *б* соответственно. На этих же рисунках показаны графики для производных $\frac{dR}{dx}$ и $\frac{d^2R}{dx^2}$. Очевидно,

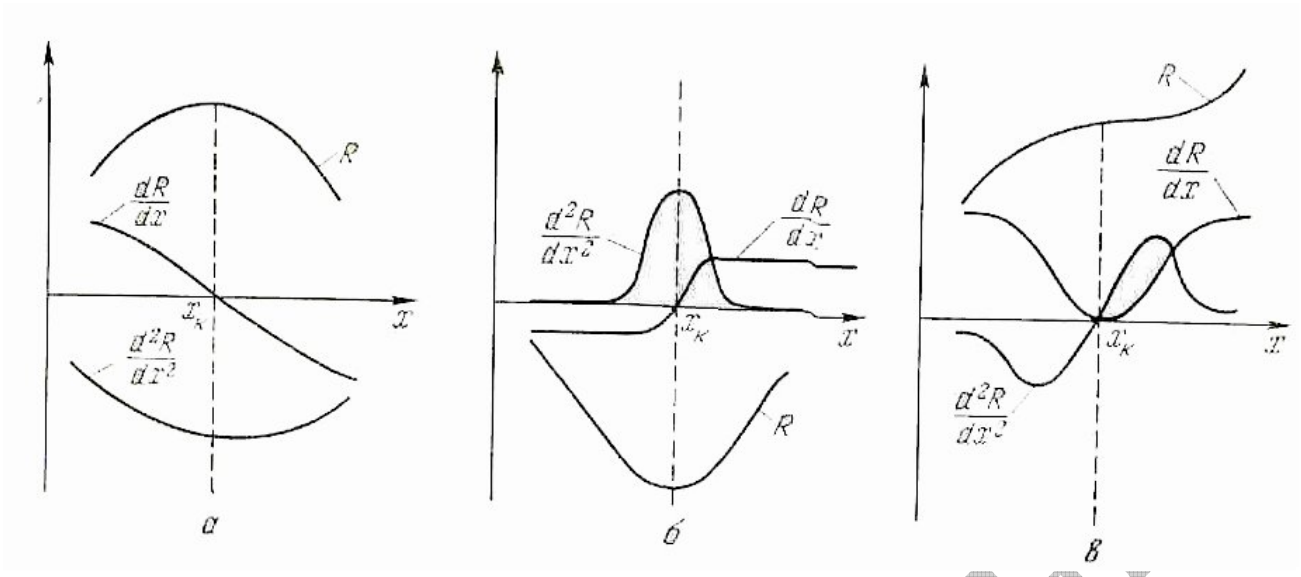


Рисунок 2.4 — Исследование знаков высших производных

что отрицательность производной $d^2R/dx^2\big|_{x=x_k}$ означает, что в точке x_k знак первой производной $\frac{dR}{dx}$ изменяется с (+) на (–), так как в этом случае при увеличении *x* производная $\frac{dR}{dx}$ убывает, проходя в точке x_k через нулевое значение, и наоборот.

Если же вторая производная в точке x_k также равна нулю, то для дальнейшего исследования необходимо вычисление следующей производной $\frac{d^3R}{dx^3}$. Причем, когда $d^3R/dx^3\big|_{x=x_k} \neq 0$, точка x_k не является точкой экстремума (рисунок 2.4, *в*). Если же в этой точке $d^3R/dx^3\big|_{x=x_k}$ =0, то вычисляют следующую производную $\frac{d^4R}{dx^4}$ и т. д. При этом нужно руководствоваться таким правилом:

когда порядок первой, не обращающейся в нуль производной в точке x_k, *для которой* $\frac{dR(x)}{dx}\Big|_{x=x_k}$=0, *нечетный, то в этой точке функция R(*x*) не имеет ни максимума, ни минимума, т.е. точка* x_k *не является точкой экстремума функции R(*x*) . Если же порядок первой, не обращающейся в нуль производной в точке* x_k *четный, то в данной точке есть экстремум функции R(*x*), который будет максимумом или минимумом в зависимости от того, отрицательна или положительна эта производная.*

Способ исследования знаков высших производных может потребовать довольно громоздких вычислений для определения в аналитическом виде производных высших порядков. Поэтому иногда значительно проще воспользоваться одним из приведенных выше двух способов.

При решении практических задач оптимизации обычно требуется отыскать не какой-нибудь экстремум (максимум или минимум) функции *R(*x*),* а наибольшее или наименьшее значение этой функции. Точки наибольшего или наименьшего значения функции *R(*x*)* обычно называют ***глобальными***

экстремумами в отличие от остальных экстремумов, которые в подобных случаях называют ***локальными***.

Для решения оптимальной задачи необходимо, во-первых, найти все точки функции *R(*x*),*в которых может быть экстремум, во-вторых, исследовать все эти точки на экстремум и, наконец, в-третьих, среди локальных экстремумов нужного типа (максимум или минимум) найти глобальный. Подобный случай представлен на рисунке 2.5, где функция *R(*x*)* имеет пять точек, "подозреваемых" на экстремум, из которых лишь четыре являются точками локальных экстремумов. Если число локальных экстремумов оптимизируемой функции велико, то может потребоваться весьма большой объем вычислений, необходимый для проверки условий экстремальности. При этом оказывается полезным следующее правило, позволяющее уменьшить число проверяемых точек. *Для непрерывных функций одной переменной максимумы и минимумы чередуются между собой; между двумя соседними максимумами расположен один минимум, а между двумя соседними минимумами – один максимум.* При использовании этого правила достаточно "подозрительные" точки поверять через одну (рисунок 2.6), что позволяет установить вид всех экстремальных точек функции *R(*x*)*.

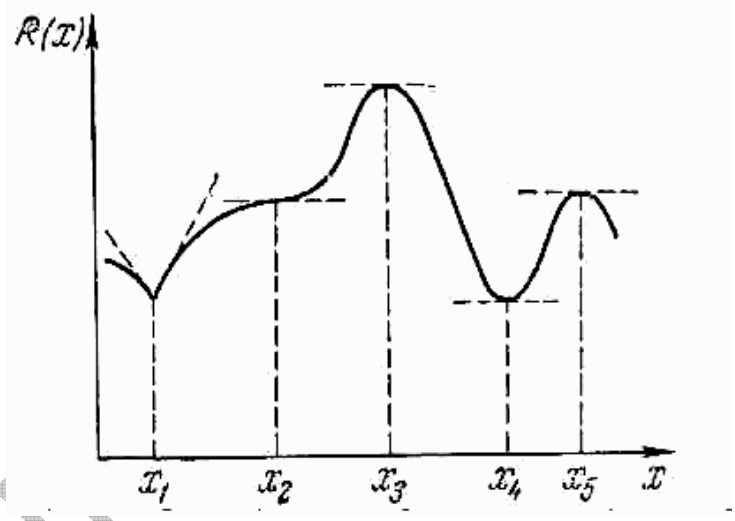


Рисунок 2.5 — Локальные экстремумы функции *R(*x*)*

Порядок рассуждений для функции *R(*x*)*, представленной на рисунке 2.6, следующий. Так как в точке x_1 – максимум, а в точке x_3 – минимум, то в точке x_2 экстремума нет. В точке x_5 нет ни максимума, ни минимума, поэтому дальнейшую проверку продолжаем с точки x_6, в которой оказывается минимум. Следовательно, в пропущенной точке x_4 может быть только максимум, поскольку два минимума не могут следовать подряд. В точке x_8 снова оказывается максимум, и, значит, в точке x_7 нет ни максимума, ни минимума. Таким образом, общее число проверенных точек в данном случае составило пять точек из восьми, подлежащих проверке. Правда, при таком способе проверки еще нельзя найти глобальный экстремум, так как величины пропущенных локальных экстремумов остаются неопределенными.

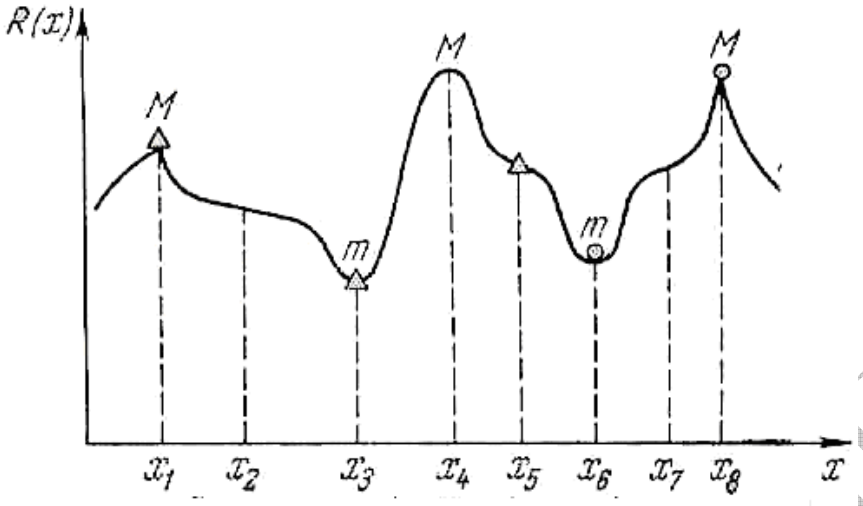


Рисунок 2.6 — Проверка точек на экстремум

Однако уже одно то, что при использовании этого правила, дополнительные вычисления требуются только для локальных экстремумов определенного типа, оправдывает его применение.

В практических задачах оптимизации диапазон изменения независимой переменной *x* часто бывает ограничен заданным интервалом [*a,b*]. Приведенную же выше методику поиска экстремальных точек, основанной на анализе первой производной оптимизируемой функции, можно использовать лишь для внутренних точек интервала [*a,b*]. Поэтому в число “подозрительных” точек должны быть включены также и крайние точки интервала [*a,b*], т.е. точки *a и b* ,в которых может иногда достигаться глобальный экстремум для функции *R(*x*)* (рисунок 2.7).

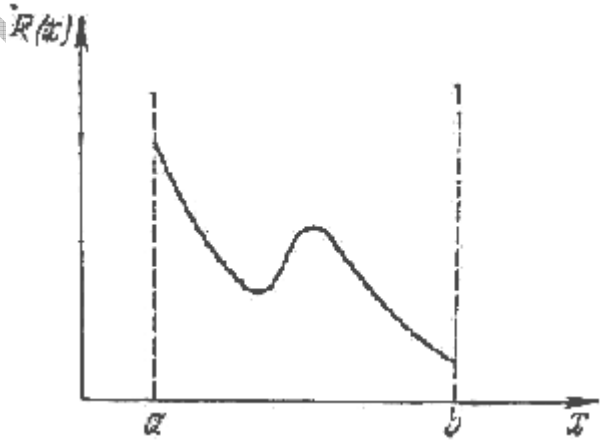


Рисунок 2.7 —Наличие экстремума на границах интервала

2.2 Экстремумы функций многих переменных

Решение задачи оптимизации существенно усложняется, когда критерий оптимальности является функцией нескольких независимых переменных даже при известном аналитическом выражении этой функции. Наибольшие трудности возникают при отсутствии непрерывности у всех или некоторых производных оптимизируемой функции. В последнем случае для решения оптимальной задачи целесообразно использовать методы нелинейного программирования.

Ниже рассмотрены необходимые и достаточные условия лишь для непрерывных функций, имеющих к тому же непрерывные производные первого и второго порядков.

Для непрерывной функции многих переменных

$$R=R(\mathrm{x}_1, \mathrm{x}_2, \ \ldots\ , \mathrm{x}_n), \tag{2.2}$$

имеющей непрерывные производные первого и второго порядков по все переменным x_i *(i=1, ... ,n),* необходимым условием экстремума в точке $\mathrm{x}^{(k)}_i$ *(i=1, ... ,n)* служит равенство нулю в этой точке частных производных по всем переменным. Другими словами, точки, в которых возможен экстремум функции (II,2), могут быть определены решением системы уравнений:

$$\frac{dR(\mathrm{x}_1,\mathrm{x}_2,\ldots,\mathrm{x}_n)}{dx_i}=0 \qquad i=1, \ldots ,n \tag{2.3}$$

Для того, чтобы проверить, действительно ли точка $\mathrm{x}^{(k)}_i$ *(i=1, ... ,n),* координаты которой удовлетворяют системе уравнений (2.3), является точкой экстремума функции (2.2), уже недостаточно проверки экстремальности по всем переменным в отдельности.

Предположим, что исследуется на экстремум точка с координатами

$$x^{(k)}=(\mathrm{x}_1^{(k)},\mathrm{x}_2^{(k)},\ldots,\mathrm{x}_n^{(k)}), \tag{2.4}$$

удовлетворяющими системе уравнений (2.3).

Разложим функцию

$$R(x)=R(\mathrm{x}_1,\mathrm{x}_2,\ldots,\mathrm{x}_n) \tag{2.5}$$

в окрестности точки (2.4) в ряд Тейлора по степеням приращений переменных

$$\delta x_i = x_i - x_i^k \tag{2.6}$$

тогда получим:

$$R(x)=R(x^k)+\sum_{i=1}^{n}\frac{\partial R(x^k)}{\partial x_i}\partial x_i+\frac{1}{2}[\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\partial x_i]^2\cdot R(x^k)+\frac{1}{6}[\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\partial x_i]^3\cdot R(x^k)+\ldots \tag{2.7}$$

В записи выражения (2.7) использована сокращенная форма представления членов разложения с производными выше первого порядка. Например, для функции двух переменных член

$$\frac{1}{2}[\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\partial x_i]^2 \cdot R(x^k)$$

раскрывается как

$$\frac{1}{2}[\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\partial x_i]^2 \cdot R(x^{(k)}) =$$

$$[\frac{\partial}{\partial x_1}\partial x_1 + \frac{\partial}{\partial x_2}\partial x_2]^2 R(x^{(k)}) = [\frac{\partial^2}{\partial x_1^2}\partial x_1^2 + 2\frac{\partial^2}{\partial x_1 \partial x_2}\partial x_1 \partial x_2 + \frac{\partial^2}{\partial x_2^2}\partial x_2^2]R(x^{(k)}) =$$

$$\frac{\partial^2 R(x^{(k)})}{\partial x_1^2}\partial x_1^2 + 2\frac{\partial^2 R(x^{(k)})}{\partial x_1 \partial x_2}\partial x_1 \partial x_2 + \frac{\partial^2 R(x^{(k)})}{\partial x_2^2}\partial x_2^2 \qquad (2.8)$$

Аналогично раскрываются и члены более высокого порядка при большем числе независимых переменных. Опуская в разложении (2.7) члены, имеющие порядок малости по ∂x_i обращаются в нуль, поскольку координаты точки $x^{(k)}$ удовлетворяют системе (2.3), получим следующее приближенное равенство:

$$\delta R = R(x) - R(x^{(k)}) \approx \frac{1}{2}[\sum_{i=1}^{n}\frac{\partial}{\partial x_i}\partial x_i]^2 \cdot R(x^{(k)}) \qquad (2.9)$$

Из выражения (2.9) следует, что знак приращения функции δR в достаточно малой окрестности точки $x^{(k)}$ определяется производными второго порядка от *R(x)* по всем переменным, включая и смешанные производные. Для того, чтобы точка $x^{(k)}$ являлась точкой экстремума функции *R(x)*, достаточно при любых малых приращениях независимых переменных ∂x_i правой части выражения (2.9) оставаться положительной для точки минимума и отрицательной для максимума.

Поскольку вторые производные в выражении (2.9) вычисляются в точке $x^{(k)}$, они могут рассматриваться как постоянные числа. В этом случае для анализа знака правой части выражения (2.9) не обязательно требовать малости величин ∂x_i. Таким образом, вопрос о знаке приращения функции ΔR может решаться анализом знака квадратичной формы

$$U = \sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij} z_i z_j \,, \qquad (2.10)$$

коэффициенты которой связаны с производными в правой части выражения (2.9) соотношениями

$$a_{ij} = \frac{\partial^2 R(x^{(k)})}{\partial x_i \partial x_j} = \frac{\partial^2 R(x^{(k)})}{\partial x_j \partial x_i} = a_{ij} \qquad i,j=1, \ldots ,n \qquad (2.11)$$

Квадратичная форма называется положительно определенной, если для любых значений z_p она сохраняет положительное значение, за исключением точки $z_p = 0\ (p=1, \ldots ,n)$, в которой она обращается в нуль. Для того, чтобы найти, является ли данная квадратичная форма положительно определенной, можно воспользоваться теоремой, которая формулируется следующим образом. *Для*

положительной определенности квадратичной формы (2.10) необходимо и достаточно, чтобы были выполнены условия Сильвестра:

$$\Delta_1 = a_{11} > 0;\ \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0;\ \ldots$$

$$\Delta_n = \begin{vmatrix} a_{11} & a_{12} & . & a_{1n} \\ a_{11} & a_{22} & . & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & . & a_{nn} \end{vmatrix} > 0 \qquad (2.12)$$

Условия (2.12) означают, что квадратичная форма (2.10) будет положительно определенной в том случае, если все главные миноры соответствующей ей матрицы

$$[A] = \begin{vmatrix} a_{11} & a_{12} & . & a_{1n} \\ a_{11} & a_{22} & . & a_{2n} \\ . & . & . & . \\ a_{n1} & a_{n2} & . & a_{nn} \end{vmatrix}, \qquad (2.13)$$

составленной из коэффициентов, будут строго положительны.

Таким образом, если решен вопрос о положительной определенности квадратичной формы (2.10), где коэффициенты рассчитываются по формулам (2.11), то тем самым решается задача о типе точки $x^{(k)}$, координаты которой удовлетворяют системе уравнений (2.3), исследуемой на экстремум.

Так, когда квадратичная форма, соответствующая правой части выражения (2.9), оказывается положительно определенной, исследуемая точка является точкой минимума.

Если условия положительной определенности не выполняются, но все главные миноры матрицы (2.13), имеющие нечетный порядок, отрицательны, т.е. для миноров нечетного порядка в неравенствах (2.12) знак неравенств изменяется на обратный, то квадратичная форма (2.10) будет ***отрицательно определенной*** и, следовательно, функция *R(x)* в точке $x^{(k)}$ имеет максимум.

Если же условия положительной и отрицательной определенностей квадратичной формы (2.10) не выполняются, но все главные миноры отличны от нуля, то в исследуемой точке $x^{(k)}$ функция *R(x)* не имеет ни максимума, ни минимума. При обращении в нуль главных миноров матрицы (2.13) вопрос о наличии экстремума в исследуемой точке решается сложнее, с использованием производных более высокого порядка.

3 Методы одномерной минимизации

3.1 Предварительные замечания

В некоторых случаях *ограничения* в *задаче оптимизации* позволяют через один из *параметров оптимизации* выразить остальные и исключить их из *целевой функции.* В результате задача будет сведена к поиску наибольшего или наименьшего (в зависимости от цели оптимизации) значения скалярной действительной функции $f(x), x \in D(f) \subset R$, выражающей *критерий оптимальности.* Выбирая тот или иной знак перед этой функцией, всегда можно ограничиться лишь поиском ее наименьшего значения в области определения *D(f),* заданной с учетом ограничений на параметр оптимизации *x*. Поэтому далее в этой главе будем рассматривать задачу

$$f(x) \to \min, x \in D(f) \subset R \tag{3.1}$$

поиска наименьшего значения $f_* = f_*(x)$ функции *f(x)* и точки $x_* \in D(f)$, в которой *f(x)* принимает это значение. Для краткости будем говорить об ***одномерной минимизации***, имея в виду нахождение наименьшего значения функции *f(x)* на множестве *D(f)* и точек, в которых это значение достигается.

Изучение методов одномерной минимизации важно не только для решения задачи (3.1), имеющей самостоятельное значение. Эти методы являются также существенной составной частью методов многомерной минимизации, при помощи которых находят наименьшее значение действительных функций многих переменных.

Пусть область определения *D(f)* функции *f(x)* есть промежуток числовой прямой. Напомним, что если *D(f)* — отрезок и *f(x)* непрерывна на нем, то она имеет на этом отрезке наименьшее значение. Но при наличии на отрезке точек разрыва функции она может не иметь на нем наименьшего значения. Оно может не существовать и в том случае, когда *D(f)* является интервалом или полуинтервалом.

Если функция *f(x)* не имеет на множестве *D(f)* наименьшего значения, то (3.1) следует заменить формулировкой задачи в виде

$$f(x) \to \inf, x \in D(f) \subset R\,. \tag{3.2}$$

Тогда под решением задачи минимизации такой функции на *D(f)* следует понимать построение последовательности $\{x_n\}$ точек из *D(f)*, для которой существует предел

$$\lim_{n\to\infty} f(x_n) = \inf_{x_n \in D(f)} f(x) = \tilde{f}_*, \tag{3.3}$$

и нахождение этого предела. Если функция *f(x)* достигает на множестве *D(f)* своего наименьшего значения f_* то $\tilde{f}_* = f_*$.

Например, функция *f(x) = 1/x* на множестве *D(f)* = [1,2) не достигает наименьшего значения, хотя и ограничена снизу. Точная нижняя грань $\tilde{f}_*$ функции в данном случае равна 1/2. В качестве последовательности $\{ x_n \}$ точек в полуинтервале [1,2), для которой справедливо (3.3), можно выбрать {2 – 1/n}. Тогда

$$f(x_n) = \frac{1}{x_n} = \frac{1}{2 - 1/n} = \frac{n}{2n-1},$$

и последовательность *{f{x$_n$)}* сходится к числу 1/2 = $\tilde{f}_*$.

Функция может достигать наименьшего значения как в единственной точке, так и на некотором множестве точек, конечном, счетном или несчетном. Например, функция $f(x) = x^4$ определена на всей числовой прямой и достигает своего наименьшего значения $f_* = 0$ в единственной точке $x_* = 0$, которая является ее точкой минимума. Функция $f(x) = x^4 - 2x^2 + 2$ также определена на всей числовой прямой и достигает наименьшего значения $f_* = 1$ в точках $x_* = \pm 1$. Функция $f(x) = \cos(x)$ достигает наименьшего значения на счетном множестве $D_* = \{x \in R : x = \pi + 2\pi k, k \in Z\}$, а функция $f(x) = |x+1| + |x-1|$ – на счетном множестве $D_* = [-1,1]$.

Функцию *f(x)* называют ***унимодальной функцией*** на отрезке [a,b], если существует такая точка $x_* \in [a,b]$, что функция $f(x)$ в полуинтервале $[a, x_*)$ убывает, а в полуинтервале $(x_*, b]$ возрастает. Примеры графиков унимодальных функций приведены на рисунке 3.1.

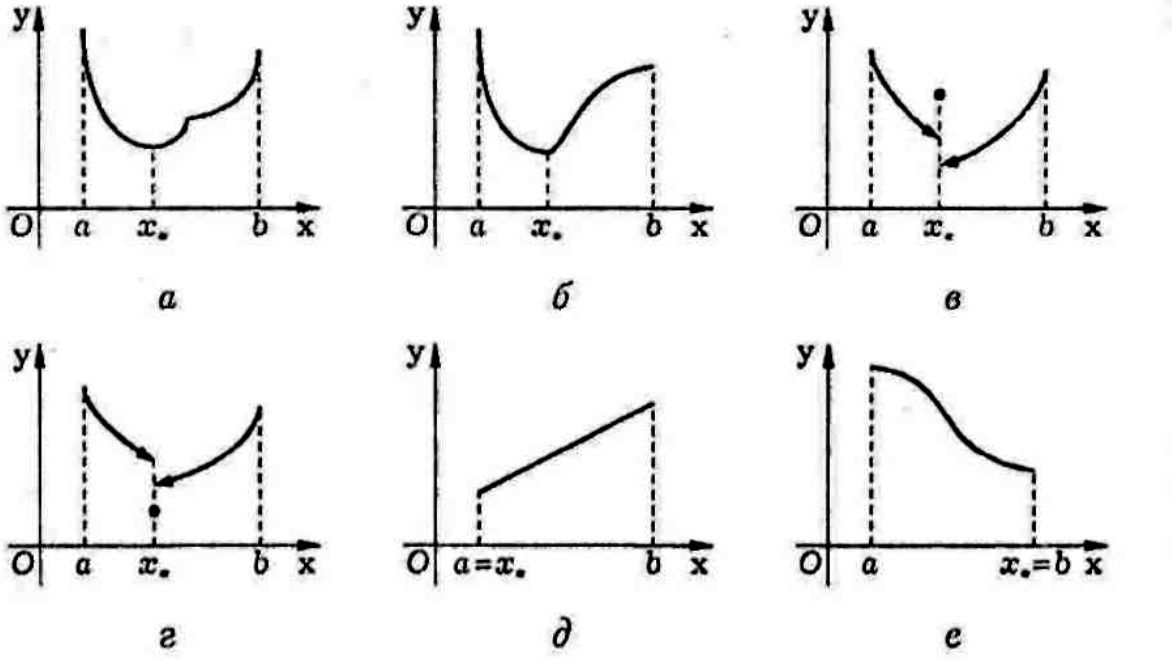


Рисунок 3.1 — Унимодальные функции

Точка x_* может быть внутренней точкой отрезка [a, b] (т.е. $a < x_* < b$, см. рисунок 3.1, *а-г*) или совпадать с одним из его концов *(*$x_* = a$ или $x_* = b$, см.

рисунок 3.1, *д, е)*. Унимодальная функция не обязательно непрерывна на отрезке [a, b] (см. рисунок 3.1, *в, г*).

Функцию *f(x)*, достигающую на отрезке [a, b] наименьшего значения в единственной точке $x_* \in [a,b]$, убывающую при $x \in [a,x_*]$ и возрастающую при $x \in [x_*,b]$ будем называть ***строго унимодальной*** на отрезке [a, b] (на рисунке 3.1 строго унимодальными являются все функции, кроме функции на рисунке 3.1. *г)*.

Область определения *D(f)* минимизируемой функции $f(x)$ может состоять из нескольких промежутков, не имеющих общих граничных точек. В этом случае, чтобы найти наименьшее значение функции на множестве *D(f),* достаточно определить наименьшее значение функции в каждом из промежутков, составляющих *D(f),* а затем, сравнивая, выбрать среди этих значений минимальное.

Если функция дифференцируема в промежутке, то возможно использование необходимого и достаточного условий локального минимума. Однако в прикладных задачах нередки ситуации, когда трудно вычислить производные функции (например, если функция не задана в аналитическом виде). Более того, не исключено, что значения функции известны или могут быть вычислены только в отдельных точках. В таких ситуациях использование необходимого и достаточного условий локального минимума невозможно и следует применять другие методы решения задачи оптимизации. Методы минимизации функции одного переменного, в которых используют значения функции в точках рассматриваемого промежутка и не используют значения ее производных, называют ***методами прямого поиска***.

3.2 Пассивный и последовательный поиск

Пусть требуется найти наименьшее значение или точную нижнюю грань f_* скалярной действительной функции $f(x)$ одного переменного на отрезке [a, b]. Предположим, что задан алгоритм вычисления значения функции для любой точки $x \in [a,b]$. Можно выделить две группы *методов прямого поиска,* соответствующие двум принципиально различным ситуациям:

1) все N точек $x_k, k = \overline{1,N},$ в которых будут вычислены значения функции, выбирают заранее (до вычисления функции в этих точках);

2) точки x_k выбирают последовательно (для выбора последующей точки используют значения функции, вычисленные в предыдущих точках).

В первом случае поиск значения f_* называют ***пассивным,*** а во втором — ***последовательным***. Естественно ожидать, что последовательный поиск лучше пассивного. В этом можно убедиться, вспомнив детскую игру, в которой надо найти спрятанную вещь, задавая вопросы и получая на них ответы „да” или „нет”. Задавая вопросы последовательно с учетом предыдущих ответов, можно найти спрятанную вещь за меньшее число вопросов (итераций), чем, задав определенное количество заранее подготовленных вопросов сразу.

Так как в прикладных задачах вычисление каждого значения функции может быть достаточно трудоемким, то целесообразно выбрать такую стратегию поиска, чтобы значение f_* с заданной точностью было найдено наиболее экономным путем. Будем считать, что стратегия поиска определена, если:

- определен алгоритм выбора точек $x_k, k = \overline{1, N},$;

- определено условие прекращения поиска, т.е. условие, при выполнении которого значение f_* считают найденным с заданной точностью.

Для методов пассивного поиска алгоритм выбора точек $x_k, k = \overline{1, N},$ — это правило, по которому заранее определяют все N точек $x_k, k = \overline{1, N},$, в которых затем будут вычислены значения функции $f(x)$. Для методов последовательного поиска алгоритм выбора точек x_k — это правило, по которому последовательно определяют каждую следующую точку x_k по информации о расположении точек $x_i, i = \overline{1, k-1},$ и о вычисленных значениях $f(x_i)$ функции *f(x)* в этих точках. Выбор очередной точки x_k и вычисление значения $f(x_k)$ называют ***шагом*** последовательного ***поиска***.

В методах последовательного поиска количество точек x_k обычно не задают заранее. Однако объективное сравнение различных методов прямого поиска нужно проводить при одинаковом количестве n вычисленных значений функции *f(x)*. После n вычислений обычно указывают интервал (или отрезок) длины l_n, называемый ***интервалом неопределенности***, в котором гарантированно находится точка x_*, соответствующая значению f_*. Условие прекращения вычислений в случае пассивного или последовательного поиска примем одинаковым — выполнение неравенства $l_n \le \varepsilon$, где ε_* — заданная наибольшая допустимая длина интервала неопределенности.

Длина l_n зависит как от самого метода прямого поиска P, так и от минимизируемой функции $f(x)$, т.е. $l_n = l_n(P,f)$. Зависимость l_n от n дает оценку скорости сходимости конкретного метода прямого поиска P к искомому значению f_* заданной функции $f(x)$. Различные методы из некоторого множества ρ методов прямого поиска сравнивают обычно при выбранном фиксированном значении $n = N$ на некотором достаточно широком классе функций. В качестве такого класса можно выбрать множество F *унимодальных функций,* определенных на фиксированном отрезке $X \subset R$. Для метода прямого поиска $P \in \rho$ примем наихудшую оценку

$$l_N(P) = \max_{f \in F} l_N(P, f).$$

Если „наихудшей” унимодальной функции не найдется, то оценку принимаем в виде

$$l_N(P) = \sup_{f \in F} l_N(P, f).$$

Значение $l_N(P)$ представляет собой оценку сверху погрешности вычисления точки $x_* \in X$, соответствующей искомому значению f_* произвольной функции $f \in F$, которая получена методом прямого поиска $P \in \rho$ по N вычисленным значениям этой функции. Метод прямого поиска P^* считаем наилучшим, если

$$l_N(P) = \min_{P \in \rho} \max_{f \in F} l_N(P, f), \text{ или } l_N(P) = \min_{P \in \rho} \sup_{f \in F} l_N(P, f).$$

Этот критерий сравнения методов поиска определяет ***минимаксный метод поиска***. Такой метод является наилучшим для всего множества F унимодальных функций на отрезке $X \subset R$ в том смысле, что он дает наименьшую погрешность вычисления точки x_*, соответствующей значению f_* любой из рассматриваемых функций $f \in F$. Хотя вполне возможно, что существует некоторый конкретный метод, который для определенной, специально подобранной унимодальной функции из множества F обеспечит еще меньшую погрешность.

