

APPLIED
NONLINEAR
PROGRAMMING

David M. Himmelblau

The University
of Texas, Austin, Texas

McGraw-Hill Book Company
1972

APPLIED
NONLINEAR
PROGRAMMING

David M. Himmelblau

The University
of Texas, Austin, Texas

McGraw-Hill Book Company
1972

Д. Химмельблau

ПРИКЛАДНОЕ
НЕЛИНЕЙНОЕ
ПРОГРАММИРОВАНИЕ

Перевод с английского
И. М. БЫХОВСКОЙ и Б. Т. ВАВИЛОВА

Под редакцией
М. Л. БЫХОВСКОГО

ИЗДАТЕЛЬСТВО «МИР»
МОСКВА 1975

Книга посвящена методам оптимального управления системами с нелинейными целевыми функциями. Описаны методы нелинейного программирования как при отсутствии ограничений на управляющие переменные, так и при наличии ограничений. Рассматриваются такие вопросы, как возможность получения решения, время оптимизации, точность решения и т. д. Для наиболее важных методов приводятся программы решения на языке ФОРТРАН. Каждая глава содержит много примеров, не только поясняющих теорию, но и иллюстрирующих необходимые вычислительные процедуры и используемые программы.

Книга представляет интерес для специалистов по автоматическому управлению, вычислительной технике и прикладной математике.

Редакция литературы по новой технике

(C) Перевод на русский язык, «Мир», 1974

ПРЕДИСЛОВИЕ

Цель этой книги состоит в том, чтобы в доступной форме изложить некоторые из наиболее эффективных методов нелинейного программирования и дать сравнительную оценку этих методов. Для решения общей задачи нелинейного программирования было предложено довольно много алгоритмов, однако лишь немногие из них оказались эффективными для задач большой размерности. Ни один из этих алгоритмов не имеет по отношению к другим таких преимуществ, чтобы его можно было считать универсальным средством решения любых задач нелинейного программирования.

В данной книге описанию используемых методов уделяется больше внимания, чем математическим доказательствам сходимости алгоритмов нелинейного программирования для определенных типов задач. Такие доказательства, разумеется, важны, однако они применимы только к весьма узким категориям задач и могут служить лишь как дополнительная информация для исследователя, применяющего тот или иной алгоритм. В отличие от линейного программирования при решении задачи нелинейного программирования выбранный алгоритм может оказаться эффективным, даже если не удается доказать его сходимость. Справедливо и обратное утверждение, а именно: наличие доказательства сходимости алгоритма в частных случаях может не означать, что он окажется удовлетворительным для более сложных задач.

В книге подробно описаны те алгоритмы нелинейного программирования, которые оказались достаточно эффективными на практике. При сравнении алгоритмов были использованы следующие критерии:

- 1) надежность;
- 2) скорость решения;
- 3) время подготовки задачи для решения;
- 4) точность решения;
- 5) степень выполнения ограничивающих условий.

Рассматриваемые методы предназначены для оптимизации значения некоторой нелинейной функции при ограничениях в виде равенств и (или) неравенств, содержащих функции большого числа переменных. Все эти переменные являются детерминированными (в отличие от случайных, или стохастических, переменных). Описанные методы могут быть практически реализованы лишь при помощи современных цифровых или гибридных ЭВМ. Использование аналоговых вычислительных машин не рассматривается. Не рассматриваются также ни целочисленное (или дискретное) программирование, ни методы поиска оптимального решения динамических задач, т. е. задач, в которых время является одним из параметров.

В этой книге мы не стремимся раскрыть отдельные тонкости методов оптимизации, однако здесь приведены все необходимые подробности, позволяющие читателю проследить существенные этапы каждого из рассматриваемых методов. Нередко оказывается, что детали программирования некоторого алгоритма, особенно в часто повторяемых процедурах, заметно влияют на качество работы алгоритма в целом. Подробно рассмотренные примеры в конце каждого раздела поясняют вычислительные аспекты алгоритмов; для иллюстрации логической структуры алгоритмов приведено большое количество блок-схем. Приложение Б содержит несколько машинных программ для наиболее удачных алгоритмов. Благодаря

тому что для всех алгоритмов используются одни и те же обозначения, можно глубже поять их структурные связи друг с другом и общие свойства.

Книга состоит из трех частей. Первая часть содержит две главы. Глава 1 представляет собой краткое введение; в гл. 2 формулируется общая задача нелинейного программирования, рассматривается связь нелинейного программирования с другими видами математического программирования и, наконец, устанавливаются необходимые и достаточные условия существования оптимального решения.

В второй части рассматриваются алгоритмы нелинейного программирования при отсутствии ограничений. В гл. 3 описывается градиентный метод, метод вторых производных и другие связанные с ними стратегии, в которых используются производные. Глава 4 посвящена рассмотрению стратегий поиска. В гл. 5 дается оценка различных алгоритмов оптимизации при отсутствии ограничений.

В третьей части описываются алгоритмы оптимизации при наличии ограничений. Глава 6 посвящена рассмотрению методов линеаризации. В гл. 7 изложены методы штрафных функций, а в гл. 8 описан метод скользящего допуска. Оценки алгоритмов оптимизации при наличии ограничений приводятся в гл. 9.

Приложение А содержит ряд упражнений с соответствующими решениями. Кроме того, в каждой главе (за исключением первой) приводятся дополнительные задачи, предлагаемые читателю для решения.

Для понимания описанных в книге алгоритмов необходимо знание основ математического анализа, некоторое умение обращаться с матрицами и векторами, а также знакомство с методами решения задач линейного программирования. В приложении В приводятся основные сведения из матричной алгебры. Для чтения машинных программ необходимо знание основ программирования на ФОРТРАНе. Однако эти программы снабжены необходимыми инструкциями, так что их можно использовать, владея лишь методикой перфорирования. Такого рода «механический» подход к использованию программ иногда оказывается вполне приемлемым, однако при отсутствии должной осторожности он может привести к ошибкам.

Д. М. Химмельблау

Ч а с т ь I

ПРЕДВАРИТЕЛЬНЫЕ СВЕДЕНИЯ



В первой части книги сформулирована задача нелинейного программирования, показана ее связь с реальными физическими задачами, а также приведена терминология, связанная с нелинейным программированием. Кроме того, здесь описаны методы, с помощью которых можно определить, действительно ли предполагаемое оптимальное решение является оптимальным.

Г л а в а 1

ВВЕДЕНИЕ



На протяжении всей своей истории люди при необходимости принимать решения прибегали к сложным ритуалам. Они устраивали торжественные церемонии, приносили в жертву животных, гадали по звездам и следили за полетом птиц. Они полагались на народные приметы и старались следовать примитивным правилам, облегчающим им трудную задачу принятия решений. В настоящее время для принятия решения используют новый и, по-видимому, более научный «ритуал», основанный на применении электронно-вычислительной машины. Без современных технических средств человеческий ум, вероятно, не может учесть многочисленные и разнообразные факторы, с которыми сталкиваются при управлении предприятием, конструировании ракеты или регулировании движения транспорта. Существующие в настоящее время многочисленные математические методы оптимизации уже достаточно развиты, что позволяет эффективно использовать возможности цифровых и гибридных вычислительных машин. Одним из этих методов является математическое программирование, включающее в себя как частный случай нелинейное программирование.

Термин «математическое программирование» предложен Робертом Дорфманом приблизительно в 1950 г.; теперь он объединяет линейное программирование, целочисленное программирование, выпуклое программирование, нелинейное программирование и программирование при наличии неопределенности. Нелинейное программирование имеет дело с оптимизацией нелинейных функций при линейных и (или) нелинейных ограничениях. Типичными областями его применения являются прогнозирование, планирование промышленного производства, управление товарными ресурсами, контроль качества выпускаемой продукции, планирование обслуживания и ремонта, проектирование технологических линий (процессов), учет и планирование капиталовложений. Пока еще не существует общего метода решения нелинейных задач оптимизации, такого, как, например, симплексный алгоритм, разработанный для задач линейного программирования. Нелинейное программирование при решении задач включает в себя элементы экспериментирования. Его развитие до сих пор сводилось к предложениям частных алгоритмов,

программированию их, проверке результатов применения этих алгоритмов в конкретных задачах, представляющих практический интерес, и построению лучших алгоритмов на основе приобретенного опыта.

Последние двадцать лет в области математического программирования значительные усилия были сконцентрированы на линейном программировании. Полученные результаты столь значительны, что достигнутый здесь уровень позволяет решать большинство практических задач. Что же касается нелинейного программирования, то, хотя здесь и было предложено большое число различных стратегий поиска решений, успешное применение нашли лишь немногие алгоритмы. Область применения разработанных алгоритмов нелинейного программирования весьма ограничена. В связи с дальнейшим развитием ЭВМ и растущей необходимостью более точно решать задачи, представляющие практический интерес, возникает необходимость в методах решения задач нелинейного программирования с более широкой областью применимости.

Большинство практических задач имеет несколько (а некоторые, возможно, даже бесконечное число) решений. Целью оптимизации является нахождение наилучшего решения среди многих потенциально возможных в соответствии с некоторым критерием эффективности или качества. Задача, допускающая лишь одно решение, не требует оптимизации. Оптимизация может быть осуществлена при помощи многих стратегий, начиная с весьма сложных аналитических и численных математических процедур и кончая разумным применением простой арифметики. Предполагая, что подлежащая оптимизации задача некоторым образом определена (не обязательно математически), можно классифицировать общие методы оптимизации следующим образом:

1. *Аналитические методы*, использующие классические методы дифференциального и вариационного исчислений. Эти методы заключаются в определении экстремума функции $f(x)$ путем нахождения тех значений x , которые обращают в нуль производные $f(x)$ по x . В случае поиска экстремума $f(x)$ при наличии ограничений применяются такие методы, как метод множителей Лагранжа и метод ограниченных вариаций. При использовании аналитических методов задача оптимизации должна быть сформулирована математически с тем, чтобы можно было обращаться со всеми фигурирующими в ней функциями и переменными при помощи известных правил. Для решения больших существенно нелинейных задач аналитические методы оказываются непригодными, и поэтому в данной книге они не рассматриваются.

2. *Численные методы*, использующие предшествующую информацию для построения улучшенных решений задачи при помощи итерационных процедур. Численные методы применяются для решения задач, которые не могут быть решены аналитически, и, поскольку практические задачи поддаются решению численными

методами, именно эти методы нелинейного программирования являются предметом обсуждения в данной книге.

К другим общим методам, которые эффективно применяются при решении задач оптимизации, но которые здесь не рассматриваются, относятся следующие:

3. *Графические методы*, основанные на графическом изображении функции, подлежащей максимизации или минимизации, в зависимости от одной или нескольких переменных. Экстремум функции в этом случае получают непосредственно путем анализа ее графика. Преимущество графических методов состоит в том, что они просты и сразу показывают, существует решение или нет. С другой стороны, они применимы в тех случаях, когда критерий качества является функцией одной или максимум двух независимых переменных.

4. *Экспериментальные методы*. Экстремум функции можно иногда найти, экспериментируя непосредственно с реальными переменными вместо того, чтобы исследовать соответствующую математическую модель. Результаты одного эксперимента используются для планирования следующего эксперимента, позволяющего получить улучшенные результаты.

5. *Методы исследования различных вариантов*. Эти методы основаны на анализе нескольких возможных решений одной и той же задачи с целью выбора наилучшего. Таким образом, «наилучшее» решение, полученное методом исследования различных вариантов, будет скорее всего лишь субоптимальным.

При выборе наиболее подходящего способа описания реальных процессов приходится сталкиваться с рядом трудностей, которые для удобства обсуждения можно подразделить на две группы. Одна группа связана с построением математической модели процесса, а другая — с численными методами решения. В этой книге мы можем только отметить эти трудности и, где возможно, указать пути их устранения при описании того или иного конкретного алгоритма.

Математическая модель содержит функции, участвующие в процедуре оптимизации. Очевидно, для того чтобы искомый экстремум имел физический смысл, выбранная модель должна адекватно отражать существенные черты реального процесса. Но даже если это требование выполнено, при построении модели встречаются следующие типичные затруднения:

1. Оптимизируемый критерий может быть *нечувствительным* к изменениям независимых (оптимизируемых) переменных и поэтому не удается определить четко выраженный экстремум.

2. Оптимизируемый критерий или некоторые из ограничений могут принимать в области поиска решения неограниченные значения; значения частных производных в математической модели также могут стать неограниченными. Особенно подвержены этой опасности

модели с полиномами в знаменателе. Так, например, значения функции

$$y = \frac{b_0 + b_1 x_1}{b_2 x_1 + b_3 x_2}$$

и ее первой частной производной по x_1

$$\frac{\partial y}{\partial x_1} = \frac{-b_0 b_2 + b_1 b_3 x_2}{(b_2 x_1 + b_3 x_2)^2}$$

обращаются в бесконечность при $b_2 x_1 = -b_3 x_2$. Эту трудность можно преодолеть, ограничив области допустимых значений независимых переменных путем введения дополнительных ограничений в задачу, или же изменив формулировку самой математической модели.

3. Переменные могут быть плохо масштабированы. Трудности масштабирования могут возникнуть, например, когда один из членов в выражении для критерия имеет существенно иной порядок величины, чем другой. При этом критерий становится нечувствительным к изменениям значений переменных в мельчайшем члене. Например, значение целевой функции

$$y = 100x_1^2 - 0,010x_2^2$$

будет мало зависеть от изменения x_2 , если только значение x_2 вследствие используемых физических единиц измерения не окажется много больше значения x_1 . Если x_2 представляет собой величину того же порядка, что и x_1 , то либо одну переменную, либо две переменные можно умножить на масштабные множители, в результате чего оба члена в правой части приведенного выше соотношения окажутся величинами одного и того же порядка. Положим, например,

$$\tilde{x}_1 = 10x_1, \quad x_1^2 = 10^{-2}\tilde{x}_1^2,$$

$$\tilde{x}_2 = 10^{-1}x_2, \quad x_2^2 = 10^2\tilde{x}_2^2.$$

Тогда члены в выражении для целевой функции становятся величинами одного порядка. После того как найден экстремум для

$$y = \tilde{x}_1^2 - \tilde{x}_2^2,$$

можно определить значения x_1 и x_2 по значениям \tilde{x}_1 и \tilde{x}_2 . Конечно, не всегда удается так легко изменить масштаб функций в математической модели, как это сделано в рассмотренном примере.

4. В плохо построенной математической модели переменные могут оказаться взаимосвязанными. Взаимное влияние переменных можно проиллюстрировать на примере очень простого критерия, в который входит произведение двух параметров:

$$y = 2x_1 x_2 + 10.$$

Здесь каждая из переменных x_1 и x_2 может принимать различные значения при заданном значении произведения x_1x_2 . В том случае, когда имеет место взаимное влияние переменных, осуществлять масштабирование гораздо сложнее. Для исключения членов, содержащих произведение двух переменных, квадратичные формы можно привести к каноническому виду. При этом определяются новые координатные оси, называемые *главными осями*, относительно которых данная поверхность второго порядка симметрична. Например, поверхность

$$y = 7x_1^2 + 6x_2^2 + 5x_3^2 - 4x_1x_2 - 4x_2x_3 - 6x_1 - 24x_2 + 18x_3 + 18$$

путем смещения начала координат и поворота координатных осей может быть приведена к виду

$$y - 18 = 3\tilde{x}_1^2 + 6\tilde{x}_2^2 + 9\tilde{x}_3^2.$$

В новой системе координат масштабирование каждого члена значительно проще, чем в исходной системе. Нелинейные функции можно сделать квадратичными при помощи подходящего преобразования модели или разложения соответствующих функций в ряд Тейлора с сохранением конечного числа членов. Более тонким примером, в котором взаимное влияние переменных оказывается также весьма ощутимым, является следующий:

$$y = x_1 e^{b_1 x_2}.$$

5. В математической модели может оказаться, что некоторые из переменных исключаются путем их надлежащего преобразования. Пусть, например,

$$y = x_1^2 + 2x_1x_2 + x_2^2 + 2 = (x_1 + x_2)^2 + 2.$$

После преобразования $x_1 + x_2 = \tilde{x}_1$ получим

$$y = \tilde{x}_1^2 + 2.$$

Таким образом, вместо двух переменных в выражении для y осталась только одна переменная \tilde{x}_1 , которую и следует варьировать, чтобы найти экстремум y .

Вторая группа трудностей связана с численными методами решения задачи оптимизации:

1. Как выбрать подходящие начальные значения независимых переменных? Поскольку задача содержит нелинейные функции, то в отличие от анализа линейных систем при этом возможно существование нескольких экстремумов. Следовательно, если начальные значения переменных находятся слишком далеко от искомого экстремума, оптимизация может закончиться не в точке глобального экстремума, а в некоторой точке, соответствующей локальному экстремуму. Часто приблизительно оптимальные значения незави-

смых переменных можно получить на основе ранее проведенных исследований или исходя из физических соображений. В крайнем случае можно выбрать несколько начальных векторов из допустимой области и проверить, дают ли они одно и то же значение критерия в точке экстремума. Но при таком подходе имеются свои трудности; о них уже упоминалось в связи с обсуждением проблемы построения адекватной модели.

2. Как учесть стохастическую (случайную) природу физических переменных? Реальная возможность того, что коэффициенты и переменные в математической модели могут быть случайными величинами, в этой книге не рассматривается.

3. Как уменьшить ошибки вычислительной процедуры? Погрешности, возникающие при учете лишь конечного числа членов при разложении функций в ряды, снижают эффективность многих алгоритмов. Устойчивость решения зависит от того, сходится ли в пределе решение «аппроксимирующей» задачи нелинейного программирования к решению исходной задачи. Могут возникнуть также неприятности из-за ошибок округления в процессе оптимизации, особенно при аппроксимации производных разностными выражениями.

Как и любой математический аппарат, методы нелинейного программирования нельзя слепо применять для решения той или иной задачи без тщательного предварительного анализа. Практическое применение методов нелинейного программирования требует от исследователя определенного искусства. При этом совершенно необходимо корректное построение модели и применение подходящих численных процедур.

Книга состоит из трех частей. Первая часть посвящена проблеме нелинейного программирования. Во второй части дается описание различных методов нелинейного программирования при отсутствии ограничений и проводится их сравнение. Наконец, в третьей части рассматриваются рабочие алгоритмы для нелинейного программирования при наличии ограничений и проводится их сравнительная оценка. В приложениях содержатся упражнения (вместе с решениями), а также неопубликованные до сих пор машинные программы алгоритмов нелинейного программирования.

Глава 2

ЗАДАЧА НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ И ЕЕ ОПТИМАЛЬНОЕ РЕШЕНИЕ



В технике, экономике и естественных науках, как, впрочем, и в других областях, часто возникают задачи оптимизации сложной совокупности оборудования, операций, цепей или процессов. Требуется минимизировать или максимизировать некоторую функцию, называемую *целевой функцией*, которая характеризует цену, вес, общую эффективность или нечто подобное при определенных ограничениях. Подобные задачи оптимизации, сформулированные математически, могут быть объединены под общим названием *задача нелинейного программирования*; методы решения таких задач лежат в основе нелинейного программирования. В этой главе сначала до некоторой степени формально описывается общая задача нелинейного программирования и некоторые специальные вопросы. Затем на примере демонстрируется связь между тем, что мы хотели бы получить при оптимизации реального процесса, и математическим описанием оптимизации. Далее приводятся необходимые определения и терминология и, наконец, описываются условия оптимальности.

2.1. ЗАДАЧА ЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Задачей линейного программирования называется задача, в которой минимизируемым или максимизируемым критерием является *линейная* функция, причем на переменные налагаются ограничения, которые представляются *линейными* функциями. Комбинация скаляров или векторов, обычно обозначаемых X_i , называется *линейной*, если она может быть записана в виде

$$c_1X_1 + c_2X_2 + \cdots + c_nX_n, \quad (2.1.1)$$

где все коэффициенты c — константы. Например, функция

$$4x_1 + 3x_2 + 5x_3 + 2$$

линейна относительно переменных x_1, x_2, x_3 , тогда как функция

$$2x_1^2 + x_1x_2 + 3e^{x_3}$$

нелинейна относительно тех же переменных. Величины X_1, \dots, X_n называются *линейно зависимыми*, если для некоторого набора c_i в предположении, что не все c_i равны нулю, имеет место следующее

равенство:

$$c_1X_1 + c_2X_2 + \dots + c_nX_n = \sum_{i=1}^n c_iX_i = 0. \quad (2.1.2)$$

С другой стороны, если $\sum_{i=1}^n c_iX_i = 0$ только тогда, когда все коэффициенты c_i равны нулю, то X_1, \dots, X_n называются *линейно независимыми*.

Линейное программирование стало быстро развиваться после второй мировой войны; привлекая внимание математиков, экономистов и инженеров благодаря возможности широкого практического применения, а также математической стройности. Применение линейного программирования оказалось плодотворным в таких областях, как:

1. Оптимальное регулирование полетов на воздушных линиях.
2. Распределение во времени транспортировки грузов с заводов и складов к базам.
3. Размещение электронного оборудования на морских судах.
4. Планирование промышленного производства.
5. Система оплаты контрактов.
6. Проектирование систем связи и распределение в них потоков сообщений.
7. Распределение кадров.
8. Организация максимальных потоков в транспортных сетях.
9. Эффективное смешивание бензина.

Широкое привлечение внимания к линейному программированию привело к тому, что до некоторой степени была преувеличена его значимость; часто упускали из виду, что оно применимо лишь тогда, когда выполняются лежащие в его основе гипотезы, которые в свою очередь базируются на линейном математическом представлении реального мира. В нелинейном программировании не используются столь сильные упрощения.

Хотя задача линейного программирования может быть сформулирована по-разному, запишем ее в следующей форме [1, 2]:

$$\text{минимизировать } y = \sum_{i=1}^n c_i x_i \quad (2.1.3a)$$

при ограничениях

$$\sum_{i=1}^n a_{ij}x_i - b_j \geq 0, \quad j = 1, \dots, m, \quad (2.1.3b)$$

$$x_i \geq 0, \quad i = 1, \dots, n, \quad (2.1.3v)$$

где a , b и c — константы, а x — искомые переменные. Для решения задачи, выраженной уравнениями (2.1.3), были предложены различные методы, которые можно найти в списке литературы, помещенном в конце этой главы. Предполагается, что читатель знаком

с модифицированным симплексным методом решения. На фиг. 2.1.1 дано геометрическое представление граничных плоскостей для следующей типичной задачи линейного программирования:

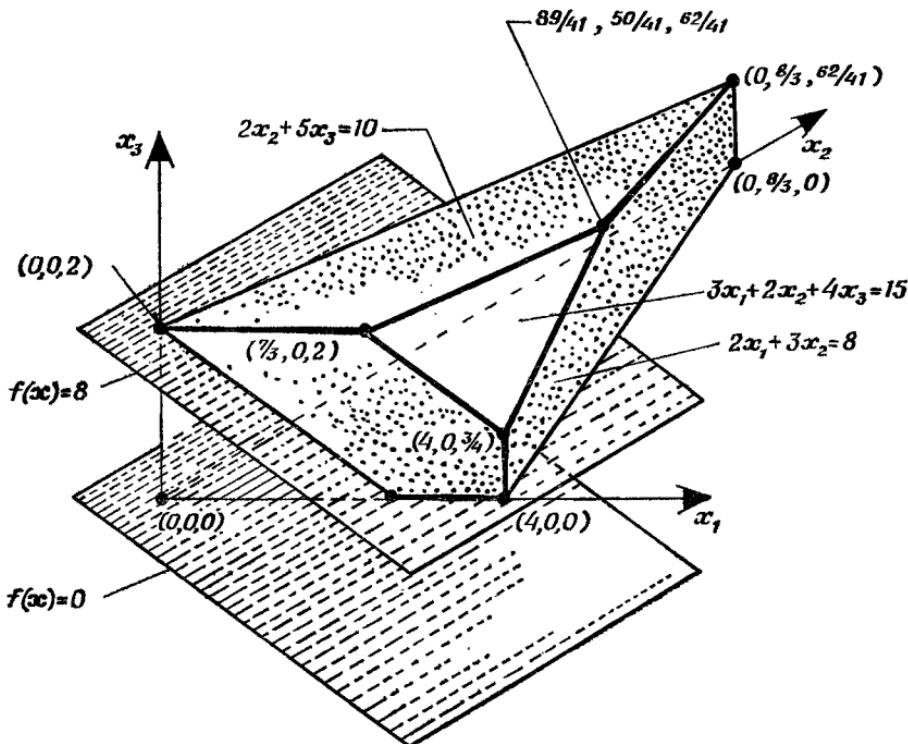
максимизировать $y = 3x_1 + 5x_2 + 4x_3$
при ограничениях

$$2x_1 + 3x_2 \leq 8,$$

$$2x_2 + 5x_3 \leq 10,$$

$$3x_1 + 2x_2 + 4x_3 \leq 15$$

для неотрицательных значений x_1 , x_2 и x_3 . Здесь целевая функция изображена двумя плоскостями, соответствующими постоянным



Фиг. 2.1.1. Геометрическое представление функций в задаче линейного программирования.

ее значениям. В обозначениях (2.1.3) эта задача может быть переформулирована следующим образом:

минимизировать $-y = -(3x_1 + 5x_2 + 4x_3)$
при ограничениях

$$-(2x_1 + 3x_2) + 8 \geq 0,$$

$$-(2x_2 + 5x_3) + 10 \geq 0,$$

$$\begin{aligned} & -(3x_1 + 2x_2 + 4x_3) + 15 \geq 0, \\ & x_1 \geq 0, x_2 \geq 0, x_3 \geq 0. \end{aligned}$$

Приведенный ниже пример показывает, как на основании имеющейся информации математически формулируется задача линейного программирования.

Пример 2.1.1. Формулировка задачи линейного программирования

Этот пример иллюстрирует связь между уравнениями (2.1.3) и реальной задачей. Фирма грузовых перевозок асигновала 600 000 долл. на приобретение грузовиков трех типов. Грузовик *A* стоит 10 000 долл., грузовик *B* — 20 000 долл. и грузовик *C* — 23 000 долл. Сколько грузовиков каждого типа нужно заказать, чтобы получить наибольшую производительность в тонно-милях на один день с учетом приведенных ниже условий?

Грузовик *A* требует для обслуживания одного водителя на каждую смену, максимальное число смен в сутки 3, производительность 2100 тонно-миль за каждую смену.

Грузовик *B* требует для обслуживания двух водителей на каждую смену, максимальное число смен в сутки 3, производительность 3600 тонно-миль за каждую смену.

Грузовик *C* требует для обслуживания двух водителей на каждую смену, максимальное число смен в сутки 3, производительность 3780 тонно-миль за каждую смену.

Кроме того, число водителей должно быть не больше 145, число грузовиков — не больше 30.

Эта задача может быть сформулирована в терминах линейного программирования следующим образом. Обозначим число смен индексами 1, 2 и 3, а число грузовиков каждого типа — прописными буквами *A*, *B* и *C*. Сначала необходимо выбрать критерий оптимальности. В данном примере этим критерием является производительность, соответствующая функции (2.1.3а):

$$\begin{aligned} y = & 2100A_1 + 4200A_2 + 6300A_3 + 3600B_1 + 7200B_2 + 10800B_3 + \\ & + 3780C_3 + 7560C_2 + 11340C_3, \end{aligned}$$

где *y* — количество тонно-миль. Запишем далее выражения для ограничений. Здесь мы имеем дело с ограничениями трех типов: 1) общая стоимость всех грузовиков не должна превышать 600 000 долл., 2) число грузовиков должно быть не более 30, 3) число водителей должно быть не более 145:

$$\begin{aligned} 1) \quad & 10000(A_1 + A_2 + A_3) + 20000(B_1 + B_2 + B_3) + 23000(C_1 + \\ & + C_2 + C_3) \leq 600000; \end{aligned}$$

$$2) A_1 + A_2 + A_3 + B_1 + B_2 + B_3 + C_1 + C_2 + C_3 \leq 30;$$

$$3) A_1 + 2A_2 + 3A_3 + 2B_1 + 4B_2 + 6B_3 + 2C_1 + 4C_2 + 6C_3 \leq 145.$$

Кроме того, поскольку грузовики являются физическими объектами и их число не может быть отрицательной величиной, имеют место следующие неравенства, соответствующие условиям (2.1.3в):

$$A_i \geq 0, B_i \geq 0, C_i \geq 0, \quad i = 1, 2, 3.$$

Поставленная так задача может быть решена соответствующим методом линейного программирования; в результате определяется количество грузовиков типов A , B и C , максимизирующее величину y .

Решение: 12 грузовиков типа A , 0 грузовиков типа B и 18 грузовиков типа C .

Матричные обозначения позволяют в компактной форме формулировать задачи математического программирования и описывать алгоритмы их решения. В приложении В приведены некоторые основные сведения о матрицах. Пусть \mathbf{x} и \mathbf{c} — векторы-столбцы размерности $n \times 1$ в E^n (т. е. в n -мерном евклидовом пространстве переменных), \mathbf{a} — матрица констант размерности $n \times m$ и \mathbf{b} — вектор-столбец размерности $m \times 1$:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ \vdots & \ddots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix}.$$

Тогда в матричных обозначениях уравнения (2.1.3) запишутся следующим образом:

$$\text{минимизировать } y \equiv f(\mathbf{x}) = \mathbf{c}^T \mathbf{x} \quad (2.1.4a)$$

при ограничениях

$$\mathbf{a}^T \mathbf{x} \geq \mathbf{b}, \quad (2.1.4b)$$

$$\mathbf{x} \geq 0, \quad (2.1.4v)$$

где верхний индекс T обозначает транспонирование. Вектор \mathbf{x}^* , удовлетворяющий соотношениям (2.1.4), представляет собой искомое решение задачи.

2.2. ОБЩАЯ ЗАДАЧА НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

В самом широком смысле общая задача нелинейного программирования заключается в отыскании экстремума целевой функции при заданных ограничениях в виде равенств и (или) неравенств. Ограничения могут быть линейными и (или) нелинейными. Однако общепринята несколько более узкая постановка общей задачи не-

линейного программирования, в которой исключаются из рассмотрения следующие специальные случаи:

1. Переменные принимают лишь целочисленные значения (нелинейное целочисленное программирование).

2. Ограничения включают как параметр время, при этом используются дифференциальные уравнения (оптимальное управление, динамическая оптимизация).

Пусть непрерывная функция $f(x)$ представляет собой целевую функцию, $h_1(x), \dots, h_m(x)$ задают ограничения в виде равенств, а $g_{m+1}(x), \dots, g_p(x)$ — ограничения в виде неравенств, где $x = [x_1, \dots, x_n]^T$ — вектор-столбец компонент x_1, \dots, x_n в n -мерном евклидовом пространстве.

Как и в линейном программировании, переменные x_1, \dots, x_n могут быть конструктивными параметрами, установками регулятора, показаниями измерительных приборов и т. д., тогда как целевая функция может представлять собой стоимость, вес, годовой доход и т. д., а ограничения — технические требования, условия работы, пропускную способность или факторы безопасности, присущие данному процессу.

Формально задача нелинейного программирования может быть сформулирована следующим образом:

$$\text{минимизировать } f(x), \quad x \in E^n, \quad (2.2.1)$$

при m линейных и (или) нелинейных ограничениях в виде равенств

$$h_j(x) = 0, \quad j = 1, \dots, m, \quad (2.2.2)$$

и $(p - m)$ линейных и (или) нелинейных ограничениях в виде неравенств

$$g_j(x) \geq 0, \quad j = m + 1, \dots, p. \quad (2.2.3)$$

Хотя в некоторых частных случаях ограничения в виде равенств могут быть явно разрешены относительно выбранных переменных, которые затем исключаются из задачи как независимые переменные, и в задаче остаются только ограничения в виде неравенств, чаще всего ограничения в виде равенств не могут быть явно разрешены и потому сохраняются.

Иногда встречается другое представление выражений (2.2.1) — (2.2.3):

$$\text{минимизировать } \{f(x) | x \in R\}, \quad (2.2.4)$$

где R — область пространства переменной x , для которой выполнены условия (2.2.2) и (2.2.3), например

$$R = \{x | h_j(x) = 0, \quad g_j(x) \geq 0 \text{ для всех } j\}. \quad (2.2.5)$$

Знак неравенства в выражении $g_j(x) \geq 0$ может быть изменен на обратный путем умножения на -1 , что не изменит математической постановки задачи.

В качестве простого примера задачи нелинейного программирования, которую можно проиллюстрировать графически, приведем следующую:

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + x_2^2 + 2x_2$$

при ограничениях

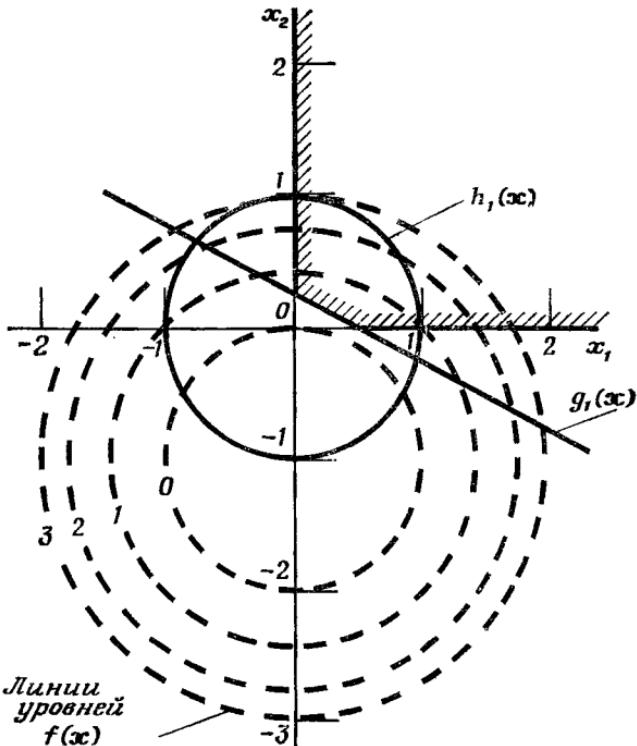
$$h_1(\mathbf{x}) = x_1^2 + x_2^2 - 1 = 0,$$

$$g_2(\mathbf{x}) = x_1 + 2x_2 - 0,5 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

На фиг. 2.2.1 целевая функция изображена пунктирными линиями, ограничение в виде равенства — жирной сплошной прямой, а гра-



Ф и г. 2.2.1. Геометрическое представление функций в задаче нелинейного программирования.

ница области, определяемой ограничениями в виде неравенств,— линиями со штриховкой (нанесенной с внутренней стороны области).

В каждой точке \$\mathbf{x}\$ \$n\$-мерного пространства переменных \$x_1, \dots, x_n\$ функция \$f(\mathbf{x})\$ имеет определенное значение, и, следовательно, это \$n\$-мерное пространство представляет собой скалярное поле

для критерия оптимальности. Как показано на фиг. 2.2.1, в этом пространстве можно вычертить семейство линий уровней (эквиденциональных гиперповерхностей) для выбранных значений функции $f(x)$. Пространство переменных x_1, \dots, x_n является также скалярным полем для функций ограничений, для которых также можно изобразить графически эквиденциональные гиперповерхности. При помощи классических методов анализа в общем случае невозможно заранее выявить расположение точки x^* , которая соответствует минимальному (или максимальному) значению функции $f(x)$, поскольку она может находиться на пересечении ограничивающих поверхностей или между ними.

Задачи линейного и квадратичного программирования могут рассматриваться как два частных случая общей задачи нелинейного программирования. Если и функция (2.2.1), и уравнения (2.2.2), и неравенства (2.2.3) линейны, то мы имеем дело с задачей линейного программирования. Если же целевая функция квадратична, а ограничения линейны, то имеет место задача квадратичного программирования:

$$\text{минимизировать } f(x) = a_0 + c^T x + x^T Q x \quad (2.2.6a)$$

при ограничениях

$$a^T x \geq b, \quad (2.2.6b)$$

$$x \geq 0, \quad (2.2.6c)$$

где Q — положительно определенная или нестрогательно определенная квадратичная симметрическая матрица, а a и b — матрицы коэффициентов, определение которых дано выше в связи с уравнением (2.1.4). В приложении В дано определение положительно определенной матрицы. Иногда в задачу квадратичного программирования включают линейные ограничения в виде равенств

$$a^T x = b. \quad (2.2.6d)$$

Примером задачи квадратичного программирования может служить следующая:

$$\text{минимизировать } f(x) = 0,5x_1^2 + 0,5x_2^2 - x_1 - 2x_2$$

при ограничениях

$$g_1(x) : 6 - 2x_1 - 3x_2 \geq 0,$$

$$g_2(x) : 5 - x_1 - 4x_2 \geq 0,$$

$$g_3(x) : x_1 \geq 0,$$

$$g_4(x) : x_2 \geq 0.$$

Во всех трех классах задач — нелинейных, линейных и квадратичных — нужно найти вектор $x^* = [x_1^*, \dots, x_n^*]^T$, минимизирующий

(или, наоборот, максимизирующий) функцию $f(x)$ при следующих условиях:

$$h_j(x) = 0, \quad j = 1, \dots, m, \text{ и } g_j(x) \geq 0, \quad j = m + 1, \dots, p.$$

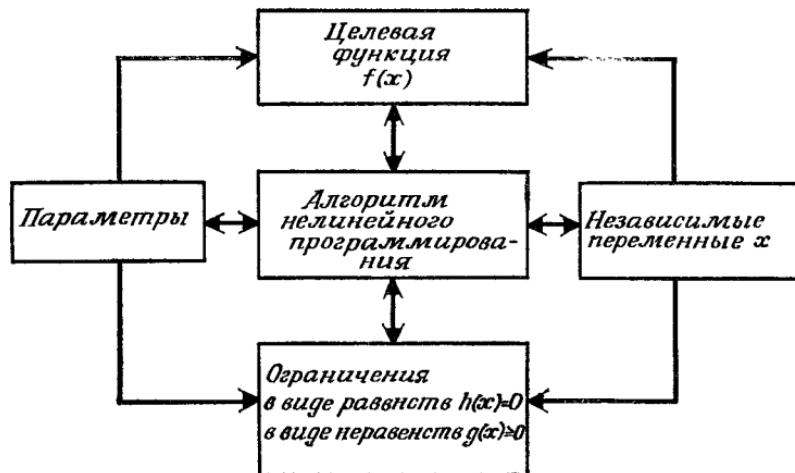
2.3. СВЯЗЬ ЗАДАЧИ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С РЕАЛЬНЫМ ПРОЦЕССОМ

В некоторых задачах оптимизации, таких, как задача минимизации по методу наименьших квадратов («минимизировать сумму квадратов отклонений наблюдаемого выходного сигнала от желаемого») или минимизация интеграла, появляющегося в вариационной задаче, постановку задачи нелинейного программирования можно легко связать с рассматриваемой физической задачей. Но в других случаях это не так. Рассмотрение только математической постановки задачи нелинейного программирования само по себе не позволяет выявить все факторы, от которых зависит оптимизация реального процесса. В этом разделе будет кратко рассмотрена взаимосвязь физической задачи и ее математического представления.

При оптимизации реального процесса параметры и (или) переменные связаны физическими законами, такими, как законы сохранения массы или энергии, которые должны быть включены в задачу нелинейного программирования как ограничения в виде равенств, даже если они только подразумеваются. Таким образом, одна группа ограничений состоит из функциональных связей, которые следует учитывать, чтобы оптимизация была физически осуществимой. Вторая группа ограничений включает заданные предельные диапазоны значений переменных или параметров, обеспечивающие их физическую реализуемость или совместимость с данным процессом; к этой группе относятся ограничения в виде неравенств. Заменять эти ограничения или добавляться к ним могут эмпирические связи, как правило, выражаемые равенствами. Наконец, для упрощения уравнений модели процесса часто вводят новые переменные, которые образуют дополнительные ограничения в виде равенств в том случае, когда исходные уравнения не могут быть разрешены в явном виде относительно определяемой переменной. На фиг. 2.3.1 показаны связи между различными частями задачи нелинейного программирования и алгоритмом решения.

Каждое ограничение в виде равенства уменьшает на единицу число степеней свободы в модели процесса и приводит к появлению еще одной зависимой переменной. Обычно предполагается, что уравнения модели процесса записываются достаточно аккуратно, так что все равенства являются независимыми; если же из-за невнимательности или ошибки в задачу включаются два избыточных, или, другими словами, зависимых, уравнения, то число степеней свободы будет отличаться от действительного. Число остаточных сте-

пеней свободы должно соответствовать числу *независимых переменных* (часто называемых *переменными решения*) в рассматриваемой задаче нелинейного программирования. Оно является важным понятием в задаче оптимизации любого типа при ограничениях в виде равенств, ибо если число переменных равняется числу независимых ограничений в виде равенств, то оптимизировать нечего — значения всех переменных можно непосредственно



Ф и г. 2.3.1. Соотношение между отдельными частями задачи нелинейного программирования и алгоритмом решения.

определить из совместного решения системы ограничений в виде равенств $h_j(x) = 0$, $j = 1, \dots, n$. Если число переменных превышает m — число независимых ограничений в виде равенств, то единственный способ решения, приемлемый для данной модели процесса, — это варьирование $(n - m)$ переменных решения, пока целевая функция не достигнет своего оптимального значения. Если число независимых ограничений в виде равенств превышает число переменных, то также возникает необходимость оптимизации, однако следует иметь в виду, что целевая функция в этом случае должна представлять собой некоторый статистический критерий, например такой, как в методе наименьших квадратов.

Поскольку в задачах нелинейного программирования, соответствующих некоторым физическим процессам, исключение зависимых переменных путем подстановки нелинейных функций может представлять некоторые трудности, одну или несколько зависимых переменных можно рассматривать как переменные решения наряду с действительно независимыми переменными. Например, в задаче

$$\text{минимизировать } f(x) = x_1^2 - 3x_2^2 + x_1x_2 + x_1x_3 + x_3 + 6$$

при ограничениях

$$\begin{aligned} h_1(x) &= (x_1 - x_2 + 2)^3 - (\sqrt{x_2} + x_3 + 2)^2 = 0, \\ h_2(x) &= x_1 + x_2 + x_3 + 3 = 0 \end{aligned}$$

весьма просто разрешить ограничение $h_2(x) = x_1 + x_2 + x_3 + 3 = 0$ относительно x_3 , исключить x_3 из $f(x)$ и $h_1(x)$ и устраниТЬ $h_2(x)$, тогда как разрешить $h_1(x)$ относительно x_1 или x_2 , чтобы исключить одну из этих переменных из целевой функции, гораздо сложнее. Обычно оказывается проще, особенно если задача содержит несколько нелинейных ограничений типа равенств, сохранить (как это имеет место в данной задаче) $h_1(x)$ как функцию-ограничение и в вычислительной процедуре рассматривать x как вектор, состоящий из двух переменных. Тем не менее на самом деле здесь имеется лишь одна остаточная степень свободы. Некоторые алгоритмы используют преимущества уменьшенного числа степеней свободы, учитывая ограничения в виде равенств и активные ограничения в виде неравенств путем поиска оптимума в пространстве меньшего числа измерений.

При исключении одной из переменных в целевой функции путем подстановки необходимо соблюдать осторожность, чтобы не изменить область определения переменных. Например, если $f(x) = -x_1^2 + x_2$ и $h(x) = -x_1^2 - x_2^2 + 1 = 0$, область изменения x_2 должна быть ограничена пределами $-1 \leq x_2 \leq 1$, поскольку меньшие или большие значения x_2 дают мнимые значения x_1 . Исключение x_1 путем подстановки $x_1^2 = 1 - x_2^2$ приводит к $f(x_2) = f(x) = x_2 + x_2^2 - 1$ и кажется, что величина x_2 может быть неограниченной, тогда как в действительности, чтобы задача не изменилась, к функции $f(x_2)$ нужно добавить ограничение $-1 \leq x_2 \leq 1$.

В качестве иллюстрации понятия остаточных степеней свободы для $n > m$ рассмотрим задачу нахождения объема и линейных размеров наибольшего ящика, у которого сумма длины и поперечных размеров не превышает 1,8 м. (Ответ: $V = 0,6 \times 0,3 \times 0,3 = 0,05 \text{ м}^3$.) Эту задачу можно сформулировать как задачу максимизации, в которой критерием качества является объем ящика:

Максимизировать $f(x) = x_1 x_2 x_3$, $x \in E^3$, при указанных выше условиях на линейные размеры; кроме того, имеются неявные условия, требующие, чтобы размеры были неотрицательными:

$$\begin{aligned} g_1(x) : 1,8 - x_1 - 2x_2 - 2x_3 &\geq 0, \\ g_{i+1}(x) : x_j &\geq 0, \quad j = 1, \dots, 3. \end{aligned}$$

Если исследователь забудет включить в задачу эти неявные ограничения, то он может получить правильное математическое решение, не имеющее никакого отношения к действительности.

Поскольку задача не включает ограничений в виде равенств, она имеет три остаточные степени свободы; это значит, что все три переменные можно изменять независимо при условии, что не нарушаются ограничения в виде неравенств. Если ввести в задачу ограничение в виде равенства, например требование о том, чтобы высота и ширина ящика были равны, т. е. $h_1(x) = x_2 - x_3 = 0$, то задача будет иметь лишь две остаточные степени свободы. Очевидно, что в этом простом примере одну переменную x_2 или x_3 можно исключить из целевой функции и задача сведется к максимизации объема путем изменения лишь двух независимых переменных. Однако в более реальных задачах исключение одной переменной из целевой функции может оказаться неосуществимым. Вопрос о том, какие конкретные переменные взять в качестве независимых, а какие — в качестве зависимых, решается в некотором отношении произвольно, но исследователь должен всегда иметь в виду определенные естественные оптимизирующие переменные, связанные с данной моделью процесса, которые либо легко контролируются, либо являются удобными в математическом отношении.

В приведенном ниже примере 2.3.1 формулируются отдельные части задачи нелинейного программирования для иллюстрации того, как в подобных задачах реальное явление связано с его математическим представлением. Заметим, что переменные являются тем механизмом, при помощи которого осуществляется передача информации между целевой функцией и заданными ограничениями. В целевую функцию могут входить как зависимые, так и оптимизирующие переменные.

Пример 2.3.1. Моделирование и оптимизация работы доменной печи

Плавка в доменной печи является распространенной и важной технологической операцией на любом сталелитейном заводе. В некоторых отношениях она может рассматриваться как типичный промышленный процесс. Имеется довольно большое число существенных переменных этого процесса (некоторые из них не могут быть измерены), взаимодействующих весьма сложным образом; следует учитывать большое число ограничений; условия работы завода часто таковы, что оптимальная рабочая точка значительно изменяется со временем. Таким образом, подробное исследование задачи оптимизации работы доменной печи оказывается поучительным, обнаруживая характер и степень трудностей, встречающихся при математическом представлении типичного промышленного процесса.

Таблица П.2.3.1

Связь между процессом и его математическим представлением

Процесс	Анализ процесса	Математическое представление
Связь между процессом и его математическим представлением	<p>Передача внешней информации и информации о процессе при помощи переменных и параметров</p> <p>Связь между целевой функцией и моделью процесса</p> <p>Входные и выходные переменные</p> <p>Сопутствующие цены и доходы</p> <p>Руда 1: x_1 Руда 2: x_2 и т. д. Загрузка железного металломолома: x_6 Кокс А: x_7 Кокс В: x_8 Чугун в чушкиах: F Газ: G</p>	<p>Оптимизирующие переменные, зависимые переменные и коэффициенты</p> <p>Целевая функция $f(x)$</p> <p>Минимизировать</p> $f(x) = c_1x_1 + c_2x_2 + \dots + c_7x_7 + c_8x_8 + \dots + c_{11}\left(\frac{P}{S}\right) + \dots + c_{12}\left(\frac{Si + Al}{Ca + Mg}\right) + \dots - c_F(F + F^{1,2}) + \dots + c_GG + \dots$ <p>Модель процесса</p>

Материальные и энергетические балансы

Баланс металла
Баланс шлака

Ограничения в виде равенств

$$h_j(x) = 0, \quad j = 1, \dots, m,$$

$$a_1x_1 + a_2x_2 + \dots +$$

$$+ a_3 \frac{\text{Ca} + \text{Mg}}{\text{Al}} + \dots = 0$$

Газовый баланс
Энергетический баланс

$$[a_{10}C + a_{11}\left(H - \frac{O}{8}\right) + a_{12}S] \times$$

$$\times (x_7 + x_8) T_1 +$$

$$+ a_{13}FT_2 + \dots = 0$$

Балансы элементов (O, H, S, Si, и т. д.)

и т. д.

Ограничения процесса
Количество кокса

Ограничения в виде неравенств

$$g_j(x) \leq 0,$$

$$j = m + 1, \dots, p,$$

$$x_7 \leq a_{30},$$

$$[x_2/(x_1 + x_2 + \dots)] \leq 0,2,$$

Скорость производства жидкого металла

Пригодность руды
Элементы в шлаке

Элементы в металле

Основность

Ограничения в реализации

$$a_{42} \leq (\text{Ca} + \text{Mg}) / (\text{Si} + \text{Al})$$

и т. д.

Описание процесса

Доменная печь работает полунепрерывно. Сырьем являются железная руда, содержащая приблизительно 20—60% железа в окислах, разнообразные окислы других металлов и неметаллов, а также кокс, образующий при горении доменный газ. Кроме газа, который может служить средством подогрева для других процессов, выход печи состоит из расплавленного железа, все еще содержащего некоторые примеси (в заметных количествах углерод и фосфор), которые должны быть удалены в процессе изготовления стали, и шлака, содержащего большинство примесей, который не представляет ценности. Стоимости железной руды и коксового топлива приблизительно одинаковы. Требуется выбирать руду, скорость производства и тип процесса, которые бы максимизировали «доход» от производства или минимизировали стоимость изготовления требуемого количества расплавленного металла нужного качества. В табл. П.2.3.1 схематически изображен поток материалов в доменной печи, являющейся частью большого завода.

Улучшение качества этого (или любого другого) процесса с использованием вычислительных, а не экспериментальных методов может быть осуществлено следующими этапами:

1. Анализ процесса, при котором определяются важные переменные и специфические характеристики задачи.

2. Определение целевой функции и критерия качества (прибыль, эксплуатационные расходы, объем и т. д.) и выражение критерия в виде *математической функции*.

3. Разработка математической модели процесса, которая связывает переменные входа и выхода и перечисляет ограничения.

4. Применение процедуры оптимизации к математической постановке задачи.

Эти этапы схематически представлены в табл. П.2.3.1. Поясним некоторые понятия, лежащие в основе представленных здесь уравнений.

Целевая функция

При формулировке критерия в виде функции стоимости следует рассмотреть две категории затрат:

1. Затраты, связанные с движением материалов (входные и выходные переменные), как, например, затраты на приобретение материалов за вычетом дохода от продажи побочных продуктов.

2. Затраты на осуществление технологических процессов, которые связываются с переменными процесса путем анализа или с помощью эмпирических кривых.

В табл. П.2.3.1 приведена типичная целевая функция.

Математическая модель

Следующим шагом в формулировании задачи является построение математической модели процесса путем рассмотрения основных химических и физических явлений, лежащих в его основе. В случае доменной печи типичными являются следующие обстоятельства:

1. *Железная руда.* В ограниченных количествах пригодны руды различных градаций. Разные руды имеют различный процент железа и различные виды и количество примесей. Предполагается, что доля руды каждого типа, переходящей в расплавленный металл, имеет определенную величину.

2. *Кокс.* Количество сжигаемого кокса в любой печи существенно ограничено ее конструкцией. Скорость расхода кокса может определяться на основе эмпирических соотношений, разработанных с помощью множественной регрессии.

3. *Качество руды.* Должно иметь место ограничение на количество используемых «твёрдых» руд. Аналогично должно быть наложено ограничение на количество очищенной руды, избыток которой нарушает поток газа в печи и ограничивает выход продукции.

4. *Фосфор.* Весь фосфор из сырья поступает в расплавленный металл. Устанавливается верхний предел допустимого содержания фосфора, хотя иногда и предписываются точные количества. Обычно дешевле производить железо с большим содержанием фосфора, однако гораздо дороже затем его очищать.

5. *Другие элементы.* Окислы кальция, кремния, магния и алюминия выходят в шлаки. Две трети марганца поступает в расплавленный металл, и устанавливается ограничение на допустимое количество этой примеси. Количество серы в загрузке не должно превышать 1,6% общего веса шлака; в противном случае слишком много серы попадает в расплавленный металл.

6. *Шлак.* По техническим причинам уровень примесей в шлаке должен контролироваться. При этом устанавливаются верхний предел на процентное содержание магния, верхний и нижний пределы на процентное содержание кремния и алюминия и узкие пределы на отношение основностей ($\text{CaO} + \text{MgO} / \text{SiO}_2 + \text{Al}_2\text{O}_3$).

Исходя из этих и других факторов оказывается возможным получить:

1. Набор входных и выходных переменных,

2. Уравнения установившихся материальных и энергетических балансов между входом и выходом.

3. Набор ограничений на входные и выходные переменные для условий производства и рынка.

Типичные взаимосвязи показаны в табл. П.2.3.1.

Как и следовало ожидать, основными недостатками формулирования этой задачи нелинейного программирования являются незнание истинных значений параметров в уравнениях мате-

риального и энергетического баланса; предположение об установившихся условиях, тогда как в действительности в процессе имеют место изменения; стохастическая, а не детерминистическая природа переменных и параметров и т. д. Тем не менее это скорее недостатки анализа на стадии построения модели, чем на стадии оптимизации, что является основным предметом этой книги.

2.4. ОБОЗНАЧЕНИЯ И ТЕРМИНОЛОГИЯ

Чтобы выделить определенные аспекты используемой в книге терминологии, в этом разделе сделаны некоторые предварительные замечания относительно ряда терминов, довольно часто встречающихся в литературе по оптимизации.

2.4.1. ОПТИМАЛЬНЫЕ РЕШЕНИЯ

Вектор $\mathbf{x}^* = [x_1^*, \dots, x_n^*]^T$, удовлетворяющий соотношениям (2.2.1) — (2.2.3), называется *оптимальной точкой*, а соответствующее значение $f(\mathbf{x}^*)$ — оптимальным значением целевой функции. Пара \mathbf{x}^* и $f(\mathbf{x}^*)$ составляет *оптимальное решение*. Как показано на примере мультимодальной функции на фиг. 2.4.1, могут существовать различные типы оптимальных решений, если целевая функция не является *унимодальной* (т. е. имеющей один экстремум). Глобальное оптимальное решение представляет собой наименьшее значение $f(\mathbf{x})$, тогда как *локальное* (или относительное) оптимальное решение представляет собой наименьшее значение $f(\mathbf{x})$ в окрестности некоторого вектора \mathbf{x} . Как для глобального, так и для локального минимума

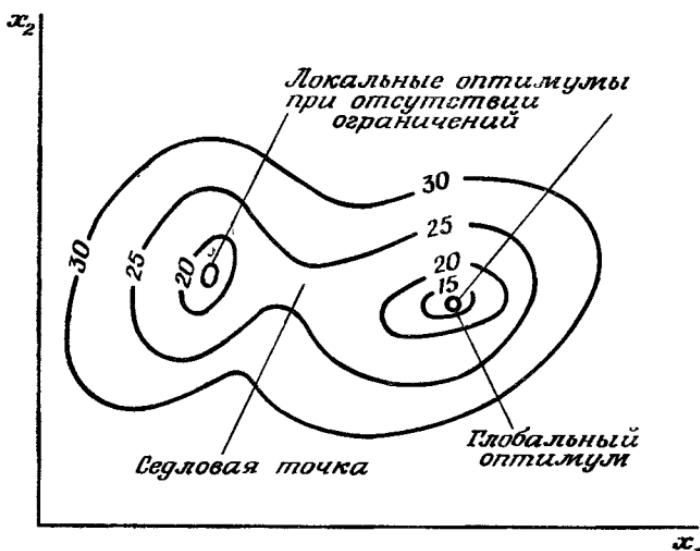
$$f(\mathbf{x}^*) \leq f(\mathbf{x}),$$

но для глобального оптимального решения это соотношение выполняется для всех $\mathbf{x} \in E^n$, тогда как для локального оптимального решения это имеет место только для малой области ζ , где $\|\mathbf{x} - \mathbf{x}^*\| < \zeta$. Если принимается во внимание и точность решения, то условие оптимальности можно представить в виде

$$f(\mathbf{x}^*) \leq f(\mathbf{x}) - \gamma,$$

где γ — некоторая малая величина.

Все алгоритмы, описываемые в последующих главах, дают лишь локально оптимальные решения, так как на каждом этапе решения при движении к \mathbf{x}^* они зависят в основном от локальных свойств целевой функции и ограничений. Правда, некоторые алгоритмы, такие, как алгоритм, описываемый в гл. 8, вероятнее всего, закон-



Ф и г. 2.4.1. Классификация оптимальных решений.

чаться в точке глобального оптимума, так как здесь при поиске рассматривается широкий диапазон изменения x , однако сходимость к глобальному оптимуму не может быть гарантирована без дополнительных сведений относительно природы целевой функции и ограничений. На практике предположение о том, что локальный экстремум является глобальным, может быть проверено путем использования нескольких начальных векторов, но даже если найдено только одно локальное решение, в общем случае нельзя показать, что это решение обязательно является глобальным оптимумом. К счастью, для задач, соответствующих действительным физическим процессам, целевая функция обычно является хорошей и обладает единственным экстремумом. Поэтому для большинства практических целей использование численных процедур, дающих локальное решение задачи программирования, не является большим недостатком.

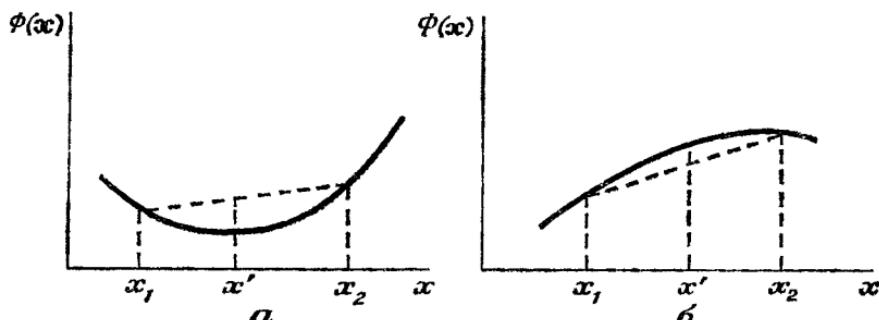
2.4.2. Вогнутость и выпуклость

Понятия вогнутости и выпуклости полезны при определении того, при каких условиях локальное оптимальное решение является также глобальным оптимальным решением, что представляется важным в случае множественных оптимумов.

Функция $\phi(x)$ называется выпуклой в области R , если для любых двух векторов x_1 и $x_2 \in R$

$$\phi(\theta \mathbf{x}_1 + (1-\theta) \mathbf{x}_2) \leq \theta \phi(\mathbf{x}_1) + (1-\theta) \phi(\mathbf{x}_2), \quad (2.4.1)$$

где 0 — скаляр, изменяющийся в диапазоне $0 \leq \theta \leq 1$. Функция $\phi(\mathbf{x})$ является *строго выпуклой*, если для $\mathbf{x}_1 \neq \mathbf{x}_2$ в выражении (2.4.1) используется знак неравенства ($<$). (Векторное неравенство $\mathbf{x} \geq \mathbf{y}$ означает, что $x_i \geq y_i$ для каждого элемента; в случае $\mathbf{x} > \mathbf{y}$ справедливо неравенство $x_i > y_i$ для всех i .) Существуют и другие



Фиг. 2.4.2. Выпуклые (а) и вогнутые (б) функции (в данной области изменения x).

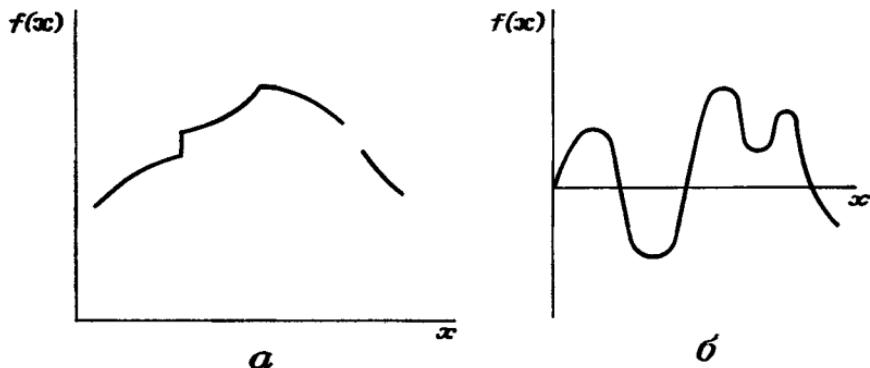
определения выпуклости, предложенные Понстейном [3]. На фиг. 2.4.2 геометрически изображена строго выпуклая функция одной независимой переменной; выпуклая функция не может принимать значения, большего чем значения функции, полученной линейной интерполяцией между $\phi(\mathbf{x}_1)$ и $\phi(\mathbf{x}_2)$. Если имеет место неравенство, обратное (2.4.1), то функция называется *вогнутой*. Таким образом, функция $\phi(\mathbf{x})$ *вогнутая* (строго вогнутая), если $-\phi(\mathbf{x})$ выпуклая (строго выпуклая). Линейные функции одновременно и вогнутые и выпуклые. Дифференцируемая выпуклая функция обладает следующими свойствами:

- 1) $\phi(\mathbf{x}_2) - \phi(\mathbf{x}_1) \geq \nabla^T \phi(\mathbf{x}_1) (\mathbf{x}_2 - \mathbf{x}_1)$ для всех $\mathbf{x}_1, \mathbf{x}_2$;
- 2) матрица вторых частных производных $\phi(\mathbf{x})$ по \mathbf{x} (матрица Гесссе) положительно определенная (или положительно полуопределенная) для всех \mathbf{x} , если $\phi(\mathbf{x})$ строго выпуклая (или просто выпуклая);
- 3) в области R функция $\phi(\mathbf{x})$ имеет только один экстремум.

(2.4.2)

Требование унимодальности функции является значительно более слабым, чем требование вогнутости или выпуклости, поскольку, как видно из фиг. 2.4.3, а, унимодальность не требует ни непрерывности, ни единственности производной.

Множество точек (или область) называется *выпуклым* в n -мерном пространстве, если для всех пар точек \mathbf{x}_1 и \mathbf{x}_2 , принадлежащих этому множеству, отрезок прямой линии, соединяющей их, также



Ф и г. 2.4.3. Унимодальная (a) и мультимодальная (б) функции.

полностью принадлежит множеству. Так, каждая точка x , определяемая выражением

$$x = \theta x_1 + (1 - \theta) x_2, \quad 0 \leq \theta \leq 1,$$

также принадлежит множеству. На фиг. 2.4.4 изображены выпуклое и невыпуклое множества в двумерном пространстве. Например, следующая группа выражений определяет выпуклую область (возможно, пустую или неограниченную):

$$g_j(x) \leq b_j,$$

$$a_j^T x = b_j,$$

если все $g_j(x)$ выпуклы (фиг. 2.4.5).

Заметим, что из понятия выпуклости вытекает один важный результат математического программирования [4]. Задача нелинейного программирования, представленная в виде задачи выпуклого программирования, формулируется следующим образом¹⁾:

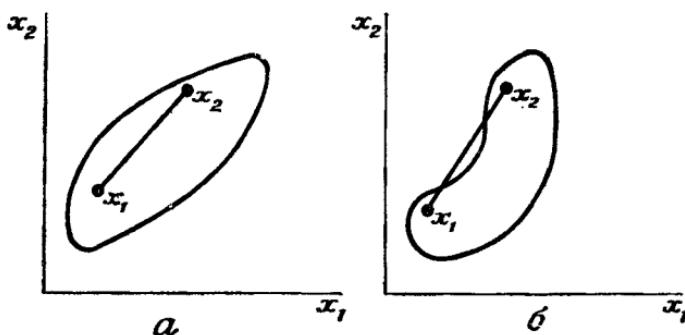
минимизировать $f(x)$

при ограничениях

$$\begin{aligned} g_j(x) &\geq 0, \quad j = 1, \dots, p, \\ x &\geq 0. \end{aligned}$$

где $f(x)$ — выпуклая функция и каждая функция, задающая ограничение в виде неравенства, — вогнутая функция (ограничения образуют выпуклое множество). При этом можно показать справедливость следующего положения: локальный минимум является также и глобальным минимумом. Аналогично локальный максимум

¹⁾ Иногда в задаче выпуклого программирования ограничения в виде равенств $h_j(x) = 0$ добавляются к ограничениям в виде неравенств; и в этом случае $h_j(x)$ должны быть линейны.



Ф и г. 2.4.4. Выпуклое (a) и невыпуклое (б) множества.

является глобальным максимумом, если целевая функция вогнутая, а ограничения образуют выпуклое множество.

Рассмотрим следующую задачу (фиг. 2.4.5):

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + x_2^2 - 4x_1 + 4$$

при ограничениях

$$g_1(\mathbf{x}) = x_1 - x_2 + 2 \geq 0,$$

$$g_2(\mathbf{x}) = -x_1^2 + x_2 - 1 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

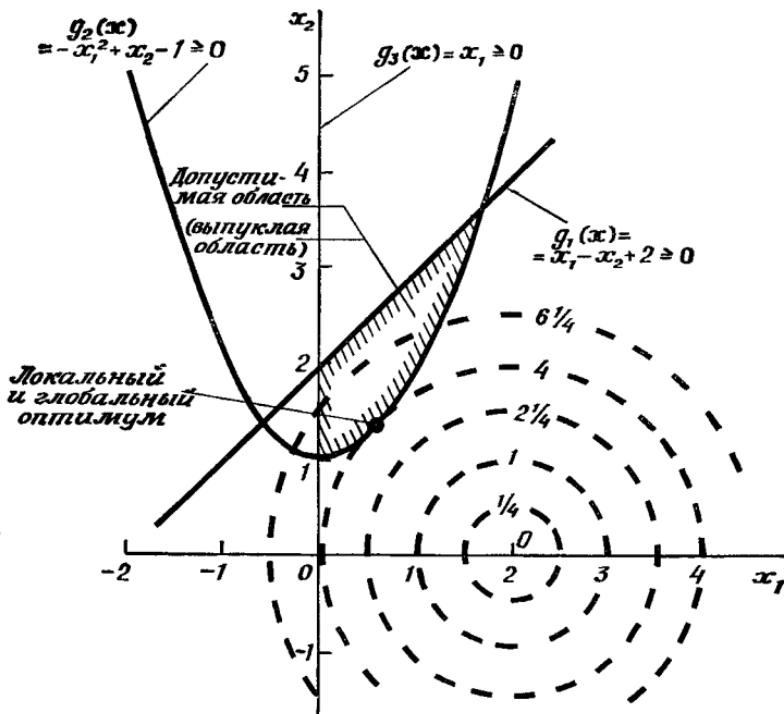
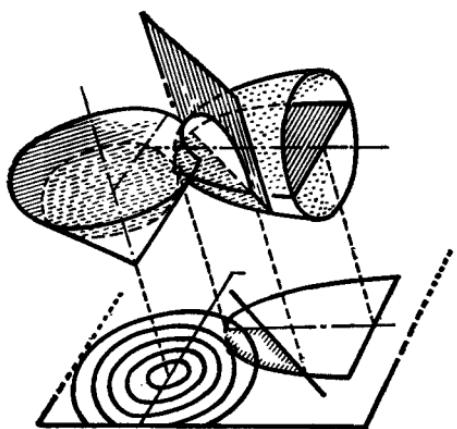
$$g_4(\mathbf{x}) = x_2 \geq 0.$$

Внимательное изучение кривых (к тому же выводу можно прийти и аналитически) показывает, что ограничения образуют выпуклую область (допустимая часть области заштрихована), поскольку функции $g_1(\mathbf{x})$ и $g_3(\mathbf{x})$ (и $g_4(\mathbf{x})$) линейные и, следовательно, вогнутые (правда, они одновременно и выпуклые), а функция $g_2(\mathbf{x})$ строго вогнутая. Можно показать, что функция $g_2(\mathbf{x})$ вогнутая, заметив, что матрица Гессе функции — $g_2(\mathbf{x})$ положительно полуопределенная:

$$-\begin{bmatrix} \frac{\partial^2 g_2(\mathbf{x})}{\partial x_1^2} & \frac{\partial^2 g_2(\mathbf{x})}{\partial x_1 \partial x_2} \\ \frac{\partial^2 g_2(\mathbf{x})}{\partial x_2 \partial x_1} & \frac{\partial^2 g_2(\mathbf{x})}{\partial x_2^2} \end{bmatrix} = -\begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix} \geq 0.$$

Целевая функция $f(\mathbf{x})$ строго выпуклая, и локальный оптимум, являющийся также и глобальным оптимумом, достигается в точке $\mathbf{x}^* = [0,58 \ 1,34]^T$, где $f(\mathbf{x}) = 3,80$.

В задаче линейного программирования, имеющей оптимальное решение, целевая функция всегда выпуклая, а ограничения обра-



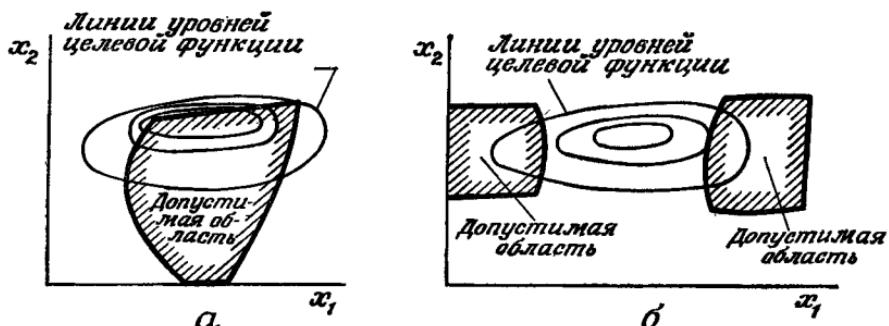
Фиг. 2.4.5. Задача выпуклого программирования, иллюстрирующая допустимую область (представляющую собой выпуклое множество) и глобальный оптимум.

зуют выпуклое множество, так что локальный оптимум всегда является в то же время и глобальным оптимумом. Ограничения в задаче квадратичного программирования те же, что и в задаче линейного программирования, а целевая функция выпуклая, если $x^T Qx$ является положительно полуопределенной; следовательно, матрица Q должна быть неотрицательно определенной. Однако только при особых обстоятельствах можно показать, что общая нелинейная функция $f(x)$ является выпуклой, а ограничения образуют выпуклое множество. Так, одной из самых серьезных трудностей является то, что нелинейные *равенства* не могут быть частью выпуклой области, содержащей больше чем одну точку, поскольку прямая линия, соединяющая две любые несмежные точки, удовлетворяющие данному равенству, не может в то же время проходить и через другие точки, удовлетворяющие этому равенству, как это требуется для выпуклости. Тем не менее в особом случае, когда ограничивающее множество образовано лишь ограничениями в виде неравенств и все функции-ограничения являются вогнутыми, так что точки, для которых $g_i(x) \geq 0$, образуют выпуклое множество, задача нелинейного программирования может представлять собой задачу выпуклого программирования.

2.4.3. ДОПУСТИМОСТЬ

Любой вектор x , удовлетворяющий ограничениям в виде равенств и в виде неравенств, называется *допустимой точкой* или *допустимым* вектором. Множество всех точек, удовлетворяющих ограничениям, образует допустимую область $f(x)$, которую будем обозначать через R ; любая точка вне R называется *недопустимой*. *Условный оптимум* представляет собой локальный оптимум, лежащий на границе допустимой области. Если ограничения имеют вид равенств, то допустимый вектор x должен лежать на пересечении всех гиперповерхностей, соответствующих $h_i(x) = 0$. При ограничениях в виде неравенств точка x может быть либо *внутренней точкой* (допустимой точкой), либо *границей точкой* (тоже допустимой точкой), либо *внешней точкой* (недопустимой точкой). Для внутренней точки $g_i(x) > 0$; в случае граничной точки удовлетворяется по крайней мере одно равенство $g_i(x) = 0$, а для внешней точки имеет место по крайней мере одно неравенство $g_i(x) < 0$. Множество точек, удовлетворяющих равенствам $g_i(x), j = 1, \dots, p$, определяет *границы поверхности* системы ограничений, заданных в виде неравенств. Активным, или связывающим, ограничивающим неравенством называется такое, которое для данного x превращается в равенство $g_i(x) = 0$. Область допустимых значений переменных может быть односвязной (фиг. 2.4.6, *a*) и неодносвязной (фиг. 2.4.6, *b*). В последнем случае может оказаться, что

данный алгоритм нелинейного программирования допустит обследование одной или нескольких допустимых областей, если процедура поиска не будет включать большое число начальных векторов. К счастью, большинство задач нелинейного программирования,



Фиг. 2.4.6. Примеры односвязной (a) и неодносвязной (б) областей допустимости (относящихся к ограничениям в виде неравенств).