Все методы прямого поиска можно строить и сравнивать между собой на отрезке X=[0, 1]. Полученные результаты при необходимости нетрудно перенести на случай произвольного отрезка [a, b], так как любую точку отрезка [0,1] можно перевести в соответствующую ей точку отрезка [a, b] растяжением в $b - a$ раз и сдвигом на a.

Если минимизируемая функция $f(x)$ не является унимодальной на отрезке [a, b] (такую функцию называют ***мультимодальной функцией*** на этом отрезке), то, даже если она непрерывна на [a, b]), при поиске наименьшего значения f_* функции на отрезке может возникнуть ошибка: будет найдена точка локального минимума, в которой значение функции не f_*, а другое, большее. Чтобы избежать такой ошибки, в процесс минимизации включают предварительный этап, на котором отрезок минимизации разделяют на несколько отрезков, на каждом из которых минимизируемая функция унимодальна. Сравнительный анализ наименьших значений функции на этих отрезках позволяет найти искомое наименьшее значение f_* на всем отрезке минимизации.

3.3 Оптимальный пассивный поиск

Пусть требуется путем *пассивного поиска* найти точку $x_* \in$ [0,1], в которой *унимодальная* на отрезке [0,1] *функция f(x)* достигает наименьшего значения $f_* = f(x_*)$. *Минимаксный метод поиска,* в котором информация о значениях функции, вычисленных в предшествующих точках, не может быть использована, называют *оптимальным пассивным поиском.* Рассмотрим алгоритм такого поиска при различном числе N точек, выбираемых на отрезке [0, 1].

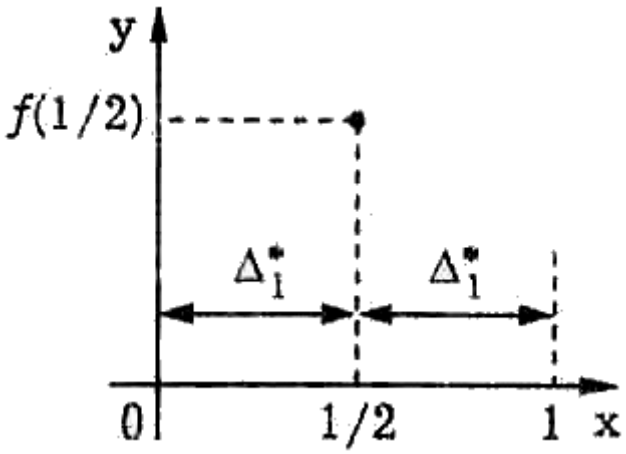


Рисунок 3.2 — Оптимальный пассивный поиск при *N=1*.

Если *N=1,* то единственную точку целесообразно выбрать в середине отрезка, т.е. принять $x_1 = 1/2$ (рисунок 3.2). В этом случае вследствие унимодальности функции *f(x)* имеем $f_* \le f(1/2)$. Поэтому наименьшая возможная длина *интервала неопределенности* равна $l_1^* = 1$ и можно гарантировать, что выбор в качестве точки $x_* \in [0,1]$ точки $x_1 = 1/2$ приведет к погрешности не более $\Delta_1^* = l_1^*/2 = 1/2$. При любом ином положении точки x_1 погрешность при выборе $x_* = x_1$ будет $\Delta_1 > \Delta_1^*$, так как в действительности точка x_* может лежать на большей части отрезка [0,1].

Если при *N = 2* (рисунок 3.3) две точки расположить на отрезке [0,1] так, чтобы они делили его на равные части, т.е. выбрать $x_1 = 1/3$ и $x_2 = 2/3$, то точка $x_* \in [0,1]$ будет найдена с точностью $\Delta_2^* = 1/3$, а наименьшая длина интервала неопределенности составит $l_2^* = 2\Delta_2^* = 2/3$. В самом деле, если $f(1/3) < f(2/3)$ (рисунок 3.3, *а*), то в силу унимодальности функции *f(x)* отрезок

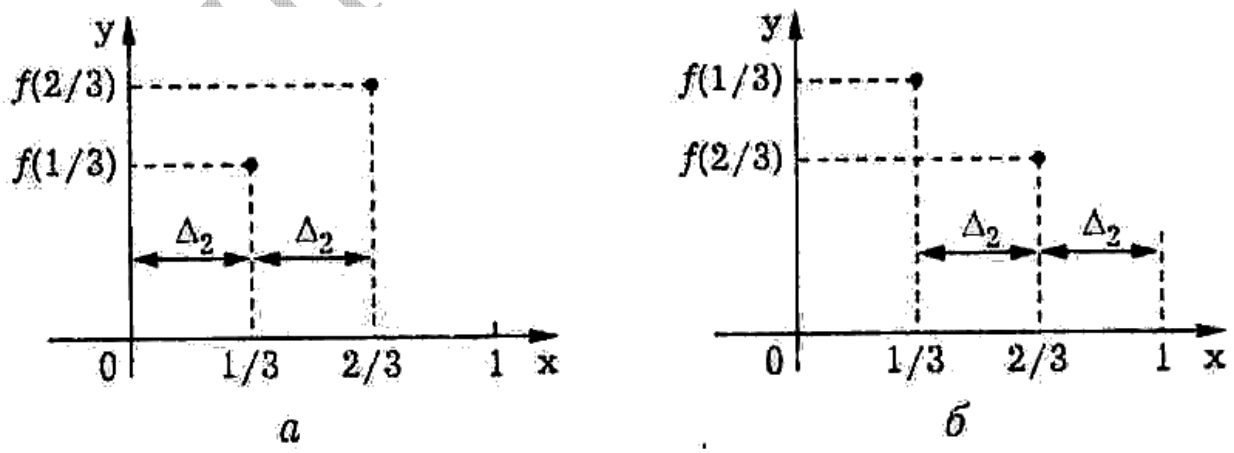


Рисунок 3.3 — Оптимальный пассивный поиск при *N=2*.

[2/3,1] можно исключить и считать, что $x_* \in [0,2/3]$. Тогда при выборе $x_* = 1/3$ наибольшая погрешность равна $\Delta_2 = 1/3$ и $f_* \approx f(1/3)$. Если же окажется, что $f(1/3)$

$> f(2/3)$ (рисунок 3.3, *б),* то можно исключить отрезок [0,1/3] и считать, что $x_* \in$ [1/3,1]. И в этом случае выбор $x_* = 2/3$ приведет к погрешности не более $\Delta_2 = 1/3$, а $f_* \approx f(1/3)$. Заметим, что при $(1/3) = f(2/3)$ можно исключить любой из указанных отрезков, гарантируя ту же точность нахождения точки $x_* \in [0,1]$. При ином делении отрезка [0, 1] на части двумя точками длина какой-то из его частей будет больше 1/3 и в действительности точка x_* может принадлежать именно этой части, так что получим погрешность $\Delta_2 > \Delta_2^* = 1/3$.

Рассуждая аналогично, можно заключить, что при $N = 3$ нужно также выбирать точки равномерно на отрезке [0,1]: $x_1 = 1/4$, $x_2 = 2/4$, $x_3 = 3/4$, обеспечив точность $\Delta_3^* = 1/4$ нахождения точки $x^* \in [0,1]$ и наименьшую длину $l_3^* = 1/2$ интервала неопределенности. В случае произвольного $N \in N$ по тем же соображениям надо выбирать точки

$$x_k = \frac{k}{N+1} \in [0,1], k = \overline{1,N}, \tag{3.4}$$

обеспечивая точность $\Delta_N^* = 1/(N+1)$ нахождения точки x^* и наименьшую возможную длину

$$l_N^* = \frac{2}{N+1} \tag{3.5}$$

интервала неопределенности. Таким образом, оптимальный пассивный поиск состоит в выборе точек, равномерно расположенных на отрезке. При этом (3.5) дает оценку скорости сходимости пассивного поиска с ростом числа N точек, так как скорость сходимости любого метода прямого поиска можно характеризовать скоростью уменьшения интервала неопределенности с возрастанием N.

3.4 Методы последовательного поиска

Метод дихотомии. Рассмотрим *последовательный поиск* точки $x^* \in [0,1]$, в которой *унимодальная* на отрезке [0,1] функция достигает наименьшего значения $f_* = f(x_*)$. Метод прямого поиска, основанный на делении отрезка пополам, на котором находится точка x^*, называют ***методом дихотомии***. Опишем алгоритм этого метода.

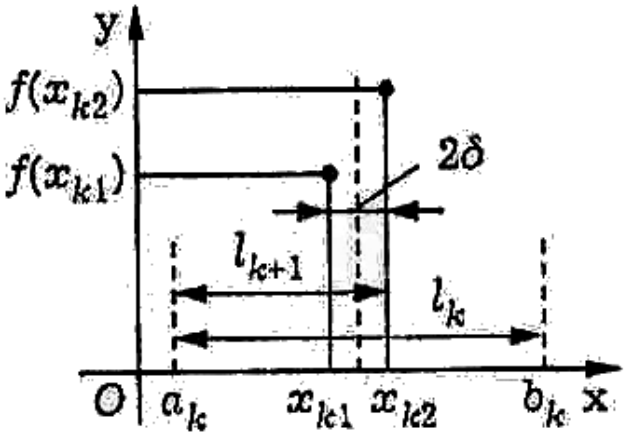


Рисунок 3.4 — Графическая иллюстрация метода дихотомии.

Пусть известно, что на *k*-м *шаге последовательного поиска* $x^* \in [a_k, b_k] \subset [0,1]$ (на первом шаге при *k*=1 имеем $a_1 = 0$ и $b_1 = 1$). На отрезке $[a_k, b_k]$ длиной l_k выберем две точки $x_{k1} = (a_k + b_k)/2 - \delta$ и $x_{k2} = (a_k + b_k)/2 + \delta$ (рисунок 3.4), где $\delta > 0$ — некоторое достаточно малое число. Вычислим значения $f(x_{k_1})$ и $f(x_{k2})$ функции $f(x)$ в этих точках и выполним *процедуру исключения отрезка.* В результате получим новый отрезок $[a_{k+1}, b_{k+1}] \subset [a_k, b_k]$. Если длина l_{k+1} нового отрезка больше заданной наибольшей допустимой длины ε_* *интервала неопределенности,* то алгоритм метода дихотомии переходит к *(k+1)*-му шагу, повторяя все описанные для *k-го* шага действия. Если же $l_{k+1} \le \varepsilon_*$, то вычисления прекращают и полагают $x_* = (a_{k+1} b_{k+1})/2$.

Так как $l_{k+1} = l_k/2 + \delta$, или $l_{k+1} - 2\delta = (l_k - 2\delta)/2$, то

$$l_k - 2\delta = \frac{l_1 - 2\delta}{2^{k-1}}.$$

Из этого равенства выводим следующую формулу длины l_k отрезка $[a_k, b_k]$, получаемого на *k*-м шаге метода дихотомии:

$$l_k = \frac{l_1 - 2\delta}{2^{k-1}} + 2\delta. \qquad (3.6)$$

Из (3.6) следует, что $l_k \to 2\delta$ при $k \to \infty$, но при этом $l_k > 2\delta$. Поэтому выполнение неравенства $l_{k+1} < \varepsilon_*$, означающее достижение заданной точности нахождения точки x^*, возможно лишь при условии выбора $2\delta < \varepsilon_*$. Кроме того, нужно учитывать неизбежную погрешность, возникающую при вычислении приближенных значений $\tilde{f}(x)$ функции $f(x)$. Это приводит к дополнительной погрешности Δ_* при нахождении точки x^* (см. 3.7). Поэтому выбор значения δ ограничен и снизу, т.е.

$$\Delta_* < 2\delta < \varepsilon_*. \qquad (3.7)$$

Если эти неравенства нарушаются, то знак разности $\tilde{f}(x_{k_1}) - \tilde{f}(x_{k2})$ может не совпадать со знаком разности $f(x_{k_1}) - f(x_{k2})$, что приводит к ошибочному выполнению процедуры исключения отрезка.

Итак, метод дихотомии — это последовательное построение на каждом *k*-м шаге поиска точек $x_{k1} = (a_k + b_k)/2 - \delta$ и $x_{k2} = (a_k + b_k)/2 + \delta$, симметричных относительно середины отрезка $[a_k, b_k]$ длины l_k. После выполнения *k*-го шага будет выделен отрезок $[a_{k+1}, b_{k+1}]$ длины l_{k+1} и вычислено $N = 2k$ значений функции. Используя формулу (3.6) для длины отрезка *(интервала неопределенности)* и полагая $l_1 = 1$, получаем

$$l_N^d = l_{k+1} = \frac{1 - 2\delta}{2^{k+1}} + 2\delta = \frac{1 - 2\delta}{2^{N/2}} + 2\delta. \qquad (3.8)$$

Сравнивая (3.8) с (3.5), видим, что скорость сходимости метода дихотомии значительно выше скорости сходимости *оптимального пассивного поиска.*

Отметим, что после исключения отрезка на *k*-м шаге описанного алгоритма точки x_{k1} и x_{k2} принадлежат новому отрезку $[a_{k+1}, b_{k+1}]$. Это значение вместе с уже вычисленным на предыдущем шаге значением функции во внутренней точке отрезка $[a_k, b_k]$ используют при выполнении процедуры исключения отрезка на следующем шаге последовательного поиска.

Метод золотого сечения. Как известно, ***золотым сечением отрезка*** называют такое его деление на две неравные части, при котором отношение длины всего отрезка к длине его большей части равно отношению длины большей части к длине меньшей.

Термин „золотое сечение" ввел Леонардо да Винчи. Золотое сечение широко применяли при композиционном построении многих произведений мирового искусства, в том числе в античной архитектуре и в эпоху Возрождения.

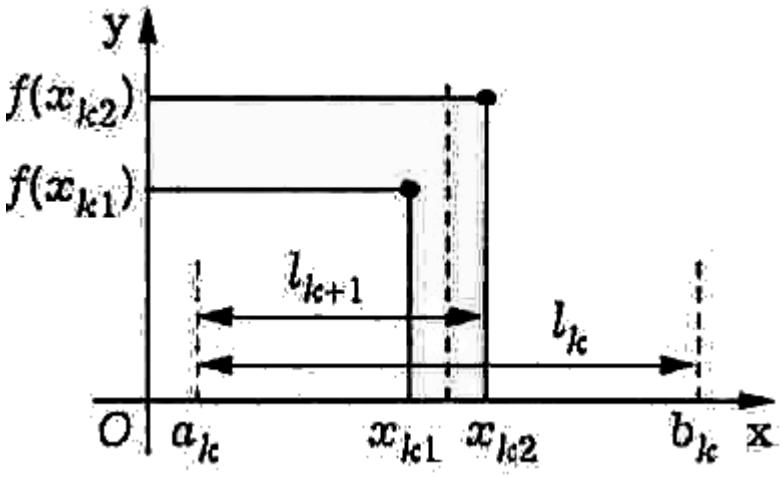


Рисунок 3.5 — Графическая иллюстрация алгоритма метода золотого сечения

Рассмотрим *k*-й шаг последовательного поиска. Чтобы выполнить процедуру исключения отрезка на этом шаге, отрезок $[a_k, b_k]$ необходимо двумя внутренними точками x_{k1}, x_{k2}, $x_{k1} < x_{k2}$, разделить на три части. Эти точки выберем симметрично относительно середины отрезка $[a_k, b_k]$ (рисунок 3.5) и так, чтобы каждая из них производила золотое сечение отрезка $[a_k, b_k]$. В этом случае отрезок $[a_{k+1}, b_{k+1}]$ внутри будет содержать одну из точек (другая будет одним из концов отрезка), причем эта точка будет производить золотое сечение отрезка $[a_{k+1}, b_{k+1}]$. Это вытекает из равенства длин отрезков $[a_k, x_{k1}]$ и $[x_{k2}, b_k]$. Таким образом, на $(k+1)$-м шаге в одной из точек $x_{k+1,1}$, $x_{k+1,2}$ значение функции вычислять не нужно. При этом отношение l_k / l_{k+1} длин отрезков сохраняется от шага к шагу, т.е.

$$\frac{l_k}{l_{k+1}} = \frac{l_{k+1}}{l_{k+2}} = \tau = const. \qquad (3.9)$$

Число τ называют *отношением* ***золотого сечения***.

Последовательный поиск, в котором на *k-м* шаге каждая из симметрично выбранных на отрезке $[a_k,b_k]$ точек x_{k1}, x_{k2} осуществляет золотое сечение этого отрезка, называют ***методом золотого сечения***. В этом методе каждое исключение отрезка уменьшает оставшийся отрезок в τ раз.

Выясним, чему равно отношение золотого сечения. Так как точки x_{k1} и $x_{k2}, x_{k1} < x_{k2}$ выбраны симметрично относительно середины отрезка $[a_k,b_k]$, то

$$b_k - x_{k2} = x_{k1} - a_k = l_k - l_{k+1}.$$

(см. рисунок 3.5). Для определенности будем считать, что на *k-м* шаге выбран отрезок $[a_k, x_{k2}]$. Тогда на *(k +1)*-м шаге одной из точек деления (а именно правой) будет точка x_{k1}. Значит, длина l_{k+2} отрезка, выбираемого на *(k + 1)*-м шаге, совпадает с длиной отрезка $[a_k, x_{k1}]$ и верно равенство $l_{k+2} = l_k - l_{k+1}$. Подставляя найденное выражение для l_{k+2} в уравнение (3.9), получаем

$$\frac{l_k}{l_{k+1}} = \frac{l_{k+1}}{l_k - l_{k+1}},$$

или $\tau = 1/(\tau - 1)$. Преобразуя это соотношение, приходим к квадратному уравнению $\tau^2 - \tau - 1 = 0$, имеющему единственное решение $\tau = \frac{1+\sqrt{5}}{2} \approx 1{,}618034$.

Предположим, что отрезком минимизации унимодальной функции $f(x)$ является [0,1], т.е. $a_1 = 0$, $b_1 = 1$ и $l_1 = 1$. На первом шаге последовательного поиска *(k*= 1) на отрезке [0,1] выбираем две точки $x_{11} = a_1 + (1 - 1/\tau)b_1 = 1 - 1/\tau$ и $x_{12} = a_1 + b_1/\tau = 1/\tau$, осуществляющие золотое сечение отрезка [0, 1]. Вычисляем значения минимизируемой функции в этих точках и выполняем процедуру исключения отрезка. Если $f(x_{11}) < f(x_{12})$, то выбираем отрезок $[a_1, x_{12}]$, т.е. полагаем $a_2 = a_1 = 0, b_2 = x_{12}$*;* в противном случае выбираем отрезок $[x_{11}, b_1]$, т.е. полагаем $a_2 = x_{11}, b_2 = b_1 = 1$. Кроме того, в первом случае принимаем $\tilde{x}_2 = x_{11}$, а во втором $\tilde{x}_2 = x_{12}$. Точка $\tilde{x}_2$ – одна из точек, осуществляющих золотое сечение отрезка $[a_2, b_2]$, меньшая в первом случае и большая во втором. Если длина вновь полученного отрезка больше заданной допустимой длины ε_* *интервала неопределенности,* то следует перейти ко второму шагу алгоритма, на котором одна из точек x_{21}, x_{22} есть точка $\tilde{x}_2$, а вторую можно найти, например, по формуле $a_2 + b_2 - \tilde{x}_2$. На втором шаге алгоритма вычисляем лишь одно значение функции в точке, симметричной $\tilde{x}_2$ относительно середины отрезка $[a_2, b_2]$. Если же длина l_2 отрезка $[a_2, b_2]$, полученного после первого шага алгоритма, оказалась меньше ε_* , то поиск прекращают и полагают $x_* \approx (a_2 + b_2)/2$.

Пусть на *k*-м шаге, $k \geq 2$, последовательного поиска по методу золотого сечения выбран отрезок $[a_k, b_k]$ и в нем точка $\tilde{x}_k$, осуществляющая золотое

сечение этого отрезка. Значение $f(\tilde{x}_k)$ функции в этой точке уже вычислено на предыдущем шаге. Находим вторую точку $\hat{x}_k$ золотого сечения по формуле

$$\hat{x}_k = a_k + b_k - \tilde{x}_k$$

и вычисляем в ней значение функции. Если $\hat{x}_k < \tilde{x}_k$, то $x_{k1} = \hat{x}_k$ и $x_{k2} = \tilde{x}_k$. Если $f(x_{k1}) < f(x_{k2})$, то выбираем отрезок $[a_k, x_{k2}]$, т.е. полагаем $a_{k+1} = a_k, b_{k+1} = x_{k2}, \tilde{x}_{k+1} = x_{k1}$, иначе выбираем отрезок $[x_{k1}, b_k]$, т.е. полагаем $a_{k+1} = x_{k1}, b_{k+1} = b_k, \tilde{x}_{k+1} = x_{k2}$. Длину l_{k+1} нового отрезка $[a_{k+1}, b_{k+1}]$ сравниваем с ε_* и принимаем решение, продолжать поиск (при $l_{k+1} \geq \varepsilon_*$) или нет (при $l_{k+1} < \varepsilon_*$). В случае прекращения поиска полагаем $x_* \approx (a_k + b_k)/2$.

Согласно описанию алгоритма, на первом шаге значение функции вычисляют в двух точках, а на каждом из последующих шагов вычисляют лишь одно значение функции. Поэтому после *k* шагов алгоритма значение функции будет вычислено в $N = k + 1$ точках. Поскольку после каждого шага интервал неопределенности уменьшается в τ раз, то для длины l_{k+1} отрезка $[a_{k+1}, b_{k+1}]$ получаем $l_{k+1} = l_1 / \tau^k$, а зависимость l_N^z длины интервала неопределенности от количества *N* вычисленных значений функции выражается формулой

$$l_N^z = l_{k+1} = \frac{1}{\tau^k} = \frac{1}{\tau^{N-1}}. \qquad (3.10)$$

Алгоритмы методов золотого сечения и дихотомии аналогичны. Различие состоит лишь в том, что в методе дихотомии расстояние 2δ между внутренними точками x_{k1}, x_{k2} отрезка $[a_k, b_k]$ на каждом *k*-м шаге остается неизменным, а в методе золотого сечения оно зависит от номера шага поиска и уменьшается с. уменьшением длины l_k отрезка по мере возрастания номера шага. Действительно, в методе золотого сечения на *k*-м шаге поиска внутренними точками отрезка $[a_k, b_k]$ будут $x_{k1} = a_k + (1 - 1/\tau) l_k$ и $x_{k2} = a_k + l_k / \tau$, а расстояние между ними равно $x_{k1} - x_{k2} = (2/\tau - 1) l_k = (\sqrt{5} - 2) l_k \approx 0{,}236068\, l_k$.

Метод Фибоначчи. Пусть при поиске точки $x_* \in [0,1]$, в которой унимодальная на отрезке [0, 1] функция $f(x)$ принимает наименьшее на этом отрезке значение, можно вычислить ее значения только в двух точках. Тогда предпочтение следует отдать методу дихотомии при $\delta << 1$, так как он позволит уменьшить интервал неопределенности почти вдвое, а метод золотого сечения — лишь в $\tau \approx 1{,}618$ раз. Сравнение (3.8) и (3.10) показывает, что при количестве вычисляемых значений функции $N \geq 4$ эффективность метода золотого сечения становится выше, чем метода дихотомии.

Однако при любом заданном общем числе $N > 2$ вычисляемых значений функции можно построить еще более эффективный метод, состоящий из $N - 1$ шагов. Он сочетает преимущество симметричного расположения внутренних точек x_{k1}, x_{k2} на отрезке $[a_k, b_k]$ относительно его середины, реализованное в методах дихотомии и золотого сечения, с возможностью на каждом шаге изменять отношение l_k / l_{k+1} длин сокращаемого и нового отрезков. Как показано

при обсуждении метода золотого сечения, в случае выбора внутренних точек симметрично относительно середины отрезка для трех последовательных шагов этого метода выполняется соотношение

$$l_{k-1} = l_k + l_{k+1}, k = 2,3,... \tag{3.11}$$

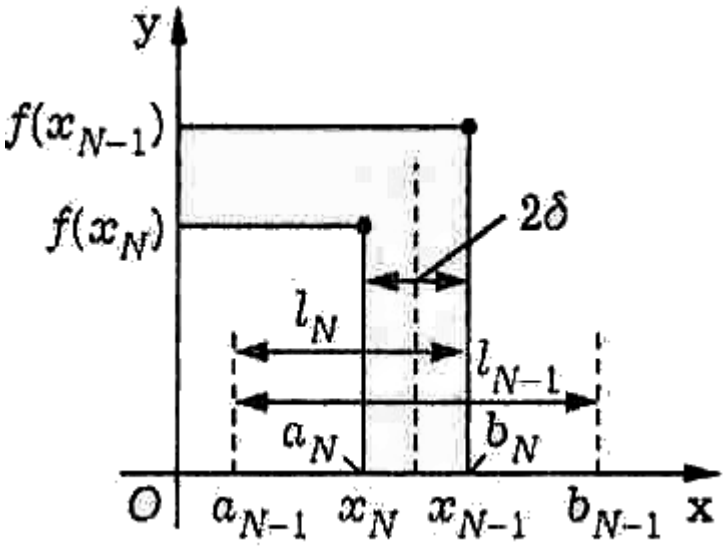


Рисунок 3.6 — Графическая иллюстрация алгоритма метода Фибоначчи

Построение алгоритма такого метода удобнее начать с последнего шага, но предварительно уточним задачу. Располагая возможностью вычислить в N точках $x_k \in [0,1]$, $k = \overline{1,N}$ значения унимодальной на отрезке функции $f(x)$, необходимо как можно точнее, т.е. с наименее возможной длиной интервала неопределенности, отыскать точку x_* наименьшего значения этой функции на отрезке [0,1].

При выполнении процедуры исключения отрезка на последнем, *(N–1)*-м шаге имеем отрезок $[a_{N-1}, b_{N-1}]$ длины l_{N-1} с двумя внутренними точками x_{N-1} и x_N, симметрично расположенными относительно середины отрезка на достаточно малом расстоянии 2δ друг от друга (рисунок 3.6). В этих точках вычислены значения $f(x_{N-1})$ и $f(x_N)$ функции *f(x)*. Пусть для определенности $f(x_N) < f(x_{N-1})$, тогда для нового отрезка $[a_N, b_N]$ длины $l_N = l_{N-1}/2 + \delta$ внутренней будет точка x_N, а точка x_{N-1} совпадет с одним из его концов. В такой ситуации при выборе $x_* = x_{N-1}$ длина интервала неопределенности равна пока неизвестной длине l_N отрезка $[a_N, b_N]$. Через l_N можно выразить длину $l_{N-1} = 2l_N - 2\delta$ отрезка $[a_{N-1}, b_{N-1}]$. Далее в соответствии с (3.11) получаем

$$l_{N-2} = l_{N+1} + l_N = 3l_N - 2\delta,\ l_{N-3} = l_{N-2} + l_{N-1} = 5l_N - 4\delta,$$

$$l_{N-4} = l_{N-3} + l_{N-2} = 8l_N - 6\delta,\ l_{N-5} = l_{N-4} + l_{N-3} = 13l_N - 10\delta,$$

и в общем виде

$$l_{N-K} = F_{K+2} l_N - 2F_K \delta, \quad K = \overline{0, N-1}, \tag{3.12}$$

где коэффициенты F_m определены рекуррентным соотношением

$$F_m = F_{m-1} + F_{m-2}, \quad m = \overline{3, N-1}, \quad F_1 = F_2 = 1. \tag{3.13}$$

Так как при *K=N-1* длина $l_{N-K} = l_1 = 1$ отрезка [0,1] известна, то из (3.12) можно найти длину интервала неопределенности

$$l_N^f = \frac{l_1}{F_{N+1}} + 2\delta\frac{F_{N-1}}{F_{N+1}}. \tag{3.14}$$

Существует алгоритм прямого поиска, удовлетворяющий соотношению (3.14).

Все коэффициенты F_m принадлежат множеству N натуральных чисел, и

Таблица 3.1–Числа Фибоначчи

m	F_m	m	F_m	m	F_m	m	F_m
0	1	6	13	12	233	18	4181
1	1	7	21	13	377	19	6765
2	2	8	34	14	610	20	10946
3	3	9	55	15	987	21	17711
4	5	10	89	16	1597	22	28657
5	8	11	144	17	2584	23	46368

их называют ***числами Фибоначчи.*** В таблице 3.1 представлены эти числа до номера $m = 23$.

Метод, использующий числа Фибоначчи для выбора длин отрезков l_k, а значит, и точек $x_k \in [0,1], k = \overline{1, N}$, в которых вычисляют значения минимизируемой функции, называют ***методом Фибоначчи*** (иногда – оптимальным последовательным поиском). Если на первом шаге поиска (*k=1, K=N-1*) интервал неопределенности имеет длину l_1, то в соответствии с (3.12) и (3.14) длина l_2 нового отрезка $[a_2, b_2]$ равна

$$l_2 = F_N l_N - 2\delta F_{N-2} = \frac{F_N}{F_{N+1}} l_1 + 2\delta\frac{F_N F_{N-1} - F_N F_{N-2}}{F_{N+1}} = \frac{F_N}{F_{N+1}} l_1 + (-1)^{N+1}\frac{2\delta}{F_{N+1}}.$$

Опишем алгоритм метода, пренебрегая малой величиной δ, т.е. принимая

$$\frac{l_1}{l_2} = \frac{F_N}{F_{N+1}}. \tag{3.15}$$

Несложно проверить, что в этом случае выполнение процедуры исключения отрезка на последнем, $(N-1)$-м шаге поиска приводит к совпадению внутренних точек x_{N-1} и x_N (см. рисунок 3.9).

Отметим, что уже при *N=11* имеем $F_{12} = F_{11} = 144/89 \approx 1{,}617978$, а при *N=21* получаем $F_{22} = F_{21} = 17711/10946 \approx 1{,}618034$, что совпадает с отношением τ золотого сечения с точностью до 10^{-6}. Таким образом, на первом шаге длина исходного отрезка уменьшается практически так же, как и в методе золотого сечения.

При $l_1 = 1$ из (3.15) находим $l_2 = F_N / F_{N+1}$. Таким образом, учитывая (3.13), заключаем, что на первом шаге выбор точек, симметричных относительно середины отрезка [0, 1], можно определить по формулам

$$x_1 = l_2 = \frac{F_N}{F_{N+1}}, x_2 = 1 - l_2 = 1 - \frac{F_N}{F_{N+1}} = \frac{F_{N-1}}{F_{N+1}}, x_2 > x_1,$$

причем расстояние между ними будет равно

$$d_1 = x_1 - x_2 = \frac{F_N}{F_{N+1}} - \frac{F_{N-1}}{F_{N+1}} = \frac{F_{N-2}}{F_{N+1}}.$$

После выполнения на этом шаге процедуры исключения отрезка одна из точек x_1, x_2 будет граничной точкой нового отрезка $[a_2, a_1]$, а другая — его внутренней точкой, которую обозначим x_2'. Вторая внутренняя точка на этом отрезке должна быть выбрана симметрично точке x_2' относительно его середины. Аналогично происходит выбор второй внутренней точки нового отрезка на всех последующих шагах поиска.

На *k*-м шаге в соответствии с равенством (3.12), в котором следует положить *K = N-k,* и равенством (3.14) длина отрезка $[a_k, b_k]$ равна $l_k = F_{N+2-k} / F_{N+1}$ и происходит ее уменьшение в $l_k / l_{k+1} = F_{N+2-k} / F_{N+1-k}$ раз. Если внутренние точки на этом отрезке обозначить α_k и β_k, то проведенные рассуждения позволяют написать

$$\alpha_k = a_k + \frac{F_{N-k}}{F_{N+1}}, \quad \beta_k = a_k + \frac{F_{N+1-k}}{F_{N+1}}, \alpha_k < \beta_k, k = \overline{1, N-1}.$$

Подчеркнем, что реализация метода Фибоначчи предполагает априорное задание требуемого количества *N* вычисляемых значений функции (или количества шагов поиска). Этот параметр необходим для реализации первого шага алгоритма при выборе точек x_{11} и x_{12} деления отрезка $[a_1, b_1]$. Если параметр *N* по каким-либо причинам не может быть задан заранее, следует использовать другие методы, например дихотомии или Золотого сечения.

3.5. Сравнение методов последовательного поиска

В качестве оценки скорости сходимости методов прямого поиска можно использовать скорость убывания длины *интервала неопределенности* в зависимости от числа *n* вычисленных значении минимизируемой функции в различных методах. Для *метода дихотомии,* пренебрегая в (3.8) малой величиной δ, находим

$$l_n^d = \frac{1 - 2\delta}{2^{n/2}} + 2\delta \approx \frac{1}{2^{n/2}}, \tag{3.16}$$

а для *метода золотого сечения* и *метода Фибоначчи*, согласно (3.10) и (3.14) получаем

$$l_n^z = \frac{1}{\tau^{n-1}},\ l_n^f = \frac{1}{F_{n+1}} + 2\delta\frac{F_{n-1}}{F_{n+1}} \approx \frac{1}{F_{n+1}} \tag{3.17}$$

соответственно, где $\tau \approx 1{,}618034$ -*отношение золотого сечения,* $F_m, m \in \mathrm{N}$, – *числа Фибоначчи*.

Используя формулу Бине

$$F_n = \frac{\tau^n - (-\tau)^{-(n+1)}}{\sqrt{5}},$$

сравним два последних метода при $n \to \infty$:

$$\lim_{n\to\infty}\frac{l_n^z}{l_n^f} = \lim_{n\to\infty}\frac{F_{n+1}}{\tau^{n-1}} = \lim_{n\to\infty}\frac{\tau^{n+1} - (-\tau)^{-(n+1)}}{\tau^{n-1}\sqrt{5}} = \frac{\tau^2}{\sqrt{5}} \approx 1{,}17082.$$

Таким образом, скорость сходимости метода Фибоначчи при больших значениях *n* всего примерно на 17% выше, чем скорость сходимости метода золотого сечения.

Сравнивая при больших значениях *n* методы золотого сечения и дихотомии, получаем

$$\lim_{n\to\infty}\frac{l_n^z}{l_n^d} = \lim_{n\to\infty}\frac{2^{n/2}}{\tau^{n-1}} = \tau\lim_{n\to\infty}(\frac{\sqrt{2}}{\tau})^n = 0,$$

Таким образом, метод золотого сечения качественно „лучше" метода дихотомии. Но из (3.5) и (3.16) следует

$$\lim_{n\to\infty}\frac{l_n^d}{l_n^*} = \lim_{n\to\infty}\frac{1/2^{n/2}}{2/(n+1)} = \lim_{n\to\infty}\frac{n+1}{2^{n/2+1}} = 0,$$

т.е. скорость сходимости метода дихотомии при больших значениях *n* выше, чем скорость сходимости метода *оптимального пассивного поиска.*

Итак, метод золотого сечения уступает по скорости сходимости лучшему методу — методу Фибоначчи — примерно в 1,17 раза, но является более гибким, поскольку не требует выбора заранее определенного числа точек, в которых предстоит вычислить значения минимизируемой функции. В таблице 3.2 приведены значения длины интервалов неопределенности для рассмотренных методов в зависимости от числа *N* вычисленных значений функции.

3.6. Методы полиномиальной аппроксимации

В *методах прямого поиска* мы не имели никакой информации о минимизируемой функции за исключением ее значений в выбранных нами точках и предположения, что она непрерывна и является *унимодальной функцией* на рассматриваемом отрезке. Если функцию в некоторой окрестности точки ее минимума можно достаточно точно заменить (аппроксимировать) многочленом, то для ее минимизации целесообразно использовать так называемые ***методы***

полиномиальной аппроксимации. Их общая особенность состоит в вычислении коэффициентов многочлена по известным значениям функции в отдельных точках и последующем нахождении минимума этого многочлена с использованием необходимых и достаточных условий экстремума. Ограничимся рассмотрением ***метода квадратичной аппроксимации*** минимизируемой функции *f(x),* в котором график этой функции приближенно заменяют параболой, проходящей через три известные точки $(x_i, f_i), i = 1,2,3,$ где $f_i = f(x_i)$.

Известно, что через три различные точки, не лежащие на одной прямой, можно провести только одну параболу $y = ax^2 + bx + c$, $a \neq 0$. Коэффициенты *a, b, c* удовлетворяют системе линейных алгебраических уравнений (СЛАУ)

$$ax_i^2 + bx_i + c = f_i, i = 1,2,3.$$

Определитель этой СЛАУ

$$\begin{vmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{vmatrix} = (x_1 - x_2)(x_1 - x_3)(x_2 - x_3)$$

представляет собой *определитель Вандермонда* и отличен от нуля, когда x_1, x_2, x_3 попарно различны. В этом случае СЛАУ имеет решение, и притом единственное. Его можно записать в виде

$$a = \frac{f_1}{(x_1 - x_2)(x_1 - x_3)} + \frac{f_2}{(x_2 - x_1)(x_2 - x_3)} + \frac{f_3}{(x_3 - x_1)(x_3 - x_2)},$$

$$b = \frac{f_1(x_2 + x_3)}{(x_1 - x_2)(x_3 - x_1)} + \frac{f_2(x_3 + x_1)}{(x_2 - x_1)(x_3 - x_2)} + \frac{f_3(x_1 + x_2)}{(x_3 - x_1)(x_2 - x_3)},$$

$$c = \frac{f_1 x_2 x_3}{(x_1 - x_2)(x_1 - x_3)} + \frac{f_2 x_3 x_1}{(x_2 - x_1)(x_2 - x_3)} + \frac{f_3 x_1 x_2}{(x_3 - x_1)(x_3 - x_2)}.$$

Если найденные выражения для коэффициентов *a* и *b* подставить в необходимое условие $y' = 2ax + b = 0$ экстремума функции, то получим ее единственную стационарную точку

$$\tilde{x}_* = -\frac{b}{2a} = \frac{1}{2}\frac{f_1 r_{23} + f_2 r_{31} + f_3 r_{12}}{f_1 s_{23} + f_2 s_{31} + f_3 s_{12}}, \qquad (3.18)$$

где $r_{ij} = x_i^2 - x_j^2$ и $s_{ij} = x_i - x_j, i, j = 1,2,3.$ Так как $y'' = 2a = const,$ то в точке $\tilde{x}_*$ при $a > 0$ имеем минимум функции *y(x)*, а при a<0 – максимум.

Если известен отрезок, на котором минимизируемая функции унимодальна, то нет необходимости вычислять значение коэффициента *a.* Достаточно этот отрезок принять в качестве отрезка $[x_1, x_3]$, а точку $x_2 \in (x_1, x_3)$ выбрать произвольно в интервале (x_1, x_3). В этом случае имеем $f_1 \geq f_2$ и $f_3 \geq f_2$, откуда $\tilde{x}_* \in [x_1, x_3]$.

На первом шаге метода квадратичной аппроксимации при помощи (3.18) вычисляют $\tilde{x}_*^{(1)}$ и затем $\tilde{f}_*^{(1)} = f(\tilde{x}_*^{(1)})$. Для вычислений на втором шаге из четырех

точек $\tilde{x}_*^{(1)}$ и $x_i, i=1,2,3,$ выбирают новую тройку точек $x_i^{(2)}$ по следующему правилу:

а) если $\tilde{x}_*^{(1)} \in [x_2, x_3]$ и $\tilde{f}_*^{(1)} \le f_2$, то $x_1^{(2)} = x_2, x_3^{(2)} = x_3, x_2^{(2)} = \tilde{x}_*^{(1)}$;

б) если $\tilde{x}_*^{(1)} \in [x_2, x_3]$ и $\tilde{f}_*^{(1)} > f_2$, то $x_1^{(2)} = x_1, x_3^{(2)} = \tilde{x}_*^{(1)}, x_2^{(2)} = x_2$;

в) если $\tilde{x}_*^{(1)} \in [x_1, x_2]$ и $\tilde{f}_*^{(1)} \le f_2$, то $x_1^{(2)} = x_1, x_3^{(2)} = x_2, x_2^{(2)} = \tilde{x}_*^{(1)}$;

г) если $\tilde{x}_*^{(1)} \in [x_1, x_2]$ и $\tilde{f}_*^{(1)} > f_2$, то $x_1^{(2)} = \tilde{x}_*^{(1)}, x_3^{(2)} = x_3, x_2^{(2)} = x_2$;

Далее из (3.18) находят $\tilde{x}_*^{(2)}$, а затем описанную процедуру повторяют на третьем шаге и так далее до тех пор, пока длина *интервала неопределенности,* в котором гарантированно лежит искомая точка x_* минимума функции *f(x)*, не станет меньше заданного наибольшего допустимого значения ε_*. Отметим, что подтверждением приближения к точке x_* и правильности вычислений может служить уменьшение значения функции *y(x)* в найденной точке $\tilde{x}_*^{(k)}$ по сравнению со значением $y(\tilde{x}_*^{(k-1)})$ на предыдущем шаге.

3.7. Методы с использованием производных

В *методах прямого поиска* при вычислении значений минимизируемой функции *f(x)* неизбежно возникают погрешности, к которым чувствительны алгоритмы прямого поиска, основанные на сравнении значений функции в отдельных точках.

Пусть *f(x)* — дважды непрерывно дифференцируемая функция в окрестности точки x_* строгого локального минимума, значения которой вычисляются с абсолютной погрешностью, не превышающей Δ_f. Оценим абсолютную погрешность Δ_*, с которой может быть найдена точка x_* с применением метода прямого поиска. Для представления функции *f(x)* в окрестности точки x_* воспользуемся формулой Тейлора с учетом равенства $f(x_*) = 0$:

$$f(x) = f(x_*) + f''(\xi)\frac{(x-x_*)^2}{2},$$

где ξ — точка, лежащая между точками x_* и x. Имея в виду, что вычисления проводятся в достаточно малой окрестности точки x_*, положим $f''(\xi) \approx f''(x_*)$. Тогда для приближенно вычисляемых значений $\tilde{f}(x)$ получим

$$\tilde{f}(x) - \tilde{f}(x_*) = f(x) - f(x_*) + (\tilde{f}(x) - f(x)) - (\tilde{f}(x_*) - f(x_*)) \ge$$

$$\ge f(x) - f(x_*) - 2\Delta_f \approx f''(x_*)\frac{(x-x_*)^2}{2} - 2\Delta_f.$$

Из этих неравенств вытекает, что можно гарантировать выполнение неравенства $\tilde{f}(x) > \tilde{f}(x_*)$, если $f''(x_*)(x-x_*)^2 > 2\Delta_f$. Это приводит к приближенной оценке абсолютной погрешности нахождения точки x_* методом прямого поиска:

$$\Delta_* \approx \sqrt{\frac{2\Delta_f}{f''(x_*)}}. \qquad (3.20)$$

Заданная точность ε_* нахождения точки x_* не должна быть меньше Δ_*, так как иначе эту точность нельзя достичь методом прямого поиска. Из (3.20) также следует, что при нахождении точки x_* происходит потеря примерно половины верных значащих цифр, с которыми можно вычислить приближенное значение минимизируемой функции.

Если унимодальная функция *f(x)* непрерывно дифференцируема на отрезке минимизации, то точку x_* наименьшего значения функции можно вычислять как корень уравнения $f'(x)=0$ с помощью тех или иных методов численного решения нелинейных уравнений. В этом случае на точность решения задачи решающее влияние оказывает погрешность вычисления производной функции.

Если абсолютная погрешность вычисления производной не превышает $\Delta_{f'}$, то для нижней оценки абсолютной погрешности вычисления корня x_* уравнения *f(x)* = 0 имеем

$$\Delta'_* = \frac{\Delta_{f'}}{f''(x_*)}. \qquad (3.21)$$

Таким образом, при нахождении этого корня можно сохранить все верные значащие цифры, с которыми можно вычислить значение производной $f'(x)$.

Рассмотрим некоторые методы одномерной минимизации, основанные на использовании производной минимизируемой функции.

Метод средней точки. Будем искать минимум *функции f(x),* непрерывно дифференцируемой и *строго унимодальной* на отрезке $[a_1,b_1]$. В этом случае единственной точкой $x_* \in [a_1,b_1]$ минимума будет стационарная точка, в которой $f'(x_*)= 0$. Отметим, что непрерывно дифференцируемая *унимодальная* на отрезке *функция* может иметь на нем более одной стационарной точки.

В ***методе средней точки*** используют простую идею: вычисляют производную $f'(\bar{x}_1) = K_1$ в средней точке $\bar{x}_1 = (a_1 + b_1)/2$ исходного отрезка и, если $K_1 > 0$, то отрезок $[\bar{x}_1, b_1]$ отбрасывают, так как на нем строго унимодальная функция только возрастает, а если $K_1 < 0$, то отбрасывают отрезок $[a_1, \bar{x}_1]$, поскольку на нем строго унимодальная функция лишь убывает. В данном случае отбрасывание половины исходного отрезка $[a_1,b_1]$ аналогично процедуре *исключения отрезка*. Ясно, что в случае *К=0* имеем $x_* = \bar{x}$.

Оставшуюся после отбрасывания половину отрезка обозначим $[a_2,b_2]$ и, вычислив производную $f'(\bar{x}_2) = K_2$ в его средней точке $\bar{x}_2 = (a_2 + b_2)/2$, повторим процедуру отбрасывания для половины отрезка и т.д. Условием прекращения вычислений на *k*-м шаге может быть выполнение неравенства $l_{k+1} < \varepsilon_*$, где

$l_{k+1} = b_{k+1} - a_{k+1}$ — длина отрезка $[a_{k+1}, b_{k+1}]$ после отбрасывания половины отрезка $[a_k, b_k]$; ε_* — наибольшая допустимая длина интервала неопределенности.

Метод средней точки напоминает *метод дихотомии,* но сходится к искомому значению x_* быстрее, поскольку в отличие от (3.8) для метода средней точки после вычисления *n* значений производной минимизируемой на отрезке [0,1] функции *f(x)* для длины интервала неопределенности получаем

$$l_n = \frac{1}{2^n}. \tag{3.22}$$

Таким образом, для одинакового уменьшения значения l_n в методе средней точки нужно вычислить вдвое меньше значений производной функции по сравнению с числом значений самой функции в методе дихотомии.

Метод Ньютона. Если строго унимодальная на отрезке [a,b] функция *f(x)* дважды непрерывно дифференцируема на этом отрезке, то точку $x_* \in [a,b]$ минимума этой функции можно найти путем решения уравнения $f'(x) = 0$ *методом Ньютона,* иногда называемым методом касательных. Пусть $x_0 \in [a,b]$ — нулевое приближение к искомой точке x_*, называемое обычно ***начальной точкой***. Линеаризуем функцию $f'(x)$ в окрестности начальной точки, приближенно заменив дугу графика этой функции касательной в точке $(x_0, f'(x_0))$:

$$f'(x) \approx f'(x_0) + f''(x_0)(x - x_0) \tag{3.23}$$

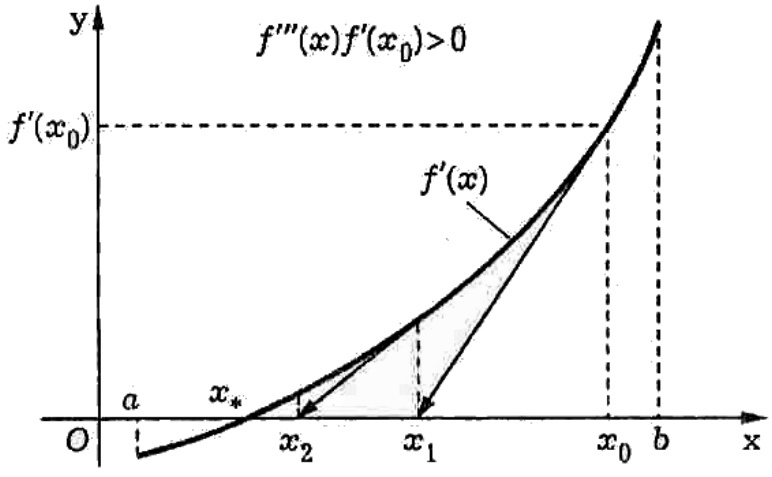


Рисунок 3.7— Графическая иллюстрация метода Ньютона

Выберем в качестве следующего приближения к x_* точку x_1 пересечения касательной с осью абсцисс (рисунок 3.7). Приравнивая нулю правую часть (3.23), получаем первый элемент $x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)}$ итерационной

последовательности $\{x_k\}$. На *(k+1)*-м шаге по найденной на предыдущем шаге точке x_k можно найти точку

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}. \qquad (3.24)$$

Для квадратичной функции *f(x)* функция $f'(x)$ линейна. Поэтому в (3.23) равенство будет точным, а метод Ньютона будет сходиться за один шаг при любом выборе точки x_0 из области определения этой функции.

В общем случае сходимость метода Ньютона существенно зависит от выбора начальной точки x_0. Для надежной работы этого метода необходимо, чтобы вторая производная $f''(x)$ в окрестности искомой точки x_* сохраняла знак, а начальная точка x_0 выбиралась из такой окрестности. В противном случае второе слагаемое в правой части (3.24) может стать неограниченным. Поскольку для дважды непрерывно дифференцируемой функции в точке минимума $f''(x_*) > 0$, то должно быть и $f''(x_0) > 0$. Поэтому говорят, что метод Ньютона обладает *локальной сходимостью* в том смысле, что надо выбрать хорошее начальное приближение, попадающее в такую окрестность x_*, где $f''(x) > 0$. Однако проверка выполнения этого условия не всегда возможна.

Достаточное условие надежной работы метода Ньютона при нахождении точки $x_* \in [a,b]$ минимума функции *f(x)* можно установить в случае, если эта функция трижды непрерывно дифференцируема на отрезке $[a,b]$. Ясно, что итерационная последовательность $\{x_k\}$ будет сходиться к пределу x_* монотонно, если $0 < \frac{x_* - x_{k+1}}{x_* - x_k} < 1$. В соответствии с формулой Тейлора с остаточным членом в форме Лагранжа имеем

$$f'(x_*) = 0 = f'(x_k) + f''(x_k)(x_* - x_k) + f'''(x)\frac{(x_* - x_k)^2}{2},$$

где точка *x* лежит между x_k и x_*. Поэтому с учетом (3.24) имеем

$$\frac{x_* - x_{k+1}}{x_* - x_k} = \frac{x_* - x_k + \frac{f'(x_k)}{f''(x_k)}}{x_* - x_k} = 1 - \frac{2}{2 + \frac{f'''(x)(x_* - x_k)^2}{f'(x_k)}}.$$

Таким образом, последовательность $\{x_k\}$ является монотонной, если $\frac{f'''(x)}{f'(x_k)} > 0$, т.е. достаточным условием монотонной сходимости метода Ньютона будут постоянство в интервале между точками x_0 и x_* знака производной $f'''(x)$ и совпадение его со знаком $f'(x_0)$. Оказывается, что в этом случае метод Ньютона обладает *квадратичной скоростью сходимости* в некоторой δ – окрестности $U(x_*, \delta)$ точки x_*, причем

$$|x_* - x_k| \le \frac{(x_* - x_{k-1})^2}{C},$$

где

$$C = 2\frac{\min|f''(x)|}{\max|f'''(x)|}, \tag{3.25}$$

минимум и максимум вычисляются по множеству $U(x_*, \delta)$.

Если радиус δ – окрестности $U(x_*, \delta)$ не превосходит C, а точка x_k попадает в окрестность $U(x_*, \delta/2)$. В этом случае верна оценка

$$|x_k - x_*| < |x_{k-1} - x_k|, \tag{3.26}$$

которую можно использовать для оценки точности найденного решения уравнения $f'(x) = 0$ (точки локального минимума функции).

Модификации метода Ньютона. Вычисление второй производной $f''(x_k)$ минимизируемой функции *f(x)* на каждом *k*-м шаге метода Ньютона может оказаться достаточно трудоемким. В этом случае целесообразно использовать *упрощенный метод Ньютона,* положив в (3.24) $f''(x_k) = f''(x_0) =$ const:

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_0)}. \tag{3.27}$$

Оказывается, что этот метод имеет линейную скорость сходимости. При этом если в интервале между точками x_* и x_0 выполнено условие $f'(x_0)f'''(x) > 0$, то последовательность $\{x_k\}$, построенная в соответствии с (3.27), сходится к точке x_* монотонно.

Можно избежать вычисления второй производной минимизируемой на отрезке [a, b] функции *f(x)*, если располагать двумя приближениями $x_0, x_1 \in$ [a, b] к искомой точке $x_* \in$ [a, b] минимума этой функции. Заменяя в (3.24) при $k = 1$ производную $f''(x_1)$ выражением

$$\frac{f'(x_1) - f(x_0)}{x_1 - x_0},$$

получаем

$$x_2 = x_1 - \frac{x_1 - x_0}{f'(x_1) - f'(x_0)} f'(x_1),$$

а в случае произвольного номера $k \in$ N приходим к формуле

$$x_{k+1} = x_k - \frac{x_k - x_{k-1}}{f'(x_k) - f'(x_{k-1})} f'(x_k). \tag{3.28}$$

Метод решения нелинейного уравнения $f'(x) = 0$ с применением рекуррентного соотношения (3.28) обычно называют *методом секущих.* Геометрическая интерпретация этого метода состоит в том, что в качестве очередного приближения x_{k+1} выбирают точку пересечения с осью абсцисс не касательной к графику функции $f'(x)$, как это делают в методе Ньютона, а

секущей, проходящей через две точки этого графика, найденные при выполнении двух предыдущих шагов метода. Выбор начальной точки x_0 в (3.28) при k= 1 проводят следующим образом. Если на отрезке [a, b] функция *f(x)* имеет знакопостоянную третью производную $f'''(x)$, то в качестве x_0 выбирают тот конец отрезка [a, b], на котором совпадают знаки $f'(x)$ и $f'''(x)$, а в качестве

$$x_1 = \frac{af'(b) - bf'(a)}{f'(b) - f'(a)} \text{—} \tag{3.29}$$

точку пересечения с осью абсцисс хорды, стягивающей дугу графика функции $f'(x)$ на отрезке [a, b] (рисунок 3.8). Таким образом, первый шаг метода секущих выполняют согласно *методу хорд,* а последующие шаги — в соответствии с (3.28). Этот метод имеет *сверхлинейную скорость сходимости*, причем $|x_* - x_k| \le C(x_* - x_{k-1})^7$, где C = const, а $\tau = \frac{1+\sqrt{5}}{2} \approx 1{,}618$ — *отношение золотого сечения.*

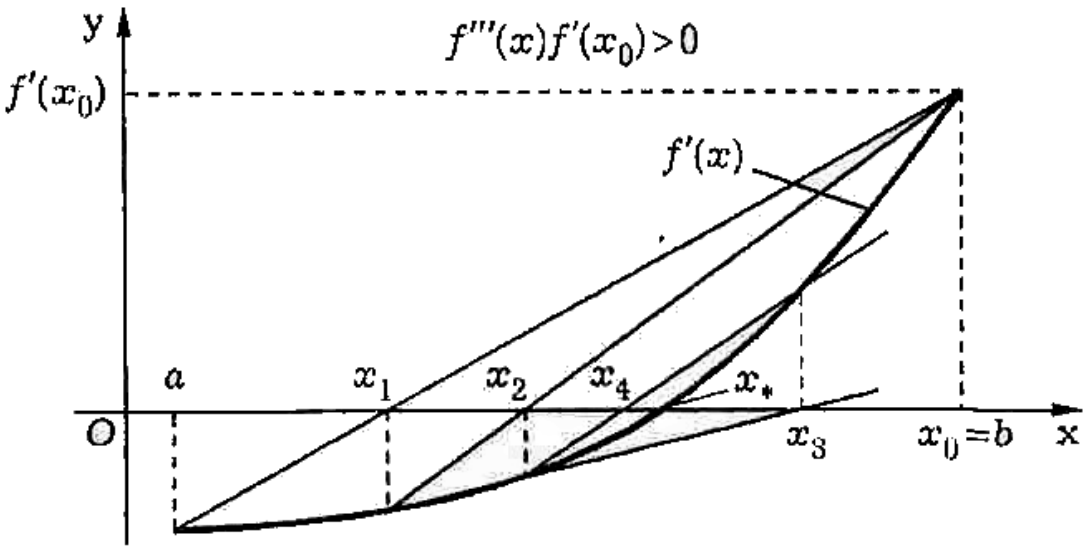


Рисунок 3.8 — Модификация метода Ньютона

Модификации метода Ньютона обладают только локальной сходимостью, т.е. сходятся, если начальная точка выбрана в некоторой окрестности точки минимума функции. Если же начальная точка расположена слишком далеко от точки минимума, то подобный метод может расходиться или “зацикливаться”. В некоторых случаях целесообразно сочетать различные модификации метода Ньютона, чередуя их применение в зависимости от номера шага.

Метод кубической аппроксимации. В *методах полиномиальной аппроксимации* при построении многочлена, аппроксимирующего минимизируемую функцию в окрестности искомой точки x_* минимума, помимо значений функции в отдельных точках могут быть использованы и значения ее производных.

Пусть для непрерывно дифференцируемой функции *f(x),* строго выпуклой на отрезке $[x_1, x_2]$, известны значения $f_1 = f(x_1), f_2 = f(x_2), f_1' = f'(x_1)$ и $f_2' = f'(x_2)$.

Для строго выпуклой функции производная $f'(x)$ возрастает на отрезке. Поэтому если значения f_1' и f_2' одного знака, т.е. $f_1'f_2'>0$, то дифференцируемая функция *f(x)* не имеет стационарных точек на отрезке $[x_1,x_2]$ и, следовательно, не имеет на нем точки минимума. Если $f_1'f_2'=0$, то один из концов отрезка является стационарной точкой функции *f(x),* в которой эта функция имеет минимум. Наконец, если $f_1'f_2' < 0$, то для строго выпуклой функции $f_1 < 0$ и $f_2 > 0$. Следовательно, лишь единственная точка $x_* \in (x_1,x_2)$ будет стационарной, и в ней функция *f(x)* достигнет минимума. Таким образом, если $f_1'f_2' < 0$ на отрезке $[x_1,x_2]$, то рассматриваемая функция строго унимодальна на этом отрезке.

Рассмотрим метод поиска точки $x_* \in (x_1,x_2)$ при условии $f_1'f_2'< 0$, называемый ***методом кубической аппроксимации,*** поскольку в этом случае на отрезке $[x_1,x_2]$ можно построить единственный многочлен третьей степени, располагая значениями $f_1,f_2,f_1'f_2'$ на концах этого отрезка. Этот многочлен, называемый *кубическим интерполяционным многочленом Эрмита,* можно преобразовать к виду

$$H_3(x) = f_1 + a_1(x-x_1) + a_2(x-x_1)(x-x_2) + a_3(x-x_1)^2(x-x_2)$$

с коэффициентами

$$\begin{cases} a_1 = \dfrac{f_2-f_1}{x_2-x_1}, \\ a_2 = \dfrac{f_2-f_1}{(x_2-x_1)^2} - \dfrac{f_1'}{x_2-x_1}, \\ a_3 = \dfrac{f_2'+f_1'}{(x_2-x_1)^2} - 2\dfrac{f_2-f_1}{(x_2-x_1)^3}. \end{cases} \tag{3.30}$$

Несложно проверить, что $H_3(x_1)=f_1, H_3(x_2)=f_2, H_3'(x_1)=f_1'$ и $H_3'(x_2)=f_2'$.

Производная $H_3'(x)$ является квадратичной функцией, непрерывной на отрезке $[x_1,x_2]$ и имеющей на его концах различные знаки. Поэтому в интервале она может изменить знак лишь один раз в точке $\tilde{x}_*$, которая является стационарной точкой многочлена $H_3(x)$, а именно точкой его минимума, так как производная меняет знак с минуса на плюс.

Из необходимого условия $H_3'(x)=0$ экстремума этого многочлена получаем с учетом (3.30) квадратное уравнение

$$3a_3(x-x_1)^2 - 2\frac{2f_1'+f_2'-3a_1}{x_2-x_1}(x-x_1) + f_1' = 0.$$

Его решение, принадлежащее интервалу (x_1,x_2), представим в виде $\tilde{x}_* = x_1 + \mu(x_2-x_1)$, где

$$\mu = \frac{\omega+z-f_1'}{2\omega-f_1'+f_2'},\ z = f_1'+f_2'-3\frac{f_2-f_1}{x_2-x_1},\ \omega = \sqrt{z^2-f_1'f_2'}, \tag{3.31}$$

и покажем, что $\mu \in (0,1)$. Действительно, поскольку $f_1'<$ и $f_2'>0$, имеем $\omega > |z|, \omega+z-f_1'+f_2' > 0$. Следовательно,

$$0 < \mu = \frac{\omega + z - f_1'}{2\omega - f_1' + f_2'} < \frac{\omega + z - f_1' + (\omega - z)}{2\omega - f_1' + f_2'} = \frac{\omega - f_1'}{2\omega - f_1' + f_2'} < 1 \text{ и } \tilde{x}_* \in (x_1, x_2).$$

Если $f'(\tilde{x}_*) = 0,$ то $\tilde{x}_* = x_*$ — искомая точка минимума функции *f(x)* на отрезке $[x_1, x_2]$. Если же $f'(\tilde{x}_*) \neq 0,$ то в случае $f_1' f'(\tilde{x}_*) < 0$ можно отбросить отрезок $[\tilde{x}, x_2]$ и продолжать описанным выше способом поиск точки минимума на отрезке $[x_1, \tilde{x}_*]$, и наоборот. После каждого приближения правильность вычислений подтверждается уменьшением минимального значения многочлена по сравнению с его минимальным значением на предыдущем шаге. Вычисления можно прекратить, когда длина интервала неопределенности, в котором гарантированно находится искомая точка x_*, станет меньше заданной наибольшей допустимой величины ε_*.