относящихся к реальным процессам, формулируется так, что допустимая область является односвязной. В разд. 6.2 излагается наиболее подходящий способ преобразования недопустимой точки в допустимую (внутреннюю) точку.

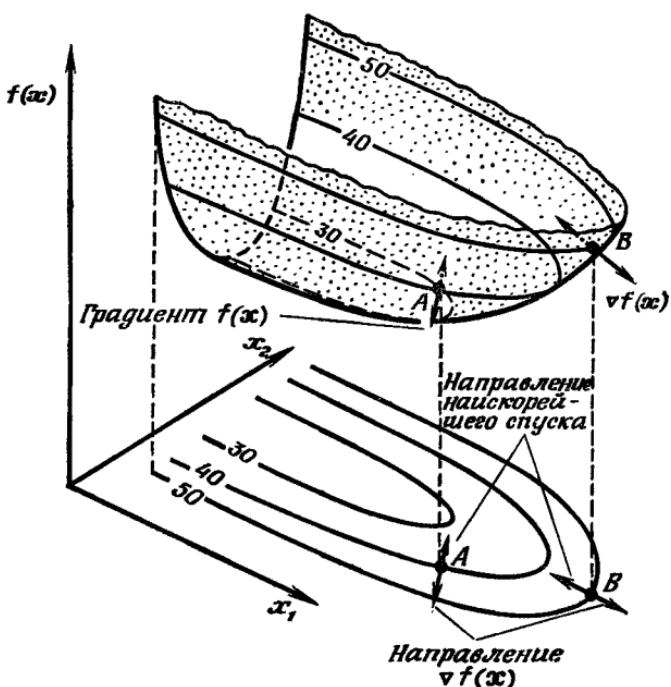
2.4.4. ГРАДИЕНТ

Множество точек, для которых целевая функция имеет постоянное значение, называется *линией уровня* $f(x)$. Несколько таких линий уровней изображено на фиг. 2.4.7. Если целевая функция непрерывна и дифференцируема, то существует *градиент* $f(x)$, определяемый как *вектор-столбец* из первых частных производных $f(x)$ по x , значения которых берутся в данной точке x . Верхний индекс k , $k = 0, 1, \dots$, используется для обозначения точки в E^n , в которой берется значение градиента, и, таким образом, градиент в $x^{(k)}$ равен

$$\nabla f(x^{(k)}) = \begin{bmatrix} \frac{\partial f(x^{(k)})}{\partial x_1} \\ \vdots \\ \frac{\partial f(x^{(k)})}{\partial x_n} \end{bmatrix}. \quad (2.4.3)$$

Выражение $\nabla^T f(x^{(k)})$ обозначает вектор-строку. В литературе, посвященной векторам и матрицам, доказывается, что гра-

диент скалярной функции направлен в сторону наискорейшего увеличения функции, т. е. *наискорейшего подъема*, и что он ортогонален линии уровня $f(x)$, проходящей через данную точку $x^{(k)}$. Вектор, противоположный этому градиенту (отрицательный градиент), направлен в сторону *наискорейшего спуска*. Любой вектор



Ф и г. 2.4.7. Градиент (направление наискорейшего подъема) и направление наискорейшего спуска в двух точках.

∇ , ортогональный $\nabla f(x^{(k)})$ [так же, как и касательная плоскость к $f(x^{(k)})$ в точке $x^{(k)}$], может быть записан как $\nabla^T \nabla f(x^{(k)}) = 0$.

Следует отметить, что градиент не является направлением наискорейшего увеличения $f(x)$, если рассмотрение ведется не в евклидовой, а в какой-то другой метрике. Например, если определять длину вектора x не по формуле $\|x\| = (x^T x)^{1/2}$, а по формуле $\|x\| = \sum_i \|x_i\|$, то направление наискорейшего увеличения $f(x^{(k)})$

можно получить, находя ту компоненту $\nabla f(x^{(k)})$, которая имеет наибольшее абсолютное значение, и полагая соответствующую компоненту x равной либо $+1$, либо -1 (в зависимости от знака компоненты градиента), а остальные компоненты приравнивая к нулю, как, например, в симплекс-методе при линейном программировании,

2.4.5. АППРОКСИМАЦИЯ ФУНКЦИЙ

Для некоторых процедур математического программирования, которые будут рассмотрены позднее, необходимо осуществлять линейную или квадратичную аппроксимацию функций $f(\mathbf{x})$, $g_i(\mathbf{x})$ и $h_j(\mathbf{x})$. Например, линейная, или первого порядка, аппроксимация целевой функции $f(\mathbf{x})$ может быть выполнена с помощью усеченного ряда Тейлора в окрестности $\mathbf{x}^{(k)}$:

$$f(\mathbf{x}) \approx f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}). \quad (2.4.4)$$

Квадратичную аппроксимацию $f(\mathbf{x})$ можно получить, отбрасывая в рядах Тейлора члены третьего и более высокого порядков:

$$\begin{aligned} f(\mathbf{x}) \approx & f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) + \\ & + \frac{1}{2} (\mathbf{x} - \mathbf{x}^{(k)})^T \nabla^2 f(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}), \end{aligned} \quad (2.4.5)$$

где $\nabla^2 f(\mathbf{x}^{(k)})$ — матрица Гессе $f(\mathbf{x})$, $\mathbf{H}(\mathbf{x})$, представляющая собой квадратную матрицу вторых частных производных $f(\mathbf{x})$, взятых в точке $\mathbf{x}^{(k)}$:

$$\nabla^2 f(\mathbf{x}^{(k)}) = \mathbf{H}(\mathbf{x}^{(k)}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x}^{(k)})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x}^{(k)})}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots & \ddots & \ddots \\ \frac{\partial^2 f(\mathbf{x}^{(k)})}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(\mathbf{x}^{(k)})}{\partial x_n^2} \end{bmatrix}. \quad (2.4.6)$$

2.5. НЕОБХОДИМЫЕ И ДОСТАТОЧНЫЕ УСЛОВИЯ ОПТИМАЛЬНОСТИ РЕШЕНИЯ

В области нелинейного программирования большое внимание было уделено определению необходимых и достаточных условий того, чтобы некоторый вектор решения \mathbf{x} являлся локальным экстремумом. Критерий оптимальности решения для некоторых особых случаев нелинейного программирования, описанных формулами (2.2.1) — (2.2.3), был сформулирован выше. Однако структура задачи нелинейного программирования в общем случае такова, что полностью критерий оптимальности еще не разработан. Вследствие этого здесь будут рассмотрены только некоторые особые случаи, но они достаточно важны и часто встречаются на практике. Условия, при которых некоторый вектор \mathbf{x} является решением задачи нелинейного программирования, будут сформулированы в виде теорем без доказательств (поскольку это выходит за рамки данной книги), а для читателей, желающих познакомиться с этим вопросом подробнее, приведены необходимые ссылки на литературу.

2.5.1. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ БЕЗ ОГРАНИЧЕНИЙ

Постановка задачи:

$$\text{минимизировать } f(x), \quad x \in E^n. \quad (2.5.1)$$

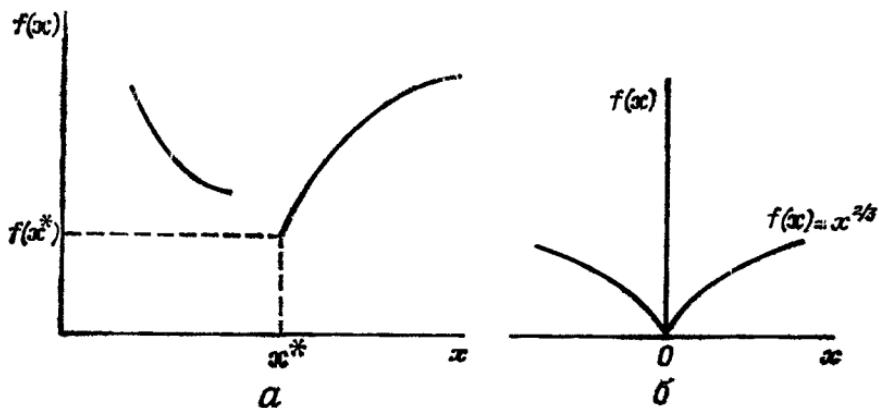
Для задачи нелинейного программирования при отсутствии ограничений *необходимыми условиями* того, что x^* — точка локального минимума задачи (2.5.1), являются:

1) функция $f(x)$ дифференцируема в точке x^* ;

2) $\nabla f(x^*) = 0$, т. е. существует стационарная точка в x^* .

Достаточные условия того, что x^* — точка локального минимума задачи (2.5.1), кроме приведенных выше условий 1 и 2, включают следующее:

3) $\nabla^2 f(x^*) > 0$, т. е. матрица Гессе положительно определенная. (Соответствующие условия для максимума те же самые, за исключение



Ф и г. 2.5.1. Геометрическое представление функций в задаче нелинейного программирования.

нием того, что матрица Гессе для $f(x^*)$ должна быть отрицательно определенной.)

На фиг. 2.5.1 показано, что возможно существование минимума, который не удовлетворяет необходимым и достаточным условиям. Однако, если достаточные условия удовлетворены, x^* обязательно будет точкой минимума.

2.5.2. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ С ОГРАНИЧЕНИЯМИ В ВИДЕ РАВЕНСТВ И НЕРАВЕНСТВ

Постановка задачи:

$$\text{минимизировать } f(x), \quad x \in E^n,$$

при ограничениях

$$\begin{aligned} h_j(x) &= 0, \quad j = 1, \dots, m, \\ g_j(x) &\geq 0, \quad j = m + 1, \dots, p. \end{aligned} \quad (2.5.2)$$

Задача (2.5.2) идентична (2.2.1) — (2.2.3). Необходимые условия того, что x^* является точкой локального минимума, могут быть сформулированы в двух теоремах, первую из которых (теорема 2) можно назвать условиями первого порядка (потому что входящие в них функции предполагаются дифференцируемыми), а вторую (теорема 3) — условиями второго порядка (потому что входящие в них функции предполагаются дважды дифференцируемыми).

Начнем со следующего понятия: если x^* — точка локального минимума, то значение функции $f(x)$ не может уменьшаться при движении вдоль любой гладкой дуги, направленной из x^* внутрь допустимой области. Пусть вектор v касателен к дуге, идущей из точки x^* . Следуя Фиакко и Мак-Кормику [5], разделим ненулевые векторы v на три класса V_i , каждый из которых представляет собой множество v , определяемое следующим образом:

Класс	Ограничения в виде неравенств	Ограничения в виде равенств	Целевая функция
V_1	$\begin{cases} v^T \nabla g_j(x^*) > 0 \\ \text{для активных ограничений} \end{cases}$	$\begin{cases} v^T \nabla h_j(x^*) = 0 \\ \text{для всех } j = 1, \dots, m \end{cases}$	$\{v^T \nabla f(x^*) \geq 0\}$
V_2	$\begin{cases} v^T \nabla g_j(x^*) \geq 0 \\ \text{для активных ограничений} \end{cases}$	$\begin{cases} v^T \nabla h_j(x^*) = 0 \\ \text{для всех } j = 1, \dots, m \end{cases}$	$\{v^T \nabla f(x^*) < 0\}$
V_3	$\begin{cases} v^T \nabla g_j(x^*) < 0 \\ \text{хотя бы для одного активного ограничения} \end{cases}$	$\begin{cases} v^T \nabla h_j(x^*) \neq 0 \\ \text{хотя бы для одного ограничения} \end{cases}$	

Все допустимые изменения x^* попадают в объединение V_1 и V_2 , и, если v содержиться в V_2 , значение функции $f(x)$ уменьшается, а если v содержиться в V_1 , значение $f(x)$ увеличивается или остается константой. По существу необходимое условие первого порядка представляет собой требование, чтобы множество V_3 было пустым.

Если V_3 пусто, то существование множителей Лагранжа может быть доказано в виде следующей теоремы.

Теорема 1 [5].

Если (а) вектор x^* удовлетворяет условиям задачи (2.5.2), (б) функции $f(x)$, $g_j(x)$ и $h_j(x)$ дифференцируемы и (в) в точке x^* множество V_3 пусто, то существуют векторы (множители Лагранжа) u^* и w^* , такие, что совокупность векторов x^* , u^* , w^* удовлетворяет соотношениям

- (1) $h_j(x^*) = 0, \quad j = 1, \dots, m,$
- (2) $g_j(x^*) \geq 0, \quad j = m + 1, \dots, p,$
- (3) $u_j^* g_j(x^*) = 0, \quad j = m + 1, \dots, p,$

$$(4) \quad u_j^* \geq 0, \quad j = m + 1, \dots, p,$$

$$(5) \quad \nabla L(x^*, u^*, w^*) = 0.$$

В соотношении (5) функцию

$$L(x, u, w) \equiv f(x) + \sum_{j=1}^m w_j h_j(x) - \sum_{j=m+1}^p u_j g_j(x)$$

можно рассматривать как обобщенную функцию Лагранжа задачи (2.5.2); следовательно, уравнение (5) можно записать как

$$\nabla L(x^*, u^*, w^*) \equiv \nabla f(x^*) + \sum_{j=1}^m w_j^* \nabla h_j(x^*) - \sum_{j=m+1}^p u_j^* \nabla g_j(x^*) = 0.$$

Чтобы выяснить, при каких условиях множество V_3 оказывается пустым, необходимо еще одно понятие, а именно *порядок ограничений*. Начнем с ограничений первого порядка. Следуя известной работе Куна и Таккера [4], предположим, что $x^{(k)}$ — допустимая точка для задачи (2.5.2) и что функции $h_1(x), \dots, h_m(x), g_{m+1}(x), \dots, g_p(x)$ дифференцируемы в точке $x^{(k)}$. К ограничениям первого порядка относятся такие, когда для каждой граничной точки множества ограничений, состоящего из ограничений в виде равенств и активных ограничений в виде неравенств, должна существовать гладкая кривая, заканчивающаяся в этой граничной точке и целиком лежащая внутри области, заданной ограничивающими поверхностями. Если x^* соответствует локальному минимуму, то значения $f(x)$ не могут убывать при перемещении вдоль такой кривой из точки x^* внутрь допустимой области. *Достаточным условием того, чтобы ограничения были первого порядка*, является то, что все градиенты активных ограничений в виде неравенств и градиенты ограничений в виде равенств, взятые в некоторой пробной точке x^* , линейно независимы.

Убедиться в том, что множество V_3 пусто, можно различными способами, но лучше всего это сделать, опираясь на понятие *ограничения первого порядка*, что приводит к теореме 2.

Теорема 2 [5, 7].

Если функции $f(x), h_1(x), \dots, h_m(x), g_{m+1}(x), \dots, g_p(x)$ дифференцируемы в точке x^ и если в точке x^* ограничения являются ограничениями первого порядка, то необходимое условие наличия в точке x^* локального минимума задачи (2.5.2) состоит в том, что существуют множители Лагранжа u^* и w^* , такие, что совокупность векторов x^*, u^*, w^* удовлетворяет соотношениям (1) — (5) теоремы 1.*

Чтобы учесть нелинейный характер функций в задаче (2.5.2), Мак-Кормик [6] сформулировал необходимые условия второго порядка наличия в точке x^* локального минимума. Предположим,

что функции $f(x), h_1(x), \dots, h_m(x), g_{m+1}(x), \dots, g_p(x)$ дважды дифференцируемы в точке $x^{(k)}$, удовлетворяющей условиям задачи (2.5.2). Пусть v — любой ненулевой вектор, такой, что

$$v^T \nabla g_i(x^{(k)}) = 0 \text{ для всех активных ограничений в виде неравенств,}$$

$$v^T \nabla h_j(x^{(k)}) = 0 \text{ для всех ограничений в виде равенств.}$$

Тогда если вектор v является касательным к некоторой дважды дифференцируемой кривой $\psi(\theta)$, $\theta \geq 0$, вдоль которой $g_i(\psi(\theta)) = 0$ для всех активных ограничений в виде неравенств и $h_j(\psi(\theta)) = 0$ для всех ограничений в виде равенств, то в $x^{(k)}$ ограничения являются *ограничениями второго порядка*. *Достаточным условием того, чтобы в точке $x^{(k)}$ ограничения были ограничениями второго порядка, является то, что градиенты активных ограничений в виде неравенств и градиенты ограничений в виде равенств в этой точке линейно независимы.*

Теперь можно сформулировать *необходимые условия, при которых ограничения являются ограничениями второго порядка*.

Теорема 3 [7].

Если (a) функции $f(x), h_1(x), \dots, h_m(x), g_{m+1}(x), \dots, g_p(x)$ дважды дифференцируемы в точке x^ и (б) в x^* выполняются классификационные условия ограничений первого и второго порядков, то необходимыми условиями наличия в точке x^* локального минимума задачи (2.5.2) является существование векторов $w^* = [w_1^*, \dots, w_m^*]^T$ и $u^* = [u_{m+1}^*, \dots, u_p^*]^T$, для которых*

$$(1) \quad h_j(x^*) = 0, \quad j = 1, \dots, m,$$

$$(2) \quad g_j(x^*) \geq 0, \quad j = m + 1, \dots, p,$$

$$(3) \quad u_j^* \geq 0, \quad j = m + 1, \dots, p,$$

$$(4) \quad u_j^* g_j(x^*) = 0, \quad j = m + 1, \dots, p,$$

$$(5) \quad \nabla L(x^*, u^*, w^*) = 0,$$

(B)

и (г) для любого ненулевого вектора v , такого, что $v^T \nabla g_i(x^*) = 0$ для всех активных ограничений в виде неравенств и $v^T \nabla h_j(x^*) = 0$ для всех ограничений в виде равенств, справедливо соотношение

$$(6) \quad v^T \nabla^2 L(x^*, u^*, w^*) v \geq 0.$$

Если условия $v^T \nabla g_i(x^*) = 0$ и $v^T \nabla h_j(x^*) = 0$ выполняются лишь при $v = 0$, то соотношение (6) удовлетворяется тривиальным образом, так как активные ограничения пересекаются, задавая, таким образом, единственное решение.

Достаточные условия того, что в точке x^* имеет место изолированный¹⁾ локальный минимум задачи (2.5.2), такие же, как и необходимые условия (а), (в) и (г) теоремы 3, если вместо (б) части (г) теоремы имеет место следующее условие: (г') для любого ненулевого вектора v , для которого $v^T \nabla g_i(x^*) = 0$ в случае активных ограничений-неравенств, $v^T \nabla g_i(x^*) \geq 0$ для неактивных ограничений-неравенств и $v^T \nabla h_i(x^*) = 0$ для всех ограничений в виде равенств, справедливо следующее соотношение [6,7]:

$$(6') v^T \nabla^2 L(x^*, u^*, w^*) v > 0.$$

Другое достаточное условие дается теоремой 4.

Теорема 4 [6].

Если функции $f(x)$, $h_1(x)$, \dots , $h_m(x)$, $g_{m+1}(x)$, \dots , $g_p(x)$ дважды дифференцируемы по x , выполняются необходимые условия (1) — (5) теоремы 3 и детерминант матрицы Якоби для функций $h_j(x)$, $u_j g_j(x)$ и $\nabla L(x, u, w)$ по (x, u, w) не обращается в нуль в (x^*, u^*, w^*) , то в точке x^* удовлетворяются фигурирующие в теореме 3 достаточные условия того, что в точке x^* имеет место локальный минимум.

Пример 2.5.1. Необходимые и достаточные условия существования локального минимума в задаче с ограничениями в виде неравенств

Рассмотрим следующую задачу:

минимизировать $f(x) = x_1^2 + x_2$

при ограничениях

$$g_1(x) = -(x_1^2 + x_2^2) + 9 \geq 0;$$

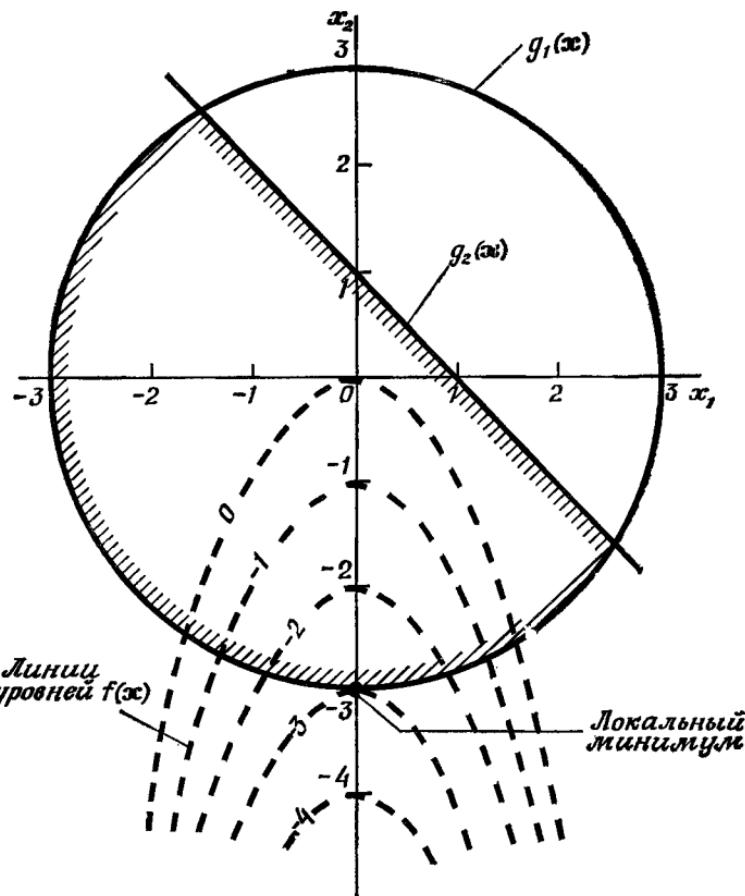
$$g_2(x) = -x_1 - x_2 + 1 \geq 0.$$

В этой задаче $g_1(x) \geq 0$ — активное ограничение, тогда как ограничение $g_2(x) \geq 0$ не является активным (фиг. П.2.5.1).

Нетрудно убедиться, что функции $g_1(x)$ и $g_2(x)$ дважды дифференцируемы. Поскольку имеет место лишь одно активное ограничение ($g_1(x) \geq 0$), нет необходимости проверять, выполняются ли условия первого и второго порядков.

С помощью теорем 1 и 2 нужно показать, что существуют векторы x^* и u^* , удовлетворяющие перечисленным ниже условиям.

¹⁾ «Изолированный» формально означает, что для $0 < \|x\| < \varepsilon$, где ε достаточно мало, $f(x) > f(x^*)$.



Ф и г. П.2.5.1.

Функции $g_1(x^*)$ и $g_2(x^*)$ неотрицательны, т. е.

$$\begin{aligned} -x_1^{*2} - x_2^{*2} + g &\geq 0, \\ -x_1^* - x_2^* + 1 &\geq 0. \end{aligned} \tag{a}$$

Любая точка внутри или на границах заштрихованной области является допустимой точкой x :

$$1 \begin{bmatrix} 2x_1^* \\ 1 \end{bmatrix} - u_1^* \begin{bmatrix} -2x_1^* \\ -2x_2^* \end{bmatrix} - u_2^* \begin{bmatrix} -1 \\ -1 \end{bmatrix} = 0, \tag{b}$$

или

$$2x_1^* + 2x_2^* u_1^* + u_2^* = 0,$$

$$1 + 2x_2^* u_1^* + u_2^* = 0,$$

$$u_1^*(-x_1^{*2} - x_2^{*2} + 9) = 0, \quad (B)$$

$$u_2^*(-x_1^* - x_2^* + 1) = 0,$$

$$u_1^* > 0 \quad \text{и} \quad u_2^* > 0. \quad (G)$$

Если в ограничениях (в) u_1^* и u_2^* положить равными нулю, то возникнет противоречие в соотношениях (б) (поскольку тогда имели бы $1 = 0$). Следовательно, не все u^* могут быть нулями. Предположим, что $u_2^* = 0$. Тогда из первого ограничения в (б) будет следовать, что либо $x_1^* = 0$, либо $(1 + u_1^*) = 0$, но последнее невозможно, потому что u_1^* не могут быть отрицательными. Следовательно, $x_1^* = 0$. Поскольку $u_1^* \neq 0$, то в силу ограничений (в) минимум возможен либо при $x_1^{*2} + x_2^{*2} = 9$, либо при $x_2^* = \pm 3$. Однако при $x_2^* = 3$, если учесть, что $x_1^* = 0$, нарушается второе ограничение в (а). Таким образом, вектор \mathbf{x}^* найден, а именно $\mathbf{x}^* = [0 \ -3]^T$. При этом $u_1^* = 1/6$, $u_2^* = 0$ и ограничивающие условия (б), (в) и (г) оказываются удовлетворенными. Таким образом, необходимые условия первого порядка полностью выполняются.

Необходимое условие второго порядка, которое также должно быть выполнено, заключается в том, что для¹⁾

$$[v_1 \ v_2] \begin{bmatrix} -2x_1^* \\ -2x_2^* \end{bmatrix} = 0$$

или $v_1 \cdot (0) + v_2 \cdot (6) = 0$, т. е. для любого v_1 и $v_2 = 0$, должно иметь место следующее соотношение:

$$[v_1 \ 0] \nabla^2 L(\mathbf{x}^*, \mathbf{u}^*) \begin{bmatrix} v_1 \\ 0 \end{bmatrix} \geq 0.$$

Поскольку

$$L = f(\mathbf{x}) - u_1 g_1(\mathbf{x}) - u_2 g_2(\mathbf{x})$$

и

$$\nabla^2 L = \begin{bmatrix} 2(1 + u_1) & 0 \\ 0 & 2u_1 \end{bmatrix},$$

получаем

$$4v_1^2 u_1^* (1 + u_1^*) \geq 0 > 0.$$

Следовательно, необходимые условия второго порядка и достаточные условия удовлетворяются.

Чтобы определить, удовлетворяются ли альтернативные доста-

¹⁾ Здесь оставляется только уравнение $\mathbf{v}^T \nabla g_1(\mathbf{x}^*) = 0$, так как ограничение, задаваемое функцией $g_2(\mathbf{x})$, является неактивным. — Прим. перев.

точные условия, образуем матрицу Якоби для функций

$$u_1(-x_1^2 - x_2^2 + 9),$$

$$u_2(-x_1 - x_2 + 1),$$

$$2x_1 + 2u_1 x_1 + u_2,$$

$$1 + 2u_1 x_2 + u_2$$

по x_1, x_2, u_1 и u_2 соответственно:

$$J = \begin{bmatrix} -2x_1 u_1 & -2x_2 u_1 & (-x_1^2 - x_2^2 + 9) & 0 \\ -u_2 & -u_2 & 0 & (-x_1 - x_2 + 1) \\ (1 + 2u_1) & 0 & 2x_1 & 1 \\ 0 & 2u_1 & 2x_2 & 1 \end{bmatrix}.$$

При $x_1^* = 0, x_2^* = -3$ и $u_1^* = 1/6, u_2^* = 0$

$$\det J = \det \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ \frac{8}{6} & 0 & 0 & 1 \\ 0 & \frac{1}{3} & -6 & 1 \end{bmatrix} \neq 0.$$

Таким образом, выполняются и альтернативные достаточные условия того, что вектор $\mathbf{x}^* = [0 \ -3]^T$ является локальным минимумом.

Пример 2.5.2. Необходимые и достаточные условия существования локального минимума при наличии ограничений как в виде равенств, так и в виде неравенств

В качестве примера задачи нелинейного программирования, содержащей ограничения как в виде равенств, так и в виде неравенств, рассмотрим следующую задачу:

минимизировать $f(\mathbf{x}) = x_1^2 + x_2, \mathbf{x} \in E^n$,
при ограничениях

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 - 9 = 0,$$

$$g_2(\mathbf{x}) = -(x_1 + x_2)^2 + 1 \geq 0,$$

$$g_3(\mathbf{x}) = -(x_1 + x_2) + 1 \geq 0.$$

В соответствии с теоремой 3 функции $h_1(\mathbf{x}), g_2(\mathbf{x})$ и $g_3(\mathbf{x})$ должны быть дважды дифференцируемы, а градиенты активных ограничений в виде неравенств, а также ограничений в виде равенств должны быть линейно независимы, чтобы выполнялись условия первого и второго порядков. Предположим, что некоторая точка A с координатами $\mathbf{x}^* = [-2,37 \ -1,84]^T$, находящаяся на пересечении $h_1(\mathbf{x})$ и $g_2(\mathbf{x})$, на фиг. П. 2.5.2 рассматривается как возможный локальный оптимум. Образуем линейную комбинацию векторов-градиентов активных

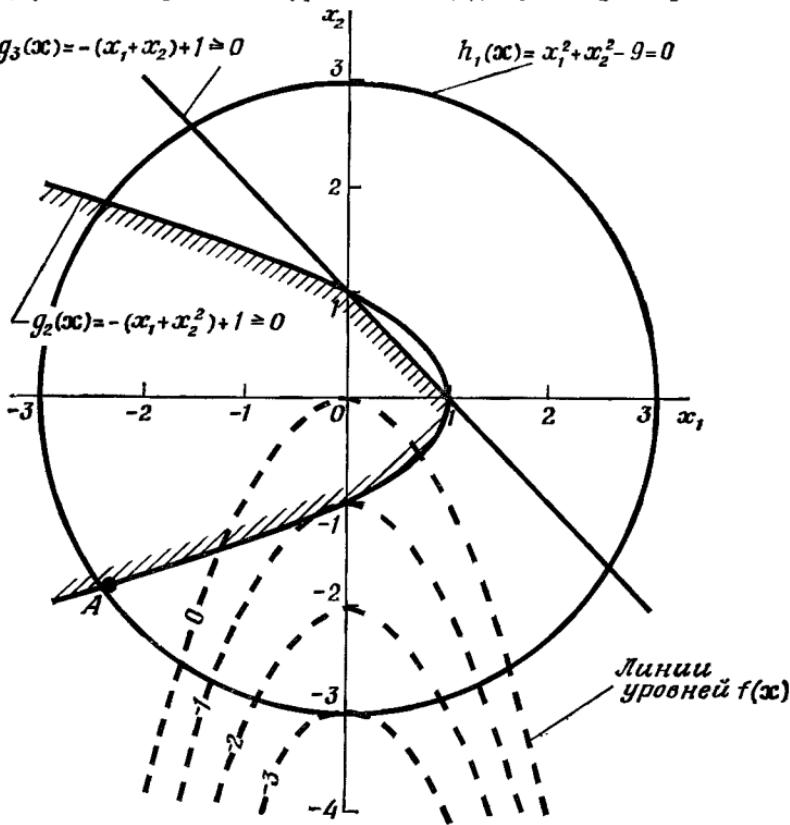
ограничений

$$c_1 \nabla h_1(\mathbf{x}^*) + c_2 \nabla g_2(\mathbf{x}^*) = 0,$$

или

$$c_1 \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} + c_2 \begin{bmatrix} -1 \\ -2x_2^* \end{bmatrix} = 0. \quad (a)$$

Можно показать несколькими способами, что единственным вектором \mathbf{c} , удовлетворяющим уравнению (a), будет $c_1 = c_2 = 0$; следо-



Фиг. П.2.5.2.

вательно, градиенты активных ограничений линейно независимы, и, таким образом, выполняются условия первого и второго порядков.

Проверим теперь, выполняются ли необходимые условия первого порядка теорем 1 и 2. При этом в выбранной точке $\mathbf{x}^* = [-2,37 - 1,84]^T$ должно быть выполнено каждое из следующих условий:

1. $h_1(\mathbf{x}^*) = x_1^{*2} + x_2^{*2} - 9 = 0, (-2,37)^2 + (-1,84)^2 - 9 = 0,$
- $g_2(\mathbf{x}^*) = -(x_1^* + x_2^{*2}) + 1 \geq 0, -(-2,37) - (-1,84)^2 + 1 = 0,$

$$g_3(x^*) = -(x_1^* + x_2^*) + 1 \geq 0, \quad -(-2,37) - (-1,84) + 1 > 0.$$

2. $u_2^* \geq 0$ и $u_3^* \geq 0$.

3. $u_2^*(-x_1^* - x_2^* + 1) = 0,$

$u_3^*(-x_1^* - x_2^* + 1) = 0.$

4. $\begin{bmatrix} 2x_1^* \\ 1 \end{bmatrix} + w_1^* \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} - u_2^* \begin{bmatrix} -1 \\ -2x_2^* \end{bmatrix} - u_3^* \begin{bmatrix} -1 \\ -1 \end{bmatrix} = 0.$

Поскольку в этом примере точка x^* выбрана произвольно для проверки на локальный минимум, подставим эту точку $x^* = [-2,37 \ -1,84]^T$ в выражения, приведенные в условиях 2—4, и проверим, существуют ли такие скаляры w_1^* , u_2^* и u_3^* , которые удовлетворяют соотношениям

$$2(-2,37) + w_1^*(2)(-2,37) + u_2^* + u_3^* = 0,$$

$$1 + w_1^*(2)(-1,84) + 2u_2^*(-1,84) + u_3^* = 0,$$

$$w_1^* = -0,779 \text{ } ^1),$$

$$u_2^* = 1,05.$$

Следовательно, скаляры

$$w_1^* = -0,779,$$

$$u_2^* = 1,05,$$

$$u_3^* = 0$$

удовлетворяют необходимым условиям первого порядка в теореме 2.

Чтобы выяснить, удовлетворяются ли дополнительные требования теоремы 3 в отношении необходимых условий второго порядка, помимо изложенных в теоремах 1 и 2, поступим следующим образом. Определим вектор v при помощи двух уравнений, полученных из ограничения в виде равенства и активного ограничения в виде неравенства:

$$[v_1 \ v_2] \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} = 0,$$

$$[v_1 \ v_2] \begin{bmatrix} -1 \\ -2x_2^* \end{bmatrix} = 0,$$

или

$$-2,37v_1 - 1,84v_2 = 0,$$

$$-v_1 + 3,68v_2 = 0. \quad (b)$$

¹⁾ Так как из второго уравнения условия 3 следует, что $u_3^* = 0$. — Прим. перев.

Единственным вектором v , удовлетворяющим (в), является нулевой вектор. Следовательно, существует единственное решение данной задачи на пересечении ограничений, заданных функциями $g_2(x)$ и $h_1(x)$. Заметим также, что в этой конкретной задаче

$$[v_1 \ v_2] \left\{ \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + w_1^* \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} - u_2^* \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix} \right\} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \geq 0 > 0$$

или (при $w_1^* = -0,779$ и $u_2^* = 1,05$) неравенство

$$0,442v_1^2 + 0,542v_2^2 \geq 0 \quad (r)$$

удовлетворяется для любых ненулевых векторов v .

С другой стороны, достаточные условия того, что в точке x^* имеет место локальный изолированный минимум, следуют и из теоремы 4, поскольку детерминант матрицы Якоби не равен нулю:

$$\mathbf{J} = \begin{bmatrix} 2x_1^* & 2x_2^* & 0 & 0 & 0 \\ -u_2^* & -2u_2^*x_2^* & 0 & (-x_1^* - x_2^{*2} + 1) & 0 \\ -u_3^* & -u_3^* & 0 & 0 & (-x_1^* - x_2^* + 1) \\ 2 + 2w_1^* & 0 & 2x_1^* & 1 & 1 \\ 0 & 2(w_1^* + u_2^*) & 2x_2^* & 2x_2^* & 1 \end{bmatrix},$$

$$\det \mathbf{J} = \det \begin{bmatrix} -4,74 & -3,68 & 0 & 0 & 0 \\ -1,05 & 3,86 & 0 & -0,02 & 0 \\ 0 & 0 & 0 & 0 & +5,21 \\ 0,44 & 0 & -4,74 & 1 & 1 \\ 0 & 0,54 & -3,68 & -3,68 & 1 \end{bmatrix} \neq 0.$$

Выше при иллюстрации применения теорем о локальной оптимальности использовались лишь простые задачи. В случае задач большой размерности эти теоремы также применимы, но здесь могут возникнуть значительные трудности, связанные с необходимостью находить аналитические выражения для производных нелинейных функций $f(x)$, $g(x)$ и $h(x)$.

2.6. ЭФФЕКТИВНЫЕ МЕТОДЫ ОДНОМЕРНОГО ПОИСКА

Многие из алгоритмов, которые будут описаны ниже, основаны на использовании эффективных методов одномерной оптимизации для обнаружения минимума функции одной переменной. В книге Уайлда [8] и в других работах, посвященных общей теории оптимизации, излагается несколько достаточно эффективных методов

одномерного поиска, позволяющих определять интервал значений, внутри которого лежит минимум той или иной функции. Для практического применения этих методов необходимо знать начальный интервал неопределенности $\Delta^{(0)}$, содержащий минимум целевой функции $f(x)$ и в котором функция $f(x)$ унимодальна. Существуют различные методы (некоторые из них приведены в табл. 2.6.1) уменьшения начального интервала $\Delta^{(0)}$ до некоторого конечного $\Delta^{(n)}$. Мы ограничимся лишь несколькими замечаниями, касающимися

Таблица 2.6.1

Эффективность одномерных методов поиска
(n — число вычислений функции, F_n — число Фибоначчи для n оценок)

Непоследовательные методы	E'	Последовательные методы	E
Равномерный поиск	$\frac{2}{n+1}$	Поиск делением пополам	$\frac{1}{2^{n/2}}$
Равномерный поиск делением пополам	$\frac{1}{\frac{n}{2}+1}$	Поиск Фибоначчи Поиск методом золотого сечения	$\frac{1}{F_n}$ $(0,618)^{1-n}$

этих методов, а затем перейдем к некоторым другим методам, которые обычно на практике оказываются более эффективными.

Уайлд определил эффективность процедуры, включающей n вычислений целевой функции, как

$$E = \frac{\Delta^{(n)}}{\Delta^{(0)}}.$$

В табл. 2.6.1 сравнивается эффективность пяти различных методов.

В табл. 2.6.2 сравниваются количества вычислений функции $f(x)$, необходимые для уменьшения начального интервала, равного $5 \cdot 10^{-1}$.

Оказывается, что относительная эффективность двух хорошо известных методов золотого сечения и поиска Фибоначчи несколько выше эффективности поиска последовательным делением пополам и существенно превосходит эффективность поиска непоследовательными методами.

Поиск с помощью метода золотого сечения основан на разбиении отрезка прямой на две части, известном как «золотое сечение». При этом отношение длины всего отрезка к большей части равно отношению большей части к меньшей. Здесь используются

две дроби Фибоначчи:

$$F_1 = \frac{3 - \sqrt{5}}{2} \approx 0,38,$$

$$F_2 = \frac{\sqrt{5} - 1}{2} \approx 0,62.$$

Заметим, что $F_1 = (F_2)^2$ и $F_1 + F_2 = 1$. Поиск необходимо начинать в таком направлении, чтобы значение функции $f(x)$ уменьшалось.

Таблица 2.6.2

Количество вычислений функции, необходимое для уменьшения начального интервала, равного $5 \cdot 10^{-1}$

Уменьшение интервала до	Непоследовательные методы		Последовательные методы		
	равномерный поиск	равномерный поиск делением пополам	поиск методом золотого сечения	поиск Фибоначчи	поиск делением пополам
$5 \cdot 10^{-3}$	199	198	11	11	14
$5 \cdot 10^{-5}$	19 999	19 998	21	21	28

Начальный отрезок Δ , включающий минимум $f(x)$, может быть получен с помощью, например, последовательной серии нескольких возрастающих шагов по независимой переменной. Обозначим последние три значения x через $x_3^{(0)}$ (последняя точка), $x_2^{(0)}$ и $x_1^{(0)}$, где $f(x_3^{(0)}) \geq f(x_2^{(0)})$, и пусть $\Delta^{(k)} = x_3^{(k)} - x_1^{(k)}$. Рассмотрим фиг. 2.6.1. В соответствии с этим рисунком, если мы находимся на k -м этапе, то $(k+1)$ -й интервал нужно вычислять следующим образом.

Определим

$$y_1^{(k)} = x_1^{(k)} + F_1 \Delta^{(k)},$$

$$y_2^{(k)} = x_1^{(k)} + F_2 \Delta^{(k)} = x_3^{(k)} - F_1 \Delta^{(k)}.$$

Если $f(y_1^{(k)}) < f(y_2^{(k)})$, то $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)})$ и $x_1^{(k+1)} = x_1^{(k)}$, $x_3^{(k+1)} = y_2^{(k)}$;

если $f(y_1^{(k)}) > f(y_2^{(k)})$, то $\Delta^{(k+1)} = (x_3^{(k)} - y_1^{(k)})$ и $x_1^{(k+1)} = y_1^{(k)}$, $x_3^{(k+1)} = x_3^{(k)}$;

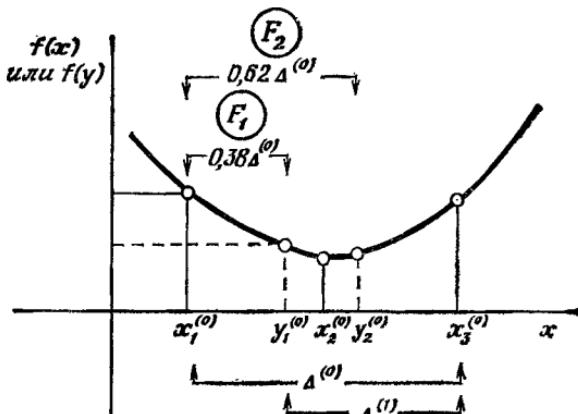
если $f(y_1^{(k)}) = f(y_2^{(k)})$, то $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)}) = (x_3^{(k)} - y_1^{(k)})$ и $x_1^{(k+1)} = x_1^{(k)}$, $x_3^{(k+1)} = y_2^{(k)}$

или

$$x_1^{(k+1)} = y_1^{(k)}, \quad x_3^{(k+1)} = x_3^{(k)}.$$

Для большей точности каждый раз определяются две новые точки (а не одна новая точка), поскольку значения F_1 и F_2 не являются точными; при оперировании только с одной новой точкой ошибки округления при вычислении могут привести к потере интервала, содержащего минимум.

В другом классе методов одномерной минимизации текущая точка x вблизи x^* [значение независимой переменной, соответствующей



Фиг. 2.6.1. Поиск методом золотого сечения (золотой поиск).

Начальные точки: $x_1^{(0)}$, $x_2^{(0)}$ и $x_3^{(0)}$.

Поскольку $f(y_1^{(0)}) > f(y_2^{(0)})$, $\Delta^{(1)} = (x_3^{(0)} - y_1^{(0)})$, а $x_1^{(1)} = y_1^{(0)}$ и $x_3^{(1)} = x_3^{(0)}$ являются границами интервала для этапа 1.

$$y_1^{(0)} = x_1^{(0)} + 0,38\Delta^{(0)},$$

$$y_2^{(0)} = x_1^{(0)} + 0,62\Delta^{(0)}.$$

минимуму $f(x)$] определяется с помощью экстраполяции и интерполяций. Были предложены квадратичные и кубические формулы аппроксимации либо только значения функции, либо и функции и производных. Коггинс [9] отметил, что, пользуясь некоторыми из методов, основанных на аппроксимации функции полиномом, проходящим через выбранные точки, можно легче обнаружить минимум $f(x)$ при заданной точности, чем методами, указанными в табл. 2.6.1. Бокс, Дэвис и Свенн [10] предложили использовать алгоритм Дэвиса, Свенна и Кэмпи (ДСК) [11] для определения интервала, содержащего точку минимума, чтобы все последующие вычисления проводились по алгоритму Пауэлла [12]. Описание этих двух алгоритмов приведено ниже.

При одномерном поиске по алгоритму ДСК проводятся возрастающие по величине шаги до тех пор, пока не будет пройден минимум, а затем выполняется одноразовая квадратичная интерполяция. На фиг. 2.6.2 показана такая процедура (здесь $x^{(m)}$ — первое

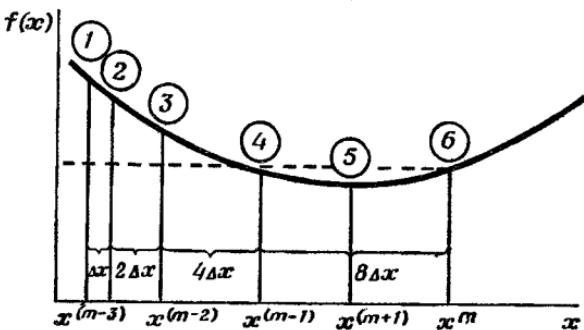
из значений x , для которого зарегистрировано отклонение от минимума, а Δx — длина шага), состоящая из следующих шагов:

Шаг 1. Вычислить $f(x)$ в начальной точке $x^{(0)}$. Если $f(x^{(0)} + \Delta x) \leq f(x^{(0)})$, то перейти к шагу 2. Если $f(x^{(0)} + \Delta x) > f(x^{(0)})$, положить $\Delta x = -\Delta x$ и перейти к шагу 2.

Шаг 2. Вычислить $x^{(k+1)} = x^{(k)} + \Delta x$.

Шаг 3. Вычислить $f(x^{(k+1)})$.

Шаг 4. Если $f(x^{(k+1)}) \leq f(x^{(k)})$, удвоить Δx и вернуться к шагу 2 при $k = k + 1$. Если $f(x^{(k+1)}) > f(x^{(k)})$, обозначить $x^{(k+1)}$ через



Ф и г. 2.6.2. Одномерная минимизация методом ДСК.

$x^{(m)}$, $x^{(k)}$ через $x^{(m-1)}$ и т. д., уменьшить Δx наполовину и вернуться к шагам 2 и 3 для еще одного (только одного) вычисления.

Шаг 5. Из четырех равноотстоящих значений $x^{(m+1)}, x^{(m)}, x^{(m-1)}, x^{(m-2)}$ исключить либо $x^{(m)}$, либо $x^{(m-2)}$ в зависимости от того, какое из них находится дальше от x , соответствующего наименьшему значению $f(x)$. Пусть $x^{(a)}, x^{(b)}$ и $x^{(c)}$ — оставшиеся три значения x , где $x^{(b)}$ — центральная точка а $x^{(a)} = x^{(b)} - \Delta x$ и $x^{(c)} = x^{(b)} + \Delta x$.

Шаг 6. Провести квадратичную интерполяцию для определения x^* :

$$x^* \approx \tilde{x}^* = x^{(b)} + \frac{\Delta x [f(x^{(a)}) - f(x^{(c)})]}{2[f(x^{(a)}) - 2f(x^{(b)}) + f(x^{(c)})]}.$$

Эти шаги завершают первый этап метода ДСК. При желании продолжить эту процедуру нужно начать из \tilde{x}^* или из $x^{(c)}$, если $f(x^{(c)}) < f(\tilde{x}^*)$, уменьшить Δx и начать с первого шага.

В алгоритме Паузелла квадратичная аппроксимация проводится путем использования первых трех точек, полученных в направлении поиска. Затем определяется точка x , соответствующая минимуму квадратичной функции. Такая квадратичная аппроксимация продолжается до тех пор, пока с требуемой точностью не обнаруживается минимум $f(x)$. Алгоритм Паузелла строится следующим образом (фиг. 2.6.3):

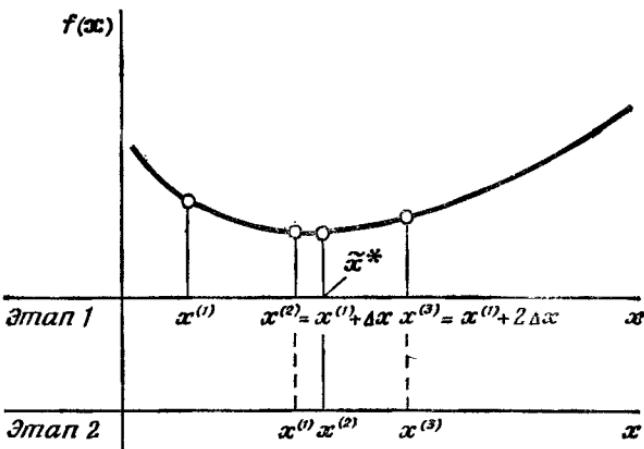
Шаг 1. От начальной точки $x^{(1)}$ нужно перейти к $x^{(2)} = x^{(1)} + \Delta x$.

Шаг 2. Вычислить $f(x^{(1)})$ и $f(x^{(2)})$.

Шаг 3. Если $f(x^{(1)}) > f(x^{(2)})$, положить $x^{(3)} = x^{(1)} + 2\Delta x$.

Если $f(x^{(1)}) \leq f(x^{(2)})$, положить $x^{(3)} = x^{(1)} - \Delta x$.

Шаг 4. Вычислить $f(x^{(3)})$.



Ф и г. 2.6.3. Одномерная минимизация методом Пауэлла.

Шаг 5. Вычислить приближенное значение x в точке минимума $f(x)$ по формуле

$$\tilde{x}^* = -\frac{1}{2} \frac{[(x^{(2)})^2 - (x^{(3)})^2] f(x^{(1)}) + [(x^{(3)})^2 - (x^{(1)})^2] f(x^{(2)}) + [[(x^{(1)})^2 - (x^{(2)})^2] f(x^{(3)})]}{(x^{(2)} - x^{(3)}) f(x^{(1)}) + (x^{(3)} - x^{(1)}) \times [f(x^{(2)}) + (x^{(1)} - x^{(2)}) f(x^{(3)})]}.$$

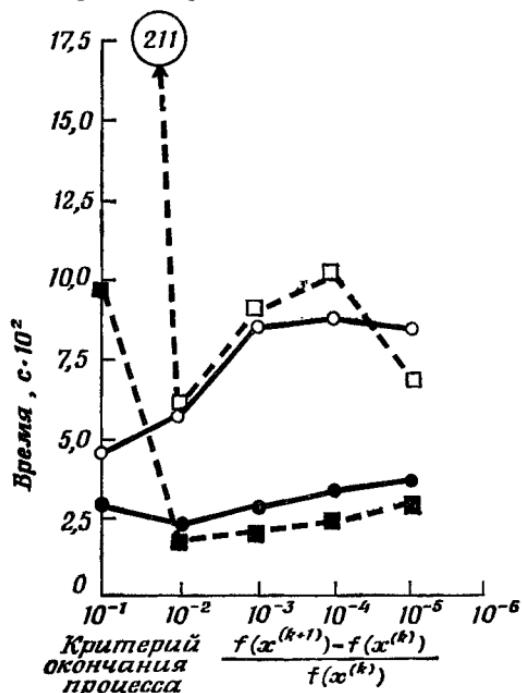
Шаг 6. Если \tilde{x}^* и любое из значений $x \{x^{(1)}, x^{(2)}, x^{(3)}\}$, соответствующих минимуму $f(x)$, отличаются меньше, чем на предписанную точность x или точность соответствующих значений функции $f(x)$, закончить поиск. В противном случае вычислить $f(\tilde{x}^*)$ и исключить из множества $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ то значение x , которое соответствует наибольшему значению $f(x)$, если, однако, при этом не будет потерян интервал, в котором находится минимум $f(x)$. В этом случае следует так исключить x , чтобы сохранить этот интервал. Перейти к шагу 5.

Работа алгоритма продолжается до тех пор, пока не будет достигнута желаемая точность, упомянутая в шаге 6.

Комбинация алгоритмов ДСК и Пауэлла оказалась эффективнее

по сравнению с каждым из этих алгоритмов в отдельности. Комбинированный алгоритм ДСК — Пауэлла состоит из одного этапа (шаги с 1 по 6) алгоритма ДСК, на котором определяется интервал, в котором находится минимум $f(x)$, шага 6 алгоритма Пауэлла и затем, если это оказывается необходимым, шагов 5 и 6 алгоритма Пауэлла.

Прямое сравнение метода одномерного поиска ДСК — Пауэлла,



Фиг. 2.6.4. Время решения двух задач при использовании различных методов одномерного поиска.

■ золотой поиск, задача 1; □ золотой поиск, задача 2; ● метод ДСК — Пауэлла, задача 1; ○ метод ДСК — Пауэлла, задача 2.

например, с методом золотого сечения с точки зрения требуемого времени или числа проведенных вычислений значений функции имеет не очень большой смысл. Правда, такое сравнение позволяет исключить плохие методы; важно, однако, знать, насколько эффективен алгоритм одномерного поиска при решении оптимизационных задач большой размерности.

На фиг. 2.6.4 приводятся данные относительно времени решения двух задач, одна из которых с двумя, а другая с четырьмя независимыми переменными (см. разд. 3.4).

Задача 1. Функция Розенброка

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

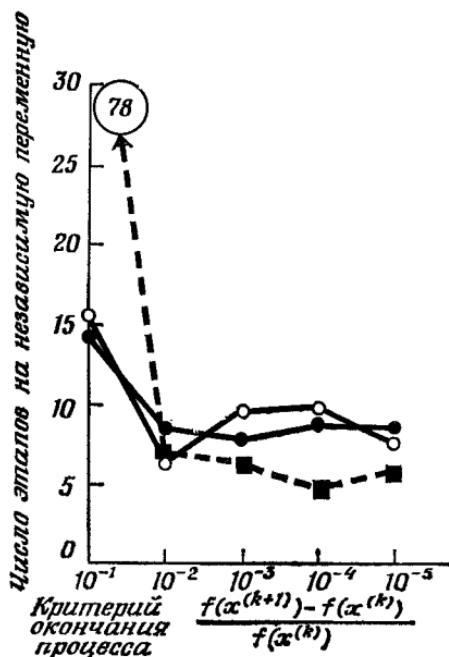
Задача 2. Функция Флетчера — Пауэлла

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

При различных критериях окончания процесса, выраженных через относительное приращение $f(x)$, методы поиска ДСК — Пауэлла и золотого сечения оказались примерно одинаково эффективными в смысле времени решения, за исключением случая самого низкого (по точности) значения критерия (10^{-1}).

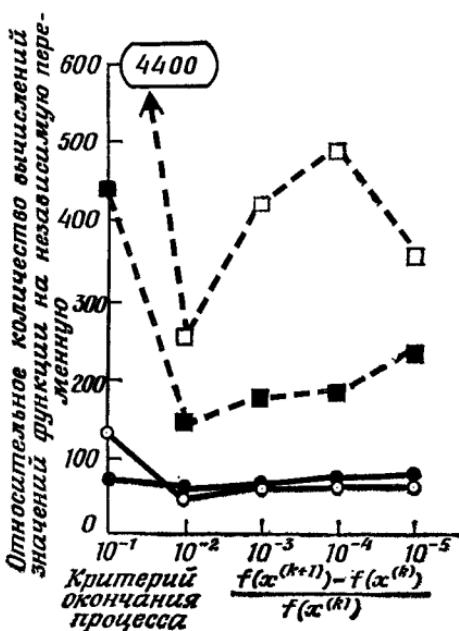
На фиг. 2.6.5 приведены графики зависимости числа оптимизационных этапов на одну независимую переменную от значений

критерия окончания процесса. Число этапов для поиска методом золотого сечения в задаче 2 оказалось настолько большим (выходящим за поле графика), что его пришлось опустить. На фиг. 2.6.6 показано, как изменялось количество вычислений значений функции



Фиг. 2.6.5. Число этапов, проведенных до окончания процесса, при использовании различных методов одномерного поиска.

■ золотой поиск, задача 1; ● метод ДСК—Паузэлла, задача 1; ○ метод ДСК—Паузэлла, задача 2.

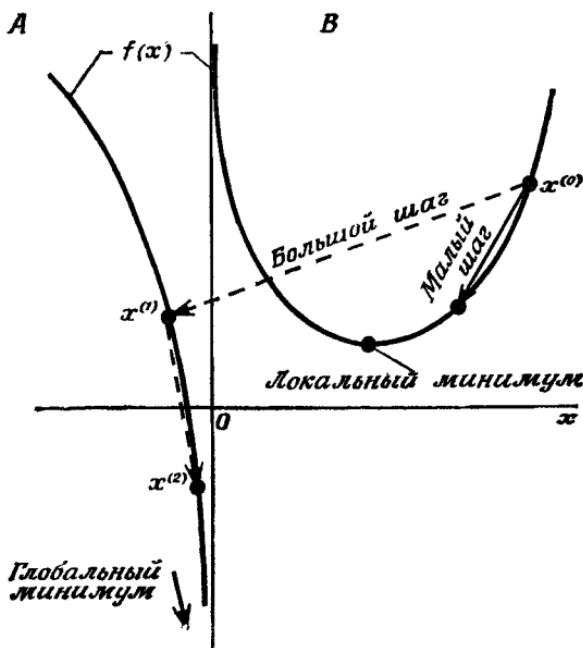


Фиг. 2.6.6. Относительное количество вычислений значений функции на независимую переменную при использовании различных методов одномерного поиска.

■ золотой поиск, задача 1; □ золотой поиск, задача 2; ● метод ДСК—Паузэлла, задача 1; ○ метод ДСК—Паузэлла, задача 2.

при различных методах одномерного поиска в зависимости от критерия окончания процесса. Если для вычисления целевой функции требуется значительное время, то, как видно из фиг. 2.6.6, поиск методом ДСК — Паузэлла предпочтительнее. Данные относительно числа вычислений частных производных (компонент градиента) не приводятся, но следует иметь в виду, что на каждом этапе задачи 1 требовалось вычислять значения производных дважды и четырех раза вычислялись производные на каждом этапе задачи 2.

Если минимизируемая функция локально не унимодальная, как это предполагалось выше, необходимо добавить к программе одномерного поиска дополнительную логическую процедуру, чтобы быть уверенным в том, что величина шага выбрана так, что гарантируется продвижение к тому локальному минимуму,



Ф и г. 2.6.7. Одномерный поиск локального минимума мультимодальной целевой функции приводит к неограниченному решению.

который определяется. Например, фиг. 2.6.7 иллюстрирует, как большой начальный шаг может привести к неограниченному решению, когда в действительности мы ищем локальный минимум.

2.7. КЛАССИФИКАЦИЯ МЕТОДОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Для решения общей задачи нелинейного программирования было предложено довольно много алгоритмов, поскольку ни один из них не оказался значительно лучше других. Выбор того или иного метода определяется конкретным содержанием задачи и опытом исследователя.

В табл. 2.7.1 приводятся некоторые способы классификации задач нелинейного программирования. Последующие главы этой книги основываются на схеме классификации задач по их постановке, которая в этой таблице обозначена номером 1.1. Но не менее важной является схема, основанная на характерных различиях методов решения (классификация 2 в табл. 2.7.1). Поскольку и технический и программный уровень современных ЭВМ изменяется очень быстро, в этой книге не рассматриваются методы оптимизации с точки зре-

Таблица 2.7.1

Классификация методов нелинейного программирования

1. Классификация по некоторым аспектам постановки задачи

1.1. Характер целевой функции (с ограничениями, без ограничений)

- (а) Без ограничений
- (б) Ограничения в виде равенств
- (в) Ограничения в виде неравенств
- (г) Ограничения как в виде равенств, так и в виде неравенств

1.2. Дискретные (целочисленные) переменные; переменные, принимающие непрерывные значения

1.3. Выпуклое, квадратичное, сепарабельное и т. д. программирование

2. Классификация по характерным чертам методов решения

2.1. Методы, использующие производные; методы, не использующие производные (поиск)

2.2. Аналитическое определение производных; численное определение производных

2.3. Методы, использующие первые производные; методы, использующие вторые производные

2.4. Градиентные методы; методы, не использующие градиент

2.5. Градиентный метод с малым шагом; градиентный метод с большим шагом

2.6. Одновременная итерация по всем переменным в процессе поиска; релаксация (последовательный поиск каждый раз по одной переменной)

2.7. Методы внутренней точки; методы внешней точки

2.8. Детерминированный поиск; случайный поиск

2.9. Начальный вектор находится в допустимой или в недопустимой области

3. Классификация по типу вычислительных машин, применяемых при проведении алгоритма

3.1 Цифровая, гибридная или аналоговая машина

4. Классификация по используемому языку программирования

ния категорий 3 или 4. Гилберт [13] составил аннотированную библиографию по методам параметрической оптимизации, применимым для гибридных ЭВМ.

ЗАДАЧИ

2.1. Запишите следующие задачи линейного программирования в матричной форме:

а) минимизировать $f(\mathbf{x}) = 3x_1 + 2x_2 + x_3$
при ограничениях

$$g_1(\mathbf{x}) = 2x_1 + 3x_2 + x_3 \geq 10,$$

$$g_2(\mathbf{x}) = x_1 + 2x_2 + x_3 \geq 15;$$

б) максимизировать $f(\mathbf{x}) = 5x_1 + 10x_2 + 12x_3$

при ограничениях

$$g_1(\mathbf{x}) = 15x_1 + 10x_2 + 10x_3 \leq 200,$$

$$g_2(\mathbf{x}) = x_4 \geq 0,$$

$$g_3(x) = x_2 \geq 0,$$

$$g_4(x) = x_3 \geq 0,$$

$$h_1(x) = 10x_1 + 25x_2 + 20x_3 = 300.$$

2.2. Запишите в матричных обозначениях путем введения соответствующих матриц следующую целевую функцию:

$$f(x) = 3 + 2x_1 + 3x_2 + 2x_1^2 + 2x_1x_2 + 6x_2^2, \quad x = [x_1 \ x_2]^T.$$

2.3. Приведите задачу 2.37 к стандартной форме задачи нелинейного программирования, описываемой уравнениями (2.2.1) — (2.2.3).

2.4. Классифицируйте следующие задачи как 1) линейные, 2) квадратичные, 3) выпуклые, 4) нелинейные задачи программирования (одна и та же задача может классифицироваться по нескольким признакам):

$$2.1a, \quad 2.15,$$

$$2.1b, \quad 2.16,$$

$$2.6, \quad 2.18.$$

$$2.11,$$

2.5. Определите, какие из приведенных ниже матриц являются 1) положительно определенными, 2) отрицательно определенными, 3) не относятся ни к одной из упомянутых категорий:

$$(a) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (b) \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad (v) \begin{bmatrix} 0,1 & 0 \\ 3 & 1 \end{bmatrix}, \quad (r) \begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}.$$

2.6. Изобразите графически целевые функции и ограничения для следующих задач нелинейного программирования:

а) минимизировать $f(x) = 2x_1^2 - 2x_1x_2 + 2x_2^2 - 6x_1 + 6$
при ограничении

$$g_1(x) = x_1 + x_2 \leq 2;$$

б) минимизировать $f(x) = x_1^3 - 3x_1x_2 + 4$

при ограничениях

$$g_1(x) = 5x_1 + 2x_2 \geq 18,$$

$$h_1(x) = -2x_1 + x_2^2 = 5;$$

в) минимизировать $f(x) = -5x_1^2 + x_2^2$

при ограничениях

$$g_1(x) = \frac{x_1^2}{x_2^2} - \frac{1}{x_2} \leq -1,$$

$$g_2(x) = x_1 \geq 0,$$

$$g_3(x) = x_2 \geq 0.$$

2.7. Исследуйте задачи 2.1а, 2.1б, 2.38 и задачу 13 (в приложении А). Для каждой задачи укажите 1) полное число переменных и 2) число независимых переменных. Предложите подходящую систему переменных для обеих категорий.

2.8. Сформулируйте следующие задачи в виде задач нелинейного программирования.

а) Найти длину, высоту и ширину прямоугольного резервуара, открытого сверху, которые дают максимальный объем при фиксированной площади поверхности A (A — площадь стенок + площадь дна).

б) Найти кривую наискорейшего спуска из одной точки в другую, т. е. траекторию, по которой должна двигаться частица, чтобы время перехода было минимальным. (Такая кривая называется брахистохроной.) Такая классическая задача впервые была предложена Джоном Бернулли в 1696 г.

в) Конденсирующийся пар при температуре T_s в теплообменнике используется для нагревания масла от температуры T_1 до T_2 . Эффективный выигрыш от добавочной площади теплообменника может быть выражен в виде дополнительной прибыли исходного состояния:

$$R = \underbrace{QC_Q\theta}_{\text{стоимость приобретенной энергии}} - \underbrace{AC_Fr}_{\text{стоимость дополнительного объема теплообменника}} - rC_c, \quad (1)$$

где

R — дополнительные годовые доходы, долл./год;

Q — скорость передачи тепла, БЕТ/ч;

C_Q — удельная стоимость переносимой дополнительной энергии, долл./БЕТ;

θ — количество рабочих часов в год;

A — дополнительная площадь теплообменника, используемая в передаче тепла, фут²;

C_F — удельная дополнительная стоимость площади теплообменника, долл./фут²;

r — относительные ежегодные издержки, [издержка, долл]/[стоимость, долл · (год)];

C_c — стоимость теплообменника без добавочной площади, долл./ч.

Уравнение энергетического баланса, представляющее собой ограничение для целевой функции (1) (полученное на основе макроскопической модели теплообменника в установившемся режиме), имеет следующий вид:

$$WC_P(T_2 - T_1) = Q = UA \frac{T_2 - T_1}{\ln [(T_s - T_1)/(T_s - T_2)]},$$

где

W — скорость потока (константа), фунт/ч;

C_p — теплоемкость (константа), БЕТ/фунт · °F;

U — общий коэффициент теплопередачи (константа),
БЕТ/ч · фут² · ΔT.

Найти максимум годового дохода.

г) Капиталовложения в трубопроводы и соответствующие установки составляют часть общих вложений в химическое предприятие. Поэтому необходимо выбирать такие размеры труб, при которых суммарные издержки, включающие в себя некоторые постоянные издержки и издержки при работе насоса, минимальны. Оптимальная стоимость трубы может быть выражена следующим образом:

$$C = \underbrace{\frac{AqKWHg}{PD^3E}}_{\substack{\text{стоимость} \\ \text{работы} \\ \text{насоса}}} + \underbrace{XD^n + (1 + F) XD^n K_F}_{\substack{\text{постоянная составляющая} \\ \text{издержек и издержки} \\ \text{монтажа}}}, \quad (1)$$

где

C — общая стоимость, долл/ (год · фут);

A — константа;

q — скорость потока жидкости, фут³/с;

ρ — плотность жидкости, фунт/фут³;

μ_0 — вязкость жидкости, сантипуаз;

K — стоимость электроэнергии, долл/квт · ч;

W — механическая работа, фут · фунт/фунт;

H_g — количество рабочих часов в год;

E — коэффициент полезного действия двигателя и насоса (безразмерная величина);

V — средняя линейная скорость жидкости, фут/с;

D — внутренний диаметр трубы, дюйм;

L — длина трубы;

X — стоимость 1 фута новой трубы диаметром 1 дюйм, долл/фут;

n — константа, зависящая от типа трубы;

F — отношение общей стоимости монтажа к стоимости трубы;

K_F — ежегодные постоянные издержки, включающие расходы на ремонт.

Уравнение баланса механической энергии для потока жидкости имеет вид

$$W = \frac{2fV^2L(1 - J)}{g_c D} .$$

Для вязкого потока ($N_{Re} < 2100$)

$$f = \frac{16}{N_{Re}}, \quad N_{Re} = \frac{LV\rho}{\mu},$$

где

f — коэффициент трения;

N_{Re} — число Рейнольдса;

g_c — 32,17 фут · фунт массы/с² · фунт силы;

J — относительные потери вследствие сочленений и изгибов (безразмерная величина).

Найти минимум стоимости, соответствующий оптимальному диаметру трубы.

2.9. Что такое унимодальная функция и каково ее значение в теории оптимизации?

2.10. Разделить локальный и глобальный экстремумы следующей целевой функции:

$$f(x) = 2x_1^3 + x_2^2 + x_1^2x_2^2 + 4x_1x_2 + 3.$$

2.11. Укажите число независимых переменных в следующей задаче:

$$\begin{aligned} \text{минимизировать } & \frac{13}{2}x_1^2 + \frac{5}{2}x_2^2 + \frac{1}{2}x_3^2 - 4x_1x_2 + 3x_1x_3 - \\ & - 2x_2x_3 + y_4^2 + y_4y_5 + y_5^2 + 3y_4 - 2y_5 \end{aligned}$$

при ограничениях

$$y_1 \equiv 13x_1 - 4x_2 + 3x_3 + y_4 + 3y_5 \geq 0,$$

$$y_2 \equiv -4x_1 + 5x_2 - 2x_3 + y_5 - 1 \geq 0,$$

$$y_3 \equiv 3x_1 - 2x_2 + x_3 - 2y_4 - 2 \geq 0,$$

$$y_4 \geq 0,$$

$$y_5 \geq 0.$$

2.12. Приведите пример выпуклой целевой функции, вогнутой целевой функции, выпуклой целевой функции при вогнутых ограничениях в виде неравенств.

2.13. Является ли следующая задача:

$$\text{минимизировать } f(x) = 100x_1 + \frac{200}{x_1x_2}$$

при ограничениях

$$2x_2 + \frac{300}{x_1x_2} \leq 1,$$

$$x_1, x_2 \geq 0$$

задачей выпуклого программирования?

2.14. Приведите пример положительно определенной матрицы; полуопределенной матрицы.

2.15. Приведите целевую функцию, у которой имеется более одного локального минимума, но не бесконечное число. Выделите глобальный минимум.

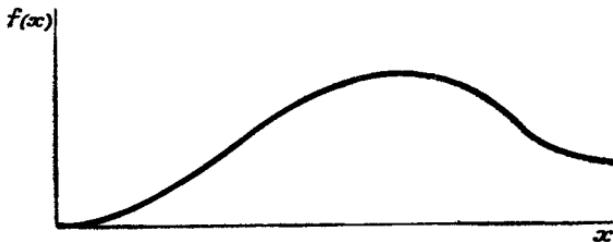
2.16. Являются ли следующие векторы:

- (1) $\mathbf{x} = [5 \ 2 \ 10]^T$,
- (2) $\mathbf{x} = [10 \ 2 \ 7,5]^T$,
- (3) $\mathbf{x} = [0 \ 0 \ 0]^T$

(а) допустимыми или недопустимыми по отношению к условиям задачи 2.1б; (б) внутренними или внешними векторами?

2.17. Заштрихуйте область допустимых значений в задачах 2.6 нелинейного программирования. Является ли в этих задачах $\mathbf{x} = [1 \ 1]^T$ внутренней, граничной или внешней точкой?

2.18. Укажите, какая часть изображенной на фиг. 3.2.18 функции является выпуклой, а какая вогнутой.



Ф и г. 3.2.18.

2.19. Является ли квадратичная функция

$$f(\mathbf{x}) = 10 + 10x_1 + x_2 - 6x_1^2 - 3x_2^2$$

выпуклой, вогнутой, невыпуклой и невогнутой или и выпуклой и вогнутой? Является ли в общем случае квадратичная функция либо только выпуклой, либо только вогнутой?

2.20. Сепарабельные функции — это такие функции, которые могут быть записаны в виде

$$\psi(\mathbf{x}) = \sum_{i=1}^n \psi_i(\mathbf{x}).$$

Например, $x_1^2 + x_2^2 + x_3^2$ является сепарабельной функцией, поскольку

$$\psi(\mathbf{x}) = \sum x_i^2.$$

Показать, что если все слагаемые, входящие в сепарабельную функцию, выпуклые, то сепарабельная функция также является выпуклой.

2.21. Выясните, являются ли выпуклыми следующие целевые функции:

$$(a) f(\mathbf{x}) = 3x_1^2 - 4x_1x_2 + x_2^2,$$

$$(b) f(\mathbf{x}) = e^{x_1} + x_2^2 + 1.$$

2.22. Что является областью допустимых значений для \mathbf{x} при указанных ниже ограничениях? Изобразите графически область допустимых значений для двумерных оптимизационных задач со следующими ограничениями:

- (а) $h_1(\mathbf{x}) = x_1 + x_2 - 3 = 0,$
 $h_2(\mathbf{x}) = 2x_1 - x_2 + 1 = 0;$
- (б) $h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 = 0,$
 $h_2(\mathbf{x}) = x_1 + x_2 + x_3 = 0;$
- (в) $g_1(\mathbf{x}) = x_1 - x_2^2 - 2 \geq 0,$
 $g_2(\mathbf{x}) = x_1 - x_2 + 4 \geq 0;$
- (г) $h_1(\mathbf{x}) = x_1^2 + x_2^2 + 3,$
 $g_1(\mathbf{x}) = x_1 - x_2 + 2 \geq 0,$
 $g_2(\mathbf{x}) = x_1 \geq 0,$
 $g_3(\mathbf{x}) = x_2 \geq 0.$

2.23. Ответьте на приведенные ниже вопросы к следующей задаче (в каждом случае докажите правильность ответа):

минимизировать $f(\mathbf{x}) = \frac{1}{4}x_1^4 - \frac{1}{2}x_1^2 - x_2$

при ограничениях

$$\begin{aligned} x_1^2 + x_2^2 &= 4, \\ x_1 - x_2 &\leq 2. \end{aligned}$$

- (а) Является ли данная задача задачей выпуклого программирования?
- (б) Является ли точка $\mathbf{x} = [1 \ 1]^T$ допустимой?
- (в) Является ли точка $\mathbf{x} = [2 \ 2]^T$ внутренней?
- (г) Является ли $f(\mathbf{x})$ унимодальной функцией?