4 Алгоритмы методов первого и второго порядков

Если ограниченная снизу *целевая функция f(x),* $x \in R^n$, является дифференцируемой на множестве R^n, то алгоритм поиска точки x^* ее минимума

можно построить, используя информацию, по крайней мере, о градиенте этой функции. Более широкие возможности построения таких алгоритмов, возникают в случае, когда целевая функция дважды дифференцируема, что позволяет использовать ее матрицу Гессе. В этой главе применительно к минимизации дифференцируемых и дважды дифференцируемых функций рассмотрены алгоритмы *методов первого и второго порядков* соответственно. Общей чертой этих алгоритмов является построение *релаксационной последовательности* $\{x^k\}$ точек $x^k \in R^n$, получаемых при выполнении каждой k-й итерации, а различие состоит в используемой информации целевой функции и ее производных.

4.1 Алгоритмы метода градиентного спуска

Направление спуска в методе градиентного спуска совпадает с направлением *антиградиента* минимизируемой *целевой функции f(x),* дифференцируемой в R^n, а элементы *релаксационной последовательности* $\{x^k\}$ строят при помощи рекуррентного соотношения вида

$$x^k = x^{k-1} + \chi_k \cdot \omega^k = x^{k-1} - \chi_k \cdot gradf(x^{k-1}), \quad \chi_k > 0, \quad k \in N,$$

где $\omega^k = gradf(x^{k-1})$ – антиградиент целевой функции в точке x^{k-1}. При этом говорят, что из точки x^k на k-й итерации алгоритма происходит спуск с *шагом спуска* $\chi_k \cdot |\omega^k|$. На первой итерации (k=1) спуск начинают из *выбранной начальной точки* x^0. Различие алгоритмов метода градиентного спуска состоит в способе выбора значения χ_k.

Сначала рассмотрим способ выбора этого значения, характерный для *метода дробления шага* и обеспечивающий выполнение на каждой k-й итерации неравенства

$$f(x^{k-1}) - f(x^k) \geq \omega \cdot \chi_k \cdot |\omega^k|^2, \qquad (4.1)$$

где $\omega \in (0,1)$, гарантирующего сходимость последовательности $\{|\omega^k|\}$ к нулю. Если на k-ой итерации не выполнено это неравенство при некотором начальном значении $\chi_k = \chi_0$, то процедуру дробления шага проводят в соответствии с формулой $\chi_k = \nu^i \chi_0$, где $\nu \in (0,1)$ – заданный постоянный ***коэффициент дробления шага***, а i – номер этапа дробления, на котором неравенство (4.1) будет выполнено.

В целом алгоритм с использованием дробления шага можно представить следующим образом. Выбираем начальную точку x^0 и *параметр точности поиска* $\varepsilon_3 > 0$ в условии $|\omega^k| < \varepsilon_3$ прекращения итераций, а также значения ν и χ_0. В неравенстве (4.1) обычно принимают $\omega = 1/2$. Вычисляем значение $f(x^0)$ целевой функции $f(x)$ в точке x^0, полагаем k=1 и переходим к основной части алгоритма.

В точке $x^{k-1} \in R^n$, найденной на (k-1)-й итерации (на первой итерации в точке x^0), вычисляем антиградиент $|\omega^k| < \varepsilon_3$ прекращаем дальнейшие итерации и полагаем $x^* \approx x^{k-1}, f(x) \approx f(x^{k-1})$. В противном случае переходим к п.2

1. Определяем точку $x^k = x^{k-1} + \chi_k \cdot \omega^k$ и значение $f(x^k)$ целевой функции в этой точке. Если неравенство (4.1) ($\omega = 1/2$) выполнено, то полагаем $\chi_k = \chi_0, k := k+1$ и возвращаемся к п.1. В противном случае переходим к п.3.

2. Полагаем $\chi_k := \nu \cdot \chi_k$ и возвращаемся к п.2.

Эффективность алгоритма, использующего спуск в направлении антиградиента, можно несколько повысить, если значение χ_k на каждой k-й итерации выбирать путем минимизации функции

$$\psi_k(\chi) = f(x^{k-1} + \chi \cdot \omega^k), \quad \chi > 0, \tag{4.2}$$

т.е. *применить метод наискорейшего спуска.* Напомним, что для *строго выпуклой функции f(x)* это равносильно процедуре *исчерпывающего спуска*, но в общем случае значение χ, соответствующее наименьшему значению функции $\psi_k(\chi)$, не совпадает со значением, получаемым при помощи исчерпывающего спуска. Отметим, что поиск минимума функции $\psi_k(\chi)$ может оказаться довольно трудоемкой самостоятельной задачей, требующей применения методов одномерной минимизации, но для *сильно выпуклой квадратичной целевой функции* значение χ_k определено однозначно.

Выбор значения χ_k как при помощи исчерпывающего спуска, так и методом наискорейшего спуска гарантирует сходимость последовательности $\{\omega^k\}$ к нулю. Поэтому в качестве условия прекращения итераций и при этом варианте выбора значения χ_k целесообразно использовать неравенство $|\omega^k| < \varepsilon_3$.

Опишем алгоритм наискорейшего спуска в случае произвольной дифференцируемой целевой функции *f(x),* $x \in R^n$. Предварительно выберем начальную точку x^0 и параметр точности поиска ε_3, положим $k = 1$ и перейдем к основной части алгоритма.

1. В точке x^{k-1}, найденной на (k-1)-й итерации (в точке x^0 на первой итерации), находим значение $f(x^{k-1})$ целевой функции и ее антиградиент $\omega^k = -gradf(x^{k-1})$. При выполнении неравенства $|\omega^k| < \varepsilon_3$ дальнейшие итерации прекращаем и полагаем $x^* \approx x^{k-1}, f(x^*) \approx f(x^{k-1})$. В противном случае переходим к п.2.

2. Минимизирую функцию $\psi_k(\chi)$ (4.2), находим значение χ_k и точку $x^k = x^{k-1} + \chi_k \omega^k$, полагаем $k := k+1$ и возвращаемся к п.1.

Алгоритм метода наискорейшего спуска в случае квадратичной функции не дает заметного выигрыша по количеству итераций по сравнению с

алгоритмом, использующим метод дробления шага. Минимизация же неквадратичной функции при сопоставимом количестве итераций, необходимых для достижения заданной точности при помощи этих алгоритмов, может потребовать существенно большего общего объема вычислений по методу наискорейшего спуска вследствие поиска на каждой итерации минимума функции $\psi_k(\chi)$ (4.2). Поэтому необходимое количество итераций еще не является критерием эффективности того или иного алгоритма.

4.2 Метод сопряженных направлений

Среди *методов первого порядка* поиска минимума *целевой функции f(x),* $x \in R^n$, особое место занимает ***метод сопряженных направлений***, основанный на свойствах векторов, сопряженных относительно некоторой квадратной матрицы. Этот метод по сравнению с *методом наискорейшего спуска* требует, как правило, заметно меньшего количества итераций для достижения той же точности поиска при сопоставимом объеме вычислений на каждой итерации. Различие в способах построения системы сопряженных векторов, определяющих *сопряженные направления спуска,* порождает несколько алгоритмов метода сопряженных направлений.

Сначала рассмотрим алгоритм, предназначенный для поиска точки x^* минимума *сильно выпуклой* на *выпуклом множестве* R^n *квадратичной функции* вида

$$f(x) = \frac{1}{2}(Qx, x) + (\tilde{n}, x), \qquad x \in R^n, \tag{4.3}$$

где Q – положительно определенная матрица порядка n, а $\tilde{n} \in R^n$ – заданный вектор. Эта функция имеет единственную точку $x^* = -Q^{-1}c$, в которой она достигает наименьшего значения. Располагая произвольной системой векторов $p^i \in R^n, i = \overline{1,n},$ сопряженных относительно матрицы Q, обратную к ней матрицу Q^{-1} можно вычислить при помощи соотношения

$$Q^{-1} = \sum_{i=1}^{n} \frac{p^i (p^i)^T}{(Qp^i, p^i)}$$

Но построение системы сопряженных векторов представляет собой самостоятельную и в общем случае достаточно сложную и трудоемкую в вычислительном отношении задачу. Особенность алгоритмов метода сопряженных направлений состоит в том, что систему сопряженных векторов строят последовательно в процессе выполнения итераций, причем найденный на очередной, k-й итерации вектора $p^k \in R^n$ определяет на этой итерации направление спуска.

Выберем начальную точку $x^0 \in R^n$, найдем на этой точке антиградиент $\omega^1 = -gradf(x^0) = -Qx^0 - c$ и положим $p^1 = \omega^1$. Ясно, что если $|p^1| \neq 0,$ то вектор p^1 определяет на первой (k=1) итерации направление спуска (в противном случае точка x^0 удовлетворяет необходимому условию минимума, а это в случае

функции (4.3) равносильно тому, что $x^0 = x^*$ — единственная точка наименьшего значения этой функции). Считая, что $|p^1| \neq 0,$ и проводя исчерпывающий спуск находим значение

$$\chi_1 = \frac{|\omega^1|^2}{(Q\omega^1, \omega^1)} = \frac{|p^1|^2}{(Qp^1, p^1)} \neq 0$$

и точку $x^1 = x^0 + \chi_1 p^1$. Таким образом, первая итерация полностью совпадает с итерацией алгоритма наискорейшего спуска.

На второй итерации (k=2) вычислим антиградиент $\omega^2 = -gradf(x^1) = -Qx^1 - c$ в найденной точке x^1. Пусть $|\omega^2| \neq 0,$ так как в противном случае $x^1 = x^*$. Положим

$$p^2 = \gamma_1 p^1 + \omega^2 \qquad (4.4)$$

и, умножив это равенство скалярно на вектор $Qp^1 \neq 0,$ запишем $(Qp^1, p^2) = (Qp^1, \gamma_1 p^1 + \omega^2)$. Потребуем, чтобы векторы p^1 и p^2 были сопряженными относительно матрицы Q, т.е. чтобы выполнялось равенство $(Qp^1, p^2) = 0$. Тогда $\gamma_1 = -\frac{(Qp^1, \omega^2)}{(Qp^1, \omega^1)}$. Убедимся, что $|p^2| \neq 0$. В самом деле, при $|p^2| = 0$ имеем $\omega^2 = -\gamma_1 p^1$ и после умножения скалярно на вектор $\omega^2 \neq 0_n$ приходим к противоречию:

$$(\omega^2, \omega^1) = |\omega^2|^2 = -\gamma_1 (\omega^2, p^1) = (\omega^2, \omega^1) = 0,$$

Поскольку при исчерпывающем спуске антиградиенты на двух следующих друг за другом итерациях ортогональны. Вектор p^2 задает направление спуска из точки x^1, так как с учетом (4.4)

$$(gradf(x^1), p^2) = -(\omega^2, \gamma_1 p^1 + \omega^2) = -\gamma_1 (\omega^2, p^1) - (\omega^2, \omega^2) = -|\omega^2|^2 < 0.$$

Если в направлении вектора p^2 провести исчерпывающий спуск, то теперь при нахождении точки $x^2 = x^1 + \chi_2 p^2$ направление спуска из точки x^1 не совпадает с направлением антиградиента в этой точке, так как в общем случае $\gamma_1 \neq 0$. Вычислим в точке x^2 антиградиент

$$\omega^3 = -gradf(x^2) = -Qx^2 - c = -Qx^1 - \chi_2 Qp^2 - c = \omega^2 - \chi_2 Qp^2$$

и после умножения скалярно на вектор p^2 получим $(\omega^3, p^2) = (\omega^2 - \chi_2 Qp^2, p^2) = 0$, откуда

$$\chi_2 = \frac{(\omega^2, p^2)}{(Qp^2, p^2)} = -\frac{(gradf(x^1), p^2)}{(Qp^2, p^2)}. \qquad (4.5)$$

Так как $\omega^3 - \omega^2 = -Qx^2 - c + Qx^1 + c = -Q(x^2 - x^1) = -\chi_2 Qp^2,$ то с учетом $\omega^1 = p^1$ имеем $(\omega^2, \omega^1) = 0,$ а $(Qp^2, p^1) = (Qp^1, p^2) = 0$ в силу симметричности матриц Q и Q-ортогональности векторов p^1 и p^2. Следовательно, при исчерпывающем спуске в пространстве R^n, n>2, антиградиенты ортогональны на первых трех итерациях.

Продолжая описанную процедуру построения сопряженных векторов, с помощью метода математической индукции можно показать, что на любой k-й итерации при $k = \overline{1, n-1}$

$$\begin{cases} p^{k+1} = \gamma_k p^k + \omega^{k+1}, \\ \gamma_k = -\dfrac{(Qp^k, \omega^{k+1})}{(Qp^k, p^k)}, \\ \omega^{k+1} = \omega^k - \chi_k Qp^k, \\ (Qp^{k+1}, p^i) = 0, \\ (\omega^{k+1}, \omega^i) = 0, i = \overline{1,k}, \end{cases} \tag{4.6}$$

и при этом $\chi_k = \dfrac{(\omega^k, p^k)}{(Qp^k, p^k)}$ и $x^k = x^{k-1} + \chi_k p^k$. Таким образом, эта процедура за n итераций позволяет построить систему векторов $p^i \in R^n, i = \overline{1,n},$ сопряженных относительно матрицы Q и линейно независимых. Антиградиенты $\omega^k = -gradf(x^{k-1}), k \in N$, в соответствии с (4.6) образуют ортогональную систему векторов. Так как в R^n не может быть более n ненулевых ортогональных векторов, то один из антиградиентов $\omega^k, k = \overline{1,n+1}$, с номером i должен быть нулевым вектором, т.е. $\omega^i = -gradf(x^{i-1}) = 0_n$. Поэтому точка $x^* = x^{i-1}$ минимума функции *f(x)* будет найдена не более чем за *n* итераций.

Чтобы использовать метод сопряженных направлений для поиска минимума неквадратичной дифференцируемой функции, предварительно необходимо привести систему (4.6) к виду, не содержащему матрицу Q. Это можно сделать, если в итерационном процессе

$$x^k = x^{k-1} + \chi_k p^k, \; k \in N,$$

с выбором значения $\chi_k > 0$ на каждой *к-й* итерации путем минимизации функции

$$\psi_k(\chi) = f(x^{k-1} + \chi p^k), \chi > 0,$$

в основу построения сопряженных направлений $p^{k+1} = \gamma_k p^k + \omega^{k+1}$ положить свойство ортогональности антиградиентов (а значит, и градиентов) минимизируемой функции

$$(\omega^{k+1}, \omega^i) = 0, i = \overline{1,k}.$$

Такой способ построения сопряженных направлений спуска в литературе получил название ***метода сопряженных градиентов.***

Используя последнее уравнение (4.6) при *i=k,* третье уравнение, числитель и знаменатель второго уравнения (4.6) можно представить в виде

$$\begin{cases} (Qp^k, \omega^{k+1}) = \dfrac{(\omega^{k+1} - \omega^k, \omega^{k+1})}{\chi_k} = -\dfrac{1}{\chi_k}\left|\omega^{k+1}\right|^2, \\ (Qp^k, p^k) = \dfrac{(\omega^{k+1} - \omega^k, p^k)}{\chi_k} = \dfrac{1}{\chi_k}(\omega^k, p^k). \end{cases} \tag{4.7}$$

Таким образом, вместо второго уравнения (4.6) имеем $\gamma_k = \frac{\left|\omega^{k+1}\right|^2}{(\omega^k, p^k)}$. Это выражение можно преобразовать, если учесть первое уравнение (4.6), заменив в нем k на k-1:

$$\gamma_k = \frac{\left|\omega^{k+1}\right|^2}{(\omega^k, \gamma_{k-1} p^k + \omega^k)} = \frac{\left|\omega^{k+1}\right|^2}{\gamma_{k-1}(\omega^k, p^{k-1}) + \left|\omega^k\right|^2}.$$

Но если заменить k на k-1 , то получим $(\omega^k, p^{k-1}) = 0.$ Следовательно,

$$\gamma_k = \frac{\left|\omega^{k+1}\right|^2}{\left|\omega^k\right|^2}, k \in N. \qquad (4.8)$$

Если преобразования в первом равенстве (4.7) не доводить до конца, то вместо (4.8) получим

$$\gamma_k = \frac{(\omega^{k+1} - \omega^k, \omega^{k+1})}{\left|\omega^k\right|^2}, k \in N \qquad (4.9)$$

В случае дважды дифференцируемой целевой функции можно получить еще одно выражение для γ_k. Так как матрица Гессе функции $\frac{(Qx, x)}{2} + (c, x)$ совпадает с матрицей Q, то во втором равенстве (4.6) можно заменить Q матрицей Гессе $H(x^k)$ неквадратичной целевой функции:

$$\gamma_k = -\frac{(H(x^k)p^k, \omega^{k+1})}{(H(x^k)p^k, p^k)}, k \in N \qquad (4.10)$$

Теперь с помощью одной из формул (4.8) – (4.10) из первого равенства (4.6), записанного в виде

$$p^{k+1} = \gamma_k p^k + \omega^{k+1}, k \in N, \quad p^1 = \omega^1 \qquad (4.11)$$

на любой k-й итерации можно найти вектор, определяющий направление спуска из точки x^k на $(k+1)$-й итерации. Для квадратичной функции результат вычислений по каждой из формул (4.8)-(4.10) будет одинаковым, но для неквадратичной функции значения γ_k будут, вообще говоря, различными. Поэтому использование каждой из этих формул при минимизации неквадратичной функции $f(x)$ может привести к построению своей системы направлений спуска. При этом в отличие от квадратичной функции точка минимума в общем случае не будет найдена за конечное число итераций.

Более того, при выборе значения $\chi_k > 0$ на каждой k-й итерации в рекуррентном соотношении $x^k = x^{k-1} + \chi_k p^k$ приходится минимизировать функцию

$$\psi_k(\chi) = f(x^{k-1}) + \chi p^k), \chi > 0, k \in N \qquad (4.12)$$

Это приводит к неизбежным погрешностям на каждой итерации, которые могут вызвать нарушение сходимости алгоритма. Чтобы ослабить

влияние погрешностей, используют процедуру «обновления» алгоритма, состоящую в том, что в (4.11) периодически через заданное число итераций полагают γ_k=0. Соответствующий номер итерации называют ***моментом обновления алгоритма,*** или ***рестартом.*** Обычно при минимизации целевой функции в R^n множество таких номеров имеет вид $I_0 = \{n, 2n, ..., mn, ...\}, m \in N$. Использование в алгоритме рестартов позволяет избежать накопления вычислительных погрешностей и уменьшить вероятность построения после каждых *n* итераций линейно зависимых направлений спуска, но приводит к росту общего числа итераций, необходимых для достижения заданной точности поиска.

Опишем алгоритм минимизации дифференцируемой неквадратичной целевой функции *f(x)*. Выбираем *параметр* $\varepsilon_3 > 0$ *точности поиска* и начальную точку $x^0 \in R^n$, в которой вычисляем антиградиент $\omega^1 = -gradf(x^0)$. Убедившись, что $|\omega^1| > \varepsilon_3$, формируем множество *Io* моментов обновления алгоритма, полагаем *k*= 1, принимаем $p^1 = \omega^1$ и переходим к основной части алгоритма.

1. Минимизируя функцию $\psi_k(\chi)$ (4.12), находим значение χ_k и точку $x^k = x^{k-1} + \chi_k p^k$. В этой точке вычисляем антиградиент $\omega^{k+1} = -gradf(x^k)$ и проверяем выполнение неравенства $|\omega^{k+1}| < \varepsilon_3$. Если оно выполнено, то итерации прекращаем и полагаем $x^* \approx x^{k-1}, f(x^*) \approx f(x^{k-1})$. В противном случае переходим к п. 2.

2. Если $k \in I_0$, то полагаем $p^{k+1} = \omega^{k+1}, k := k+1$ и возвращаемся к п. 1. В противном случае переходим к п. 3.

3. При помощи (4.8) или (4.9) вычисляем значение γ_k и, используя (4.11), находим вектор p^{k+1}. Полагаем *k:=k + 1* и возвращаемся к п. 1.

4.3 Метод Ньютона

Если *целевая функция f(x)* является дважды дифференцируемой в R^n, то эффективность процесса поиска точки x^* ее минимума можно повысить, используя информацию не только о градиенте *gradf(x)* этой функции, но и о ее матрице Гессе *H(x).* Алгоритмы такого поиска обычно относят к *методу Ньютона.* В простейшем варианте алгоритма на каждой *k-й* итерации целевая функция аппроксимируется в окрестности точки x^{k-1} (на первой итерации в окрестности *начальной точки* x^0*) квадратичной функцией* $\psi_k(x)$ и затем определяется точка $x^к$ минимума функции $\psi_k(x)$. На следующей, *(k+1)*-й итерации строится новая квадратичная аппроксимация уже в окрестности точки $x^к$.

При помощи формулы Тейлора с остаточным членом в форме Пеано представим целевую функцию в окрестности точки x^{k-1} в виде

$$f(x) = f(x^{k-1}) + (gradf(x^{k-1}), x - x^{k-1}) + \frac{1}{2}(H(x^{k-1})(x - x^{k-1}), x - x^{k-1}) + o(|x - x^{k-1}|^2).$$

Пренебрегая последним слагаемым в правой части, получаем квадратичную функцию

$$\psi_k(x) = f(x^{k-1}) + (gradf(x^{k-1}), x - x^{k-1}) + \frac{1}{2}(H(x^{k-1})(x - x^{k-1}), x - x^{k-1}).$$

Если матрица Гессе $H(x^{k-1})$ целевой функции, вычисленная в точке x^{k-1}, является положительно определенной, то точка x^k минимума функции $\psi_k(x)$ единственна и может быть найдена из условия, что ее градиент равен нулевому вектору:

$$grad\,\psi_k(x) = gradf(x^{k-1}) + H(x^{k-1})(x - x^{k-1}) = 0_n.$$

Отсюда получаем

$$x^k = x^{k-1} - H^{-1}(x^{k-1})gradf(x^{k-1}), k \in N. \quad (4.13)$$

Отметим, что (4.13) можно трактовать как рекуррентное соотношение итерационной процедуры метода Ньютона, предназначенного для решения системы $gradf(x)=0_n$, нелинейных уравнений. Этим можно объяснить и название метода, применяемого для минимизации целевой функции $f(x)$.

Если размерность n пространства R^n велика, то вычисление на каждой k-й итерации матрицы $H^{-1}(x^{k-1})$, обратной к матрице Гессе целевой функции, может оказаться весьма трудоемкой процедурой. В таком случае точку $x^к$ целесообразнее искать путем минимизации функции $\psi_k(x)$, например *методом сопряженных направлений* или *методом градиентов.* Подробнее этот вопрос рассмотрен ниже.

Точку минимума функции $\psi_k(x)$ можно считать лишь вспомогательным приближением и, обозначив эту точку $\tilde{x}^k$, для построения *релаксационной последовательности* $\{x^к\}$ использовать рекуррентное соотношение

$$x^k = x^{k-1} + \chi_k(\tilde{x}^k - x^{k-1}) = x^{k-1} + \chi_k p^k, k \in N, \quad (4.14)$$

в котором значение $\chi_k > 0$ можно выбрать различными способами. Вектор $p^k = \tilde{x}^k - x^{k-1}$ задает на k-й итерации *направление спуска.* Если матрица $H(x^{k-1})$ положительно определенная, то это *ньютоновское направление спуска.* Действительно, с учетом (4.13) и (4.14) этот вектор можно представить в виде

$$p^k = -H^{-1}(x^{k-1})gradf(x^{k-1}), k \in N, \quad (4.15)$$

так что

$$(gradf(x^{k-1}), p^k) = -(gradf(x^{k-1}), H^{-1}(x^{k-1})gradf(x^{k-1})) < 0.$$

Геометрически это означает, что вектор $p^к$ образует тупой угол с градиентом целевой функции, вычисленным в точке x^{k-1} (рисунок 4.1).

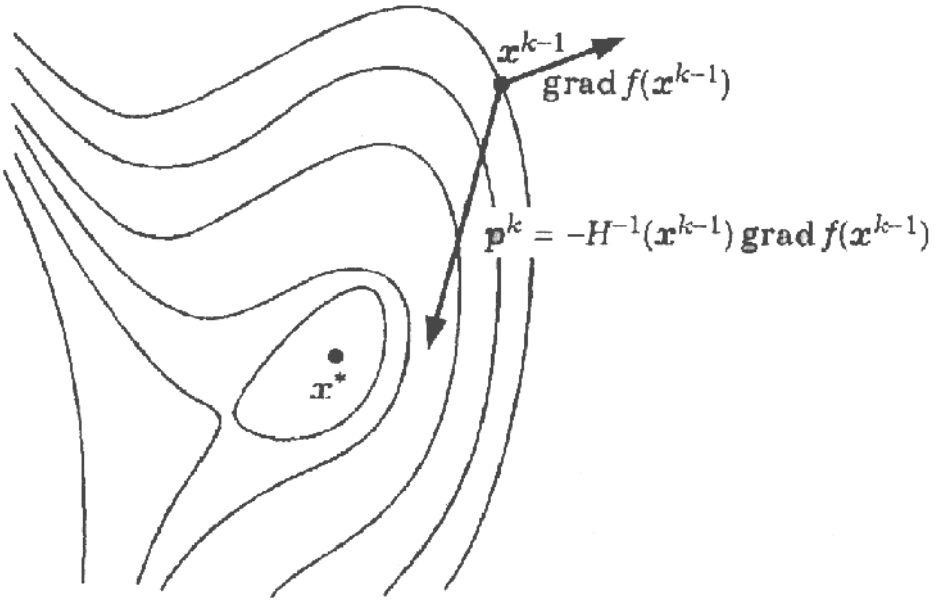


Рисунок 4.1—Траектория движения к экстремуму методом Ньютона

Ясно, что если в (4.14) выбрать $\chi_k = 1$, то (4.13) и (4.14) будут равносильны. Однако значение $\chi_k > 0$ можно найти как точку наименьшего значения функции $\psi_k(\chi) = f(x^{k-1} + \chi p^k)$ или же при помощи *исчерпывающего спуска* в направлении вектора p^k. Для выбора значения x_k можно также использовать *метод дробления шага.*

Если целевая функция является квадратичной вида (4.3) с положительно определенной матрицей Q, то исчерпывающий спуск из произвольной начальной точки $x^0 \in R^n$ в ньютоновском направлении приводит в точку x^* минимума этой функции за одну итерацию. Для неквадратичной функции ньютоновское направление в общем случае не проходит через точку ее минимума, хотя часто оказывается к этой точке ближе, чем направление *антиградиента.* Если это так, то спуск в ньютоновском направлении за одну итерацию позволяет достичь более существенного убывания минимизируемой неквадратичной функции и получить более близкое приближение к искомой точке x^* минимума по сравнению со спуском в направлении антиградиента.

Если график целевой функции имеет *овражную структуру,* то вектор p^k (4.15) может составлять с осью оврага меньший угол, чем антиградиент. Эта особенность при минимизации таких функций делает алгоритмы метода Ньютона более эффективными, чем алгоритмы *метода градиентного спуска.*

Сходимость метода Ньютона существенно зависит от выбора начального приближения. Можно показать, что если целевая *функция сильно выпуклая* и для любых точек $x, y \in R^n$ относительно матрицы Гессе *H(x)* целевой функции выполнено неравенство $\|H(x) - H(y)\| \le L|x - y|, L > 0$, а начальное приближение выбрано удачно (точка x^0 расположена достаточно близко к точке x^* минимума), то алгоритм метода Ньютона при значении χ_k=1 в (4.14) обладает *квадратичной скоростью сходимости,* т.е. справедлива оценка

$$\left|x^k - x^*\right| \le C\left|x^{k-1} - x^*\right|^2, k \in N, C = const.$$

Это позволяет для такой функции использовать в качестве условия прекращения итераций выполнение первого неравенства $\left|x^k - x^{k-1}\right| < \varepsilon_1$, где ε_1 – заданное достаточно малое положительное число. Но в случае, когда целевая функция не является сильно выпуклой или же начальное приближение находится далеко от искомой точки *x**, метод Ньютона может расходиться.

Квадратичная скорость сходимости, а также возможность контроля за соблюдением достаточного условия минимума целевой функции *f(x)* на каждой *k*-й итерации при помощи матрицы Гессе $H(x^{k-1})$ этой функции способствуют высокой эффективности рассматриваемого алгоритма. Однако при его практической реализации возникают две проблемы.

Первая из них — это сохранение положительной определенности матрицы Гессе $H(x^{k-1})$ целевой функции на каждой *k-й* итерации, так как иначе вектор $p^k = -H^{-1}(x^{k-1})gradf(x^{k-1})$ может не соответствовать направлению спуска, вдоль которого эта функция убывает. Более того, матрица $H(x^{k-1})$ может быть вырожденной и не иметь обратной матрицы. Эту проблему можно решить, если направление спуска задавать вектором $\tilde{p}^k = -(\eta_k I_n + H(x^{k-1}))^{-1} gradf(x^{k-1})$, где I_n — единичная матрица порядка *n,* а $\eta_k \ge 0$ – параметр, выбираемый так, чтобы в точке x^{k-1} матрица

$$\tilde{H}_k = \eta_k I_n + H(x^{k-1}), k \in N \qquad (4.16)$$

была положительно определена.

В связи с этим более серьезной проблемой является необходимость вычисления и обращения на каждой итерации матрицы порядка *n,* что в случае большой размерности пространства R^n является достаточно трудоемкой операцией. На практике обычно не вычисляют матрицу, обратную к положительно определенной матрице $\tilde{H}_k$, а вектор p^k находят из решения системы линейных алгебраических уравнений (СЛАУ)

$$\tilde{H}_k p_k = -gradf(x^{k-1}), k \in N. \qquad (4.17)$$

Эту СЛАУ можно решить различными численными методами, например прямыми и итерационными. Можно также решать СЛАУ путем минимизации квадратичной функции $(\tilde{H}_k p^k, p^k)/2 + (gradf(x^{k-1}), p^k)$ методами сопряженных направлений или градиентов, поскольку выполнение (4.17) является необходимым и достаточным условием минимума такой функции.

Опишем алгоритм варианта метода Ньютона поиска точки *x** минимума дважды дифференцируемой целевой функции $f(x), x \in R^n$, в котором направление спуска определяется путем решения СЛАУ (4.17). Предварительно выбираем начальную точку $x^0 \in R^n$ и значение $\varepsilon_3 > 0$ в условии $\left|gradf(x^{k-1})\right| < \varepsilon_3$ прекращения итераций. Полагаем *k* = 1 и переходим к основной части алгоритма.

1. В точке x^{k-1} находим $gradf(x^{k-1})$ и матрицу Гессе $H(x^{k-1})$ целевой функции. Если $\left|gradf(x^{k-1})\right| < \varepsilon_3$, то итерации прекращаем, принимая $x^* \approx x^{k-1}$ и

$f(x^*) \approx f(x^{k-1})$. Если при этом матрица $H(x^{k-1})$ положительно определенная, то *x** — точка минимума целевой функции, а иначе необходимо провести дополнительное исследование поведения функции в окрестности точки *x**. При $\left|gradf(x^{k-1})\right| \geq \varepsilon_3$ и положительно определенной матрице $H(x^{k-1})$ полагаем $\tilde{H}_k = H(x^{k-1})$ и переходим к п. 3. В противном случае переходим к п. 2.

2. Подбираем значение $\eta_k > 0$, при котором матрица $\tilde{H}_k$ (4.16) будет положительно определенной, и переходим к п. 3.

3. Из решения СЛАУ (4.17) находим вектор p^k и затем точку $x^k = x^{k-1} + p^k$. Полагаем *k:=k+1* и возвращаемся к п. 1.

4.4 Модификация метода Ньютона

Напомним, что в *методе Ньютона* для построения *релаксационной последовательности* $\{x^k\}$ при поиске минимума дважды непрерывно дифференцируемой и ограниченной снизу в R^n *целевой функции f(x)* используют рекуррентное соотношение вида (4.14)

$$x^k = x^{k-1} + \chi_k p^k,$$

где p^k — вектор, задающий на *k-й* итерации *ньютоновское направление спуска* из точки x^{k-1} (на первой итерации из *начальной точки* x^0). Выше (см. 4.3) перечислены некоторые способы выбора значения χ_k. Обычно собственно метод Ньютона связывают с таким вариантом выбора этого значения, когда на *k-я* итерации принимают $\chi_k p^k = -H^{-1}(x^{k-1}) gradf(x^{k-1})$, где $H(x^{k-1})$ — матрица Гессе целевой функции с элементами, вычисленными в точке x^{k-1}. Если принять во внимание (4.15), то следует считать, что в методе Ньютона $\chi_k = 1$.

Одна из модификаций метода Ньютона поиска точки *x** минимума функции *f(x)* связана с применением для выбора значения χ_k *метода дробления шага,* при котором на каждой итерации выполняется неравенство

$$f(x^{k-1}) - f(x^k) \geq -\omega\chi_k (gradf(x^{k-1}), p^k), k \in N, \qquad (4.18)$$

где $\omega \in (0,1/2)$. Если на *k-й* итерации это неравенство не выполнено при начальном значении $\chi_k = \chi_0 = 1$, то процедуру дробления шага проводят в соответствии с формулой $\chi_k = \nu^i$, где $\nu \in (0,1)$ — выбранный постоянный *коэффициент дробления шага,* а *i* — номер этапа дробления, на котором будет выполнено (4.18).

Алгоритм с дроблением шага и нахождением направления спуска путем решения системы линейных алгебраических уравнений (СЛАУ) (4.17) можно представить следующим образом.

На предварительном этапе выбираем начальную точку $x^0 \in R^n$, *параметр* $\varepsilon_3 > 0$ *точности поиска* в условии $\left|gradf(x^{k-1})\right| < \varepsilon_3$ прекращения итераций, коэффициент и дробления шага и значение $\omega \in (0,1/2)$ в (4.18). Вычисляем

значение $f(x^0)$ целевой функции $f(x)$ в точке x^0, полагаем $k = 1$, $\chi_k = 1$ и переходим к основной части алгоритма.

1. В точке x^{k-1}, найденной на $(k-1)$-й итерации (на первой итерации в точке x^0), вычисляем градиент $gradf(x^{k-1})$ и матрицу Гессе $H(x^{k-1})$ целевой функции. Если $\left|gradf(x^{k-1})\right| < \varepsilon_3$, то итерации прекращаем, принимая $x^* \approx x^{k-1}$ и $f(x^*) \approx f(x^{k-1})$. В противном случае переходим к п. 2.

2. Из решения СЛАУ (4.17) находим вектор p^k, точку $x^k = x^{k-1} + \chi_k p^k$ и значение $f(x^k)$ в этой точке. Если при этом выполнено неравенство (4.18), то полагаем $\chi_k = 1$, $k := k + 1$ и возвращаемся к п.1. В противном случае переходим к п. 3.

3. Полагаем $\chi_k := \nu\chi_k$ и возвращаемся к п. 2.

Другой способ выбора значения χ_k в (4.14) основан на применении на каждой k-й итерации *исчерпывающего спуска* в направлении, определяемом вектором p^k, т.е. связан с минимизацией функции $\psi_k(\chi) = f(x^{k-1} + \chi p^k)$. Общая трудоемкость этого способа может оказаться значительной вследствие необходимости на каждой итерации для поиска минимума этой функции использовать один из методов одномерной минимизации. Алгоритм, в котором реализуется такой способ выбора значения χ_k, отличается от описанного выше тем, что во втором пункте алгоритма после нахождения вектора p^k минимизируется функция $\psi(\chi)$, а затем полученное значение χ_k используется для вычисления точки $x^k = x^{k-1} + \chi_k p^k$. После этого осуществляется переход к первому пункту алгоритма. Отметим, что на предварительном этапе нет необходимости задавать значения ν, ω и $\chi_k = 1$.

Отметим, что рассмотренные модификации метода Ньютона менее чувствительны к выбору начальной точки x^0 по сравнению с «классическим» вариантом этого метода, соответствующим значению $\chi_k = 1$ в (4.14). Можно показать, что если целевая *функция* является *сильно выпуклой* и ее матрица Гессе $H(x)$ для любых точек $x, y \in R^n$ удовлетворяет неравенству

$$\left\|H(x) - H(y)\right\| \le L\left|x - y\right|, \tag{4.19}$$

то при произвольном выборе начальной точки обе рассмотренные выше модификации метода Ньютона обладают *квадратичной скоростью сходимости.* При этом справедлива оценка

$$\left|x^k - x^*\right| \le \frac{L}{\lambda_1}\left|x^{k-1} - x^*\right|^2, k \in N, \tag{4.20}$$

где λ_1 — оценка снизу наименьшего из собственных значений матрицы Гессе $H(x)$ целевой функции в окрестности точки x^*, содержащей точку x^{k-1}, если в этой окрестности матрица Гессе является положительно определенной, т.е. $\lambda_1\left|x\right|^2 \le (H(x)x, x)$ для всех точек ж из этой окрестности.

Ясно, что попытка реализовать рассмотренные алгоритмы наталкивается на те же проблемы, что и при выборе $\chi_k = 1$ в (4.14) (см. 4.3). Главная из них состоит в том, что на каждой *k-й* итерации необходимо вычислять и обращать матрицу Гессе $H(x^{k-1})$ целевой функции или же искать направление спуска путем решения СЛАУ (4.17), где матрица $\widetilde{H}_k$ определяется равенством (4.16). Один из способов разрешения этой проблемы заключается в использовании для выбора направления спуска вместо (4.15) соотношения

$$p^k = -H^{-1}(x^0)gradf(x^{k-1}), k \in N. \quad (4.21)$$

Если в начальной точке $x°$ матрица Гессе *H(x°)* целевой функции не является положительно определенной, то в (4.21) вместо *H(x°)* можно использовать матрицу $\widetilde{H}_1 = \eta_1 I_n + H(x^0)$, вычисленную в соответствии с (4.16) при $k = 1$.

Оказывается, что использование (4.21) приводит к *линейной скорости локальной сходимости,* причем справедлива оценка

$$\left|x^k - x^*\right| \le q\left|x^{k-1} - x^*\right|, k \in N, q = const. \quad (4.22)$$

Выбор начальной точки $x°$ влияет на знаменатель q геометрической прогрессии: чем ближе $x°$ к искомой точке x^*, тем меньше значение q.

Другой способ разрешения указанной выше проблемы состоит в периодическом «обновлении» матрицы Гессе в (4.21). В этом случае на первых *K* итерациях в (4.21) используют матрицу *H(x°)* (или $\widetilde{H}_1$, если матрица *H(x°)* не является положительно определенной), а на *(K+1)-й* итерации ее заменяют матрицей $H(x^K)$ (или $\widetilde{H}_{K+1}$, если матрица $H(x^K)$ не окажется положительно определенной матрицей) и т.д. Выбор значения *K* зависит от целевой функции. Если целевая функция сильно выпукла и удовлетворяет условию (4.19), то алгоритм с «обновлением» матрицы Гессе при произвольном выборе начальной точки будет обладать *сверхлинейной скоростью сходимости.*

При этом для любого из рассмотренных выше способов выбора в (4.14) значения χ_k справедлива оценка

$$\left|x^{kK} - x^*\right| \le C\left|x^{(k-1)K} - x^*\right|^{K+1}, k \in N, C = const.$$

При *K = 1,* т.е. при «обновлении» матрицы Гессе на каждой итерации, эта оценка равносильна (4.20), а при довольно редком «обновлении», т.е. при больших значения *K,* она приближается к (4.22).

4.5 Квазиньютоновские методы

Среди алгоритмов многомерной минимизации следует выделить группу алгоритмов, которые объединяют достоинства *метода наискорейшего спуска* и *метода Ньютона.* Такие алгоритмы принято относить к так называемым ***квазиньютоновским методам.*** Особенность этих алгоритмов состоит в том, что при их применении нет необходимости вычислять и обращать матрицу

Гессе *целевой функции f(x)* и в то же время удается сохранить высокую скорость сходимости алгоритмов, присущую методу Ньютона и его модификациям (см. 4.3 и 4.4).

Элементы *релаксационной последовательности* $\{x^к\}$ в алгоритмах квазиньютоновских методов минимизации непрерывно дифференцируемой в R^n целевой функции *f(x)* строят в соответствии с рекуррентным соотношением $x^k = x^{k-1} + \chi_k p^k$, но *направление спуска* на каждой *k-й* итерации задают в виде

$$p^k = -A_k\, gradf(x^{k-1}) = A_k \omega^k, k \in N \qquad (4.23)$$

Здесь $\omega^k = -gradf(x^{k-1})$ — *антиградиент* целевой функции в точке x^{k-1}, а A_k — положительно определенная матрица порядка n, обновляемая на *k-й* итерации. Отметим, что направление, задаваемое на каждой *k-й* итерации вектором $p^к$ (4.23), является направлением спуска, так как с учетом (4.23)

$$(gradf(x^{k-1}), p^k) = -(\omega^k, A_k \omega^k) < 0.$$

Матрицы $\{A_k\}$ определяют таким образом, чтобы последовательность $\{A_к\}$ при $k \to \infty$ имела предел, равный $H^{-1}(x^*)$, где $H(x^*)$— матрица Гессе целевой функции, вычисленная в точке минимума *x** этой функции.

На *k-й* итерации алгоритма поиска точки минимума происходит спуск из точки x^{k-1} с *шагом спуска* $\chi_k \left| p^k \right|$, причем значение χ_k выбирают путем *исчерпывающего спуска* в направлении вектора $p^к$. На первой итерации *(k = 1)* спуск начинают из выбранной *начальной точки x°* и при этом обычно в качестве A_1 берут единичную матрицу I_n порядка *n.* Отметим, что в этом случае $p^1 = \omega^1 = -gradf(x^0)$, т.е. первая итерация алгоритма квазиньютоновского метода полностью совпадает с итерацией метода наискорейшего спуска (см. 4.1). Если принять $A_k = H^{-1}(x^{k-1})$ в (4.23) и χ_k = 1 в (4.14), то придем к «классическому» методу Ньютона (см. 4.3).

Как было отмечено выше (см. 4.3), если целевая *функция* является *сильно выпуклой,* то алгоритмы метода Ньютона обладают *квадратичной скоростью сходимости.* Поэтому можно ожидать, что алгоритмы квазиньютоновских методов будут иметь достаточно высокую скорость сходимости, если на каждой *k-й* итерации матрица $A_к$ выбрана близкой к матрице $H^{-1}(x^{k-1})$ в точке $x^{k-1} \in R^n$. Используя при конструировании матрицы $A_к$ аппроксимацию матрицы $H^{-1}(x^{k-1})$ с учетом информации о градиенте целевой функции в той же точке x^{k-1}, можно существенно упростить процедуру нахождения направления спуска на *k-й* итерации. Именно эти соображения лежат в основе построения алгоритмов квазиньютоновских методов.

Остановимся на вопросе построения последовательности матриц A_k. Эти матрицы строят в соответствии с рекуррентным соотношением

$$A_{k+1} = A_k + \Delta A_k, k \in N, \qquad (4.24)$$

где ΔA_k — положительно определенная матрица порядка *n,* называемая ***поправочной***. Способ ее выбора определяет вариант алгоритма

квазиньютоновского метода. На первой итерации, как уже отмечалось, обычно принимают $A_1 = I_n$.

Предположим, что $f(x) = \frac{1}{2}(Qx, x) + (c, x)$ — *квадратичная функция* с положительно определенной матрицей Q. В этом случае *функция f(x) сильно выпукла,* $gradf(x) = Qx + c$,

$$\Delta\omega^k = gradf(x^{k-1}) - gradf(x^k) = Q(x^{k-1} - x^k),$$

и мы можем записать

$$Q^{-1}\Delta\omega^k = -\Delta x^k, k \in N, \tag{4.25}$$

где матрица Q совпадает с матрицей Гессе *H(x)* функции *f(x).*

В общем случае, проводя аналогию с (4.25), потребуем, чтобы матрица A_{k+1} в (4.24) удовлетворяла так называемому ***квазиньютоновскому условию***

$$A_{k+1}\Delta\omega^k = -\Delta x^k, k \in N, \tag{4.26}$$

а поправочную матрицу ΔA_k в (4.24) выберем, исходя из условия

$$\lim_{k->\infty} \left\| A_{k+1} - H^{-1}(x^k) \right\| = 0,$$

т.е. так, чтобы последовательность $\{A_к\}$ имела при $k \to \infty$ предел, равный $H^{-1}(x^*)$.

Подставляя (4.24) в (4.26), получаем $\Delta A_k \Delta\omega^k = -\Delta x^k - A_k \Delta\omega^k$. Непосредственной подстановкой можно убедиться, что для любого $k \in N$, одним из решений этого уравнения будет матрица

$$\Delta A^k = -\frac{\Delta x^k y^T}{(\Delta\omega^k, y)} - \frac{A_k \Delta\omega^k z^T}{(\Delta\omega^k, z)},$$

где $y, z \in R^n$ — произвольные элементы из R^n. Полагая $y = \Delta x^k$ и $z = A_k\omega^k$, получаем с учетом (4.24) формулу

$$A_{k+1} = A_k - \frac{\Delta x^k (\Delta x^k)^T}{(\Delta\omega^k, \Delta x^k)} - \frac{A_k \Delta\omega^k (\Delta\omega^k)^T A_k^T}{(A_k \Delta\omega^k, \Delta\omega^k)}, k \in N, \tag{4.27}$$

для выбора элементов последовательности $\{A_к\}$. Такой способ выбора матриц $A_к$ приводит к одному из алгоритмов квазиньютоновского типа, известного в литературе как метод Давидона — Флетчера — Пауэлла, или ДФП — метод.

Рассмотренный способ «обновления» матрицы A_{k+1} сохраняет ее свойство положительной определенности на каждой *k-й* итерации. Для того чтобы убедиться в этом, воспользуемся методом математической индукции. Очевидно, что матрица A_1 положительно определенная, если принять $A_1 = I_n$. Предположим, что при $k > 1$ матрица $A_к$ также положительно определенная, и докажем, что матрица A_{k+1} в (4.27) положительно определенная.

Выберем ненулевой вектор $y \in R^n$. Тогда с учетом (4.27) можно записать

$$(A_{k+1}y, y) = (A_k y, y) - \frac{(\Delta x^k (\Delta x^k)^T y, y)}{(\Delta\omega^k, \Delta x^k)} - \frac{(A_k \Delta\omega^k (\Delta\omega^k)^T A_k^T y, y)}{(A_k \Delta\omega^k, \Delta\omega^k)}. \tag{4.28}$$

Из курса линейной алгебры известно, что любая положительно определенная матрица $A_к$ порядка n может быть представлена в виде $A_k = C_k C_k^T$, где $C_к$ — невырожденная нижняя треугольная матрица порядка n. Обозначив $u = C_k^T y$ и $v = C_k^T \Delta\omega^k$, перепишем (4.28) в виде

$$(A_{k+1}y, y) = (u,u) - \frac{(\Delta x^k, y)^2}{(\Delta\omega^k, \Delta x^k)} - \frac{(u,v)^2}{(u,v)} = \frac{|u|^2|v|^2 - (u,v)^2}{|v|^2} - \frac{(\Delta x^k, y)^2}{(\Delta\omega^k, \Delta x^k)}. \qquad (4.29)$$

Покажем, что знаменатель второго слагаемого в правой части (4.29) отрицателен. В самом деле, учитывая, что $\Delta x^k = \chi_k p^k$, а направление спуска $p^к$ на *k-й* итерации определяется согласно (4.23), получаем

$$(\Delta\omega^k, \Delta x^k) = (gradf(x^{k-1}) - gradf(x^k), x^k - x^{k-1}) =$$
$$= -\chi_k (gradf(x^{k-1}), A_k gradf(x^{k-1})) - \chi_k (gradf(x^k), p^k) < -\chi_k (gradf(x^k), p^k),$$

поскольку по предположению индукции матрица $A_к$ положительно определенная. Так как точку x^k находим при помощи исчерпывающего спуска из точки x^{k-1} в направлении вектора $p^к$, то в соответствии со свойством исчерпывающего спуска правая часть последнего соотношения равна нулю, т.е. $(\Delta\omega^k, \Delta x^k) < 0$.

В силу неравенства Коши — Буняковского имеем $|u|^2|v|^2 \geq (u,v)^2$, причем равенство возможно лишь при $u = v$, что с учетом невырожденности матрицы C_k эквивалентно равенству $y = \Delta x^k$. Но тогда $(\Delta x^k, y) = (\Delta x^k, \Delta\omega^k) < 0$ и в итоге при любом ненулевом векторе $y \in R^n$ из (4.29) следует, что

$$(A_{k+1}y, y) = \frac{|u|^2|v|^2 - (u,v)^2}{|v|^2} - \frac{(\Delta x^k, y)^2}{(\Delta\omega^k, \Delta x^k)} > 0,$$

т.е. матрица A_{k+1} положительно определенная.

К соотношению (4.23) можно прийти и из других соображений. Пусть A — положительно определенная матрица порядка n. Введем в R^n наряду со стандартным скалярным произведением (x, y) скалярное произведение

$$(x, y)_A = (Ax, y). \qquad (4.30)$$

Из представления

$$f(x+h) - f(x) = (AA^{-1}gradf(x), h) + o(|h|) = (A^{-1}gradf(x), h)_A + o(|h|)$$

заключаем, что в евклидовом пространстве R^n со скалярным произведением (4.30) градиент целевой функции принимает вид $grad_A f(x) = A^{-1}gradf(x)$. Отсюда следует, что алгоритм квазиньютоновского метода, в котором используется (4.23), можно рассматривать как алгоритм наискорейшего спуска в меняющейся на каждой k-й итерации метрике $p(x,y) = (A_k^{-1}x, y)$ при условии, что A_k — положительно определенная матрица. Поэтому квазиньютоновские методы иногда называют методами переменной метрики.

Можно показать, что если

$$\chi_k = \frac{(\omega^k, p^k)}{(Qp^k, p^k)}, k = \overline{1,n},$$

то при минимизации квадратичной функции с положительно определенной матрицей Q алгоритм ДФП — метода позволяет получить точку минимума не более чем за n итераций. Векторы p^k, $k = \overline{1,n}$, определяющие направления спуска, являются сопряженными относительно матрицы Q и связаны соотношениями

$$A_k Q p^i = p^i \; i = \overline{1,k}, k = \overline{1,n}.$$

При $k=n$ эти соотношения означают, что симметрическая матрица $A_k Q$ имеет n собственных значений, равных единице, а такая матрица является единичной, т.е. $A_n Q = I_n$. Отсюда $A_n = Q^{-1}$, и , следовательно, матрица A_n является обратной к матрице Гессе $H(x^*) = Q$ квадратичной функции, вычисленной в точке минимума x^*.

Отметим, что если в алгоритме ДФП — метода $A_1 = I_n$, то для квадратичной функции траектория поиска совпадает с траекторией, полученной по *методу сопряженных направлений.*

При минимизации целевой функции, не являющейся квадратичной, использование ДФП — метода в общем случае не позволяет найти точку минимума за конечное число итераций. Как и в методе сопряженных направлений, чтобы ослабить влияние накапливаемых погрешностей на сходимость ДФП — метода и уменьшить вероятность появления после очередных n итераций линейно зависимых направлений спуска, применяют процедуру «обновления» алгоритма, в соответствии с которой через каждые n итераций в качестве матрицы A_k используют единичную матрицу, т.е. полагают $A_{mn+1} = A_1 = I_n, m \in N$

Опишем алгоритм ДФП — метода в случае, когда (не квадратичная) целевая функция $f(x)$ дифференцируема в R^n. На предварительном этапе задаем значение ε_3 *параметра точности поиска* в условии прекращения итераций $\left|\omega^k\right| < \varepsilon_3$ и выбираем начальную точку $x°$. Принимаем $A_1 = I_n$, формируем множество $I_0 = \{n, 2n, ...\}$ *моментов обновления алгоритма,* полагаем $k = 1$ и переходим к основной части алгоритма.

1. На *k-й* итерации в точке x^{k-1} вычисляем антиградиент $\omega^k = -gradf(x^{k-1})$ целевой функции и проверяем выполнение неравенства $\left|\omega^k\right| < \varepsilon_3$. Если оно выполнено, то итерации прекращаем и полагаем $x^* \approx x^{k-1}, f(x^*) \approx f(x^{k-1})$. В противном случае переходим к п. 2.

2. Используя (4.23), вычисляем вектор $p^k = A_k \omega_k$, определяющий направление спуска из точки x^{k-1}, и, минимизируя функцию $\psi_k(\chi) = f(x^{k-1} + \chi p^k)$, находим значение χ_k и точку $x^k = x^{k-1} + \chi_k p^k$. Если $k \in I_0$, то принимаем $A_{k+1} = I_n$, полагаем $k := k + 1$ и возвращаемся к п. 1. В противном случае переходим к п. 3.

3. Полагаем $\Delta x^k = x^k - x^{k-1}, \Delta\omega^k = \omega^{k+1} - \omega^k$, по формуле (4.27) вычисляем матрицу A_{k+1}, полагаем $k := k + 1$ и возвращаемся к п. 1.

Как отмечено выше (см. 4.2), применение процедуры «обновления» алгоритма увеличивает общее число итераций, необходимых для достижения заданной точности поиска. Поэтому на практике используют варианты ДФП — метода, в которых (как и в случае метода сопряженных направлений) не используется «обновление» алгоритма через фиксированное число итераций.

Возможны и другие способы построения последовательности $\{A_k\}$ матриц с применением (4.24). Так, например, в алгоритме квазиньютоновского метода, называемого в литературе методом Бройдена-Флетчера-Шенно, или БФШ — методом, используют соотношение

$$A_{k+1} = A_k - \frac{\Delta x^k (\Delta x^k)^T}{(\Delta \omega^k, \Delta x^k)} - \frac{A_k \Delta \omega^k (\Delta \omega^k)^T A_k^T}{\rho^k} + \rho_k r^k (r^k)^T, k \in N \ , \qquad (4.31)$$

где

$$r^k = \frac{A_k \Delta \omega^k}{\rho^k} - \frac{\Delta x^k}{(\Delta x^k, \Delta \omega^k)}, \qquad \rho^k = (A_k \Delta \omega^k, \Delta \omega^k),$$

В алгоритме метода Пауэлла –

$$A_{k+1} = A_k - \frac{\Delta \tilde{x}^k (\Delta \tilde{x}^k)^T}{(\Delta \omega^k, \Delta \tilde{x}^k)}, k \in N,$$

где $\Delta \tilde{x}^k = \Delta x^k + A_k \Delta \omega^k, k \in N$, а в алгоритме метода Мак — Кормика

$$A_{k+1} = A_k - \frac{(\Delta x^k + A_k \Delta \omega^k)(\Delta x^k)^T}{(\Delta \omega^k, \Delta x^k)}, k \in N,$$

причем, как и в случае ДФП — метода, обычно полагают $A_1 = I_n$.

Можно показать, что любой из рассмотренных способов «обновления» матрицы A_{k+1} сохраняет ее свойство положительной определенности, а последовательность $\{A_k\}$ при $k \to \infty$ сходится к $H^{-1}(x^*)$.

Отметим, что сравнение эффективности алгоритмов минимизации принято проводить на специально сконструированных функциях. Графики этих функций имеют четко выраженную *овражную структуру*. На рисунке 4.2 представлены линии уровня унимодальной функции Розенброка

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \qquad (4.32)$$

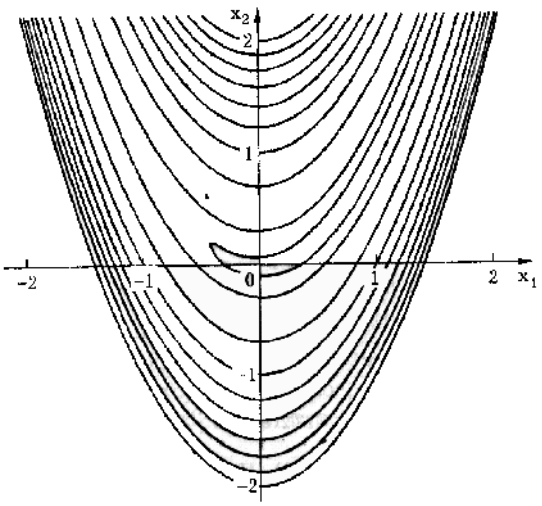


Рисунок 4.2—Линии уровня функции Розенброка

На рисунке 4.3 представлены линии уровня функции Химмельблау

$$f(x_1, x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,$$

имеющей четыре точки минимума.

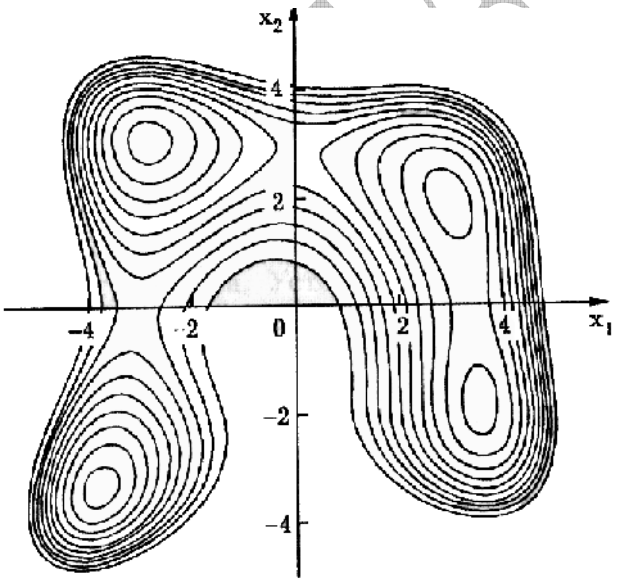


Рисунок 4.3— Линии уровня функции Химмельблау

Для испытания алгоритмов используют также унимодальную функцию Пауэлла

$$f(x) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^2,$$

достигающую минимума в точке $x^* = (0,0,0,0)$.

5 Алгоритмы прямого поиска

В методах прямого поиска минимума целевой функции (или методах нулевого порядка) используют информацию только о значениях этой функции. Многие из этих методов не имеют строгого теоретического обоснования и построены на основе эвристических соображений. Поэтому вопросы сходимости методов прямого поиска еще мало изучены, а оценки скорости сходимости обычно отсутствуют. Вместе с тем эти методы идейно связаны с методами первого и второго порядков, что в ряде случаев позволяет оценивать эффективность алгоритмов прямого поиска применительно к минимизации некоторых классов функций.

Распространенным способом оценки эффективности методов прямого поиска являются вычислительные эксперименты и сравнительный анализ методов по результатам таких экспериментов. Однако следует учитывать, что этот анализ не во всех случаях может приводить к однозначным выводам о преимуществах одного метода перед другими.

Во - первых, это связано с тем, что сравнению обычно подвергаются не только методы, но и программные реализации соответствующих алгоритмов. Хороший метод можно «загубить» плохим программированием, неудачным выбором параметров алгоритма. Во-вторых, методы могут вести себя по-разному на различных этапах процесса минимизации. Удовлетворительного способа преодоления указанных трудностей не существует. Единственное, что можно сделать в подобной ситуации — привести данные о результатах вычислений в развернутой форме, позволяющей сравнить методы по различным критериям. Кроме того, не следует забывать, что поиск решения всегда остается искусством, которому можно научиться лишь путем проб и ошибок, применяя различные методы при решении конкретных задач.

5.1 Особенности прямого поиска минимума

Для применения методов прямого поиска минимума целевой функции достаточно располагать лишь возможностью вычисления значения функции в любой точке ее области определения. Это обстоятельство существенно расширяет сферу практического применения таких методов.

Опишем один из наиболее простых алгоритмов минимизации функции $f(x)$, определенной в R^n. Выберем в R^n точку x^0, называемую обычно базовой. Вычислим в ней значение $f(x^0)$ функции $f(x)$, а затем построим n-мерный куб с центром в этой точке и с ребрами длиной L, параллельными ортам базиса. Множество вершин этого куба вместе с точкой x^0 составляют так называемый шаблон (или образец). Вычислив значения функции в вершинах куба, выберем в качестве новой базовой точки ту из вершин, в которой значение функции меньше $f(x^0)$, и повторим описанную процедуру построения шаблона и выбора следующей базовой точки. Если же такой вершины не оказалось, то оставим прежнюю базовую точку x^0 и построим шаблон с уменьшенной (например, вдвое) длиной ребер куба. Процесс поиска закончим, когда длина ребра куба

станет меньше заданного числа $\varepsilon>0$. Геометрическая иллюстрация описанного алгоритма, называемого обычно поиском при помощи шаблона (или образца), представлена на рисунке 5.1 в двумерном случае.

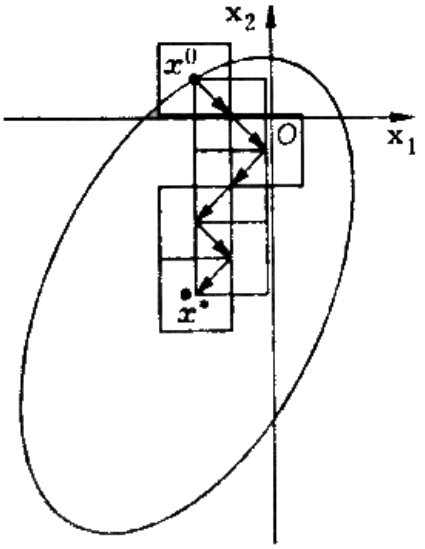


Рисунок 5.1—Геометрическая интерпретация прямого поиска

Основной недостаток поиска при помощи шаблона состоит в резком росте количества вычислений значений целевой функции при увеличении размерности n (порядка 2^n), а главное преимущество – в простоте алгоритма. Эффективность этого алгоритма можно повысить за счет использования информации о ранее вычисленных значениях функции, что позволяет уменьшить число рассматриваемых точек путем отбрасывания тех точек, в которых значения функции не существенны для выбора следующей базовой точки. Геометрически это соответствует процедуре построения вокруг базовой точки в качестве шаблона не n–мерного куба, а конфигурации с меньшим количеством вершин, в которых вычисляются значения функции.