2.24. При каких условиях локальный минимум будет одновременно и глобальным минимумом? (Постарайтесь дать краткий ответ.)

2.25. Найдите выражения для градиента приведенных ниже целевых функций и вычислите значения градиента в указанных точках:

(а) $f(\mathbf{x}) = 3x_1^2 - 2x_1x_2 + 6x_2^2$

в точке $\mathbf{x} = [0 \ 0]^T,$

в точке $\mathbf{x} = [1 \ 2]^T;$

(б) $f(\mathbf{x}) = 4x_1^2 - 2x_1x_2 + 2x_2^3$

в точке $\mathbf{x} = [-1 \ 0]^T,$

в точке $\mathbf{x} = [-1 \ -1]^T.$

2.26. Чему равен градиент $f(\mathbf{x}) = x_1 e^{x_2} + x_1 x_2$ в точке $\mathbf{x} = [2 \quad 3]^T$?

2.27. Аппроксимируйте $f(\mathbf{x})$ задачи 2.26 в точке $\mathbf{x} = [1 \quad 1]^T$ с помощью квадратичного приближения.

2.28. Аппроксимируйте с помощью (1) линейных членов (первого порядка) ряда Тейлора и (2) квадратичных (второго порядка) членов ряда Тейлора целевую функцию

$$f(\mathbf{x}) = e^{x_1^2 + 2x_2^2} - 10x_1 x_2 \text{ в точке } \mathbf{x} = [1 \quad 1]^T.$$

2.29. Если для ограничений выполняются условия второго порядка, выполняются ли также условия первого порядка?

2.30. Укажите необходимые и достаточные условия максимума функции одной переменной.

2.31. Было найдено два решения задачи нелинейного программирования

$$\text{минимизировать } f(\mathbf{x}) = 7x_1 - 6x_2 + 4x_3$$

при ограничениях

$$h_1(\mathbf{x}) = x_1^2 + 2x_2^2 + 3x_3^2 - 1 = 0,$$

$$h_2(\mathbf{x}) = 5x_1 + 5x_2 - 3x_3 - 6 = 0$$

следующего вида:

$$\mathbf{x} = \begin{bmatrix} 0,947 \\ 0,207 \\ -0,0772 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 0,534 \\ 0,535 \\ -0,219 \end{bmatrix},$$

$$f(\mathbf{x}) = 5,08, \quad f(\mathbf{x}) = -0,346.$$

Одно из этих решений, очевидно, является максимумом, а другое — минимумом.

Убедитесь, что эти векторы \mathbf{x} удовлетворяют необходимым и достаточным условиям соответственно для максимума и для минимума.

2.32. Известно, что следующая задача:

$$\text{минимизировать } f(\mathbf{x}) = 100(x_2 - x_1^2) + (1 - x_1)^2$$

при ограничении

$$x_1^2 + x_2^2 \leqslant 2$$

имеет локальный минимум в точке $\mathbf{x}^* = [1 \quad 1]^T$. Убедитесь, что необходимые условия для локального оптимума удовлетворены. Является ли этот локальный оптимум также и глобальным?

2.33. Очевидно, что целевая функция

$$f(x) = |x^3|$$

имеет минимум в точке $x = 0$.

Можно ли к этой целевой функции применить критерии разд. 2.5?

2.34. Определите, является ли локальным минимумом седловая точка функции $f(x) = 1 - x_1^2 - 4x_1x_2 - x_2^2$, ограниченной цилиндром $x_1^2 + x_2^2 = 2$.

2.35. Для следующей задачи:

$$\text{минимизировать } f(x) = x_2^2$$

при ограничениях

$$g_1(x) = -x_1^3 + x_2^3 \geq 0,$$

$$g_2(x) = x_1^3 + x_2^3 \geq 0,$$

$$g_3(x) = x_1^2 + x_2^2 + 2x_2 \geq 0$$

проверьте, удовлетворяет ли предполагаемое решение $x = [0 \ 0]^T$ теоремам, приведенным в разд. 2.5.

2.36. Определите, является ли оптимальным (с точностью до двух значащих цифр) предполагаемое решение $x = [0,82 \ 0,43 \ 0,77]^T$ задачи

$$\text{минимизировать } f(x) = \frac{2}{x_1 + 0,5} + \frac{1}{x_2 + 0,2} + \frac{3}{x_3 + 0,5}$$

при ограничениях

$$4x_1 + 7x_2 + 3x_3 \leq 10,$$

$$3x_1 + 4x_2 + 5x_3 \leq 8,$$

$$x_1, x_2, x_3 \geq 0.$$

2.37. Определите, является ли оптимальным (с точностью до двух значащих цифр) предполагаемое решение $x = [0,75 \ 0,75 \ 0,69]^T$ задачи

$$\text{минимизировать } f(x) = \frac{2}{x_1 + 0,5} + \frac{1}{x_2 + 0,2} + \frac{3}{x_3 + 0,5}$$

при ограничениях

$$3x_1 + 4x_2 + 5x_3 = 8,$$

$$x_1, x_2, x_3 \geq 0.$$

2.38. Следующая задача:

$$\text{минимизировать } f(x) = 4x_1 - x_2^2 - 12$$

при ограничениях

$$25 - x_1^2 - x_2^2 = 0,$$

$$10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$(x_1 - 3)^2 + (x_2 - 1)^2 \geq 0,$$

$$x_1, x_2 \geq 0$$

была численно решена, и получен ответ, который, какказалось, является искомым решением, а именно вектор $\mathbf{x} = [1,000 \ 4,900]^T$. Покажите, удовлетворяет ли найденный вектор необходимым и достаточным условиям того, что он является решением данной задачи. Приведите все вычисления.

2.39. Минимизируйте $f(\mathbf{x}) = x^2 - x$, начиная из точки $x = 3$, с помощью следующих методов одномерного поиска:

- (а) метода золотого сечения;
- (б) метода ДСК — Паузла;
- (в) метода ДСК.

Пусть $|\Delta x^{(0)}| = 0,1$. Выполните число вычислений функции, достаточное, чтобы получить $|\Delta x^{(k)}| < 10^{-3}$. Постройте график зависимости $[f(x^{(k+1)}) - f(x^{(k)})]$ от последовательности номеров k для каждого метода. Для этого вы можете методом ДСК определить интервал, в котором находится минимум, при использовании процедуры поиска методом золотого сечения.

2.40. Найдите минимум $f(\mathbf{x})$ в направлении наискорейшего спуска, начиная из точки $\mathbf{x}^{(0)} = [2 \ 2]^T$ для $f(\mathbf{x}) = x_1^2 + 25x_2^2$. [Указание: пусть $\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda \nabla f(\mathbf{x}^{(0)})$. Минимизируйте $f(\mathbf{x}^{(1)})$ по λ .]

2.41. Пусть требуется найти действительные корни уравнения

$$f_1(\mathbf{x}) = 3000 - 100x^2 - 4x^5 - 6x^6 = 0. \quad (\text{а})$$

Как можно привести эту задачу к виду задачи оптимизации? Предположим далее, что требуется найти действительные корни одновременно уравнений (а) и (б).

$$f_2(\mathbf{x}) = 6x^4 + 4x^3 + 100x^2 - 3000 = 0. \quad (\text{б})$$

Как можно привести эту последнюю задачу к задаче оптимизации?

2.42. Коэффициенты в эмпирической модели $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$ должны оцениваться по методу наименьших квадратов, т. е. путем минимизации суммы квадратов разностей между предсказанными значениями y (используя оценки коэффициентов) и экспериментальными значениями y . Сформулируйте эту задачу как задачу оптимизации.

2.43. Покажите, что целевая функция

$$f(\mathbf{x}) = 55,84 + [7,31 \quad 26,65] \mathbf{x} + \mathbf{x}^T \begin{bmatrix} -3,03 & 1,345 \\ 1,345 & -6,96 \end{bmatrix} \mathbf{x},$$

где $\mathbf{x}^T = [x_1 \quad x_2]^T$, строго выпукла. Покажите, что $-g(\mathbf{x})$ строго вогнута, если

$$g(\mathbf{x}) = 85,72 + 21,85x_1 + 8,59x_2 - 9,20x_1^2 - 5,18x_2^2 - 6,26x_1x_2.$$

Затем покажите, что экстремум, полученный при минимизации $f(\mathbf{x})$, является глобальным экстремумом.

ЛИТЕРАТУРА

1. Dantzig G. B., Linear Programming and Extension, Princeton Univ. Press, Princeton, N. J., 1963.
2. Wilde D. J., Beightler C. S., Foundations of Optimization, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1967.
3. Ponstein J., *J. SIAM Rev.*, 9, 115 (1967).
4. Kuhn W. W., Tucker A. W., Nonlinear Programming, Proc. 2nd Berkeley Symp. on Mathematical Statistics and Programming, Univ. of California Press, Berkeley, 1951, pp. 481—493.
5. Fiacco A. V., McCormick G. P., Nonlinear Programming, Wiley, Inc., N. Y., 1968.
6. McCormick G. P., *SIAM J. Appl. Math.*, 15, 641 (1967).
7. Pennisi L., *Trans. Am. Math. Soc.*, 74, 177 (1953).
8. Wilde D. J., Optimum Seeking Methods, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1964.
9. Coggins G. F., Univariate Search Methods, Imperial Chemical Industries Ltd., Central Instr. Lab. Res. Note 64/11, 1964.
10. Box M. J., Davies D., Swann W. H., Nonlinear Optimization Techniques, Chemical Industries Monograph 5, Oliver and Boyd, Edinburgh, 1970.
11. ICI Note 64/3, 1964.
12. Powell M. J. D., *Computer J.*, 7, 155 (1964); см. также Walsh J., ed., Numerical Analysis, Academic Press Inc., London, 1966.
13. Gilbert E. G., *Simulation*, 10, 350 (1967).

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

- Fiacco A. V., Sequential Unconstrained Minimization Methods for Nonlinear Programming, Ph. D. Dissertation, Northwestern Univ., Evanston, Ill., 1967.
- Fiacco A. V., McCormick G. P., Sequential Unconstrained Minimization Techniques for Nonlinear Programming, Wiley, N. Y., 1968.
- Kunzi H. P., Zum heutigen stand der nichtlinearen Optimierungs Theorie, Unternehmersforsch, 12, 1 (1968).
- Kunzi H. P., Krelle W., Oettli W., Nonlinear Programming, Blaisdell, Waltham, Mass., 1966.

- Kunzi H. P., Tzscharc H. G., Zehnder C. A., Numerical Methods of Mathematical Optimization, Academic Press, N. Y., 1968.
- Mangasarian O. L., Nonlinear Programming, McGraw-Hill, N. Y., 1969.
- Mangasarian O. L., Fromovitz, The Fritz-John Necessary Optimality Conditions in the Presence of Equality and Inequality Constraints, *J. Math. Anal. Appl.*, 17, 34 (1967)
- Schechter R. S., Beveridge G. S. G., Sufficiency Conditions in Constrained Variations, *Ind. Eng. Chem. Fundamentals*, 5, 571 (1966).
- Schechter R. S., Beveridge G. S. G., Optimization: Theory and Practice, McGraw-Hill, N. Y., 1970.
- Wilde D. J., Beightler C. S., Foundations of Optimization, Prentice-Hall, Englewood Cliffs, N. J., 1967.
- Zangwill W. I., Nonlinear Programming: A Unified Approach, Prentice-Hall, Englewood Cliffs, N. J., 1969.

Ч а с т ь II

МЕТОДЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ БЕЗ ОГРАНИЧЕНИЙ



В приведенной выше табл. 2.7.1 при классификации задач нелинейного программирования по характерным чертам метода решения выделяются методы, использующие при оптимизации производные, и методы, которые не используют производные,— обычно известные как методы поиска. Ниже в гл. 3 описываются методы первой группы, а в гл. 4 рассматриваются методы второй группы, причем обе для задач без ограничений. Затем в гл. 5 сравнивается эффективность различных методов. Конечно, это не слишком четкая классификация, поскольку некоторые методы являются смешанными, например определение составляющих градиента целевой функции по разностным схемам или минимизация в направлении градиента с помощью методов поиска. Здесь мы имеем возможность рассмотреть только некоторые из многих методов решения задачи нелинейного программирования без ограничений; выбраны были те из них, которые, с одной стороны, достаточно эффективны, а с другой стороны, связаны с алгоритмами, обсуждаемыми в последующих главах, посвященных нелинейному программированию при ограничениях. Некоторые важные алгоритмы нелинейного программирования при ограничениях требуют использования эффективной процедуры минимизации без ограничений.

Глава 3

МЕТОДЫ МИНИМИЗАЦИИ БЕЗ ОГРАНИЧЕНИЙ, ИСПОЛЬЗУЮЩИЕ ПРОИЗВОДНЫЕ



Общая задача нелинейного программирования без ограничений сводится к следующей:

$$\text{минимизировать } f(\mathbf{x}), \quad \mathbf{x} \in E^n, \quad (3.0.1)$$

где $f(\mathbf{x})$ является целевой функцией. Руководствуясь замечаниями разд. 2.5.1, при решении этой задачи мы используем методы минимизации, которые приводят к стационарной точке $f(\mathbf{x})$, определяемой уравнением $\nabla f(\mathbf{x}^*) = 0$. В этой главе рассматривается вопрос о том, как решить задачу (3.0.1) с помощью алгоритмов, использующих первую и вторую частные производные $f(\mathbf{x})$. Сначала описывается поиск методом наискорейшего спуска, затем излагаются метод Ньютона, метод сопряженных направлений и, наконец, некоторые из методов, аппроксимирующих (путем использования только первых производных) направления, определяемые методом Ньютона.

3.1. ГРАДИЕНТНЫЕ МЕТОДЫ

В этом разделе кратко излагается стратегия градиентных (наискорейшего спуска) методов оптимизации без ограничений; в вычислительном аспекте эти методы используют только первые производные целевой функции. На k -м этапе переход из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$ описывается следующим соотношением:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \Delta \mathbf{x}^{(k)} = \mathbf{x}^{(k)} + \lambda^{(k)} \hat{\mathbf{s}}^{(k)} = \mathbf{x}^{(k)} + \lambda^{*(k)} \mathbf{s}^{(k)}, \quad (3.1.1)$$

где

$\Delta \mathbf{x}^{(k)}$ — вектор перехода из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$;

$\hat{\mathbf{s}}^{(k)}$ — единичный вектор в направлении $\Delta \mathbf{x}^{(k)}$;

$\mathbf{s}^{(k)}$ — любой вектор в направлении $\Delta \mathbf{x}^{(k)}$;

$\lambda^{(k)}, \lambda^{*(k)}$ — скаляры, определяемые соотношениями

$$\Delta \mathbf{x}^{(k)} = \lambda^{(k)} \hat{\mathbf{s}}^{(k)} = \lambda^{*(k)} \mathbf{s}^{(k)}.$$

3.1.1. МЕТОД НАИСКОРЕЙШЕГО СПУСКА

Применение метода наискорейшего спуска для решения задачи минимизации без ограничений было рассмотрено еще известным французским математиком Коши. Как отмечалось в разд. 2.4.4, градиент целевой функции $f(\mathbf{x})$ в любой точке \mathbf{x} есть вектор в направлении наибольшего локального увеличения $f(\mathbf{x})$. Следовательно, нужно двигаться в направлении, противоположном градиенту $f(\mathbf{x})$, т. е. в направлении *наискорейшего спуска*, поскольку отрицательный градиент $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$ направлен в сторону наибольшего уменьшения $f(\mathbf{x})$ по всем компонентам \mathbf{x} и ортогонален линии уровня $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$. Введение направления, противоположного нормированному (единичному) градиенту $f(\mathbf{x})$, т. е. направления наискорейшего спуска, определяемого в точке $\mathbf{x}^{(k)}$ по формуле

$$\hat{\mathbf{s}}^{(k)} = -\frac{\nabla f(\mathbf{x}^{(k)})}{\|\nabla f(\mathbf{x}^{(k)})\|}, \quad (3.1.2)$$

в (3.1.1) дает следующую формулу перехода из $\mathbf{x}^{(k)}$ в $\mathbf{x}^{(k+1)}$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \frac{\lambda^{(k)} \nabla f(\mathbf{x}^{(k)})}{\|\nabla f(\mathbf{x}^{(k)})\|} = \mathbf{x}^{(k)} - \lambda^{*(k)} \nabla f(\mathbf{x}^{(k)}). \quad (3.1.3)$$

Отрицательный градиент дает только направление оптимизации, но не величину шага. При этом можно использовать различные процедуры метода наискорейшего спуска в зависимости от выбора λ и определения выражения $\|\nabla f(\mathbf{x}^{(k)})\|$. Поскольку один шаг в направлении наискорейшего спуска в общем случае не приводит в точку минимума $f(\mathbf{x})$, формула (3.1.3) должна применяться несколько раз, до тех пор пока минимум не будет достигнут. В точке минимума все составляющие вектора градиента равны нулю. В случае когда целевая функция $f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{A} \mathbf{x}$, выражение $\nabla f(\mathbf{x}^{(k)}) = \mathbf{A} \mathbf{x}^{(k)}$ может быть непосредственно подставлено в (3.1.3).

Процедура строгого наискорейшего спуска может закончиться в стационарной точке (в которой составляющие градиента $f(\mathbf{x})$ равны нулю) различного типа. Обычно бывает необходимо определить, является ли данная точка точкой локального минимума (т. е. решением) или седловой точкой. Если это седловая точка, то следует применить какой-либо неградиентный метод, чтобы выйти из нее, после чего минимизация может продолжаться, как и ранее. Тип стационарной точки может быть проверен путем исследования матрицы Гессе (если ее возможно получить) целевой функции, взятой в данной стационарной точке. Если эта матрица не является положительно определенной, то стационарная точка — седловая. В качестве критерия окончания последовательной процедуры при движении в направлении наискорейшего спуска применяются

различные правила, основанные либо на значении $f(\mathbf{x})$ и величинах \mathbf{x} , λ , $\nabla f(\mathbf{x})$, либо на некоторой их комбинации, а также на соответствующих значениях этих величин на предыдущих шагах. Успех того или иного метода в смысле эффективности сходимости к локальному минимуму зависит от этих правил, а также и от самой задачи.

При выборе размера шага применяются два общих метода, хотя могли бы быть рассмотрены и многие другие возможности. В первом методе при переходе из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$ целевая функция минимизируется по λ , в другом методе величина λ выбирается фиксированной или меняется от шага к шагу.

Рассмотрим сначала случай, когда λ выбирается так, чтобы минимизировать $f(\mathbf{x})$ в заданном направлении. При этом на каждом шаге старая информация отбрасывается и заменяется новой информацией, так что никакого ускорения оптимизации осуществить нельзя. Сходимость метода наискорейшего спуска в такой постановке может быть доказана [1]. Можно показать, что для выпуклой целевой функции, имеющей производные до третьего порядка (и для некоторых других функций при еще более слабых ограничениях), этот метод в пределе сходится при $k \rightarrow \infty$. Тем не менее упомянутое свойство метода наискорейшего спуска не является большим его достоинством на практике, поскольку скорость сходимости может быть слишком медленной, как это оказалось в многочисленных экспериментах, а также предсказывается теорией [2].

При определении точки $\mathbf{x}^{(k+1)}$ на основе формулы (3.1.3) $f(\mathbf{x})$ может быть формально минимизирована путем вычисления λ из уравнения

$$\frac{df(\mathbf{x}^{(k)} + \lambda \hat{\mathbf{s}}^{(k)})}{d\lambda} = 0.$$

В качестве конкретного примера предположим, что $f(\mathbf{x})$ — квадратичная функция [имея это в виду, подставим в уравнение (2.4.5) $\hat{\lambda} \hat{\mathbf{s}}^{(k)}$ вместо $(\mathbf{x} - \mathbf{x}^{(k)})$]. Тогда

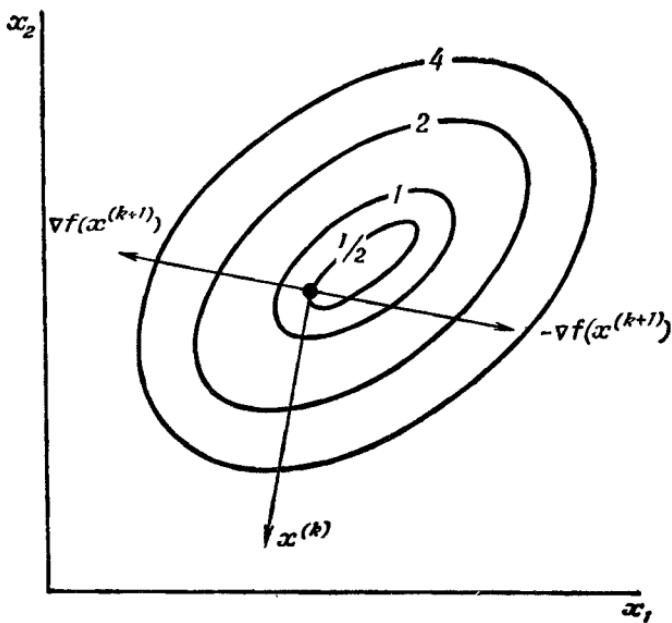
$$\frac{df(\mathbf{x}^{(k)} + \lambda \hat{\mathbf{s}}^{(k)})}{d\lambda} = 0 = \nabla^T f(\mathbf{x}^{(k)}) \hat{\mathbf{s}}^{(k)} + (\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}, \quad (3.1.4)$$

что дает следующее выражение для $\lambda^{(k)}$:

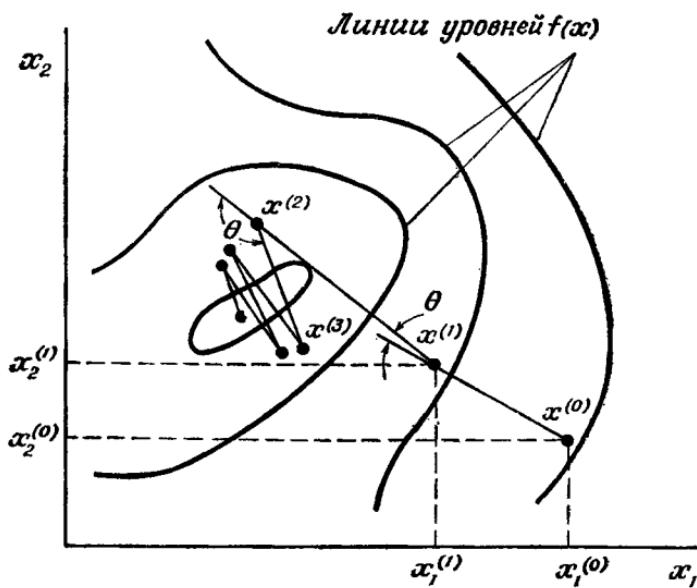
$$\lambda^{(k)} = - \frac{\nabla^T f(\mathbf{x}^{(k)}) \hat{\mathbf{s}}^{(k)}}{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}}. \quad (3.1.5)$$

Функцию $f(\mathbf{x})$ можно минимизировать также с помощью одного из методов одномерного численного поиска, изложенных в разд. 2.6.

Интересной чертой процедуры минимизации в случае квадратичной целевой функции является то, что $\nabla f(\mathbf{x}^{(k+1)})$ ортогонален $\hat{\mathbf{s}}^{(k)}$.



Ф и г. 3.1.1. Иллюстрация того, что для квадратичной целевой функции $(s^{(k)})^T \nabla f(x^{k+1}) = 0$, если $f(x)$ минимизируется в направлении $s^{(k)}$.



Ф и г. 3.1.2. Зигзагообразная траектория оптимизации при использовании метода наискорейшего спуска.

Это можно показать следующим образом. Отметим, что если

$$f(\mathbf{x}) = \mathbf{a} + \mathbf{x}^T \mathbf{b} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x},$$

то градиент $f(\mathbf{x})$ равен

$$\nabla f(\mathbf{x}) = \mathbf{b} + \mathbf{H} \mathbf{x}, \quad (3.1.6)$$

так что

$$\nabla f(\mathbf{x}^{(0)}) = \mathbf{b} + \mathbf{H} \mathbf{x}^{(0)},$$

· · · · ·

$$\nabla f(\mathbf{x}^{(k)}) = \mathbf{b} + \mathbf{H} \mathbf{x}^{(k)}.$$

Подстановка выражения для $\nabla f(\mathbf{x}^{(k)})$ в (3.1.4) приводит к уравнению

$$(\mathbf{b} + \mathbf{H} \mathbf{x}^{(k)})^T \hat{s}^{(k)} + (\hat{s}^{(k)})^T \mathbf{H} \lambda^{(k)} \hat{s}^{(k)} = 0.$$

После подстановки $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ вместо $\lambda^{(k)} \hat{s}^{(k)}$ и преобразования получаем

$$(\hat{s}^{(k)})^T (\mathbf{b} + \mathbf{H} \mathbf{x}^{(k)}) + (\hat{s}^{(k)})^T \mathbf{H} (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) = 0,$$

или

$$(\hat{s}^{(k)})^T (\mathbf{b} + \mathbf{H} \mathbf{x}^{(k+1)}) = (\hat{s}^{(k)})^T \nabla f(\mathbf{x}^{(k+1)}) = 0. \quad (3.1.7)$$

Другими словами, градиент в $\mathbf{x}^{(k+1)}$ ортогонален предыдущему направлению поиска $\hat{s}^{(k)}$ (фиг. 3.1.1). Если в методе наискорейшего спуска выбирается фиксированное значение скаляра λ или величина λ переменная, то она должна тщательно контролироваться во избежание как неожиданного роста $f(\mathbf{x})$, так и чрезмерного числа шагов, необходимого для достижения решения. Первое произойдет, если λ слишком велико, а второе, если λ очень мало или если λ настолько велико, что приводит к колебаниям около точки минимума (фиг. 3.1.2). Таким образом, величина λ должна уменьшаться при приближении к точке минимума. Один из возможных методов контроля λ предполагает установление некоторого критерия для λ , основанного на использовании угла θ между последовательными векторами шагов в процессе минимизации. Например, если этот угол становится меньше, чем некоторая заданная величина, то величина λ должна быть умножена на некоторую заранее определенную константу α ; если угол становится больше, то λ нужно разделить на α .

Пример 3.1.1. Метод наискорейшего спуска

В этом примере описывается несколько циклов метода наискорейшего спуска с целью иллюстрации методики решения задач минимизации.

Рассмотрим задачу

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + 25x_2^2.$$

Возьмем сначала фиксированную длину шага λ , начальное значение которой равно единице. На каждом этапе нам понадобятся значения следующих функций:

$$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_1} = 2x_1^{(k)}, \quad \frac{\partial f(\mathbf{x}^{(k)})}{\partial x_2} = 50x_2^{(k)},$$

$$\|\nabla f(\mathbf{x}^{(k)})\| = \sqrt{\left(\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_1}\right)^2 + \left(\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_2}\right)^2}.$$

Начиная с точки $\mathbf{x}^{(0)} = [2 \ 2]^T$, поиск минимума осуществляем следующими этапами:

Этап	x_1	x_2	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_1}$	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_2}$	$\ \nabla f(\mathbf{x}^{(k)})\ $	Величина шага при переходе к следующему этапу	
						Δx_1	Δx_2
0	2	2	4	100	~100	-0,04	-1,00
1	1,96	1,00	3,92	50	50,1	-0,078	-1,00
2	1,88	0	3,76	0	3,76	-1,00	0
3	0,88	0					

На фиг. П.3.1.1, а можно проследить траекторию поиска.

Для того чтобы метод сходился, λ обычно нужно уменьшать, иначе при подходе к минимуму возникнут колебания («назад—вперед»). Заметим, что в точке минимума $\mathbf{x} = [0 \ 0]^T$ $\nabla f(\mathbf{x}) = 0$.

Ниже приводятся результаты трех соответствующих этапов вычисления, в которых вместо использования фиксированного λ ищется минимум $f(\mathbf{x})$ в направлении наискорейшего спуска:

Этап k	$\lambda^{(k)}$	x_1	x_2	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_1}$	$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_2}$	$f(\mathbf{x}^{(k)})$
0		2	2	4	100	104
1	2,003	1,92	-0,003	3,84	-0,15	3,19
2	1,850	0,070	0,070	0,14	3,50	0,13
3	0,070	0,070	-0,000			

Следует обратить внимание на то, что на фиг. П.3.1.1, а градиент $f(\mathbf{x})$ в начале поиска не направлен в точку минимума, поскольку коэффициенты при x_1 и x_2 различны.

Путем замены переменной

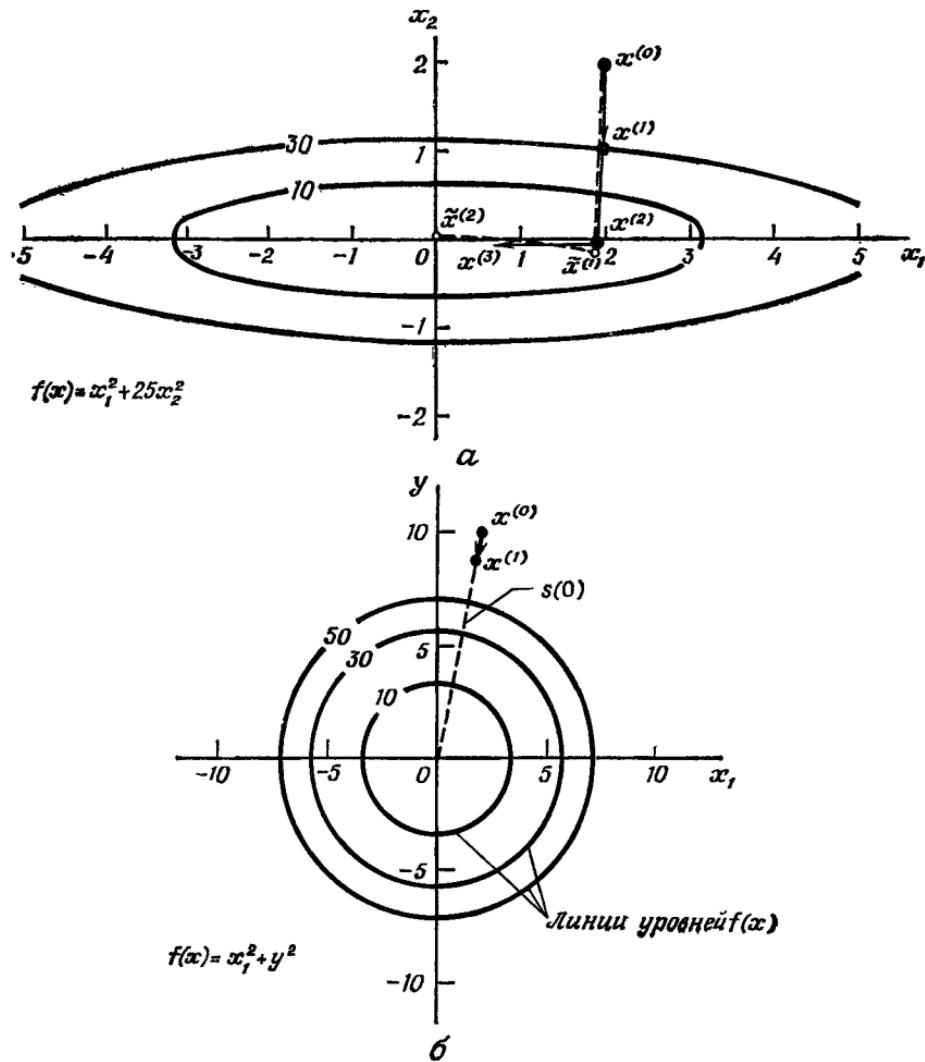
$$y = 5x_2$$

минимизируемая функция принимает вид

$$f(x) = x_1^2 + y^2,$$

и вектор градиента в точке $x_1 = 2$, $y = 5x_2 = 10$ действительно направлен в точку минимума, поскольку коэффициенты при x_1 и y теперь одни и те же; см. фиг. П.3.1.1, б.

Если нелинейная целевая функция слишком сложна, чтобы ее можно было продифференцировать аналитически, то составля-



Фиг. П.3.1.1.

— фиксированное $\lambda(x^{(k)})$; — — — $f(x)$, минимизируемая на каждом шаге.

ющие градиента, являющиеся частными производными по оптимизируемым переменным, аппроксимируются разностными соотношениями. Например, для функции двух переменных разностные формулы имеют вид (если разности берутся вперед)

$$\frac{\partial f(x^{(k)})}{\partial x_1} \approx \frac{f((x_1^{(k)} + \delta_1), x_2^{(k)}) - f(x_1^{(k)}, x_2^{(k)})}{\delta_1}, \quad (a)$$

$$\frac{\partial f(x^{(k)})}{\partial x_2} \approx \frac{f(x_1^{(k)}, (x_2^{(k)} + \delta_2)) - f(x_1^{(k)}, x_2^{(k)})}{\delta_2}, \quad (b)$$

где δ_i — некоторые малые отклонения. При этом необходимо вычислить целевую функцию лишь в трех точках, а именно в точках $(x_1^{(k)}, x_2^{(k)})$, $[(x_1^{(k)} + \delta_1), x_2^{(k)}]$ и $[x_1^{(k)}, (x_2^{(k)} + \delta_2)]$ на каждый цикл k . Величины δ_i в общем случае выбираются так, чтобы ошибка в аппроксимации производной не превышала разумного уровня. Поскольку градиент не обязательно направлен в сторону минимума в $x^{(k)}$ и градиент заново вычисляется на каждом этапе, величина этой ошибки существенна главным образом в окрестности минимума, где $\nabla f(x) \rightarrow 0$.

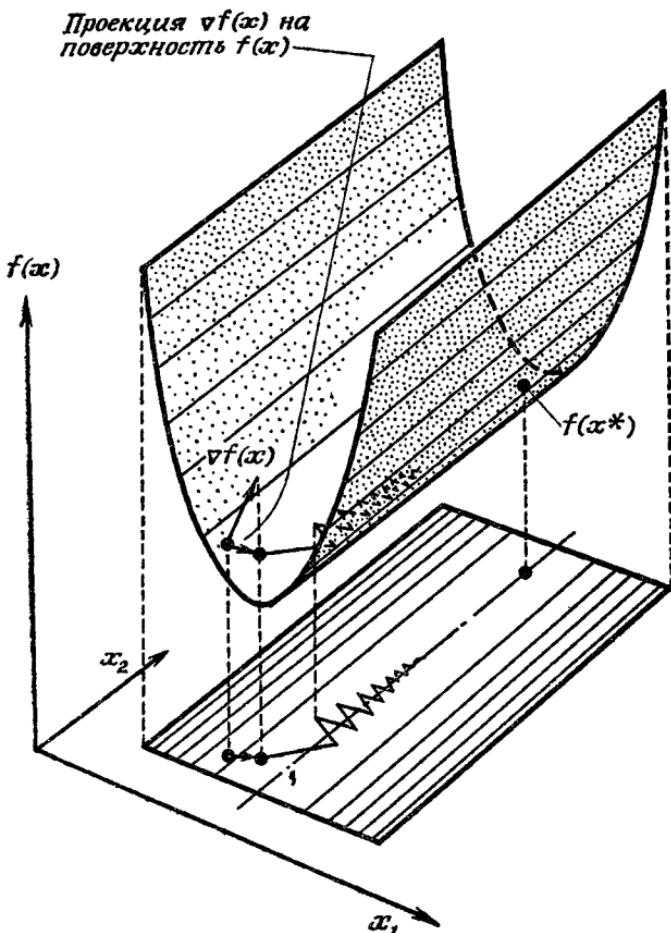
Для иллюстрации ниже приводятся компоненты градиента для данного примера, вычисленные по разностным формулам (a) и (b) при $\delta_1 = \delta_2 = 0,05$:

Точка				$\frac{\partial f(x^{(k)})}{\partial x_1}$		$\frac{\partial f(x^{(k)})}{\partial x_2}$	
x_1	x_2	$x_1 + \delta_1$	$x_2 + \delta_2$	приближенно	точно	приближенно	точно
2 0,01	2 0,01	2,05 0,06	2,05 0,06	4,05 0,070	4,00 0,02	101,25 1,80	100,0 0,50

Из таблицы видно, что на некотором расстоянии от минимума разностная аппроксимация производных вполне приемлема, однако вблизи минимума она недостаточно удовлетворительна. Конечно, можно уменьшать величину δ_i в процессе поиска или использовать более подходящие аппроксимирующие формулы для производных, но здесь мы не будем обсуждать эти вопросы, поскольку имеются другие методы оптимизации без ограничений, более предпочтительные, чем метод наискорейшего спуска.

Основной трудностью при использовании метода наискорейшего спуска, как уже видно из приведенного примера, является его зависимость от выбора масштаба оптимизируемых переменных. Если гиперпространство очень вытянуто, так что образует

«хребет» или «овраг» (отношение максимального значения $H(x)$ к минимальному в каждой точке велико), процедура наискорейшего спуска сходится слишком медленно, чтобы быть эффективной, или может вообще не сойтись за разумное время. На фиг. 3.1.3 иллюстрируется



Фиг. 3.1.3. Типичная зигзагообразная траектория оптимизации при использовании метода наискорейшего спуска в узкой впадине.

этая трудность; направление наискорейшего спуска оказывается почти ортогональным наилучшему направлению достижения минимума $f(x)$. Одним из выходов в этой ситуации является использование информации второго порядка (информации, даваемой вторыми частными производными целевой функции по независимым переменным или их приближениями). Этот вопрос будет рассматриваться в разд. 3.2 и 3.4. Другим подходом, с которого мы и начнем, является изменение масштаба независимых переменных в целевой функции.

3.1.2. МАСШТАБИРОВАНИЕ НЕЗАВИСИМЫХ ПЕРЕМЕННЫХ

В примере 3.1.1 было показано, что с изменением масштаба переменных (координат) меняется и направление наискорейшего спуска. Таким образом, произвольное изменение единиц измерения независимых переменных изменяет направление наискорейшего спуска и влияет на эффективность градиентной минимизации. Избежать эту трудность можно путем введения безразмерных переменных, полученных делением каждой переменной на интервал ее изменения, как, например,

$$\hat{T} = \frac{T - T_1}{T_2 - T_1}, \quad \hat{p} = \frac{p - p_1}{p_2 - p_1},$$

где в данном случае T обозначает температуру, а p — давление. Но даже это преобразование искусственно подразумевает, что расстояние $p_2 - p_1$ то же, что и $T_2 - T_1$.

Если в n -мерной задаче принять следующее общее определение расстояния в E^n :

$$ds^r = \sqrt{\frac{|dx_1|^r + |dx_2|^r + \cdots + |dx_n|^r}{n}},$$

где r — индекс метрики, то можно показать [3], что отношение изменений любых двух немасштабированных координат (при наискорейшем спуске) дается формулой

$$\frac{dx_i}{dx_j} = \pm \left| \frac{\partial f(\mathbf{x})/\partial x_i}{\partial f(\mathbf{x})/\partial x_j} \right|^{1/(r-1)}. \quad (3.1.8)$$

Знак плюс используется, когда частные производные одного знака, а минус — когда производные разных знаков. В случае обычной метрики ($r = 2$) получаем

$$\frac{dx_i}{dx_j} = \pm \frac{\partial f(\mathbf{x})/\partial x_i}{\partial f(\mathbf{x})/\partial x_j},$$

что соответствует направлению, определяемому градиентом.

Предположим далее, что для оптимизируемых переменных производится изменение масштаба (единиц измерения), так что

$$\tilde{x}_i = \psi(x_i),$$

где \tilde{x}_i — масштабированная переменная. Обычно такая замена линейна, как, например, в случае перевода температуры из шкалы в градусах Цельсия в шкалу в градусах Фаренгейта:

$$T(\text{в } ^\circ\text{F}) = 1,8T(\text{в } ^\circ\text{C}) + 32,$$

при этом величины $\mu_i = \partial\psi(x_i)/\partial x_i$ — константы. Тогда можно показать, что отношение изменений любых двух масштабированных

переменных будет следующим:

$$\frac{d\tilde{x}_i}{d\tilde{x}_j} = \pm \left| \frac{\partial f(x)/\partial \tilde{x}_i}{\partial f(x)/\partial \tilde{x}_j} \right|^{1/(r-1)} \left| \frac{\mu_i}{\mu_j} \right|^{r/(r-1)}. \quad (3.1.9)$$

Влияние выбора масштаба определяется отношением $|\mu_i/\mu_j|^{r/(r-1)}$. В случае $r = 2$

$$\frac{d\tilde{x}_i}{d\tilde{x}_j} = \frac{\partial f(x)/\partial \tilde{x}_i}{\partial f(x)/\partial \tilde{x}_j} \left(\frac{\mu_i}{\mu_j} \right)^2,$$

откуда можно заключить, что относительное изменение переменных x_i и x_j зависит от масштабных множителей. Следовательно, единственного направления наискорейшего спуска не существует.

Некоторые интересные выводы могут быть получены при таких индексах метрики, как $r = 0$, $r = 1$ и $r = \infty$. В случае $r = 0$, что соответствует $ds = (|dx_1| |dx_2| \dots |dx_n|)^{-n}$, отношение dx_i/dx_j одно и то же как для немасштабированных, так и для масштабированных переменных. Бокс и Вильсон [4] осуществили такое же масштабирование путем приравнивания скорости изменения целевой функции по одной из безразмерных переменных к скорости изменения по другой безразмерной переменной.

В случае $r = 1$ уравнение (3.1.8) приводит к

$$\frac{dx_i}{dx_j} = \pm \infty,$$

откуда следует, что на каждом этапе оптимизации соответствующий шаг осуществляется лишь в одном координатном направлении, а именно в направлении, соответствующем наибольшей частной производной. Это соответствует попеременному поиску по одной координате, причем выбор шага определяется величиной $\partial f / \partial x_i$.

И наконец, $r = \pm \infty$ соответствует $ds = \pm dx_k$, где $|dx_k| = \max_j (|dx_j|)$ при $r = +\infty$ и $|dx_k| = \min_j (|dx_j|)$ при $r = -\infty$.

При этом из уравнения (3.1.8) следует

$$\frac{dx_i}{dx_j} = \pm 1,$$

т. е. на каждом последовательном этапе оптимизации все координаты изменяются на одну и ту же величину. Очевидно, что, применяя подходящий индекс метрики и проводя масштабирование, можно получить широкий выбор направлений для использования их в качестве направления наискорейшего спуска. Проблема выбора индекса метрики в задаче оптимизации еще не изучена, но, видимо, этот подход может послужить основой улучшения алгоритмов.

3.2. МЕТОД ВТОРЫХ ПРОИЗВОДНЫХ (МЕТОД НЬЮТОНА) И СВЯЗАННЫЕ С НИМ АЛГОРИТМЫ

Направление поиска, соответствующее наискорейшему спуску, можно интерпретировать как следствие линейной аппроксимации целевой функции (фиг. 3.2.1, a). С другой стороны, методы вторых производных, среди которых лучше всего известен метод Ньютона¹⁾, возникли из квадратичной аппроксимации $f(x)$, определяемой уравнением (2.4.5). Они используют информацию второго порядка, содержащуюся во вторых частных производных целевой функции $f(x)$ по независимым переменным.

3.2.1. МЕТОД НЬЮТОНА

Направление поиска s в методе Ньютона выбирается следующим образом. Если $x - x^{(k)}$ в уравнении (2.4.5) заменить на величину $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$, определенную из уравнения (3.1.1), то квадратичная аппроксимация функции $f(x)$ через переменные $\Delta x^{(k)}$ представляется следующим образом:

$$f(x^{(k+1)}) = f(x^{(k)}) + \nabla^T f(x^{(k)}) \Delta x^{(k)} + \frac{1}{2} (\Delta x^{(k)})^T \nabla^2 f(x^{(k)}) \Delta x^{(k)}. \quad (3.2.1)$$

Минимум функции $f(x)$ в направлении $\Delta x^{(k)}$ определяется дифференцированием $f(x)$ по каждой из компонент Δx и приравниванием нулю полученных выражений. Последнее приводит к

$$\Delta x^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)}), \quad (3.2.2)$$

где $[\nabla^2 f(x^{(k)})]^{-1}$ — матрица, обратная матрице Гессе $H(x^{(k)})$, определенной в разд. 2.4 (матрица вторых частных производных $f(x)$ по x , взятая в точке $x^{(k)}$).

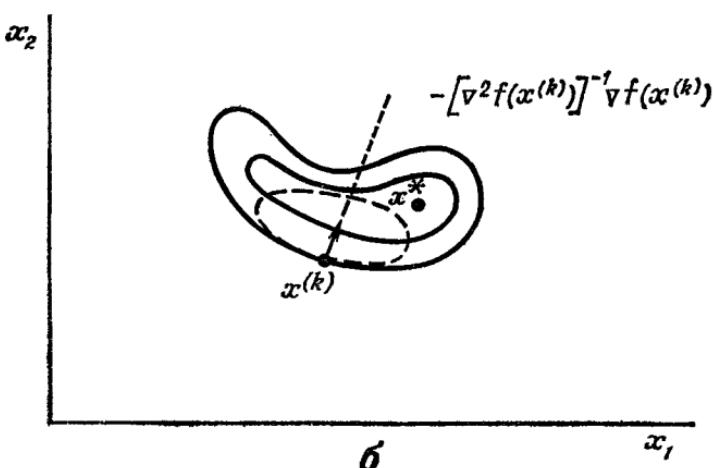
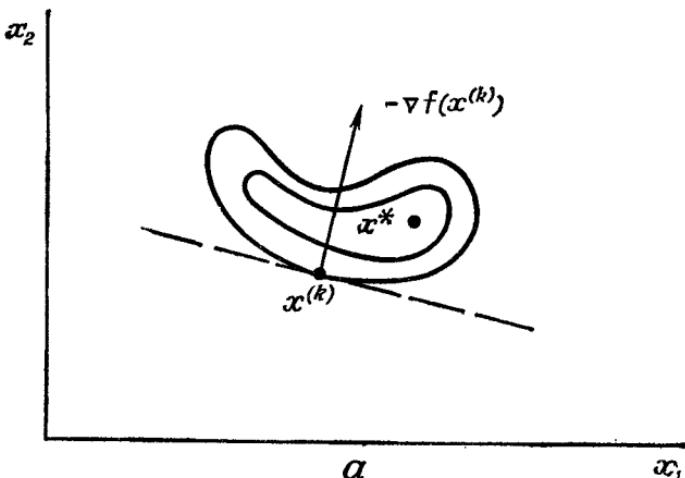
Подстановка выражения (3.2.2) в уравнение (3.1.1) определяет переход из $x^{(k)}$ в $x^{(k+1)}$ по методу Ньютона:

$$x^{(k+1)} = x^{(k)} - [\nabla^2 f(x^{(k)})]^{-1} \nabla f(x^{(k)}). \quad (3.2.3)$$

Заметим, что здесь и направление и величина шага точно определены. Если $f(x)$ — квадратичная функция, то для достижения минимума $f(x)$ достаточно только одного шага. Но в случае общей нелинейной целевой функции минимума $f(x)$ нельзя достичь за один шаг, поэтому уравнение (3.2.3) обычно приводят к виду (3.1.3) путем введения специального параметра длины шага λ :

$$x^{(k+1)} = x^{(k)} - \lambda^{(k)} \frac{[\nabla^2 f(x^{(k)})]^{-1} [\nabla f(x^{(k)})]}{\|[\nabla^2 f(x^{(k)})]^{-1} [\nabla f(x^{(k)})]\|}. \quad (3.2.4)$$

¹⁾ Название объясняется тем, что решение системы уравнений $\nabla f(x) = 0$ методом Ньютона приводит к уравнению (3.2.2); иногда они называются *методами квазилинейаризации*.



Ф и г. 3.2.1. Сравнение метода наискорейшего спуска и метода Ньютона с точки зрения аппроксимаций целевой функции.

a — наискорейший спуск: аппроксимация первого порядка (линеаризация) функции $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$; *б* — метод Ньютона: аппроксимация второго порядка (квадратичная) функции $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$.

Отношение $\lambda^{(k)} / \|[\nabla^2 f(\mathbf{x}^{(k)})]^{-1} [\nabla f(\mathbf{x}^{(k)})]\|$ — просто некоторый скаляр $\lambda^{*(k)}$; поэтому уравнение (3.2.4) чаще записывают следующим образом:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda^{*(k)} \mathbf{H}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)}). \quad (3.2.4a)$$

Следует обратить внимание на то, что направление поиска теперь задается вектором $\mathbf{s}^{(k)} = -\mathbf{H}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})$. Уравнение (3.2.4a)

применяется итеративно, как и уравнение (3.1.3), пока не будет удовлетворен некоторый критерий окончания процесса.

Заметим, что уравнение (3.2.4а) включает в себя обращение матрицы, и необходимо соблюдать осторожность с тем, чтобы выбираемая процедура решения обеспечивала положительную определенность обратной матрицы (см. ниже). В этом отношении многие стандартные ЭВМ-программы для обращения матриц являются неудовлетворительными [5]. Заметим также, что этот метод требует вычисления значений аналитических вторых частных производных или их аппроксимаций, что может оказаться в некоторых случаях непрактичным. Критерий, гарантирующий сходимость метода Ньютона в предположении, что функция $f(x)$ дважды дифференцируема, заключается в том, что матрица, обратная матрице Гессе целевой функции, должна быть положительно определенной (см. приложение В) ¹⁾:

$$[\nabla^2 f(x^{(k)})]^{-1} \equiv H^{-1}(x^{(k)}) > 0. \quad (3.2.5)$$

Пример 3.2.1. Сравнение методов, основанных на использовании производных первого и второго порядков

Чтобы продемонстрировать использование уравнений (3.1.3) и (3.2.3), возьмем плохо масштабированную целевую функцию, рассмотренную Розенброком [6], две линии уровней которой ($f(x) = 8$ и $f(x) = 4$) приведены на фиг. П.3.2.1:

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \quad (a)$$

Геометрически $f(x)$ интерпретируется как медленно спадающий искривленный овраг с самой низкой точкой $x^* = [1 \ 1]^T$, где $f(x^*) = 0$.

Метод наискорейшего спуска

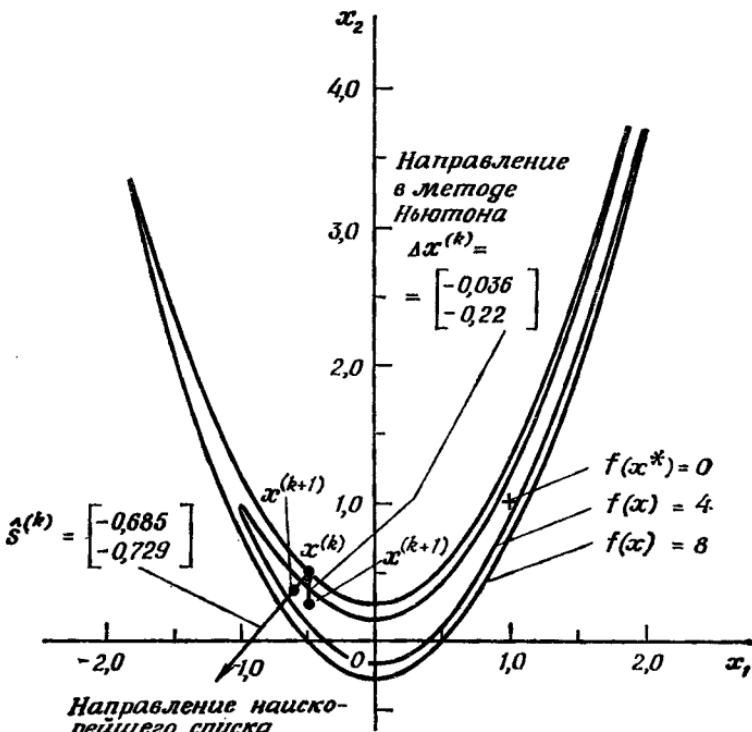
Рассмотрим точку $x^{(k)} = [-0,5 \ 0,5]^T$, в которой $f(x^{(k)}) = 8,5$. Нормированный градиент $f(x)$ в точке $x^{(k)} = [-0,5 \ 0,5]^T$ равен

$$\frac{1}{[(\partial f / \partial x_1)^2 + (\partial f / \partial x_2)^2]^{1/2}} \begin{bmatrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \end{bmatrix}_{x^{(k)}} = \frac{1}{68,6} \begin{bmatrix} 47 \\ 50 \end{bmatrix} = [0,685 \ 0,729]^T. \quad (6)$$

Вектор, противоположный вектору нормированного градиента в $x^{(k)}$, $\hat{s}^{(k)} = [-0,685 \ -0,729]^T$, как видно из фиг. П.3.2.1, указывает направление наискорейшего спуска и ортогонален уровню $f(x)$, проходящему через $x^{(k)}$.

¹⁾ Здесь мы оставили принятное автором краткое обозначение положительной определенности матрицы $A : A > 0$, хотя оно и не совсем точное. — Прим. перев.

Чтобы найти новый вектор $\mathbf{x}^{(k+1)}$, необходимо выбрать величину λ . Например, можно выбрать заранее определенное значение λ или найти то значение λ , при котором $f(\mathbf{x})$ достигает минимума в направлении



Фиг. П.3.2.1. Градиентный метод и метод вторых производных в случае функции Розенброка.

лении, задаваемом единичным вектором $\hat{s}^{(k)}$. В точке $\mathbf{x}^{(k)} = [-0,5, 0,5]^T$ уравнение (3.1.1) имеет вид

$$\mathbf{x}^{(k+1)} = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} - \lambda^{(k)} \begin{bmatrix} 0,685 \\ 0,729 \end{bmatrix}. \quad (\text{в})$$

Отсюда

$$f(\mathbf{x}^{(k+1)}) = f(\lambda) = 100[0,5 - 0,729\lambda - (0,5 + 0,685\lambda)^2] + (1,5 + 0,685\lambda)^2. \quad (\text{г})$$

Минимум $f(\lambda)$ по λ достигается в точке $\lambda = 0,164$. Подстановка $\lambda^{(k)} = 0,164$ в уравнение (в) дает новую точку $\mathbf{x}^{(k+1)} = [-0,612, 0,381]^T$, в которой $f(\mathbf{x}) = 2,6$ (см. фиг. П.3.2.1). Новый вектор $\hat{s}^{(k+1)}$ определяется уравнением (3.1.2) для точки $\mathbf{x}^{(k+1)}$, а затем находится и $\mathbf{x}^{(k+2)}$ в направлении $\hat{s}^{(k+1)}$ аналогично тому, как была найдена $\mathbf{x}^{(k+1)}$.

Эта итерационная процедура продолжается до тех пор, пока становится уже невозможным уменьшать величину $f(\mathbf{x})$ или пока не будет удовлетворен некоторый выбранный критерий окончания процесса. Интересно, что метод наискорейшего спуска, как видно из этого примера, не применим для продвижения вдоль изогнутого оврага функции Розенброка.

Метод Ньютона

Рассмотрим теперь метод Ньютона, начиная из точки $\mathbf{x}^{(k)} = [-0,5 \ 0,5]^T$. Тогда $\mathbf{x}^{(k+1)}$ определяется путем использования уравнения (3.2.3) или (3.2.4а) следующим образом:

$$\nabla^2 f(\mathbf{x}^{(k)}) = \begin{bmatrix} (-400x_2 + 1200x_1^2 + 2) & (-400x_1) \\ (-400x_1) & 200 \end{bmatrix}_{\mathbf{x}^{(k)}} = \begin{bmatrix} 102 & 200 \\ 200 & 200 \end{bmatrix},$$

$$\Delta \mathbf{x}^{(k)} = -[\nabla^2 f(\mathbf{x}^{(k)})]^{-1} \nabla f(\mathbf{x}^{(k)}) = \frac{1}{98} \begin{bmatrix} 1 & 1 \\ -1 & 0,51 \end{bmatrix} \begin{bmatrix} 47 \\ 50 \end{bmatrix} = \begin{bmatrix} -0,03 \\ -0,22 \end{bmatrix},$$

$$\Delta \mathbf{x}^{(k+1)} = \begin{bmatrix} -0,5 \\ 0,5 \end{bmatrix} + \begin{bmatrix} -0,03 \\ -0,22 \end{bmatrix} = \begin{bmatrix} -0,53 \\ 0,28 \end{bmatrix}.$$

В точке $\mathbf{x}^{(k+1)} = [-0,53 \ 0,28]^T$ значение $f(\mathbf{x})$ равно 2,33. Полученный вектор $\Delta \mathbf{x}^{(k)} = [-0,03 \ -0,22]^T$ изображен на фиг. П.3.2.1. Новый вектор $\Delta \mathbf{x}^{(k+1)}$ вычисляется в точке $\mathbf{x}^{(k+1)}$ с помощью уравнения (3.2.2), $\mathbf{x}^{(k+2)}$ определяется из уравнения (3.2.3).

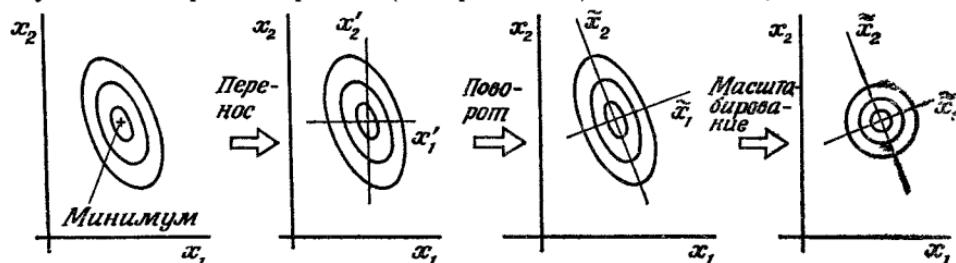
Альтернативная процедура заключается в вычислении $\mathbf{s}^{(k)} = \mathbf{H}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})$, как и в предыдущем методе, и в поиске некоторого λ в направлении $\mathbf{s}^{(k)}$, минимизирующего $f(\mathbf{x})$ в соответствии с уравнением (3.2.4а). В любом случае итерационная процедура повторяется, пока не будет удовлетворен определенный критерий окончания процесса или до тех пор, пока становится невозможным уменьшать величину $f(\mathbf{x})$. На практике, чтобы уменьшить время решения, элементы $\mathbf{H}(\mathbf{x})$ могут вычисляться не на каждом шаге.

Если используется уравнение (3.2.3), то метод Ньютона автоматически дает последовательность длин шагов, соответствующих расстоянию до минимума для квадратичных функций, аппроксимирующих функцию $f(\mathbf{x})$ в последовательных точках $\mathbf{x}^{(k)}$. Например,

квадратичная аппроксимация функции Розенброка в $\mathbf{x}^{(k)} = [-0,5 \ 0,5]^T$ представляет собой $q(\mathbf{x}^{(k)}) = -5,25 - 2x_1 + 50x_2 + 200x_1x_2 + 51x_1^2 + 100x_2^2$ и минимум $q(\mathbf{x}^{(k)})$ достигается в $\tilde{\mathbf{x}} = [-0,53 \ 0,28]^T$. С другой стороны, методы, основанные на производных первого порядка, линеаризуют $f(\mathbf{x})$ в $\mathbf{x}^{(k)}$, но линейная функция не имеет минимума (или максимума), не считая концов; следовательно, нужно выбирать определенную длину шага в направлении, ортогональном линеаризованной функции $f(\mathbf{x})$.

3.2.2. ГЕОМЕТРИЧЕСКАЯ ИНТЕРПРЕТАЦИЯ

Теперь дадим геометрическую интерпретацию локальной квадратичной аппроксимации целевой функции и проиллюстрируем важность требования положительной определенности $\mathbf{H}^{-1}(\mathbf{x})$. Функция второго порядка (квадратичная) может быть, как видно



Ф и г. 3.2.2. Преобразование координат к главным осям.

x_1, x_2 — исходные координаты; \tilde{x}_1, \tilde{x}_2 — канонические координаты.

из фиг. 3.2.2, преобразована к новой системе координат путем переноса начала системы координат в точку экстремума функции и последующего поворота осей для достижения симметрии. Обозначим старые координаты через \tilde{x}_1 и \tilde{x}_2 , а новые координаты, называемые *главными осями*, — через x_1 и x_2 . Эти два преобразования дают новое, записанное в главных осях выражение для целевой функции, называемое *канонической функцией*, причем эта функция много проще исходной, поскольку исключены все члены первого порядка и перекрестные члены.

Например, в случае двух независимых переменных квадратичная целевая функция или квадратичная аппроксимация целевой функции в соответствии с уравнением (2.4.5) будет иметь следующий вид:

$$f(\mathbf{x}) = b_0 + b_1x_1 + b_2x_2 + b_{11}x_1^2 + b_{22}x_2^2 + b_{12}x_1x_2 + b_{21}x_2x_1. \quad (3.2.6)$$

Уравнение (3.2.6) преобразуется в каноническое уравнение

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \tilde{b}_{11}\tilde{x}_1^2 + \tilde{b}_{22}\tilde{x}_2^2, \quad (3.2.7)$$

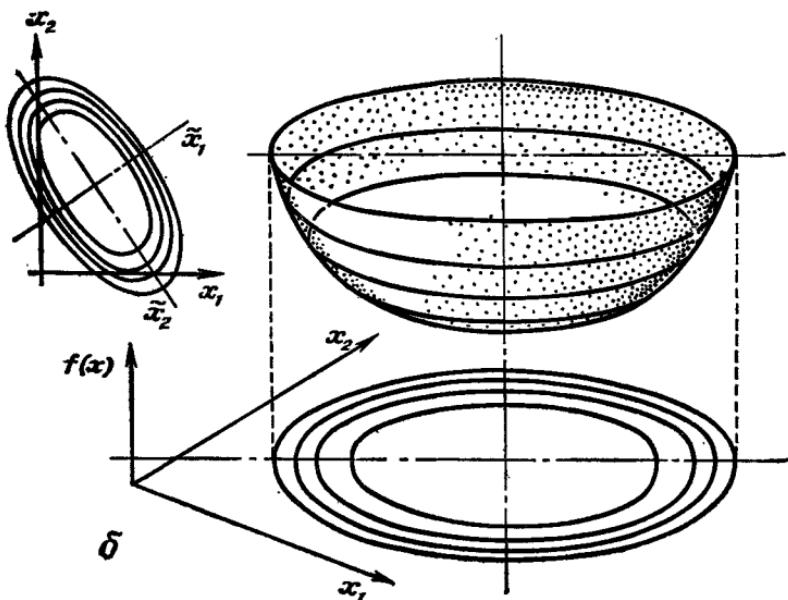
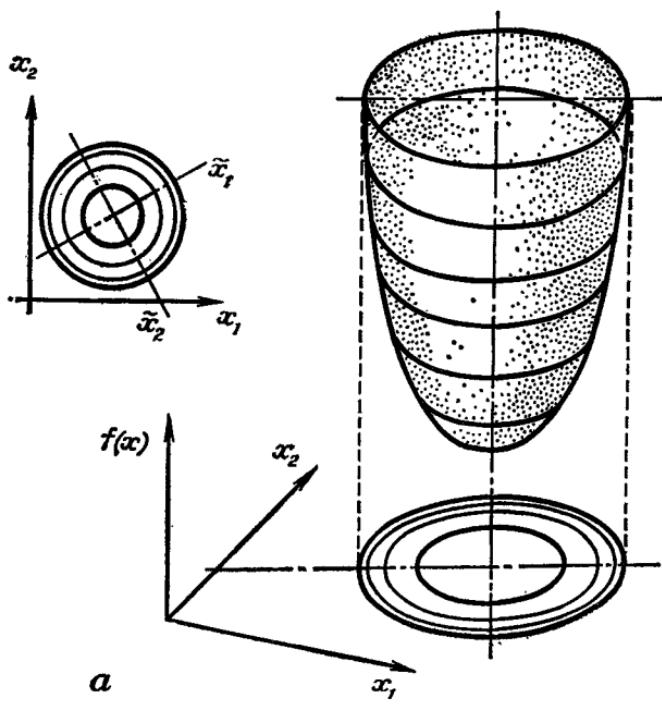
где $f(\mathbf{x}^*)$ — значение $f(\mathbf{x})$ в центре квадратичной поверхности, а \tilde{b}_{11} и \tilde{b}_{22} — преобразованные коэффициенты (знак \sim обозначает «в канонической форме»). Перенос начала координат на фиг. 3.2.2 соответствует исключению в уравнении (3.2.6) линейных членов, а поворот осей — исключению перекрестных членов. Кроме того, на фиг. 3.2.2 на крайнем справа рисунке представлен результат масштабирования преобразованных переменных, так что уровни целевой функции становятся окружностями. На фиг. 3.2.3 приводятся типичные примеры двумерных функций в исходных и в главных осях. Элементы матрицы Гессе целевой функции $\mathbf{H}(\mathbf{x})$ можно легко связать с коэффициентами уравнения (3.2.6):

$$\frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} = 2b_{11}, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_2} = 2b_{12} = \frac{\partial^2 f(\mathbf{x})}{\partial x_2 \partial x_1} = 2b_{21}, \quad \frac{\partial^2 f(\mathbf{x})}{\partial x_2^2} = 2b_{22}.$$

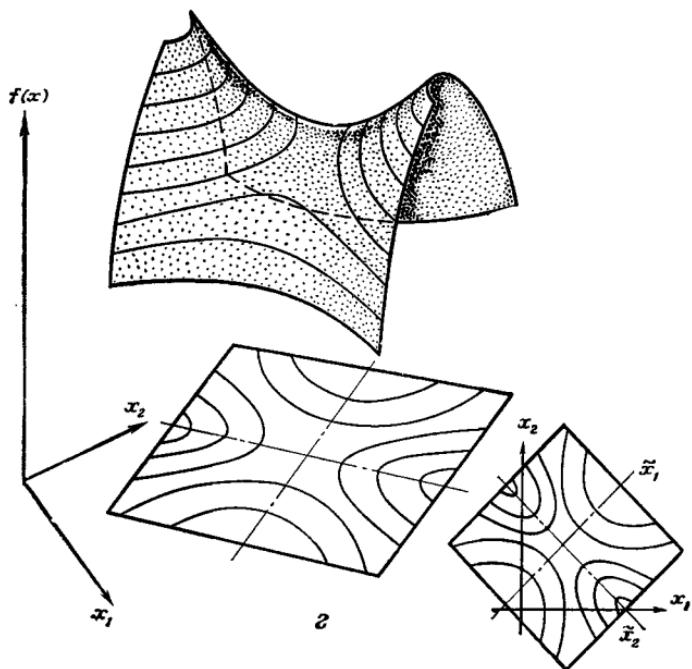
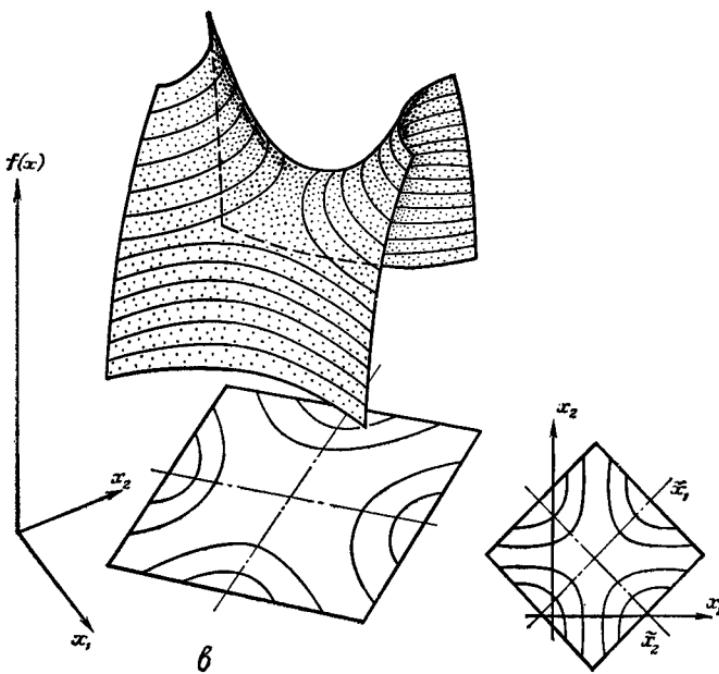
Кроме того, коэффициенты \tilde{b}_{11} и \tilde{b}_{22} в уравнении (3.2.7) являются собственными значениями матрицы $\frac{1}{2}\mathbf{H}(\mathbf{x}^{(k)})$ (доказательство этого факта можно найти во многих руководствах по теории матриц и линейной алгебре). Матрица, обратная матрице Гессе целевой функции, взятой в точке $\mathbf{x}^{(k)}$, дает меру кривизны функции $f(\mathbf{x})$ в окрестности $\mathbf{x}^{(k)}$.

В табл. 3.2.1 интерпретируется информация, содержащаяся в каноническом представлении (3.2.7) исходной функции (3.2.6). Если $|\tilde{b}_{11}| > |\tilde{b}_{22}|$, то линии уровней вытягиваются вдоль x_2 (меньший коэффициент) и наоборот. Если центр расположен по оси x_2 на бесконечности и \tilde{b}_{11} отрицателен, то линии уровней представляют собой параболы, как показано на фиг. 3.2.3, e. Каждая из вырожденных поверхностей, изображенных на фиг. 3.2.3, d, называемая оврагом или хребтом, имеет место, когда один из коэффициентов по абсолютной величине много меньше другого. В качестве простого примера возьмем функцию $f(\mathbf{x}) = x_1^2 + x_2^2 + 2x_1x_2$, которая в результате подстановки $\mathbf{x} = \mathbf{x}_1 + \mathbf{x}_2$ преобразуется в $f(\mathbf{x}) = f(\mathbf{x}) = x^2$. Тогда $\tilde{b}_{11} = 1$, а $\tilde{b}_{22} = 0$, что соответствует варианту 9 в табл. 3.2.1. Изложенное можно легко распространить на случай функций более чем двух независимых переменных. Например, когда все \tilde{b}_{ii} равны между собой, поверхности постоянного уровня целевой функции представляют собой гиперсферы; когда один или большее число коэффициентов \tilde{b}_{ii} относительно малы, имеет место гиперовраг (хребет) и т. д.

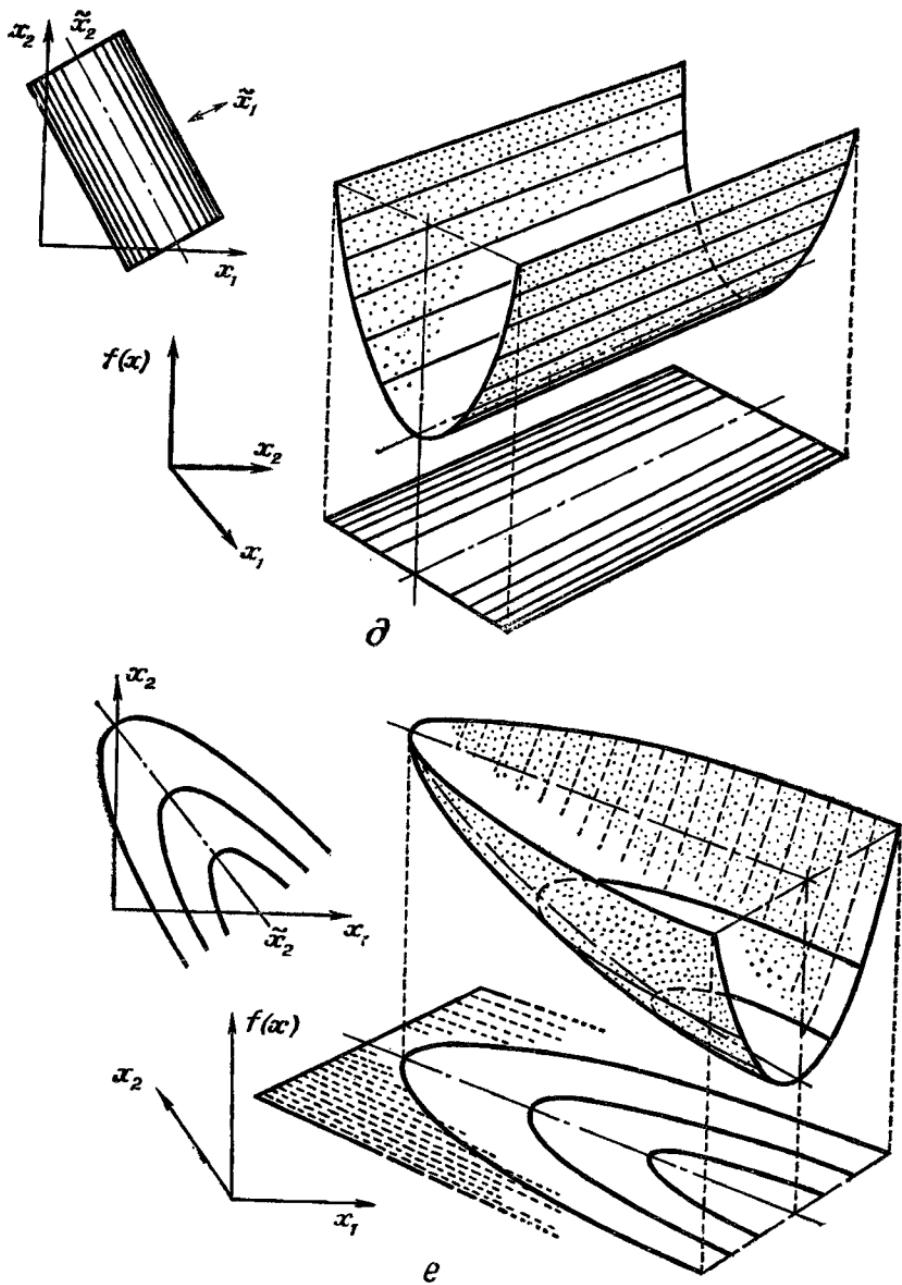
Одним из основных требований того, чтобы любая процедура оптимизации была успешной, является возможность быстро двигаться в локальной области вдоль узкого оврага (при минимизации) в направлении минимума целевой функции. Другими словами,



Ф и г. 3.2.3 (*a*, *б*). Геометрия целевых функций второго порядка двух независимых переменных.



Фиг. 3.2.3 (б, г).



Фиг. 3.2.3 (∂ , e).

хороший алгоритм должен дать направление поиска, имеющее большую составляющую вдоль оврага. Овраги (хребты в случае максимизации) встречаются весьма часто, по крайней мере локально. При этом вовсе не обязательно, чтобы функция, аппроксимирующая

Интерпретация канонической функции

$$f(\mathbf{x}) - f(\mathbf{x}^*) = \tilde{b}_{11} \tilde{x}_1^2 + \tilde{b}_{22} \tilde{x}_2^2$$

Таблица 3.2.1

Коэффициенты		Знаки	Тип линии уровня	Геометрическая интерпретация	Центр	Фиг. 3.2.3
вариант	соотношение	\tilde{b}_{11} \tilde{b}_{22}				
1	$\tilde{b}_{11} = \tilde{b}_{22}$	— —	Окружность	Круговой холм	Максимум	a
2	$\tilde{b}_{11} = \tilde{b}_{22}$	++	»	Круговая впадина	Минимум	a
3	$\tilde{b}_{11} > \tilde{b}_{22}$	— —	Эллипс	Эллиптический холм	Максимум	b
4	$\tilde{b}_{11} > \tilde{b}_{22}$	++	»	Эллиптическая впадина	Минимум	b
5	$\tilde{b}_{11} = \tilde{b}_{22}$	+ —	Гипербола	Симметричное седло	Седловая точка	v
6	$\tilde{b}_{11} = \tilde{b}_{22}$	— +	»	То же	То же	v
7	$\tilde{b}_{11} > \tilde{b}_{22}$	+ —	»	Вытянутое седло	»	g
8	$\tilde{b}_{22} = 0$	—	Прямая	Стационарный ¹⁾ хребет	Нет	d
9	$\tilde{b}_{22} = 0$	+	»	Стационарный ¹⁾ овраг	»	d
10	$\tilde{b}_{22} = 0$	—	Парабола	Поднимающийся ^{1,2)} хребет	На ∞	e
11	$\tilde{b}_{22} = 0$	+	»	Спадающий ^{1,2)} овраг	То же	e

1) Вырождение поверхности.

2) К соотношению, приведенному в названии таблицы, нужно добавить два линейных члена.

целевую функцию, была вырожденной. Так, например, в типичном случае, представленном на фиг. 3.2.3, б в верхней и нижней областях, линии уровней напоминают овраги. Овраг располагается в направлении собственного вектора, соответствующего малому собственному значению матрицы Гессе целевой функции. Например, для случая функции $f(\mathbf{x}) = x_1^2 + 25x_2^2$ из примера 3.1.1 (проиллюстрированного на фиг. П.3.1.1, а) матрица Гессе постоянна и равна

$$\mathbf{H} = \begin{pmatrix} 2 & 0 \\ 0 & 50 \end{pmatrix}.$$

Для этой матрицы собственные значения, полученные из характеристического уравнения $(2 - \beta)(50 - \beta) = 0$, равны $\beta_{11} = 2$ и $\beta_{22} = 50$. (В случае диагональной матрицы собственные значения — это элементы главной диагонали.) Собственный вектор, соответствующий β_{11} , может иметь любое значение для компоненты в направлении x_1 до тех пор, пока компонента в направлении x_2 равна нулю. Читатель, видимо, уже заметил, как овраг на фиг. П.3.1.1, *a* вытягивается в направлении x_1 . Еще раз отметим, что направление $-\mathbf{H}^{-1}\nabla f(\mathbf{x})$, вычисленное в различных точках пространства \mathbf{x} , всегда совпадает с направлением к точке минимума $f(\mathbf{x})$.

Функция	$\mathbf{x} = [2 \ 2]^T$	$\mathbf{x} = [2 \ 0]^T$
$\nabla f(\mathbf{x})$	$\begin{bmatrix} 4 \\ 100 \end{bmatrix}$	$\begin{bmatrix} 4 \\ 0 \end{bmatrix}$
$-\mathbf{H}^{-1}\nabla f(\mathbf{x})$	$(-1) \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{50} \end{bmatrix} \begin{bmatrix} 4 \\ 100 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}$	$(-1) \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{50} \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$

В этом примере вектор $-\mathbf{H}^{-1}(\mathbf{x})\nabla f(\mathbf{x})$ совпадает с направлением в точку минимума, поскольку функция $f(\mathbf{x})$ квадратичная, но, даже если целевая функция не квадратичная, в общем случае вектор $-\mathbf{H}^{-1}\nabla f(\mathbf{x})$ имеет большую составляющую, параллельную собственному вектору, соответствующему малому собственному значению $\mathbf{H}(\mathbf{x})$, и направления поиска, как требуется, приблизительно совпадают с главными осями при аппроксимации функции в точке $\mathbf{x}^{(k)}$.

Основным недостатком метода Ньютона (кроме трудностей, связанных с определением аналитических производных) является то, что улучшение значения целевой функции с каждым циклом гарантируется только в том случае, если матрица Гессе целевой функции $\mathbf{H}(\mathbf{x}) = \nabla^2 f(\mathbf{x}^{(k)})$ положительно определена. $\mathbf{H}(\mathbf{x})$ является положительно определенной для строго выпуклых функций, поэтому в случае функций более общего вида метод Ньютона может привести к направлениям поиска, расходящимся от точки минимума $f(\mathbf{x})$. Чтобы ясно себе представить, почему выполнение условия (3.2.5) необходимо для обеспечения сходимости, вспомним, что действительная симметрическая матрица положительно определена тогда и только тогда, когда ее собственные значения положительны. При минимизации, как это видно из фиг. 3.2.3, когда все собственные значения $\mathbf{H}(\mathbf{x}^{(k)})$ положительны, локальная квадратичная аппроксимация соответствует круговой или эллиптической впадине, имеющей минимум.

С другой стороны, если пара собственных значений имеет противоположные знаки, то, как видно из фиг. 3.2.3, квадратичная аппроксимация представляет собой седло, не имеющее локального минимума. В последнем случае вектор, определяемый уравнением (3.2.3) [или (3.2.4a)] и представляющий в смысле метода Ньютона наилучшее направление поиска, будет направлен к седловой точке вместо того, чтобы уходить от нее в направлении «вниз».

3.2.3. ПРИВЕДЕНИЕ МАТРИЦЫ ГЕССЕ К ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННОМУ ВИДУ

Некоторые авторы предложили дополнительные преобразования, делающие матрицу Гессе положительно определенной на каждом этапе минимизации. Гринштадт [7] построил схему на основе анализа собственных значений, которая обеспечивает положительную определенность приближения¹⁾. Пусть матрица $\tilde{H}(x)$ аппроксимирует $H(x)$. Масштабируем матрицу $\tilde{H}(x)$ следующим образом:

$$\Pi(x) = C^{-1}(x) \tilde{H}(x) C^{-1}(x), \quad (3.2.8)$$

где $C(x)$ — диагональная матрица, элементы которой равны $c_{ii} = (\tilde{h}_{ii})^{1/2}$, т. е. представляют собой положительные квадратные корни из абсолютных значений элементов главной диагонали $\tilde{H}(x)$. Тогда в матрице Π все элементы главной диагонали либо положительны, либо отрицательны. Например, пусть

$$\tilde{H} = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix},$$

тогда

$$C = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}, \quad C^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

$$\Pi = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & 1 \end{bmatrix}.$$

Поскольку $C^{-1}(x)$ и $\tilde{H}(x)$ неособенные и порядка n , то обратная матрица к их произведению является произведением их обратных матриц, взятых в обратном порядке, или

$$\Pi^{-1}(x) = C(x) \tilde{H}^{-1}(x) C(x).$$

¹⁾ Аппроксимации, используемой вместо $H^{-1}(x)$ в уравнениях (3.2.3) и (3.2.4a). — Прим. перев.

Тогда $\tilde{\mathbf{H}}^{-1}$ может быть вычислена по масштабированной матрице:

$$\tilde{\mathbf{H}}^{-1}(\mathbf{x}) = \mathbf{C}^{-1}(\mathbf{x}) \Pi^{-1}(\mathbf{x}) \mathbf{C}^{-1}(\mathbf{x}). \quad (3.2.9)$$

В своей работе Гринштадт указал, что $\Pi^{-1}(\mathbf{x})$ можно выразить через собственные значения α_i и собственные векторы матрицы $\Pi(\mathbf{x})$. Собственные векторы матрицы, обратной к данной, равны собственным векторам исходной матрицы, а собственные значения обратной матрицы — просто обратные величины собственных значений исходной матрицы (α_i^{-1}). Таким образом,

$$\Pi^{-1}(\mathbf{x}) = \sum_{i=1}^n \alpha_i^{-1} \mathbf{e}_i \mathbf{e}_i^T,$$

где \mathbf{e}_i — нормированный собственный вектор, соответствующий собственному значению α_i . Затем вместо $\Pi^{-1}(\mathbf{x})$ в формуле (3.2.9) используется $\tilde{\Pi}^{-1}(\mathbf{x})$:

$$\tilde{\Pi}^{-1}(\mathbf{x}) = \sum_{i=1}^n |\alpha_i|^{-1} \mathbf{e}_i \mathbf{e}_i^T$$

(где любое $\alpha_i = 0$ заменяется малой положительной величиной), так что теперь обеспечена положительная определенность $\tilde{\mathbf{H}}^{-1}(\mathbf{x})$ при вычислении ее по формуле

$$\tilde{\mathbf{H}}^{-1}(\mathbf{x}) = \mathbf{C}^{-1}(\mathbf{x}) \tilde{\Pi}^{-1}(\mathbf{x}) \mathbf{C}^{-1}(\mathbf{x}). \quad (3.2.10)$$

В качестве простого примера предположим, что матрица $\tilde{\mathbf{H}} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$; $\tilde{\mathbf{H}}^{-1}$ не является положительно определенной.

Вычислим

$$\Pi = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}.$$

Собственными значениями Π являются $\alpha_1 = -1$ и $\alpha_2 = 3$, и соответствующие нормированные собственные векторы имеют вид

$$\mathbf{e}_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix}, \quad \mathbf{e}_2 = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

Следовательно,

$$\Pi^{-1} = \frac{1}{-1} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} + \\ + \frac{1}{3} \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} -\frac{1}{3} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{3} \end{bmatrix}.$$

Затем вместо Π^{-1} возьмем

$$\tilde{\Pi}^{-1} = \left| \frac{1}{-1} \right| \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} + \\ + \left| \frac{1}{3} \right| \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 \\ -\frac{1}{\sqrt{2}} & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix},$$

и после подстановки $\tilde{\Pi}^{-1}$ в уравнение (3.2.10) новая матрица \tilde{H}^{-1} становится положительно определенной:

$$\tilde{H}^{-1} = \begin{bmatrix} \frac{2}{3} & \frac{1}{3} \\ \frac{1}{3} & \frac{2}{3} \end{bmatrix}.$$

Заметим, что здесь любое собственное значение, абсолютная величина которого меньше 10^{-4} , заменяется на 10^{-4} .

Маркуардт [8], Левенберг [9] и Голдфилд, Квондт и Тротер [10] предложили другую вычислительную схему для обеспечения положительной определенности приближения матрицы $H^{-1}(x)$:

$$\tilde{H}^{*-1}(x) = C^{-1}(x)(\Pi(x) + \beta I)^{-1}C^{-1}(x) = (\tilde{H}(x) + \beta C^2(x))^{-1}, \quad (3.2.11)$$

где β — положительная константа, такая, что $\beta > -\min\{\alpha_i\}$. Так как собственными значениями $(\Pi(x) + \beta I)$ являются $(\alpha_i + \beta)$, то уравнение (3.2.11) обеспечивает положительную определенность $\tilde{H}^{*-1}(x)$, поскольку использование подходящего значения β в формуле (3.2.11) в сущности «уничижает» отрицательные и малые собственные значения матрицы, аппроксимирующей матрицу Гессе. Заметим, что при достаточно большом β произведение βI может

«подавить» $\mathbf{H}(\mathbf{x})$, и тогда процесс минимизации приблизится к поиску методом наискорейшего спуска.

Зварт [11] предложил третий путь обеспечения положительной определенности для приближения $\tilde{\mathbf{H}}(\mathbf{x})$. Основным шагом является нахождение унитарной вещественной матрицы \mathbf{U} , такой, что после применения преобразования $\mathbf{x} = \tilde{\mathbf{U}}\mathbf{x}$ получается диагональная матрица Гессе целевой функции

$$\tilde{\mathbf{H}}(\mathbf{x}^{(k)}) = \mathbf{U}^T \mathbf{H}(\mathbf{x}^{(k)}) \mathbf{U} = \begin{bmatrix} \tilde{h}_{11} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \tilde{h}_{nn} \end{bmatrix}. \quad (3.2.12)$$

Этот шаг соответствует повороту осей на фиг. 3.2.2. Для построения унитарной матрицы \mathbf{U} необходимо определить собственные значения и собственные функции $\mathbf{H}(\mathbf{x}^{(k)})$. Затем, если какие-либо элементы на диагонали матрицы $\tilde{\mathbf{H}}(\mathbf{x}^{(k)})$ оказываются отрицательными или малыми, то они заменяются некоторой малой положительной величиной, что приводит к модифицированной матрице $\tilde{\mathbf{H}}^*(\mathbf{x}^{(k)})$. Таким образом, в процессе поиска направление, соответствующее собственному значению, имеющему не тот знак, который нужен, не рассматривается, и, кроме того, на выбор направления поиска не оказывают доминирующего влияния никакие слишком малые собственные значения. Наконец, направление поиска находится с помощью следующего соотношения:

$$\mathbf{s}^{(k)} = -\tilde{\mathbf{H}}^*(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)}). \quad (3.2.13)$$

Метод Ньютона имеет очевидное преимущество, заключающееся в квадратичной сходимости в окрестности минимума целевой функции (если $\mathbf{H}(\mathbf{x}) > 0$ и целевую функцию можно достаточно хорошо аппроксимировать квадратичной функцией), т. е. как раз там, где особенно проявляются слабые стороны методов наискорейшего спуска. Наоборот, в области, далекой от минимума, методы наискорейшего спуска могут оказаться лучше. Отсюда можно сделать вывод, что подходящая комбинация метода наискорейшего спуска и метода Ньютона должна оказаться эффективнее каждого из них в отдельности. Такие комбинированные процедуры излагаются в разд. 3.4.

3.3. СОПРЯЖЕННОСТЬ И СОПРЯЖЕННЫЕ НАПРАВЛЕНИЯ

Как будет видно ниже, квадратичная целевая функция n независимых переменных, имеющая минимум, может быть минимизирована за n шагов (или менее), если шаги предпринимаются в так называемых *сопряженных направлениях*. Хотя используемые в ре-

альных задачах схемы оптимизации, оказывающиеся эффективными для квадратичных целевых функций, иногда могут плохо работать (и в общем случае основанный на этом подход не даст сопряженных направлений поиска) при более сложных целевых функциях, тем не менее этот подход представляется вполне разумным. Прежде чем описывать какие-либо алгоритмы, следует сначала определить и проиллюстрировать свойство сопряженности. В этом разделе предполагается, если это не оговаривается особо, что целевая функция квадратична и имеет вид

$$f(\mathbf{x}) = a + \mathbf{b}^T \mathbf{x} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x},$$

где \mathbf{H} — положительно определенная матрица.

3.3.1. СОПРЯЖЕННОСТЬ

Предположим, что процесс минимизации $f(\mathbf{x})$ начинается в $\mathbf{x}^{(0)}$ с начальным направлением $\hat{\mathbf{s}}^{(0)}$, выбранным произвольно или по какому-либо определенному правилу. Для удобства возьмем его в виде единичного вектора, т. е. $(\hat{\mathbf{s}}^{(0)})^T \hat{\mathbf{s}}^{(0)} = 1$. Тогда вектор $\mathbf{x}^{(1)}$ будет равен

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \lambda^{(0)} \hat{\mathbf{s}}^{(0)}, \quad (3.3.1)$$

где длина шага $\lambda^{(0)}$ определяется из условия минимума $f(\mathbf{x}^{(0)} + \lambda \hat{\mathbf{s}}^{(0)})$ по λ , как это следует из уравнения (3.1.4):

$$\frac{df(\mathbf{x}^{(0)} + \lambda \hat{\mathbf{s}}^{(0)})}{d\lambda} = 0 = \nabla f(\mathbf{x}^{(0)}) \hat{\mathbf{s}}^{(0)} + (\hat{\mathbf{s}}^{(0)})^T \nabla^2 f(\mathbf{x}^{(0)}) \lambda \hat{\mathbf{s}}^{(0)}.$$

Таким образом,

$$\lambda^{(0)} = - \frac{\nabla f(\mathbf{x}^{(0)}) \hat{\mathbf{s}}^{(0)}}{(\hat{\mathbf{s}}^{(0)})^T \nabla^2 f(\mathbf{x}^{(0)}) \hat{\mathbf{s}}^{(0)}}. \quad (3.3.2)$$

После того как по формулам (3.1.1) и (3.2.2) вычислен $\mathbf{x}^{(1)}$, для продолжения процесса минимизации $f(\mathbf{x})$ должно быть выбрано новое направление. Это новое направление $\hat{\mathbf{s}}^{(1)}$ называется *сопряженным* к старому направлению $\hat{\mathbf{s}}^{(0)}$, если $(\hat{\mathbf{s}}^{(1)})^T [\nabla^2 f(\mathbf{x}^{(0)})] \hat{\mathbf{s}}^{(0)} = 0$. В общем случае система n линейно независимых направлений поиска $\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots, \mathbf{s}^{(n-1)}$ называется сопряженной по отношению к некоторой положительно определенной (квадратной) матрице \mathbf{Q} [12], если

$$(\mathbf{s}^{(i)})^T \mathbf{Q} \mathbf{s}^{(j)} = 0, \quad 0 \leq i \neq j \leq n-1. \quad (3.3.3)$$

(Мы предполагаем, что \mathbf{Q} — невырожденная матрица.) Примером \mathbf{Q} может служить матрица Гессе целевой функции \mathbf{H} .

Сопряженность — понятие, аналогичное ортогональности; действительно, когда $\mathbf{H} = \mathbf{I}$, то в соответствии с уравнением (3.3.3) $(\mathbf{s}^{(l)})^T \mathbf{s}^{(l)} = 0$. Заметим далее, что с точки зрения топологических характеристик квадратичной функции, обсуждавшихся в предыдущем разделе, если целевая функция преобразована к каноническому виду, скажем, $f(\mathbf{x}) = \tilde{b}_{11}x_1^2 + \tilde{b}_{22}x_2^2 = \tilde{\mathbf{x}}^T \tilde{\mathbf{H}} \tilde{\mathbf{x}}$, то собственные значения $\frac{1}{2} \mathbf{H}$ находятся на диагонали $\tilde{\mathbf{H}}$:

$$\tilde{\mathbf{H}} = \begin{bmatrix} \tilde{b}_{11} & 0 \\ 0 & \tilde{b}_{22} \end{bmatrix}.$$

Таким образом, сопряженность можно интерпретировать как ортогональность в пространстве преобразованных координат, если направления поиска масштабированы подходящими функциями собственных значений. В новых координатах имеем

$$\begin{aligned} (\tilde{\mathbf{s}}^{(l)})^T \tilde{\mathbf{H}} \tilde{\mathbf{s}}^{(l)} &= \tilde{b}_{11} \tilde{s}_1^{(l)} \tilde{s}_1^{(l)} + \tilde{b}_{22} \tilde{s}_2^{(l)} \tilde{s}_2^{(l)} = \\ &= \frac{\tilde{s}_1^{(l)} \tilde{s}_1^{(l)}}{\tilde{b}_{22}} + \frac{\tilde{s}_2^{(l)} \tilde{s}_2^{(l)}}{\tilde{b}_{11}} = (\tilde{\mathbf{s}}^{(l)})^T \tilde{\mathbf{s}}^{(l)} = 0, \end{aligned}$$

где

$$\tilde{s}_1^{(l)} = \frac{\tilde{s}_1^{(l)}}{\sqrt{\tilde{b}_{22}}} \quad \text{и} \quad \tilde{s}_2^{(l)} = \frac{\tilde{s}_2^{(l)}}{\sqrt{\tilde{b}_{11}}}.$$

Поскольку сопряженные направления линейно независимы, то любой вектор \mathbf{v} в пространстве E^n можно выразить через $\mathbf{s}^{(0)}, \mathbf{s}^{(1)}, \dots$ следующим образом:

$$\mathbf{v} = \sum_{j=0}^{n-1} v^{(j)} \mathbf{s}^{(j)}, \tag{3.3.4}$$

где

$$v^{(j)} = \frac{(\mathbf{s}^{(j)})^T \mathbf{H}(\mathbf{x}) \mathbf{v}}{(\mathbf{s}^{(j)})^T \mathbf{H}(\mathbf{x}) \mathbf{s}^{(j)}}.$$

Для некоторой матрицы \mathbf{H} всегда существует по крайней мере одна система из n взаимно сопряженных направлений, так как сами собственные векторы \mathbf{H} представляют собой такую систему.

В дальнейшем при использовании сопряженных направлений потребуется следующее соотношение для квадратичной функции:

$$\mathbf{H}^{-1} = \sum_{j=0}^{n-1} \frac{\mathbf{s}^{(j)} (\mathbf{s}^{(j)})^T}{(\mathbf{s}^{(j)})^T \mathbf{H} \mathbf{s}^{(j)}}. \tag{3.3.4a}$$

Чтобы убедиться в его справедливости, рассмотрим матрицу $\sum_{j=0}^{n-1} \alpha_j s^{(j)} (s^{(j)})^T$. Умножение ее справа на $Hs^{(k)}$ дает

$$\left[\sum_{j=0}^{n-1} \alpha_j s^{(j)} (s^{(j)})^T \right] Hs^{(k)} = \alpha_k s^{(k)} (s^{(k)})^T Hs^{(k)} = s^{(k)},$$

если положить $\alpha_k = [(s^{(k)})^T Hs^{(k)}]^{-1}$.

Таким образом, соотношение (3.3.4a) доказано.

Приведенный ниже числовой пример поможет уяснить понятие сопряженности.

Пример 3.3.1. Сопряженные направления

В качестве примера выбора и построения сопряженных направлений рассмотрим следующую задачу:

минимизировать $f(x) = x_1^2 + x_2^2 - 4$,

начиная из точки $x^{(0)} = [4 \ 4]^T$. Поскольку $f(x)$ с самого начала представляет собой квадратичное выражение, нет необходимости ее аппроксимировать. Потребуются следующие матрицы и векторы:

$$(x - x^{(k)}) = \begin{bmatrix} x_1 - x_1^{(k)} \\ x_2 - x_2^{(k)} \end{bmatrix},$$

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix},$$

$$H(x) = \nabla^2 f(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}.$$

Вектор $x^{(1)}$ вычисляется по формуле $x^{(1)} = x^{(0)} + \lambda^{(0)} \hat{s}^{(0)}$.

$$\begin{bmatrix} x_1^{(1)} \\ x_2^{(1)} \end{bmatrix} = \begin{bmatrix} x_1^{(0)} \\ x_2^{(0)} \end{bmatrix} + \lambda^{(0)} \begin{bmatrix} \hat{s}_1^{(0)} \\ \hat{s}_2^{(0)} \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix} + \lambda^{(0)} \begin{bmatrix} \hat{s}_1^{(0)} \\ \hat{s}_2^{(0)} \end{bmatrix}.$$

В качестве $\hat{s}^{(0)}$ можно выбрать любое направление, например направление градиента, но в иллюстративных целях мы выберем произвольное направление

$$\hat{s}^{(0)} = \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}.$$

Заметим, что

$$(\hat{s}^{(0)})^T \hat{s}^{(0)} = 1,$$

так как

$$\left[\frac{1}{2} \quad \frac{\sqrt{3}}{2} \right] \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} = 1.$$

Длина шага $\lambda^{(0)}$ вычисляется по формуле (3.3.2):

$$\lambda^{(0)} = - \frac{[8 \quad 8] \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}}{\left[\frac{1}{2} \quad \frac{\sqrt{3}}{2} \right] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix}} = -5,46.$$

Тогда

$$\lambda^{(0)} \hat{s}^{(0)} = \begin{bmatrix} -2,73 \\ -4,74 \end{bmatrix}$$

и

$$\mathbf{x}^{(1)} = \begin{bmatrix} 1,27 \\ -0,74 \end{bmatrix}.$$

Следующее направление минимизации $\hat{s}^{(1)}$ выбирается сопряженным к $\hat{s}^{(0)}$, используя уравнение (3.3.3) при $\mathbf{Q} = \mathbf{H}$:

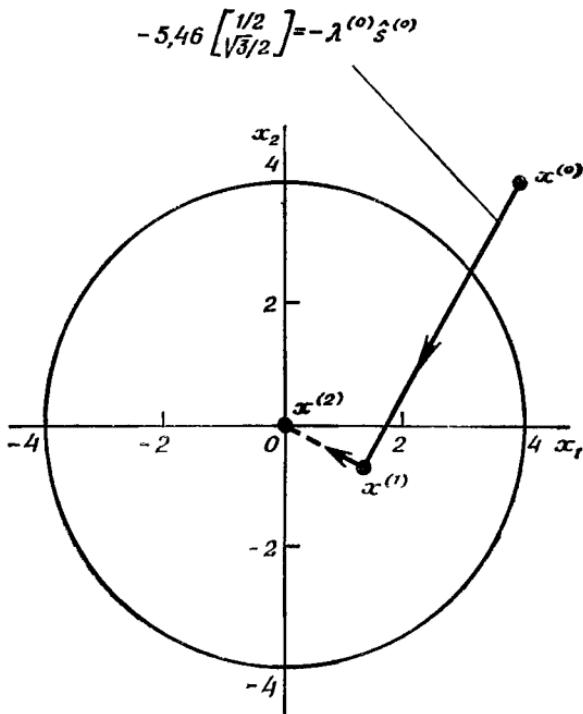
$$[\hat{s}_1^{(1)} \quad \hat{s}_2^{(1)}] \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{\sqrt{3}}{2} \end{bmatrix} = 0.$$

Необходимо еще одно добавочное уравнение (нормировки)

$$[\hat{s}_1^{(1)} \quad \hat{s}_2^{(1)}] \begin{bmatrix} \hat{s}_1^{(1)} \\ \hat{s}_2^{(1)} \end{bmatrix} = 1.$$

Из этих уравнений имеем

$$\left. \begin{aligned} \hat{s}_1^{(1)} + \sqrt{3} \hat{s}_2^{(1)} &= 0 \\ (\hat{s}_1^{(1)})^2 + (\hat{s}_2^{(1)})^2 &= 1 \end{aligned} \right\}, \text{ так что } \hat{s}^{(1)} = \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix}.$$



Ф и г. П.3.3.1.

Затем из уравнения, аналогичного (3.3.2), вычисляется длина шага $\lambda^{(1)}$:

$$\lambda^{(1)} = - \frac{[2,54 - 1,48] \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix}}{\begin{bmatrix} -\frac{\sqrt{3}}{2} & \frac{1}{2} \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix}} = 1,47,$$

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} 1,27 \\ -0,74 \end{bmatrix} + 1,47 \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

На фиг. П.3.3.1 изображена траектория поиска. В этом конкретном случае целевая функция сферическая; таким образом, направления поиска не только сопряжены, но и ортогональны.

Можно показать, что векторы $\hat{s}^{(k)}$, построенные как сопряженные направления, линейно независимы в смысле, изложенном в разд. 2.1. Единственный возможный дополнительный вектор, который взаимно сопряжен со всеми n направлениями $\hat{s}^{(0)}, \dots, \hat{s}^{(n-1)}$, — нулевой. Таким образом, изложенный выше пример иллюстрирует общее правило, заключающееся в том, что *если используются сопряженные направления, то любая квадратичная функция n переменных, имеющая минимум, может быть минимизирована за n шагов*, по одному в каждом из сопряженных направлений. Более того, порядок использования этих направлений несуществен.

Это правило для сопряженных направлений можно доказать следующим образом. Запишем целевую функцию в виде $f(\mathbf{x}) = a + \mathbf{x}^T \mathbf{b} + \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x}$, при этом $\nabla f(\mathbf{x}) = \mathbf{b} + \mathbf{H} \mathbf{x}$ и в точке минимума $f(\mathbf{x})$, где $\nabla f(\mathbf{x}^*) = 0$, $\mathbf{x}^* = -\mathbf{H}^{-1} \mathbf{b}$. Следует также заметить, что из коммутативности скалярного произведения

$$\nabla^T f(\mathbf{x}^{(k)}) \hat{\mathbf{s}}^{(k)} = (\hat{\mathbf{s}}^{(k)})^T \nabla f(\mathbf{x}^{(k)})$$

(что можно получить и непосредственным перемножением элементов). На n -м шаге в результате применения формул (3.1.1) и (3.3.2) имеем

$$\mathbf{x}^{(n)} = \mathbf{x}^{(0)} + \sum_{k=0}^{n-1} \lambda^{(k)} \hat{\mathbf{s}}^{(k)}.$$

На каждом шаге мы минимизируем $f(\mathbf{x}^{(k)} + \lambda \hat{\mathbf{s}}^{(k)})$ в направлении $\hat{\mathbf{s}}^{(k)}$, чтобы получить $\lambda^{(k)}$, что приводит к выражению

$$\mathbf{x}^{(n)} = \mathbf{x}^{(0)} - \sum_{k=0}^{n-1} \left(\frac{(\hat{\mathbf{s}}^{(k)})^T \nabla f(\mathbf{x}^{(k)})}{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}} \right) \hat{\mathbf{s}}^{(k)}. \quad (3.3.5a)$$

Кроме того,

$$\begin{aligned} (\hat{\mathbf{s}}^{(k)})^T \nabla f(\mathbf{x}^{(k)}) &= (\hat{\mathbf{s}}^{(k)})^T (\mathbf{H} \mathbf{x}^{(k)} + \mathbf{b}) = \\ &= (\hat{\mathbf{s}}^{(k)})^T \left\{ \mathbf{H} \left[\mathbf{x}^{(0)} + \sum_{i=1}^{k-1} \lambda^{(i)} \hat{\mathbf{s}}^{(i)} \right] + \mathbf{b} \right\} = \\ &= (\hat{\mathbf{s}}^{(k)})^T \{ \mathbf{H} \mathbf{x}^{(0)} + \mathbf{b} \}, \end{aligned}$$

так как все произведения $(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(i)}$ обращаются в нуль вследствие того, что векторы $\hat{\mathbf{s}}$ сопряжены. Таким образом,

$$\mathbf{x}^{(n)} = \mathbf{x}^{(0)} - \sum_{k=0}^{n-1} \frac{(\hat{\mathbf{s}}^{(k)})^T (\mathbf{H} \mathbf{x}^{(0)} + \mathbf{b}) \hat{\mathbf{s}}^{(k)}}{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}}. \quad (3.3.5b)$$

По аналогии с формулой (3.3.4) выразим векторы $\mathbf{x}^{(0)}$ и $\mathbf{H}^{-1}\mathbf{b}$ через систему сопряженных векторов $\hat{\mathbf{s}}^{(k)}$:

$$\mathbf{x}^{(0)} = \sum_{k=0}^{n-1} \frac{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \mathbf{x}^{(0)} \hat{\mathbf{s}}^{(k)}}{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}}$$

и

$$\mathbf{H}^{-1}\mathbf{b} = \sum_{k=0}^{n-1} \frac{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} (\mathbf{H}^{-1}\mathbf{b}) \hat{\mathbf{s}}^{(k)}}{(\hat{\mathbf{s}}^{(k)})^T \mathbf{H} \hat{\mathbf{s}}^{(k)}}.$$

Подстановка этих выражений в (3.3.5б) дает

$$\mathbf{x}^{(n)} = -\mathbf{H}^{-1}\mathbf{b}. \quad (3.3.6)$$

Таким образом, точка \mathbf{x}^n (результат n -го шага) совпадает с минимумом $f(\mathbf{x})$.

Говорят, что некоторый метод обладает свойством квадратичного окончания, если применение его гарантирует достижение минимума квадратичной целевой функции за точно определенное число шагов. В случае методов сопряженных градиентов требуется n шагов, тогда как, например, в методе Ньютона — только один шаг.

Покажем далее, что для сопряженных направлений (или просто линейно независимых направлений, что в этом случае одно и то же), если каждый раз $f(\mathbf{x})$ минимизируется в сопряженном направлении \mathbf{s} в соответствии с формулой (3.1.4), то $[(\mathbf{s}^{(j)})^T \nabla f(\mathbf{x}^{(l)})] = 0$, $0 \leq j \leq l-1$, при использовании не более n направлений. Мы уже видели ранее в разд. 3.1, что для квадратичной функции

$$\nabla f(\mathbf{x}^{(l)}) = \mathbf{b} + \mathbf{Hx}^{(l)}.$$

Следовательно,

$$\nabla f(\mathbf{x}^{(l)}) = \mathbf{b} + \mathbf{H} \left(\mathbf{x}^{(k)} + \sum_{i=k}^{l-1} \lambda^{(i)} \mathbf{s}^{(i)} \right) = \mathbf{b} + \mathbf{Hx}^{(k)} + \mathbf{H} \sum_{i=k}^{l-1} \lambda^{(i)} \mathbf{s}^{(i)},$$

где $\mathbf{x}^{(k)}$ — произвольная точка, из которой начинается поиск в сопряженных направлениях. Поскольку

$$\nabla f(\mathbf{x}^{(k)}) = \mathbf{b} + \mathbf{Hx}^{(k)},$$

то

$$\nabla f(\mathbf{x}^{(l)}) = \nabla f(\mathbf{x}^{(k)}) + \sum_{i=k}^{l-1} \lambda^{(i)} \mathbf{Hs}^{(i)}.$$

Умножение этого уравнения слева на $(\mathbf{s}^{(k-1)})^T$ дает

$$(\mathbf{s}^{(k-1)})^T \nabla f(\mathbf{x}^{(l)}) = (\mathbf{s}^{(k-1)})^T \nabla f(\mathbf{x}^{(k)}) + \sum_{i=k}^{l-1} \lambda^{(i)} (\mathbf{s}^{(k-1)})^T \mathbf{Hs}^{(i)}.$$

Первый член в правой части равен нулю, поскольку, как было показано в подразд. 3.1.1, градиент в данной точке ортогонален

предыдущему направлению поиска, если точка получается в результате минимизации функции в этом направлении. Кроме того, все члены в сумме исчезают вследствие сопряженности. Таким образом, $(\mathbf{s}^{(k-1)})^T \nabla f(\mathbf{x}^{(l)}) = 0$, и поскольку приведенные выше рассуждения справедливы для любого индекса k , изменяющегося от 1 до l , то

$$(\mathbf{s}^{(j)})^T \nabla f(\mathbf{x}^{(l)}) = 0, \quad 0 \leq j \leq l-1. \quad (3.3.7)$$

Определим теперь матрицу $\mathbf{X}^{(l)}$ размерности $n \times i$, в которой каждый элемент представляет собой

$$\begin{aligned} \mathbf{x}^{(i+1)} - \mathbf{x}^{(i)} &\equiv \Delta \mathbf{x}^{(i)} = \lambda^{(i)} \hat{\mathbf{s}}^{(i)}, \\ \mathbf{X}^{(i)} &= [(\mathbf{x}^{(1)} - \mathbf{x}^{(0)}) (\mathbf{x}^{(2)} - \mathbf{x}^{(1)}) \dots (\mathbf{x}^{(i)} - \mathbf{x}^{(i-1)})] = \\ &= [\Delta \mathbf{x}^{(0)} \Delta \mathbf{x}^{(1)} \dots \Delta \mathbf{x}^{(i-1)}], \end{aligned}$$

или

$$\begin{aligned} (\mathbf{X}^{(i)})^T &= \begin{bmatrix} (\Delta \mathbf{x}^{(0)})^T \\ (\Delta \mathbf{x}^{(1)})^T \\ \vdots \\ (\Delta \mathbf{x}^{(i-1)})^T \end{bmatrix} = \begin{bmatrix} \lambda^{(0)} (\hat{\mathbf{s}}^{(0)})^T \\ \lambda^{(1)} (\hat{\mathbf{s}}^{(1)})^T \\ \vdots \\ \lambda^{(i-1)} (\hat{\mathbf{s}}^{(i-1)})^T \end{bmatrix} = \\ &= \begin{bmatrix} \lambda^{(0)} \hat{s}_1^{(0)} & \lambda^{(0)} \hat{s}_2^{(0)} \dots \lambda^{(0)} \hat{s}_n^{(0)} \\ \lambda^{(1)} \hat{s}_1^{(1)} & \lambda^{(1)} \hat{s}_2^{(1)} \dots \lambda^{(1)} \hat{s}_n^{(1)} \\ \vdots & \vdots & \vdots \\ \lambda^{(i-1)} \hat{s}_1^{(i-1)} & \lambda^{(i-1)} \hat{s}_2^{(i-1)} \dots \lambda^{(i-1)} \hat{s}_n^{(i-1)} \end{bmatrix}. \end{aligned}$$

В соответствии с уравнением (3.3.7) имеем

$$(\mathbf{X}^{(i)})^T \nabla f(\mathbf{x}^{(i)}) = 0, \quad i \leq n-1. \quad (3.3.8)$$

Берем далее еще одну матрицу размерности $n \times i$:

$$\begin{aligned} \mathbf{G}^{(i)} &= [(\nabla f(\mathbf{x}^{(1)}) - \nabla f(\mathbf{x}^{(0)})) (\nabla f(\mathbf{x}^{(2)}) - \nabla f(\mathbf{x}^{(1)})) \dots \\ &\dots (\nabla f(\mathbf{x}^{(i)}) - \nabla f(\mathbf{x}^{(i-1)}))] = [\Delta \mathbf{g}^{(0)} \Delta \mathbf{g}^{(1)} \dots \Delta \mathbf{g}^{(i-1)}], \end{aligned}$$

где $\Delta \mathbf{g}^{(i)} \equiv \nabla f(\mathbf{x}^{(i+1)}) - \nabla f(\mathbf{x}^{(i)})$. Из выражения для градиента $f(\mathbf{x})$ следует

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}^{(k)}) + \nabla^2 f(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}), \quad (3.3.9)$$

и если положить $\mathbf{x} = \mathbf{x}^{(k+1)}$, то

$$\nabla f(\mathbf{x}^{(k+1)}) - \nabla f(\mathbf{x}^{(k)}) = \nabla^2 f(\mathbf{x}^{(k)}) (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \equiv \mathbf{H}(\lambda^{(k)} \hat{\mathbf{s}}^{(k)}). \quad (3.3.10)$$

Тогда на основании (3.3.8) имеем

$$(\mathbf{X}^{(i)})^T [\nabla f(\mathbf{x}^{(i+1)}) - \nabla f(\mathbf{x}^{(i)})] = 0 \quad (3.3.11)$$

или

$$(\mathbf{G}^{(i)})^T [\mathbf{x}^{(i+1)} - \mathbf{x}^{(i)}] = 0, \quad 0 \leq i \leq j \leq n-1, \quad (3.3.12)$$

если шаги $\lambda^{(0)} \hat{\mathbf{s}}^{(0)}, \lambda^{(1)} \hat{\mathbf{s}}^{(1)}, \dots$ осуществляются в сопряженных направлениях.

После изложения некоторых важных свойств сопряженности можно перейти к рассмотрению алгоритмов минимизации $f(\mathbf{x})$, использующих сопряженные направления.

3.3.2. МЕТОД СОПРЯЖЕННОГО ГРАДИЕНТА

В методе сопряженного градиента Флетчера — Ривса [13] строится последовательность направлений поиска \mathbf{s} , являющихся линейными комбинациями — $\nabla f(\mathbf{x}^{(k)})$, текущего направления наискорейшего спуска, и $\mathbf{s}^{(0)}, \dots, \mathbf{s}^{(k-1)}$, предыдущих направлений поиска, причем весовые коэффициенты выбираются так, чтобы сделать направления поиска сопряженными. Упомянутые веса такие, что для вычисления нового направления поиска в точке $\mathbf{x}^{(k)}$ используются только текущий градиент и предпоследний градиент. Эта идея заимствована из метода решения систем линейных уравнений, предложенного Хестенсом и Штифлем [14], а также Бекменом [15].

Изложим существование идеи. Пусть исходным направлением поиска будет $\mathbf{s}^{(0)} = -\nabla f(\mathbf{x}^{(0)})$. Затем положим $\mathbf{x}^{(1)} - \mathbf{x}^{(0)} = \lambda^{*(0)} \mathbf{s}^{(0)}$ и построим

$$\mathbf{s}^{(1)} = -\nabla f(\mathbf{x}^{(1)}) + \omega_1 \mathbf{s}^{(0)},$$

где ω_1 — скалярный вес, который выбирается так, чтобы сделать $\mathbf{s}^{(1)}$ и $\mathbf{s}^{(0)}$ сопряженными по отношению к \mathbf{H} :

$$(\mathbf{s}^{(0)})^T \mathbf{H} \mathbf{s}^{(1)} = 0. \quad (3.3.13)$$

Чтобы исключить $(\mathbf{s}^{(0)})^T$ из уравнения (3.3.13), воспользуемся уравнением (3.3.10) и заметим, что для квадратичной функции $\mathbf{H} = \mathbf{H}^T$

$$(\mathbf{s}^{(0)})^T = \frac{(\mathbf{x}^{(1)} - \mathbf{x}^{(0)})^T}{\lambda^{*(0)}} = \frac{[\nabla f(\mathbf{x}^{(1)}) - \nabla f(\mathbf{x}^{(0)})]^T \mathbf{H}^{-1}}{\lambda^{*(0)}}.$$

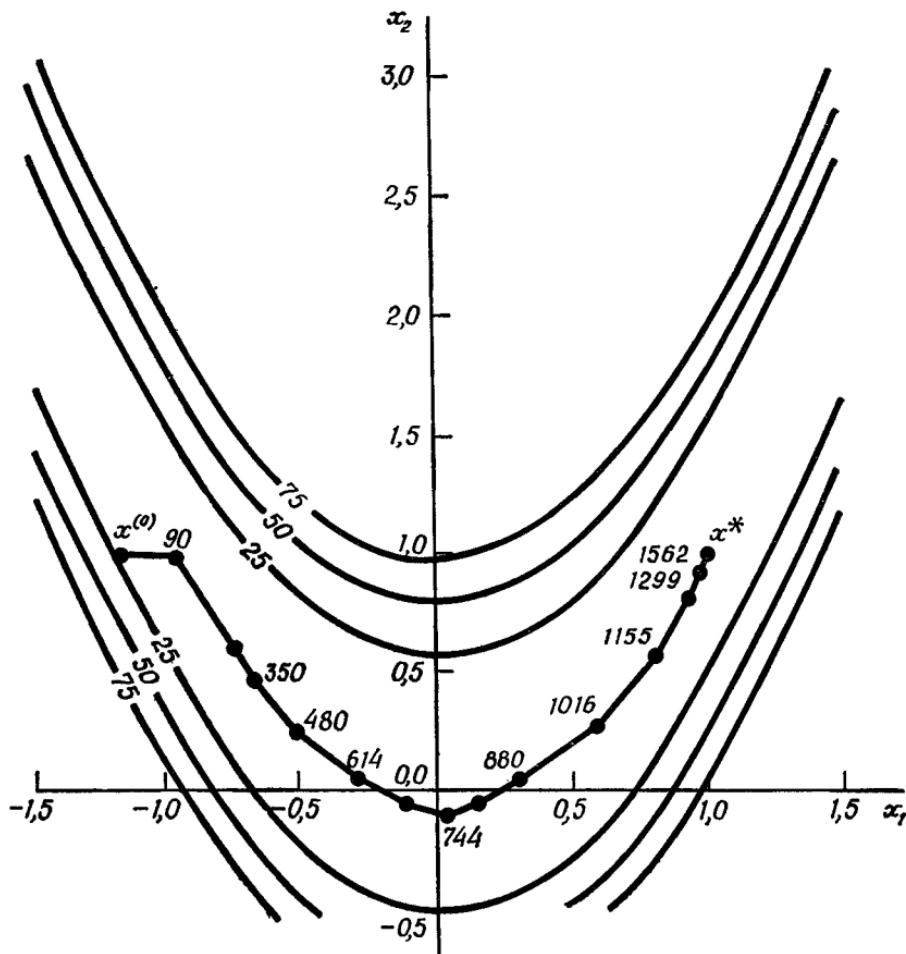
Следовательно,

$$[\nabla f(\mathbf{x}^{(1)}) - \nabla f(\mathbf{x}^{(0)})]^T [-\nabla f(\mathbf{x}^{(1)}) + \omega_1 \mathbf{s}^{(0)}] = 0.$$

Вследствие изложенных в подразд. 3.3.1 свойств все перекрестные члены исчезают, так что

$$\omega_1 = \frac{\nabla^T f(\mathbf{x}^{(1)}) \nabla f(\mathbf{x}^{(1)})}{\nabla^T f(\mathbf{x}^{(0)}) \nabla f(\mathbf{x}^{(0)})}.$$

Направление поиска $s^{(2)}$ представляется в виде линейной комбинации — $\nabla f(x^{(2)})$, $s^{(1)}$ и $s^{(0)}$, причем так, чтобы оно было сопряжено с $s^{(1)}$. Распространяя эти выкладки на $s^{(2)}, s^{(3)}, \dots$ (детали пре-



Ф и г. 3.3.1. Траектория поиска минимума функции Розенброка методом Флэтчера — Ривса (числа указывают номер шага, т. е. последовательные направления поиска).

образований мы вынуждены опустить, но они являются прямым продолжением изложенного, если учесть, что $(s^{(k)})^T \nabla f(x^{(k+1)}) = 0$ приводит к $\nabla^T f(x^{(k)}) \nabla f(x^{(k+1)}) = 0$, получаем общее выражение для ω_k :

$$\omega_k = \frac{\nabla^T f(x^{(k)}) \nabla f(x^{(k)})}{\nabla^T f(x^{(k-1)}) \nabla f(x^{(k-1)})}. \quad (3.3.14)$$

Все весовые множители, предшествующие ω_k , а именно $\omega_{k-1}, \omega_{k-2}, \dots$, оказываются нулями, что представляется весьма тонким и интересным результатом.

Ниже приводятся операции этого алгоритма:

1. В $\mathbf{x}^{(0)}$ вычисляется

$$\mathbf{s}^{(0)} = -\nabla f(\mathbf{x}^{(0)}).$$

2. На k -м шаге с помощью одномерного поиска в направлении $\mathbf{s}^{(k)}$ находится минимум $f(\mathbf{x})$. Это определяет точку $\mathbf{x}^{(k+1)}$.

3. Вычисляются $f(\mathbf{x}^{(k+1)})$ и $\nabla f(\mathbf{x}^{(k+1)})$.

4. Направление $\mathbf{s}^{(k+1)}$ определяется из соотношения

$$\mathbf{s}^{(k+1)} = -\nabla f(\mathbf{x}^{(k+1)}) + \mathbf{s}^{(k)} \frac{\nabla^T f(\mathbf{x}^{(k+1)}) \nabla f(\mathbf{x}^{(k+1)})}{\nabla^T f(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})}.$$

После $(n+1)$ -й итерации ($k = n$) процедура циклически повторяется с заменой $\mathbf{x}^{(0)}$ на $\mathbf{x}^{(n+1)}$.

5. Алгоритм заканчивается, когда $\|\mathbf{s}^{(k)}\| < \epsilon$, где ϵ — произвольная константа.

Прежде всего заметим, что здесь не требуется обратимость матрицы. Другим преимуществом этого алгоритма является то, что программа требует довольно ограниченной памяти ЭВМ по сравнению с $(n \times n)$ -матрицами разд. 3.4. На фиг. 3.3.1 изображена траектория минимизации для функции Розенброка. Оценивание алгоритма Флетчера — Ривса для некоторых тестовых задач проведено в гл. 5.

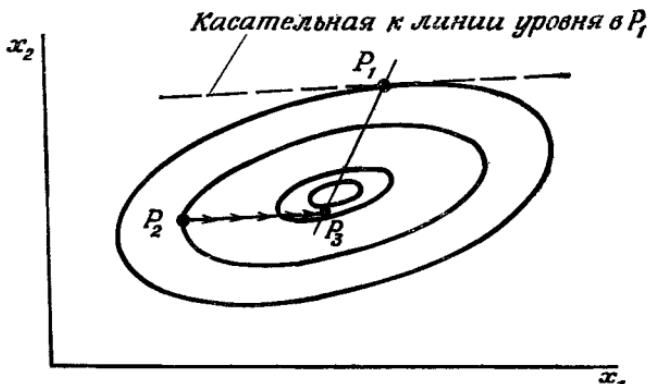
3.3.3. ПАРТАН-МЕТОДЫ

В работе Ша, Бюлера и Кемпторна [16] описаны несколько алгоритмов, использующих сопряженные направления (общий партан-метод) и сопряженные градиенты (партан-метод наискорейшего спуска, модифицированный партан-метод). Рабочую программу для этих методов можно получить в фирме Electronic Associates Inc., Princeton, N. J. Партан представляет собой сокращение от термина parallel tangents (параллельные касательные). На фиг. 3.3.2 проиллюстрирована сущность этой процедуры для случая квадратичной функции двух независимых переменных. P_1 и P_2 — любые две точки плоскости (x_1, x_2) . Сначала движемся из P_2 параллельно касательной к линии уровня в P_1 до тех пор, пока не будет достигнут минимум $f(\mathbf{x})$ в некоторой точке P_3 . При этом оказывается, что касательные в P_1 и P_3 параллельны, а минимум $f(\mathbf{x})$ находится на линии, проходящей через точки P_1 и P_3 .

В этом методе используется тот факт, что параллельность линий сохраняется при общем аффинном (т. е. по существу каноническом)

преобразовании пространства независимых переменных, так что на некоторые аспекты поиска минимума не будет влиять масштаб соответствующих переменных. Часть процедуры основана на методе акселерации, разработанном Форситом и Моцкиным [17] для использования его в алгоритме наискорейшего подъема.

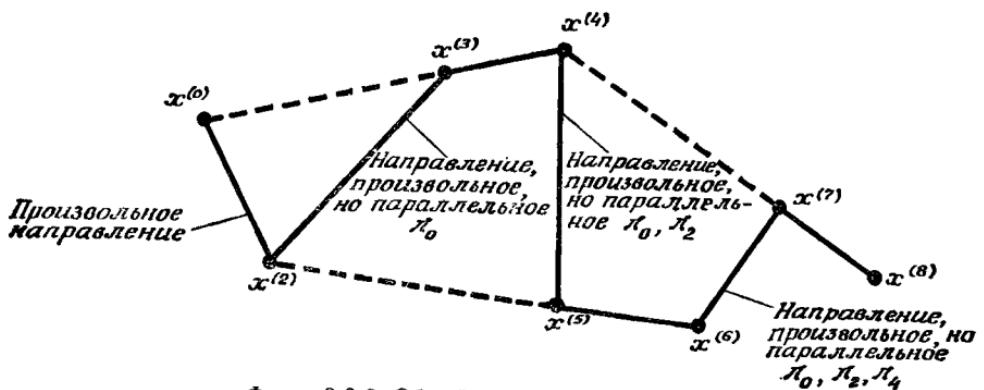
Общий партан-алгоритм по существу заключается в следующем (фиг. 3.3.3). Пусть $\mathbf{x}^{(0)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \dots$ — последовательность вектор-



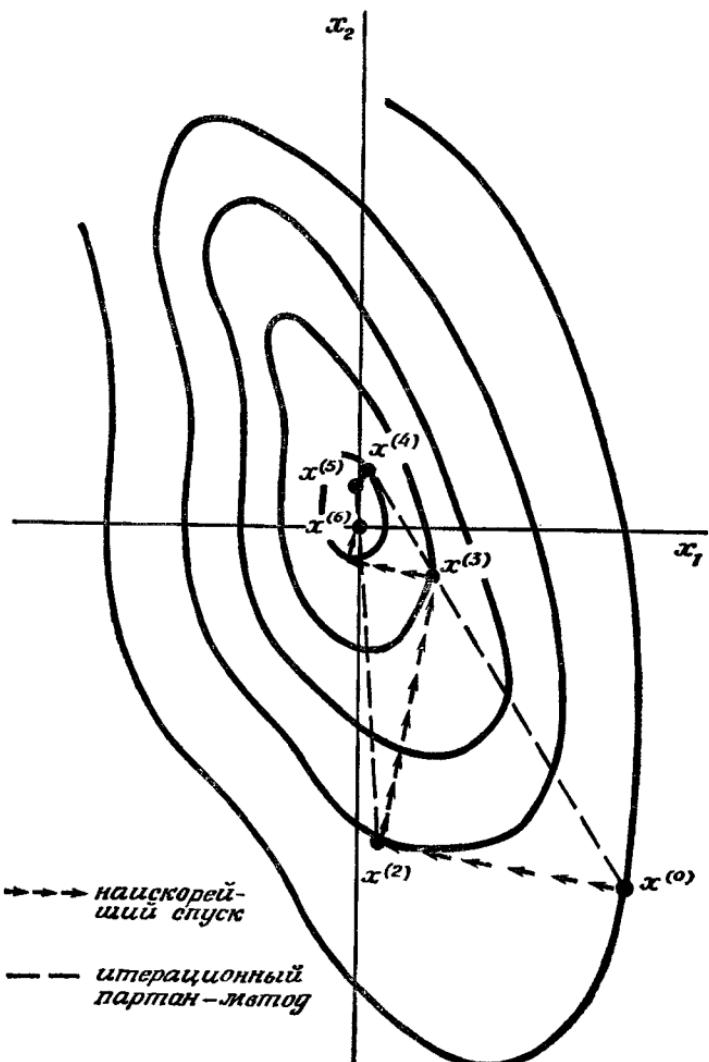
Ф и г. 3.3.2. Метод параллельных касательных (партан-метод) в двумерном случае.

ров \mathbf{x} в E^n , π_k — плоскость, касательная к линии уровня $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$. Каждый шаг делается в направлении минимума $f(\mathbf{x})$. Из точки $\mathbf{x}^{(0)}$ (пронумерованной так для симметрии) нужно двигаться вдоль ломаной линии $\mathbf{x}^{(0)} \mathbf{x}^{(2)} \mathbf{x}^{(3)} \mathbf{x}^{(4)} \dots$, на продолжении которой $\mathbf{x}^{(m-1)} \mathbf{x}^{(m)}$ точка $\mathbf{x}^{(m)}$ представляет собой точку минимума. Начальное направление $\mathbf{x}^{(0)} \mathbf{x}^{(2)}$ произвольно; направление $\mathbf{x}^{(2)} \mathbf{x}^{(3)}$ произвольно, но параллельно π_0 ; затем берется $\mathbf{x}^{(4)}$ коллинеарно $\mathbf{x}^{(0)}$ и $\mathbf{x}^{(3)}$. Повторяя процедуру для $k = 2, 3, \dots$, проводится $\mathbf{x}^{(2k)} \mathbf{x}^{(2k+1)}$ параллельно $\pi_0, \pi_2, \pi_4, \dots, \pi_{2k-2}$, а $\mathbf{x}^{(2k+2)}$ берется коллинеарно $\mathbf{x}^{(2k-2)}$ и $\mathbf{x}^{(2k+1)}$. В работе Ша, Бюлера и Кемпторна показано, что если $f(\mathbf{x})$ — квадратичная n -мерная функция, имеющая единственный минимум, т. е. положительно определенная квадратичная форма или ее монотонная функция, то общий партан-алгоритм приводит к минимуму в точке $\mathbf{x}^{(2n)}$ или ранее. Таким образом, этот метод обладает свойством квадратичного окончания.

Направления, полученные с помощью общего партан-алгоритма, являются сопряженными, что можно показать для случая квадратичной целевой функции ($f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{H} \mathbf{x}$) n независимых переменных. Действительно, градиент $f(\mathbf{x})$ равен $\nabla f(\mathbf{x}) = \mathbf{H} \mathbf{x}$. Поскольку путь оптимизации из $\mathbf{x}^{(0)}$ проходит вдоль вектора $(\mathbf{x}^{(0)} - \mathbf{x}^{(2)})$



Ф и г. 3.3.3. Общий партан-алгоритм.



Ф и г. 3.3.4. Сравнение траекторий метода наискорейшего спуска, итерационного и модифицированного партан-метода.

(фиг. 3.3.4), пока не будет достигнут минимум $f(\mathbf{x})$ (точка $\mathbf{x}^{(2)}$), вектор $(\mathbf{x}^{(0)} - \mathbf{x}^{(2)})$ параллелен π_2 , касательной плоскости в $\mathbf{x}^{(2)}$. Кроме того, вектор $(\mathbf{x}^{(0)} - \mathbf{x}^{(2)})$ перпендикулярен градиенту $f(\mathbf{x})$ в точке $\mathbf{x}^{(2)}$, $\nabla f(\mathbf{x}^{(2)}) = \mathbf{Hx}^{(2)}$. Следовательно,

$$(\mathbf{x}^{(0)} - \mathbf{x}^{(2)})^T \nabla f(\mathbf{x}^{(2)}) = (\mathbf{x}^{(0)} - \mathbf{x}^{(2)})^T \mathbf{Hx}^{(2)} = 0,$$

$$\nabla^T f(\mathbf{x}^{(2)}) (\mathbf{x}^{(0)} - \mathbf{x}^{(2)}) = (\mathbf{x}^{(2)})^T \mathbf{H} (\mathbf{x}^{(0)} - \mathbf{x}^{(2)}) = 0,$$

так что $(\mathbf{x}^{(0)})^T \mathbf{Hx}^{(2)} = (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(2)}$. Продолжая построение, получаем

$$\begin{aligned} (\mathbf{x}^{(0)})^T \mathbf{Hx}^{(2)} &= (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(2)} = (\mathbf{x}^{(4)})^T \mathbf{Hx}^{(2)} = \dots = (\mathbf{x}^{(2k)})^T \mathbf{Hx}^{(2)} = \\ &= (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(0)} = (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(2)} = (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(4)} = \\ &= \dots = (\mathbf{x}^{(2)})^T \mathbf{Hx}^{(2k)}. \end{aligned}$$

В общем случае все эти величины равны $(\mathbf{x}^{(2j)})^T \mathbf{Hx}^{(2k)}$. Кроме того, из построения требуется, чтобы вектор $(\mathbf{x}^{(2)} - \mathbf{x}^{(3)})$ (фиг. 3.3.4) был параллелен π_0 , или

$$(\mathbf{x}^{(2)} - \mathbf{x}^{(3)})^T \nabla f(\mathbf{x}^{(0)}) = 0.$$

Аналогичный анализ приводит к

$$\mathbf{x}^{(2j)} \mathbf{Hx}^{(2k)} = \mathbf{x}^{(2j)} \mathbf{Hx}^{(2k+1)},$$

$$j = 0, 1, \dots, k-1, \quad k = 1, 2, \dots, n-1.$$

Особый интерес представляют следующие разности:

$$(\mathbf{x}^{(k+2)})^T \mathbf{Hx}^{(2n)} - (\mathbf{x}^{(k+2)})^T \mathbf{Hx}^{(2n-2)} = 0,$$

$$(\mathbf{x}^{(k)})^T \mathbf{Hx}^{(2n)} - (\mathbf{x}^{(k)})^T \mathbf{Hx}^{(2n-2)} = 0.$$

Вычитание второго уравнения из первого дает

$$(\mathbf{x}^{(k+2)} - \mathbf{x}^{(k)})^T \mathbf{H} (\mathbf{x}^{(2n)} - \mathbf{x}^{(2n-2)}) = 0. \quad (3.3.15)$$

Уравнение (3.3.15) показывает, что направления $(\mathbf{x}^{(2)} - \mathbf{x}^{(0)})$, $(\mathbf{x}^{(4)} - \mathbf{x}^{(2)})$, ... вплоть до $(\mathbf{x}^{(2n)} - \mathbf{x}^{(2n-2)})$ взаимно сопряжены.

В партан-методе наискорейшего спуска направления, произвольные в общем партан-методе, выбираются следующим образом. В точках $\mathbf{x}^{(0)}, \mathbf{x}^{(2)}, \mathbf{x}^{(4)}, \dots, \mathbf{x}^{(2k)}$ нужно двигаться в направлении отрицательного градиента. Этот подход согласуется со случаем, когда $f(\mathbf{x})$ квадратична и при определении первых производных не имеют места ошибки. При этом партан-метод наискорейшего спуска имеет конечную сходимость. Следовательно, в выборе некоторых направлений уже отсутствует произвольность, и процедура становится инвариантной относительно поворота осей, т. е. преобразований вида $y = \mathbf{Ux}$, где \mathbf{U} — ортогональная матрица. Тем не менее выбор масштаба в партан-методе наискорейшего спуска оказывает влияние на промежуточные шаги.

Ниже приводятся операции алгоритма *итерационного партан-метода*:

1. Определяется направление отрицательного градиента (наискорейший спуск) в точке $x^{(0)}$.

2. Определяется точка минимума $x^{(2)}$ функции $f(x)$ вдоль направления отрицательного градиента из $x^{(0)}$.

3. Определяется направление отрицательного градиента в точке $x^{(2)}$.

4. Определяется точка минимума $x^{(3)}$ функции $f(x)$ вдоль отрицательного градиента из $x^{(2)}$.

5. Точки $x^{(0)}$ и $x^{(3)}$ соединяются прямой, вдоль которой определяется местонахождение минимума $f(x)$; эта точка обозначается через $x^{(4)}$.

6. Процедура повторяется с начальной точкой $x^{(4)}$.

Если задача содержит n переменных, то следует проделать n градиентных шагов, а затем соединить $x^{(0)}$ с $x^{(n+1)}$. Показанные на фиг. 3.3.4 шаги из $x^{(0)}$ в $x^{(2)}$, из $x^{(2)}$ в $x^{(3)}$, из $x^{(4)}$ в $x^{(5)}$ и из $x^{(5)}$ в $x^{(6)}$ (не показан) являются шагами в направлении наискорейшего спуска.

Итерационный партан-метод оказался значительно менее эффективным по сравнению с *модифицированным партан-методом*, который начинается пятью первыми шагами итерационного партан-метода, а затем продолжается следующим образом (для случая двух переменных):

6'. Находится направление отрицательного градиента в точке $x^{(4)}$.

7. Определяется точка минимума $x^{(5)}$ функции $f(x)$ вдоль направления отрицательного градиента.

8. Точки $x^{(2)}$ и $x^{(5)}$ соединяются прямой, вдоль которой определяется местонахождение минимума; эта точка обозначается через $x^{(6)}$.

9. Шаг 6' повторяется с использованием точки, определенной на шаге 8, вместо $x^{(4)}$. В каждом цикле в точках $x^{(2k)}$, $k = 2, 3, \dots$, используется направление наискорейшего спуска, а на шаге акселерации точки $x^{(2k-2)}$ соединяются с $x^{(2k+1)}$. Эффективность партан-алгоритма оценивается в гл. 5.

3.3.4. МЕТОД ЗАУТЕНДАЙКА (МЕТОД ПРОЕКЦИИ)

Заутендейк [18] предложил алгоритм, использующий сопряженные направления, полученные с помощью проектирующей матрицы. Хотя проектирующие матрицы описываются в разд. 6.3, можно привести здесь основные шаги этого алгоритма:

1. Начать в $\mathbf{x}^{(0)}$ и положить проектирующую матрицу $\mathbf{P}^{(0)} = \mathbf{I}$.
2. Для k -го шага проектирующая матрица вычисляется следующим образом:

$$\begin{aligned}\mathbf{P}^{(k)} &= \mathbf{I} - \mathbf{G}^{(k)} [(\mathbf{G}^{(k)})^T \mathbf{G}^{(k)}]^{-1} (\mathbf{G}^{(k)})^T = \\ &= \mathbf{P}^{(k-1)} - \mathbf{P}^{(k-1)} \Delta \mathbf{g}^{(k)} [(\Delta \mathbf{g}^{(k)})^T \mathbf{P}^{(k-1)} \Delta \mathbf{g}^{(k)}]^{-1} (\Delta \mathbf{g}^{(k)}) \mathbf{P}^{(k-1)}. \quad (3.3.16)\end{aligned}$$

3. Если $\mathbf{P}^{(k)} \nabla f(\mathbf{x}^{(k)}) \neq 0$, положить $\mathbf{s}^{(k)} = -\mathbf{P}^{(k)} \nabla f(\mathbf{x}^{(k)})$ и минимизировать $f(\mathbf{x})$ в направлении $\mathbf{s}^{(k)}$; точка минимума $\mathbf{x}^{(k+1)}$. Повторить шаг 2. После того как пройдено n направлений поиска, начать снова с шага 1 при $\mathbf{x}^{(0)} = \mathbf{x}^{(n)}$.

4. Если $\mathbf{P}^{(k)} \nabla f(\mathbf{x}^{(k)}) = 0$ и $\nabla f(\mathbf{x}^{(k)}) = 0$, закончить поиск.

5. Если $\mathbf{P}^{(k)} \nabla f(\mathbf{x}^{(k)}) = 0$, а $\nabla f(\mathbf{x}^{(k)}) \neq 0$, снова начать с шага 1, положив $\mathbf{x}^{(0)} = \mathbf{x}^{(k)}$.

Так как после проведения n итераций $\mathbf{P}^{(n)} = 0$, то должен быть начат новый цикл итераций. Поскольку первым направлением поиска при повторном цикле является направление наискорейшего спуска, можно показать [19], что этот алгоритм минимизирует квадратичную функцию, имеющую положительно определенную матрицу Гессе, не более чем за n этапов. Полезно сравнить уравнение (3.3.16) с (3.4.11).

3.3.5. МНОГОПАРАМЕТРИЧЕСКИЙ ПОИСК

В работе Миля и Кентрелла [20] предложен метод поиска, основанный на использовании двух подбираемых параметров для минимизации $f(\mathbf{x})$ в каждом из направлений поиска. В этом алгоритме последовательность шагов определяется формулой

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda_0^{(k)} \nabla f(\mathbf{x}^{(k)}) + \lambda_1^{(k)} \Delta \mathbf{x}^{(k-1)}. \quad (3.3.17)$$

На каждом шаге целевая функция $f(\mathbf{x}^{(k)} - \lambda_0 \nabla f(\mathbf{x}^{(k)}) + \lambda_1 \Delta \mathbf{x}^{(k-1)})$ минимизируется как по λ_0 , так и по λ_1 , а затем вычисляется $\mathbf{x}^{(k+1)}$ по формуле (3.3.17). При этом можно показать, что $\nabla^T f(\mathbf{x}^{(k)}) \times \nabla f(\mathbf{x}^{(k+1)}) = 0$, $\nabla^T f(\mathbf{x}^{(k+1)}) \Delta \mathbf{x}^{(k+1)} = 0$ и $\nabla^T f(\mathbf{x}^{(k+1)}) \Delta \mathbf{x}^{(k)} = 0$.

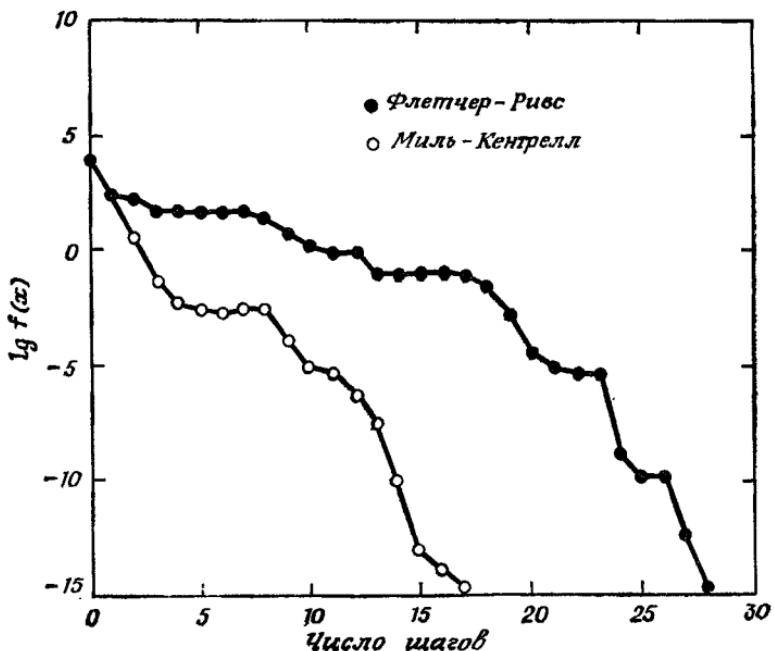
На первом шаге $\Delta \mathbf{x}^{(k-1)} = 0$, а $\mathbf{x}^{(0)}$ должно быть задано. На k -м шаге:

1. Вычисляются $\mathbf{x}^{(k)}$, $\nabla f(\mathbf{x}^{(k)})$ и $\Delta \mathbf{x}^{(k-1)} = \mathbf{x}^{(k)} - \mathbf{x}^{(k-1)}$.

2. Пользуясь одним из эффективных способов двумерного поиска типа описанных в этой и следующей главе, находятся с требуемой точностью $\lambda_0^{(k)}$ и $\lambda_1^{(k)}$.

3. По уравнению (3.3.17) вычисляют $x^{(k+1)}$ и переходят к п. 1.
4. Каждый $(n + 1)$ -й шаг начинается с $\Delta x^{(k-1)} = 0$.
5. Процесс заканчивается, когда $|\Delta f(x)| < \varepsilon$.

Для квадратичной функции приведенный выше алгоритм совпадает с алгоритмом Флетчера — Ривса, однако он требует большего



Ф и г. 3.3.5. Алгоритм Миля и Кентрелла двухпараметрического поиска в случае функции VI из табл. 5.2.1.

времени, так как на каждом шаге проводится двумерный поиск. В случае неквадратичных функций при использовании алгоритма Флетчера — Ривса выбирается в сущности на каждом шаге наилучшее значение параметра λ_0 при постоянном отношении λ_1/λ_0 , тогда как алгоритм Миля и Кентрелла оптимизирует как λ_0 , так и λ_1 . На фиг. 3.3.5 приведены результаты проведенного Милем и Кентреллом сравнения их алгоритма с разновидностью метода Ньютона для двумерного поиска при оптимизации целевой функции VI из табл. 5.2.1. Значения времени реализации алгоритмов Миля — Кентрелла и Флетчера — Ривса на ЭВМ Барроуз Б 5500 составляли 8,8 и 11,9 с соответственно.

Крэгг и Леви [21] распространяли метод двухпараметрического поиска на случай большего числа параметров, когда на каждом шаге осуществляется поиск более высокой размерности с целью получения значений параметров, минимизирующих $f(x)$ в данном

Таблица 3.3.1

Сравнение алгоритмов многопараметрической оптимизации
при уменьшении $f(x)$ до 10^{-13} [21]

Метод	Раздел данной книги, где описан метод	Функция I 1)		Функция VI 1)		Функция X 1)	
		число шагов	время 2)	число шагов	время 2)	число шагов	время 2)
Градиентный	3.1	Неудачен для решения задачи					
Флетчера — Ривса	3.3.2	29	1,2	29	2,4	68	14,0
Дэвидона — Флетчера — Пауэлла	3.4.2	21	1,1	39	4,5	30	6,7
Миля — Кентрелла (двуухпараметрический) ³⁾	3.3.5	2	0,6	18	2,0	32	12,8
Крэгга — Леви (четырехпараметрический) ⁴⁾	3.3.5	2	0,6	4	1,4	7	5,0
Ньютона, неисправляемый	3.2	6	0,1	Сходится к неоптимальной стационарной точке		25	1,6
Ньютона, исправляемый	3.2	21	0,3	39	1,5	25	1,6

1) См. табл. 5.2.1.

2) Время в секундах на ЭВМ Барроуз Б 5500.