Реализация этой идеи привела к разработке методов симплексного поиска, получивших к началу 1970-х годов широкое распространение. Симплекс (то латинского слова simplex-простой) – это простейший выпуклый многогранник в R^n с $n+1$ вершинами (в R^n – треугольник, в R^3 – тетраэдр). Каждую вершину этого многогранника называют вершиной симплекса. Идея построения алгоритма симплексного поиска состоит в следующем: в вершинах симплекса вычисляют значения минимизируемой функции, наихудшую из вершин ту, в которой значение функции наибольшее, заменяют по определенному правилу новой вершиной, образуя тем самым новый симплекс. Затем эту процедуру повторяют.

В одном из первых разработанных вариантов симплексного поиска был использован регулярный симплекс, у которого все вершины равноудалены друг от друга (в R^2 – равносторонний треугольник, в R^3 – правильный тетраэдр). Дж. Нелдер и Р. Мид в 1965 году впервые рассмотрели произвольный (нерегулярный) симплекс и предложили способ изменения скорости его движения путем расширения или сжатия симплекса в зависимости от того, удачно ли был выбран шаг поиска в направлении убывания функции или нет.

Позднее в исходные варианты методов симплексного поиска были введены различные способы изменения размеров и формы симплекса. Это привело к разработке метода управляемого прямого поиска, под которым понимают выбор по определенному правилу направления смещения центра используемой конфигурации и величины этого смещения, приводящего к убиванию значений функции в вершинах или центре симплекса. Метод, в котором в процессе поиска осуществляют управление изменением формы симплекса, обычно называют методом деформируемых конфигураций. Изменение формы симплекса позволяет улучшить процесс адаптации используемой конфигурации к рельефу графика минимизируемой функции.

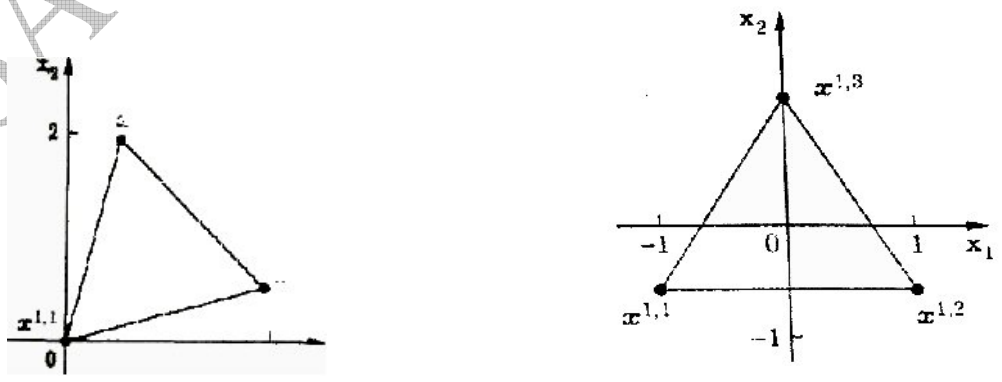
5.2 Использование регулярного симплекса

Сначала рассмотрим простейший алгоритм симплексного поиска минимума ограниченной снизу целевой функции $f(x)$, $x \in R^n$, с использованием регулярного симплекса постоянных размеров. На первом шаге поиска строим регулярный симплекс с вершинами $x^{1,i}$, $i=1,....,n+1$, который обозначаем S_1. Построение симплекса можно провести различными способами.

1. Если $x^{1,1} \in S_1$ – заданная базовая точка, то координаты $x_j^{1,i}$ $(i=2,....,n+1, j=1,....,n)$ остальных n вершин $x^{1,i} \in S_1$ регулярного симплекса S_1, имеющего ребра длиной L, можно вычислить по формулам

$$x x_j^{1i} = \begin{cases} x_j^{11} + \dfrac{\sqrt{n+1}-1}{n\sqrt{2}} L, i = j; \\ x_j^{11} + \dfrac{\sqrt{n+1}+n-1}{n\sqrt{2}} L, i \neq j. \end{cases} \qquad (5.1)$$

где $x^{1,i}$, $j=1,....,n$, – координаты вершины $x^{1,1} \in S_1$. Например, если в R^2 выбрана базовая точка $x^{1,1} = (0,0)$, то остальные две вершины $x^{1,2}$ и $x^{1,3}$ имеют координаты $x_1^{1,2} = 1{,}932$, $x_2^{1,2} = 0{,}518$ и $x_1^{1,3} = 0{,}518$, $x_2^{1,3} = 1{,}932$. На рисунке 5.2, а приведен построенный симплекс.



а ***б***

Рисунок 5.2—Примеры построения симплекса

2. Если $x^0 \in R^n$ – заданная базовая точка, определяемая как центр регулярного симплекса S_1 , имеющего ребра длиной L, то координаты $x_j^{1,i}$ $(i=1,....,n+1, j=1,....,n)$ всех вершин $x^{1,i} \in S_1$ находят по формулам

$$x_j^{1,i} = \begin{cases} x_j^0, j < i-1; \\ x_j^0 + \sqrt{\dfrac{j}{2(j+1)}} L, j = i-1; \\ x_j^0 - \dfrac{1}{\sqrt{2j(j+1)}} L, j > i-1, \end{cases} \quad (5.2)$$

где x_j^0, $j=1,....,n$ – координаты точки x^0. Например, если в R^2 задана точка $x^0=(0.0)$ и $L=2$, то вершинами правильного симплекса будут точки $x^{1,1}=(-1.000, -0.578)$, $x^{1,2}=(1.000, -0.578)$, $x^{1,3}=(0.000, 1.156)$. На рисунке 5.2, *б* приведен построенный симплекс.

После построения симплекса S_1 на первом шаге поиска в вершинах $x^{1,I} \in S_1$, $i=1,....,n+1$, вычисляют значения минимизируемой функции. По результатам вычислений выбирают вершину, в которой значение функции является наибольшим (пусть для определения это будет вершина $x^{1,n+1}$; если таких вершин несколько, то может быть взята любая из них), и по формуле

$$x^{2,n+1} = 2x_c^1 - x^{1,n+1} = \frac{2}{n}\sum_{i=1}^{n} x^{1,i} - x^{1,n+1} \quad (5.3)$$

находят точку $x^{2,n+1}$, симметричную вершине $x^{1,n+1}$ относительно гиперплоскости, в которой лежат остальные вершины симплекса S_1 . Точка x_c^1 равноудалена от вершин $x^{1,I}$, $i=1,....,n+1$, т.е. является центром регулярного симплекса с n вершинами, или, иначе говоря, центром масс системы материальных точек, расположенных в вершинах симплекса и имеющих одинаковую массу.

Нахождение точки $x^{2,n+1}$ называют отражение вершины. В результате получают новый регулярный симплекс S_2 , образованный новой вершиной $x^{2,n+1}$ и n вершинами $x^{1,I}$, $i=1,....,n+1$, принадлежавшими симплексу S_1. На рисунке 5.3 представлена процедура построения нового регулярного симплекса в случае R^2.

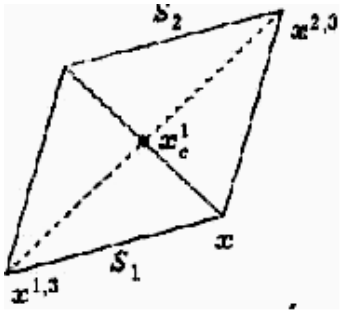


Рисунок 5.3—Процедура построения нового симплекса

На втором шаге поиска в новой вершине $x^{2,n+1} \in S_2$ симплекса S_2 вычисляют значение $f(x^{2,n+1})$, и если $f(x^{2,n+1}) < f(x^{1,n+1})$, то описанную выше процедуру повторяют. При построении алгоритма удобно на каждом k –м шаге поиска отражаемой вершине симплекса S_k присваивать номер $n+1$, т.е. обозначать ее $x^{k,n+1} \in S_k$. нумерацию вершин симплекса S_k назовем правильной, если выполнены неравенства

$$f(x^{k,1}) \le \ldots \le f(x^{k,i}) \le \ldots \le f(x^{k,n+1}).$$

Если на каждом шаге после отражения вершины $x^{k,n+1} \in S_k$ в точку $x^{k+1,n+1}$ и вычисление значения $f(x^{k+1,n+1})$ окажется, что $f(x^{k+1,n+1}) \ge f(x^{k,n+1})$, то следует вернуться к симплексу S_k, считая отражение вершины $x^{k,n+1}$ неудачным. После этого, предполагая нумерацию вершин симплекса S_k правильной, проводят отражение вершины $x^{k,n} \in S_k$, получая точку $x^{k,n+1}$ и сравнивают значения $f(x^{k+1,n})$ и $f(x^{k,n+1})$. Если $f(x^{k+1,n}) \ge f(x^{k,n+1})$, то отражение и этой вершины считают неудачным и проводят отражение вершины $x^{k,n-1} \in S_k$ и т.д.

При фиксированной длине ребра симплекса поиск прекращают на шаге с номером K, если отражения всех вершин симплекса S_k оказались неудачными. Тогда в качестве оценки искомого наименьшего значения f минимизируемой функции $f(x)$ можно принять наименьшее из вычисленных значений $f(x^{K,i})$, $x^{K,I} \in S_k$, $i=1,\ldots,n+1$. Полученной в процессе поиска конечной последовательности $\{S_K\}K$ построенных симплексов S_K, $k=1,\ldots,K$, соответствует невозрастающая конечная последовательность $\{f(x^{K,1})\}$ K минимизируемой функции.

Исследование свойств последовательности $\{f(x^{K,1})\}K$ весьма затруднено. Поэтому на практике вместо нее изучают свойства последовательности $\{f_k\}$ значения $f_k = f(x^K)$ минимизируемой функции,каждое из которых вычислено в центре x^k симплекса S_k, $k=1,\ldots,K$. Процедуру поиска точки $x \in R^n$, в которой функция $f(x)$

$$x^{k+1} = x^k + \alpha_k p^k, k = 1,\ldots, K,$$

где p^k – единичный n-мерный вектор, определяющий направление смещения центра симплекса на k-м шаге; $\alpha_k > 0$ – смещение центра симплекса в направлении вектора p^k при переходе от x^k к x^{k+1}. геометрическая иллюстрация процедуры такого поиска в случае R^2 представлена на рисунке 5.4.

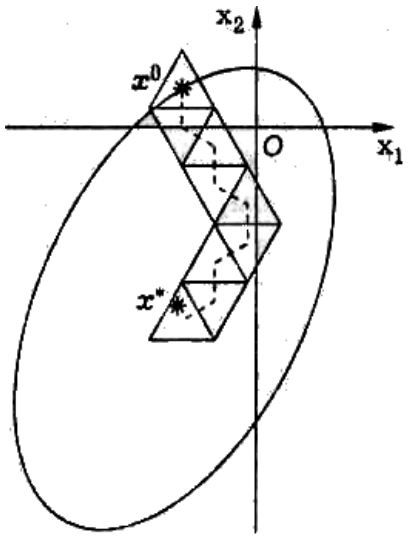


Рисунок 5.4—Геометрическая интерпретация поиска экстремума симплексного метода

Рассмотрим некоторые подходы к построению алгоритмов, позволяющих повысить эффективность процесса поиска при помощи регулярного симплекса. В общем случае управление процессом поиска можно осуществлять, выбирая как направление смещения центра симплекса, так и значение α_k. При этом выбор вектора p^k из множества возможных направлений должен быть связан с определенным правилом, допускающим отражение сразу нескольких вершин симплекса, а изменение α_k при выбранном векторе p^k предполагает изменение длины ребра регулярного симплекса при сохранении его формы.

На рисунке 5.5 приведен пример построения регулярного симплекса в пространстве R^2 по правилу, допускающему отражение всех тех вершин правильного треугольника, в которых значения минимизируемой функции больше, чем в центре треугольника. Если значения этой функции во всех вершинах симплекса превышают ее значения в центре, то процесс поиска прекращают.

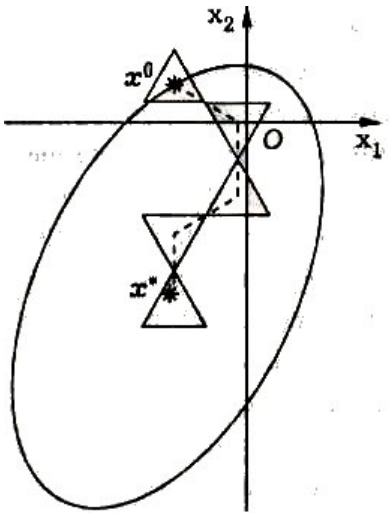


Рисунок 5.5—Модификация симплексного метода

Иной путь построения эффективных алгоритмов связан с идеей изменения размера регулярного симплекса в процессе поиска. Правила такого изменения являются одним из существенных элементов построения алгоритмов управляемого прямого поиска. Это особенно важно в тех случаях, когда наименьшее значение функции *f(x)* необходимо найти с высокой точностью. Действительно, чем меньше размер симплекса, тем точнее можно локализировать точку x, в которой эта функция достигает наименьшего значения. При этом, однако, малый размер симплекса приводит к его замедленному смещению в направлении точки *x*, т.е. к росту числа шагов поиска, связанному с увеличением вычислительных затрат. Большой размер симплекса, наоборот, позволяет за каждый шаг поиска осуществлять большое смещение центра симплекса, но обеспечивает лишь грубую локализацию точки *x*. Поэтому возможность изменения размера симплекса наделяет алгоритм поиска высокими адаптивными свойствами.

Выделим два основных подхода к построению алгоритма, предусматривающего изменение размера симплекса в процессе поиска. В первом из них это изменение происходит лишь при выполнении определенного условия, проверяемого на каждом шаге поиска, а во втором – на каждом шаге поиска заранее заданному закону. Простейшая схема алгоритма, построенного на основе первого подхода, такова.

Пусть $S_k \in R^n$ — регулярный симплекс на k – м шаге поиска, имеющий правильную нумерацию вершин. Процедура отыскания вершин $x^{k+1,n+1} \in S_{k+1}$ нового симплекса S_k , в которой функция *f(x)* имеет меньшее значение, чем в вершине $x^{k,n+1}$, включая два этапа – отражение вершины $x^{k,n+1} \in S_k$ и уменьшение размера симплекса S_k при сохранении его формы, называемое редукцией симплекса.

1.Отражение вершины $x^{k,n+1} \in S_k$ осуществляют в соответствии с формулой, аналогичной (5.3):

$$x^{k+1,n+1} = 2x_c^k - x^{k,n+1} = \frac{2}{n}\sum_{i=1}^{n} x^{k,i} - x^{k,n+1}, \qquad (5.4)$$

где x_c^k –точка, равноудаленная от вершин $x^{K,i}$, $i=1,....,n$. Затем вычисляют значение $f(x^{k+1,n+1})$ функция *f(x)* в найденной точке $x^{k+1,n+1}$.

2.Редукцию симплекса S_k проводят только при выполнении неравенства $f(x^{k+1,n+1}) \geq f(x^{k,n+1})$. При этом длину всех ребер симплекса уменьшают в $1/\delta$ раз, где $\delta \in (0,1)$ –заданный коэффициент редукции, и находят вершины нового симплекса S_{k+1} по формуле

$$x^{k+1,i} = x^{k,1} + \delta(x^{k,i} - x^{k,1}), i = 2,...n+1,$$

Сжимая симплекс в $1/\delta$ раз к вершине $x^{K,1}$, в которой значение целевой функции меньше, чем в других вершинах симплекса. Затем осуществляют переход к этапу 1 при $k=k+1$.

Если же $f(x^{k+1,n+1})<f(x^{k,n+1})$, то редукцию симплекса S_k не проводят и далее рассматривают симплекс S_{k+1} с новой вершиной $x^{k+1,n+1}$ и n вершинами $x^{K,i}$, $i=1,....,n$, симплекса S_k, переходя к этапу 1 при $k=k+1$.

Эту схему алгоритма следует дополнить условием прекращения поиска, зависящим от конкретной постановки задачи минимизации. Например, поиск можно вести до тех пор, пока длина ребер регулярного симплекса не станет меньше заранее выбранного значения или же не будет выполнено заданное число шагов. Одним из возможных условий прекращения поиска является выполнение неравенства

$$(\frac{1}{n+1})\sum_{i=1}^{n+1}(f(x^{k,i})-f(x^k))^2)^{1/2}<\varepsilon, \qquad (5.5)$$

где $\varepsilon>0$ –заданное достаточно малое число.

5.3 Поиск при помощи нерегулярного симплекса

Стандартные симплексные процедуры безусловной минимизации, в которых используются регулярными симплексы, эффективны лишь в тех случаях, когда топография поверхностей уровня ограниченной снизу целевой функции достаточна проста. В противном случае эффективность применения симплексного поиска минимума функции значительно снижается. В частности, это характерно для функции, скорость убывания которой по одному или нескольким направлениям значительно больше, чем по остальным. О графике такой функции говорят, что он имеет овражную структуру.

Примером функции, график которой обладает овражной структурой, является функция

$$f(x_1,x_2)=10(x_1^2-x_2)^2+(x_1-1)^2$$

С минимум в точке $x\ (1,1)$. Топография линий уровня $f(x_1,x_2)$=С этой функции приведена на рисунке 5.8. Препятствием для успешного поиска минимума $f(x)$ является в этом случае форма симплекса: регулярный симплекс нельзя изменить так, чтобы он «вытянулся» вдоль «оврага», а это не позволяет успешно продолжить поиск минимума. Естественной в этих условиях выглядит идея деформирования симплекса в процессе поиска, т.е. изменения его формы и размера.

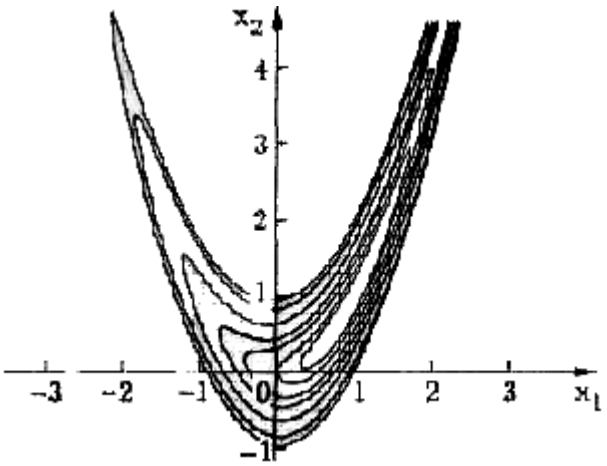


Рисунок 5.8—Топография овражной функции

Существует много вычислительных конструкций прямого поиска с деформируемыми симплексами, различающихся по способам изменения размеров и формы симплекса. Общим для всех них является то, что в качестве основной геометрической конфигурации на каждом шаге поиска выступает симплекс. Информацию о значениях *f(x)* в вершинах симплекса используют для принятия решения о смещении в область меньших значений функции.

Рассмотрим простейшую схему алгоритма прямого поиска по деформируемому симплексу — конструкцию вычислительного алгоритма Нелдера — Мида. Она предусматривает возможность отражения только одной вершины симплекса на каждом шаге поиска.

Пусть $S_k \in R^n$ — заданный произвольный симплекс на *k-м* шаге поиска, имеющий правильную нумерацию вершин. Процедура отыскания вершины вновь генерируемого симплекса S_{k+1}, в которой функция имеет меньшее значение, чем в вершине $x^{k,\ n+1} \in S_k$, включает несколько этапов: отражение вершины $x^{k,n+1}$. Изменение размера и формы симплекса, полученного в результате отражения этой вершины, путем растяжения или сжатия в направлении новой вершины в зависимости от того, удачным был шаг поиска в направлении убывания функции *f(x)* или нет, и редукция симплекса S_k в случае неудачного шага поиска.

1. Отражение вершины $x^{k,\ n+1} \in S_k$, осуществляют в соответствии с формулой, аналогичной (5.3):

$$x^{k+1,n+1} = x_c^k + \alpha(x_c^k - x^{k,n+1}) = \frac{1+\alpha}{n}\sum_{i=1}^{n} x^{k,i} - \alpha x^{k,n+1},$$

где $a > 0$ — заданный коэффициент отражения; x_c^k – точка, равноудаленная от вершин $x^{k,i}$, $i=1,....,n$, и являющаяся центром симплекса с этими вершинами. В результате получают новый симплекс S_{k+1}, образованный новой вершиной $x^{k+1,n+1}$ и n вершинами $x^{k,i} \in S_k$, $i=1,...,n$. На рисунке 5.9, *а* представлена процедура построения нового симплекса в пространстве R^2 при $a = 1$.

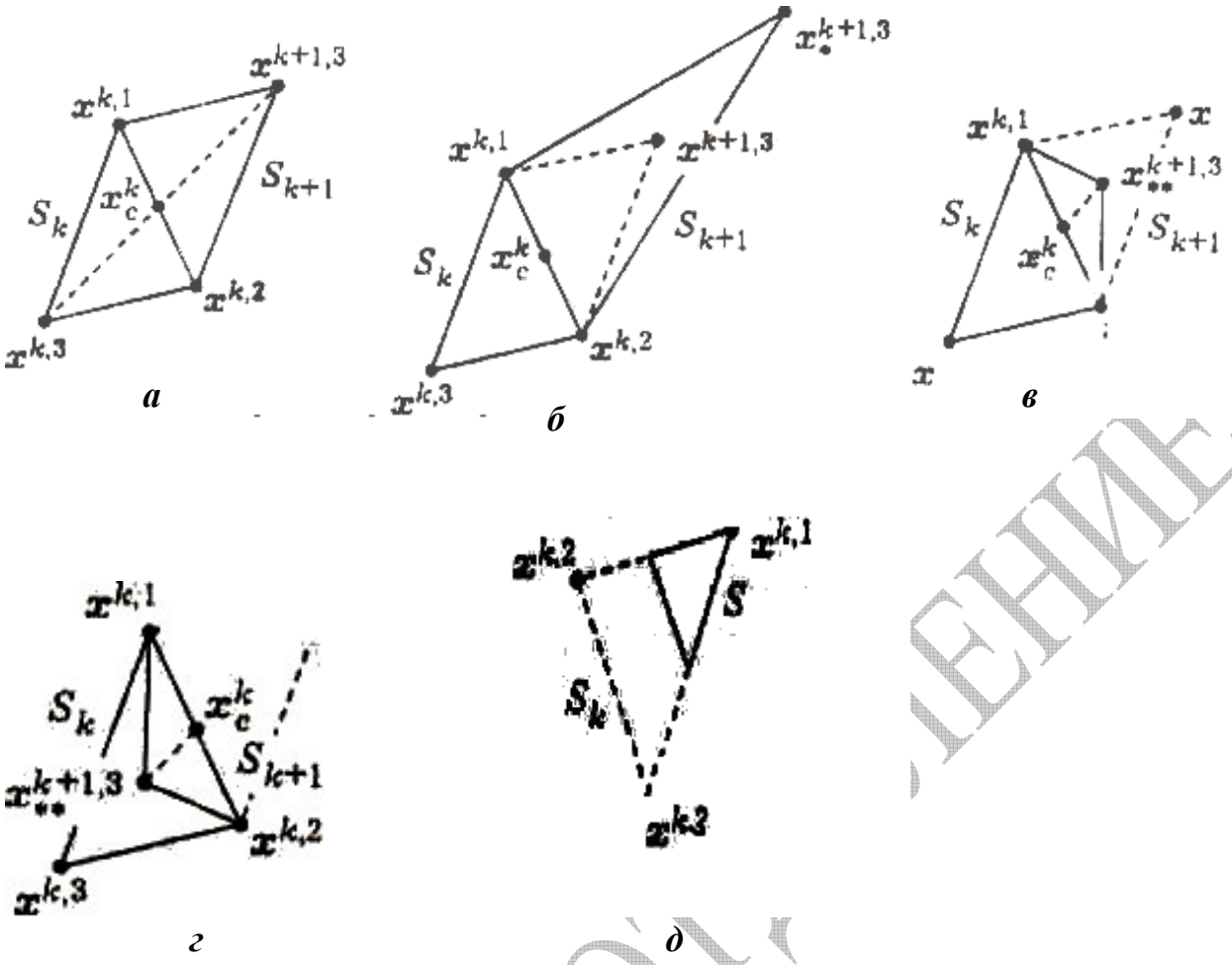


Рисунок 5.9—Основные процедуры метода Нелдера-Мида

В новой вершине вычисляют значение $f(x^{\kappa+1,n+1})$ функции $f(x)$. Если $f(x^{\kappa+1,n+1}) < f(x^{\kappa,1})$, то шаг поиска считают удачным: получен новый симплекс S_{k+1} с наименьшим значением $f(x^{\kappa+1,n+1})$ функции $f(x)$ в новой вершине $x^{\kappa+1,n+1}$. При этом переходят к этапу 2 растяжения симплекса S_{k+1}. При выполнении неравенства $f(x^{\kappa,n}) \geq f(x^{\kappa+1,n+1}) \geq f(x^{\kappa,1})$ далее используют полученный симплекс S_{k+1} с новой вершиной $x^{\kappa+1,n+1}$ и n вершинами $x^{k,i} \in S_k$, $i=1,\ldots,n$, и переходят к этапу 1 при $k:=k+1$. Если же $f(x^{\kappa+1,n+1}) > f(x^{\kappa,n}) > f(x^{k,1})$, то переходят к этапу 3 сжатия симплекса S_{k+1}.

2. При выполнении неравенства $f(x^{\kappa+1,n+1}) < f(x^{k,1})$ растяжение симплекса S_{k+1} осуществляют в соответствии с формулой

$$x_*^{k+1,n+1} = x_c^k + \beta(x^{k+1,n+1} - x_c^k) = \frac{1-\beta}{n}\sum_{i=1}^{n} x^{k,i} + \beta x^{k+1,n+1},$$

где $\beta > 1$ — заданный *коэффициент растяжения*. Геометрически это можно интерпретировать как увеличение длины вектора $x^{\kappa+1,n+1} - x^{\kappa}_c$ в β раз. После растяжения вычисляют значение $f(x_*^{k+1,n+1})$ функции $f(x)$ в найденной точке $x_*^{k+1,n+1}$.

Если $f(x_*^{k+1,n+1}) < f(x^{k,1})$, то далее рассматривают симплекс S_{k+1} с вершиной $x_*^{k+1,n+1}$ и n вершинами $x^{k,i} \in S_k$, $i=1,\ldots,n$, и переходят к этапу 1 при $k:=k+1$. На рисунке 5.9, *б* построен симплекс S_{k+1} в R^2 с применением операции растяжения при $\beta = 2$. При $f(x_*^{k+1,n+1}) \geq f(x^{k,1})$ растянутый симплекс не используют, а далее рассматривают симплекс S_{k+1} с вершиной $x^{к+1,n+1}$ *и вершинами* $x^{k,i} \in S_k$, $i=1,\ldots,n$, и переходят к этапу 1 при $k:=k+1$.

Сжатие симплекса S_{k+1} проводят различными способами в зависимости от результатов сравнения значений функции $f(x)$ в вершинах $x^{k,n+1} \in S_k$ и $x^{к+1,n+1} \in S_{k+1}$. Если $f(x^{к+1,n+1}) \leq f(x^{k,n+1})$, то сжатие осуществляют в соответствии с формулой

$$x_{**}^{k+1,n+1} = x_c^k + \gamma(x^{k+1,n+1} - x_c^k) = \frac{1-\gamma}{n}\sum_{i=1}^{n} x^{k,i} + \gamma x^{k+1,n+1},$$

где $\gamma \in (0,1)$ — заданный *коэффициент сжатия*. Рисунок 5.9, *в* иллюстрирует нахождение вершины $x_{**}^{k+1,n+1}$ в R^2 при $\gamma = 1/2$. Если же $f(x^{к+1,n+1}) > f(x^{k,n+1})$, то новую вершину определяют по формуле

$$x_{**}^{k+1,n+1} = x_c^k + \gamma(x^{k,n+1} - x_c^k) = \frac{1-\gamma}{n}\sum_{i=1}^{n} x^{k,i} + \gamma x^{k,n+1}.$$

На рисунке 5.9, *г* представлена процедура нахождения вершины $x_{**}^{k+1,n+1}$ в этом случае при $\gamma=1/2$.

После сжатия симплекса вычисляют значение $f(x_{**}^{k+1,n+1})$ функции $f(x)$ в найденной вершине $x_{**}^{k+1,n+1}$ и сравнивают его со значением $f(x^{k,n+1})$. Если $f(x_{**}^{k+1,n+1}) < f(x^{k,n+1})$, то далее используют симплекс S_{k+1} с найденной вершиной $x_{**}^{k+1,n+1}$ и n вершинами $x^{k,i} \in S_k$, $i=1,\ldots,n$, и переходят к этапу 1 при $k:=k+1$. В противном случае этап сжатия не приводит к уменьшению значения функции в найденной вершине. Поэтому далее рассматривают симплекс S_k и переходят к этапу 4 — редукции этого симплекса.

Редукция симплекса S_k состоит в уменьшении длин всех его ребер в $1/\delta$ раз, где $\delta \in (0, 1)$ — заданный *коэффициент редукции*. Вершины нового симплекса с находят по формуле

$$x^{k+1,i} = x^{k,1} + \delta(x^{k,i} - x^{k,1}), i = 2,\ldots, n+1.$$

Симплексы S_k и S_{k+1} до и после редукции при $\delta = 1/2$ изображены на рисунке 5.9, *д*. После вычисления значений $f(x^{к+1,i})$ функции $f(x)$ в найденных вершинах $x^{к+1,i}$, $i=2,\ldots,n+1$, переходят к этапу 1 при $k:=k+1$.

Помимо описанных этапов периодически (после N шагов поиска) следует проводить так называемую *операцию восстановления симплекса*. Она состоит в следующем: сохраняют две «лучшие» точки текущего симплекса S_k, расстояние между которыми принимают за длину ребра вновь генерируемого симплекса. Это позволяет ликвидировать излишние деформации симплекса, накопившиеся за предшествующие N шагов поиска.

Рассмотренную схему алгоритма Нелдера-Мида следует дополнить условием прекращения поиска. Выбор этого условия диктуют те же правила, что и в методах поиска при помощи регулярного симплекса (см. 5.2).

Построение начального симплекса в методах поиска при помощи нерегулярного симплекса проводят различными способами. Один из способов состоит в выборе в качестве начального регулярного симплекса S_1, координаты вершин которого вычисляют при помощи (5.1) или (5.2). Другой способ заключается в выборе некоторой *базовой точки* $x^0 \in R^n$, которую принимают за вершину $x^{1,1}$ начального симплекса S_k, а остальные n вершин находят по формуле

$$x^{1,i+1} = x^0 + le_i, i = 1,..., n, \tag{5.7}$$

где e_i — векторы стандартного базиса в R^n; l — заданное число. Ясно, что в этом случае начальный симплекс S_1 будет нерегулярным (например, в R^2 получим равнобедренный треугольник).

С теоретической точки зрения методы поиска при помощи нерегулярного симплекса изучены недостаточно полно, однако практические вычисления указывают на их работоспособность. В задачах безусловной минимизации рекомендуют *выбирать* $a = 1, \beta=2, \gamma = 1/2$ или $a = 2,\beta = 5/2, \gamma = 1/4$. Вопрос выбора оптимальных параметров для каждой конкретной целевой функции может быть решен после дополнительного исследования свойств метода с помощью вычислительного эксперимента.