3) Использует метод Ньютона, модифицированный так, чтобы матрица Гессе была положительно определенной при оптимизации параметров λ .

4) В случае функции Розенброка (функция I) применялись только два параметра.

направлении. Каждый следующий вектор x вычисляется по формуле

$$x^{(k+1)} = x^{(k)} - \lambda_0^{(k)} \nabla f(x^{(k)}) + \sum_{i=1}^m \lambda_i^{(k)} \Delta x^{(i-1)} \quad (3.3.18)$$

при $m \leq n - 1$. На начальных шагах алгоритма неизвестные $\Delta x^{(i)}$ можно положить равными нулю. В табл. 3.3.1 проводится сравнение некоторых алгоритмов с алгоритмом Крэгга — Леви, в котором для проведения $(m + 1)$ -мерного поиска λ был использован модифицированный метод Ньютона. Тот факт, что для поиска был использован метод Ньютона, безусловно, способствовал большей эффективности соответствующих алгоритмов. В работе, к сожалению, не приводятся данные о количестве потребовавшихся вычислений целевой функции.

3.4. МЕТОДЫ ПЕРЕМЕННОЙ МЕТРИКИ

Существует класс методов, называемых методами *переменной метрики*¹⁾, *квазиньютоновскими* или *градиентными* с большим шагом, которые аппроксимируют матрицу Гессе или обратную к ней, но используют для этого только первые производные. В большинстве этих методов применяются сопряженные направления, хотя в некоторых это не делается. При использовании методов переменной метрики новый вектор \mathbf{x} вычисляется по вектору предыдущего шага с помощью уравнения, аналогичного уравнениям (3.1.3) и (3.2.4а):

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \hat{\mathbf{s}}^{(k)} = \mathbf{x}^{(k)} - \lambda^{*(k)} \boldsymbol{\eta}(\mathbf{x}^{(k)}) \nabla^f(\mathbf{x}^{(k)}), \quad (3.4.1)$$

где матрица $\boldsymbol{\eta}(\mathbf{x}^{(k)})$, которую иногда называют матрицей направлений, представляет собой аппроксимацию $\mathbf{H}^{-1}(\mathbf{x})$. В уравнениях разд. 3.1 $\boldsymbol{\eta}(\mathbf{x}^{(k)})$ является единичной матрицей, а в разд. 3.2 $\boldsymbol{\eta}(\mathbf{x}^{(k)})$ представляет собой матрицу, обратную матрице Гессе целевой функции $\mathbf{H}^{-1}(\mathbf{x})$. Однако при использовании $\mathbf{H}^{-1}(\mathbf{x})$ необходимо было точно вычислять вторые частные производные $f(\mathbf{x})$ и обращать матрицу $\mathbf{H}(\mathbf{x})$, тогда как в методах переменной метрики для вычисления $\boldsymbol{\eta}(\mathbf{x}^{(k)})$ используются различные соотношения, не требующие ни того, ни другого.

Вспомним полезное соотношение (3.3.10), связывающее $\mathbf{x}^{(k+1)}$ и $\mathbf{x}^{(k)}$, для случая квадратичной целевой функции (или квадратичной аппроксимации целевой функции)

$$\nabla^f(\mathbf{x}^{(k+1)}) - \nabla^f(\mathbf{x}^{(k)}) = \mathbf{H}(\mathbf{x}^{(k)}) (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}). \quad (3.4.2a)$$

Умножая обе части этого уравнения на $\mathbf{H}^{-1}(\mathbf{x}^{(k)})$, получаем

$$\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)} = \mathbf{H}^{-1}(\mathbf{x}^{(k)}) [\nabla^f(\mathbf{x}^{(k+1)}) - \nabla^f(\mathbf{x}^{(k)})]. \quad (3.4.2b)$$

При этом если $f(\mathbf{x})$ квадратична, то $\mathbf{H}(\mathbf{x}^{(k)}) = \mathbf{H}$, т. е. постоянная матрица. Уравнение (3.4.2б) можно рассматривать как систему n линейных уравнений, содержащих n неизвестных параметров, которые нужно оценить для того, чтобы аппроксимировать $\mathbf{H}^{-1}(\mathbf{x})$ или $\mathbf{H}(\mathbf{x})$ при заданных значениях $f(\mathbf{x})$, $\nabla^f(\mathbf{x})$ и $\Delta\mathbf{x}$ на более ранних этапах поиска. Для решения этих линейных уравнений могут быть использованы различные методы, каждый из которых приводит к различным методам переменной метрики.

В довольно большой группе методов $\mathbf{H}^{-1}(\mathbf{x}^{(k+1)})$ аппроксимируется с помощью информации, полученной на k -м шаге:

$$\mathbf{H}^{-1}(\mathbf{x}^{(k+1)}) \approx \omega \boldsymbol{\eta}^{(k+1)} = \omega (\boldsymbol{\eta}^{(k)} + \Delta \boldsymbol{\eta}^{(k)}), \quad (3.4.3)$$

¹⁾ В более строгом смысле понятие «переменная метрика» относится лишь к тем методам, в которых делается преобразование независимых переменных, чтобы промасштабировать их более равномерным образом, но мы не будем употреблять этот термин в таком смысле.

где η — матрица, аппроксимирующая $H^{-1}(x)$. (Аргумент x матрицы η опущен в целях экономии места; таким образом, верхний индекс в записи η будет обозначать шаг.) $\Delta\eta^{(k)}$ представляет собой определяемую матрицу, а ω — масштабный множитель, константа, обычно равная единице. Выбор $\Delta\eta^{(k)}$ по существу определяет метод переменной метрики. Для обеспечения сходимости $\omega\eta^{(k+1)}$ должна быть положительно определенной и удовлетворять уравнению (3.4.2б) в том случае, когда она заменяет H^{-1} .

На $(k+1)$ -м шаге мы знаем $x^{(k)}$, $\nabla f(x^{(k)})$, $\nabla f(x^{(k+1)})$ и $\eta^{(k)}$ и хотим вычислить $\eta^{(k+1)}$, так чтобы удовлетворялось соотношение

$$\eta^{(k+1)} \Delta g^{(k)} = \frac{1}{\omega} \Delta x^{(k)}. \quad (3.4.2в)$$

Пусть $\Delta\eta^{(k)} = \eta^{(k+1)} - \eta^{(k)}$. Тогда уравнение

$$\Delta\eta^{(k)} \Delta g^{(k)} = \frac{1}{\omega} \Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)} \quad (3.4.2г)$$

нужно разрешить относительно $\Delta\eta^{(k)}$. Прямой подстановкой результата можно показать, что уравнение (3.4.2г) имеет следующее решение:

$$\Delta\eta^{(k)} = \frac{1}{\omega} \frac{\Delta x^{(k)} y^T}{y^T \Delta g^{(k)}} - \frac{\eta^{(k)} \Delta g^{(k)} z^T}{z^T \Delta g^{(k)}}, \quad (3.4.4)$$

где y и z — произвольные векторы размерности $n \times 1$. Если для $\omega = 1$ выбирается специальная линейная комбинация двух направлений $\Delta x^{(k)}$ и $\eta^{(k)} \Delta g^{(k)}$, а именно

$$y = z = \Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)},$$

то используем алгоритм Бройдена (описанный в разд. 3.4.1); если же берется

$$y = \Delta x^{(k)}, \quad z = \eta^{(k)} \Delta g^{(k)},$$

то матрицу $\eta^{(k+1)}$ вычисляем с помощью алгоритма Дэвидона — Флетчера — Пауэлла (описанного в разд. 3.4.2). Поскольку y и z — произвольные векторы, то оказываются допустимыми и другие возможности, которые будут рассмотрены в последующих разделах. Если шаги $\Delta x^{(k)}$ определяются последовательно путем минимизации $f(x)$ в направлении $s^{(k)}$, то все методы, с помощью которых вычисляют симметрическую матрицу $\eta^{(k+1)}$, удовлетворяющую (3.4.2в), дают направления, являющиеся взаимно сопряженными (в случае квадратичной целевой функции).

3.4.1. СЛУЧАЙ, КОГДА $\Delta\eta^{(k)}$ ИМЕЕТ РАНГ 1

Бройден [22], описывая методы решения систем линейных уравнений, показал, что если $\Delta\eta^{(k)}$ оказывается симметрической матрицей ранга 1 и должно удовлетворяться соотношение $\eta^{(k+1)}\Delta g^{(k)} = \Delta x^{(k)}$, то единственным возможным выбором $\Delta\eta^{(k)}$ является

$$\Delta\eta^{(k)} = \frac{[(\Delta x^{(k)}) - \eta^{(k)}(\Delta g^{(k)})] \cdot [(\Delta x^{(k)}) - \eta^{(k)}(\Delta g^{(k)})]^T}{[(\Delta x^{(k)}) - \eta^{(k)}(\Delta g^{(k)})]^T (\Delta g^{(k)})}, \quad (3.4.5)$$

где в целях экономии места мы положили

$$(\Delta x^{(k)}) = x^{(k+1)} - x^{(k)},$$

$$(\Delta g^{(k)}) = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)}).$$

В простейшем алгоритме этого типа минимизация начинается с выбора начальной точки $x^{(0)}$ и некоторого $\eta^{(0)} > 0$; затем последовательно применяются уравнения (3.4.1), (3.4.3) и (3.4.5) до тех пор, пока, скажем, $\|\nabla f(x^{(k)})\| < \varepsilon$. Если для каждого направления поиска $\lambda^{(k)}$ представляет собой скаляр, минимизирующий $f(x)$ в этом направлении, то данный метод дает сопряженные направления поиска. Таким образом, при определенных ограничивающих условиях описанному алгоритму обеспечена сходимость. В частном случае, когда целевая функция n независимых переменных квадратична, так что несингулярная матрица Гессе H является постоянной, то можно доказать, что после n шагов $\eta^{(n)} = H^{-1}$, если $\eta^{(0)} > 0$, если $\eta^{(k+1)}$ вычисляется из уравнений (3.4.3) и (3.4.5), если $x^{(k+1)}$ вычисляется из уравнения (3.4.1) и, наконец, если $(\Delta x^{(k)})$ являются линейно независимыми направлениями. Одной из интересных особенностей методов ранга 1 является то, что λ (или λ^*) в уравнении (3.4.1) не обязательно должно быть параметром, минимизирующим $f(x)$. Бройден показал, что λ может быть произвольным параметром, пока не возникла сингулярность η или знаменатель в правой части уравнения (3.4.5) не обратился в нуль. Это свойство позволяет отказаться от одномерного поиска, если можно найти адекватный альтернативный метод определения λ (см. разд. 3.4.5). В работе Голдфарба [23] приводятся теоремы, детализирующие достаточные условия сходимости алгоритмов ранга 1. Следует отметить, что излагаемые в данном разделе методы не требуют обращения матриц.

В случае, когда целевая функция не является квадратичной, применение уравнения (3.4.5) может привести к следующим нежелательным явлениям:

1. Матрица η может перестать быть положительно определенной. В этом случае необходимо обеспечить положительную

определенность матрицы $\eta^{(k+1)}$ с помощью одного из методов, отмеченных в разд. 3.2.

2. Вычисляемая величина $\Delta\eta^{(k)}$ может стать неограниченной (иногда даже в случае квадратичных функций вследствие ошибок округления).

3. Если $\Delta x^{(k)} = -\lambda^{*(k)} \eta(x^{(k)}) \nabla f(x^{(k)})$ случайно совпадает с направлением предыдущего этапа, матрица $\eta(x^{(k+1)})$ становится сингулярной или неопределенной. В алгоритме Брайдена это тоже будет иметь место, если в процессе определения направления поиска по уравнению (3.4.1) либо уравнение

$$\eta^{(k)} \Delta g^{(k)} = \Delta x^{(k)},$$

либо уравнение

$$(\eta^{(k)} \Delta g^{(k)} - \Delta x^{(k)})^T \Delta g^{(k)} = 0$$

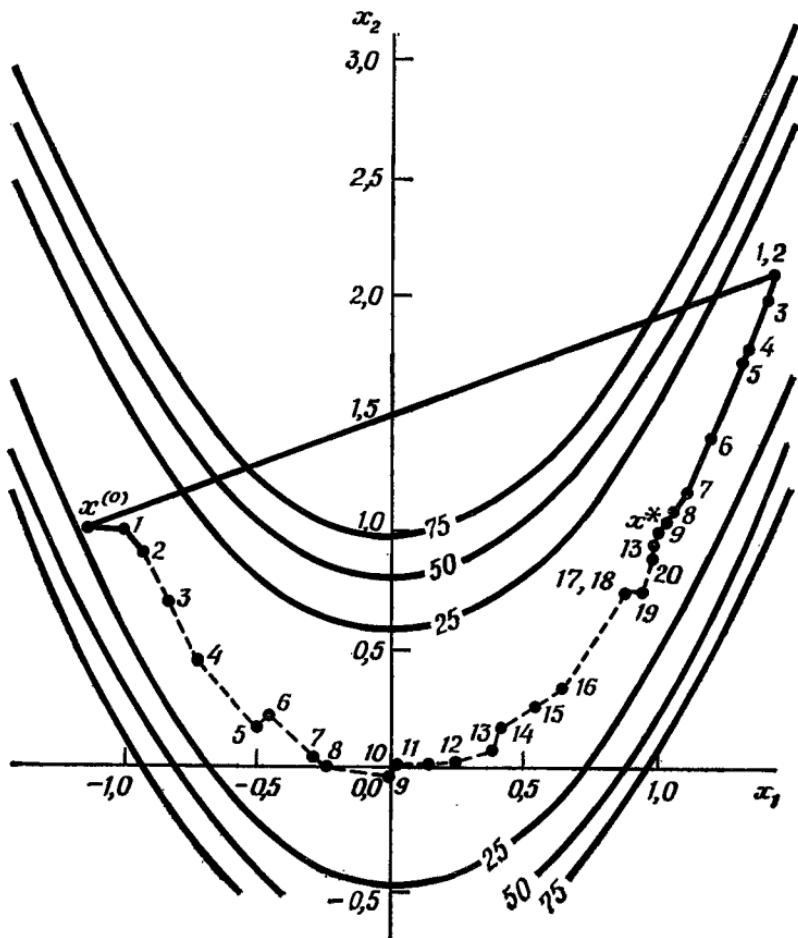
приводит к

$$\eta^{(k+1)} = \eta^{(k)}, \text{ т. е. к } \Delta\eta^{(k)} = 0.$$

На фиг. 3.4.1 изображены траектории поиска минимума функции Розенброка по алгоритму Брайдена с помощью одномерного поиска ДСК—Паузелла. Следует обратить внимание на то, что большой начальный шаг (шаг вдоль отрицательного градиента) в одномерном поиске приводит к проскакиванию левой части изогнутого оврага, тогда как при малом начальном шаге минимизация проходит полностью вдоль оврага. Машинная программа алгоритма Брайдена приводится в приложении Б.

Дэвидон [24] предложил по существу ту же схему вычисления $\Delta\eta^{(k)}$, что и уравнение (3.4.5), за исключением того, что $\Delta\eta^{(k)}$ умножается на некоторую функцию двух параметров для того, чтобы ограничить изменение $\eta^{(k)}$ на каждом шаге, с тем чтобы $\Delta\eta^{(k)}$ не было «слишком большим», и сохранить положительную определенность $\eta^{(k+1)}$. В этом алгоритме принимается $\lambda^* = 1$. Если после вычисления $\eta^{(k+1)}$ оказывается, что $f(x^{(k+1)}) > f(x^{(k)})$, то на следующей итерации $x^{(k+1)}$ заменяется на $x^{(k)}$. Хотя здесь не используется одномерный поиск, хороший выбор λ^* и упомянутых двух параметров оказывает значительное влияние на эффективность алгоритма. В работе Паузелла [25] приводится дополнительный список работ, в которых предлагается использовать уравнение (3.4.5) или его эквивалент.

Муртаг и Сарджент [26] показали, что одно из условий сходимости к стационарной точке заключается в том, что норма $\eta^{(k)}$ должна быть ограничена сверху, и снизу. Что касается нижней границы, то можно избежать прямого вычисления нормы матрицы, используя более сильное условие по отношению к уже вычислен-



Ф и г. 3.4.1. Траектории поиска при минимизации функции Розенброка с помощью алгоритма Бройдена (числа обозначают этапы, т. е. различные направления поиска).

— большой начальный шаг в выбранном направлении поиска; — — — малый начальный шаг в выбранном направлении поиска.

ным векторам:

$$\frac{\|\eta^{(k)} \nabla f(x^{(k)})\|}{\|\nabla f(x^{(k)})\|} \geq \rho_1,$$

где ρ_1 (здесь и ниже) — константы. Другое условие сходимости заключается в ограничении угла между направлением поиска и направлением наискорейшего спуска так, чтобы функция $f(x)$ уменьшалась при $\lambda^* > 0$ (конечно, при $\eta^{(k)} > 0$)¹⁾:

$$|\nabla^T f(x^{(k)}) \eta^{(k)} \nabla f(x^{(k)})| \geq \rho_2 \|\nabla f(x^{(k)})\| \|\eta^{(k)} \nabla f(x^{(k)})\|.$$

¹⁾ Напомним, что у автора $\eta^{(k)} > 0$ означает положительную определенность матрицы. — Прим. перев.

В случае, если эти, а также еще несколько менее значительных условий удовлетворены, можно выбрать λ^* таким, что

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)}) \geq \rho_3 \lambda^{*(k)} \nabla^T f(\mathbf{x}^{(k)}) \boldsymbol{\eta}^{(k)} \nabla f(\mathbf{x}^{(k)}).$$

Основываясь на этих концепциях, Муртаг и Сарджент предложили следующий алгоритм.

На k -м этапе:

1. Положить $\lambda^{*(k)} = 1$.
2. Сделать шаг, используя уравнение (3.4.1).
3. Если $\|\nabla f(\mathbf{x}^{(k)})\| \leq v_1$, продолжить одномерный поиск минимума.
4. Провести проверку (тест) выполнения неравенства

$$f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)}) \geq \rho_3 \lambda^{*(k)} \nabla^T f(\mathbf{x}^{(k)}) \boldsymbol{\eta}^{(k)} \nabla f(\mathbf{x}^{(k)}) > 0.$$

Если это неравенство не удовлетворяется, исключить один шаг из одномерного поиска и вернуться к п. 2. Если одномерный поиск сходится, но тест не удовлетворяется, уменьшить λ^* в 2 раза и вернуться к п. 2.

При одномерном поиске считается, что сходимость имеет место, когда либо $|\nabla^T f(\mathbf{x}^{(k+1)}) \Delta \mathbf{x}^{(k)}| < v_1$, либо $\Delta \lambda^* \|\boldsymbol{\eta}^{(k)} \nabla f(\mathbf{x}^{(k)})\| < v_1$, где $\Delta \lambda^*$ — изменение λ^* .

5. Проверить выполнение условия $\lambda^* > v_2$ и

$$\frac{\|\boldsymbol{\eta}^{(k)} \nabla f(\mathbf{x}^{(k)})\|}{\|\nabla f(\mathbf{x}^{(k)})\|} > \rho_1.$$

Увеличить масштаб $\lambda^{*(k)}$ и уменьшить масштаб $\boldsymbol{\eta}^{(k)}$, если это необходимо, чтобы выполнялись тесты 4 и 5.

6. Проверить, является ли все еще положительно определенной матрица направлений. Если нет, задать заново $\boldsymbol{\eta}^{(k)}$; в противном случае вычислить поправку для $\boldsymbol{\eta}^{(k)}$ с помощью формулы (3.4.5). Задать заново матрицу направления можно, положив $\boldsymbol{\eta}^{(k+1)} = \mathbf{I}$ или $\boldsymbol{\eta}^{(k+1)} = \boldsymbol{\eta}^{(k)}$.

3.4.2. МЕТОД ДЭВИДОНА — ФЛЕТЧЕРА — ПАУЭЛЛА

В хорошо известном методе Дэвидона [27], модифицированном Флетчером и Паузеллом [28], выбирается матрица $\Delta \boldsymbol{\eta}$, имеющая ранг 2. Здесь также не нужна операция обращения матрицы. Матрица направлений $\boldsymbol{\eta}$ вычисляется таким образом, чтобы для квадратичной целевой функции в пределе после n шагов она равнялась \mathbf{H}^{-1} . Исходная матрица $\boldsymbol{\eta}$ обычно выбирается в виде единичной матрицы $\boldsymbol{\eta}^{(0)} = \mathbf{I}$ (но может быть и любой симметрической положительно определенной матрицей), так что исходное направление ми-

нимизации — это направление наискорейшего спуска. Оценка элементов H^{-1} в точке x^* (экстремум) тем лучше, чем лучше мы выберем по сравнению с единичной матрицей исходную $\eta^{(0)}$, однако выбор $\eta^{(0)} = I$ определенно предпочтительнее приравнивания элементов $\eta^{(0)}$ значениям аналитических частных производных или их конечно-разностных приближений в начальной точке $x^{(0)}$. В ходе оптимизации имеет место постепенный переход от градиентного направления к ньютоновскому; при этом используются преимущества каждого из этих двух методов на соответствующем этапе. Доказательство сходимости данного алгоритма может быть приведено только для случая квадратичной целевой функции с положительно определенной матрицей Гессе [см. формулу (3.4.6)].

Соотношение для $\Delta\eta^{(k)}$ в алгоритме Дэвидона — Флетчера — Пьюзелла, как было отмечено ранее, можно получить путем подстановки

$$\mathbf{y}^{(k)} = \Delta \mathbf{x}^{(k)} \quad \text{and} \quad \mathbf{z}^{(k)} = \eta^{(k)} \Delta \mathbf{g}^{(k)}$$

в уравнение (3.4.4). Тогда имеем

$$\begin{aligned}\eta^{(k+1)} &= \eta^{(k)} + A^{(k)} - B^{(k)} = \\ &= \eta^{(k)} + \frac{(\Delta x^{(k)}) (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T (\Delta g^{(k)})} - \frac{\eta^{(k)} (\Delta g^{(k)}) (\Delta g^{(k)})^T (\eta^{(k)})^T}{(\Delta g^{(k)})^T \eta^{(k)} (\Delta g^{(k)})},\end{aligned}\quad (3.4.5a)$$

где обозначения те же, что и в формуле (3.4.5). Следует отметить что вторая и третья матрицы в правой части (3.4.5а) являются симметрическими, так что если матрица $\eta^{(k)}$ — симметрическая, то и $\eta^{(k+1)}$ будет симметрической.

Рекуррентное соотношение (3.4.5а) на практике вполне удовлетворительно, если:

- 1) ошибка при вычислении $\nabla f(x^{(k)})$ невелика;
 - 2) $\eta^{(k)}$ не становится «плохой».

Роль матрицы $A^{(k)}$ в формуле (3.4.5а) заключается в обеспечении того, чтобы $\eta \rightarrow H^{-1}$, тогда как матрица $B^{(k)}$ обеспечивает положительную определенность $\eta^{(k+1)}$ на всех этапах и в пределе исключает начальную матрицу $\eta^{(0)}$. Используем формулу (3.4.5а) на нескольких этапах, начиная с $\eta^{(0)}$:

$$\eta^{(1)} = \mathbf{I} + \mathbf{A}^{(0)} - \mathbf{B}^{(0)},$$

$$\eta^{(2)} = \eta^{(1)} + A^{(1)} - B^{(1)} = I + (A^{(0)} + A^{(1)}) - (B^{(0)} + B^{(1)}),$$

.....

$$\eta^{(k+l)} = \mathbf{I} + \sum_{i=0}^k \mathbf{A}^{(i)} - \sum_{i=0}^k \mathbf{B}^{(i)}.$$

В случае квадратичной функции сумма матриц $A^{(i)}$ должна равняться H^{-1} при $k = n - 1$, а сумма матриц $B^{(i)}$ строится так, чтобы она сократилась с матрицей, выбранной в качестве исходной матрицы $\eta^{(0)}$ (здесь единичной матрицей). Таким образом, метод Дэвидона — Флетчера — Пауэлла отражает до некоторой степени в текущем значении η всю предыдущую информацию.

Следует отметить, что в случае квадратичной целевой функции в алгоритме Дэвидона — Флетчера — Пауэлла используются сопряженные направления. Для того чтобы последнее направление $s^{(n-1)}$ было сопряжено по отношению ко всем предыдущим направлениям, должно выполняться равенство

$$(X^{(n-1)})^T H s^{(n-1)} = 0$$

или при $s^{(n-1)} = -\eta^{(n-1)} \nabla f(x^{(n-1)})$

$$(X^{(n-1)})^T H \eta^{(n-1)} \nabla f(x^{(n-1)}) = 0, \quad (3.4.6)$$

где X определяется в соответствии с уравнением (3.3.8). Уравнение (3.4.6) будет справедливо, если $H\eta^{(n-1)} = I$ или $\eta^{(n-1)} = H^{-1}$, поскольку при этом оно сводится к уравнению (3.3.8). Таким образом, метод Дэвидона — Флетчера — Пауэлла можно отнести к категории методов, использующих сопряженные направления. В случае общей целевой функции эффективность метода Дэвидона — Флетчера — Пауэлла является скорее следствием использования сопряженных направлений, чем близкой аппроксимации H^{-1} матрицей η .

Зная теперь, что в случае квадратичной функции направления поиска являются сопряженными, можно легко показать, что $\sum_{i=0}^{n-1} A^{(i)} = H^{-1}$. Заметим, что из уравнения (3.4.2б) следует, что $\Delta g^{(k)} = H \Delta x^{(k)}$. При этом числитель и знаменатель $A^{(k)}$ соответственно равны

$$(\Delta x^{(k)}) (\Delta x^{(k)})^T = (\lambda^{*(k)} s^{(k)}) (\lambda^{*(k)} s^{(k)})^T$$

и

$$(\Delta x^{(k)})^T \Delta g^{(k)} = (\lambda^{*(k)} s^{(k)})^T (H \lambda^{*(k)} s^{(k)}).$$

Следовательно,

$$\sum_{i=0}^{n-1} A^{(i)} = \sum_{i=0}^{n-1} \frac{s^{(i)} (s^{(i)})^T}{(s^{(i)})^T H s^{(i)}} = H^{-1}, \quad (3.4.7)$$

где правое равенство вытекает из уравнения (3.3.4а).

Хотя Дэвидон в данном направлении поиска использовал только один шаг длиной $\lambda^{*(k)}$, которая определяется с помощью куби-

ческой интерполяции между $\lambda^* = 0$ и

$$\lambda^* = \min \left\{ 1, \frac{2[f(x^{(k)}) - f_0]}{\nabla^T f(x^{(k)}) \eta^{(k)} \nabla f(x^{(k)})} \right\}, \quad (3.4.8)$$

а f_0 — наименьшее ожидаемое значение $f(x)$, в большинстве вариаций алгоритма Дэвидона функция минимизируется в каждом выбранном направлении поиска. Для определения минимума $f(x)$ по λ в данном направлении можно применить почти любую эффективную процедуру одномерного поиска (см. разд. 2.6). Очень важно, чтобы эта процедура была эффективной, поскольку относительно большая часть всего времени вычисления приходится на одномерный поиск.

Флетчер и Паузелл предложили выбирать первую длину из последовательности λ^* с помощью уравнения (3.4.8) либо положить $\lambda^{*(0)} = 1$. В гл. 5 при оценке эффективности этого метода используются два хорошо известных различных способа одномерного поиска. В приложении Б содержатся их соответствующие машинные программы.

Чувствительность метода Дэвидона — Флетчера — Паузелла к критерию окончания процесса в одномерном поиске оказалась меньшей, чем можно было ожидать, в отношении как времени оптимизации, так и числа вычислений функции. Например, в табл. 3.4.1 приведены число шагов, количество вычислений функции и значения функции Розенброка, начиная с точки $x^{(0)} = [-1,2 \ 1]^T$. Из таблицы ясно видно, что, хотя требуется значительно больше шагов, если одномерный поиск проведен недостаточно точно, тем не менее общее число вычислений функции и время, требуемое для минимизации функции Розенброка, приблизительно одинаковы для разной точности одномерного поиска, хотя, пожалуй, цифры несколько лучше в случае $\varepsilon = 10^{-3}$, чем для $\varepsilon = 10^{-1}$ или $\varepsilon = 10^{-5}$, где ε — критерий окончания одномерного поиска по отношению к $\lambda^{*(k)}$ [уравнение (3.4.1)]:

$$\left| \frac{\lambda^{*(k+1)} - \lambda^{*(k)}}{\lambda^{*(k+1)} + \lambda^{*(k)}} \right| < \varepsilon.$$

Опыт показал, что в некоторых задачах нельзя достичь минимума целевой функции с помощью методов переменной метрики, если степень точности одномерного поиска недостаточна; поэтому рекомендуется, чтобы точность одномерного поиска была по крайней мере эквивалентна точности, требуемой для окончания основного алгоритма. Цена этого в смысле времени и (или) числа вычислений функции относительно невелика, тогда как надежность любого из этих методов значительно увеличивается.

Флетчер и Паузелл предложили, чтобы минимизация заканчивалась в точке, где при вычислении как вектора $\eta^{(k)} \nabla f(x^{(k)})$, так и

Таблица 3.4.1

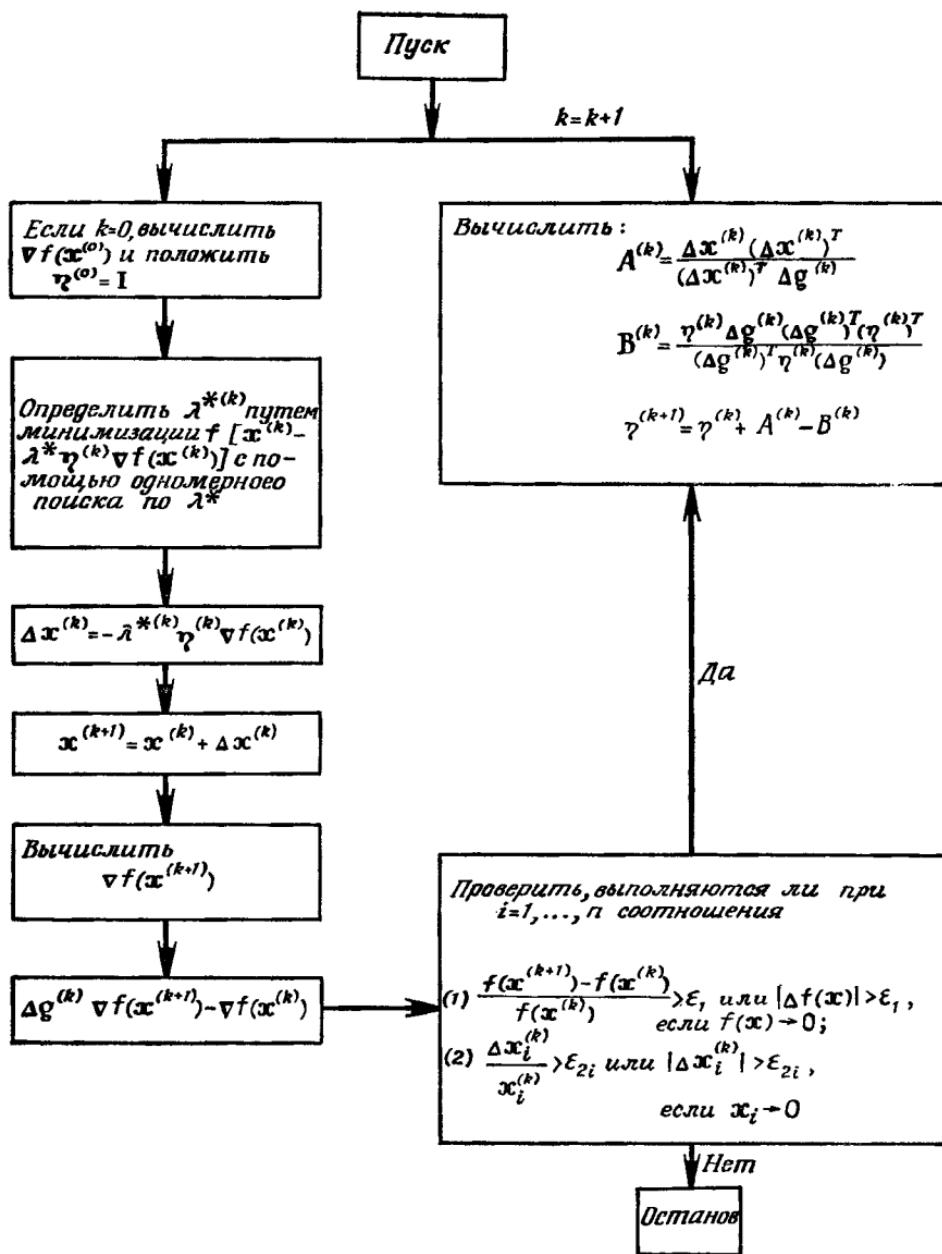
Число вычислений функции Розенброка в процессе ее минимизации до значения 10^{-10} при различных значениях критерия окончания одномерного поиска ε

$\varepsilon = 10^{-1}$			$\varepsilon = 10^{-3}$			$\varepsilon = 10^{-5}$		
шаг <i>k</i>	<i>n</i> ¹⁾	<i>f</i> (<i>x</i>) ²⁾	шаг <i>k</i>	<i>n</i> ¹⁾	<i>f</i> (<i>x</i>) ²⁾	шаг <i>k</i>	<i>n</i> ¹⁾	<i>f</i> (<i>x</i>) ²⁾
0	16	$1,4 \cdot 10^{-1}$	0	27	$1,9 \cdot 10^{-1}$	0	36	$1,9 \cdot 10^{-1}$
1	22	$2,3 \cdot 10^{-1}$	1	32	$1,9 \cdot 10^{-1}$	1	31	$1,9 \cdot 10^{-1}$
2	11	$1,6 \cdot 10^{-1}$	2	24	$1,2 \cdot 10^{-1}$	2	34	$1,2 \cdot 10^{-1}$
3	16	$1,5 \cdot 10^{-1}$	3	25	$8,2 \cdot 10^{-2}$	3	34	$7,6 \cdot 10^{-2}$
4	15	$1,5 \cdot 10^{-1}$	4	25	$4,3 \cdot 10^{-2}$	4	35	$4,2 \cdot 10^{-2}$
5	16	$1,4 \cdot 10^{-1}$	5	27	$2,2 \cdot 10^{-2}$	5	36	$2,1 \cdot 10^{-2}$
6	14	$1,4 \cdot 10^{-1}$	6	27	$1,1 \cdot 10^{-2}$	6	36	$9,5 \cdot 10^{-3}$
7	18	$1,0 \cdot 10^{-1}$	7	23	$3,9 \cdot 10^{-3}$	7	33	$3,2 \cdot 10^{-3}$
8	15	$1,0 \cdot 10^{-1}$	8	25	$8,6 \cdot 10^{-4}$	8	34	$6,1 \cdot 10^{-4}$
9	18	$6,6 \cdot 10^{-2}$	9	25	$1,7 \cdot 10^{-5}$	9	35	$2,6 \cdot 10^{-5}$
10	15	$6,3 \cdot 10^{-2}$	10	19	$2,0 \cdot 10^{-7}$	10	28	$4,5 \cdot 10^{-7}$
11	18	$2,5 \cdot 10^{-2}$	11	11	$8,8 \cdot 10^{-11}$	11	9	$4,6 \cdot 10^{-11}$
12	17	$2,0 \cdot 10^{-2}$						
13	18	$1,6 \cdot 10^{-2}$						
14	16	$7,2 \cdot 10^{-3}$						
15	13	$5,2 \cdot 10^{-3}$						
16	17	$4,1 \cdot 10^{-4}$						
17	13	$1,6 \cdot 10^{-4}$						
18	16	$6,6 \cdot 10^{-6}$						
19	6	$1,5 \cdot 10^{-6}$						
20	9	$7,6 \cdot 10^{-10}$						
21	1	$1,2 \cdot 10^{-10}$						

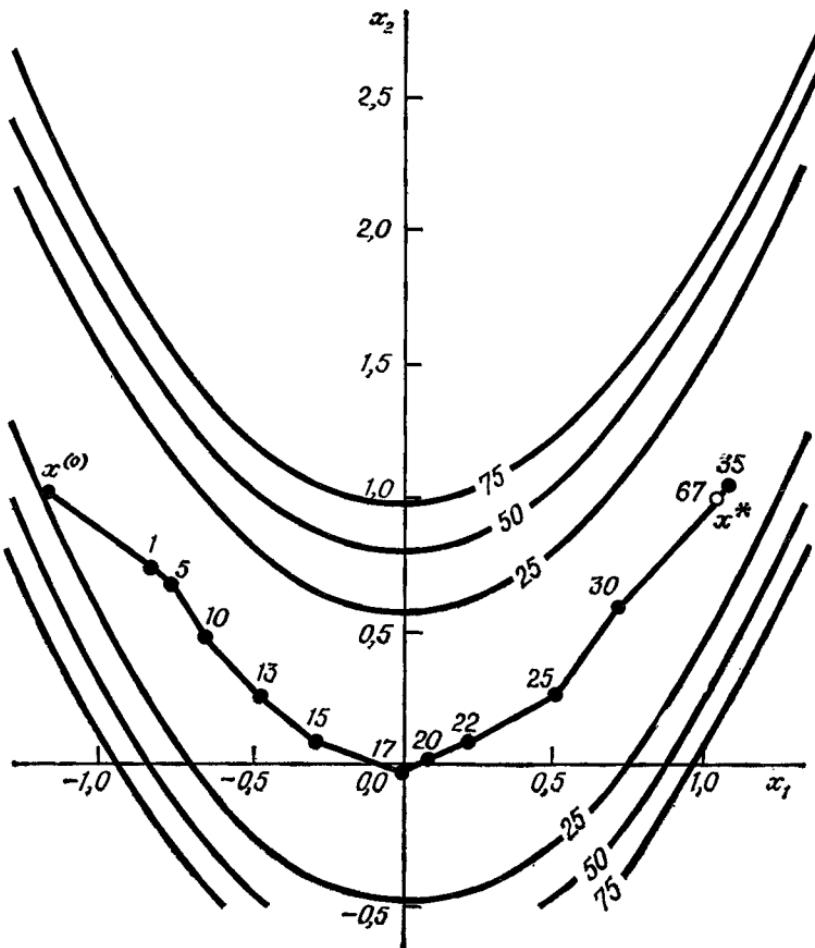
1) *n* — число вычислений функции на одном шаге.

2) Значение функции в конце шага.

	$\varepsilon = 10^{-1}$	$\varepsilon = 10^{-3}$	$\varepsilon = 10^{-5}$
Общее число вычислений функции	320	290	380
Относительное время	1,14	1,00	1,12



Ф и г. 3.4.2. Блок-схема метода Дэвидона — Флетчера — Пауэлла.



Ф и г. 3.4.3. Траектории поиска при минимизации функции Розенброка методом Дэвидона — Флетчера — Паузлла (числа указывают номера шагов, т. е. различных направлений поиска).

— большой начальный шаг в выбранном направлении поиска, $\lambda^* = 1$; — — — малый начальный шаг в выбранном направлении поиска, $\lambda^* = 10^{-3}$.

вектора $-\lambda^{*(k)} \eta^{(k)} \nabla f(\mathbf{x}^{(k)})$ выполняется один из следующих двух пунктов:

1. Каждая составляющая этих двух векторов меньше, чем некоторая наперед заданная величина.

2. Вычисленная длина ($\|\cdot\|$) каждого из этих векторов в точке минимума меньше наперед заданной величины.

В гл. 5 при сравнении различных методов вместо этих показателей успешно применены два других критерия окончания процесса, приведенные на фиг. 3.4.2, где представлена блок-схема метода Дэвидона — Флетчера — Паузлла. На фиг. 3.4.3 приведены

траектории алгоритма Дэвидона — Флетчера — Пауэлла для функции Розенброка при двух различных величинах начальных шагов в одномерном поиске $\lambda^* = 1$ и $\lambda^* = 10^{-3}$; эти траектории совершенно аналогичны приведенным на фиг. 3.4.1.

Пример 3.4.1. Метод Дэвидона — Флетчера — Пауэлла

Проиллюстрируем применение метода Дэвидона — Флетчера — Пауэлла на примере следующей задачи:

минимизировать $4(x_1 - 5)^2 + (x_2 - 6)^2$.

В соответствии с изложенным выше алгоритмом используем следующее рекуррентное соотношение:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \lambda^{*(k)} \boldsymbol{\eta}^{(k)} \nabla f(\mathbf{x}^{(k)}), \quad (a)$$

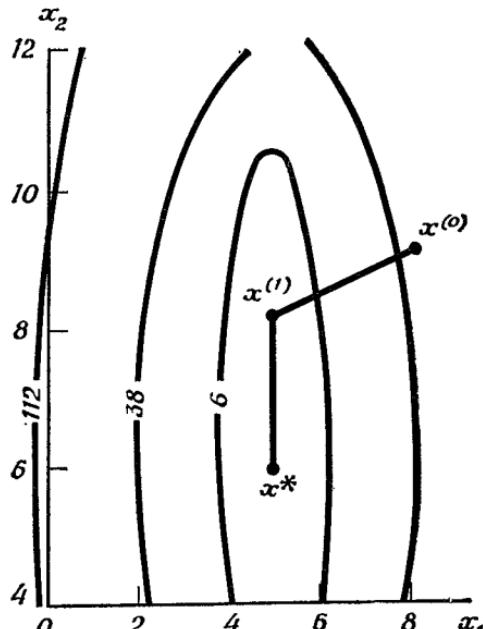
где

$$\nabla f(\mathbf{x}^{(k)}) = \begin{bmatrix} 8(x_1 - 5) \\ 2(x_2 - 6) \end{bmatrix}$$

и $\boldsymbol{\eta}^{(k)}$ задается уравнением (3.4.5а).

Пусть $\boldsymbol{\eta}^{(0)} = \mathbf{I}$ и начальный вектор $\mathbf{x}^{(0)} = [8 \ 9]^T$. Тогда новый вектор $\mathbf{x}^{(1)}$ вычисляется по формуле (а):

$$\begin{bmatrix} x^{(1)} \\ x^{(2)} \end{bmatrix} = \begin{bmatrix} 8 \\ 9 \end{bmatrix} - \lambda^{*(0)} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 24 \\ 6 \end{bmatrix}.$$



Фиг. П.3.4.1. Траектория поиска алгоритма Дэвидона — Флетчера — Пауэлла.

Поскольку целевая функция весьма проста, то для наглядности можно определить $\lambda^{*(0)}$ путем минимизации $f(\mathbf{x}^{(1)})$ по λ^* , используя аналитические методы вместо процедуры поиска:

$$f(\mathbf{x}^{(1)}) = 4[(8 - 24\lambda^*) - 5]^2 + [(9 - 6\lambda^*) - 6]^2, \quad \frac{df(\mathbf{x}^{(1)})}{d\lambda^*} = 0 = 51 - 390\lambda^*,$$

или $\lambda^{*(0)} = 0,1307$. Итак, на этом шаге $\mathbf{x}^{(1)} = [4,862 \ 8,215]^T$, а $f(\mathbf{x}) = 4,985$, как показано на фиг. П.3.4.1. В точке $\mathbf{x}^{(1)}$

$$\nabla f(\mathbf{x}^{(1)}) = [-1,108 \ 4,431]^T,$$

так что

$$(\Delta g)^{(0)} = [-25,108 \quad -1,569]^T.$$

Затем вычисляется $\eta^{(1)}$:

$$\begin{aligned} \eta^{(1)} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \frac{\begin{bmatrix} -3,13 & 0 \\ -0,785 & 0 \end{bmatrix} \begin{bmatrix} -3,13 & -0,785 \\ 0 & 0 \end{bmatrix}}{\begin{bmatrix} -3,13 & -0,785 \end{bmatrix} \begin{bmatrix} -25,108 \\ -1,569 \end{bmatrix}} - \\ &- \frac{\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -25,108 & 0 \\ -1,569 & 0 \end{bmatrix} \begin{bmatrix} -25,108 & -1,569 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}}{\begin{bmatrix} -25,108 & -1,569 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -25,108 \\ -1,569 \end{bmatrix}} = \\ &= \begin{bmatrix} 1,270 \cdot 10^{-1} & -3,149 \cdot 10^{-2} \\ -3,149 \cdot 10^{-2} & 1,0038 \end{bmatrix}. \end{aligned}$$

Теперь снова по уравнению (а) можно вычислить $x^{(2)}$:

$$\begin{bmatrix} x_1^{(2)} \\ x_2^{(2)} \end{bmatrix} = \begin{bmatrix} 4,862 \\ 8,215 \end{bmatrix} - \lambda^{*(1)} \begin{bmatrix} 1,270 \cdot 10^{-1} & 3,149 \cdot 10^{-2} \\ -3,149 \cdot 10^{-2} & 1,0038 \end{bmatrix} \begin{bmatrix} -1,108 \\ 4,431 \end{bmatrix}.$$

Как и ранее, $\lambda^{*(1)}$ получается путем минимизации $f(x^{(2)})$ по λ^* . Последующие шаги оптимизации сведены в табл. П.3.4.1а.

Таблица П.3.4.1а

Шаг k	$x_1^{(k)}$	$x_2^{(k)}$	$\frac{\partial f(x^{(k)})}{\partial x_1}$	$\frac{\partial f(x^{(k)})}{\partial x_2}$	$f(x^{(k)})$	$\lambda^{*(k)}$
0	8,000	9,000	24,000	6,000	45,000	0,1307
1	4,862	8,215	-1,108	4,431	4,985	0,4942
2	5,000	6,000	$3,81 \cdot 10^{-7}$	$2,55 \cdot 10^{-9}$	$9,06 \cdot 10^{-15}$	1,000
3	5,000	6,000	0	0	0	

Таблица П.3.4.1б

Шаг 0	Шаг 1
$\eta \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1,270 \cdot 10^{-1} & -3,149 \cdot 10^{-2} \\ -3,149 \cdot 10^{-2} & 1,0038 \end{bmatrix}$
Шаг 2	Шаг 3
$\begin{bmatrix} 1,250 \cdot 10^{-1} & -8,882 \cdot 10^{-16} \\ -8,882 \cdot 10^{-16} & 5,000 \cdot 10^{-1} \end{bmatrix}$	$\begin{bmatrix} 1,250 \cdot 10^{-1} & 1,387 \cdot 10^{-17} \\ -1,387 \cdot 10^{-17} & 5,000 \cdot 10^{-1} \end{bmatrix}$

В табл. П.3.4.1б приведены величины элементов матрицы η на каждом шаге поиска, которую можно сравнить с матрицей, обратной матрице Гессе H^{-1} , в точке $x^* = [5 \ 6]^T$:

$$H = \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix} \quad \text{и} \quad H^{-1} = \begin{bmatrix} \frac{1}{8} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

На практике оказалось, что в методе Дэвидона — Флетчера — Пауэлла и других методах переменной метрики могут иногда встречаться отрицательные шаги, или эти методы могут оканчиваться в нестационарной точке. Бард [29] показал, что такое течение процесса является следствием того, что матрица η становится сингулярной. Если матрица η становится почти сингулярной, то направления поиска могут быть выбраны так, как в случае, если бы в задаче были плохо выбраны масштабы переменных, несмотря на то что на самом деле это не так. «Почти сингулярности» можно избежать путем увеличения числа получаемых значащих цифр при вычислениях или путем масштабирования элементов вектора x с тем, чтобы сделать порядок диагональных элементов $A^{(0)}$ близким к единице. Если же эти операции недопустимы, т. е. если косинус угла между $\nabla f(x^{(k)})$ и $\eta^{(k)} \nabla f(x^{(k)})$ меньше, чем 10^{-5} , матрицу η можно перезадать в виде диагональной матрицы, у которой элемент η_{ii} представляет собой отношение i -го элемента $\Delta x^{(k)}$ к i -му элементу $\nabla f(x^{(k)})$. Были предложены и другие методы, в которых поиск начинается заново с пересмотренной матрицей направлений; несколько подобных примеров приведено в гл. 5.

Осталось рассмотреть последний вопрос: может ли метод Дэвидона — Флетчера — Пауэлла успешно применяться в том случае, когда оценивание производных осуществляется с помощью разностных схем? Если да, то исчезнет неудобство, заключающееся в необходимости получения аналитических формул для вычисления производных, и процедура Дэвидона приблизится к методам, не включающим вычисление производных (методам поиска). Здесь следует обратить внимание на то, что ошибки округления и аппроксимации при вычислениях могут сместить оценки производных.

Стюарт [30] показал, как компоненты градиента можно оценить с помощью отношений разностей, распространяя таким образом метод Дэвидона на задачи нелинейного программирования (без ограничений), в которых производные или вообще не могут быть найдены аналитически, или это нельзя сделать достаточно легко. Информация, получаемая в процессе минимизации, используется для определения оптимального размера шага в разностных

уравнениях. Хотя модифицированный метод Стюарта вычисления приближенных значений производных очень эффективен в некоторых задачах, он иногда либо совсем не годится, либо приводит к существенно большему числу вычислений функций для достижения той же степени точности, которая получается методом пере-

Таблица 3.4.2

Сравнение методов переменной метрики¹⁾, использующих разностные оценки производных, с алгоритмом Пауэлла на примере пяти тестовых задач
(Приведенные числа представляют собой количество вычислений функции.)

Метод	Задача 2 (приложение A)	Задача 29 (приложение A)	Задача 32 (приложение A)	Задача 34 (приложение A)	Задача VI, табл. 5 (приложение A)
Пауэлла ²⁾	262	136	700	86	291
Дэвидона — Флетчера — Пауэлла					
(a)	132	1625	305	218	702
(b)	121	1123	205	231	Не приводит к решению
Бройдена					
(a)	139	4336	167	230	1251
(b)	112	Не приводит к решению		176	Не приводит к решению

1) Включающих одномерный поиск ДСК-Пауэлла.

2) См. гл. 4.

(a) $\eta = 10^{-8}$, $m = 10^{-8}$, $\delta = 10^{-7}$.

(b) $\eta = 10^{-14}$, $m = 10^{-14}$, $\delta = 10^{-13}$.

менной метрики, использующим аналитические выражения для производных.

В табл. 3.4.2 сравниваются количества вычислений функции для алгоритма Пауэлла, одного из лучших алгоритмов, не использующих производные, и алгоритмов Дэвидона — Флетчера — Пауэлла и Бройдена, в которых аналитические выражения производных заменены оценками производных по методу Стюарта. В методе Стюарта требуется, чтобы пользователем были заданы следующие три произвольных параметра:

η — порядок значащих цифр, допускаемых ЭВМ;

m — верхняя граница отрицательного логарифма отношения ошибки текущей оценки производной к предыдущей оценке производной;

δ — величина шага, который должен быть использован при вычислении компонент первых двух градиентов.

Как η , так и t зависят от используемой вычислительной машины, тогда как δ связана с минимизируемой функцией. В табл. 3.4.2 приведены результаты для двух систем значений упомянутых параметров. В нее не включены данные для алгоритмов Дэвидона — Флетчера — Пауэлла и Брайдена, применяющих аналитические производные (эти данные можно найти в гл. 5), поскольку трудно решить, какой вес придавать трудоемкости операций по вычислению производных относительно операций по вычислению функций.

Результаты, полученные для случая изогнутого оврага (задача 34 из приложения А) и функции Розенброка (задача 2 из приложения А), отличаются от опубликованных Стюартом, но это и не удивительно, поскольку использовались различные δ и разные критерии окончания процесса. Во всех задачах, кроме задачи 2, начальный выбор $\delta = 10^{-7}$ давал оценки производных с малой начальной ошибкой, но в последующем ошибка становилась существенной. В одной задаче такая ошибка случайно уменьшила число вычислений функции до величины, меньшей, чем имеет место в аналитических процедурах. С другой стороны, при $\delta = 10^{-13}$ получали большие начальные ошибки. Поскольку компоненты двух начальных градиентов должны вычисляться на основе параметров, задаваемых пользователем, не стоит рекомендовать методы Стюарта как более предпочтительные по сравнению с методами, использующими аналитические производные, и, как видно из табл. 3.4.2, метод Стюарта, как правило, не будет таким же эффективным, как алгоритм Пауэлла.

3.4.3. АЛГОРИТМЫ ПИРСОНА

Пирсон [31] предложил несколько методов вычисления матрицы η , использующих сопряженные направления поиска. Алгоритмы Пирсона можно получить, задавая различным образом векторы y и z в уравнении (3.4.4).

1. Алгоритм Пирсона № 2. Положим в уравнении (3.4.4) $y = z = \Delta x^{(k)}$, а $\omega = 1$. Тогда

$$\eta^{(k+1)} = \eta^{(k)} + \frac{(\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}) (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T (\Delta g^{(k)})}, \quad (3.4.9)$$

$$\eta^{(0)} = R^{(0)},$$

где $R^{(0)}$ — произвольная положительно определенная симметрическая матрица. Алгоритм Пирсона № 2 обычно приводит к «плохим» матрицам направлений. Фиг. 3.4.4 иллюстрирует задачу 35 из приложения А, для которой (начиная из $x^{(0)} = [2 \ 0,2]^T$ при $\eta^{(0)} = I$)

была получена следующая последовательность матриц направлений:

Шаг	Матрица направлений $\eta^{(k)}$	Определитель $\eta^{(k)}$
0	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	1,00000
1	$\begin{bmatrix} 0,36754 & 0,15788 \\ 0,99148 & 0,75250 \end{bmatrix}$	0,12004
2	$\begin{bmatrix} 0,36636 & 0,15383 \\ 0,88737 & 0,39527 \end{bmatrix}$	0,00831
3	$\begin{bmatrix} 0,35793 & 0,11330 \\ 0,85656 & 0,32020 \end{bmatrix}$	0,00043
4	$\begin{bmatrix} 0,35576 & 0,12809 \\ 0,85077 & 0,30633 \end{bmatrix}$	-0,00001

Таким образом, через четыре шага матрица направлений перестает быть положительно определенной и на последующих шагах остается «плохой», колеблясь между положительно определенной и не положительно определенной. Повторение начала алгоритма через каждые n шагов (т. е. приравнивание $\eta^{(k+1)}$ к $R^{(0)}$ после каждого n шагов) помогает избежать трудностей подобного рода.

2. Алгоритм Пирсона № 3. Положим в уравнении (3.4.4) $y = z = \eta^{(k)} \Delta g^{(k)}$, а $\omega = 1$.

Тогда

$$\eta^{(k+1)} = \eta^{(k)} + [(\Delta x^{(k)} - \eta^{(k)} (\Delta g^{(k)})] \frac{[\eta^{(k)} (\Delta g^{(k)})]^T}{(\Delta g^{(k)})^T \eta^{(k)} (\Delta g^{(k)})}, \quad (3.4.10)$$

$$\eta^{(0)} = R^{(0)}.$$

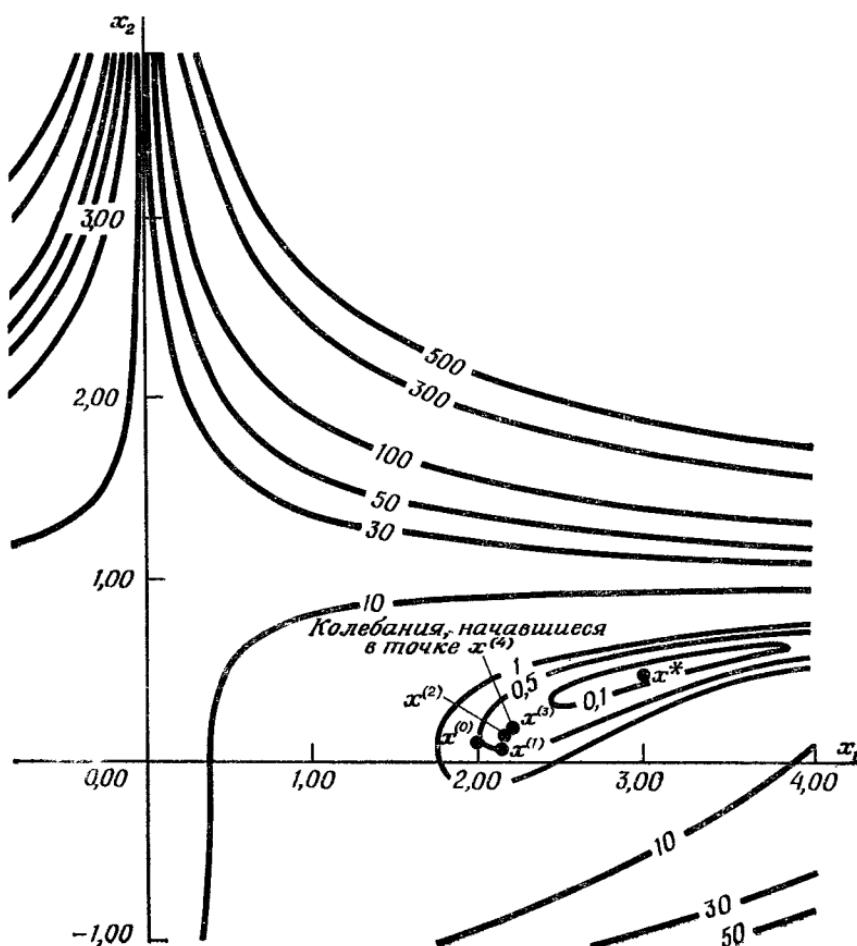
Траектория поиска алгоритма Пирсона № 3 при минимизации функции Розенброка по существу та же, что и изображенная на фиг. 3.4.3, за исключением того, что на каждом этапе делаются меньшие шаги. Пирсон исследовал также разновидность этого алгоритма с циклическим перезаданием матрицы η .

3. Проективный алгоритм Ньютона — Рафсона. Пирсон предложил так называемый проективный алгоритм Ньютона — Рафсона (PNR), который может быть получен из уравнения (3.4.4) при $\omega \rightarrow \infty$ и $z = \eta^{(k)} \Delta g^{(k)}$:

$$\eta^{(k+1)} = \eta^{(k)} - \frac{(\eta^{(k)} \Delta g^{(k)}) (\eta^{(k)} \Delta g^{(k)})^T}{(\Delta g^{(k)})^T \eta^{(k)} (\Delta g^{(k)})}, \quad (3.4.11)$$

$$\eta^{(0)} = R^{(0)}.$$

Уравнение (3.4.11) аналогично алгоритму Заутендайка [описан-



Ф и г. 3.4.4. Задача 35, приложение А.

ному в разд. 3.3.4], если проектирующую матрицу в уравнении (3.3.12) заменить матрицей направлений $\eta^{(k)}$. Величина $\eta^{(k)} \Delta g^{(k)}$ является проекцией $\Delta g^{(k)}$, ортогональной $G^{(k)}$, и на каждом n шагах $R^{(k)}$ является аппроксимацией $H^{-1}(x^{(k)})$, так что в сущности осуществляется (приближенно) поиск Ньютона. Когда k кратно n , числу независимых переменных, матрица $\eta^{(k)}$ заменяется на $R^{(k)}$. Тогда

$$R^{(k+1)} = R^{(k)} + \frac{(\Delta x^{(k)} - R^{(k)} \Delta g^{(k)})^T (\eta^{(k)} \Delta g^{(k)})^T}{(\Delta g^{(k)})^T \eta^{(k)} (\Delta g^{(k)})}. \quad (3.4.12)$$

Некоторые результаты, относящиеся к алгоритмам Пирсона, включены в материалы по оцениванию алгоритмов, приведенные в гл. 5.

3.4.4. ДРУГИЕ МЕТОДЫ УЛУЧШЕНИЯ НАПРАВЛЕНИЙ

Гринштадт [32] вывел общее соотношение для $\Delta\eta$ путем минимизации с помощью множителей Лагранжа нормы $\Delta\eta$, определенной следующим образом:

$$N(\Delta\eta) = \text{Tr} [\mathbf{W}\Delta\eta^{(k)} \mathbf{W} (\Delta\eta^{(k)})^T]$$

(где \mathbf{W} — положительно определенная симметрическая матрица) при условиях

$$1) \Delta\eta^{(k)} \text{ всегда симметрическая: } \Delta\eta^{(k)} = (\Delta\eta^{(k)})^T;$$

$$2) \eta^{(k+1)} \Delta g^{(k)} = \Delta x^{(k)}, \text{ или имеет место его эквивалент — уравнение (3.4.2в).}$$

Полученное соотношение имеет вид

$$\begin{aligned} \Delta\eta^{(k)} = & \frac{1}{(\Delta g^{(k)})^T \mathbf{W}^{-1} \Delta g^{(k)}} \left\{ \Delta x^{(k)} (\Delta g^{(k)})^T \mathbf{W}^{-1} + \right. \\ & + \mathbf{W}^{-1} \Delta g^{(k)} (\Delta x^{(k)})^T - \eta^{(k)} \Delta g^{(k)} (\Delta g^{(k)})^T \mathbf{W}^{-1} - \\ & - \mathbf{W}^{-1} \Delta g^{(k)} (\Delta g^{(k)})^T \eta^{(k)} - \frac{1}{(\Delta g^{(k)})^T \mathbf{W}^{-1} \Delta g^{(k)}} [(\Delta g^{(k)})^T \Delta x^{(k)} - \\ & \left. - (\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}] \mathbf{W}^{-1} \Delta g^{(k)} (\Delta g^{(k)})^T \mathbf{W}^{-1} \right\}. \quad (3.4.13) \end{aligned}$$

Гринштадт рассмотрел два варианта: $\mathbf{W}^{-1} = \eta^{(k)}$ и $\mathbf{W}^{-1} = \mathbf{I}$; однако применение уравнения (3.4.13) в тестовых задачах в случае $\mathbf{W}^{-1} = \mathbf{I}$ дало плохие результаты, тогда как $\mathbf{W}^{-1} = \eta^{(k)}$ привело к результатам, вполне сравнимым с результатами, полученными с помощью алгоритма Дэвидона — Флетчера — Паузелла.

В работе Голдфарба [33] указывается, что если положить $\mathbf{W}^{-1} = \eta^{(k)}$, то получается следующее уравнение:

$$\begin{aligned} \Delta\eta_1^{(k)} = & \frac{1}{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}} \left\{ \Delta x^{(k)} (\Delta g^{(k)})^T \eta^{(k)} + \eta^{(k)} \Delta g^{(k)} (\Delta x^{(k)})^T - \right. \\ & \left. - \left[1 + \frac{(\Delta g^{(k)})^T \Delta x^{(k)}}{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}} \right] \eta^{(k)} \Delta g^{(k)} (\Delta g^{(k)})^T \eta^{(k)} \right\}. \quad (3.4.14) \end{aligned}$$

Если же принять $\mathbf{W}^{-1} = \eta^{(k+1)}$, то

$$\begin{aligned} \Delta\eta_{11}^{(k)} = & \frac{1}{(\Delta g^{(k)})^T \Delta x^{(k)}} \left\{ - \Delta x^{(k)} (\Delta g^{(k)})^T \eta^{(k)} - \eta^{(k)} \Delta g^{(k)} (\Delta x^{(k)})^T + \right. \\ & \left. + \left[1 + \frac{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}}{(\Delta g^{(k)})^T \Delta x^{(k)}} \right] \Delta x^{(k)} (\Delta x^{(k)})^T \right\}. \quad (3.4.15) \end{aligned}$$

Наконец, если взять $\mathbf{W}^{-1} = \eta^{(k+1)} - \eta^{(k)} = \Delta\eta^{(k)}$, то получается уравнение (3.4.5) (даже если \mathbf{W}^{-1} и не имеет обратной матрицы, поскольку $\Delta\eta^{(k)}$ в уравнении (3.4.5) имеет ранг 1). Аналогично к урав-

нению (3.4.5) приводит подстановка

$$\mathbf{W}^{-1} = \boldsymbol{\eta}^{(k)} - \frac{\boldsymbol{\eta}^{(k)} \Delta g^{(k)} (\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)}}{(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}},$$

или

$$\mathbf{W}^{-1} = \boldsymbol{\eta}^{(k)} - \frac{\Delta x^{(k)} (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}}.$$

Уравнение (3.4.5a) можно получить, положив

$$\begin{aligned} \mathbf{W}^{-1} &= I[(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}]^{1/2} \boldsymbol{\eta}^{(k+1)} - [(\Delta g^{(k)})^T \Delta x^{(k)}]^{1/2} \boldsymbol{\eta}^{(k)} = \\ &= [(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k+1)} \Delta g^{(k)}]^{-1/2} \boldsymbol{\eta}^{(k+1)} - [(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}]^{-1/2} \boldsymbol{\eta}^{(k)} = \\ &= \boldsymbol{\eta}^{(k+1)} - \left[\frac{(\Delta g^{(k)})^T \Delta x^{(k)}}{(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}} \right]^{1/2} \frac{\boldsymbol{\eta}^{(k)} \Delta g^{(k)} (\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)}}{(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}} = \\ &= \boldsymbol{\eta}^{(k)} - \left[\frac{(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}}{(\Delta g^{(k)})^T \Delta x^{(k)}} \right]^{1/2} \frac{\Delta x^{(k)} (\Delta x^{(k)})^T}{(\Delta g^{(k)})^T \Delta x^{(k)}}. \end{aligned}$$

Выражения (3.4.14) и (3.4.15) дают поправки к $\boldsymbol{\eta}^{(k)}$, обеспечивающие квадратичное окончание процесса для случая строго выпуклой квадратичной целевой функции, так же, как это имеет место и при использовании формул (3.4.5) и (3.4.5a). Тем не менее только формулы (3.4.5a) и (3.4.15) сохраняют положительную определенность $\boldsymbol{\eta}$. Голдфарб показал, что каждое из уравнений (3.4.5), (3.4.5a), (3.4.14) и (3.4.15) может быть записано в виде линейной комбинации остальных, и в частности

$$\Delta \boldsymbol{\eta}_1 = \gamma \Delta \boldsymbol{\eta}_{(3.4.5a)} + (1 - \gamma) \Delta \boldsymbol{\eta}_{(3.4.5)},$$

$$\Delta \boldsymbol{\eta}_{11} = \gamma^{-1} \Delta \boldsymbol{\eta}_{(3.4.5a)} + (1 - \gamma^{-1}) \Delta \boldsymbol{\eta}_{(3.4.5)},$$

где

$$\gamma = \frac{(\Delta g^{(k)})^T \Delta x^{(k)}}{(\Delta g^{(k)})^T \boldsymbol{\eta}^{(k)} \Delta g^{(k)}}.$$

Следовательно, в общем случае можно записать

$$\Delta \boldsymbol{\eta}^{(k)} = \alpha \Delta \boldsymbol{\eta}_1^{(k)} + (1 - \alpha) \Delta \boldsymbol{\eta}_{11}^{(k)}. \quad (3.4.16)$$

Выбирая различные α , можно получить любое из полученных выше уравнений для $\Delta \boldsymbol{\eta}^{(k)}$.

3.4.5. МЕТОД ФЛЕТЧЕРА

Во всех описанных выше процедурах минимизации после вычисления направления поиска целевая функция минимизируется в этом направлении при помощи одномерного поиска. Поскольку наибольшее количество вычислений приходится на операции

одномерного поиска, естественно возникает вопрос, будут ли достаточны для решения задачи один или несколько шагов минимизации. Флетчер [34] предложил алгоритм, в котором отброшено условие квадратичного окончания процесса, т. е. окончания процесса за n шагов в случае квадратичной целевой функции, но в то же время сохранено свойство, состоящее в том, что для квадратичных функций матрица направлений $\eta \rightarrow H^{-1}(x)$ в том смысле, что собственные значения η стремятся к собственным значениям H^{-1} . Заметим, что соотношение для $\Delta\eta^{(k)}$, которое входит в уравнение (3.4.3), основано именно на этом свойстве при условии выполнения следующего равенства:

$$\eta^{(k+1)} \Delta g^{(k)} = \Delta x^{(k)}, \text{ т. е. } \Delta g^{(k)} = (\eta^{(k+1)})^{-1} \Delta x^{(k)}.$$

Улучшающее соотношение Флетчера основано на рекуррентном соотношении для обратных матриц

$$\begin{aligned} (\eta^{(k+1)})^{-1} &= \left[I - \frac{\Delta g^{(k)} (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}} \right] (\eta^{(k)})^{-1} \left[I - \frac{\Delta x^{(k)} (\Delta g^{(k)})^T}{(\Delta g^{(k)})^T \Delta g^{(k)}} \right] + \\ &\quad + \frac{\Delta g^{(k)} (\Delta g^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}}, \end{aligned} \quad (3.4.17)$$

соотношении, которое приводит к равенству $(\eta^{(k+1)})^{-1} \Delta x^{(k)} = \Delta g^{(k)}$. Поменяв в (3.4.17) местами $\Delta x^{(k)}$ с $\Delta g^{(k)}$ и $\eta^{(k+1)}$ с $(\eta^{(k+1)})^{-1}$, приходим к рекуррентной формуле

$$\eta^{(k+1)} = \left[I - \frac{\Delta x^{(k)} (\Delta g^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}} \right] \eta^{(k)} \left[I - \frac{\Delta g^{(k)} (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}} \right] + \frac{x^{(k)} (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}}. \quad (3.4.18)$$

Описываемый алгоритм использует соотношение (3.4.5а), если

$$(\Delta g^{(k)})^T H^{-1}(x^{(k)}) \Delta g^{(k)} < (\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)},$$

и (3.4.18), если

$$(\Delta g^{(k)})^T H^{-1}(x^{(k)}) \Delta g^{(k)} \geq (\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}.$$

Можно также использовать линейные комбинации уравнений (3.4.18) и (3.4.5а). Из рассмотрения тестовых задач, описанных в гл. 5, видно, что в большинстве случаев направления, полученные на первых этапах оптимизации, основаны на уравнении (3.4.5а), а уравнение (3.4.18) применяется лишь на последних этапах. Поскольку матрица η , вычисленная с помощью уравнения (3.4.5а), стремится к иулю с увеличением числа шагов, а η , вычисленная с помощью соотношения (3.4.18), стремится к бесконечности, имеет смысл применять оба типа уравнений.

Тем не менее эффективной¹⁾ процедуру Флетчера делает скорее не метод вычисления η , а кубическая интерполяция при отыска-

¹⁾ Поиск ДСК-Пауэлла с масштабированием (см. приложение Б) оказывается не менее эффективным.

ния минимума в данном направлении и ограниченная длина шага. При этом скаляр λ выбирается с помощью уравнения, аналогичного (3.4.8):

$$\lambda' = \frac{2[f(x^{(k)}) - f_{\text{ниж}}]}{\nabla^T f(x^{(k)}) s^{(k)}},$$

где $f_{\text{ниж}}$ — нижняя оценка значений $f(x)$. (Если $f(x)$ оказывается ниже, чем $f_{\text{ниж}}$, то имеет место окончание процесса.) Для достижения минимума в направлении поиска используется кубическая интерполяция на интервале между $x^{(k)}$ и $x^{(k)} + \lambda' s^{(k)}$. Поскольку для проведения кубической интерполяции не требуется использование всего интервала, содержащего минимум, и результат интерполяции может оказаться неудовлетворительным, если $x^{(k)}$ находится на вогнутой части аппроксимирующего полинома, Флетчер ограничил значение λ' следующим образом: либо оно должно быть меньше, чем значение λ' , получаемое с помощью кубической интерполяции, либо

$$\lambda^{(s+1)} = 0,1\lambda^{(s)},$$

где верхний индекс s обозначает номер в последовательности шагов при одномерном поиске.

После каждого шага проводится тест

$$-\frac{f(x^{(k)}) - f(x^{(k)} + \lambda^{(s)} s^{(k)})}{\lambda^{(s)} \nabla^T f(x^{(k)}) s^{(k)}} \geq 10^{-4},$$

и если он удовлетворен, то этап считается завершенным. В противном случае продолжается кубическая интерполяция. Окончательно процесс завершается, когда $\Delta x_i \leq \epsilon = 10^{-5}$. Во избежание большого влияния ошибок округления или ошибок программ вычисления производных программа останавливается, когда $f(x^{(k+1)}) > f(x^{(k)})$ и $(\Delta x^{(k)})^T \Delta g^{(k+1)} < 0$ и (или) $(\Delta x^{(k)})^T \Delta g^{(k)} \geq 0$.

Используя ряд тестовых задач, Флетчер сравнил приведенный выше алгоритм с алгоритмом Дэвидона — Флетчера — Паулла и показал, что он столь же эффективен, как и последний. При этом оказалось, что число итераций в каждом направлении поиска уменьшилось, однако для компенсации пришлось увеличить число направлений поиска. В гл. 5 приводятся некоторые результаты применения алгоритма Флетчера к другим тестовым задачам.

3.4.6. АППРОКСИМАЦИЯ МАТРИЦЫ ГЕССЕ

Вместо аппроксимации $H^{-1}(x^{(k+1)})$, как это делается в уравнении (3.4.3), можно аппроксимировать матрицу $H(x^{(k+1)})$, а затем построить обратную к ней. Таким образом,

$$H(x^{(k+1)}) \approx \Gamma^{(k+1)} = \Gamma^{(k)} + \Delta\Gamma^{(k)}, \quad (3.4.19)$$

где $\Gamma^{(k)}$ — оценка $\mathbf{H}(\mathbf{x}^{(k)})$, а матрица $\Delta\Gamma^{(k)}$ — симметрическая матрица ранга 1, такая, что $\Gamma^{(k+1)}$ удовлетворяет уравнению $\Gamma^{(k+1)}\Delta\mathbf{x}^{(k)} = \Delta\mathbf{g}^{(k)}$. При этом

$$\Delta\Gamma^{(k)} = \frac{[(\Delta\mathbf{g}^{(k)}) - \Gamma^{(k)}(\Delta\mathbf{x}^{(k)})][(\Delta\mathbf{g}^{(k)}) - \Gamma^{(k)}(\Delta\mathbf{x}^{(k)})]^T}{[(\Delta\mathbf{g}^{(k)}) - \Gamma^{(k)}(\Delta\mathbf{x}^{(k)})]^T(\Delta\mathbf{x}^{(k)})}. \quad (3.4.20)$$

Условия сходимости здесь те же, что и описанные в подразд. 3.4.1. Поскольку $(\Gamma^{(k+1)})^{-1}$ может не быть положительно определенной, следует использовать ограничительные предположения, сделанные в разд. 3.2, для обеспечения положительной определенности. В качестве исходной $\Gamma^{(0)}$ можно выбрать $(\eta^{(0)})^{-1}$.

Соотношение (3.4.20) аналогично (3.4.5), если иметь в виду, что в случае квадратичной аппроксимации $\Delta\mathbf{g}^{(k)} = \mathbf{H}\Delta\mathbf{x}^{(k)}$ (здесь) и $\Delta\mathbf{x}^{(k)} = \mathbf{H}^{-1}\Delta\mathbf{g}^{(k)}$ (в подразд. 3.4.1). В принципе возможно записать выражения для $\Delta\Gamma^{(k)}$, эквивалентные приведенным в подразд. 3.4.2—3.4.4, но сомнительно, будет ли при этом уменьшаться объем вычислений при решении задачи оптимизации, поскольку обращение матрицы должно будет производиться на каждом этапе.

Рассмотрим далее алгоритм Голдштейна и Прайса [35], оказавшийся достаточно эффективным при решении ряда задач. Хотя этот алгоритм и не попадает строго в категорию методов переменной метрики, тем не менее здесь на каждом этапе проводится аппроксимация $\mathbf{H}(\mathbf{x})$ при помощи разностной схемы, основанной на полуфакториальном построении, а затем осуществляется обращение матрицы. При этом для оценки $\mathbf{H}(\mathbf{x}^{(k)})$ требуется лишь информация о $f(\mathbf{x}^{(k)})$ и $\nabla f(\mathbf{x}^{(k)})$. Голдштейн и Прайс утверждают, что данный алгоритм минимизирует $f(\mathbf{x})$, если она представляет собой выпуклую целевую функцию, при условии выполнения некоторых сравнительно простых ограничений. На k -м этапе алгоритм выглядит следующим образом (величины $0 < \delta < \frac{1}{2}$ и $r > 0$ задаются заранее):

Шаг 1. Вычисляется в качестве аппроксимации $\mathbf{H}(\mathbf{x}^{(k)})$ матрица $\tilde{\mathbf{H}}(\mathbf{x}^{(k)})$ размерности $n \times n$, j -й столбец которой определяется по формуле

$$\nabla f(\mathbf{x}^{(k)} + \theta^{(k)}\mathbf{I}_j) - \nabla f(\mathbf{x}^{(k)}),$$

где

$$\theta^{(k)} = r \|\varphi(\mathbf{x}^{(k-1)})\| \quad \text{для } k > 0,$$

$$\theta^{(0)} = r,$$

$\mathbf{I}_j = j$ -й столбец единичной матрицы \mathbf{I} размерности $n \times n$.

$\varphi(\mathbf{x}^{(k)})$ — вектор-столбец, определяемый по формуле

$$\varphi(\mathbf{x}^{(k)}) = \begin{cases} -\nabla f(\mathbf{x}^{(k)}), & \text{если } k=0 \text{ или } \tilde{\mathbf{H}}(\mathbf{x}^{(k)}) \\ & \text{сингулярна, или } [\nabla^T f(\mathbf{x}^{(k)}) \tilde{\mathbf{H}}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})] \leq 0, \\ & \text{так что } \tilde{\mathbf{H}}^{-1} \text{ не является положительно определенной;} \\ -\tilde{\mathbf{H}}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)}), & \text{в противном случае.} \end{cases}$$

Заметим, что $\tilde{\mathbf{H}}(\mathbf{x}^{(k)})$ не обязательно симметрическая матрица, и если

$$\nabla^T f(\mathbf{x}^{(k)}) [\mathbf{H}^{-1}(\mathbf{x}^{(k)}) \nabla f(\mathbf{x}^{(k)})] \leq 0,$$

то предлагаемое направление поиска $\varphi(\mathbf{x}^{(k)})$ и направление градиента $\nabla f(\mathbf{x}^{(k)})$ отличаются более чем на 90° . Поскольку поиск проводится в «минус»-направлении, то отрицательный знак при квадратичной форме приводит к компоненте в направлении положительного градиента.

Шаг 2. Вычисляется

$$F(\mathbf{x}^{(k)}, \lambda) = \frac{f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k)} + \lambda \varphi(\mathbf{x}^{(k)}))}{\lambda [\nabla^T f(\mathbf{x}^{(k)}) \varphi(\mathbf{x}^{(k)})]}.$$

Выбирается $\lambda^{(k)}$ так, чтобы $\delta \leq F(\mathbf{x}^{(k)}, 1)$ или $\delta \leq F(\mathbf{x}^{(k)}, \lambda^{(k)}) \leq 1 - \delta$, $\lambda^{(k)} \neq 1$. Эти критерии нужны для того, чтобы не допускать шагов поиска, которые далеко выходят за область линейного изменения целевой функции в окрестности $\mathbf{x}^{(k)}$, предполагавшуюся при аппроксимации $\mathbf{H}(\mathbf{x})$.

Шаг 3. Берется $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \varphi(\mathbf{x}^{(k)})$.

Шаг 4. Процесс заканчивается, когда $\|\varphi(\mathbf{x}^{(k)})\| < \epsilon$.

Таким образом, если матрица $\tilde{\mathbf{H}}(\mathbf{x}^{(k)})$ сингулярна, или не приводит к «направлению вниз», этот метод сводится к методу наискорейшего спуска. Параметр r следует выбирать так, чтобы матрица $\tilde{\mathbf{H}}(\mathbf{x}^{(k)})$ аппроксимировала $\mathbf{H}(\mathbf{x}^{(k)})$ как можно ближе. Величина δ выбирается так, чтобы значения $f(\mathbf{x}^{(k)})$, $k = 1, 2, \dots$, представляли собой монотонно убывающую последовательность; чем ближе значение δ к $1/2$, тем в большей степени $f(\mathbf{x}^{(k)} + \lambda \varphi(\mathbf{x}^{(k)}))$ приближается к своему минимуму по λ . Голдштейн и Прайс отметили, что этот алгоритм по своей эффективности в отношении необходимого объема функциональных и градиентных вычислений при решении задачи Розенброка эквивалентен алгоритму Дэвидона — Флетчера — Паузелла. Для задач с плохо выбранными масштабами переменных он оказывается менее удовлетворительным, поскольку при этом фактически осуществляется движение по методу

наискорейшего спуска. Наиболее интересной чертой данного алгоритма является сравнительно малое число шагов одномерного поиска, приходящихся на каждое новое направление движения (этап): около одного-двух на этап. Некоторые результаты применения этого алгоритма для решения тестовых задач приведены в гл. 5.

3.5. КРАТКИЙ ОБЗОР АЛГОРИТМОВ ПРОГРАММИРОВАНИЯ БЕЗ ОГРАНИЧЕНИЙ

В этой главе были рассмотрены некоторые методы оптимизации без ограничений. В табл. 3.5.1 приведены в виде сводки для основных алгоритмов рекуррентные соотношения, используемые для вычисления $\eta(x^{(k)}) \equiv \eta^{(k)}$ или $s^{(k)}$, входящее в соотношение

$$\Delta x^{(k)} = x^{(k+1)} - x^{(k)} = \lambda^{*(k)} s^{(k)} = -\lambda^{*(k)} \eta(x^{(k)}) \nabla f(x^{(k)}).$$

Сравнение относительной эффективности алгоритмов мы отложили до гл. 5., так чтобы методы поиска, описанные в гл. 4, также могли быть включены в это сравнение.

ЗАДАЧИ ¹⁾

3.1. Ответьте на следующие вопросы относительно метода наискорейшего спуска:

а) Если целевая функция (без ограничений) плохо масштабирована, каким будет начальное развитие поиска: медленным или быстрым? Каким будет движение вблизи экстремума: быстрым или медленным?

б) Если выбрано направление градиентного поиска в точке $x^{(k)}$, как нужно выбрать размер шага, чтобы достичь следующей точки $x^{(k+1)}$?

3.2. Докажите, является ли метод наискорейшего спуска методом, использующим сопряженные направления при минимизации квадратичной функции с положительно определенной матрицей Гессе.

3.3. Что является направлением наискорейшего спуска в точке $x = [1 \ 1]^T$ для целевой функции $f(x) = x_1^2 + 2x_2^2$? Является ли эта функция плохо масштабированной?

3.4. Определите вектор, представляющий собой направление наискорейшего подъема в точке $x = [1 \ 1]^T$ для целевой функции в задачах 2.1а, 2.6а, 2.13, 2.32.

3.5. Проведите два шага оптимизации методом наискорейшего спуска, начиная из $x = [1 \ 1]^T$ для целевой функции $f(x) = x_1^2 + 2x_2^2$.

¹⁾ Дополнительные задачи, которые можно использовать при изучении этой главы, можно найти в списке задач, помещенном в конце гл. 4.

Таблица 3.5.1.

Методы нелинейного программирования без ограничений, использующие производные

Метод	Раздел	Рекуррентное соотношение
Наискорейшего спуска	3.1.1	$\eta^{(k)} = I$
Ньютона	3.2	$\eta^{(k)} = -[\nabla^2 f(x^{(k)})]^{-1} \equiv H^{-1}(x^{(k)})$
Гринштадта	3.2	$\eta^{(k)} = C^{-1}(x^{(k)}) \Pi^{-1} C^{-1}(x^{(k)})$
Маркуардта	3.2	$\eta^{(k)} = C^{-1}(x^{(k)}) (\Pi + \beta I)^{-1} C^{-1}(x^{(k)})$
Бройдена, ранг 1	3.4.1	$\eta^{(k+1)} = \eta^{(k)} + \left\{ \frac{[\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}] [\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}]^T}{[\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}]^T \Delta g^{(k)}} \right\}$
Дэвидона — Флетчера — Пауэлла	3.4.2	$\eta^{(k+1)} = \eta^{(k)} + \left\{ \frac{\Delta x^{(k)} (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}} - \frac{\eta^{(k)} \Delta g^{(k)} [\eta^{(k)} \Delta g^{(k)}]^T}{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}} \right\}$
Пирсона № 2	3.4.3	$\eta^{(k+1)} = \eta^{(k)} + \left\{ \frac{[\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}] (\Delta x^{(k)})^T}{(\Delta x^{(k)})^T \Delta g^{(k)}} \right\}$
Пирсона № 3	3.4.3	$\eta^{(k+1)} = \eta^{(k)} + \left\{ \frac{[\Delta x^{(k)} - \eta^{(k)} \Delta g^{(k)}] [\eta^{(k)} \Delta g^{(k)}]^T}{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}} \right\}$
Проективный Ньютона	3.4.3	
Проекции Заутендейка (если $P = \eta$)	3.3.4	$\eta^{(k+1)} = \eta^{(k)} - \left\{ \frac{[\eta^{(k)} \Delta g^{(k)}] [\eta^{(k)} \Delta g^{(k)}]^T}{(\Delta g^{(k)})^T \eta^{(k)} \Delta g^{(k)}} \right\}$
Гринштадта и Голдфарба	3.4.4	$\eta^{(k+1)} = \eta^{(k)} + \text{уравнение (3.4.13)}$
Флетчера	3.4.5	$\eta^{(k+1)} = \text{уравнение (3.4.18)}$
Флетчера — Ривса	3.3.2	$s^{(k+1)} = -\nabla f(x^{(k)}) + \left\{ \frac{s^{(k)} \nabla^T f(x^{(k+1)}) \nabla f(x^{(k+1)})}{\nabla^T f(x^{(k)}) \nabla f(x^{(k)})} \right\}$
Аппроксимации $H(x)$	3.4.6	$\tilde{H}^{(k+1)} = \tilde{H}^{(k)} + \left\{ \frac{[\Delta g^{(k)} - \tilde{H}^{(k)} \Delta x^{(k)}] [\Delta g^{(k)} - \tilde{H}^{(k)} \Delta x^{(k)}]^T}{[\Delta g^{(k)} - \tilde{H}^{(k)} \Delta x^{(k)}]^T \Delta x^{(k)}} \right\}$
Голдштейна и Прайса	3.4.6	Применяется разностная схема для аппроксимации элементов $H(x^{(k)})$ или используется $s^{(k)} = -\nabla f(x^{(k)})$

Сначала используйте фиксированную величину шага. Затем минимизируйте $f(\mathbf{x})$ на каждом шаге либо численно, либо аналитически.

3.6. Рассмотрите следующие целевые функции:

а) $f(\mathbf{x}) = 1 + x_1 + x_2 + \frac{4}{x_1} + \frac{9}{x_2}$;

б) $f(\mathbf{x}) = (x_1 + 5)^2 + (x_2 + 8)^2 + (x_3 + 7)^2 + 2x_1^2x_2^2 + 4x_1^2x_3^2$.

Будет ли метод Ньютона сходиться для этих функций?

3.7. Рассмотрите минимизацию целевой функции

$$f(\mathbf{x}) = x_1^3 + x_1x_2 - x_2^2x_1^2$$

методом Ньютона, начиная из точки $\mathbf{x}^{(0)} = [1 \ 1]^T$. Машинная программа, тщательно составленная для метода Ньютона, оказалась неудачной. Объясните возможную причину (причины) неудачи.

3.8. Что является начальным направлением поиска по методу Ньютона для задачи 3.3? Какова длина шага? Сколько понадобится шагов (направлений поиска) для решения задачи 3.17, для решения задачи 3.37?

3.9. Объясните топологию следующих целевых функций, используя фиг. 3.2.2 и табл. 3.2.1:

а) $3x_1 + 2x_2^2$;

б) $3 + 2x_1 + 3x_2 + 2x_1^2 + 2x_1x_2 + 6x_2^2$,

в) $3 + 2x_1 - 3x_2 + 2x_1^2 + 2x_1x_2 + 6x_2^2$;

г) $3x_1^2 - 4x_1x_2 + x_2^2$.

3.10. Аппроксимируйте целевую функцию из задачи 3.34д с помощью квадратичной функции. Интерпретируйте топологию целевой функции, используя аппроксимирующую функцию.

3.11. Является ли матрица Гессе для $f(\mathbf{x}) = 5x_1^2 + 3x_2^2 + x_3^2 - 2x_1x_2$ всегда положительно определенной? Выясните то же для $2x_1^2 - x_2^2 - x_1x_2$.

3.12. Как можно аппроксимировать матрицу Гессе для функции $f(\mathbf{x}) = 2x_1^2 - 2x_2^2 - x_1x_2$ с помощью положительно определенной матрицы (а) методом Гринштадта, (б) методом Маркуарда?

3.13. Что можно сказать о сходимости метода Ньютона, исходя из рассмотрения матрицы Гессе целевой функции в задачах 3.34б, 3.34д, 3.36?

3.14. Какие типы поверхностей представлены следующими выражениями (имеется в виду классификация, представленная на фиг. 3.2.3 и в табл. 3.2.1):

а) $x_1^2 - x_1x_2 + x_2^2$;

б) $x_1^2 + 2x_1x_2 + x_2^2$;

в) $2x_1^2 + 2x_2^2 + 8x_3^2 - 4x_1x_2 + 12x_1x_3 + 8x_2x_3$?

3.15. Задана функция $f(\mathbf{x}) = x_1^2 + x_2^2 + 2x_3^2 - x_1x_2$; образуйте систему сопряженных направлений. Проведите два этапа минимизации в сопряженных направлениях, минимизируя $f(\mathbf{x})$ в каждом направлении (т. е. на каждом шаге). Изобразите траекторию поиска и несколько линий уровня целевой функции.

3.16. При каких значениях \mathbf{x} направления

$$\mathbf{s}^{(1)} = \begin{bmatrix} -\frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{3}} \end{bmatrix}, \quad \mathbf{s}^{(2)} = \begin{bmatrix} -\frac{1}{\sqrt{3}} \\ \frac{2}{\sqrt{3}} \\ \frac{1}{\sqrt{3}} \\ 0 \end{bmatrix}.$$

являются сопряженными для функции $f(\mathbf{x}) = x_1^2 + x_1x_2 + 16x_2^2 + x_3^2 - x_1x_2x_3$?

3.17. Являются ли направления $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ и $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ линейно независимыми? Ортогональными? Сопряженными?

3.18. Возможно ли, чтобы ортогональные направления были сопряженными направлениями? Объясните.

3.19. Покажите, что направления поиска $\mathbf{s}^{(1)} = [0,453 \ -0,892]^T$ и $\mathbf{s}^{(2)} = [0,608 \ -0,794]^T$, используемые при минимизации функции Розенброка $f(\mathbf{x}) = 100(x_2 - x_1)^2 + (1 - x_1)^2$ в точке $\mathbf{x} = [-0,702 \ 0,462]^T$, сопряжены.

3.20. Покажите, что выражение (3.3.14) является правильным весовым множителем для метода сопряженного градиента.

3.21. Дают ли одну и ту же последовательность направлений поиска $\mathbf{s}^{(1)}, \mathbf{s}^{(2)}$ и т. д. методы, использующие сопряженные направления и начинающиеся из одной и той же исходной точки $\mathbf{x}^{(0)}$ и с одним и тем же $\mathbf{s}^{(0)}$? Ответьте да или нет и приведите простой пример для объяснения ответа.

3.22. Сравните методы переменной метрики по трем показателям:

- значениям элементов матрицы направлений;
- направлениям поиска;
- векторам \mathbf{x}

для выбранных вами задач по нескольким этапам, начинающимся с одного и того же вектора \mathbf{x} .

3.23. Для каждого из методов переменной метрики, приведенных в разд. 3.4, укажите значения элементов матрицы направлений на первых трех этапах поиска для задач 3.34г, 3.37, 3.40а.

3.24. Вычислите матрицу направлений с помощью выбранного вами метода переменной метрики для одной из целевых функций, приведенных в задаче 3.39.

3.25. Данна начальная матрица направлений $\eta^{(0)} = I$; что представляет собой $\eta^{(1)}$ по алгоритму Дэвидона — Флетчера — Пауэлла, если $f(x) = x_1^2 + 2x_2^2$, а $x^{(0)} = [1 \ 1]^T$?

3.26. После начального поиска при минимизации функции Розенброка матрица направлений в точке $x^{(1)} = [1,441 \ 2,078]^T$ имела вид

$$\eta^{(1)} = \begin{bmatrix} 1,544 \cdot 10^{-1} & 3,467 \cdot 10^{-1} \\ -3,467 \cdot 10^{-1} & 8,578 \cdot 10^{-1} \end{bmatrix}.$$

Каково будет следующее направление поиска в выбранном вами методе?

3.27. После 13 этапов минимизации функции Розенброка машинная программа остановилась в точке $x = [1 \ 1]^T$. В этой точке градиент в сущности — нулевой вектор. Какова будет матрица направлений в выбранных вами методе и задаче?

3.28. Использует ли алгоритм Голдштейна — Прайса сопряженные направления?

3.29. Выведите рекуррентное уравнение для вычисления матрицы направлений с помощью (а) алгоритма Дэвидона — Флетчера — Ривса, б) алгоритмов Пирсона, в) алгоритма Брайдена, если нужно максимизировать, а не минимизировать целевую функцию.

3.30. Все алгоритмы гл. 3 выражены в виде задачи минимизации. Как простейшим способом использовать эти алгоритмы для максимизации, а не минимизации?

3.31. На седьмом этапе метода Дэвидона — Флетчера — Пауэлла для случая функции Розенброка были получены следующие значения:

$$f(x) = 0,19469; \quad x_1 = 1,4409; \quad x_2 = 2,0779;$$

$$\partial f(x)/\partial x_1 = -0,1455; \quad \partial f(x)/\partial x_2 = 0,3565; \quad \text{матрица направлений}$$

$$\begin{bmatrix} 0,1544 & -0,3467 \\ -0,3467 & 0,8577 \end{bmatrix}.$$

Каковы будут направления поиска на последующих двух этапах?

3.32. Целевая функция

$$f(x) = \left(1 + 8x_1 - 7x_2^2 + \frac{7}{3}x_1^3 - \frac{1}{4}x_1^4\right)(x_2^2 e^{-x_2})$$

является бимодальной и имеет седловую точку в $x = [2 \ 2]^T$, где $f(x) = 2$.

Определите, какой из локальных оптимумов является глобальным. Проведите один этап поиска, начиная из седловой точки. Для облег-

чения решения проведите линии уровней функции $f(\mathbf{x})$ на плоскости (x_1, x_2) .

3.33. Решите каждую из следующих систем уравнений путем минимизации суммы квадратов разностей $\Sigma (d_i - 0)^2$:

$$\begin{aligned} \text{а) } & x_1 + 3 \lg x_1 - x_2^2 = 0, \\ & 2x_1^2 - x_1 x_2 - 5x_1 + 1 = 0; \\ \text{б) } & x_1^2 + (x_2 - 1)^2 - 5 = 0, \\ & (x_1 - 1)^2 + x_2^2 - 1 = 0. \end{aligned}$$

в) Систему 24 нелинейных уравнений с 24 переменными, предложенную Пэком и Суоном [36].

3.34. Минимизируйте следующие функции двух независимых переменных:

$$\begin{aligned} \text{а) } & \frac{1}{2} \left\{ 1 - \cos 360 [(2x - 1)^2 + (2y - 1)^2]^{1/2} \right\} \left[1 - \frac{(y - 3x)^2}{8} \right]; \\ \text{б) } & y + \sin x; \\ \text{в) } & -1(y - x)^4 + (1 - x)^2; \\ \text{г) } & -x^2 + x - y^2 + y + 4; \\ \text{д) } & \exp[-(x - 1)^2] - \frac{(y^2 - 0,5)^2}{0,132}. \end{aligned}$$

3.35. Предприниматель может производить товар А с затратами в 20 центов за фунт и товар В с затратами в 10 центов за фунт. Служащие, занимающиеся вопросами сбыта, полагают, что фирма может продавать $1\ 000\ 000/x^2y$ фунтов товара А в день и $2\ 000\ 000/xy^2$ фунтов товара В в день, x — продажная цена А в центах за фунт, а y — продажная цена В в центах за фунт. Определите максимальную прибыль, если А и В продаются по одной и той же цене. Чему равны в этом случае x и y ? Определите максимальную прибыль, если А и В продаются по разным ценам. Чему равны при этом x и y ?

3.36. Ежегодные расходы, связанные с эксплуатацией газового компрессора на трансконтинентальном газопроводе, выражаются формулой

$$C = \frac{KQZ}{10^6 L} \left(\ln \frac{P_1}{P_2} + b \right) + K_1 D^2 \left[\frac{P_1}{2(s - P_1)} + \frac{P_1^2}{4(s + P_1)^2} \right],$$

где

C — эксплуатационные расходы, долл/год;

Q — количество накачиваемого газа, фут³/день;

L — расстояние между компрессорными станциями, мили;

P_1 — давление на выходе, фунт/кв. дюйм;

P_2 — давление при всасывании, фунт/кв. дюйм;
 D — диаметр трубопровода, дюйм;
 K, K_1, Z, s, b — константы.

Кроме того,

$$Q = K_2 \frac{D^{2,6} (P_1^2 - P_2^2)^{0,54}}{L^{0,54} Z^{0,54}}.$$

При $Z = 1, K = 1370, L = 20, b = 1,476, K_1 = 0,081, s = 100$ и $K_2 = 1,13$ определите P_1 и P_2 , минимизирующие C .

3.37. Максимизируйте следующую целевую функцию:

$$f(\mathbf{x}) = x_1^3 \exp [x_2 - x_1^2 - 10(x_1 - x_2)^2].$$

Сравните траектории оптимизации в пространстве \mathbf{x} при использовании следующих методов:

- а) наискорейшего спуска;
- б) модифицированного партан-метода;
- в) метода Ньютона;
- г) метода переменной метрики;
- д) Флетчера — Ривса;
- е) Голдштейна — Прайса.

3.38. Максимизируйте следующую целевую функцию:

$$\begin{aligned} f(\mathbf{x}) = & (0,35 + 0,40x_1 + 0,31x_2)^4 (0,85 - 0,60x_1 + \\ & + 0,85x_2)^4 \exp [2,00 - (0,35 + 0,40x_1 + 0,35x_2)^4 - \\ & - (0,85 - 0,60x_1 + 0,85x_2)^4]. \end{aligned}$$

Сравните траектории оптимизации в пространстве \mathbf{x} при использовании следующих методов:

- а) наискорейшего спуска;
- б) продолженного (модифицированного) партан-метода;
- в) метода Ньютона;
- г) метода переменной метрики;
- д) Флетчера — Ривса;
- е) Голдштейна — Прайса.

3.39. Начиная с точки $\mathbf{x}^{(k)} = [1 \quad -2 \quad 3]^T$, определите точку $\mathbf{x}^{(k+1)}$:
1) методом наискорейшего спуска; 2) модифицированным партан-методом; 3) методом Ньютона; 4) методом Дэвидона — Флетчера — Паузелла; 5) методом сопряженного градиента для следующих целевых функций:

- а) $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2$;
- б) $f(\mathbf{x}) = 2x_1^2 + 2x_1x_2 + 3x_2^2 + x_3$;
- в) $f(\mathbf{x}) = \exp(x_1^2 + x_2^2 - x_3 - x_1 + 4)$.

3.40. Минимизируйте следующие функции, начиная с вектора

$$\mathbf{x}^{(0)} = [2 \ -2,5 \ 2 \ -2,5]^T;$$

а) $f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 + x_4^2$;

б) $f(\mathbf{x}) = (x_1 - x_2)^2 + (x_3 - x_4)^2$;

в) $f(\mathbf{x}) = x_1^3 + x_2 + x_3^3 + x_4 + 16x_1x_2 + 8x_2x_3 + x_3^2x_4 + 2$.

3.41. Функция, описанная Уилингом [37], позволяет оценивать способность того или иного алгоритма преодолевать разрывы. Эта функция

$$f(\mathbf{x}) = -3|x_1| - |x_2|$$

имеет форму пирамиды в трех измерениях, а ее линии уровня на плоскости x_1, x_2 представляют собой ромбы с разрывами вдоль главных осей. Найти максимум $f(\mathbf{x})$, а также \mathbf{x}^* : а) методом наискорейшего спуска; б) модифицированным партан-методом; в) методом Ньютона, г) методом переменной метрики; д) методом сопряженного градиента. (Максимум имеет место в начале координат.) Уилинг начинал оптимизацию с точки $\mathbf{x}^{(0)} = [10 \ 10]^T$, где $f(\mathbf{x}) = -40$; попробуйте другие начальные векторы.

3.42. Если целевая функция имеет общий вид

$$f(\mathbf{x}) = \prod_{i=1}^n (x_i^i e^{-x_i}), \quad (a)$$

то можно показать, что максимум $f(\mathbf{x})$ имеет место в точке, соответствующей максимуму каждого сомножителя, т. е. в точке $x_i = i$. Используйте функцию (a) в качестве тестовой функции и вычислите выбранным вами методом максимальное значение следующей функции: $f(\mathbf{x}) = x_1 x_2^2 x_3^3 x_4^4 e^{-(x_1+x_2+x_3+x_4)}$. Сравните результат с известным максимумом. Используйте две начальные точки: $\mathbf{x}^{(0)} = \left[3 \ 4 \ \frac{1}{2} \ 1 \right]^T$ и $\mathbf{x}^{(0)} = [1 \ 1 \ 1 \ 1]^T$.

3.43. Следующая функция, согласно Бруксу [38], представляет собой длинный узкий хребет с максимумом в точке $\mathbf{x}^* = [1 \ 1]^T$:

$$f(\mathbf{x}) = x_1^2 \exp[1 - x_1^2 - 20,25(x_1 - x_2)^2].$$

Найдите максимум $f(\mathbf{x})$, начиная с векторов $\mathbf{x}^{(0)} = [0,1 \ 0,1]^T$ и $\mathbf{x}^{(0)} = [1,9 \ 0,1]^T$, с помощью выбранного вами метода. Изобразите эту функцию и траекторию оптимизации на плоскости (x_1, x_2) .

3.44. Известно, что при статистическом анализе данных аппроксимировать некоторый процесс экспоненциальными функциями довольно трудно вследствие взаимодействия параметров. Последнее приводит в пространстве параметров к гиперболическим хребтам. В качестве примера минимизируйте сумму квадратов

отклонений

$$\Phi = \sum_{i=1}^n (y_{\text{наблюданное}} - y_{\text{предсказанное}})_i^2,$$

где

$$y_{\text{предсказанное}} = \frac{k_1}{k_1 - k_2} (e^{-k_2 t} - e^{-k_1 t})$$

для следующих данных:

t	$y_{\text{наблюданное}}$
0,5	0,263
1,0	0,455
1,5	0,548

Изобразите графически поверхность суммы квадратов для полученных значений коэффициентов.

3.45. Повторите задачу 3.44 для модели

$$y = \frac{k_1 x_1}{1 + k_2 x_1 + k_3 x_2}$$

и данных

$y_{\text{наблюданное}}$	x_1	x_2
0,126	1	1
0,219	2	1
0,076	1	2
0,126	2	2
0,186	0,1	0

3.46. Аппроксимируйте минимальное значение интеграла

$$\int_0^1 \left[\left(\frac{dy}{dx} \right)^2 - 2yx^2 \right] dx$$

при следующих граничных условиях:

$$dy/dx = 0 \quad \text{при } x = 0 \quad \text{и} \quad y = 0 \quad \text{при } x = 1.$$

Указание. В качестве пробной функции возьмите $y(x) = a(1 - x_2)$, удовлетворяющую граничным условиям, и найдите значение a , минимизирующее интеграл. Улучшит ли оценку минимума?

мума интеграла более сложная пробная функция, удовлетворяющая граничным условиям?

3.47. В задаче, связанной с проблемой принятия решения, желательно минимизировать ожидаемый риск, определяемый следующим образом:

$$\mathbb{E}\{\text{риск}\} = (1 - P) c_1 [1 - F(b)] + P c_2 \theta \left(\frac{b}{2} + \frac{2\pi}{4} \right) F \left(\frac{b}{2} - \frac{\sqrt{2\pi}}{4} \right),$$

где $F(b) = \int_{-\infty}^b e^{-u^2/2\theta^2} du,$

$$c_1 = 1,25 \cdot 10^5, \quad c_2 = 15, \quad \theta = 2000, \quad P = 0,25.$$

Найдите минимальный ожидаемый риск и значение b .

3.48. Пекарня, производящая однофунтовые батоны хлеба, обычно выпускает 10 000 фунтов хлеба в день. Вследствие возникшего периода острой конкуренции рыночная цена этого продукта упала так низко, что предприятие работает с убытком. При нормальном производстве (10 000 фунтов в день, 300 дней в году) имеют место следующие расходы:

1) расходы, связанные с затратами труда и контролем, стоимость пара, расходы по продаже и на социальное обеспечение и т. д.: 600 долл/день;

2) цена сырья: 0,075 долл/фунт продукции;

3) ежегодные налоги, страхование и т. д., доходящие до 25% вложений: 300 000 долл.

Предположим, что расходы, указанные в п. 1 и 2, линейно зависят от количества продукции, тогда как расходы п. 3 остаются постоянными. На хлеб спрос постоянный, и эксперты в этой области считают, что ни один новый процесс по его изготовлению не является допустимым.

Сотрудники производственного отдела совместно с отделом реализации и учета выяснили, исходя из лучших доступных данных, что себестоимость в долларах на единицу продукции задается (в пределах производства от 0 до 20 000 фунт/день) уравнением

$$C = \frac{(300 000 \text{ долл/день}) (0,25/\text{год})}{(300 \text{ день/год}) (P \text{ фунт/год})} + \frac{400 \text{ долл/день}}{P_N \text{ фунт/день}} + 0,075/\text{фунт} + \\ + 10^{-7} P^{1,3},$$

где P — фактическая продукция в фунтах в день, а P_N — производительность при нормальных условиях, 10 000 фунт/день. Отдел реализации определил, что продажная цена (включая возвраты) в следующем месяце в среднем будет равна 18,1 цент/фунт. Основываясь на приведенных выше данных, какой объем произ-

водства вы посоветуете установить в качестве оптимального на следующий месяц?

3.49. Оправдана ли экономически тепловая изоляция большой цистерны для хранения нефти и если да, то какова оптимальная толщина этой изоляции? Расходы могут быть разделены на две категории: а) стоимость установки и б) текущие расходы. Стоимость установки (в долл/год) тепловой изоляции равна $c_1 c_2 A x$. Для поддержания нужной вязкости нефти в цистерне должна быть установлена также обогревающая спираль; стоимость ее установки (в долл/год) равна

$$c_3 c_4 = \frac{Q}{U(t_s t_0)},$$

где

$$Q = \frac{A(t_0 - t_a)}{\left(\frac{x}{k} + \frac{1}{h_a} + \frac{1}{h_0} \right)}.$$

Текущие расходы на цистерну (в долл/год) составляют $c_5 c_6 Q$. Найти минимальную величину расходов и требуемую толщину изоляции x при следующих данных: $A = 8000$; $t_0 = 120$; $t_a = 40$; $t_s = 250$; $k = 0,024$; $c_1 = 4$; $c_2 = 0,20$; $c_3 = 0,8$; $c_4 = 0,262$; $c_5 = 1,5 \cdot 10^{-6}$; $c_6 = 4000$; $h_0 = 15 + 60x + 10x^2$; $h_a = 4,0 - 10x$; $U = 17$.

Что означает отрицательное значение x ?

Обозначения

- c_1 — стоимость изоляционного материала, долл/(фут² поверхности) · (фут толщины);
- c_2 — амортизационный коэффициент (безразмерный);
- c_3 — стоимость нагревающей поверхности, долл/фут²;
- c_4 — амортизационный коэффициент (безразмерный);
- c_5 — стоимость пара, долл/БЕТ;
- c_6 — количество рабочих часов в год, ч;
- A — площадь изоляции, фут²;
- h_a — коэффициент теплоотдачи стенки цистерны воздуху, БЕТ/ч · фут² · °F;
- h_0 — коэффициент теплоотдачи нефти стенке цистерны, БЕТ/ч · фут² · °F;
- k — теплопроводность изоляции, БЕТ/ч · фут · °F;
- Q — тепловые потери, БЕТ/ч;
- U — коэффициент теплопередачи между спиралью и нефтепродуктом, БЕТ/ч · фут² · °F;
- t_a — температура окружающего воздуха, °F;
- t_0 — температура нефти, °F;
- t_s — температура спирали обогрева, °F;
- x — толщина изоляции, фут.

ЛИТЕРАТУРА

1. Goldstein A. A., *Numerical Math.*, **4**, 146 (1962).
2. Akaike H., *Ann. Inst. Statist. Math., Tokyo*, **11**, 1 (1959).
3. Elder H., Ph. D. Dissertation, Purdue Univ., Lafayette, Ind., 1966.
4. Box G. E. P., Wilson K. B., *J. Roy. Statist. Soc.*, **B13**, 1 (1951).
5. Langley J. W., *J. Am. Statist. Assoc.*, **62**, 819 (1967).
6. Rosenbrock H. H., *Computer J.*, **3**, 174 (1960).
7. Greenstadt J., *Math. Computation*, **21**, 360 (1967).
8. Marquardt D. W., *J. SIAM*, **11**, 431 (1963).
9. Levenberg K., *Quart. Appl. Math.*, **2**, 164 (1944).
10. Goldfeld S. M., Quandt R. E., Trotter H. F., *Econometrica*, **34**, 541 (1966).
11. Zwart P. B., Nonlinear Programming: A Quadratic Analysis of Ridge Paralysis, Washington Univ., Rep. COO-1493-21, St. Louis, Mo., Jan. 1969.
12. Hestenes M. R., The Conjugate Gradient Method for Solving Linear Systems, in Proc. of the Symp. on Applied Mathematics, Vol. VI, McGraw-Hill, N. Y., 1956, pp. 83—102.
13. Fletcher R., Reeves C. M., *Computer J.*, **7**, 149 (1964).
14. Hestenes M. R., Stiefel E. L., *J. Res. Natl. Bur. Std.*, **B49**, 409 (1952).
15. Beckman F. S., The Solution of Linear Equations by the Conjugate Gradient Method, in: Mathematical Methods for Digital Computers, Ralston A., Wilf H. S., eds., Vol. 1, Wiley, Inc., N. Y., 1960.
16. Shah B. V., Buehler R. J., Kempthorne O., *J. SIAM*, **12**, 74 (1964).
17. Forsythe G. E., Motzkin T. S., *Bull. Am. Math. Soc.*, **57**, 183 (1951).
18. Zoutendijk G., Methods of Feasible Directions, American Elsevier Publ. Co., N. Y., 1960.
19. McCormick G. P., Pearson J. D., Chap. 21 in: Optimization, Fletcher R., ed., Academic Press Inc., London, 1969.
20. Miele A., Cantrell J. W., Rice Univ. Aero-Astronautics Rept. 56, Houston, Tex., 1969.
21. Cragg E. E., Levy A. V., Rice Univ. Aero-Astronautics Rept. 58, Houston, Tex., 1969.
22. Broyden C. G., *Math. Computation*, **21**, 368 (1967).
23. Goldfarb D., Chap. 18 in: Optimization, Fletcher R., ed., Academic Press Inc., N. Y., 1969.
24. Davidon W. C., *Computer J.*, **10**, 406 (1968); Chap. 2 in: Optimization, Fletcher R., ed., Academic Press Inc., N. Y., 1969.
25. Powell M. J. D., Rank One Methods for Unconstrained Optimization, AERE Rept. TP 372, 1969.
26. Murtagh B. A., Sargent R. W. H., in: Optimization, Fletcher R., ed., Academic Press Inc., London, 1969.
27. Davidon W. C., USAEC Doc. ANL-5990 (rev.), Nov. 1959.
28. Fletcher R., Powell M. J. D., *Computer J.*, **6**, 163 (1963).
29. Bard Y., On a Numerical Instability of Davidon-like Methods, IBM N. Y. Sci. Center Rept. 320-2913, Aug. 1967.
30. Stewart G. W., *J. Assoc. Computer Machinery*, **14**, 72 (1967).
31. Pearson J. D., *Computer J.*, **13**, 171 (1969).
32. Greenstadt J., *Math. Computation*, **24**, 1 (1970).
33. Goldfarb D., *Math. Computation*, **24**, 23 (1970).
34. Fletcher R., *Computer J.*, **13**, 317 (1970).
35. Goldstein A. A., Price J. F., *Numerical Math.*, **10**, 184 (1967).
36. Pack D. C., Swan G. W., *J. Fluid Mech.*, **25**, 165 (1966).
37. Wheeling R. F., *Comm. Assoc. Computer Mach.*, **3**, 632 (1960).
38. Brooks S. H., *Operations Res.*, **7**, 430 (1959).

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

ОБЩИЕ ВОПРОСЫ

- Box M. J., A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems, *Computer J.*, **9**, 67 (1966).
- Dorn W. S., Nonlinear Programming: A. Survey, *Management Sci.*, **9**, 171 (1963).
- Hestenes M. R., *J. Opt. Theory and Appl.*, **4**, 303 (1969).
- Hurt J. J., A Review of Algorithms for Optimization, Univ. of Iowa Rep. 22, June 1970.
- Kowalik J., Osborne M. R., Methods for Unconstrained Optimization Problems, American Elsevier, N. Y., 1968.
- Meyers G. E., Properties of the Conjugate Gradient and Davidon Methods, *J. Opt. Theory Appl.*, **2** (1968).
- Powell M. J. D., A Survey of Numerical Methods for Unconstrained Optimization, *SIAM Rev.*, **12**, 79 (1970).
- Ribiére G., Sur la méthode de Davidon-Fletcher-Powell pour la minimisation des fonctions, *Management Sci.*, **16**, 572 (1970).
- Schechter R. S., Beveridge G. S. C., Optimization: Theory and Practice, McGraw-Hill, N. Y., 1970.
- Spang H. A., III, A Review of Minimization Techniques for Nonlinear Functions, *SIAM Rev.*, **5**, 343 (1962).
- Topkis D. M., Veinott A. F., On the Convergence of Some Feasible Direction Algorithms for Nonlinear Programming, *J. SIAM Control*, **5**, 268 (1967).
- Wilde D. J., Optimum Seeking Methods, Prentice-Hall, Englewood Cliffs, N. J., 1964.
- Wilde D. J., Beightler C. S., Foundations of Optimization, Prentice-Hall, Englewood Cliffs, N. J., 1967.
- Wolfe P., Recent Developments in Nonlinear Programming, *Advan. Computers*, **3**, 155—187 (1962).

ДРУГИЕ МЕТОДЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ БЕЗ ОГРАНИЧЕНИЙ, ИСПОЛЬЗУЮЩИЕ ПРОИЗВОДНЫЕ

- Booth A. D., An Application of the Method of Steepest Descents to the Solution of Systems of Nonlinear Simultaneous Equations, *Quart. J. Mech. Appl. Math.*, **11**, 460, 191 (1949).
- Broyden C. G., The Convergence of a Class of Double-Rank Minimization Algorithms; 1. General Considerations, *J. Inst. Math. Appl.*, **6**, 76 (1970).
- Broyden C. G., The Convergence of a Class of Double-Rank Minimization Algorithms; 2. The New Algorithms, *J. Inst. Math. Appl.*, **6**, 222 (1970).
- Crockett J. G., Chernoff H., Gradient Methods of Maximization, *Pacific J. Math.*, **5**, 33 (1955).
- Curry H. D., The Method of Steepest Descent for Nonlinear Minimization Problems, *Quart. Appl. Math.*, **2**, 258 (1944).
- Dixon L. C. W., Biggs M. C., Meander — A Newton Based Procedure for N -Dimensional Function Minimization, Technical Rept. № 9, The Hatfield Polytechnic, Hatfield, England, April 1970.
- Goldfeld S. M., Quandt R. E., Trotter H. F., Maximization by Quadratic Hill Climbing, *Econometrica*, **34**, 541 (1966).
- Huang H. Y., Levy A. B., *J. Opt. Theory and Appl.*, **6**, 269 (1970).
- Jacobson D. H., Oksman W., An Algorithm That Minimizes Homogeneous Functions of N Variables in $N + 2$ Iterations and Rapidly Minimizes General Functions, Technical Rept. № 618, Division of Engineering and Applied Physics, Harvard Univ., Cambridge, Mass., Oct. 1970.

- Marquardt D. W., An Algorithm for Least Squares Estimation of Nonlinear Parameters, *SIAM J.*, 11, 431 (1963).
- Murtagh B. A., Sargent R. W. H., A Constrained Minimization Method with Quadratic Convergence, Chap. 14 in: Optimization, Fletcher R., ed., Academic Press Inc., London, 1969.
- Murtagh B. A., Sargent R. W. H., *Computer J.*, 13, 185 (1970).
- Papaioannou T., Kempthorne O., Parallel Tangents and Steepest Descent Optimization Algorithm, Wright-Patterson Air Force Base Rept. ARL 70-0117, July 1970.
- Powell M. J. D., An Iterative Method for Finding Stationary Values of a Function of Several Variables, *Computer J.*, Vol. 5, 1962.
- Shah B. V., Buehler R. J., Kempthorne O., Iowa State Univ. Statist. Lab. Tech. Rept. 3, 1961; 2 (rev.), 1962; *J. Soc. Ind. Appl. Math.*, Vol. 12, 1964.
- Shanno D. F., *SIAM J. Numer. Anal.*, 7, 366 (1970).
- Shanno D. F., *Math. Computation*, 24, 647 (1970).
- Shanno D. F., Kettler P. C., *Math. Computation*, 24, 657 (1970).
- Siddall J. N., Optisep Designers Optimization Subroutines, McMaster Univ. Rept. ME/70/DSN/REP/1, Faculty of Engineering, Hamilton, Ontario, Canada, 1970.

Глава 4

МЕТОДЫ МИНИМИЗАЦИИ БЕЗ ОГРАНИЧЕНИЙ, НЕ ИСПОЛЬЗУЮЩИЕ ПРОИЗВОДНЫЕ (МЕТОДЫ ПОИСКА)



В отличие от гл. 3, где изложено решение задачи нелинейного программирования (3.0.1) с помощью методов, использующих производные (или их аппроксимации), в этой главе рассматриваются методы оптимизации, не использующие производные. Эти методы обычно называют *методами поиска*. В типичном методе поиска направления минимизации полностью определяются на основании последовательных вычислений целевой функции $f(x)$.

Как правило, при решении задач нелинейного программирования при отсутствии ограничений градиентные методы и методы, использующие вторые производные, сходятся быстрее, чем прямые методы поиска. Тем не менее, применяя на практике методы, использующие производные, приходится сталкиваться с двумя главными препятствиями. Во-первых, в задачах с достаточно большим числом переменных довольно трудно или даже невозможно получить производные в виде аналитических функций, необходимых для градиентного алгоритма или алгоритма, использующего производные второго порядка. Хотя вычисление аналитических производных можно заменить вычислением производных с помощью разностных схем, как описано в разд. 3.4, возникающая при этом ошибка, особенно в окрестности экстремума, может ограничить применение подобной аппроксимации. В принципе можно использовать символические методы для аналитических выражений производных, но эти методы требуют значительного развития, прежде чем они станут подходящим практическим инструментом. Во всяком случае, методы поиска не требуют регулярности и непрерывности целевой функции и существования производных. Вторым обстоятельством, правда, связанным с предыдущей проблемой, является то, что при использовании методов оптимизации, основанных на вычислении первых и при необходимости вторых производных, требуется по сравнению с методами поиска довольно большое время на подготовку задачи к решению.

Вследствие изложенных выше трудностей были разработаны алгоритмы оптимизации, использующие прямой поиск, которые, хотя и медленнее реализуются в случае простых задач, на практике могут оказаться более удовлетворительными с точки зрения пользователя, чем градиентные методы или методы, использующие

вторые производные, и решение задачи с их помощью может обойтись дешевле, если стоимость подготовки задачи к решению высока по сравнению со стоимостью машинного времени. В данной главе рассмотрены лишь некоторые из многих существующих алгоритмов прямого поиска, причем при выборе мы руководствовались их эффективностью при решении тестовых задач.

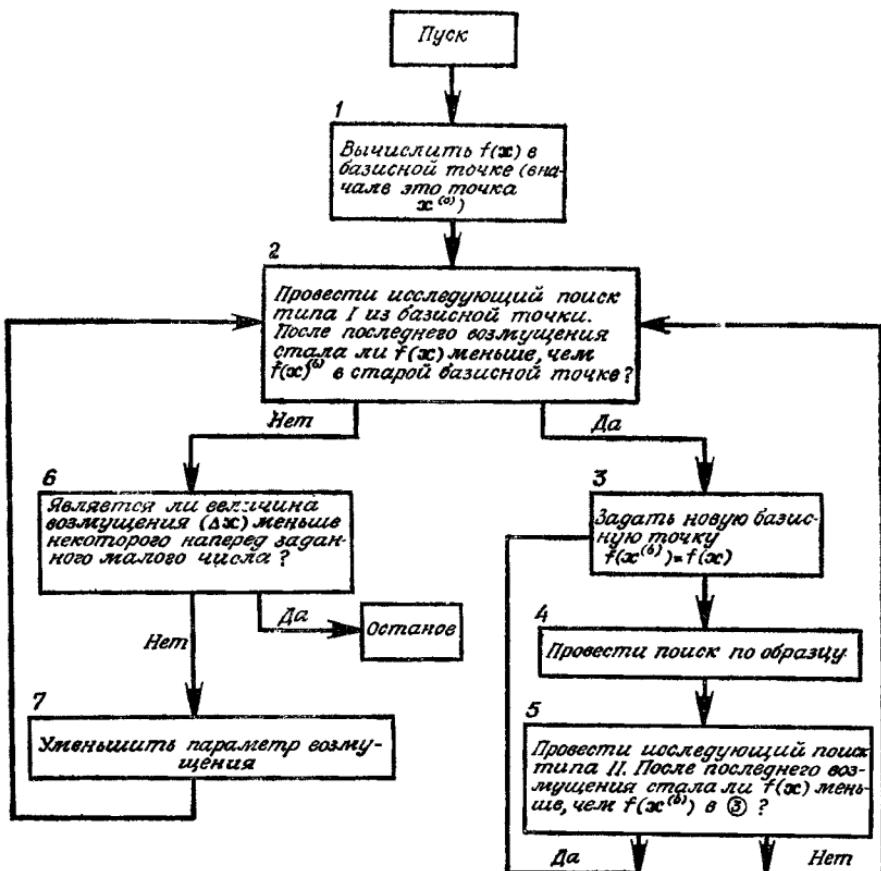
4.1. ПРЯМОЙ ПОИСК

По существу методы поиска простейшего типа заключаются в изменении каждый раз одной переменной, тогда как другие остаются постоянными, пока не будет достигнут минимум. Например, в одном из таких методов переменная x_1 устанавливается постоянной, а x_2 изменяют до тех пор, пока не будет получен минимум. Затем, сохраняя новое значение x_2 постоянным, изменяют x_1 , пока не будет достигнут оптимум при выбранном значении x_2 и т. д. Однако такой алгоритм работает плохо, если имеет место взаимодействие между x_1 и x_2 , т. е. если в выражение для целевой функции входят члены, содержащие произведение x_1x_2 . Таким образом, этот метод нельзя рекомендовать, если пользователь не имеет дела с целевой функцией, в которой взаимодействия не существенны.

Хук и Дживс [1] предложили логически простую стратегию поиска, использующую априорные сведения и в то же время отвергающую устаревшую информацию относительно характера топологии целевой функции в E^n . В интерпретации Вуда [2] этот алгоритм включает два основных этапа: «исследующий поиск» вокруг базисной точки и «поиск по образцу», т. е. в направлении, выбранном для минимизации. На фиг. 4.1.1 представлена упрощенная информационная блок-схема этого алгоритма.

Рассматриваемый алгоритм прямого поиска состоит из следующих операций. Прежде всего задаются начальные значения всех элементов \mathbf{x} , а также начальное приращение $\Delta\mathbf{x}$. Чтобы начать «исследующий поиск», следует вычислить значение функции $f(\mathbf{x})$ в базисной точке (базисная точка представляет собой начальный вектор предполагаемых искомых значений независимых переменных на первом цикле). Затем в циклическом порядке изменяется каждая переменная (каждый раз только одна) на выбранные величины приращений, пока все параметры не будут таким образом изменены. В частности, $x_i^{(0)}$ изменяется на величину $\Delta x_i^{(0)}$, так что $x_i^{(1)} = x_i^{(0)} + \Delta x_i^{(0)}$. Если приращение не улучшает целевую функцию, $x_i^{(0)}$ изменяется на $-\Delta x_i^{(0)}$ и значение $f(\mathbf{x})$ проверяется, как и ранее. Если значение $f(\mathbf{x})$ не улучшают ни $x_i^{(0)} + \Delta x_i^{(0)}$, ни $x_i^{(0)} - \Delta x_i^{(0)}$, то $x_i^{(0)}$ оставляют без изменений. Затем $x_2^{(0)}$ изменяют на величину $\Delta x_2^{(0)}$ и т. д., пока не будут изменены все независимые переменные,

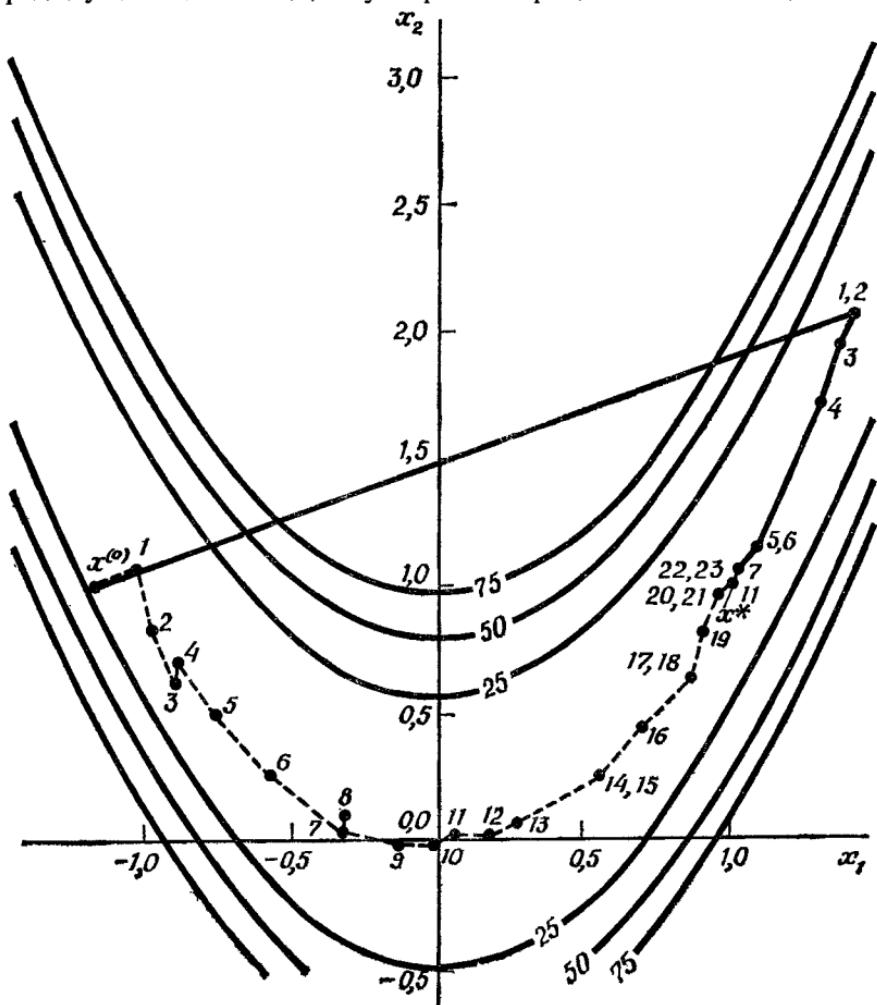
что завершает один исследующий поиск. На каждом шаге или сдвиге по независимой переменной значение целевой функции сравнивается с ее значением в предыдущей точке. Если целевая функция улучшается на данном шаге, то ее старое значение заменяется на новое при последующих сравнениях. Однако если произ-



веденное возмущение по \mathbf{x} неудачно, то сохраняется прежнее значение $f(\mathbf{x})$.

После проведения одного (или более) исследующего поиска применяется стратегия поиска по образцу. Удачные изменения переменных в исследующем поиске [т. е. те изменения переменных, которые уменьшили $f(\mathbf{x})$] определяют вектор в E^n , указывающий некоторое направление минимизации, которое может привести к успеху. Серия ускоряющихся шагов, или поиск по образцу, проводится вдоль этого вектора до тех пор, пока $f(\mathbf{x})$ уменьшается при каждом таком поиске. Длина шага при поиске по образцу

в данном координатном направлении приблизительно пропорциональна числу удачных шагов, имевших место ранее в этом координатном направлении во время исследующих поисков за несколько предыдущих циклов. Для ускорения процесса оптимизации изме-



Фиг. 4.1.2. Прямой поиск минимума функции Розенброка, начиная из точки $x^{(0)} = [-1,2 \ 1,0]^T$. В отмеченных точках указано число удачных шагов поиска по образцу.

нение размера шага Δx в поиске по образцу осуществляется путем введения некоторого множителя при величине Δx , используемой в исследующих поисках. Исследующий поиск, проводимый после поиска по образцу, называется исследующим поиском типа II; успех или неудачу поиска по данному образцу нельзя установить до завершения исследующего поиска типа II.

Если $f(x)$ не уменьшается в процессе исследующего поиска типа II, то говорят, что данный поиск по образцу неудачен, и проводится новый исследующий поиск типа I для определения нового удачного направления. Если исследующий поиск типа I не дает нового удачного направления, то последовательно уменьшают Δx , пока либо можно будет определить новое удачное направление, либо Δx , не станет меньше, чем некоторая заранее установленная допустимая величина. Невозможность уменьшить $f(x)$, когда Δx достаточно мало, указывает на то, что достигнут локальный оптимум. Описанная последовательность поисков заканчивается, если оказываются удовлетворенными условия трех основных тестов. Первый тест проводится после каждого исследующего поиска и поиска по образцу: изменение целевой функции сравнивается с заранее установленной малой величиной. Если значение целевой функции не отличается на величину, большую, чем это число, от предыдущего основного значения целевой функции, исследующий поиск или поиск по образцу считается неудачным. В противном случае проводится тест для определения, увеличилась ли целевая функция (неудача) или уменьшилась (удачный поиск). Этот второй тест нужен для того, чтобы быть уверенными, что значение целевой функции все время улучшается. Третий тест проводится после неудачи в исследующем поиске на стадии уменьшения изменения Δx . Поиск может быть закончен, если на данном шаге изменение каждой переменной $\Delta x_i^{(k)}$ оказывается меньше, чем некоторое заранее определенное число.

На фиг. 4.1.2 приведена траектория прямого поиска при минимизации функции Розенброка, начиная с вектора $x^{(0)} = [-1,2 \ 1,0]^T$.

Пример 4.1.1. Максимизация (без ограничений) прямым поиском по Хуку и Дживсу

Максимизировать целевую функцию

$$f(x) = \frac{1}{(x_1 + 1)^2 + x_2^2}.$$

начиная из $x^{(0)} = [2,00 \ 2,80]^T$ с начальным Δx , равным $[0,60 \ 0,84]^T$. Исходное значение $f(2,00; 2,80)$ в базисной точке $x^{(0)}$ равно 0,059. Сначала проводится исследующий поиск типа I для определения удачного направления. (Такая процедура называется исследующим поиском типа I в противоположность исследующему поиску типа II, который следует за поиском по образцу. После проведения исследующего поиска типа II принимается решение по поводу того, было ли предыдущее движение по образцу успешным

или неудачным.)

$$x_1^{(1)} = 2,00 + 0,60 = 2,50, \quad f(2,60; 2,80) = 0,048 \text{ (неудача);}$$

$$x_1^{(1)} = 2,0 - 0,60 = 1,40, \quad f(1,40; 2,80) = 0,073 \text{ (успех);}$$

$$x_2^{(1)} = 2,80 + 0,84 = 3,64, \quad f(1,50; 3,64) = 0,052 \text{ (неудача);}$$

$$x_2^{(1)} = 2,80 - 0,84 = 1,96, \quad f(1,40; 1,96) = 0,104 \text{ (успех).}$$

Исследующий поиск оказался удачным. Заметим, что при каждом поиске выбирается последний удачный вектор \mathbf{x} . Новым базисным вектором будет $(1,40; 1,96)$.

Теперь из точки $(1,40; 1,96)$ проводится поиск по образцу в соответствии с правилом акселерации

$$x_i^{(k+1)} = 2x_i^{(k)} - x_i^{(b)},$$

где $x_i^{(b)}$ — предыдущий базисный вектор \mathbf{x} . В данном случае это начальный вектор $\mathbf{x}^{(0)}$.

$$x_1^{(2)} = 2(1,40) - 2,00 = 0,80,$$

$$x_2^{(2)} = 2(1,96) - 2,80 = 1,12,$$

$$f(0,8; 1,12) = 0,22.$$

Наконец проводится исследующий поиск типа II; неудача или успех его оценивается путем сравнения с $f(0,8; 1,12) = 0,22$:

$$x_1^{(3)} = 0,80 + 0,60 = 1,40, \quad f(1,40; 1,12) = 0,14 \text{ (неудача);}$$

$$x_1^{(3)} = 0,80 - 0,60 = 0,20, \quad f(0,20; 1,12) = 0,38 \text{ (успех);}$$

$$x_2^{(3)} = 1,12 + 0,84 = 1,96, \quad f(0,20; 1,96) = 0,19 \text{ (неудача);}$$

$$x_2^{(3)} = 1,12 - 0,84 = 0,28, \quad f(0,20; 0,28) = 0,67 \text{ (успех).}$$

Чтобы определить, оказался ли поиск по образцу успешным, сравнивают $f(0,20; 0,28) = 0,67$ с $f(1,40; 1,96) = 0,104$. Поскольку поиск по образцу успешен, то новой базисной точкой будет $\mathbf{x}^{(3)} = [0,20, 0,28]^T$; при этом старая базисная точка представлена вектором $\mathbf{x}^{(1)} = [1,40, 1,96]^T$.

Затем вновь проводится поиск по образцу:

$$x_1^{(4)} = 2(0,20) - 1,40 = -1,00,$$

$$x_2^{(4)} = 2(0,28) - 1,96 = -1,40,$$

$$f(-1,00; -1,40) = 0,51.$$

После этого проводится исследующий поиск типа II:

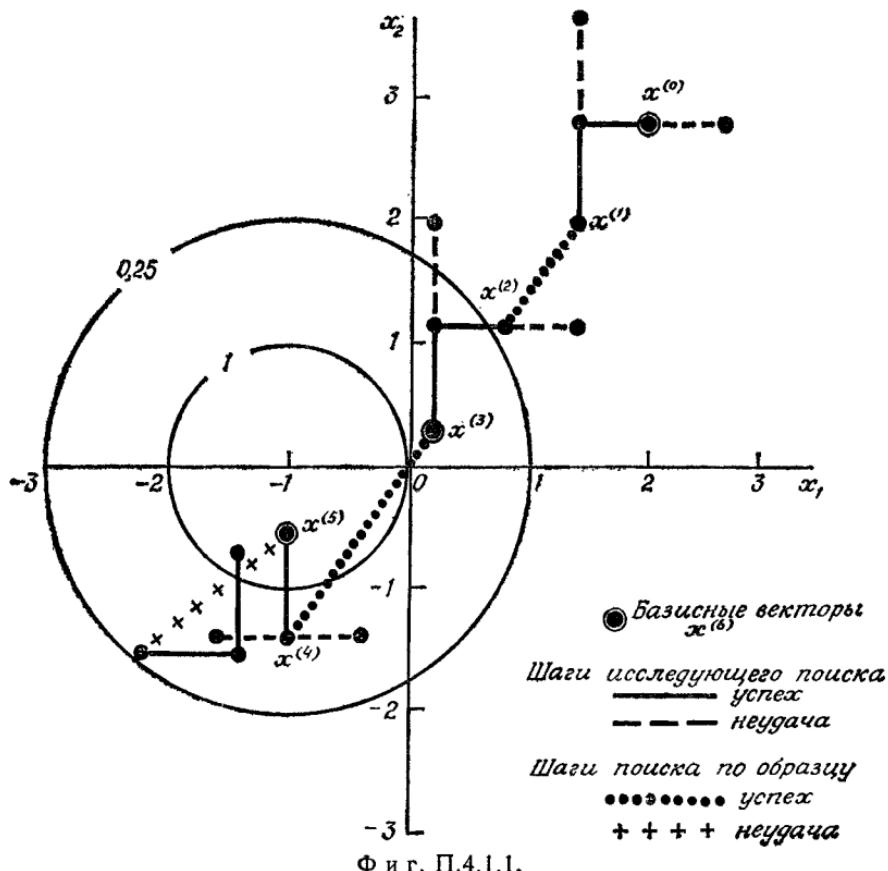
$$x_1^{(5)} = -1,00 + 0,60 = -0,40, \quad f(-0,40; -1,40) = 0,43 \text{ (неудача);}$$

$$x_1^{(5)} = -1,00 - 0,60 = -1,60, \quad f(-1,60; -1,40) = 0,43 \text{ (неудача);}$$

$$x_2^{(5)} = -1,40 + 0,84 = -0,56, \quad f(-1,00; -0,56) = 3,18 \text{ (успех).}$$

Поскольку $f(-1,00; -0,56) = 3,18 > f(0,20; 0,28) = 0,67$, поиск по образцу представляется успешным и $x^{(5)} = [-1,00 \ -0,56]^T$ становится новой базисной точкой, а $x^{(3)}$ — старой базисной точкой.

Эта последовательность поисков продолжается до тех пор, пока не будет достигнута ситуация, при которой в конце исследующего поиска типа II значение $f(x)$ окажется меньшим, чем значение $f(x^{(b)})$ в последней базисной точке. Тогда, если даже исследующий поиск типа II является успешным при одном или более возмущениях, говорят, что последний поиск по образцу неудачен и проводят из предыдущей базисной точки исследующий поиск типа I для определения нового удачного направления. В иллюстративных целях продолжим поиск из $x^{(5)} = [-1,00 \ -0,56]^T$.



Поиск по образцу:

$$x_1^{(6)} = 2(-1,00) - 0,20 = -2,20,$$

$$x_2^{(6)} = 2(-0,56) - 0,28 = -1,40,$$

$$f(-2,20; -1,40) = 0,29.$$

Исследующий поиск типа II:

$$x_1^{(7)} = -2,20 + 0,60 = -1,60, \quad f(-1,60; -1,40) = 0,43 \text{ (успех);}$$

$$x_2^{(7)} = -1,40 + 0,84 = -0,56, \quad f(-1,60; -0,56) = 1,49 \text{ (успех).}$$

Однако поскольку $f(-1,60; -0,56) = 1,49 < f(-1,00; -0,56) = 3,18$, то, несмотря на то что исследующий поиск типа II оказался успешным, поиск по образцу считается неудачным, и из $\mathbf{x}^{(5)} = [-1,00 \ -0,56]^T$ начинают исследующий поиск типа I.

Когда достигается стадия, на которой ни исследующий поиск типа I, ни поиск по образцу (вместе с исследующим поиском типа II) не являются успешными в любом координатном направлении, говорят, что они оба неудачны и возмущение $\Delta \mathbf{x}$ уменьшают следующим образом:

$$\Delta x_{i,\text{новое}} = \Delta x_{i,\text{предыдущее}} \frac{\Delta x_i^{(0)}}{e^{\xi}},$$

где ξ — число последовательных неудач в исследующих поисках при данной величине шага, начиная от последнего успешного исследующего поиска.

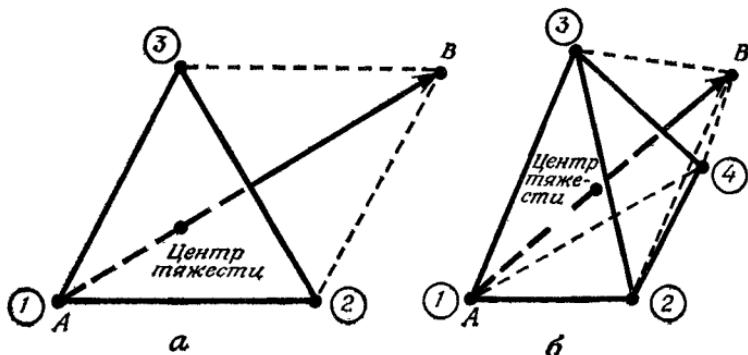
(В этом примере максимум $f(\mathbf{x}) \rightarrow \infty$ при $x_1 \rightarrow -1$ и $x_2 \rightarrow 0$.)

4.2. ПОИСК ПО ДЕФОРМИРУЕМОМУ МНОГОГРАННИКУ

Нелдер и Мид [3] предложили метод поиска, несколько более сложный по сравнению с прямым поиском, но оказавшийся весьма эффективным и легко осуществляемым на ЭВМ. Чтобы читатель смог оценить стратегию Нелдера и Мида, кратко опишем симплексный поиск Спенди, Хекста и Химсвортса [4], разработанный в связи со статистическим планированием эксперимента. Вспомним, что регулярные многогранники в E^n являются симплексами. Например, как видно из фиг. 4.2.1, для случая двух переменных регулярный симплекс представляет собой равносторонний треугольник (три точки); в случае трех переменных регулярный симплекс представляет собой тетраэдр (четыре точки) и т. д.

При поиске минимума целевой функции $f(\mathbf{x})$ пробные векторы \mathbf{x} могут быть выбраны в точках E^n , находящихся в вершинах симплекса, как было первоначально предложено Спенди, Хекстом и

Химсвортом. Из аналитической геометрии известно, что координаты вершин регулярного симплекса определяются следующей матрицей D , в которой столбцы представляют собой вершины, пронумерованные от 1 до $(n + 1)$, а строчки — координаты, i при-



Ф и г. 4.2.1. Регулярные симплексы для случая двух (а) и трех (б) независимых переменных.

① обозначает наибольшее значение $f(x)$. Стрелка указывает направление наискорейшего улучшения.

нимает значения от 1 до n :

$$D = \begin{bmatrix} 0 & d_1 & d_2 & \dots & d_n \\ 0 & d_2 & d_1 & \dots & d_n \\ 0 & d_2 & d_2 & \dots & d_n \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & d_n & d_2 & \dots & d_1 \end{bmatrix} \text{ — матрица } n \times (n + 1),$$

где

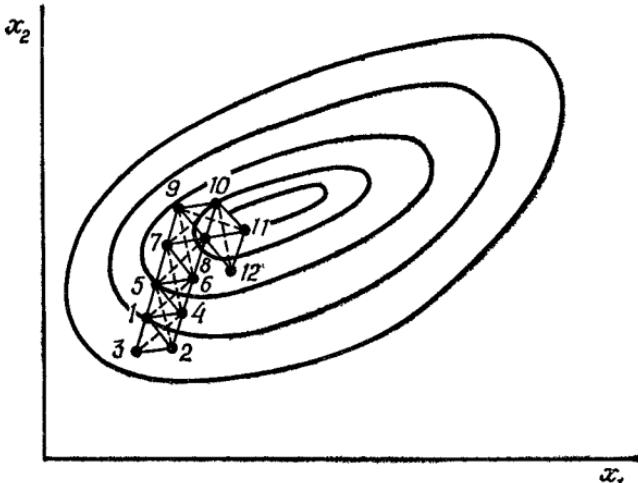
$$d_1 = \frac{t}{n\sqrt{2}} (\sqrt{n+1} + n - 1),$$

$$d_2 = \frac{t}{n\sqrt{2}} (\sqrt{n+1} - 1),$$

t — расстояние между двумя вершинами. Например, для $n = 2$ и $t = 1$ треугольник, приведенный на фиг. 4.2.1, имеет следующие координаты:

Вершина	x_1, i	x_2, i
1	0	0
2	0,965	0,259
3	0,259	0,965

Целевая функция может быть вычислена в каждой из вершин симплекса; из вершины, где целевая функция максимальна (точка A на фиг. 4.2.1), проводится проектирующая прямая через центр тяжести симплекса. Затем точка A исключается и строится новый симплекс, называемый *отраженным*, из оставшихся прежних точек и одной новой точки B , расположенной на проектирующей прямой на надлежащем расстоянии от центра тяжести. Продолже-



Фиг. 4.2.2. Последовательность регулярных симплексов, полученных при минимизации $f(x)$.

— — — проекция.

ние этой процедуры, в которой каждый раз вычеркивается вершина, где целевая функция максимальна, а также использование правил уменьшения размера симплекса и предотвращения циклического движения в окрестности экстремума позволяют осуществить поиск, не использующий производные и в котором величина шага на любом этапе k фиксирована, а направление поиска можно изменять. На фиг. 4.2.2 приведены последовательные симплексы, построенные в двумерном пространстве с «хорошей» целевой функцией.

Определенные практические трудности, встречающиеся при использовании регулярных симплексов, а именно отсутствие ускорения поиска и трудности при проведении поиска на искривленных «оврагах» и «хребтах», привели к необходимости некоторых улучшений методов [5]. В этом разделе мы изложим метод Нелдера и Мида, в котором симплекс может изменять свою форму и таким образом уже не будет оставаться симплексом. Именно поэтому здесь использовано более подходящее название «деформируемый многогранник».

В методе Нелдера и Мида минимизируется функция n независимых переменных с использованием $n+1$ вершин деформируемого многогранника в E^n . Каждая вершина может быть идентифицирована

вектором \mathbf{x} . Вершина (точка) в E^n , в которой значение $f(\mathbf{x})$ максимально, проектируется через центр тяжести (центроид) оставшихся вершин. Улучшенные (более низкие) значения целевой функции находятся последовательной заменой точки с максимальным значением $f(\mathbf{x})$ на более «хорошие» точки, пока не будет найден минимум $f(\mathbf{x})$.

Более подробно этот алгоритм может быть описан следующим образом.