5.4 Циклический покоординатный спуск

Поиск точки x^*, в которой функция $f(x)$ достигает наименьшего значения, можно описать рекуррентным соотношением

$$x^k = x^{k-1} + \alpha_k u^k, k \in N, \tag{5.8}$$

где x^0 — *начальная точка;* $u^к$ — единичный вектор, определяющий направление спуска на k *-м* шаге; $a_к > 0$ — длина *шага спуска,* т.е. расстояние в направлении вектора $u^к$, отделяющее точку x^{k-1} от новой точки x^k. Различные методы отличаются друг от друга способом выбора направления спуска и значения $a_к$.

Поиск точки x^* обычно прекращают при выполнении одного или обоих неравенств:

$$\left|x^k - x^{k-1}\right| = \alpha_k < \varepsilon_1, \qquad \left|f(x^k) - f(x^{k-1})\right| < \varepsilon_2, \tag{5.9}$$

где ε_1 и ε_2 –заданные параметры точности поиска.

В одном из алгоритмов *прямого поиска,* базирующихся на идее методов спуска, на каждом k *-м* шаге поиска проводят одномерную минимизацию целевой функции последовательно по каждой из n координат ее аргумента $x \in R^n$. Такой метод называют обычно ***методом циклического покоординатного*** *спуска.* Рассмотрим алгоритм этого метода.

Пусть e_i, $i = 1,\ldots,n$, — стандартный базис в R^n. Выберем начальную точку x^0. Поиск точки минимума функции в методе циклического покоординатного спуска проводят в соответствии с рекуррентным соотношением

$$x_j^k = x_j^{k-1} + \alpha_j^k, k \in N, j = 1,\ldots,n. \quad (5.10)$$

Значение $a_j^к \in R$ находят из решения задачи одномерной минимизации (см. 2)

$$\varphi_j^k(\alpha) \to \min, \varphi_j^k(\alpha) = f(x^{k-1} + \alpha e_j). \quad (5.11)$$

Подчеркнем, что индекс *j* изменяется циклически, пробегая на каждом *шаге покоординатного спуска* все значения *от 1 до n*, причем значение $a_j^к$ в отличие от значения $a_к$ в (5.8) может быть как положительным, так и отрицательным.

Пусть выбраны значения ε_1 и (или) ε_2 в (5.9) и перед *k-м* шагом циклического покоординатного спуска по найденной на предыдущем шаге точке $x^{к-1}$ вычислено значение $f(x^{к-1})$ целевой функции (на первом шаге значение $f(x^0)$ вычисляют в выбранной начальной точке x^0). Тогда для каждого значения $j = 1,\ldots,n$ из решения задачи (5.11) находят значение $a_j^к$ и затем вычисляют

$$x^k = x^{k-1} + \sum_{j=1}^{n} \alpha_j^k e_j \quad (5.12)$$

и значение $f(x^к)$. Далее проверяют выполнение условий (5.9) или того из них, которое выбрано в качестве условия прекращения поиска. При положительном результате поиск точки x^* минимума функции $f(x)$ прекращают на *k -м* шаге и принимают $x^* \approx x^к$ и $f(x^*) \approx f(x^к)$. В противном случае переходят к следующему шагу метода, полагая $k:=k+1$.

Рассмотренный алгоритм прост в реализации, но эффективен лишь в случаях, когда минимизируемая *функция* является *сепарабельной,* т.е. представляет собой сумму функций, каждая из которых зависит лишь от одной координаты:

$$f(x) = \sum_{j=1}^{n} h_j(x_j), x \in R^n$$

В этом случае решение задачи минимизации можно получить за один шаг поиска.

В самом деле, так как

$$f(x^*) = \min_{x \in R^n} f(x) = \sum_{j=1}^{n} \min_{x_j \in R} h_j(x_j) = \sum_{j=1}^{n} h_j(x_j^*),$$

то достаточно на первом шаге поиска последовательно решить *n* задач одномерной минимизации функций $h_j(x_j)$, $j=1,\ldots,n$, что позволит найти все *n* координат x_j^* искомой точки x^*.

Для функций более общего вида может нарушаться даже условие релаксационности последовательности $\{x^k\}$, т.е. $f(x^{к-1}) — f(x^k) > 0$ для некоторых номеров *k*. Поэтому обычно используют модификацию этого метода, состоящую в следующем. На каждом шаге при спуске по очередной координате x_j учитывают результаты, полученные на этом шаге для координат X_l, $l = 1 \ldots, j-1$. В этом случае

значение a_j^κ находят из решения задачи (5.11), в которой полагают $\varphi_j^k = f(\tilde{x}_{j-1}^k + \alpha e_j)$,, где

$$\tilde{x}_{j-1}^k = x^{k-1} + \sum_{l=1}^{j-1} \alpha_j^k e_l$$

После перебора всех *n* координат вычисляют $x^\kappa = x^\kappa{}_{n\_1} + a^k{}_n e_n$.

5.5 Метод Хука — Дживса

Эффективность *прямого поиска* точки минимума ограниченной снизу *целевой функции* можно повысить, если на каждом *k-м шаге поиска* соответствующим образом выбирать направление спуска. Для этого на каждом *k-м* шаге выделяют предварительный этап *исследующего поиска*. Целью этого этапа является выбор направления спуска путем исследования поведения целевой функции *f(x)* в окрестности точки $x^{\kappa-1}$, найденной на предыдущем шаге. В результате выполнения этапа исследующего поиска находится точка x^κ, для которой $f(x^\kappa) < f(x^{k-1})$. Направление спуска, завершающего *k-й* шаг поиска, определяется вектором $x^\kappa - x^{k-1}$. Такая стратегия поиска, предложенная в 1961 году, получила название метода Хука — Дживса.

Опишем один из алгоритмов исследующего поиска. Пусть выбрана *начальная точка* $x°$ и вектор $b=(b_1 \ldots b_n)^T$, удовлетворяющий условию $|b| \geq \varepsilon$, где $\varepsilon > 0$ — заданный *параметр точности* исследующего *поиска*. Координаты вектора *b,* называемого *вектором перемещений,* определяют приращения координат точки x^0 на этапе исследующего поиска. Полагаем $\kappa = j = 1$, $x^k_j = x^\kappa = x^0$, $f^k_j = f(x^0)$ и переходим к основной части алгоритма исследующего поиска.

1. Вычисляем

$$f^k_{+j} = f(\tilde{x}^k_j + b_j e_j) \text{ и } f^k_{-j} = f(\tilde{x}^k_j - b_j e_j),$$

где $e_1, \ldots, e_n$ –стандартный базис в R^n, находим точку

$$\begin{cases} \tilde{x}^k_j + b_j e_j, & f^k_{+j} < f^k_j \text{ и } f^k_{+j} \leq f^k_{-j}; \\ \tilde{x}^k_j - b_j e_j, & f^k_{-j} < f^k_j \text{ и } f^k_{-j} < f^k_{+j}; \\ \tilde{x}^k_j, & \text{во всех остальных случаях,} \end{cases}$$

полагаем $f^k_{j+1} = f(x^\kappa_{j+1})$ и переходим к п. 2.

2. Если $j < n$, то принимаем $j := j+1$ и переходим к п. 1. В противном случае переходим к п.

3. Если $x^k_{n+1} \neq x^k$, то переходим к п. 4. В противном случае уменьшаем длину вектора *b,* полагая $b := M/\gamma$, где $\gamma > 1$ —*коэффициент дробления шага* и исследующего поиска, и, полагая $j=1$, $\tilde{x}^k_j = \tilde{x}^k, f^k_j = f(\tilde{x}^k)$ возвращаемся к п.1

4. Если $\left|\tilde{x}^k_{n+1} - \tilde{x}^k\right| < \varepsilon$, то дальнейший поиск точки минимума прекращаем, полагая $x^* \approx x^{k-1}$ и $f(x^*) \approx f(x^{\kappa-1})$. В противном случае полагаем $\tilde{x}^{k+1} = \tilde{x}^k_{n+1}$ и переходим на *k-м шаге* поиска к этапу спуска в направлении

вектора $\tilde{x}^{k+1} - \tilde{x}^k$, имея при этом $f(\tilde{x}^{k+1}) < f(\tilde{x}^k) \le f(x^{k-1})$. На этапе спуска по формуле

$$x^k = \tilde{x}^k + \alpha_k(\tilde{x}^{k+1} - \tilde{x}^k), \qquad (5.13)$$

подбирая так называемый ***ускоряющий множитель*** $a_k > 0$, находим такую точку $x^к$, чтобы $f(x^к) \le f(x^{к+1})$. *С* увеличением a_k увеличивается длина $a_k\ |x^{к+1} - x^к|$ *шага спуска* в направлении вектора $x^{к+1} - x^к$. Значение a_k можно подобрать из условия минимума функции *f(x)* при смещении точки x^k в направлении этого вектора. Может оказаться, что $a_k \in (0,1]$. После нахождения точки $x^к$ переходим к следующему шагу поиска (к п.1 этапа исследующего поиска), полагая $j=1$, $\tilde{x}_j^{k+1} = \tilde{x}^{k+1} = x^k$, $f_j^{k+1} = f(x^k)$ и затем $k:=k+1$.

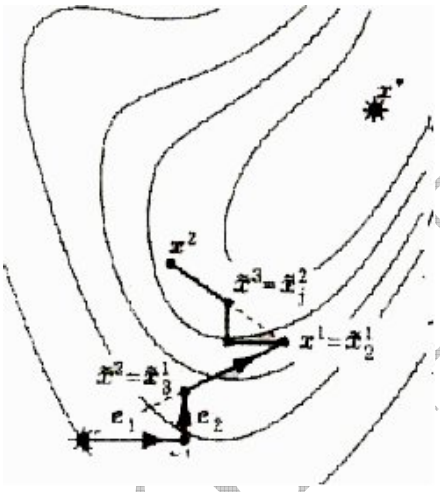


Рисунок 5.16 —Этапы исследующего поиска методом Хука –Дживса

В простейшем варианте метода Хука — Дживса значение a_k в (5.13) не подбирают, а задают постоянным, причем обычно полагают $a_k = 2$. На рисунке 5.16 иллюстрируются этапы исследующего поиска и спуска для первых двух шагов поиска точки x^* минимума целевой функции двух переменных при $a_1 = a_2 = 2$, $\gamma = 2$ и начальной точке x^0.

Известно много модификаций метода Хука — Дживса. Одна из модификаций связана с введением дополнительных правил выбора точки $x^к$ на каждом *k-м* шаге при проведении этапа исследующего поиска. Например, координаты этой точки можно выбирать, используя модифицированный *метод циклического покоординатного спуска.* На рисунке 5.17 иллюстрируется первый шаг поиска точки x^* минимума целевой функции двух переменных с применением на этапе исследующего поиска модифицированного метода циклического покоординатного спуска, а на этапе спуска — процедуры подбора ускоряющего множителя $a_k = a_1 > 0$ в формуле (5.13), исходя из условия минимума целевой функции в направлении вектора $x^2 — x^1$.

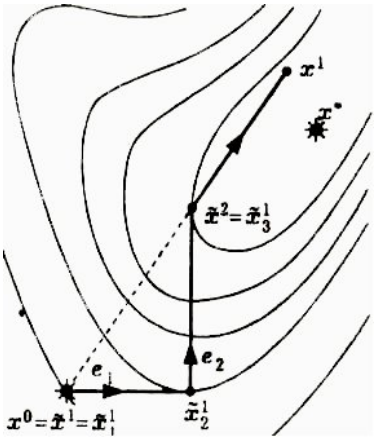


Рисунок 5.17—Этапы метода циклического покоординатного спуска

Для определения точки $x^к$ на *k-м* шаге при проведении этапа исследующего поиска можно также использовать процедуры случайного поиска.

Другой путь повышения эффективности поиска точки минимума функции состоит в выполнении на каждом шаге повторного этапа исследующего поиска. В случае квадратичной целевой функции $f(x)=1/2(Qx,x)+(c,x)$ с положительно определенной матрицей Q порядка n эта модификация метода Хука–Дживса позволяет получить точку минимума за один шаг, если на этапах исследующего поиска использовать модифицированный метод циклического покоординатного спуска, а выбор ускоряющего множителя на этапе спуска осуществлять исходя из условия минимум целевой функции в установленном направлении спуска.

5.6 Методы Розенброка и Пауэлла

Рассмотрим еще одну стратегию поиска точки минимума ограниченной снизу целевой функции $f(x)$, $x \in R^n$. Метод, реализующий эту стратегию поиска, также предусматривает проведение исследующего поиска на каждом *k-м* шаге. Целью исследующего поиска является выбор текущего направления спуска с учетом информации о поведении целевой функции в окрестности точки x^{k-1}, найденной на предыдущем шаге.

Отличие этого метода от метода Хука – Дживса состоит в способе выбора направлений исследующего поиска. Если в методе Хука – Дживса они фиксированы и коллинеарны направлениям векторов стандартного базиса в R^n, то в рассматриваемом методе выбор этих направлений проводят в процессе минимизации целевой функции путем построения на каждом *к-м* шаге поиска нового ортонормированного базиса в R^n. Итогом выполнения этого этапа является нахождение точки x^k , для которой $f(x^к) < f(x^{k-1})$. Тогда вектор $x^к – x^{k-1}$ определит направление спуска на *k-м* шаге.

Такая стратегия поиска впервые была реализована в 1960 году и получила название метода Розенброка. В первоначальном варианте метода Розенброка процедура нахождения точки $x^к$ (как и в методе Хука — Дживса)

была основана на дискретных шагах исследующего поиска по выбранным направлениям. Опишем модификацию этого метода, в которой на каждом *k-м* шаге поиска выбор координат точки $x^к$ проводят методом модифицированного *циклического покоординатного спуска.*

Пусть выбраны начальная точка $x^0 \in R^n$ и параметр $\varepsilon > 0$ точности поиска. Возьмем в качестве векторов p_1^j, $j=1,...,n,$ определяющих направления исследующего поиска на первом шаге, векторы e_j стандартного базиса в R^n . Полагаем $k = j=1$, $\tilde{x}_1^k = x^0$ и переходим к основной части алгоритма.

1. Минимизируя функцию $\varphi_j^{(k)}(x) = f(\tilde{x}_j^k + \chi p_j^k)$, находим *значение* $\chi^{(k)}{}_j \in R$, *вычисляем* $\tilde{x}_{j+1}^k = \tilde{x}_j^k + \chi_l^{(k)} p_j^k$ *и* переходим к *п. 2.*

Если $j < n$, то принимаем $j:=j+1$ в возвращаемся к п.1. В противном случае полагаем $x^k = \tilde{x}_{n+1}^k$ и переходим к п. 3.

Если $|x^к - x^{к-1}| < \varepsilon$, где $\varepsilon > 0$ — достаточно малое число, характеризующее точность выполнения этапа исследующего поиска, то дальнейший поиск точки минимума прекращаем, принимая $x^* \approx x^к$ и $f(x^*) \approx f(x^к)$. В противном случае переходим к п. 4.

На *k-м* шаге поиска строим новый ортонормированный базис, векторы p_j^{k+1}, $j = 1,...,n,$ которого задают направления исследующего поиска на *(k+1)-м* шаге. При построении этого базиса используем процесс ортогонализации Грамма – Шмидта и проводим его следующим образом. Полагаем

$$\alpha_j^{k+1} = \begin{cases} p_j^k, \chi_j^k = 0; \\ \sum_{i=j}^{n} \chi_i^{(k)} p_i^k, \chi_j^k \neq 0. \end{cases} \qquad j=1,...,n$$

Далее вычисляем

$$b_j^{k+1} = \begin{cases} \alpha_j^{k+1} - \sum_{i=1}^{j-1} (\alpha_j^{k+1}, p_i^{k+1}) p_i^{k+1}, j = 2,...,n \\ \alpha_j^{k+1}, j = 1 \end{cases}$$

и

$$p_j^{k+1} = \frac{b_j^{k+1}}{|b_j^{k+1}|}, j = 1,...,n$$

После вычисления векторов p_j^{k+1}, $j=1,...,n,$ переходим к п. 1, полагая $j:=1$, $\tilde{x}_j^{k+1} = x^k$ и затем $k:= k+ 1.$

На рисунке 5.21 иллюстрируются этапы одного шага поиска точки минимума целевой функции двух переменных из начальной точки x^0. Отметим, что рассмотренный алгоритм можно модифицировать, вводя дополнительные правила выбора точки $x^к$ на каждом *k-м* шаге при проведении этапа исследующего поиска (см. 5.4).

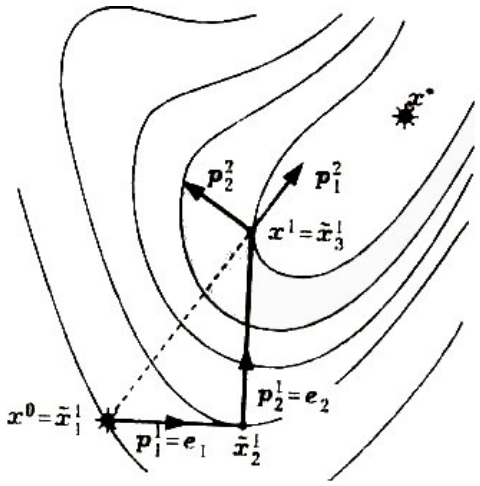


Рисунок 5.21—Этапы модифицированного циклического покоординатного спуска

6 Минимизация выпуклых функций

Широкий класс задач *математического программирования* связан с минимизацией *выпуклых функций* многих переменных, определенных на выпуклом *множестве*. Такие задачи относят к *задачам выпуклого программирования*. В этой главе рассмотрены основные свойства выпуклых множеств и функций и описаны некоторые методы минимизации выпуклой целевой функции в случае, когда на область изменения *параметров оптимизации* не наложено *ограничений*.

6.1 Условия минимума выпуклых функций

Поиск наименьшего значения функции многих переменных – сложная задача. Напомним некоторые факты, относящиеся к этой задаче.

Точка, в которой функция достигает своего наименьшего значения, является точкой локального минимума этой функции. Поэтому для определения наименьшего значения функции, если функция достигает его, достаточно найти все точки локального минимума этой функции и среди них выбрать ту (или те), в которых функция имеет наименьшее значение. Такой подход возможен, если функция имеет конечное (и относительно небольшое) число точек локального минимума.

Точки локального минимума функции, являющиеся внутренними точками области определения функции, можно искать, опираясь на необходимое условие экстремума функции. Если внутренняя точка $x^* \in \Omega$ множества $\Omega \in R^n$ есть точка локального минимума функции $f(x)$, определенной на Ω, и функция $f(x)$ дифференцируема в точке x^*, то эта точка является стационарной точкой функции $f(x)$, т.е. в этой точке выполняется условие

$$gradf\,(x^*) = 0, \qquad (3.1)$$

где grad $f(x^*)$ –градиент функции в точке x*. Таким образом, точки локального минимума внутри области определения — это либо стационарные точки, либо точки недифференцируемости функции (критические точки). Поэтому точку наименьшего значения функции, если она существует, можно искать, сравнивая значения функции во всех стационарных и критических точках функции. Выяснять, какие из этих точек являются точками локального минимума, не нужно.

Проверить, является ли стационарная точка точкой локального минимума, можно с помощью достаточного условия экстремума. Если функция $f(x)$ дважды непрерывно дифференцируема в стационарной точке x^* и матрица Гессе $H(x^*)$ функции $f(x)$ в этой точке положительно определена, то x^* является точкой строгого локального минимума.

Функция может достигать наименьшего значения в граничной точке области определения. Дифференциальные свойства функции многих переменных можно в этом случае использовать, если границей области

определения является гладкая кривая или поверхность. В этом случае можно поставить задачу на условный экстремум, в которой уравнения связи описывают границу области, и определить точку, в которой функция достигает наименьшего значения, как точку условного локального минимума. В остальных случаях необходимо непосредственного исследование функции на границе области определения, а ее дифференциальные свойства не могут помочь в решении задачи.

Необходимые и достаточные условия экстремума функции многих переменных далеко не всегда обеспечивают решение задачи минимизации. Использование необходимого условия локального экстремума приводит к необходимости решать систему нелинейных уравнений, что само по себе является сложной задачей. Функция может иметь большое (или даже бесконечное) число стационарных и критических точек, и тогда выбрать точку наименьшего значения функции среди них сложно. Наконец, трудности возникают, если точка наименьшего значения находится на границе области определения функции.

Достаточные условия экстремума могут помочь выделить среди стационарных точек те, которые являются точками локального минимума. Тем самым сокращается общее число точек, «подозрительных на наименьшее значение». Однако даже если стационарная точка определена, функция дважды непрерывно дифференцируема в ней, матрица Гессе в этой точке вычислена, то это еще не гарантирует определенного ответа, является ли исследуемая точка точкой локального минимума. Дело в том, что матрица Гессе может оказаться неотрицательно определенной. А в этом случае на поведение функции в окрестности стационарной точки оказывают влияние дифференциалы высших порядков (если такие, конечно, существуют).

Все указанные трудности можно преодолевать с помощью методов, в которых не используются дифференциальные свойства функции. И в этом случае важную роль могут играть другие, не связанные с дифференцируемостью, свойства функции. Некоторые выводы о существовании и единственности точек локального минимума можно сделать для *выпуклых функций*.

Теорема 6.1. Если функция *f(x)*, определенная на выпуклом множестве $\Omega \in R^n$, является выпуклой, то эта функция в любой точке локального минимума достигает наименьшего в Ω значения. Множества всех точек локального минимума функции выпукло. *Функция, строго выпуклая* на выпуклом множестве, имеет не более одной точки локального минимума.

Теорема 6.2. Пусть функция *f(x)* выпукла на выпуклом множестве $\Omega \in R^n$ и дифференцируема в точке $x^* \in \Omega$. Для того чтобы точка x^* была точкой локального минимума функции *f(x)* , необходимо и достаточно, чтобы для любой точки $x \in \Omega$ выполнялось неравенство

$$(gradf(x^*), x - x^*) \geq 0. \qquad (6.2)$$

6.2 Минимизация позиномов

В прикладных задачах минимизации *целевая функция* часто имеет вид

$$y(x) = \sum_{i=1}^{m} c_i p_i(x), \tag{6.3}$$

$c_i > 0$, $i=1,...,m$, а функция

$$p_i(x) = \prod_{j=1}^{n} x_j^{a_{ij}}, \tag{6.4}$$

$a_{ij} \in R$, $i=1,...,m$, $j=1,...,n$, определены на множестве

$$R_+^n = \{(x_1,...,x_n) \in R^n : x_i > 0, i = 1,...,n\}$$

точек с положительными координатами. Функцию вида (6.3) называют *позиномом*.

Позиномом при определенных значениях показателей степени a_{ij} может не быть *выпуклой функцией*. Однако с помощью замены переменных

$$x_j = e^{\xi_j}, j = 1,...,n, \tag{6.5}$$

он преобразуется в выпуклую функцию в R^n. Так же эта функция является *строго выпуклой функцией* в том случае, когда матрица $A=(a_{ij})$ имеет ранг, равный n.

Поскольку функция $y(\xi_1,...,\xi_n)$ выпукла в R^n, любая точка ее локального минимума является точкой наименьшего значения функции (см. теорему 6.1). более того, так как эта функция дифференцируема в R^n, множество ее точек наименьшего значения совпадает с множеством стационарных точек. В случае строгой выпуклости функция имеет не более одной точки локального минимума. Поскольку якобиан замены переменных (6.5) при любых значениях $\xi_1,...,\xi_n$ отличен от нуля, любой стационарной точке $(\xi_1^*,...,\xi_n^*)$ функции $y(\xi_1,...,\xi_n)$ соответствует стационарная точка $x^* = (x_1^*,...,x_n^*), x_j^* = e^{\xi_j^*}$, $i=1,...,n$, и наоборот. Таким образом, минимизация позинома сводится к поиску его стационарных точек, причем для определения наименьшего значения позинома достаточно найти хотя бы одну стационарную точку.

Для строго выпуклой функции $exp(t)=e^t$ верно *неравенство Иенсена*

$$\exp(\sum_{i=1}^{m} w_i t_i) \le \sum_{i=1}^{m} w_i \exp(t_i),$$

где $w_i \ge 0$, $I=1,...,m$, и $\sum_{i=1}^{m} w_i = 1$. Преобразуем левую часть этого равенства:

$$\exp(\sum_{i=1}^{m} w_i t_i) = \prod_{i=1}^{m} \exp(w_i t_i) = \prod_{i=1}^{m} (e^{t_i})^{w_i}.$$

Выполнив замену переменных $e^{t_i} = u_i, i = 1,...,m$, получим ***неравенство взвешенных средних***

$$\sum_{i=1}^{m} w_i u_i \geq \prod_{i=1}^{m} u_i^{w_i}, \qquad (6.6)$$

где $u_i, w_i > 0, i = 1,...,m$ и $\sum_{i=1}^{m} w_i = 1$. Неотрицательные коэффициенты w_i в этом неравенстве, в сумме оставляющие единицу, называют *нормированными весами*.

Выполнив в неравенстве (6.6) замену $y_i = w_i u_i, i = 1,...,m,$ получим

$$\sum_{i=1}^{m} y_i \geq \prod_{i=1}^{m} \left(\frac{y_i}{w_i}\right)^{w_i}.$$

Пусть $y_i = c_i p_i(x), i = 1,...,m,$ т.е. сумма в левой части последнего неравенства есть позином вида (6.3). Тогда

$$y(x) \geq v(x,w), \qquad (6.7)$$

где $w = (w_1, w_2, ..., w_m)$, а правая часть $v(x,w)$ неравенства с учетом вида функций $p_i(x)$ равна

$$v(x,w) = \prod_{i=1}^{m}\prod_{j=1}^{n} \left(\frac{c_i}{w_i}\right)^{w_i} x_j^{a_{ij} w_i} = \prod_{i=1}^{m} \left(\frac{c_i}{w_i}\right)^{w_i} \prod_{j=1}^{n} x_j^{b_j}.$$

Здесь

$$b_j = \sum_{i=1}^{m} a_{ij} w_i, i = 1,..., n.$$

Подберем нормированные веса w_i таким образом, чтобы функция $v(x,w)$ не зависела от x. Для этого необходимо и достаточно, чтобы выполнялись равенства $b_j=0$. Другими словами, нормированные веса должны удовлетворять системе линейных уравнений

$$\sum_{i=1}^{m} a_{ij} w_i = 0, j = 1,..., n. \qquad (6.8)$$

Такой выбор нормированных весов не всегда возможен. Для существования нормированных весов, удовлетворяющих системе уравнений (6.8), необходимо, чтобы эта система имела ненулевые решения. Однако даже если система (6.8) и имеет ненулевые решения, то среди них может не оказаться решений с положительными (или хотя бы неотрицательными) значениями неизвестных w_i: так будет, например, в случае, когда все коэффициенты a_{ij} являются положительными.

Если нормированные веса w_i выбраны в соответствии с равенствами (6.8), то

$$v(x,w) = \prod_{i=1}^{m} \left(\frac{c_i}{w_i}\right)^{w_i}$$

и неравенство (6.7) преобразуется к виду

$$y(x) = \sum_{i=1}^{m} c_i \prod_{j=1}^{n} x_j^{a_{ij}} \geq \prod_{i=1}^{m} \left(\frac{c_i}{w_i}\right)^{w_i} = d(w). \qquad (6.9)$$

Функцию *d(x)* называют ***двойственной к позиному*** *y(x),* а равенства (6.8) – ***условиями ортогональности*** (вектор с координатами $w_i, i = 1,...,m$, ортогонален каждому из n столбцов матрицы (a_{ij}) размера $(m \times n)$.

В R^m рассмотрим множество

$$W = \left\{ (w_1,...,w_m) \in R^m : w_i > 0, i = 1,...,m, \sum_{i=1}^{m} w_i = 1 \right\} \right\} \qquad (6.10)$$

Нетрудно показать, что это множество представляет собой *выпуклую оболочку* элементов $w_i e_i \in R^m, i = 1,...,m$ где $e_1,...,e_m$ – стандартный базис в R^m. Другими словами, *W* – *m*-мерный *выпуклый многогранник*. Для двойственной функции *d(w)* верна следующая теорема.

Теорема 6.3. Двойственная функция *d(w)* на множестве *W* достигает наибольшего значения $d^*=c_1+C_2+...+c_m$ в точке $w^*=(c_1/d^*,c_2/d^*,...,c_m/d^*) \in W$, и эта точка единственная.

Пусть *W** – множество точек $w=(w_1,...,w_m)$ в выпуклом многограннике W, которые удовлетворяют условию (6.8). Множество *W** может быть и пустым, т.е *W*=Ø.* но если *W*=Ø*, то позином *y(x)* имеет двойственную функцию *d(w),* причем для любых $w \in W^*$ и $x \in R_+^n$ выполняется неравенство $d(w) \le y(x)$. Следовательно, позином *y(x)* ограничен снизу положительным числом, в качестве которого можно взять значение $d(\tilde{x})$ в любой точке $\tilde{w} \in W^*$. Можно показать, что в этом случае позином достигает наименьшего значения. При этом число $d(\tilde{x})$ можно рассматривать как оценку снизу наименьшего значения позинома. Отметим, что в некоторых случаях множество *W** может состоять из единственной точки.

Позином может не достигать наименьшего значения, хотя он и ограничен снизу нулем (например, функция $1/x_1$). Как показывает предыдущее рассуждение, это возможно лишь в случае, когда множество *W** пусто.

Теорема 6.4. Регулярный позином *y(x)* достигает в R_+^n своего наименьшего значения $y_* = y(x^*) = d^* = c_1 + c_2 + ... + c_m$ в точке $x^* = (1,1,...,1) \in R_+^n$.

Итак, поиск наименьшего значения регулярного позинома не составляет труда: оно достигается в точке *x** с единичными координатами. Нерегулярный позином *y(x)* может не достигать наименьшего значения, но если известно, что наименьшее значение этим позиномом достигается, то значение $d^*=y(x^*)$ может рассматриваться как оценка сверху для наименьшего значения позинома.

Как отмечено выше, если позином достигает наименьшего значения, то все точки наименьшего значения есть стационарные точки позинома. Поэтому задачу минимизации позинома можно решать, определяя его стационарные точки.

Используя представления (6.3) и (6.4), получаем уравнения для стационарных точек позинома:

$$\frac{\partial y}{\partial x_j} = \frac{1}{x_j} \sum_{i=1}^{m} c_i a_{ij} \prod_{j=1}^{n} x_j^{a_{ij}} = 0, j = 1,...,n. \qquad (6.11)$$

Эти уравнения сложные. Установить по ним существование стационарных точек, а тем более найти их не так просто, особенно при большом числе m слагаемых в (6.3) и дробных показателях степеней.