Пусть $\mathbf{x}_i^{(k)} = [x_{1i}^{(k)}, \dots, x_{ij}^{(k)}, \dots, x_{in}^{(k)}]^T$, $i = 1, \dots, n+1$, является i -й вершиной (точкой) в E^n на k -м этапе поиска, $k = 0, 1, \dots$, и пусть значение целевой функции в $\mathbf{x}_i^{(k)}$ равно $f(\mathbf{x}_i^{(k)})$. Кроме того, отметим те векторы \mathbf{x} многогранника, которые дают максимальное и минимальное значения $f(\mathbf{x})$.

Определим

$$f(\mathbf{x}_h^{(k)}) = \max \{f(\mathbf{x}_1^{(k)}), \dots, f(\mathbf{x}_{n+1}^{(k)})\},$$

где $\mathbf{x}_h^{(k)} = \mathbf{x}_i^{(k)}$, и

$$f(\mathbf{x}_l^{(k)}) = \min \{f(\mathbf{x}_1^{(k)}), \dots, f(\mathbf{x}_{n+1}^{(k)})\},$$

где $\mathbf{x}_l^{(k)} = \mathbf{x}_i^{(k)}$. Поскольку многогранник в E^n состоит из $(n+1)$ вершин $\mathbf{x}_1, \dots, \mathbf{x}_{n+1}$, пусть \mathbf{x}_{n+2} будет центром тяжести всех вершин, исключая \mathbf{x}_h .

Тогда координаты этого центра определяются формулой

$$\mathbf{x}_{n+2,j}^{(k)} = \frac{1}{n} \left[\left(\sum_{i=1}^{n+1} \mathbf{x}_{ij}^{(k)} \right) - \mathbf{x}_h^{(k)} \right], \quad j = 1, \dots, n, \quad (4.2.1)$$

где индекс j обозначает координатное направление.

Начальный многогранник обычно выбирается в виде регулярного симплекса (но это не обязательно) с точкой 1 в качестве начала координат; можно начало координат поместить в центре тяжести, как это имеет место в машинной программе в приложении Б. Процедура отыскания вершины в E^n , в которой $f(\mathbf{x})$ имеет лучшее значение, состоит из следующих операций:

1. *Отражение* — проектирование $\mathbf{x}_h^{(k)}$ через центр тяжести в соответствии с соотношением

$$\mathbf{x}_{n+3}^{(k)} = \mathbf{x}_{n+2}^{(k)} + \alpha (\mathbf{x}_{n+2}^{(k)} - \mathbf{x}_h^{(k)}), \quad (4.2.2)$$

где $\alpha > 0$ является коэффициентом отражения; $\mathbf{x}_{n+2}^{(k)}$ — центр тяжести, вычисляемый по формуле (4.2.1); $\mathbf{x}_h^{(k)}$ — вершина, в которой функция $f(\mathbf{x})$ принимает наибольшее из $n+1$ ее значений на k -м этапе.

2. *Растяжение*. Эта операция заключается в следующем: если $f(\mathbf{x}_{n+3}^{(k)}) \leq f(\mathbf{x}_l^{(k)})$, то вектор $(\mathbf{x}_{n+3}^{(k)} - \mathbf{x}_{n+2}^{(k)})$ растягивается в соответствии с соотношением

$$\mathbf{x}_{n+4}^{(k)} = \mathbf{x}_{n+2}^{(k)} + \gamma (\mathbf{x}_{n+3}^{(k)} - \mathbf{x}_{n+2}^{(k)}), \quad (4.2.3)$$

где $\gamma > 1$ представляет собой коэффициент растяжения. Если $f(\mathbf{x}_{n+4}^{(k)}) < f(\mathbf{x}_i^{(k)})$, то $\mathbf{x}_h^{(k)}$ заменяется на $\mathbf{x}_{n+4}^{(k)}$ и процедура продолжается снова с операции 1 при $k = k + 1$. В противном случае $\mathbf{x}_h^{(k)}$ заменяется на $\mathbf{x}_{n+3}^{(k)}$ и также осуществляется переход к операции 1 при $k = k + 1$.

3. Сжатие. Если $f(\mathbf{x}_{n+3}^{(k)}) > f(\mathbf{x}_i^{(k)})$ для всех $i \neq h$, то вектор $(\mathbf{x}_h^{(k)} - \mathbf{x}_{n+2}^{(k)})$ сжимается в соответствии с формулой

$$\mathbf{x}_{n+5}^{(k)} = \mathbf{x}_{n+2}^{(k)} + \beta (\mathbf{x}_h^{(k)} - \mathbf{x}_{n+2}^{(k)}), \quad (4.2.4)$$

где $0 < \beta < 1$ представляет собой коэффициент сжатия. Затем $\mathbf{x}_h^{(k)}$ заменяется на $\mathbf{x}_{n+5}^{(k)}$ и возвращаемся к операции 1 для продолжения поиска на $(k + 1)$ -м шаге.

4. Редукция. Если $f(\mathbf{x}_{n+3}^{(k)}) > f(\mathbf{x}_i^{(k)})$, все векторы $(\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k)})$, $i = 1, \dots, n+1$, уменьшаются в 2 раза с отсчетом от $\mathbf{x}_i^{(k)}$ в соответствии с формулой

$$\mathbf{x}_i^{(k)} = \mathbf{x}_i^{(k)} + 0.5 (\mathbf{x}_i^{(k)} - \mathbf{x}_i^{(k)}), \quad i = 1, \dots, n+1. \quad (4.2.5)$$

Затем возвращаемся к операции 1 для продолжения поиска на $(k + 1)$ -м шаге:

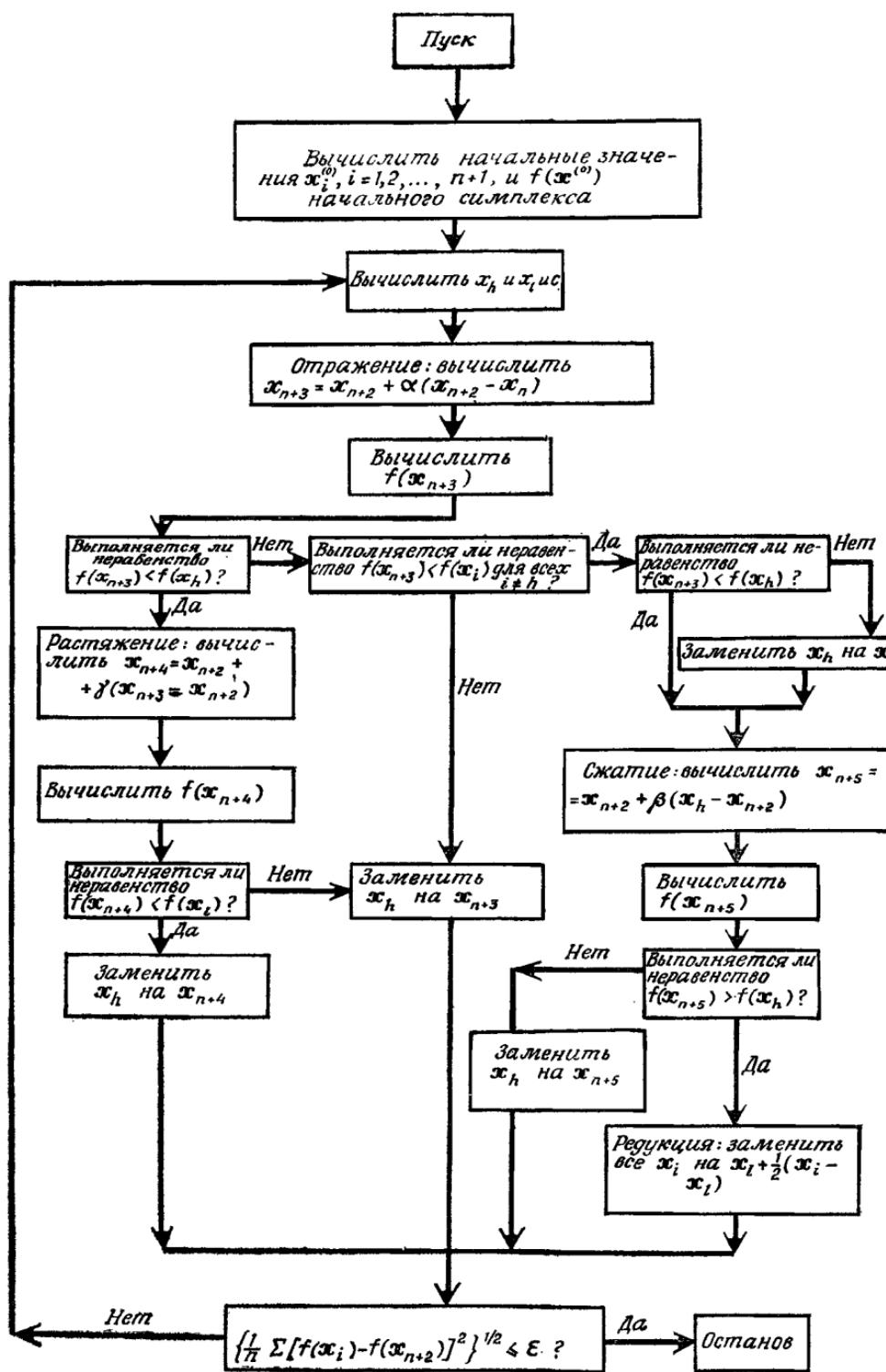
Критерий окончания поиска, использованный Нелдером и Мидом, состоял в проверке условия

$$\left\{ \frac{1}{n+1} \sum_{i=1}^{n+1} [f(\mathbf{x}_i^{(k)}) - f(\mathbf{x}_{n+2}^{(k)})]^2 \right\}^{1/2} \leq \varepsilon, \quad (4.2.6)$$

где ε — произвольное малое число, а $f(\mathbf{x}_{n+2}^{(k)})$ — значение целевой функции в центре тяжести $\mathbf{x}_{n+2}^{(k)}$.

На фиг. 4.2.3 приведена блок-схема поиска методом деформируемого многогранника, а на фиг. 4.2.4 показана последовательность поиска для функции Розенброка, начиная из $\mathbf{x}^{(0)} = [-1, 2, 1, 0]^T$. Деформируемый многогранник в противоположность жесткому симплексу адаптируется к топографии целевой функции, вытягиваясь вдоль длинных наклонных плоскостей, изменяя направление в изогнутых впадинах и сжимаясь в окрестности минимума.

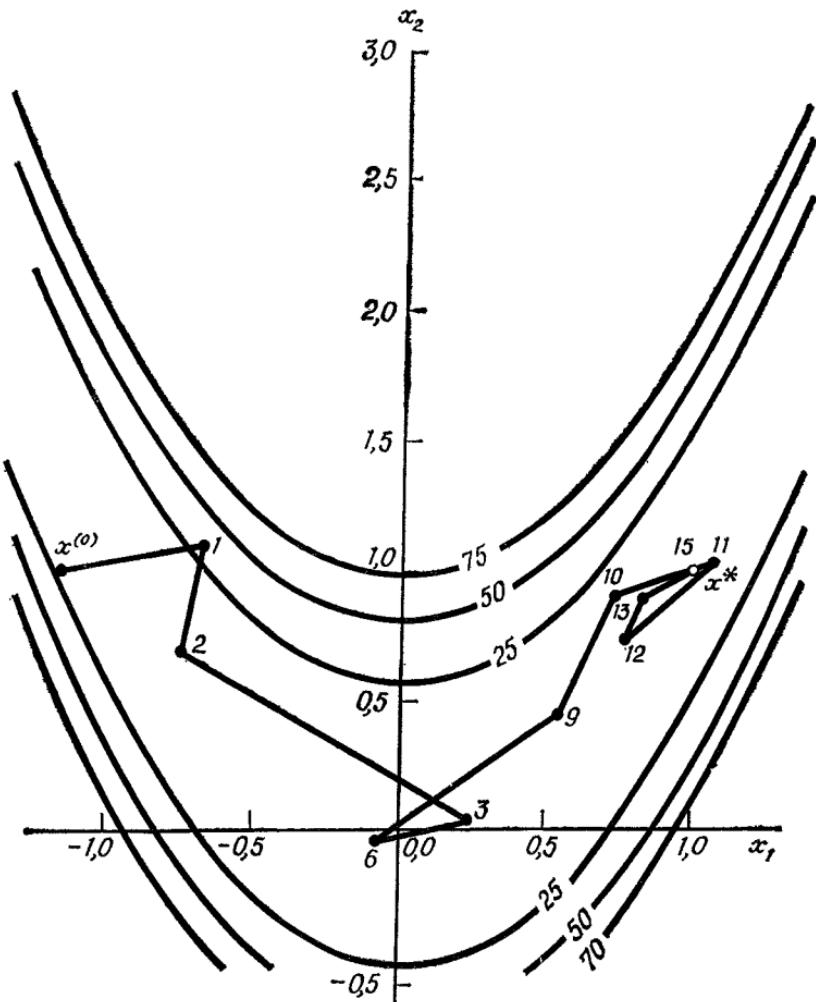
Коэффициент отражения α используется для проектирования вершины с наибольшим значением $f(\mathbf{x})$ через центр тяжести деформируемого многогранника. Коэффициент γ вводится для растяжения вектора поиска в случае, если отражение дает вершину со значением $f(\mathbf{x})$, меньшим, чем наименьшее значение $f(\mathbf{x})$, полученное до отражения. Коэффициент сжатия β используется для уменьшения вектора поиска, если операция отражения не привела к вершине со значением $f(\mathbf{x})$, меньшим, чем второе по величине (после наибольшего) значение $f(\mathbf{x})$, полученное до отражения. Таким образом, с помощью операций растяжения или сжатия размеры и форма



Фиг. 4.2.3. Информационная блок-схема поиска методом деформируемого мно-
гогранника

деформируемого многогранника масштабируются так, чтобы они удовлетворяли топологии решаемой задачи.

Естественно возникает вопрос, какие значения параметров α , β и γ должны быть выбраны. После того как деформируемый



Фиг. 4.2.4. Поиск минимума функции Розенброка методом деформируемого многогранника, начиная с точки $x^{(0)} = [-1,2 \ 1,0]^T$ (числа указывают номер шага).

многогранник подходящим образом промасштабирован, его размеры должны поддерживаться неизменными, пока изменения в топологии задачи не потребуют применения многогранника другой формы. Это возможно реализовать только при $\alpha = 1$. Кроме того, Нелдер и Мид показали, что при решении задачи с $\alpha = 1$ требуется меньшее количество вычислений функции, чем при $\alpha < 1$. С другой

стороны, α не должно быть много больше единицы, поскольку 1) деформируемый многогранник легче адаптируется к топологии задачи при меньших значениях α , особенно когда необходимо изменять направление поиска, столкнувшись с изогнутой впадиной, и 2) в области локального минимума размеры многогранника должны уменьшаться и большое α в этих условиях замедлит сходимость. Таким образом, значение $\alpha = 1$ выбирается как компромисс.

Таблица 4.2.1

Влияние β и γ на решение тестовой задачи 3 (приложение А)¹⁾

β	γ	$f(x^*)$	Количество шагов k	Время решения, с	Точность определения x^* и $f(x^*)$
0,2	3	58,903	90	0,381	$2,7 \cdot 10^{-7}$
0,4	3	58,903	55	0,287	$7,1 \cdot 10^{-8}$
0,6	3	58,903	81	0,388	$2,8 \cdot 10^{-7}$
0,8	3	58,903	95	0,414	$9,3 \cdot 10^{-7}$
0,5	2,2	58,903	47	0,238	$9,1 \cdot 10^{-7}$
0,5	2,4	58,903	45	0,247	$6,5 \cdot 10^{-7}$
0,5	2,6	58,903	49	0,287	$8,5 \cdot 10^{-7}$
0,5	2,8	58,903	57	0,304	$8,1 \cdot 10^{-7}$
0,5	3,0	58,903	70	0,343	$8,6 \cdot 10^{-7}$
0,5	3,2	58,903	46	0,266	$8,5 \cdot 10^{-7}$

1) Другие параметры, использованные при решении тестовой задачи 3, имели следующие значения: $\alpha = 1$, $t = 0,5$, $e = 10^{-5}$. В качестве начальной взята точка $x^{(0)} = [90 \quad 10]^T$.

Чтобы выяснить, какое влияние на процедуру поиска имеет выбор β и γ , Нелдер и Мид (а также Павиани [6]) провели решение нескольких тестовых задач, используя большое число различных комбинаций значений β и γ . В качестве удовлетворительных значений этих параметров при оптимизации без ограничений Нелдер и Мид рекомендовали $\alpha = 1$, $\beta = 0,5$ и $\gamma = 2$. Размеры и ориентация исходного многогранника в некоторой степени влияли на время решения, а значения α , β и γ оказывали значительно большее влияние. Павиани отмечает, что нельзя четко решить вопрос относительно выбора β и γ и что влияние выбора β на эффективность поиска несколько более заметно, чем влияние γ . В табл. 4.2.1 приведены типичные результаты для тестовой задачи 3 (см. приложение А), в которой в качестве оптимизирующей подпрограммы использовался алгоритм Нелдера и Мида. Павиани рекомендует следующие диапазоны значений для этих параметров:

$$0,4 \leq \beta \leq 0,6,$$

$$2,8 \leq \gamma \leq 3,0.$$

При $0 < \beta < 0,4$ существует вероятность того, что из-за уплощения многогранника будет иметь место преждевременное окончание процесса. При $\beta > 0,6$ может потребоваться избыточное число шагов и больше машинного времени для достижения окончательного решения.

Пример 4.2.1. Поиск методом деформируемого многогранника

Для иллюстрации метода Нелдера и Мида рассмотрим задачу минимизации функции $f(x) = 4(x_1 - 5)^2 + (x_2 - 6)^2$, имеющей минимум в точке $x^* = [5 \ 6]^T$. Поскольку $f(x)$ зависит от двух переменных, в начале поиска используется многоугольник с тремя вершинами. В этом примере в качестве начального многогранника взят треугольник с вершинами $x_1^{(0)} = [8 \ 9]^T$, $x_2^{(0)} = [10 \ 11]^T$ и $x_3^{(0)} = [8 \ 11]^T$, хотя можно было бы использовать любую другую конфигурацию из трех точек.

На нулевом этапе поиска, $k = 0$, вычисляя значения функции, получаем $f(8, 9) = 45$, $f(10, 11) = 125$ и $f(8, 11) = 65$. Затем отражаем $x_2^{(0)} = [10 \ 11]^T$ через центр тяжести точек $x_1^{(0)}$ и $x_3^{(0)}$ [по формуле (4.2.1)], который обозначим через $x_4^{(0)}$:

$$x_{4,1}^{(0)} = \frac{1}{2} [(8 + 10 + 8) - 10] = 8,$$

$$x_{4,2}^{(0)} = \frac{1}{2} [(9 + 11 + 11) - 11] = 10$$

с тем, чтобы получить $x_5^{(0)}$.

$$x_{5,1}^{(0)} = 8 + 1(8 - 10) = 6,$$

$$x_{5,2}^{(0)} = 10 + 1(10 - 11) = 9,$$

$$f(6, 9) = 13.$$

Поскольку $f(6, 9) = 13 < f(8, 9) = 45$, переходим к операции растяжения:

$$x_{6,1}^{(0)} = 8 + 2(6 - 8) = 4,$$

$$x_{6,2}^{(0)} = 10 + 2(9 - 10) = 8,$$

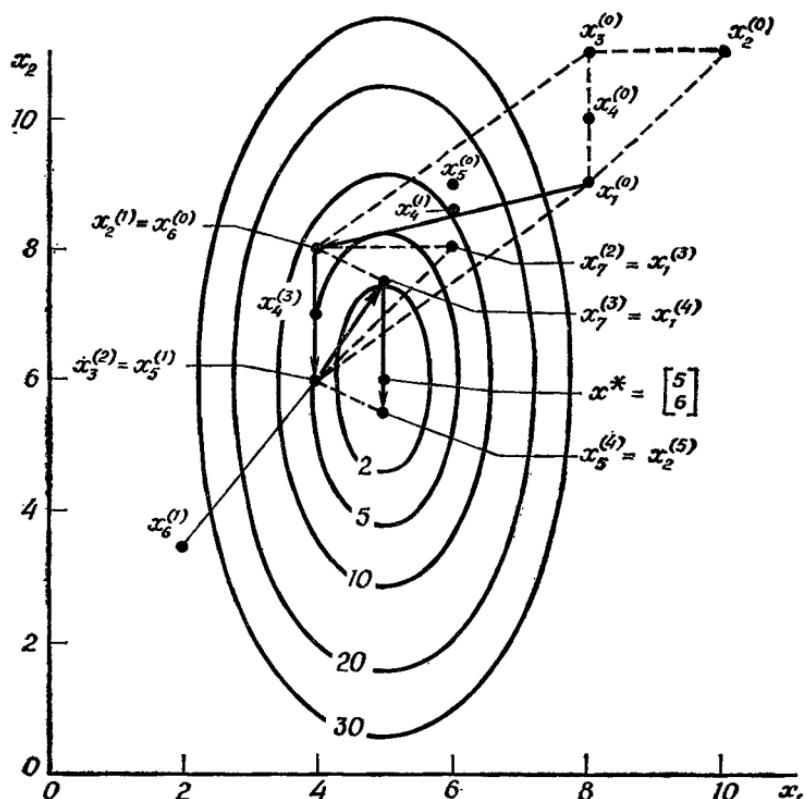
$$f(4, 8) = 8.$$

Поскольку $f(4, 8) = 8 < f(8, 9) = 45$, заменяем $x_2^{(0)}$ на $x_6^{(0)}$ и полагаем $x_6^{(0)} = x_1^{(1)}$ на следующем этапе поиска.

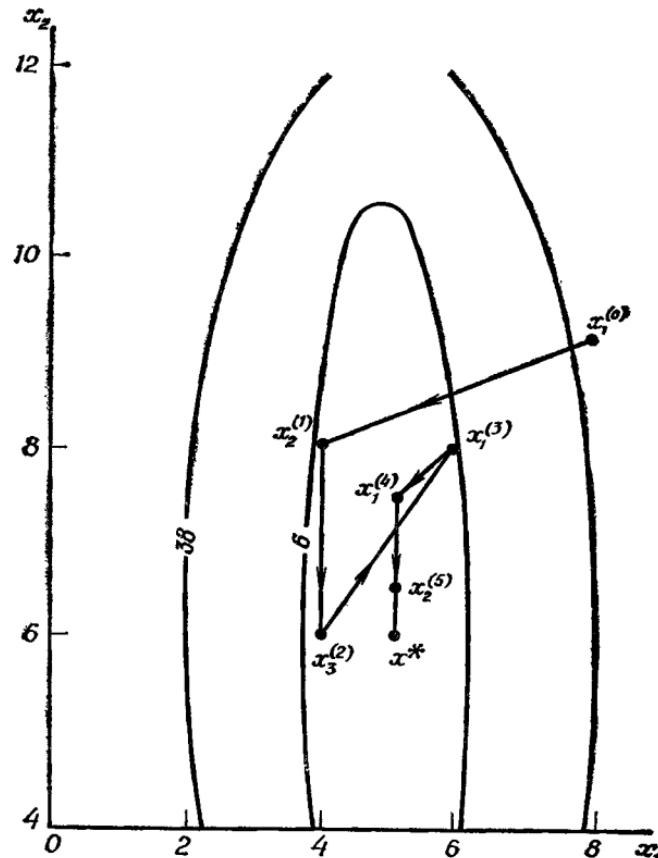
Наконец, поскольку

$$\frac{1}{3} [7^2 + 13^2 + 44^2]^{1/2} = 26,8 > 10^{-6},$$

$$f(x) = 4x_1^2 + x_2^2 - 40x_1 - 12x_2 + 136$$



Ф и г. П.4.2.1а. Метод Нелдера и Мида при отсутствии ограничений.



Ф и г. П.4.2.1б. Траектория поиска с помощью алгоритма Нелдера и Мида.

начинаем этап поиска $k = 1$. На фиг. П 4.2.1а приведена траектория поиска на начальных этапах, а в табл. П.4.2.1 приведены координаты вершин и значения $f(x)$ для четырех дополнительных

Таблица П.4.2.1

Пять этапов вычисления методом Нелдера и Мида при минимизации функции $f(x) = 4(x_1 - 5)^2 + (x_2 - 6)^2$

Этап	Вершины многогранника			Значения $f(x)$	Замечания
		x_1	x_2		
0	$x_1^{(0)}$	8	9	45	$x_l = x_1^{(0)}$
0	$x_2^{(0)}$	10	11	125	$x_h = x_2^{(0)}$
0	$x_3^{(0)}$	8	11	65	
0	$x_5^{(0)}$	6	9	13	Получена отражением с помощью формулы (4.2.2)
0	$x_2^{(1)} = x_6^{(0)}$	4	8	8	Получена растяжением по формуле (4.2.3) $x_2^{(0)}$ заменяется на $x_6^{(0)} = x_2^{(1)}$
1	$x_3^{(2)} = x_5^{(1)}$	4	6	4	$x_3^{(0)}$ заменяется на $x_5^{(1)} = x_3^{(2)}$
2	$x_1^{(3)} = x_7^{(2)}$	6	8	8	$x_1^{(0)}$ заменяется на $x_7^{(3)} = x_1^{(4)}$
3	$x_1^{(4)} = x_7^{(3)}$	5	7,5	2,25	$x_1^{(3)}$ заменяется на $x_7^{(4)}$
4	$x_2^{(5)} = x_5^{(4)}$	5	5,5	0,25	$x_2^{(1)}$ заменяется на $x_5^{(5)}$

этапов. На фиг. П 4.2.1б изображена полная траектория поиска до его окончания. Для уменьшения $f(x)$ до значения $1 \cdot 10^{-6}$ потребовалось 32 этапа.

4.3. МЕТОДЫ РОЗЕНБРОКА И ДЭВИСА, СВЕННА, КЕМПИ

Метод Розенброка [7] является итерационной процедурой, имеющей некоторое сходство с исследующим поиском Хука и Дживса (изложенным в разд. 4.1), которое состоит в том, что предпринимаются малые шаги во время поиска в ортогональных направлениях. Однако здесь вместо непрерывного поиска в координатах, соответствующим направлениям независимых переменных, после каждого цикла координатного поиска можно сделать улучшение путем сведения направлений поиска в ортогональную систему, принимая весь шаг предыдущего этапа в качестве первого блока

при построении новой системы координат. Метод Розенброка определяет местонахождение $x^{(k+1)}$, используя последовательные одномерные поиски, начиная с исходной точки $x^{(k)}$, вдоль системы ортонормированных направлений $\hat{s}_1^{(k)}, \dots, \hat{s}_n^{(k)}$, полученных при помощи процедуры Грама — Шмидта, так что направления поиска вытягиваются вдоль главных осей квадратичной аппроксимации целевой функции. Поскольку эти оси являются собственными векторами $H(x)$ и представляют собой особый случай сопряженных направлений, этот метод в некоторой степени аналогичен методу сопряженных направлений в смысле сходимости, если применяется к квадратичной аппроксимации целевой функции.

Рассматриваемый метод реализуется следующим образом.

Пусть $\hat{s}_1^{(k)}, \dots, \hat{s}_n^{(k)}$ — единичные векторы в E^n , где индекс $k = 0, 1, \dots$ обозначает этапы поиска. Ортонормированные векторы $\hat{s}_1^{(k)}, \dots, \hat{s}_n^{(k)}$ строятся на основании информации, полученной на $(k - 1)$ -м этапе, по формулам (4.3.1) и (4.3.2), приведенным ниже. На начальном этапе $k = 0$ направления $\hat{s}_1^{(0)}, \dots, \hat{s}_n^{(0)}$ обычно берутся параллельными осям x_1, \dots, x_n . В общем случае ортогональные направления поиска могут быть выражены как комбинации координат независимых переменных следующим образом:

$$\hat{s}_1^{(k)} = a_{11}^{(k)}\delta_1 + a_{12}^{(k)}\delta_2 + \dots + a_{1n}^{(k)}\delta_n,$$

$$\hat{s}_2^{(k)} = a_{21}^{(k)}\delta_1 + a_{22}^{(k)}\delta_2 + \dots + a_{2n}^{(k)}\delta_n,$$

.....

$$\hat{s}_n^{(k)} = a_{n1}^{(k)}\delta_1 + a_{n2}^{(k)}\delta_2 + \dots + a_{nn}^{(k)}\delta_n,$$

где δ_i — единичный вектор в направлении x_i , а a_{ij} представляют собой направляющие косинусы \hat{s}_i . В матричном обозначении

$$\hat{s}^{(k)} = \mathbf{a}^{(k)}\delta.$$

Рассмотрим k -й этап, и пусть $x_0^{(k)} = x_n^{(k-1)}$ представляет собой точку в E^n , из которой начал поиск, а $\lambda_1, \dots, \lambda_n$ — соответствующие длины шагов, связанные с направлениями $\hat{s}_1, \dots, \hat{s}_n$.

Поиск начинается из $x_0^{(k)}$ путем введения возмущения, равного $\lambda_1^{(k)}\hat{s}_1^{(k)}$, в первом координатном направлении. Если значение $f(x_0^{(k)} + \lambda_1^{(k)}\hat{s}_1^{(k)})$ равно или меньше, чем $f(x_0^{(k)})$, шаг считается успешным и полученная пробная точка заменяет $x_0^{(k)}$, $\lambda_1^{(k)}$ умножается на множитель $\alpha > 0$ и вводится возмущение по направлению $\hat{s}_2^{(k)}$. Если зна-

чение $f(\mathbf{x}_0^{(k)} + \lambda_1^{(k)} \hat{\mathbf{s}}_1^{(k)})$ больше, чем $f(\mathbf{x}_0^{(k)})$, то шаг считается неудачным, $\mathbf{x}_0^{(k)}$ не заменяется, $\lambda_1^{(k)}$ умножается на множитель $\beta < 0$ и снова задается возмущение по направлению $\hat{\mathbf{s}}_2^{(k)}$. Розенброк предложил в общем случае брать значения параметров равными $\alpha = 3$ и $\beta = -0,5$.

После того как пройдены все n направлений $\hat{\mathbf{s}}_1^{(k)}, \dots, \hat{\mathbf{s}}_n^{(k)}$, снова возвращаются к первому направлению $\hat{\mathbf{s}}_1^{(k)}$ и вводится возмущение с длиной шага, равной $\alpha\lambda^{(k)}$ или $\beta\lambda^{(k)}$, в зависимости от результата предыдущего возмущения по направлению $\hat{\mathbf{s}}_1^{(k)}$. Возмущения по выбранным направлениям поиска задаются до тех пор, пока по каждому из направлений за успехом не последует неудача. При этом k -й этап поиска заканчивается. Поскольку получение одинаковых значений функции рассматривается как успех, успех в конце концов достигается по каждому направлению, так как множители при $\lambda_i^{(k)}$ уменьшают последовательно длины шагов. Последняя полученная точка становится начальной точкой следующего этапа, $\mathbf{x}_0^{(k+1)} = \mathbf{x}_n^{(k)}$. Нормированное направление $\hat{\mathbf{s}}_1^{(k+1)}$ берется параллельным $(\mathbf{x}_0^{(k+1)} - \mathbf{x}_0^{(k)})$, а остальные направления выбираются ортонормированными друг к другу и к $\hat{\mathbf{s}}_1^{(k+1)}$ с помощью одного из методов, описанных ниже.

Дэвис, Свенн и Кемпи (ДСК), как это изложено в работе Свенна [8], модифицировали поиск Розенброка в направлениях $\hat{\mathbf{s}}_1^{(k)}, \dots, \hat{\mathbf{s}}_n^{(k)}$ путем отыскания минимума $f(\mathbf{x})$ в каждом из направлений $\hat{\mathbf{s}}_i^{(k)}$ способом, напоминающим поиск Дэвидона — Флетчера — Пауэлла. Объясним это подробнее. Пусть $\mathbf{x}_i^{(k)}$ представляет собой точку, в которой $f(\mathbf{x}_i^{(k)})$ принимает минимальное значение в направлении $\hat{\mathbf{s}}_i^{(k)}$. Для каждого этапа k существует n векторов $\mathbf{x}_i^{(k)}$ и n оптимальных значений целевой функции $f(\mathbf{x}_i^{(k)})$. Начиная с $\mathbf{x}_0^{(k)}$, определяется $\lambda_1^{*(k)}$ в направлении $\hat{\mathbf{s}}_1^{(k)}$ так, чтобы значение $f(\mathbf{x}_0^{(k)} + \lambda_1^{*(k)} \hat{\mathbf{s}}_1^{(k)})$ становилось минимальным. Затем принимается $\mathbf{x}_1^{(k)} = \mathbf{x}_0^{(k)} + \lambda_1^{*(k)} \hat{\mathbf{s}}_1^{(k)}$. Начиная с $\mathbf{x}_1^{(k)}$, определяется $\lambda_2^{*(k)}$ так, чтобы $f(\mathbf{x}_1^{(k)} + \lambda_2^{*(k)} \hat{\mathbf{s}}_2^{(k)})$ принимало минимальное значение. Затем принимается $\mathbf{x}_2^{(k)} = \mathbf{x}_1^{(k)} + \lambda_2^{*(k)} \hat{\mathbf{s}}_2^{(k)}$ и т. д.

Изложенная схема поиска обобщается следующим образом. Начиная с $\mathbf{x}_{i-1}^{(k)}$, определяется $\lambda_i^{*(k)}$ в направлении $\hat{\mathbf{s}}_i^{(k)}$ так, чтобы $f(\mathbf{x}_{i-1}^{(k)} + \lambda_i^{*(k)} \hat{\mathbf{s}}_i^{(k)})$ обращалась в минимум. Затем принимается $\mathbf{x}_i^{(k)} = \mathbf{x}_{i-1}^{(k)} + \lambda_i^{*(k)} \hat{\mathbf{s}}_i^{(k)}$. Этот поиск последовательно повторяется, всегда начиная из предыдущей точки, пока не будут найдены все \mathbf{x}_i , $i = 1, \dots, n$. Дэвис, Свенн и Кемпи определяли $\lambda_i^{*(k)}$ с помощью алгоритма линейного поиска, описанного в разд. 2.6. Однако вместо

него может быть использован любой эффективный метод одномерного поиска.

После завершения k -го этапа либо с помощью оригинального метода Розенброка, либо его модификации с использованием метода ДСК в точке $\mathbf{x}_0^{(k+1)} = \mathbf{x}_n^{(k)}$ вычисляются векторы новых направлений поиска. В сущности здесь ортогональные направления поиска поворачиваются по отношению к предыдущим направлениям так, что они оказываются вытянутыми вдоль оврага (или хребта), и таким образом исключается взаимодействие переменных. Пусть $\Lambda_i^{(k)}$ представляет собой алгебраическую сумму всех успешных шагов (суммарное перемещение) в направлении $s_i^{(k)}$ на k -м этапе; в случае метода ДСК справедливо равенство $\Lambda_i^{(k)} = \lambda_i^{*(k)}$. Определим n векторов $\mathbf{A}_1, \dots, \mathbf{A}_n$ следующим образом:

$$\begin{aligned} \mathbf{A}_1^{(k)} &= \Lambda_1^{(k)} s_1^{(k)} + \Lambda_2^{(k)} s_2^{(k)} + \dots + \Lambda_n^{(k)} s_n^{(k)}, \\ \mathbf{A}_2^{(k)} &= \Lambda_2^{(k)} s_2^{(k)} + \dots + \Lambda_n^{(k)} s_n^{(k)}, \\ &\dots \\ \mathbf{A}_n^{(k)} &= \Lambda_n^{(k)} s_n^{(k)}. \end{aligned} \quad (4.3.1)$$

Таким образом, $\mathbf{A}_1^{(k)}$ является вектором перехода из $\mathbf{x}_0^{(k)}$ в $\mathbf{x}_0^{(k+1)}$, $\mathbf{A}_2^{(k)}$ — вектором перехода из $\mathbf{x}_1^{(k)}$ в $\mathbf{x}_0^{(k+1)}$ и т. д. $\mathbf{A}_1^{(k)}$ представляет собой полное перемещение с k -го этапа на $(k+1)$ -й этап, $\mathbf{A}_2^{(k)}$ — полное перемещение, но без учета продвижения, сделанного во время поиска в направлении $s_1^{(k)}$ и т. д.

В модифицированном методе Розенброка с использованием алгоритма ДСК во избежание обращения в нуль любого из $s_i^{(k)}$ (ибо в этом случае данный алгоритм перестает работать) Λ и s в (4.3.1) нумеруются нижними индексами, так что направления поиска $|\Lambda_1^{(k)}| > |\Lambda_2^{(k)}| > \dots > |\Lambda_n^{(k)}|$. Тогда если любые m из $\Lambda_i^{(k)}$ обращаются в нуль, то отыскивают новые направления, как описано ниже, лишь для тех $(n-m)$ направлений, для которых $\Lambda_i^{(k)} \neq 0$; оставшиеся m направлений остаются неизменными:

$$\mathbf{s}_i^{(k+1)} = \mathbf{s}_i^{(k)}, \quad i = (n-m)+1, \dots, n.$$

Таким образом, в методе ДСК векторам с ненулевыми Λ приписываются первые $(n-m)$ номеров. Так как первые $(n-m)$ векторов взаимно ортогональны и $\Lambda_i = 0$ для $i = n-m+1, \dots, n$, первые $n-m$ векторов не будут иметь составляющих в направлениях $\mathbf{s}_i^{(k+1)}$, $i = n-m+1, \dots, n$. А поскольку эти последние направления взаимно ортогональны, то из этого следует, что все направления являются взаимно ортогональными.

Свенн указывает, что на практике оказалось более удобным изменить критерий для перегруппировки направлений и вместо

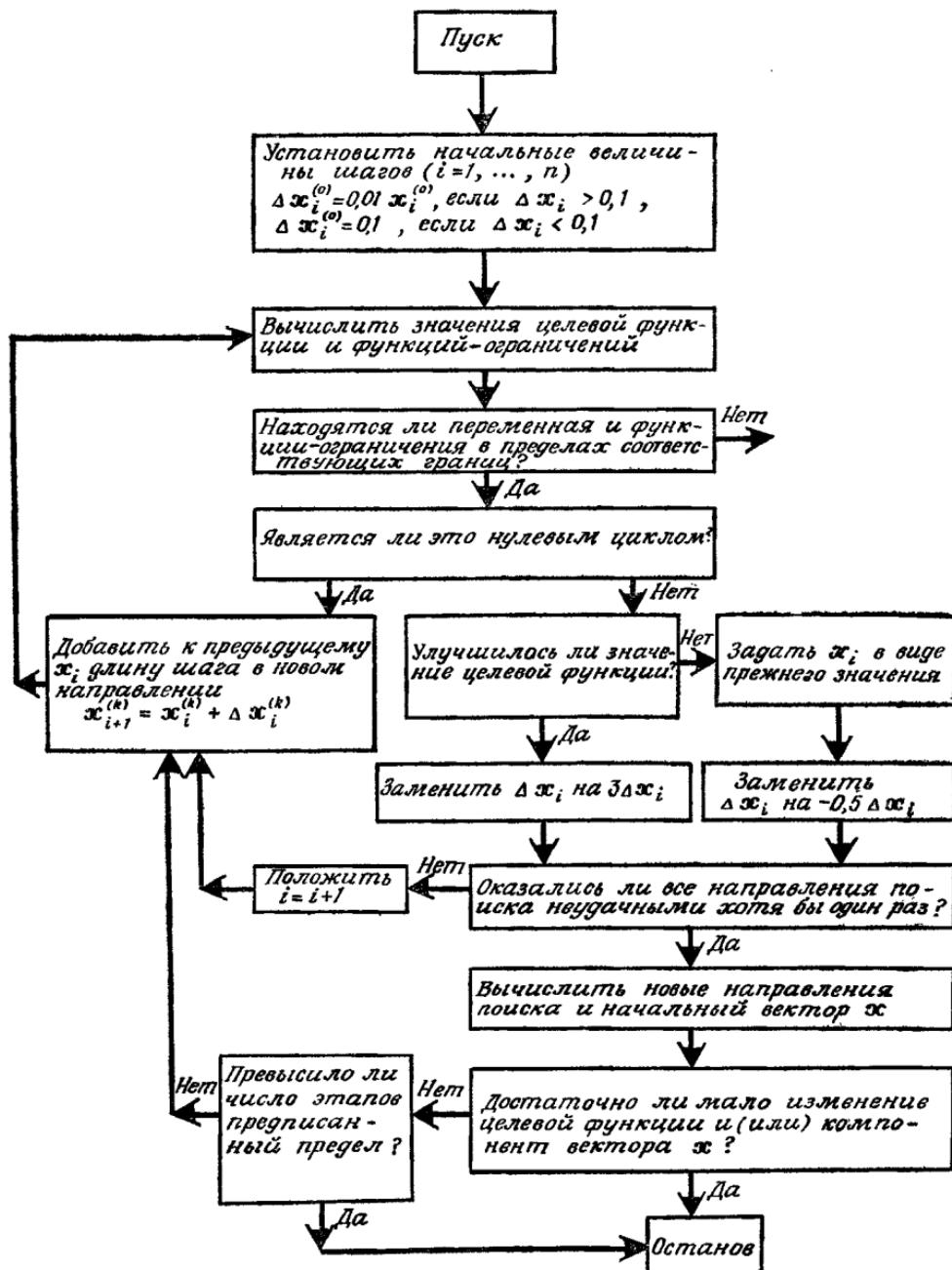
$\Lambda_i^{(k)} = 0$ использовать $|\Lambda_i^{(k)}| < \varepsilon$, где ε — заданная точность x или $f(x)$. Эта модификация слегка повлияла на ортогональность векторов $s_i^{(k)}$, но весьма несущественно. Оказалось, что в случае линейного поиска редукция длины шага с коэффициентом 0,1 уменьшает число вычислений целевой функции, и поэтому она вводилась в программу ДСК каждый раз так, чтобы расстояние между $x_0^{(k)}$ и $x_i^{(k+1)}$ оказывалось меньше, чем длина шага на k -м этапе.

В обоих методах новые направления определяются следующим образом. Первый единичный вектор $\hat{s}_1^{(k+1)}$ из нового набора направлений на $(k+1)$ -м этапе направляют так, чтобы он совпадал с направлением результирующего перемещения на предыдущем этапе $A_1^{(k)}$. Остальные направления поиска строятся как взаимно ортогональные к $A_1^{(k)}$ единичные векторы (ортонормированные векторы) с помощью метода Грама — Шмидта. Подробное описание этого построения можно найти в работах по теории матриц и линейной алгебре. Таким образом, набор ортонормированных векторов $\hat{s}_1^{(k+1)}, \dots, \hat{s}_n^{(k+1)}$ на $(k+1)$ -м этапе вычисляется при помощи следующих соотношений:

$$\begin{aligned}\hat{s}_1^{(k+1)} &= \frac{A_1^{(k)}}{\|A_1^{(k)}\|}, \\ B_2^{(k)} &= A_2^{(k)} - [(A_2^{(k)})^T \quad \hat{s}_1^{(k+1)}] \hat{s}_1^{(k+1)}, \\ \hat{s}_2^{(k+1)} &= \frac{B_2^{(k)}}{\|B_2^{(k)}\|}, \\ &\dots \dots \dots \dots \dots \dots \dots \\ B_n^{(k)} &= A_n^{(k)} - \sum_{i=1}^{n-1} [(A_n^{(k)})^T \quad \hat{s}_i^{(k+1)}] \hat{s}_i^{(k+1)}, \\ \hat{s}_n^{(k+1)} &= \frac{B_n^{(k)}}{\|B_n^{(k)}\|},\end{aligned}\tag{4.3.2}$$

где $\|A_i\|$ — норма A_i . Та же процедура поиска, которая проводилась на k -м этапе, повторяется затем на $(k+1)$ -м этапе, начиная с точки $x_0^{(k+1)} = x_n^{(k)}$.

Палмер [9] показал, что $B_{j+1}^{(k)}$ и $\|B_{j+1}^{(k)}\|$ пропорциональны $\Lambda_j^{(k)}$ (при условии, что $\sum_{i=j}^n (\Lambda_i^{(k)})^2 \neq 0$). Следовательно, при вычислении $\hat{s}_j^{(k+1)} = B_j^{(k)} / \|B_j^{(k)}\|$ величина $\Lambda_j^{(k)}$ сокращается, и, таким образом, $\hat{s}_j^{(k+1)}$ остается определенным, если даже $\Lambda_j^{(k)} = 0$. Имея это в виду,



Ф и г. 4.3.1. Информационная блок-схема алгоритма Розенброка (логическая часть, служащая для реализации ограничений, здесь опущена).

Палмер предложил для вычисления $\hat{s}_i^{(k+1)}$ пользоваться приводимыми ниже формулами, что значительно уменьшит необходимое количество арифметических операций и объем памяти ЭВМ [вывод формул (4.3.3) приведен в упомянутой статье Палмера]:

$$\begin{aligned} A_i^{(k)} &= \sum_{j=i}^n \Lambda_j^{(k)} \hat{s}_j^{(k)}, \quad 1 \leq i \leq n, \\ \hat{s}_i^{(k+1)} &= \frac{A_i^{(k)} \| A_{i-1}^{(k)} \|^2 - A_{i-1}^{(k)} \| A_i^{(k)} \|^2}{\| A_{i-1}^{(k)} \| \| A_i^{(k)} \| (\| A_{i-1}^{(k)} \|^2 - \| A_i^{(k)} \|^2)^{1/2}}, \quad 2 \leq i \leq n, \quad (4.3.3) \\ \hat{s}_1^{(k+1)} &= \frac{A_1^{(k)}}{\| A_1^{(k)} \|}. \end{aligned}$$

При этом не требуется перегруппировка элементов $\Lambda_i^{(k)}$. Если $\Lambda_{i-1}^{(k)} = 0$, то $\hat{s}_i^{(k+1)} = \hat{s}_{i-1}^{(k)}$, кроме случая, когда $\Sigma (\Lambda_i^{(k)})^2 = 0$.

Метод Розенброка не обеспечивает автоматическое окончание поиска после того, как найден экстремум $f(x)$. Поиск либо проводится на определенном числе этапов, либо заканчивается, как только величина A_1 становится меньше определенного значения на нескольких последовательных этапах. В случае модифицированного метода Розенброка (с использованием алгоритма ДСК) после каждого этапа расстояние $\Lambda_i^{(k)}$ сравнивается с размером шага $\delta_i^{(k)}$, использованного для получения $\Lambda_i^{(k)}$ в линейном поиске. Если $\Lambda_i^{(k)} < \delta_i^{(k)}$, то $\delta_i^{(k)}$ делят на 10, и дальнейший поиск осуществляется в k прежних направлениях с новым δ_i . Если $\Lambda_i^{(k)} > \delta_i^{(k)}$, то поиск продолжается на $(k+1)$ -м этапе, как было описано выше.

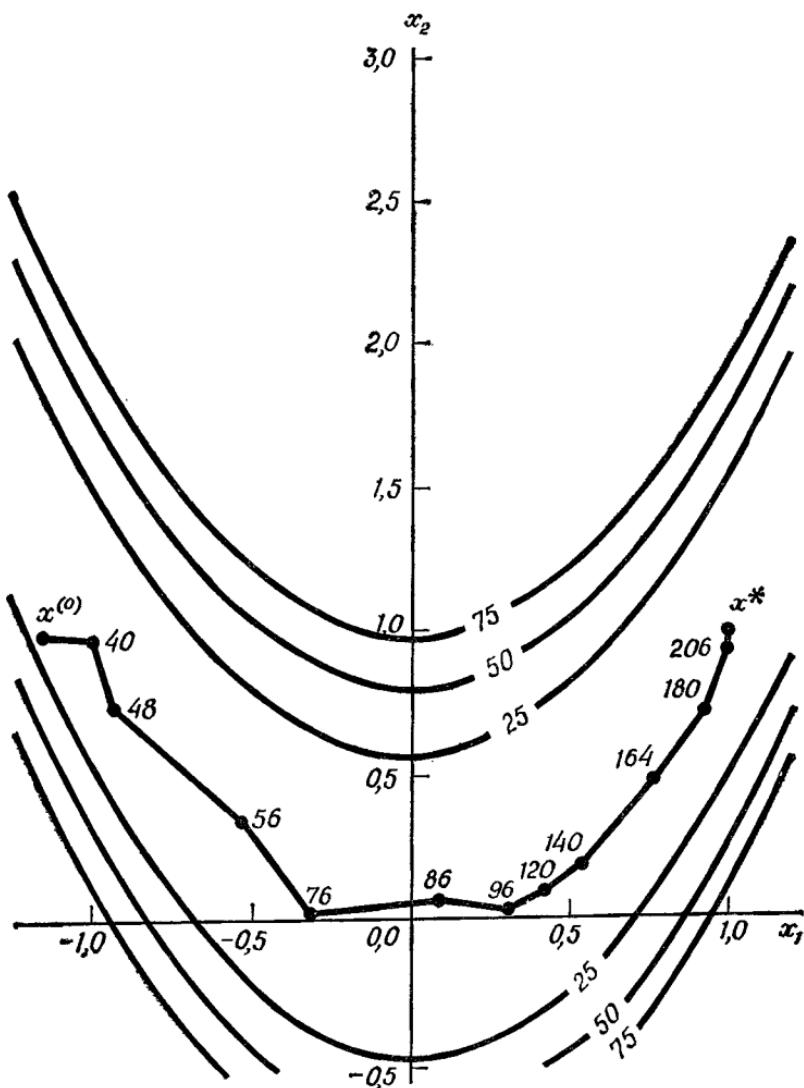
На фиг. 4.3.1 представлена информационная блок-схема алгоритма Розенброка, а на фиг. 4.3.2 приведена траектория поиска при минимизации функции Розенброка с помощью этого алгоритма.

Пример 4.3.1. Метод Розенброка

Проиллюстрируем алгоритм Розенброка в приложении к задаче примера 3.4.1.

Минимизировать $f(x) = 4(x_1 - 5)^2 + (x_2 - 6)^2$. Известно, что эта функция имеет минимум в точке $x^* = [5 \ 6]^T$, где $f(x^*) = 0$. (В целях экономии места будем округлять числа.) Начнем из точки $x^{(0)} = [8 \ 9]^T$, где $f(x^{(0)}) = 45,000$, $\lambda_1 = 0,10$. При этом направления начального поиска совпадают с координатными осями x_1 и x_2 :

$$\hat{s}_1^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \hat{s}_2^{(0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$



Ф и г. 4.3.2. Траектория поиска при минимизации функции Розенброка методом Розенброка (числа обозначают количество вычислений целевой функции).

Сначала вычислим $f(\mathbf{x})$ в точке

$$\mathbf{x} = \begin{bmatrix} 8,00 + 0,10 \cdot 1 \\ 9,00 + 0,10 \cdot 0 \end{bmatrix} = \begin{bmatrix} 8,10 \\ 9,00 \end{bmatrix}.$$

Здесь $f(\mathbf{x}) = 47,44$, т. е. имеет место неудача. Затем вычислим $f(\mathbf{x})$ в точке

$$\mathbf{x} = \begin{bmatrix} 8,00 + 0,10 \cdot 0 \\ 9,00 + 0,10 \cdot 1 \end{bmatrix} = \begin{bmatrix} 8,00 \\ 9,10 \end{bmatrix}.$$

Здесь $f(x) = 45,61$ и снова имеет место неудача. Таким образом, на следующем цикле λ_1 , и λ_2 должны быть умножены на $\beta = -0,5$, или $\lambda_3 = \lambda_4 = -0,50 \cdot 0,10 = -0,05$. Возмущение проводится из последнего успешного значения x , т. е. из точки $x = [8,00 \ 9,00]^T$.

Сначала вычислим $f(x)$ в точке

$$x = \begin{bmatrix} 8,00 - 0,05 \cdot 1 \\ 9,00 - 0,05 \cdot 0 \end{bmatrix} = \begin{bmatrix} 7,95 \\ 9,00 \end{bmatrix}.$$

Здесь $f(x) = 43,81$, что означает успех. Затем вычислим $f(x)$ в точке

$$x = \begin{bmatrix} 7,95 - 0,05 \cdot 0 \\ 9,00 - 0,05 \cdot 1 \end{bmatrix} = \begin{bmatrix} 7,95 \\ 8,95 \end{bmatrix},$$

где $f(x) = 43,125$, что снова означает успех. На следующем цикле λ умножается на 3, т. е. $\lambda_5 = \lambda_6 = 3 \cdot (-0,05) = -0,15$.

Ниже приведены несколько последовательных циклов первого этапа.

Номер поиска	λ	x_1	x_2	$f(x)$	Успех (S) или неудача (F)
5	-0,15	7,80	8,95	40,06	S
6	-0,15	7,80	8,80	39,20	S
7	-0,45	7,35	8,80	29,93	S
8	-0,45	7,35	8,35	27,61	S
9	-1,35	6,00	8,35	9,522	S
10	-1,35	6,00	7,00	5,000	S
11	-4,05	1,95	7,00	32,21	F
12	-4,05	6,00	2,95	13,30	F

Теперь уже в каждом координатном направлении за успехом последовала неудача, и, таким образом, закончился нулевой этап поиска. Затем вычисляются новые направления поиска так, чтобы $\hat{s}_1^{(1)}$ было направлено вдоль вектора, идущего из $x_0^{(0)} = [8 \ 9]^T$ в $x_0^{(1)} = [6 \ 7]^T$, причем последняя точка соответствует наилучшему значению $f(x)$, полученному на нулевом этапе; $\hat{s}_2^{(1)}$ ортого нормировано к $\hat{s}_1^{(1)}$ (фиг. П. 4.3.1а). Векторы $A_1^{(0)}$ и $A_2^{(0)}$ вычисляются по формуле (4.3.1), а $\hat{s}_1^{(1)}$ и $\hat{s}_2^{(1)}$ — по формуле (4.3.2).

$$A_1^{(0)} = \begin{bmatrix} -2 \\ -2 \end{bmatrix}, \quad A_2^{(0)} = \begin{bmatrix} 0 \\ -2 \end{bmatrix},$$

$$\hat{s}_1^{(1)} = \frac{[-2 -2]^T}{[(-2)^2 + (-2)^2]^{1/2}} = \left[-\frac{1}{\sqrt{2}} - \frac{1}{\sqrt{2}} \right]^T,$$

$$\mathbf{B}_2^{(0)} = \begin{bmatrix} 0 \\ -2 \end{bmatrix} - [0 \ -2] \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} -\frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix},$$

$$\hat{s}_2^{(0)} = \frac{[1 -1]^T}{[(1)^2 + (-1)^2]^{1/2}} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix}.$$

В первом цикле на первом этапе поиска ($k = 1$) берем $\lambda_1 = \lambda_2 = -4,05 \cdot (-0,5) = 2,025$. Сначала вычислим $f(x)$ в направлении поиска 1 в точке

$$x = \begin{bmatrix} 6,000 + (2,025) \cdot (-0,706) \\ 7,000 + (2,025) \cdot (-0,706) \end{bmatrix} = \begin{bmatrix} 4,568 \\ 5,568 \end{bmatrix}.$$

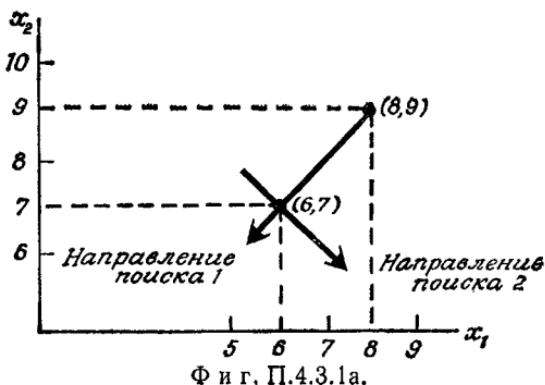
Здесь $f(x) = 0,9327$, что означает успех. Затем вычислим $f(x)$ в направлении поиска 2 в точке

$$x = \begin{bmatrix} 4,568 + (2,025) \cdot (0,706) \\ 5,568 + (2,025) \cdot (-0,706) \end{bmatrix} = \begin{bmatrix} 6,000 \\ 4,136 \end{bmatrix}.$$

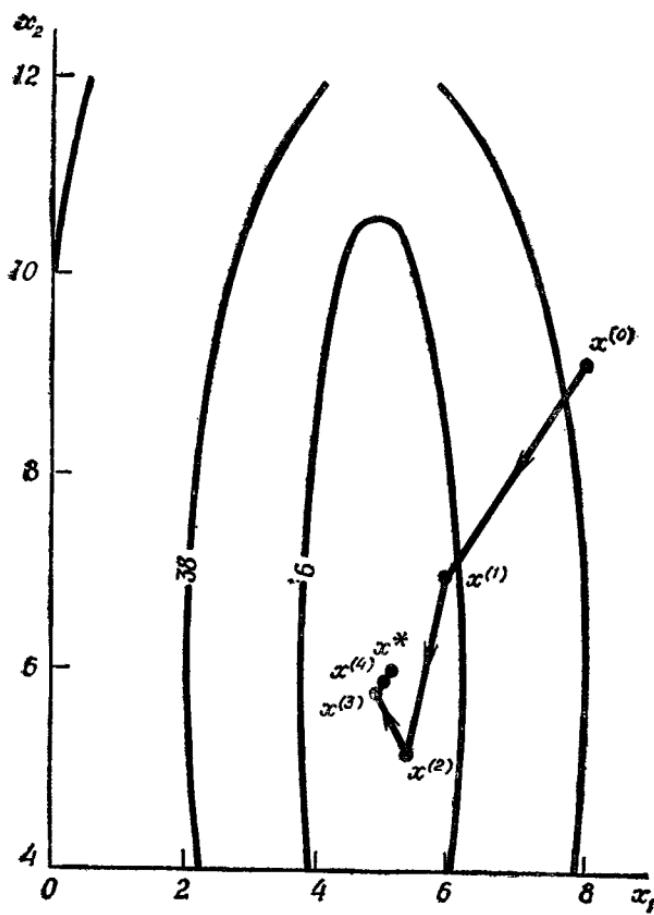
Здесь $f(x) = 7,474$, что означает неудачу. Остальные данные для первого этапа приведены ниже:

Номер поиска	λ	x_1	x_2	$f(x)$	Успех (S) или неудача (F)
3	6,075	0,272	1,272	11,175	F
4	-1,125	3,852	6,284	5,351	F
5	-3,038	6,716	7,716	14,722	F
6	0,506	4,926	5,210	0,646	S
7	1,518	3,852	4,136	8,743	F
8	1,518	6,000	4,136	7,473	F

Поскольку движение в направлении 1 приводит к успеху в первых двух поисках, а в направлении 2 — к неудаче, то в поисках 3 и 4 берем $\lambda_3 = 3 \cdot 2,025 = 6,075$ и $\lambda_4 = -0,5 \cdot 2,025 = -1,0125$ соответственно. Заметим, что после неудачи в поисках 3 и 4 $\lambda_3 = 6,075$ и $\lambda_4 = -1,0125$ умножаются на $-0,5$, что дает соответственно λ_5 и λ_6 . После поиска 6 имел место успех по каждому направлению, а после поиска 8 произошли две последовательные неудачи, так что в соответствии с данным алгоритмом на этом



Ф и г. П.4.3.1а.



Ф и г. П.4.3.1б. Траектория поиска для алгоритма Розенброка.

этап 1 закончился. Теперь должно быть определено новое направление поиска. На этом мы закончим рассмотрение данного примера. Однако приведенная выше процедура была продолжена пока не был удовлетворен критерий сходимости [это потребовало 111 вычислений значений целевой функции, и относительное отклонение $f(\mathbf{x}^{(k)})$ от правильного значения, равного 0, составило $5,5 \cdot 10^{-11}$, относительное отклонение x_1 равнялось $3,0 \cdot 10^{-6}$, а отклонение x_2 составило $4,4 \cdot 10^{-6}$].

На фиг. П.4.3.1б показана траектория основной части минимизации на первых четырех этапах (34 вычисления функции); после проведения этих этапов были получены $\mathbf{x}^{(4)} = [5,036 \ 5,938]^T$ и $f(\mathbf{x}^{(4)}) = 9,46 \cdot 10^{-3}$. Остальные 77 вычислений функции понадобились для увеличения точности \mathbf{x} и $f(\mathbf{x})$.

4.4. МЕТОД ПАУЭЛЛА

В методе Пауэлла [10], который является развитием алгоритма Смита [11], определяется местонахождение минимума некоторой квадратичной функции $f(\mathbf{x})$ при $H > 0$ путем проведения последовательных одномерных поисков, начиная с точки $\mathbf{x}_0^{(k)}$, вдоль системы полученных *сопряженных направлений*. Напомним (см. подразд. 3.3.1), что два направления поиска s_i и s_j называются сопряженными, если

$$\begin{aligned} (\mathbf{s}_j)^T \mathbf{Q} \mathbf{s}_i &= 0, & i \neq j, \\ (\mathbf{s}_j)^T \mathbf{Q} \mathbf{s}_i &\geq 0, & i = j, \end{aligned}$$

где $\mathbf{Q} = \nabla^2 f(\mathbf{x}^{(k)})$ — положительно определенная квадратная матрица.

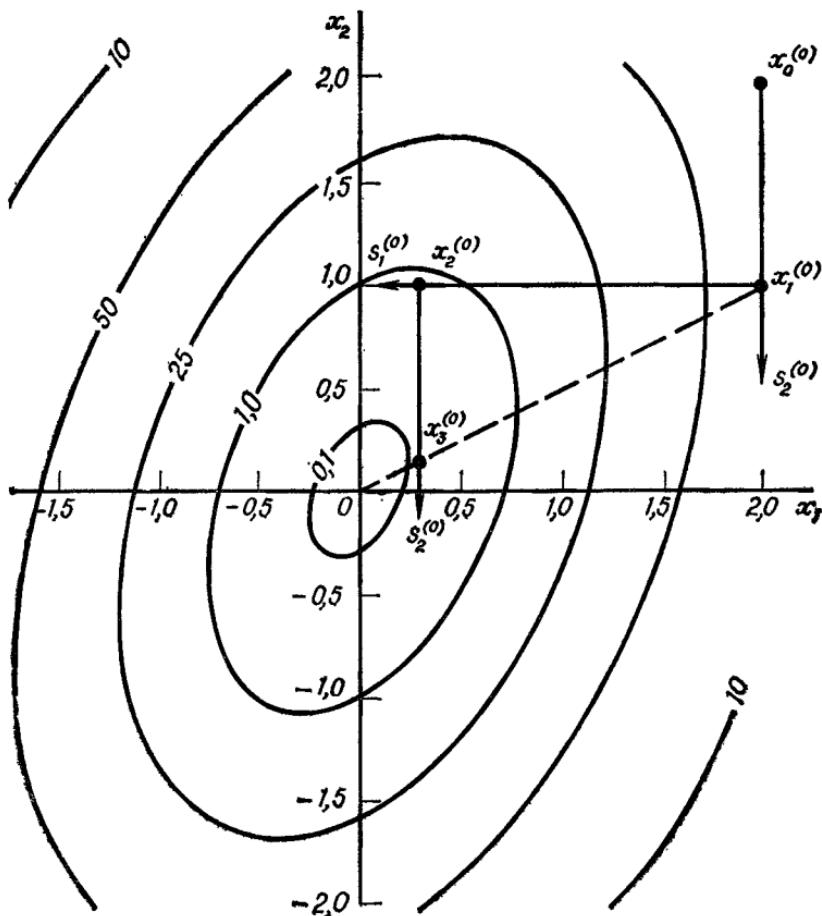
Нижние индексы обозначают векторы одного этапа (последние обозначаются верхним индексом). Идея алгоритма Пауэлла в сущности заключается в том, что если на данном этапе поиска определяется минимум квадратичной функции $f(\mathbf{x})$ вдоль каждого из p ($p < n$) сопряженных направлений и если затем в каждом направлении делается некоторый шаг, то полное перемещение от начала до p -го шага сопряжено ко всем поднаправлениям поиска. Таким образом, этот метод аналогичен партан-методу, описанному в подразд. 3.3.3.

Переход из точки $\mathbf{x}_0^{(k)}$ в точку $\mathbf{x}_m^{(k)}$ определяется формулой

$$\mathbf{x}_m^{(k)} = \mathbf{x}_0^{(k)} + \sum_{i=0}^{m-1} \lambda_i^{(k)} \mathbf{s}_i^{(k)}, \quad i = 1, \dots, m-1. \quad (4.4.1)$$

Вычислительная процедура данного алгоритма проводится следующим образом. В точке $\mathbf{x}_0^{(0)}$ в E^n начальные направления $\mathbf{s}_1^{(0)}, \dots, \mathbf{s}_n^{(0)}$

берутся параллельными координатным осям E^n . Первый шаг делается в направлении $s_n^{(0)}$, т. е. минимизируется $f(x_0^{(0)} + \lambda s_n^{(0)})$ по λ (с помощью одномерного поиска) для вычисления $\lambda_0^{(0)}$. Затем полагается



Фиг. 4.4.1. Метод поиска Паузеля.

$x_1^{(0)} = x_0^{(0)} + \lambda_0^{(0)} s_n^{(0)}$. Как видно на фиг. 4.4.1, в точке $x_1^{(0)}$ имеет место первый обнаруженный минимум. Затем вдоль каждого из n направлений $s_i^{(0)}$, $i = 1, \dots, n$, в свою очередь минимизируется $f(x_1^{(0)} + \lambda s_i^{(0)})$, определяется соответствующее $\lambda_i^{(0)}$ и последовательно вычисляются по формуле (4.4.1) новые значения $x_i^{(0)}$. На фиг. 4.4.1 показано расположение точек $x_0^{(0)}$, $x_1^{(0)}$, $x_2^{(0)}$ и $x_3^{(0)}$ для следующей целевой функции, представляющей собой функцию двух переменных:

$$f(\mathbf{x}) = 2x_1^2 + x_2^2 - x_1x_2.$$

Координатные оси в E^n имеют направления

$$\mathbf{s}_1^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{s}_2^{(0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Они ортогональны, т. е. $(\mathbf{s}_1^{(0)})^T (\mathbf{s}_2^{(0)}) = 0$, но не сопряжены, поскольку $(\mathbf{s}_1^{(0)})^T H \mathbf{s}_2^{(0)} = 4 \neq 0$. Расположение соответствующих минимумов $f(\mathbf{x})$ показано в следующей таблице:

Итерация	$(\mathbf{s}_i^{(0)})^T$	$\lambda_i^{(0)}$	$(\mathbf{x}_{i+1}^{(0)})^T$	$f(\mathbf{x}_{i+1}^{(0)})$ в точке минимума
0	[0 1]	-1	[2 1]	7
1	[1 0]	-1,75	[0,25 1]	0,875
2	[0 1]	-0,875	[0,25 0,125]	0,109

Чтобы понять, какую роль в алгоритме Пауэлла играют сопряженные направления, рассмотрим следующую теорему, имеющую место для квадратичных целевых функций:

Теорема

Если при начальной точке $\mathbf{x}^{(0)}$ поиска в направлении s минимум $f(\mathbf{x})$ находится в некоторой точке $\mathbf{x}^{(a)}$ и если при начальной точке $\mathbf{x}^{(1)} \neq \mathbf{x}^{(0)}$ поиска в том же самом направлении s минимум $f(\mathbf{x})$ находится в точке $\mathbf{x}^{(b)}$; то при $f(\mathbf{x}^{(b)}) < f(\mathbf{x}^{(a)})$ направление $(\mathbf{x}^{(b)} - \mathbf{x}^{(a)})$ сопряжено с s .

Приведем доказательство этой теоремы. Из формулы (3.3.7) следует, что для первого поиска

$$s^T \nabla f(\mathbf{x}^{(a)}) = s^T (H\mathbf{x}^{(a)} + b) = 0$$

и для второго

$$s^T \nabla f(\mathbf{x}^{(b)}) = s^T (H\mathbf{x}^{(b)} + b) = 0.$$

Вычитая, получаем

$$s^T H(\mathbf{x}^{(b)} - \mathbf{x}^{(a)}) = 0.$$

Следовательно, s и $(\mathbf{x}^{(b)} - \mathbf{x}^{(a)})$ сопряжены. Из фиг. 4.4.1 видно, что

$$(\mathbf{x}_3^{(0)} - \mathbf{x}_2^{(0)})^T H(\mathbf{x}_3^{(0)} - \mathbf{x}_1^{(0)}) = 0,$$

где $\mathbf{x}_0^{(0)}$ соответствует точке $\mathbf{x}^{(0)}$ в формулировке теоремы, а любая точка на прямой $(\mathbf{x}_3^{(0)} - \mathbf{x}_2^{(0)})$ соответствует точке $\mathbf{x}^{(1)}$. Для примера фиг. 4.4.1

$$[(0,25 - 0,25)(0,125 - 1,000)] \begin{bmatrix} 4 & -1 \\ -1 & 2 \end{bmatrix} \begin{bmatrix} (0,25 - 2,00) \\ (0,125 - 1,000) \end{bmatrix} = 0.$$

Эта теорема непосредственно может быть распространена на случай нескольких сопряженных направлений; если, начиная из $\mathbf{x}^{(0)}$, точка $\mathbf{x}^{(a)}$ определяется после использования p сопряженных направлений ($p < n$) и аналогично если, начиная из $\mathbf{x}^{(1)}$, точка $\mathbf{x}^{(b)}$ определяется после использования тех же направлений и $f(\mathbf{x})$ минимизируется на каждом шаге, то вектор $(\mathbf{x}^{(b)} - \mathbf{x}^{(a)})$ сопряжен ко всем p направлениям.

Таким образом, мы определили два сопряженных направления, в которых следует вести поиск. Следовательно, направление $s_1^{(0)}$ должно быть заменено на направление вектора $(\mathbf{x}_3^{(0)} - \mathbf{x}_1^{(0)})$, представляющего собой полное перемещение из первого минимума. Направления поиска на следующем этапе таковы:

$$s_1^{(1)} = s_2^{(0)},$$

$$s_2^{(1)} = (\mathbf{x}_3^{(0)} - \mathbf{x}_1^{(0)}).$$

Второй этап поиска снова начинается минимизацией, на этот раз вдоль направления $s_2^{(1)}$. Затем осуществляются, если это оказывается необходимым, перемещения в направлениях $s_1^{(1)}$ и $s_2^{(1)}$. Вспомним, однако, что для квадратичной целевой функции двух переменных после использования двух сопряженных направлений сразу достигается минимум (фиг. 4.4.1).

Алгоритм Пауэлла несколько отличается от простых начальных шагов, описанных выше.

В общем случае на k -м этапе метода Пауэлла используются n линейно независимых направлений поиска; при этом поиск начинается в некоторой точке $\mathbf{x}_0^{(k)} = \mathbf{x}_{n+1}^{(k-1)}$ и проводится следующим образом:

Шаг 1. Начиная из $\mathbf{x}_0^{(k)}$, с помощью какого-либо одномерного поиска определяется $\lambda_1^{(k)}$ так, чтобы $f(\mathbf{x}_0^{(k)} + \lambda_1 s_1^{(k)})$ принимала минимальное значение, и полагается $\mathbf{x}_1^{(k)} = \mathbf{x}_0^{(k)} + \lambda_1 s_1^{(k)}$. Начиная из $\mathbf{x}_1^{(k)}$, определяется $\lambda_2^{(k)}$ так, чтобы $f(\mathbf{x}_1^{(k)} + \lambda_2 s_2^{(k)})$ обращалась в минимум, и полагается $\mathbf{x}_2^{(k)} = \mathbf{x}_1^{(k)} + \lambda_2 s_2^{(k)}$. Поиск продолжается последовательно в каждом направлении, всегда начиная из самой последней точки последовательности, пока не будут определены все $\lambda_i^{(k)}$, $i = 1, \dots, n$. Величина $\lambda_0^{(k)}$, полученная при минимизации $f(\mathbf{x})$ в направлении $s_n^{(k-1)}$, используется на шаге 4 (см. ниже).

Шаг 2. Пауэлл отметил, что поиск, осуществляемый в соответствии с шагом 1, может привести к линейно зависимым направлениям поиска (см. разд. 4.3), как, например, в случае, когда одна из компонент $s^{(k)}$ становится тождественным иулем, поскольку в этом направлении не может быть движения. Следовательно, два направления могут стать коллинеарными. Таким образом, не

следует заменять старое направление на новое, если после этого направления нового набора становятся линейно зависимыми. Пауэлл показал также (на примере квадратичной функции), что при нормировании направлений поиска в соответствии с формулой

$$(\mathbf{s}_i^{(k)})^T \mathbf{H} \mathbf{s}_i^{(k)} = 1, \quad i = 1, \dots, n,$$

определитель матрицы, столбцы которой представляют собой направления поиска, принимает максимальное значение тогда и только тогда, когда $\mathbf{s}_i^{(k)}$ взаимно сопряжены относительно \mathbf{H} . Он пришел к выводу, что направление полного перемещения на k -м этапе $\mathbf{s}^{(k)}$ должно заменять предыдущее направление только в том случае, когда заменяющий вектор увеличивает определитель матрицы направлений поиска, поскольку только тогда новый набор направлений будет более эффективным. Следовательно, после минимизации $f(\mathbf{x})$ в каждом из n направлений, как на шаге 1, проводится один дополнительный шаг величиной $(\mathbf{x}_n^{(k)} - \mathbf{x}_0^{(k)})$, соответствующий полному перемещению на k -м этапе и приводящий в точку $(2\mathbf{x}_n^{(k)} - \mathbf{x}_0^{(k)})$. Затем проводится тест (см. шаг 3), чтобы убедиться, уменьшается ли определитель матрицы направлений поиска путем включения нового направления и отбрасывания старого.