Введем дополнительные переменные

$$u_i = c_i \prod_{j=1}^{n} x_j^{a_{ij}}, i = 1,...,m, \qquad z_j = \ln x_j, j = 1,...,n.$$

Тогда

$$\ln \frac{u_i}{c_i} = \sum_{j=1}^{n} a_{ij} \ln x_j,$$

и условия (6.11) можно записать в виде двух систем линейных уравнений

$$\sum_{i=1}^{m} a_{ij} u_i = 0, j = 1,...,n, \qquad \sum_{j=1}^{n} a_{ij} z_j = \ln \frac{u_i}{c_i}, i = 1,...,m. \tag{6.12}$$

Теорема 6.5. Если позином $y(x)$ достигает в точке $x^*=(x^*_1,...,x^*_n) \in R^n_+$ наименьшего значения $y_*=y(x^*)$, то двойственная функция d(w) достигает на множестве W^* наибольшего значения $d(w^*)=y_*$ в точке $w^* \in W^*$ с координатами $w_i^* = u_i / y_*,$ где $u_i = c_i p_i(x^*), i = 1,...,m.$.

Теорема 6.6. Пусть двойственная позиному $y(x)$ функция $d(w)$ достигает в точке $w^*=(w^*_1,...,w^*_m) \in W^*$ наибольшего значения и $u_i = w_i^* d(w^*), i = 1,...,m.$. Позином $y(x)$ достигает в R^n_+ наименьшего значения тогда и только тогда, когда система m уравнений

$$\sum_{j=1}^{n} a_{ij} z_j = \ln \frac{u_i}{c_i}, i = 1,...,m, \tag{6.13}$$

где $z_j = \ln x_j, j = 1,....,n,$ имеет решения, принадлежащие R^n_+. При этом любое решение $(z_1^*,...,z_n^*)$ системы определяет точку $x^* = (x_1^*,...,x_n^*)$, где $z^*_j = \ln x^*_j, j = 1,....,n,$, которая является точкой наименьшего значения позинома.

Теоремы 6.4 и 6.5 позволяют задачу поиска стационарных точек позинома, т.е. решение системы уравнений (6.12), заменить задачей исследования на условный максимум двойственной функции, ограничения в которой являются линейными. Такая задача может оказаться более простой, а решив ее, можно найти решение исходной задачи минимизации позинома.

6.3 Линейное программирование

Значительное число планово-производственных задач имеет выражение критерия оптимальности в виде линейной функции от входящих в него переменных. При этом на указанные переменные могут быть также наложены некоторые ограничивающие условия в форме линейных равенств или неравенств. Примером подобных задач является задача отыскания такого распределения ограниченного количества сырья между различными производствами, когда общая стоимость получаемой продукции заданного ассортимента максимальна. Другим примером служит транспортная задача,

когда необходимо так организовать доставку товаров из различных складов к нескольким пунктам назначения, чтобы затраты на перевозку были минимальны.

Решение этих задач, математическая формулировка которых сводится к требованию максимизации или минимизации критерия оптимальности, заданного в виде линейной функции независимых переменных с линейными ограничениями на них, и составляет предмет специального раздела математики *–линейного программирования*, являющегося частным случаем задачи выпуклого программирования.

6.3.1. Постановка задач линейного программирования

В задачах линейного программирования критерий оптимальности представляется в виде:

$$R = \sum_{j=1}^{n} c_j x_j \,, \tag{6.14}$$

где c_j *(j=1,…,n)* –заданные постоянные коэффициенты, положительные или отрицательные, среди которых могут быть также равные нулю.

В терминах линейного программирования соотношение (6.14) иногда также называют линейной формой, а в приложениях линейного программирования к решению экономических задач – экономической функцией.

На выбор оптимальных значений переменных x_j *(j=1,…,n)* накладываются дополнительные условия, которые заключаются в том, что искомая совокупность значений независимых переменных должна удовлетворять системе линейных соотношений, включающей в общем случае как неравенства, так и равенства:

$$\sum_{j=1}^{n} a_{ij} x_j \le b_j \qquad \text{i=1,…,m}_1 \tag{6.15\textit{а}}$$

$$\sum_{j=1}^{n} a_{ij} x_j \ge b_j \qquad \text{i= m}_1 \text{+1,…,m}_2 \tag{6.15\textit{б}}$$

$$\sum_{j=1}^{n} a_{ij} x_j = b_j \quad \text{i= m}_2 \text{+1,…,m} \tag{6.15\textit{в}}$$

Оптимальным решением задачи линейного программирования или, как его еще называют, оптимальным планом является такая совокупность неотрицательных значений независимых переменных

$$x_j = \lambda_j \qquad j=1,\ldots,n \,, \tag{6.16}$$

которая удовлетворяет условиям и обеспечивает в зависимости от поставки задачи максимальное или минимальное значение линейной формы (6.14). В дальнейших рассуждениях обычно подразумевается, что оптимум достигается при максимальном значении формы (6.14). Случай, когда требуется найти

минимально значение линейной формы, может быть сведен к задаче максимизации простым изменением знаков у всех коэффициентов c_j *(j=1,...,n).*

6.3.2. Симплексный метод решения задач линейного программирования

Вывод основных соотношений. Симплексный метод или, как его еще называют, метод последовательного улучшения плана позволяет по известному базисному решению построить другое базисное решение, для которого значение линейной формы больше, чем исходного. Свое название этот метод получил от ограничения, входившего в одну из первых задач, решенных указанным методом. Данное ограничение имеет вид

$$\sum_{j=1}^{n} x_j \le 1$$

и представляет собой выпуклой многогранник в n-мерном пространстве с вершинами на осях координат в точках $x_j = 1$ *(j=1,...,n)*, называемый обычно симплексом.

Для вывода основных соотношений симплексного метода запишем систему уравнений в векторной форме

$$\sum_{j=1}^{n+m} A_j x_j = B, \qquad (6.17)$$

где через A_j и B обозначены векторы

$$A_j = \begin{pmatrix} a_{1j} \\ a_{2j} \\ . \\ . \\ . \\ a_{mj} \end{pmatrix} \quad j=1,...,n+m \qquad B = \begin{pmatrix} b_1 \\ b_2 \\ . \\ . \\ . \\ b_m \end{pmatrix} \qquad (6.18)$$

Допустим, что любые m векторов из общего числа $n+m+1$ векторов A_j и В линейного независимы, т.е. соотношение:

$$\sum_{i=1}^{m} \alpha_i A_{(i)} = 0, \qquad (6.19)$$

где через $A_{(i)}$ *(i=1,...,m)* обозначена совокупность любых m векторов (6.18) выполняется лишь в том случае, если все α_i *(i=1,...,m)* равны нулю.

Любые m векторов, для которых условие (6.19) может быть выполнено только при нулевых значениях α_i , можно принять в качестве базисных векторов или базиса n мерного пространства.

В этом пространстве любой другой вектор A_k можно представить в виде разложения по векторам базиса:

$$A_k = \sum_{i=1}^{m} A_{(i)} x_{(i),k} \ , \tag{6.20}$$

где $x_{(i),k}$ – коэффициент при базисном векторе $A_{(i)}$ в разложении вектора A_k по векторам базиса.

Коэффициенты $x_{(i),k}$ разложения (6.20) могут быть найдены решением системы уравнений

$$\left.\begin{array}{l} a_{1,(1)} x_{(1),k} + a_{1,(2)} x_{(2),k} + ... + a_{1,(m)} x_{(m),k} = a_{1k} \\ a_{2,(1)} x_{(1),k} + a_{2,(2)} x_{(2),k} + ... + a_{2,(m)} x_{(m),k} = a_{2k} \\ ..... \\ a_{m,(1)} x_{(1),k} + a_{m,(2)} x_{(2),k} + ... + a_{m,,(m)} x_{(m),k} = a_{mk} \end{array}\right\} \tag{6.21}$$

коэффициентами, которой являются компоненты базисных векторов, а правые части уравнений представляют собой компоненты небазисного вектора, для которого определяется разложение (6.20).

В приведенных соотношениях индекс базисных векторов заключен в скобки для того, чтобы показать, что фактическая нумерация базисных векторов может отличаться от использованной в соответствии (6.20).

Для векторов A_j $(j=1,...,m+n)$, любые m которых линейно независимы и, следовательно, могут быть приняты в качестве базиса, среди коэффициентов разложения $x_{(i),k}$ $(i=1,...,m)$ небазисного вектора A_k нет ни одного, равного нулю. Это следует из того, что если, например, хотя бы один из указанных коэффициентов равен нулю, а остальные отличны от нуля, то соотношение (6.20) определяет взаимосвязь m векторов и его можно записать в форме (6.19), причем не все α_i будут равны нулю. Однако сказанное противоречит предположению о линейной независимости любых m векторов (6.18), поэтому ни один из коэффициентов $x_{(i),k}$ не может быть равен нулю.

Предположим, что известно какое-нибудь базисное решение $\lambda^{(0)}$ системы (6.17), в котором m значений переменных x_j отличны от нуля, например

$$\left.\begin{array}{l} x_j = 0, j = 1,...,n \\ x_j = \lambda_j^{(0)} \neq 0, j = n+1,...,n+m \end{array}\right\} \tag{6.22}$$

Таким образом, ненулевых значения переменных x_j удовлетворяют векторному уравнению:

$$\sum_{j=n+1}^{n+m} A_j \lambda_j^{(0)} = B \tag{6.23}$$

Векторы A_j $(j=n+1,...,n+m)$, могут быть приняты в качестве базиса m-мерного пространства, поэтому любой небазисный вектор A_k $(k=1,...,n)$,можно представить в виде разложения по векторам этого базиса:

$$\sum_{j=n+1}^{n+m} A_j x_{jk} = A_k \qquad (6.24)$$

Умножим соотношение (6.24) на произвольную положительную константу θ и вычтем полученное выражение из уравнения (6.23). В результате найдем:

$$\sum_{j=n+1}^{n+m} A_j \lambda_j^{(0)} - \theta \sum_{j=n+1}^{n+m} A_j x_{jk} = B - \theta A_k \qquad (6.25)$$

Объединяя слагаемые левой части соотношения (6.25) под знаком одной суммы, перепишем его в виде:

$$A_k \theta + \sum_{j=n+1}^{n+m} A_j (\lambda_j^{(0)} - \theta x_{jk}) = B \qquad (6.26)$$

Поскольку величина θ выбрана произвольно, она может быть взята настолько малой, что независимо от знаков коэффициентов $x_{j,k}$ в соответствии (6.26) выражения

$$y_j = \lambda_j^{(0)} - \theta x_{j,k} \qquad j = \overline{n+1, n+m} \qquad (6.27)$$

будут положительными, так как значения $\lambda_j^{(0)}$ ($j = \overline{n+1, n+m}$), положительны по определению исходного базисного решения (6.22).
Выводя также обозначение

$$y_k = \theta, \qquad (6.28)$$

получим, что значения переменных y_j ($j = k, n+1, ..., n+m$), определяемые выражениями (6.27) и (6.28), при достаточно малой величине θ составляют допустимое решение системы уравнений (6.17), так как все *m+1* его ненулевых составляющих положительны. Для того, чтобы указанное решение стало базисным, необходимо одну из его составляющих сделать равной нулю. В данном случае это можно выполнить соответствующим выбором значения величины θ.

Очевидно, что при $\theta = 0$ имеем исходное базисное решение (6.22). Поэтому, для получения другого базисного решения, отличного от существующего, необходимо взять $\theta > 0$. Возможность нахождения нового решения целиком зависит от того, имеется ли среди коэффициентов разложения x_{jk} ($j = \overline{n+1, n+m}$) вектора A_k по векторам исходного базиса (6.24) хотя бы один положительный. Если все x_{jk} отрицательны, то ни одна из величин y_j, определяемых выражениями (6.27), не может быть сделала равной нулю при любом значении θ.Последнее указывает на невозможность получения нового базисного решения при введении вектора A_k в исходный базис вместо одного из его векторов. В этом случаем вместо вектора A_k следует взять любой другой небазисный вектор и найти его разложение. Если и его коэффициенты разложения все отрицательны, то нужно взять следующий и т.д., пока не будут опробованы все небазисные векторы, или не будет найдено разложение какого либо вектора, содержащее хотя бы один положительный коэффициент.

Пусть не все x_{jk} ($j = \overline{n+1, n+m}$) в разложении вектора (6.24) отрицательны. Тогда при непрерывном возрастании величины θ от значения

$\theta = 0$ первой обратится в ноль та переменная, y_{n+l} $(1 \le l \le m)$ в выражениях (6.27), для которой отношение $\frac{\lambda_{n+1}^{(0)}}{x_{n+1,k}}$ будет минимальным среди всех отношений $\frac{\lambda_j^{(0)}}{x_{j,k}}$ $(j = \overline{n+1, n+m})$, определенных только для положительных значений величины x_{jk}.

Положим

$$\theta = \theta_{n+1,k} = \min_{n+1 \le j \le n+m} \frac{\lambda_j^{(0)}}{x_{j,k}} > 0 \tag{6.29}$$

где знак минимизации означает, что ищется минимальное среди всех положительных отношений:

$$\frac{\lambda_j^{(0)}}{x_{j,k}} \tag{6.30}$$

Допустим, что минимальное значение положительных отношений (6.30) получается для $l = 1$, т.е. с учетом соотношения (6.29) находим:

$$\theta = \theta_{n+1,k} = \frac{\lambda_{n+1}^{(0)}}{x_{n+1,k}} \tag{6.31}$$

При таком наборе величины θ значение переменной y_{n+1} равно нулю, тогда как остальные m значений переменных y_j $(j = k, n+2, ..., n+m)$ отличны от нуля и положительны. Таким образом, вместо исходного базисного решения (6.22) получаем новое базисное решение, компоненты которого определяются выражениями:

$$\begin{cases} \lambda_j^{(1)} = 0, j = 1, ..., n (j \ne k) \\ \lambda_k^{(1)} = \theta, 1 \le k \le n \\ \lambda_{n+1}^{(1)} = \lambda_{n+1}^{(0)} - \theta x_{n+1,k} = 0 \\ \lambda_j^{(1)} = \lambda_j^{(0)} - \theta x_{jk} > 0, j = n+2, ..., n+m \end{cases} \tag{6.32}$$

Согласно выражениям (6.32), соотношение (6.26) может быть теперь записано в виде:

$$A_k \lambda_k^{(1)} + \sum_{j=n+2}^{n+m} A_j \lambda_j^{(1)} = B \tag{6.33}$$

Сравнение выражений (6.23) и (6.33) показывает, что в данном случае в исходном базисе векторов A_j $(j = \overline{n+1, n+m})$, один из них при сделанных предположениях вектор A_{n+1} заменен на вектор A_k, и новое базисное решение (6.32) теперь удовлетворяет системе уравнений (6.33).

Изложенная процедура позволяет находить при известном каком-нибудь базисном решении другое базисное решение, отличающееся от первоначального

одним базисным вектором. Однако при этом еще неясно, как изменяется значение линейной формы

$$R = \sum_{j=1}^{n+m} c_j x_j \tag{6.34}$$

и имеет ли смысл переход к новому базисному решению. Естественно, что этот переход следует осуществить лишь тогда, когда новое значение линейной формы будет больше прежнего, при условии, что оптимум соответствует ее максимуму.

Для выяснения целесообразности перехода к новому базисному решению вычислим значение линейной формы в обоих случаях.

Подставляя первоначальное базисное решение (6.22) в выражение (6.34), получим:

$$R^{(0)} = \sum_{j=n+1}^{n+m} c_j \lambda_j^{(0)} \tag{6.35}$$

Число членов под знаком суммирования (6.34) сократилось за счет того, что только последние m значений переменных x_j отличны от нуля в исходном базисном решении (6.22).

Для нового базисного решения с учетом значений его ненулевых компонентов (6.32) можно также вычислить значение линейной формы (6.34), которое определяется формулой:

$$R_1^{(1)} = c_k \theta + \sum_{j=n+1}^{n+m} c_j (\lambda_j^{(0)} - \theta x_{jk}) \tag{6.36}$$

Вычитая из выражения (6.36) выражение (6.35), получим:

$$\Delta R = R^{(1)} - R^{(0)} = \theta (c_k - \sum_{j=n+1}^{n+m} c_j x_{jk}) \tag{6.37}$$

Нетрудно увидеть, что если величина

$$z_k = \sum_{j=n+1}^{n+m} c_j x_{jk} \tag{6.38}$$

удовлетворяет соотношению

$$c_k > z_k, \tag{6.39}$$

то значение ΔR, характеризуемое выражением (6.37), будет положительным, и, следовательно, величина критерия оптимальности R при переходе к новому базисному решению возрастает.

Если же

$$c_k < z_k \ , \tag{6.40}$$

то $\Delta R < 0$ и переход к новому базисному решению приводит к уменьшению критерия оптимальности.

Разности

$$v_k = z_k - c_k \tag{6.41}$$

в задачах линейного программирования иногда называют *маргинальными значениями*. Величины v_k имеют определенный смысл в так называемых двойственных задачах. Условие (6.39) означает, что если маргинальное значение v_k отрицательно, т.е.

$$v_k < 0 \ , \tag{6.42}$$

то переход к новому базисному решению вызывает увеличение критерия оптимальности (6.34). Наоборот, если

$$v_k > 0 , \tag{6.43}$$

то переход к новому базисному решению сопровождается уменьшением критерия оптимальности (6.34).

Таким образом, вопрос о целесообразности перехода к новому базисному решению может решаться проверкой условия (6.39) или (6.42) еще до выбора значения θ, для чего необходимо знать лишь коэффициенты x_{jk} в разложении (6.24) вектора A_k $(1 \le k \le n)$ по векторам исходного базиса A_j ($j = \overline{n+1, n+m}$).

Следует заметить, что если при выполнении условия (6.39) все коэффициенты x_{jk} отрицательны, то приращение ΔR (6.37) может быть сделано сколь угодно большим за счет выбора достаточно большого положительного значения θ. Другими словами, если обнаружено, что все коэффициенты разложения (6.24) какого-либо небазисного вектора A_k отрицательны и, вместе с тем, выполнено условие (6.39), то максимальное значение критерия оптимальности не ограничено.

7 Аналитические методы нелинейного программирования

Общая задача нелинейного программирования может включать *ограничения типа равенства* или *ограничения типа неравенства*, а также оба вида ограничений одновременно. В этой главе сначала кратко обсудим задачу нелинейного программирования, в которую входят только ограничения типа равенства, а затем перейдем к общему случаю.

Задачу нелинейного программирования с ограничениями типа равенства можно записать следующим образом:

$$\begin{cases} f_0(x) \to \min; \\ f_l(x) = 0, l = \overline{1,k} \end{cases} \tag{7.1}$$

В данном случае *допустимое множество* $\Omega = \left\{ x \in R^n; f_l(x) = 0, l = \overline{1,k} \right\}$.

Сформулированную задачу в случае дифференцируемых функций $f_l(x)$ можно связать с задачей исследования целевой функции на условный экстремум. Если точно x^* есть решение задачи (7.1), т.е. если *целевая функция* $f_0(x)$ достигает в этой точке наименьшего значения на множестве Ω, то точка x^* является точкой условного локального минимума функции $f_0(x)$ при условиях $f_l(x) = 0, l = \overline{1,k}$ (эти уравнения называют *уравнениями связи*). Поэтому решать задачу (7.1) можно следующим образом. Найти все точки, в которых функция может иметь условный локальный минимум, а затем путем сравнения значений функции выбрать из этих точек ту, в которой функция имеет наименьшее значение. Если задача имеет решение (т.е. целевая функция при заданных ограничениях достигает наименьшего значения), то этим решением будет найденная точка.

Поиск точек «подозрительных» на условный экстремум, выполняют с помощью необходимого условия условного локального экстремума, суть которого в следующем. Если точка x^* является точкой уловного экстремума функции $f_0(x)$ при условиях $f_l(x) = 0, l = \overline{1,k}$, причем функция $f_l(x), l = \overline{0,k}$, непрерывно дифференцируемы в окрестности точки x^* матрица Якоби *J(x)* векторной функции $f(x) = (f_1(x) \ldots f_k(x))$ в точке x^* имеет ранг k (максимальный ранг, равный количеству строк), то существуют такие множители Лагранжа $\lambda_1, \ldots, \lambda_k$, что для функции Лагранжа

$$L(x) = f_0(x) + \sum_{l=1}^{k} \lambda_l f_l(x) \tag{7.2}$$

в точке x^* выполняется необходимое условие экстремума

$$gradL(x^*) = 0,$$

или

$$\frac{\partial L(x^*)}{\partial x_1} = \frac{\partial L(x^*)}{\partial x_2} = ... = \frac{\partial L(x^*)}{\partial x_n} = 0.$$

Поиск стационарных точек функции Лагранжа сводится к решению системы $n+k$ уравнений

$$\begin{cases} \dfrac{\partial L(x)}{\partial x_j} = 0, j = \overline{1,n}; \\ f_1(x) = 0, l = \overline{1,k}, \end{cases}$$

содержащей $n+k$ неизвестных $x_1,...,x_n,\lambda_1,...,\lambda_k$.

Таким образом, исследование функции на условный экстремум состоит в выделении некоторого набора точек, «подозрительных» на экстремум, и в проверке каждой точки, является ли она в действительности точкой условного экстремума. Набор точек «подозрительных» на экстремум, можно разделить на три группы:

1. точки, в которых хотя бы одна из функций $f_l(x), l = \overline{0,k}$, не дифференцируема или одна из ее частных производных разрывна;
2. точки, в которых ранг матрицы Якоби *J(x)* системы уравнений связи меньше количества k уравнений связи;
3. стационарные точки функции Лагранжа.

Отличие задачи минимизации при ограничениях типа равенства от задачи исследования на условный экстремум состоит в том, что в задаче минимизации нет необходимости проверять, является ли точка, «подозрительная» на экстремум, точкой условного локального минимума. Можно просто сравнить значения минимизируемой функции в этих точках и выбрать ту, в которой значение функции наименьшее. Эта точка и будет решением задачи минимизации. Правда, последнее утверждение верно лишь в случае, когда задача минимизации имеет решение. В действительности возможна ситуация, когда целевая функция имеет точки условного локального минимума, но не достигает наименьшего значения.

Необходимое условие условного локального экстремума можно модифицировать так, что определение стационарных точек функции Лагранжа будет включать в себя и поиск точек, в которых нарушается условие на ранг матрицы Якоби системы уравнений связи (необходимое условие условного локального экстремума формулируется в предложение, что ранг системы уравнений связи максимален и совпадают с числом уравнений связи). Для этого изменим понятие функции Лагранжа, используя вместо (7.2) формулу

$$\widetilde{L}(x) = \lambda_0 f_0(x) + \sum_{l=1}^{k} \lambda_l f_l(x), \qquad (7.3)$$

в которую добавлен еще один множитель Лагранжа λ_0. Следующее утверждение называют ***обобщенным правилом множителей Лагранжа.***

Теорема 7.1. Если $x^* \in R^n$ – точка условного экстремума функции $f_0(x)$ при условиях $f_l(x) = 0, l = \overline{1,k}$, причем функции $f_l(x), l = \overline{0,k}$ непрерывно дифференцируемы в некоторой окрестности точки x^*, то существуют такие множители Лагранжа $\lambda_l, l = \overline{0,k}$, не все равные нулю, что для функции Лагранжа $\tilde{L}(x)$ точка x^* является стационарной, т.е.

$$grad\tilde{L}(x^*) = 0. \tag{7.4}$$

Нетрудно увидеть, что множители Лагранжа, существование которых вытекает из теоремы 7.1, определяются однозначно: их можно, не нарушая утверждения теоремы, умножить на любой постоянный множитель. Поэтому для определения *k+1* множителя Лагранжа $\lambda_0,...,\lambda_k$ и *n* координат точки x^* достаточно *n+k* уравнений. Такое количество дают условие (7.4) (это *n* уравнений) и совокупность уравнений связи (их количество равно *k*).

Общую задачу нелинейного программирования можно сформулировать следующим образом:

$$\begin{cases} f_0(x) \to \min; \\ f_l(x) = 0, l = \overline{1,k}; g_i(x) \le 0, i = \overline{1,m} \end{cases} \tag{7.5}$$

При этом предполагается, что одна из фигурирующих в задаче функций не является линейной. Считаем, что все функции $f_l(x), l = \overline{0,k}$ и $g_i(x), i = \overline{1,m}$, определены на некотором открытом множестве $X \subset R^n$. Упрощая изложение, мы далее ограничимся случаем $X = R^n$. В конкретной ситуации можно добиться выполнения этого ограничения, доопределив все функции вне множества *X*, например, большим постоянным значением, которое не повлияет на процесс минимизации целевой функции.

Сформулированная задача представляет собой частный случай *общей задачи математического программирования*, заключающейся в определении наименьшего значения *целевой функции* $f_0(x)$ *на допустимом множестве*

$$\Omega = \left\{ x \in X : f_i(x) = 0, l = \overline{1,k}, g_i(x) \le 0, i = \overline{1,m} \right\}. \tag{7.6}$$

Отметим, что если все функции $f_l(x), l = \overline{1,k}$ и $g_i(x), i = \overline{1,m}$, непрерывны в R^n, то множество Ω замкнуто. В самом деле, это множество можно представить как пересечение конечного числа множеств вида $\left\{ x \in R^n : \varphi(x) \le \alpha \right\}$, образованных непрерывной в R^n функцией $\varphi(x)$. В данном случае в качестве $\varphi(x)$ используются функции $f_l(x)$, $-f_l(x)$, $l = \overline{1,k}$, и $g_i(x), i = \overline{1,m}$, а $\alpha = 0$. Такие множества являются замкнутыми. Поэтому и множество Ω, как пересечение конечного числа замкнутых множеств, является замкнутым.

Большинство известных методов решения задачи нелинейного программирования позволяют найти лишь точки локального минимума целевой функции на заданном множестве. В этой связи важную роль играет априорная информация о существовании решения задачи, т.е. информация о том, достигает

ли целевая функция наименьшего значения на допустимом множестве. Если допустимое множество Ω в задаче нелинейного программирования замкнуто, то доказательство существования в этой задаче можно строить, проверяя непрерывность целевой функции $f_0(x)$ и ограниченность при некотором α множества $\Omega_\alpha = \{x \in \Omega : f_0(x) \le \alpha\}$. В частности, если множество Ω ограничено, а целевая функция непрерывна на Ω, то задача нелинейного программирования имеет решение.

Однако проверить ограниченность множества Ω (или Ω_α) не просто. Поэтому интересны другие условия, налагаемые на целевую функцию и ограничения и позволяющие делать заключение о существовании решения без использования ограниченности допустимого множества. В *задаче выпуклого программирования*, в которой целевая *функция выпукла* и допустимое множество Ω выпукло, любая точка локального минимума целевой функции является решением этой задачи.

Теорема 7.2. Если функции $g_i(x), i = \overline{1,m}$, выпуклы в R^n, то множество

$$\widetilde{\Omega} = \left\{x \in R^n : g_i(x) \le 0, i = \overline{1,m}\right\} \tag{7.7}$$

является выпуклым.

Ограничения типа неравенства в задаче нелинейного программирования введением дополнительных переменных можно преобразовать в ограничения типа равенства. Например, ограничения $g_i(x) \le 0, i = \overline{1,m}$, с помощью дополнительных переменных z_i можно записать в виде

$$g_i(x) + z_i^2 = 0, i = \overline{1,m}. \tag{7.8}$$

Таким образом, общую задачу нелинейного программирования можно свести к частному случаю, когда в задаче нет ограничений типа неравенства. Это позволяет в решении задачи использовать технику исследования целевой функции на условный экстремум.

Впрочем, техника исследования функции на условный экстремум может использоваться и непосредственно в задаче, имеющей ограничения типа неравенства.

Теорема 7.3. Если точка $x^* \in \Omega$ является точкой локального минимума функции $f_0(x)$ на множестве Ω вида (7.6), причем функции $f_l(x), l = \overline{0,k}$ и $g_i(x), i = \overline{1,m}$ непрерывно дифференцируемы в окрестности точки x^*, то существуют такие числа $\lambda_l, l = \overline{0,k}$ и $\mu_i \ge 0, i = \overline{1,m}$ одновременно не равные нулю, что для функции

$$\widetilde{L}(x) = \lambda_0 f_0(x) + \sum_{l=1}^{k} \lambda_l f_l(x) + \sum_{i=1}^{m} \mu_l g_l(x)$$

выполняется необходимое условие экстремума

$$grad\widetilde{L}(x^*) = 0$$

и, кроме того, выполняются ***условия дополняющей нежесткости***

$$\mu_l g_l(x^*) = 0, i = \overline{1,n}$$

Теорему 7.3 принято называть ***теоремой Куна–Таккера,*** а функцию $\tilde{L}(x)$ – ***функцией Лагранжа.*** Хотя функция Лагранжа определена для любых значений параметров $\lambda_l, l = \overline{0,k}$ и $\mu_i, i = \overline{1,m}$, мы, учитывая утверждение теоремы Куна–Таккера, будем рассматривать ее лишь при $\mu_i \geq 0, i = \overline{1,m}$.

Отметим особую роль в этой теореме условий дополняющей нежесткости. В рассматриваемой точке x^* локального минимума функции $f_0(x)$ на множестве Ω каждое из ограничений $g_i(x) \leq 0, i = \overline{1,m}$, выполняется либо в виде равенства, $g_i(x^*) = 0$, либо в виде строгого неравенства, $g_i(x^*) < 0$. Ограничения первого вида называют ***активными ограничениями***, в то время как остальные – ***неактивные ограничениями***. Для активных ограничений соответствующее условие дополняющей нежесткости выполнено. Неактивные ограничения в силу непрерывности фигурирующих в задаче функций выполняются в виде строгого неравенства не только в точке x^*, но и в некоторой окрестности этой точки. Поэтому после удаления этих ограничений из задачи точка x^* останется точкой локального минимума. Полагая для таких ограничений $\mu_i = 0$, мы, с одной стороны, обеспечиваем выполнения условия дополняющей нежесткости, а с другой – удаляем соответствующее слагаемое из функции Лагранжа. В конечном счете утверждение теоремы Куна-Таккера сводится к *обобщенному принципу Лагранжа*.

Литература

1. Аттетков, А.В. Методы оптимизации: учеб. для вузов / А.В. Аттетков, С.В. Галкин, В.С. Зарубин. – 2-е изд., стереотип. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. – 440 с.Лесин В.В., Лисовец Ю.П. Основы методов оптимизации. – М.: Изд-во МАИ, 1995. – 341 с.

2. Лесин, В.В. Основы методов оптимизации / В.В. Лесин, Ю.П. Лисовец. – М.: Изд-во МАИ, 1995. – 341 с.

3. Бояринов, А.И., Методы оптимизации в химической технологии / А.И. Бояринов, В.В. Кафаров. – 2-е изд. – М.: Химия, 1975. – 576 с.

4. Химмельблау, Д. Прикладное нелинейное программирование: пер. с англ. / Д. Химмельблау. – М.: МИР, 1975. – 536 с.

5. Банди, Б. Методы оптимизации. Вводный курс: пер. с англ. / Б. Банди. – М.: Радио и связь, 1988. – 128 с.

6. Гилл, Ф. Практическая оптимизация: пер. с англ. / Ф. Гилл, У. Мюррей, М. Райт. – М.: Мир, 1985. – 510 с.

7. Корн, Г. Справочник по математике. Для научных работников и инженеров: пер. с англ. / Г. Корн, Т. Корн. – 5-е изд. – Наука, 1984. – 830 с.