Шаг 3. Обозначим наибольшее уменьшение $f(\mathbf{x})$ в каком-либо направлении поиска на k -м этапе через

$$\Delta^{(k)} = \max_{i=1, \dots, n} \{f(\mathbf{x}_{i-1}^{(k)}) - f(\mathbf{x}_i^{(k)})\}.$$

Направление поиска, соответствующее этому максимальному изменению $f(\mathbf{x})$, обозначим через $\mathbf{s}_m^{(k)}$. Чтобы сделать обозначения более компактными, положим $f_1 = f(\mathbf{x}_0^{(k)})$, $f_2 = f(\mathbf{x}_n^{(k)})$ и $f_3 = f(2\mathbf{x}_n^{(k)} - \mathbf{x}_0^{(k)})$, где $\mathbf{x}_0^{(k)} = \mathbf{x}_n^{(k-1)}$ и $\mathbf{x}_n^{(k)} = \mathbf{x}_{n-1}^{(k)} + \lambda_n^{(k)} \mathbf{s}_n^{(k)} = \mathbf{x}_0^{(k)} + \sum_{i=1}^n \lambda_i^{(k)} \mathbf{s}_i^{(k)}$. Тогда если

$f_3 \geq f_1$ и (или) $(f_1 - 2f_2 + f_3)(f_1 - f_2 - \Delta^{(k)})^2 \geq 0,5\Delta^{(k)}(f_1 - f_3)^2$, то следует использовать на $(k+1)$ -м этапе те же направления $\mathbf{s}_1^{(k)}, \dots, \mathbf{s}_n^{(k)}$, что и на k -м этапе, т. е. $\mathbf{s}_i^{(k+1)} = \mathbf{s}_i^{(k)}$ для $i = 1, \dots, n$, и начать поиск из точки $\mathbf{x}_0^{(k+1)} = \mathbf{x}_n^{(k)}$ [или из $\mathbf{x}_0^{(k+1)} = 2\mathbf{x}_n^{(k)} - \mathbf{x}_0^{(k)} = \mathbf{x}_{n+1}^{(k)}$ в зависимости от того, в какой точке \mathbf{x} функция $f(\mathbf{x})$ принимает наименьшее значение].

Шаг 4. Если тест на шаге 3 не удовлетворен, то ищется минимум $f(\mathbf{x})$ в направлении вектора $\mathbf{s}_m^{(k)}$, проведенного из $\mathbf{x}_0^{(k)}$ в $\mathbf{x}_n^{(k)}$; точка этого минимума берется в качестве начальной для следующего $(k+1)$ -го этапа. Система направлений, используемых на $(k+1)$ -м этапе, та же, что и на k -м этапе, за исключением направления $\mathbf{s}_m^{(k)}$, которое заменяется на $\mathbf{s}^{(k)}$. Однако $\mathbf{s}^{(k)}$ помещают в последний столбец матрицы направлений, а не на место $\mathbf{s}_m^{(k)}$. Следовательно, на

($k + 1$)-м этапе будут использоваться следующие направления:

$$[\mathbf{s}_1^{(k+1)} \mathbf{s}_2^{(k+1)} \dots \mathbf{s}_n^{(k+1)}] = [\mathbf{s}_1^{(k)} \mathbf{s}_2^{(k)} \dots \mathbf{s}_{m-1}^{(k)} \mathbf{s}_{m+1}^{(k)} \dots \mathbf{s}_n^{(k)} \mathbf{s}_0^{(k)}].$$

Шаг 5. Критерий удовлетворительной сходимости для метода Пауэлла, используемый для определения момента окончания поиска в конце любого этапа, состоит в том, что изменение по каждой независимой переменной должно быть меньше, чем заданная точность ε_i , $i = 1, \dots, n$, или $\|x_n^{(k)} - x_0^{(k)}\| \leq 0,1 \varepsilon$.

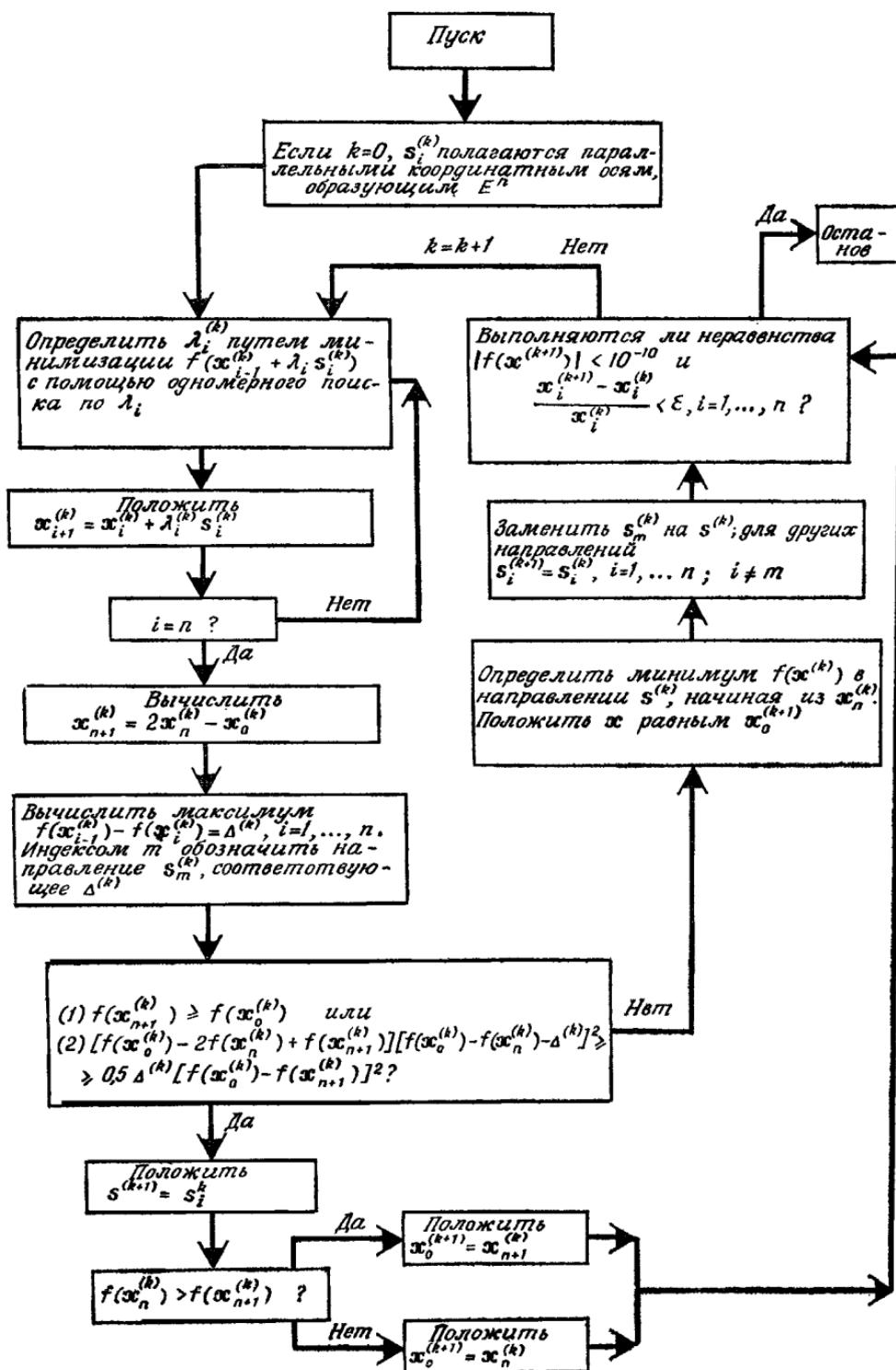
Зангвилл [12] показал, что процедура Пауэлла (немного модифицированная) будет сходиться к точке, в которой градиент $f(\mathbf{x})$ равен нулю, если $f(\mathbf{x})$ — строго выпуклая функция. Такая точка представляет собой локальный экстремум.

На фиг. 4.4.2 приведена информационная блок-схема алгоритма Пауэлла, а на фиг. П.4.4.2 показана траектория поиска при минимизации функции Розенброка.

Пример 4.4.1. Метод Пауэлла

Продемонстрируем метод Пауэлла на примере следующей задачи: минимизировать $f(\mathbf{x}) = 4(x_1 - 5)^2 + (x_2 - 6)^2$, начиная из точки $\mathbf{x}^{(0)} = [8 \ 9]^T$. В точке $\mathbf{x}^{(0)}$ функция $f(\mathbf{x}^{(0)})$ имеет значение 45,000. Сначала минимизируется $f(\mathbf{x}^{(0)}, \lambda)$ по λ с помощью одномерного поиска в координатном направлении x_1 (x_2 остается равным 9,000):

x_1	$f(\mathbf{x})$
8,000	45,000
7,992	44,808
7,952	43,857
7,752	39,294
6,752	21,278
1,752	51,198
5,460	9,847
4,043	12,657
6,335	16,135
4,919	9,026
.	.
.	.
5,000	9,000 (минимум $f(\mathbf{x}^{(0)})$)



Затем одномерный поиск осуществляется в направлении x_2 , начиная из точки с координатами $x_1^{(1)} = 5,000$ (эта координата остается далее постоянной) и $x_2^{(1)} = 9,000$:

x_2	$f(x)$
9,000	9,000
8,991	8,946
8,948	8,678
8,721	7,403
7,596	2,547
1,971	16,232
6,142	0,024
4,549	2,104
7,127	1,271
5,534	0,217
.	.
.	.
.	.
6,000	$5,44 \cdot 10^{-15}$

Осуществление шага $\mathbf{x}_{n+1}^{(0)} = 2\mathbf{x}_n^{(0)} - \mathbf{x}_0^{(0)}$ приводит к

$$2 \begin{bmatrix} 5,000 \\ 6,000 \end{bmatrix} - \begin{bmatrix} 8,000 \\ 9,000 \end{bmatrix} = \begin{bmatrix} 2,000 \\ 3,000 \end{bmatrix}.$$

В этой точке $f(\mathbf{x}_{n+1}^{(0)}) = 45,000$. При этом значение $\Delta^{(0)}$ равно

$$\Delta^{(0)} = \max \{ [f(8,9) - f(5,9)], [f(5,9) - f(5,6)] \} = 36.$$

Заметим также, что

$$f(2,3) = 45,000 \geq f(8,9) = 45,000$$

и

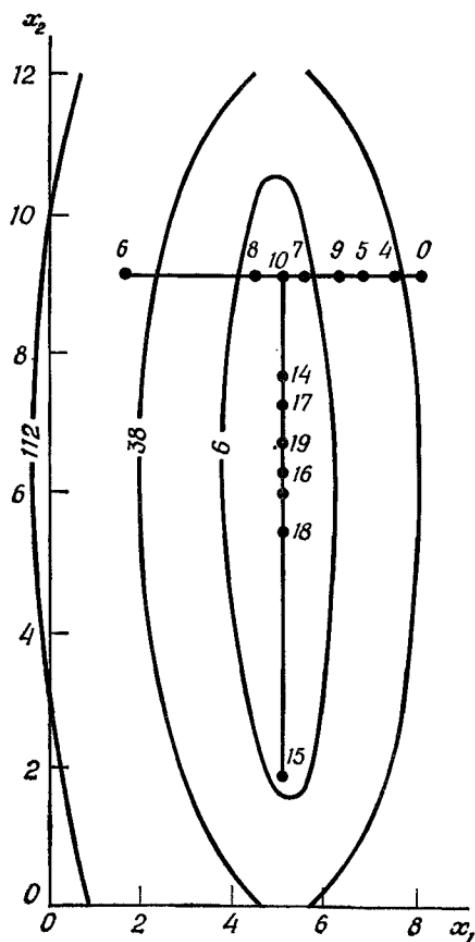
$$[f(8,9) - 2f(5,6) + f(2,3)][f(8,9) - f(5,6) - 36]^2 \geq 0,5 \cdot 36[f(8,9) - f(2,3)]^2 \geq 729 \geq 0.$$

Следовательно, на следующем этапе вектор поиска будет тот же, что и на предыдущем. Тем не менее поиск заканчивается в точке $\mathbf{x}^* = [5 \ 6]^T$, поскольку критерий сходимости удовлетворен. На фиг. П.4.4.1 приведена траектория поиска.

Пример 4.4.2. Метод Пауэлла для функции Розенброка

Применение алгоритма Пауэлла к функции Розенброка

$$f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$



Фиг. П.4.4.1. Траектория поиска в случае использования алгоритма Пауэлла (числа обозначают количество вычислений целевой функции).

показывает, что происходит при минимизации неквадратичной функции. Начнем с точки $\mathbf{x}^{(0)} = [-1,2 \ 1,0]^T$, где $f(\mathbf{x}^{(0)}) = 24,2$. Начальные направления поиска следующие:

$$\mathbf{s}_1^{(0)} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \mathbf{s}_2^{(0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Сначала минимизация $f(\mathbf{x})$ осуществляется в направлении поиска $\mathbf{s}_1^{(0)}$, т. е. в направлении x_1 (детали этой процедуры здесь не приводятся), что приводит к точке $\mathbf{x}_1^{(0)} = [-0,995 \ 1,000]^T$, в которой $f(\mathbf{x}_1^{(0)}) = 3,990$. Затем осуществляется поиск, использующий вектор $\mathbf{s}_2^{(0)}$, т. е. в направлении x_2 , что дает $\mathbf{x}_2^{(0)} = [-0,995 \ 0,990]^T$, где $f(\mathbf{x}_2^{(0)}) = 3,980$. После выполнения шага

$$\mathbf{x}_3^{(0)} = 2 \begin{bmatrix} -0,995 \\ 0,990 \end{bmatrix} - \begin{bmatrix} -1,200 \\ 1,000 \end{bmatrix} = \begin{bmatrix} -0,790 \\ 0,980 \end{bmatrix},$$

где $f(\mathbf{x}_3^{(0)}) = 15,872$, вычисляются новые направления поиска:

$$\mathbf{s}_1^{(1)} = \mathbf{s}_2^{(0)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

$$\mathbf{s}_2^{(1)} = \mathbf{x}_2^{(0)} - \mathbf{x}_0^{(0)} = \begin{bmatrix} -0,995 \\ 0,990 \end{bmatrix} - \begin{bmatrix} -1,200 \\ 1,000 \end{bmatrix} = \begin{bmatrix} 0,205 \\ -0,010 \end{bmatrix},$$

$$\sqrt{(s_2^{(1)})_1^2 + (s_2^{(1)})_2^2} = \sqrt{4,21 \cdot 10^{-2}} = 0,206,$$

$$\mathbf{s}_2^{(1)} = \begin{bmatrix} 0,999 \\ -0,0488 \end{bmatrix},$$

поскольку не удовлетворяется упомянутый выше (см. шаг 3) критерий ($\Delta^{(0)} = 24,2 - 3,99 = 20,21$),

$$f(\mathbf{x}_3^{(0)}) = 15,872 < f(\mathbf{x}^{(0)}) = 24,2 \quad (a)$$

и

$$[24,2 - 2 \cdot 3,980 + 15,372][24,2 - 2,980 - 20,21]^2 < \\ < 0,5 \cdot 20,21 \cdot (24,2 - 15,872)^2. \quad (6)$$

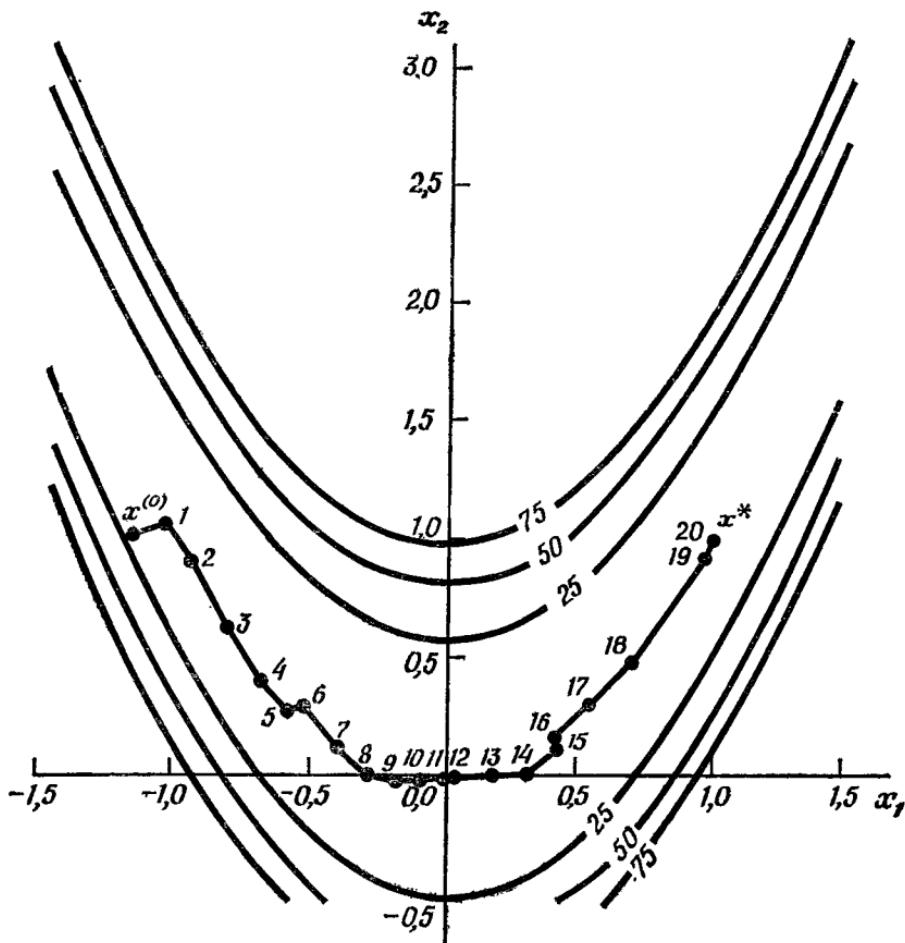
Результаты нескольких дополнительных этапов поиска, начиная с точки $\mathbf{x}_0^{(1)} = [-0,990 \ 0,990]^T$, приведены в следующей таблице:

Этап	$(\mathbf{s}_1^{(k)})^T$	$(\mathbf{s}_2^{(k)})^T$	$\mathbf{x}^{(k)}$	$t(\mathbf{x}^{(k)})$
1	[0 1]	[0,205 -0,010]	$x_0^{(1)} = [-0,990 \ 0,990]^T$	3,969
			$x_1^{(1)} = [-0,990 \ 0,990]^T$	3,959
			$x_2^{(1)} = [-0,984 \ 0,979]^T$	3,948
2	[0 1]	[0,453 -0,892]	$x_0^{(2)} = [-0,761 \ 0,540]^T$	3,257
			$x_1^{(2)} = [-0,761 \ 0,579]^T$	3,101
			$x_2^{(2)} = [-0,702 \ 0,462]^T$	2,986
3	[0,453 - 0,892]	[0,608 -0,794]	$x_0^{(3)} = [-0,503 \ 0,203]^T$	2,510
			$x_1^{(3)} = [-0,538 \ 0,273]^T$	2,396
			$x_2^{(3)} = [-0,466 \ 0,178]^T$	2,301

На фиг. П.4.4.2 представлена траектория поиска. В общей сложности для получения минимума целевая функция вычислялась 1562 раза, при этом минимум имел место в точке $\mathbf{x}^* = [1,000 \ 1,000]^T$ и $f(\mathbf{x}^*) = 1,338 \cdot 10^{-16} \approx 0$.

4.5. МЕТОДЫ СЛУЧАЙНОГО ПОИСКА

Излагаемые ниже методы случайного поиска наименее изящны и эффективны по сравнению со всеми другими алгоритмами поиска, однако благодаря использованию современных высокоскоростных цифровых и гибридных вычислительных машин эти методы все-таки оказываются практически полезными. Обзор различных типов методов случайного поиска сделан Бруксом [13]. Фавро [14] и Мансон предложили методы выполнения случайного поиска на аналоговой вычислительной машине. Митчелл [15] предложил метод, использующий гибридную вычислительную машину, в которой выбор различных стратегий случайного поиска и изменение величины



Ф и г. П.4.4.2. Траектория поиска при использовании алгоритма Паузлла (числа обозначают количество вычислений целевой функции).

шага осуществляются логическим блоком цифровой части. Ниже кратко излагаются некоторые процедуры, использующие случайный поиск.

4.5.1. КОМПЛЕКСНЫЙ МЕТОД

Хотя комплексный метод был разработан Боксом [16] применительно к задачам нелинейного программирования с ограничениями в виде неравенств, мы включили его в настоящий раздел потому, что он основывается на использовании случайных направлений поиска. Этот метод возник из симплексного метода Спенди и др., описанного в разд. 4.2. Здесь вершины вычеркиваются и добавля-

ются, как и в симплексном методе, но не делаются попытки сохранить регулярную фигуру, что характерно для симплекс-метода.

Затруднение, встречающееся при использовании метода Спенди и Нелдера, а также метода Мида при часто повторяющемся ограничении, состоит в том, что необходимо удалять каждый раз недопустимую вершину многогранника, пока не будет получена допустимая вершина. После ряда таких операций многогранник становится размерности ($n - 1$) или меньшей и процедура поиска значительно замедляется. Более того, если данное ограничение перестает быть активным, то такой «сплющенный» многогранник трудно «расширить» снова в n -мерное пространство. Чтобы избежать этих трудностей, Бокс выбрал многогранник с более чем $n + 1$ вершиной, который он назвал *комплексом*. (Митчелл и Каплан, работа которых приведена в списке литературы в конце этой главы, описали комплексный метод, не использующий случайную процедуру.)

В комплексном методе используются ($n + 1$) или более вершин p (каждая из которых должна удовлетворять ограничениям на всех k этапах). Сначала определяется некоторая начальная точка $x_i^{(0)}$, а затем выбираются ($p - 1$) дополнительных вершин с помощью псевдослучайных чисел в соответствии со следующим соотношением:

$$x_i^{(0)} = L_i + r_i^{(0)}(U_i - L_i), \quad i = 2, \dots, p,$$

где L_i и U_i представляют собой соответственно нижнюю и верхнюю границы для x_i , а $r_i^{(0)}$ является диагональной матрицей псевдослучайных чисел, равномерно распределенных на интервале (0,1). Если границы неизвестны, то исходный многогранник должен быть выбран так, чтобы он покрывал область поиска.

Затем целевая функция вычисляется в каждой вершине, и вершина, в которой $f(x)$ имеет наихудшее значение, заменяется новой вершиной, находящейся на прямой, проходящей через отброшенную точку и центр тяжести оставшихся точек на расстоянии, равном или большем, чем расстояние от отброшенной точки до центра тяжести. Если окажется, что в новой вершине имеет место наихудшее значение $f(x)$ по сравнению со всеми вершинами в новом многограннике, она заменяется другой вершиной, расположенной на расстоянии, равном половине расстояния от новой вершины до центра тяжести. (Если нарушается ограничение, то новая вершина также передвигается на половину расстояния к центру тяжести.) Исходя из эмпирических исследований, Бокс предложил, чтобы растяжение многогранников определялось множителем, равным 1,3, и чтобы при поиске использовались $p = 2n$ вершин. Растяжение на этапе отражения многогранника и использование более чем ($n + 1$) вершин являются как раз теми чертами процедуры, которые предназначены для предотвращения «уплощения» многогранника, когда поиск происходит вблизи ограничений. Процедура

поиска продолжается, пока многогранник не будет стянут в центр тяжести в пределах заданной точности.

Числовые результаты, полученные Хиллари [17], показали, что скорость сходимости комплексного метода зависит от характера исходного многогранника. С другой стороны, Бокс пришел к выводу, что метод Розенброка более эффективен, чем симплексный или комплексный метод для задач без ограничений, и что при увеличении размерности n функции $f(\mathbf{x})$ количество необходимых вычислений целевой функции для комплексного и симплексного методов возрастает вдвое быстрее, чем для метода Розенброка.

4.5.2. ПОВТОРЯЮЩИЙСЯ СЛУЧАЙНЫЙ ПОИСК

Келли и Уилинг [18] предложили алгоритм полностью случайного поиска на каждом этапе минимизации. После того как задана начальная точка $\mathbf{x}^{(0)}$, строится случайная траектория для последовательности шагов, каждый из которых проводится в направлении от некоторой точки \mathbf{x} , где $f(\mathbf{x})$ минимальна на данном этапе случайного поиска, до следующей точки \mathbf{x} , соответствующей еще более низкому значению функции. Таким образом, начиная из точки $\mathbf{x}^{(0)}$, случайное значение $\mathbf{x}^{(1)}$ получается по следующему соотношению:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \left[\beta \frac{\mathbf{z}^{(k)}}{\|\mathbf{z}^{(k)}\|} + (1 - \beta) \mathbf{r}^{(k)} \right], \quad (4.5.1)$$

где

$\lambda^{(k)}$ — величина шага, скаляр, который увеличивается после успешного шага и уменьшается после неудачного шага;

$\mathbf{z}^{(k)}$ — вектор «предыстории», указывающий среднее направление поиска на предыдущих шагах:

$$\mathbf{z}^{(k+1)} = \gamma \mathbf{z}^{(k)} + (1 - \gamma) (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}) \mathbf{s}^{(k)};$$

$\mathbf{r}^{(k)}$ — единичный вектор нормальных отклонений, реализуемый генератором псевдослучайных чисел;

β — коэффициент, изменяемый в процессе поиска;

γ — постоянный весовой множитель;

$\mathbf{s}^{(k)}$ — вектор масштабных множителей для «подходящего» масштабирования пространства \mathbf{x} ;

На k -м этапе, чтобы получить \mathbf{x}^{k+1} , случайный вектор $\mathbf{r}^{(k)}$ и вектор предыстории $\mathbf{z}^{(k)}$ усредняются, как видно из соотношения (4.5.1). Вектор \mathbf{x}^{k+1} будет принят или отвергнут в зависимости от того, выполняется или нет неравенство $f(\mathbf{x}^{k+1}) < f(\mathbf{x}^{(k)})$. После того как \mathbf{x}^{k+1} принят (или отвергнут), $\lambda^{(k)}$ увеличивают (или уменьшают) с помощью множителя, грубо говоря, зависящего



Ф и г. 4.5.1. Типичная траектория при случайному поиске. Пунктиром стрелками представлены исследуемые шаги, которые были отвергнуты, а сплошными стрелками — шаги, на которых значения целевой функции уменьшались.

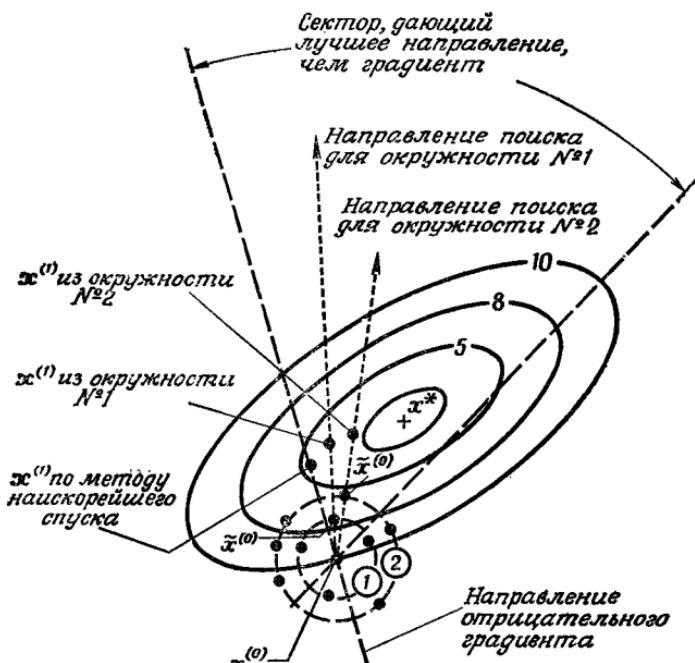
от того, был ли поиск трудным или легким. На фиг. 4.5.1 приведена гипотетическая траектория поиска в двумерном случае.

Проведенные испытания этой программы на некоторых задачах показывают, что она работает в общем случае менее удовлетворительно, чем другие алгоритмы, описанные ранее в гл. 3 и в настоящей главе. Мы не будем обсуждать этот вопрос подробно (тем не менее читатель может обратиться к табл. 9.3.3 и 9.3.4).

4.5.3. СЛУЧАЙНЫЙ ПОИСК С ПОСТОЯННЫМ РАДИУСОМ ПОИСКА И СЛУЧАЙНЫМ НАПРАВЛЕНИЕМ

На фиг. 4.5.2 проиллюстрирован метод поиска, в котором радиус поиска в любом направлении постоянен, но направление поиска случайно. В двумерном случае проводится окружность с начальным вектором $x^{(0)}$ в качестве центра. В n -мерном случае это соответствует гиперсфере. В выбранных случайным образом точках на этой окружности вычисляют $f(x)$ и отмечают точку $\tilde{x}^{(0)}$ с наилучшим значением целевой функции. Затем проводят поиск вдоль прямой, проходящей через $x^{(0)}$ и $\tilde{x}^{(0)}$, и отмечают точку $x^{(1)}$, где $f(x)$ имеет минимальное значение. После этого процедура повторяется, начиная с точки $x^{(1)}$. С увеличением радиуса окружности поиска (однако так, чтобы он не был слишком большим) точка $x^{(1)}$ будет приближаться к x^* (искомая точка минимума). Любая точка на сегменте, заключенном между пунктирными линиями на фиг. 4.5.2, лучше, чем точка, определенная минимизацией по методу

наискорейшего спуска, но хуже, чем точка $\bar{x}^{(1)}$, полученная по методу, использующему производные второго порядка. Чтобы получить удовлетворительную сходимость, последовательность радиусов окружностей (или гиперсфер) периодически уменьшается. Поиск может быть ускорен, если точки на гиперсфере выбираются случайно, но с ограничением, чтобы они отклонялись по крайней мере



Ф и г. 4.5.2. Случайный поиск, иллюстрирующий эффект изменения радиуса поиска.

не более чем на некоторый минимальный угол как по отношению друг к другу, так и к предыдущему направлению поиска. Этот минимальный угол может быть также изменен во время поиска.

Можно показать, что если выбирать направляющие косинусы D_i , определенным образом, то случайные направления поиска будут покрывать пространство x по закону равной вероятности. Действительно, определим направляющие косинусы следующим образом:

$$D_i = \frac{d_i}{(d_1^2 + \dots + d_n^2)^{1/2}},$$

где d_i — случайная переменная, подчиняющаяся нормальному закону распределения с математическим ожиданием, равным 0, и дисперсией σ^2 . В двумерном случае угол поиска равен

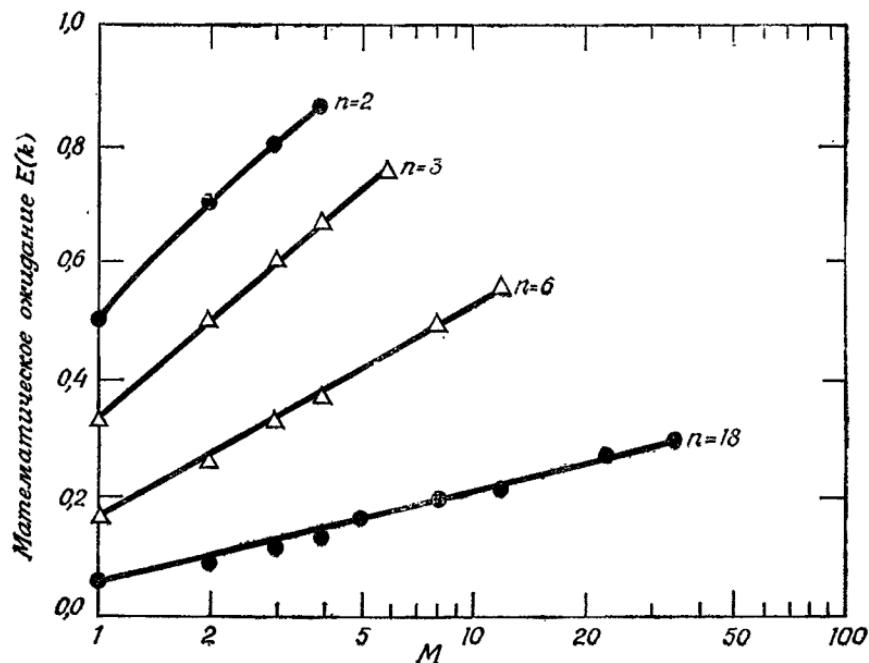
$$\phi = \operatorname{arctg} \frac{D_1}{D_2} = \operatorname{arctg} \frac{d_1}{d_2}.$$

Можно показать, что переменная ϕ равномерно распределена в интервале $(-\pi/2, \pi/2)$. Если размерность задачи равна n , то углы ϕ_k , спроектированные на любую координатную плоскость, также равномерно распределены.

Для сравнения случайного поиска, описанного выше, с неслучайным поиском определим относительное улучшение, или «выигрыш», в $f(\mathbf{x})$ за один этап как

$$E(k) = \frac{f(\mathbf{x}^{(k)}) - f(\mathbf{x}^{(k+1)})}{f(\mathbf{x}^{(k)})}.$$

Если в окрестности $\mathbf{x}^{(k)}$ на поверхности гиперсферы выбирается M векторов \mathbf{x} и линейный поиск проводится в каждом направлении



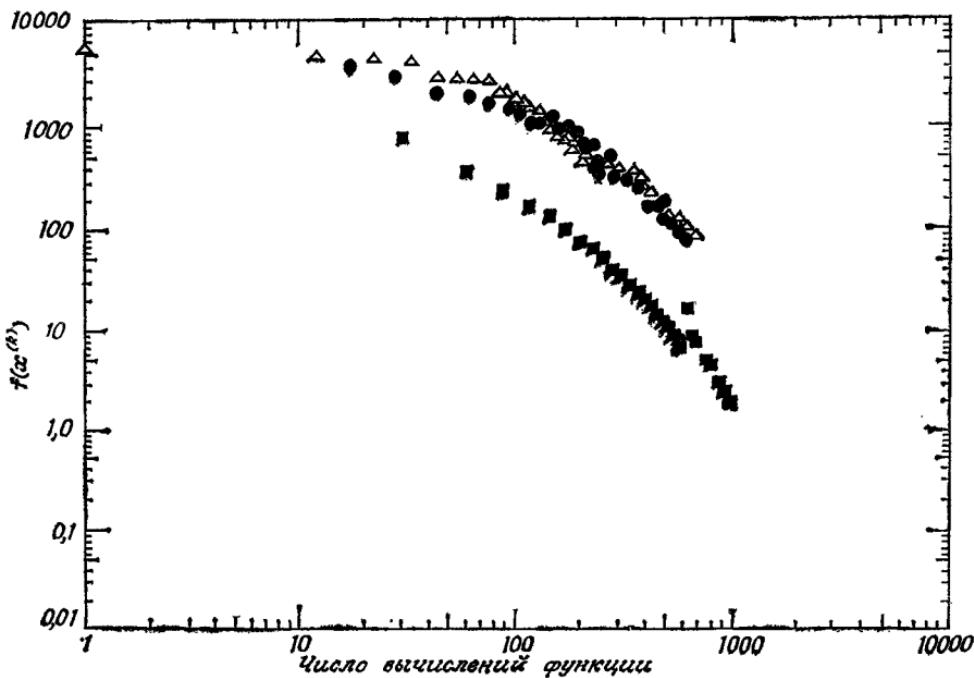
Фиг. 4.5.3. Математическое ожидание выигрыша на цикл в зависимости от числа векторов x , использованных для определения направления поиска (целевая функция — сфероидальная) [20].

n — размерность задачи.

из $\tilde{\mathbf{x}}^{(k)}$, то можно получить математическое ожидание величины $E(k)$ как функции M и размерности n . На фиг. 4.5.3 приведены графики математического ожидания $E(k)$ для задачи со сфероидальной целевой функцией. Эти графики могут быть интерпретированы следующим образом. Сравним $n = 18$ и $M = 3$, что дает $E(3) \approx 0,12$, с $n = 18$ и $M = 6$, что дает $E(6) \approx 0,17$. На двух циклах с $M = 3$ выигрыш будет $\sim 0,24$ по сравнению с 0,17 для того же числа

вычислений целевой функции при $M = 6$. Следовательно, при $n = 18$ более эффективно использовать меньшее число векторов \mathbf{x} на гиперсфере, но чаще делать итерации.

Общий принцип, заключающийся в том, что для хорошо промасштабированных функций регулярная процедура минимизации более



Ф и г. 4.5.4. Сравнение трех методов для случая 18-мерной задачи.

■ градиентный метод; Δ повторяющийся случайный поиск; ● метод, использующий направления линейного поиска, определяемые по 18 случайным точкам на гиперсфере,

$$\begin{aligned} f(\mathbf{x}) = & x_1^2 + 2x_2^2 + 3x_3^2 + 4x_4^2 + 5x_5^2 + 6x_6^2 + 7x_7^2 + 8x_8^2 + 9x_9^2 + 10x_{10}^2 + \\ & + 0,9x_{11}^2 + 0,8x_{12}^2 + 0,7x_{13}^2 + 0,6x_{14}^2 + 0,5x_{15}^2 + 0,4x_{16}^2 + 0,3x_{17}^2 + 0,2x_{18}^2. \end{aligned}$$

эффективна, чем чисто случайная, проиллюстрирован на фиг. 4.5.4, где сравниваются три метода:

- 1) градиентный метод (подразд. 3.1.1);
- 2) повторяющийся случайный поиск (подразд. 4.5.2);
- 3) метод, использующий направления линейного поиска, определяемые по случайным точкам на гиперсфере (подразд. 4.5.3).

Второй и третий методы минимизации приблизительно равнозначны по эффективности и оба менее предпочтительны, чем первый метод. Случайные методы и методы «последовательного перебора переменных» без поворота осей менее эффективны, чем неслучайные методы, описанные в предыдущих разделах.

ЗАДАЧИ¹⁾

4.1. Для целевой функции

$$f(\mathbf{x}) = 3x_1^2 + 5x_2^2$$

проводите три этапа поиска Хука — Дживса. Используйте $\Delta \mathbf{x}^{(0)} = [0,5 \ 0,5]^T$, начиная с базисной точки $\mathbf{x}^{(0)} = [2 \ 1]^T$. (На каждом этапе проводится локальное исследование с последующим ускорением.)

4.2. Продолжите поиск Хука — Дживса из точки $x^{(7)}$ примера 4.4.1 для — этапов.

4.3. Определите регулярную симплексную фигуру в трехмерном пространстве, такую, что расстояние между вершинами равно 0,2 и одна вершина находится в точке $(-1,2 \ -2)$.

4.4. Используйте симплексную фигуру, построенную в задаче 4.3, для проведения восьми циклов отbrasывания вершин и получения новых при поиске минимума целевой функции

$$f(\mathbf{x}) = x_1^2 + 3x_2^2 + 5x_3^2.$$

4.5. Трехмерный оптимальный симплексный поиск минимума дал следующие промежуточные результаты:

Вектор \mathbf{x}	Значение целевой функции
$[0 \ 0 \ 0]^T$	4
$[-1/3 \ -1/3 \ -1/3]^T$	7
$[-1/3 \ -4/3 \ -1/3]^T$	10
$[-1/3 \ -1/3 \ -4/3]^T$	5

Какая следующая точка подлежит вычислению в процессе поиска? Какая точка опущена? Что будет центром тяжести нового симплекса?

4.6. Для функции

$$f(\mathbf{x}) = 4x_1^2 + x_2^2 - 40x_1 - 12x_2 + 136,$$

начиная из точки $\mathbf{x}^{(0)} = [4 \ 8]^T$, проведите — этапов поиска Нелдера и Мида, включая отражение, расширение или сжатие и т. д., пока не будут исключены три исходные вершины симплекса.

4.7. Что будет исходным симплексом в случае поиска Нелдера

¹⁾ Дополнительные задачи, подходящие для этой главы, можно найти среди задач, помещенных в конце гл. 3.

и Мида для функции $f(\mathbf{x}) = 2x_1^2 + x_2^4$? Какая из вершин исключается первой? Второй?

4.8. Путем подбора величин β и γ в алгоритме Нелдера и Мида найдите их оптимальные значения для задач 26—32 из приложения А.

4.9. Придавая параметрам α и β в методе Розенброка разные значения, определите наиболее эффективные из них для случая квадратичной функции

$$f(\mathbf{x}) = 10x_1^2 + 0,1x_2^2.$$

Повторите то же для

$$f(\mathbf{x}) = 0,1x_1^2 + 10x_2^2.$$

4.10. В конце одного этапа метода Розенброка имели место следующие значения при поиске минимума $f(\mathbf{x})$:

Результат		x_1	x_2
x_1	x_2		
Начало	Начало	3	7
Неудача	Неудача	4	8
Успех	Успех	2	8
Успех	Успех	1,5	7,5
Неудача	Неудача	-1,5	10,5
Неудача			

- а) Какие два направления поиска использовались на этом этапе?
 б) Каковы должны быть два новых направления?

4.11. Используйте поиск Розенброка для обнаружения минимума целевой функции

$$f(\mathbf{x}) = 3x_1^2 + x_2^2$$

при $\beta = 1/2$ и $\alpha = 3$, пока дальнейшее улучшение не станет невозможным.

4.12. Каковы будут два следующих этапа в методе Розенброка после вычисления функции $f(\mathbf{x}) = 4(x_1 - 5)^2 + (x_2 - 6)^2$ в точке $\mathbf{x} = [7,35 \ 8,80]^T$? Предположите, что должно быть получено новое направление поиска и при этом $\mathbf{s}_1 = [1 \ 0]^T$, $\mathbf{s}_2 = [0 \ 1]^T$.

4.13. После минимизации функции $f(\mathbf{x}) = 2x_1^2 + x_2^2 + x_1x_2$ методом Пауэлла вектор \mathbf{x} оказался равным $\mathbf{x} = [0,371 \ 0,116]^T$ и $f(\mathbf{x}) = 0,443$.

Затем было вычислено новое направление поиска и достигнута точка $\mathbf{x} = [0,574 \ 0,308]^T$. Каково следующее направление поиска?

4.14. Выпишите направление поиска для метода Пауэлла в слу-

чае минимизации $f(\mathbf{x}) = 2x_1^2 + x_2^2 - x_1x_2$, начиная из точки $\mathbf{x}^{(0)} = [2 \ 2]^T$.

4.15. Найдите направление, ортогональное вектору

$$\mathbf{s} = \left[\frac{1}{\sqrt{3}} \ - \frac{1}{\sqrt{3}} \ - \frac{1}{\sqrt{3}} \right]^T$$

в точке

$$\mathbf{x} = [0 \ 0 \ 0]^T.$$

Найдите направление, сопряженное к \mathbf{s} для целевой функции $f(\mathbf{x}) = x_1 + 2x_2^2 - x_1x_2$ в той же точке.

4.16. Поскольку метод Пауэлла использует сопряженные направления, можно ли гарантировать стысканье минимума квадратичной целевой функции n переменных за n шагов?

4.17. Выпишите четыре первых направления поиска для метода Пауэлла в случае минимизации $f(\mathbf{x}) = x_1^2 + \exp(x_1^2 + x_2^2)$, начиная из точки $\mathbf{x}^{(0)} = [2 \ 2]^T$.

4.18. Объясните, как методом Пауэлла получают набор сопряженных направлений поиска для трехмерной задачи. После продвижения в двух направлениях поиска опишите подробно определение третьего направления. Станут ли все направления поиска сопряженными после двух этапов?

4.19. Даст ли метод Пауэлла тот же набор направлений поиска, что и метод Розенброка, если начать из той же начальной точки \mathbf{x} ?

4.20. Пара уравнений $x_1^2 + x_2^2 = 1$ и $x_1 + x_2 = 1$ имеют решения $(1,0)$ и $(0,1)$. Линии уровней

$$f(\mathbf{x}) = (x_1^2 + x_2^2 - 1)^2 + (x_1 + x_2 - 1)^2$$

показывают, что имеют место минимумы в точках $(1,0)$ и $(0,1)$ и седловая точка $(4^{-1/3}, 4^{-1/3})$. Могут ли методы поиска обнаружить оба минимума или только единственный минимум? Объясните.

4.21. Начиная из точки $\mathbf{x} = [3 \ 3]^T$, постройте траектории (с указанием значений целевой функции $f(\mathbf{x})$ в зависимости от числа итераций) для следующих методов поиска при решении задачи 4.20:

- а) Хука — Дживса,
- б) Нелдера — Мида,
- в) Розенброка,
- г) Пауэлла.

Что произойдет, если начать из седловой точки?

4.22. Рассмотрите функцию

$$f(\mathbf{x}) = (x_1^2 + x_2^2) + \frac{0,9x_1}{\sqrt{x_1^2 + x_2^2}} \left[\frac{3}{2}(x_1^2 - x_2^2) - x_1x_2 + \sqrt{x_1^2 + x_2^2} + 0,25x_1 - \frac{x_1x_2}{\sqrt{x_1^2 + x_2^2}} \right].$$

а) Постройте линию уровня $f(\mathbf{x})$ для $f(\mathbf{x}) = 1$, используя прибор для вычерчивания кривых.

б) С помощью метода наискорейшего спуска за 8 итераций были достигнуты следующие точки:

	x_1	x_2	$f(\mathbf{x})$
(1)	0,119	0,677	1,000
	0,007	0,006	0,006
(2)	-0,598	-0,502	1,000
	-0,012	-0,005	0,005

Сравните их с результатами других методов поиска.

Какой метод оказывается лучшим?

4.23. Найдите наилучшие оценки параметров b_0 , b_1 и c модели

$$\hat{y} = b_0 + b_1 e^{-cx}$$

путем минимизации суммы квадратов разностей $\sum (\hat{y}_i - y_{\text{эксп. } i})^2$, если дано

$y_{\text{эксп. } i}$	x
51,6	0,4
53,4	1,4
20,0	5,4
-4,2	19,5
-3,0	48,2
-4,8	95,9

Используйте методы

- а) Хука — Дживса,
- б) Нелдера — Мида,
- в) Пауэлла,
- г) Розенброка,
- д) случайного поиска.

4.24. Минимизируйте полную величину капиталовложений на расширение производства P в (долларах), если дано, что

$$P = 4900k_0^{0,88} + \sum_{i=1}^{n-1} k_i^{0,88} \left(\frac{10}{\sum_{j=0}^{n-1} k_j} \right)^{0,559} + \left(60 - \sum_{j=0}^{n-1} k_j \right)^{0,88} \left(\frac{10}{\sum_{j=0}^{n-1} k_j} \right)^{0,559},$$

где k_0 — начальный размер вложений, а k_i — размер дополнительных вложений, в целых числах, т. е. $k_i = 1, 2, 3, \dots$. Найдите минимальное n , $n = 1, 2, 3, \dots$, и соответствующие k , если $k_0 = 1$.

4.25. Найдите минимум и максимум функции

$$f(x) = 20 + 0,3x_1 - 4x_2 + 0,3x_1^2 + 0,3x_2^2 + 0,4x_1x_2$$

одним из методов поиска. Начинайте из точек

- a) $x^{(0)} = [0,25 \quad 2,5]^T$,
- б) $x^{(0)} = [2,5 \quad 2,5]^T$,
- в) $x^{(0)} = [-0,25 \quad -2,5]^T$.

4.26. Функция

$$f(x) = \left(1 + 8x_1 - 7x_1^2 + \frac{7}{3}x_1^3 - \frac{1}{4}x_1^4\right)(x_2^2 e^{-x_2}) F(x_3)$$

имеет два максимума и седловую точку. Для (а) $F(x_3) = 1$ и (б) $F(x_3) = x_3 e^{-(x_3+1)}$ найдите глобальный оптимум с помощью какого-либо метода поиска.

[Ответ: (а) $x^* = [4 \quad 2]^T$ и (б) $x^* = [4 \quad 2 \quad 1]^T$.]

4.27. Можно ли найти решение задачи 4.26, начиная из точек (а) $x^{(0)} = [2 \quad 1]^T$ и (б) $x^{(0)} = [2 \quad 1 \quad 1]^T$.

Повторите для (а) $x^{(0)} = [2 \quad 2]^T$ и (б) $x^{(0)} = [2 \quad 2 \quad 1]^T$.

(Указание: $[2 \quad 2 \quad 1]^T$ — седловая точка.)

4.28. Имеет ли функция

$$f(t) = 1 + t + t^2 + t^3$$

минимум? Максимум?

4.29. Минимизируйте

(а) $f(x) = 1 + x_1 + x_2 + x_3 + x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1^2 + x_2^2 + x_3^2 + x_4^2$, начиная из точек $x^{(0)} = [-3 \quad -30 \quad -4 \quad -0,1]^T$ и $x^{(0)} = [0,5 \quad 1,0 \quad 8,0 \quad -0,7]^T$,

(б) $f(x) = x_1^2 x_2^3 x_3^3 x_4^4 [\exp(-(x_1 + x_2 + x_3 + x_4))]$, начиная из точки $x^{(0)} = [3 \quad 4 \quad 0,5 \quad 1]^T$.

4.30. Определите коэффициенты уравнения

$$y = ax_1^{b_1} x_2^{b_2}$$

по следующим экспериментальным данным путем минимизации суммы квадратов разностей между экспериментальными и предсказанными значениями y .

$y_{\text{эксп. } t}$	x_1	x_2
46,5	2,0	36,0
591	6,0	8,0
1285	9,0	3,0
36,8	2,5	6,25
241	4,5	7,84
1075	9,5	1,44
1024	8,0	4,0
151	4,0	7,0
80	3,0	9,0
485	7,0	2,0
632	6,5	5,0

4.31. Стоимость очищенной нефти, перевозимой морским путем через Малаккский пролив в Японию (в долларах на килолитр), определяется в виде линейной суммы стоимости неочищенной нефти, затрат на страхование, таможенных тарифов, затрат на фрахт нефти, затрат на погрузку и разгрузку, платы за морскую стоянку судна, затрат, связанных с подводным перекачиванием и хранением, стоимости площади под цистернами, стоимости очистки и затрат на перевозку продуктов [19]:

$$c = c_c + c_i + c_x + \frac{2,09 \cdot 10^4 t^{-0,3017}}{360} + \frac{1,064 \cdot 10^6 a t^{0,4925}}{52,47q \cdot 360} + \\ + \frac{4,242 \cdot 10^4 a t^{0,7952} + 1,813 i p (n + 1,2q)}{52,47q \cdot 360} + \frac{4,25 \cdot 10^8 a (nt + 1,2q)}{52,47q \cdot 360} + \\ + \frac{5,042 \cdot 10^3 q^{-0,1899}}{360} + \frac{0,1049 q^{0,671}}{360},$$

где

a — фиксированные ежегодные расходы в относительных величинах (0,20);

c_c — цена неочищенной нефти, долл/кл (12,50);

c_i — страховка, долл/кл (0,50);

c_x — таможенные тарифы, долл/кл (0,90);

i — норма процента (0,10);

n — число портов (2);

p — цена земли, долл/м² (7000);

q — производительность установки для очистки нефти, баррель/день;

t — объем танкера, кл.

Считая значения, указанные в скобках, заданными, вычислите минимальную стоимость нефти, оптимальный объем танкера и

производительность установки для очистки нефти следующими методами (заметим, что 1 кл = 6,29 баррель):

- а) Хука — Дживса,
- б) Нелдера — Мида,
- в) Розенброка,
- г) Пауэлла,
- д) случайного поиска.

ЛИТЕРАТУРА

1. Hooke R., Jeeves T. A., *J. Assoc. Computer Mach.*, **8**, 212 (1962).
2. Wood C. F., Application of «Direct Search» to the Solution of Engineering Problems, Westinghouse Res. Lab. Sci. Paper 6-41210-1-P1, 1960.
3. Nelder J. A., Mead R., *Computer J.*, **7**, 308 (1964).
4. Spendley W., Hext G. R., Hinsworth F. R., *Technometrics*, **4**, 441 (1962).
5. Box M. J., *Computer J.*, **8**, 42 (1965); Campey I. G., Nickols D. G., Simplex Minimization, Imperial Chem. Industries, Ltd., 1961.
6. Paviani D., Ph. D. Dissertation, The Univ. of Texas, Austin, Tex., 1969.
7. Rosenbrock H. H., *Computer J.*, **3**, 175 (1960).
8. Swann W. H., Report on the Development of a New Direct Search Method of Optimization, Imperial Chem. Industries, Ltd. Central Instr. Lab. Res. Note 6413, 1964.
9. Palmer J. R., *Computer J.*, **12**, 69 (1969).
10. Powell M. J. D., *Computer J.*, **7**, 155 (1964); **7**, 303 (1965).
11. Smith C. S., The Automatic Computation of Maximum Likelihood Estimates, NCB Sci. Dept. Rept. SC846/MR/40, 1962.
12. Zangwill W. I., *Computer J.*, **10**, 293 (1967).
13. *J. Operations Res.*, **6**, 244 (1958).
14. Favreau R. R., Franks R. G. E., Statistical Optimization, Proc. 2nd Intern. Conf. for Analog Computation, Strasbourg, 1958, Presses Académiques Européennes, Brussels, 1959, p. 437.
15. Mitchell B. A., *Simulation*, **4**, 399 (1965).
16. Box M. J., *Computer J.*, **8**, 42 (1965).
17. Hilleary R. R., U. S. Naval Postgraduate School Techn. Rept./Res. Paper 59, March 1966.
18. Kelly R. J., Wheeling R. F., A Digital Computer Program for Optimizing Non-linear Functions, Mobil Oil Corp., Research Dept., Central Research Div., Princeton, N. J., July 1962.
19. Uchiyama T., *Hydrocarbon Process*, **47** (12), 85 (1968).
20. Kushner H., Efficient Iterative Methods for Optimizing the Performance of Multi-parameter Noisy Systems, MIT Lincoln Lab. Rept. 22G-0043 (AD 245802), Oct. 1960.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

ОБЩИЕ ВОПРОСЫ

- Box M. J., A Comparison of Several Current Optimization Methods, and the Use of Transformations in Constrained Problems, *Computer J.*, **9**, 67 (1966).
 Dorn W. S., Nonlinear Programming: A Survey, *Management Sci.*, **9**, 171 (1963).
 Kowalik J., Osborne M. R., Methods for Unconstrained Optimization Problems, American Elsevier, 1968.

- Powell M. J. D., A Survey of Numerical Methods for Unconstrained Optimization, *SIAM Rev.*, **12**, 79 (1970).
- Schechter R. S., Beveridge G. S. G., Optimization: Theory and Practice, McGraw-Hill, N. Y., 1970.
- Spang H. A., III, A Review of Minimization Techniques for Nonlinear Functions, *SIAM Rev.*, **4**, 343 (1962).
- Wilde D. J., Optimum Seeking Methods, Prentice-Hall, Englewood Cliffs, N. J., 1964.
- Wilde D. J., Beightler C. S., Foundations of Optimization, Prentice-Hall Englewood Cliffs, N. J., 1967.
- Wolfe P., Recent Developments in Nonlinear Programming, *Advan. Computers*, **3**, 155—187 (1962).

ДРУГИЕ МЕТОДЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ
БЕЗ ОГРАНИЧЕНИЙ, НЕ ИСПОЛЬЗУЮЩИЕ ПРОИЗВОДНЫЕ
НЕСЛУЧАЙНЫЙ ПОИСК

- Berman G., Minimization by Successive Approximations, *SIAM Numerical Analysis*, **3**, 123 (1966).
- Berman G., Lattice Approximations to the Minima of Functions of Several Variables, *J. Assoc. Computer Mach.*, **16**, 286 (1969).
- Campey I. G., Nickols D. G., Simplex Minimization, Imperial Chemical Industries, Ltd., Aug. 1961.
- Fletcher R., Functional Minimization without Evaluating Derivatives, *Computer J.*, **8**, 33 (April 1965).
- Kiefer J., Sequential Minimax Search for a Minimum, *Proc. Am. Math. Soc.*, **4**, 502 (1953).
- Kiefer J., Optimum Sequential Search and Approximation Methods under Minimum Regularity Assumptions, *SIAM J.*, **5**, 105 (1957).
- Mitchell R. A., Kaplan J. L., Nonlinear Constraint Optimization by a Nonrandom Complex Method, *J. Res. Natl. Bur. Std.*, **72C**, 249 (1968).
- Spendley W., Hext G. R., Hinsworth F. R., The Sequential Application of Simplex Designs in Optimization and Evolutionary Operation, *Technometrics*, **4**, 441 (1962).
- Swann W. H., Report on the Development of a New Direct Search Method of Optimization, Imperial Chemical Industries, Ltd., Central Instr. Lab. Res. Note 64/3, 1964.
- Vignes J., Algorithme pour la détermination d'un extremum local d'une fonction de plusieurs variables, *Rev. Inst. Franç. Pétrole*, **23**, 537 (1968).
- Whitte B. F. W., Two New Direct Minimum Search Procedures for Functions of Several Variables, Spring Joint Computer Conf., Washington, D. C., April 1964.
- Wood C. F., Application of «Direct Search» to the Solution of Engineering Problems, Westinghouse Res. Lab. Sci. Paper 6-41210-1-PI, 1960.

СЛУЧАЙНЫЙ ПОИСК

- Bekey G. A., Gran M. H., Sabroff A. E., Wong A., Parameter Optimization by Random Search Using Hybrid Computer Techniques, Proc. Fall Joint Computer Conf., 1966, p. 191.
- Brooks S. H., A Discussion of Random Methods for Seeking Maxims, *J. Operations Res.*, **6**, 244 (1958).

- Brooks S. H., A Comparison of Maximum Seeking Methods, *J. Operations Res. Soc. Am.*, 7 (1959).
- Favreau R. R., Franks R. G. E., Random Optimization by Analog Techniques, Proc. 2nd Intl. Conf. for Analog Computation, Strasbourg, France, Sept. 1958, Presses Académiques Européennes, Brussels, 1959, pp. 437, 443.
- Gallagher P. J., MOP-1, An Optimizing Routine for the IBM 650, Can. G. E. Civilian Atomic Power Dept. Rept. R60cAP35, 1960.
- McArthur D. S., Strategy in Research, Alternative Methods for the Design of Experiments, *IRE Trans.*, EM-8, 34 (1961).
- Matyas J., Random Optimization, *Automatic and Remote Control*, 26, 244 (1965).
- Mitchell B. A., A. Hybrid Analog-Digital Parameter Optimizer for ASTRAC II, *Simulation*, 4, 398 (1965).
- Munson J. K., Rubin A. I., Optimization by Random Search on the Analog Computer, *IRE Trans.*, EC-8, 200 (1959).
- Schumer M. A., Steiglitz K., *IEEE Trans. Autom. Control*, AC-13, 270 (1968).
- Shimuzu T., A Stochastic Approximation Method for Optimization Problems, *J. Assoc. Computer Mach.*, 16, 511 (1969).
- Zellnik H. E., Sondak N. E., Davis R. S., Gradient Search Optimization, *Chem. Eng. Progr.*, 58 (8), 35 (1962).

Г л а в а 5

СРАВНЕНИЕ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ОТСУТСТВИИ ОГРАНИЧЕНИЙ



В этой главе будет проведено сравнение эффективности алгоритмов, описанных в гл. 3 и 4 (не всех, конечно, но большинства из них). Особый интерес представляют ответы на следующие вопросы:

1. Какие алгоритмы являются лучшими, а какие худшими?
2. Как влияет природа задачи, а именно степень нелинейности, число переменных и т. д. на качество работы алгоритма?
3. Какова эффективность алгоритмов, не использующих производных, по сравнению с алгоритмами, использующими их?
4. Почему определенные алгоритмы в некоторых условиях не работают?

Прежде чем проводить сравнение качества работы различных алгоритмов, необходимо сначала установить подходящие критерии оценки. Сначала рассмотрим различные критерии, а затем, используя два из них, оценим некоторые алгоритмы.

5.1. КРИТЕРИИ ОЦЕНКИ

Прежде чем оценивать эффективность различных алгоритмов при отсутствии ограничений, сделаем несколько замечаний относительно критериев, используемых при оценивании эффективности алгоритмов. Алгоритмы можно исследовать как с теоретической, так и с экспериментальной точек зрения. Первый подход может быть использован только для весьма ограниченного класса задач, поэтому мы будем оценивать эффективность алгоритмов с помощью эксперимента, т. е. решения тестовых задач. Алгоритмы могут быть проверены на специальных задачах как с малым, так и с большим числом переменных, на задачах с различной степенью нелинейности, а также на задачах, возникших из практических приложений, таких, как задачи минимизации суммы квадратов, решение систем нелинейных уравнений и т. п. Можно надеяться, что исследование эффективности алгоритма при решении различных задач позволит предсказать общую эффективность алгоритма при решении других задач.

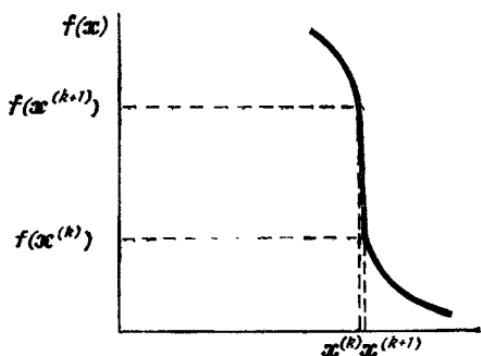
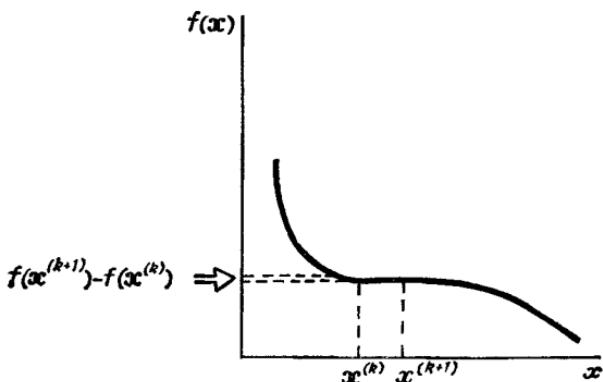
Одним из важнейших вопросов является вопрос о широте применения алгоритма, т. е. можно ли с помощью данного алгоритма решить большинство задач? Конечно, любой алгоритм может ока-

заться бессильным решить специальным образом построенную («патологическую») задачу. Даже и для других, не обязательно «патологических» задач мы не можем требовать, чтобы алгоритм решал *все* возможные задачи, поскольку легко поставить задачу нелинейного программирования без ограничений, приводящую к отрицательным аргументам, делению на нуль, разрывам и т. п. Более того, нельзя ожидать, что данный алгоритм выберет глобальный минимум, если в задаче имеется более одного минимума, но он может считаться успешным, если ему удается достичь по крайней мере локального минимума. Кроме того, то, что один человек считает успехом, другой может расценивать как неудачу. Например, рассмотрим задачу 31 из приложения А, иллюстрированную фиг. 5.2.7 (метод штрафной функции), у которой глобальный минимум находится на $-\infty$, а локальный минимум — в окрестности точки (1,7; 1,3). Если процедура минимизации осуществляется в направлении глобального минимума, должно ли это считаться успехом или неудачей? Если эта процедура используется как подпрограмма в методе штрафной функции, то здесь скорее ищется локальный, а не глобальный минимум.

Результаты любого экспериментального сравнения алгоритмов в значительной степени зависят от того, как алгоритмы запрограммированы для ЭВМ. Мелкие детали программирования могут оказывать существенное влияние на эффективность алгоритма. Небольшие изменения в критериях окончания процесса, процедурах одномерного поиска, тестах на сингулярность матриц, процедурах обращения матриц, перезадания и т. п. сильно влияют на эффективность алгоритма. Даже изменение начального шага в одномерном поиске, как было показано в разд. 3.4, оказывает серьезное влияние на траекторию поиска при минимизации функции Розенброка. Некоторые из этих факторов игнорировались авторами при сообщении результатов проверки того или другого алгоритма, поскольку они считались не связанными собственно с алгоритмом, тем не менее их вклад в работоспособность алгоритма не должен оставаться незамеченным. Для повышения эффективности многих алгоритмов требуется введение эвристической логики. Такая логика основывается на опыте экспериментальных неудач и имеет мало общего с фундаментальным понятием, лежащим в основе алгоритма, но тем не менее делает его работоспособным.

В этой главе рассматриваются следующие критерии для оценивания алгоритмов нелинейного программирования при отсутствии ограничений:

1. Успех в достижении оптимального решения (в пределах заданной точности) для широкого круга задач.
2. Число необходимых вычислений целевой функции.
3. Машинное время, требуемое для реализации алгоритма (в пределах желаемой степени точности).



Ф и г. 5.1.1. Объединенные критерии окончания процесса минимизации $f(x)$ для двумерного случая. Критерий, основанный на неравенстве

$$\frac{f(x^{(k+1)}) - f(x^{(k)})}{f(x^{(k)})} < \varepsilon,$$

приводит к преждевременному окончанию процесса на плоском плато; критерий, основанный только на неравенстве

$$\frac{x^{(k+1)} - x^{(k)}}{x^{(k)}} < \varepsilon,$$

приводит к преждевременному окончанию на крутом спаде.

Ниже мы рассмотрим некоторые вопросы, касающиеся выбора того или другого из этих критериев и связанной с ними неопределенности.

Основной критерий, используемый при оценивании алгоритмов общего типа, заключается в том, может ли данный алгоритм решить большинство предложенных задач. С этим связано понятие о приемлемой точности решения, т. е. точности определения значений целевой функции $f(x^*)$ и элементов вектора x^* . Обычно

точность решения зависит от критериев окончания, применяющихся для определения момента завершения вычислительной процедуры.

Чтобы добиться некоторой стандартизации в критерии окончания, во всех излагаемых здесь (ранее не опубликованных) работах алгоритмы модифицировались так, чтобы одна и та же относительная ошибка как в определении оптимального вектора \mathbf{x} , \mathbf{x}^* , так и в определении $f(\mathbf{x}^*)$ была общей основой при выборе момента прекращения поиска в каждой программе. На фиг. 5.1.1 видно, почему должны удовлетворяться оба критерия. Если алгоритм заканчивается только из-за того, что относительное изменение $f(\mathbf{x})$ меньше некоторого малого числа, плоское плато может вызвать преждевременное окончание. Если же алгоритм заканчивается только на основании относительного изменения в элементах \mathbf{x} , крутой склон может вызвать преждевременное окончание. Использование для этой цели только составляющих градиента может привести к окончанию в седловой точке, а при минимизации штрафных функций составляющие градиента могут быть малы, но, несмотря на это, вектор \mathbf{x} может значительно изменяться. Следует отметить одно обстоятельство, касающееся критериев окончания и заключающееся в том, что, когда $f(\mathbf{x})$ и (или) \mathbf{x} стремятся к нулю, критерием окончания должно быть изменение функции или переменной по отношению к некоторому значению, а не по отношению к их текущему значению, чтобы избежать деления на малое число. Некоторые авторы в качестве критериев окончания использовали норму $\nabla f(\mathbf{x})$, норму \mathbf{x} или норму s , и, хотя эти критерии являются вполне адекватными, в большинстве случаев они имеют те же недостатки, которые проиллюстрированы на фиг. 5.1.1.

Если данная задача может быть решена рассматриваемым алгоритмом, то второй широко применяемый критерий начинает играть важную роль. В этом случае мерой эффективности алгоритма является количество вычислений функции $f(\mathbf{x})$, необходимых для достижения определенной точности в $f(\mathbf{x})$ и \mathbf{x} . Безусловно, этот критерий лучше, чем критерий, использующий число этапов, поскольку последнее меняется в очень широких пределах для разных алгоритмов и во многих алгоритмах понятие «этап» означает нечто совершенно отличное от выбора нового направления поиска. Тем не менее число вычислений целевой функции само по себе не слишком подходит в качестве меры эффективности для алгоритмов с очень разными стратегиями, поскольку число вычислений целевой функции для выбора направления поиска по отношению к числу вычислений функции при движении в данном направлении сильно меняется от стратегии к стратегии. Более того, с каким весом должно браться количества вычислений производных по отношению к вычислениям самой целевой функции, чтобы при оценке методов, использующих производные, учитывались и количество вычислений целевой функции, и количество вычислений производных?

Наконец, количество вычислений функции может быть уменьшено с помощью всевозможных тестов, также требующих машинного времени, специальных эвристических операций, матричных операций и т. д., так что сравнение, основанное только на количестве вычислений функции, может легко ввести в заблуждение.

Следовательно, третий из упомянутых выше критерииев — машинное время, необходимое для реализации алгоритма, также может рассматриваться как мера эффективности алгоритма. Хотя относительное время решения не является особенно подходящей мерой эффективности алгоритмов нелинейного программирования при отсутствии ограничений, за неимением лучшей меры часто приходится использовать именно ее. В разд. 9.1 сделаны некоторые замечания по поводу трудностей, которые могут возникнуть при использовании в качестве критерия времени решения. Для простых тестовых задач времени, требуемое для ввода данных и извлечения из памяти команд печати (но не самой печати) в программе с не очень детальной распечаткой, скажем только x и $f(x)$ на каждом этапе, может быть в 2—3 раза больше времени, оцениваемого без учета этих фаз программы. В ЭВМ, где центральный процессор работает по нескольким программам в режиме разделения времени, полное время с учетом суммарного времени ввода — вывода может в 2 или 3 раза превысить время, необходимое для решения. Таким образом, тип ЭВМ, тщательность программирования алгоритма и характер учитываемого времени имеют существенное значение при использовании времени решения в качестве критерия. Подобная информация обычно опускается в публикациях, описывающих работу того или иного алгоритма.

5.2. ТЕСТОВЫЕ ЗАДАЧИ

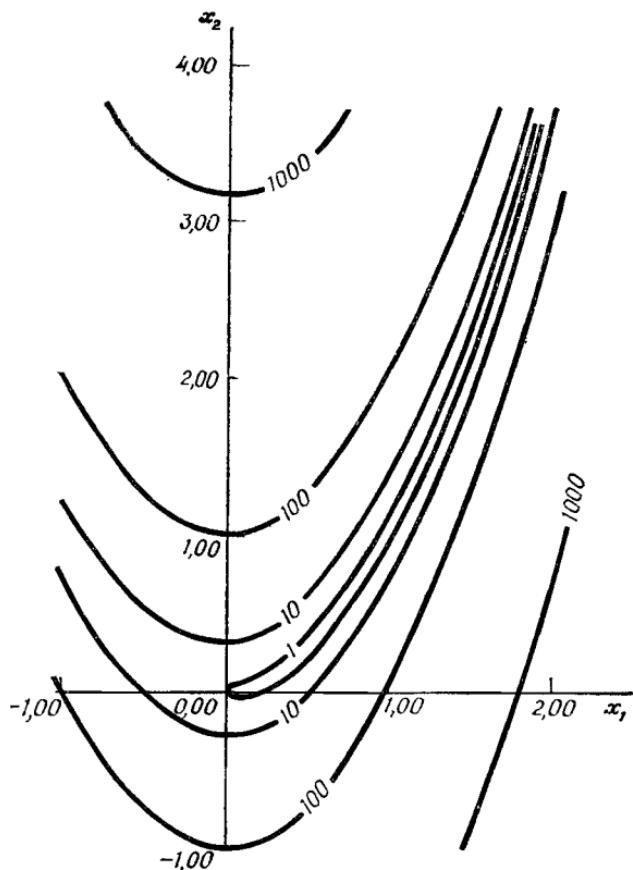
Различные авторы алгоритмов нелинейного программирования использовали для их проверки большое число тестовых задач. Некоторые из этих тестов использовались так часто, что стали играть роль «классических», поскольку много раз применялись для сравнения качества работы разных алгоритмов, обычно, чтобы показать, что новый алгоритм не хуже или лучше предшествующего. В табл. 5.2.1 приведены десять целевых функций, которые достаточно часто встречались в литературе в качестве тестовых функций; задачи 25—35 из приложения А представляют собой дополнительный набор тестовых задач, причем некоторые из них раньше не использовались. На фиг. 5.2.1—5.2.8 приведены линии уровней двумерных целевых функций. Функция Розенброка (функция I в табл. 5.2.1 и задача 2 в приложении А), которая многократно рассматривалась в гл. 3 и 4, имеет крутую закругленную впадину вдоль кривой $x_2 = x_1^2$; целевая функция II также имеет впадину, но более мелкую; у функции III имеется крутая впадина вдоль

Таблица 5.2.1

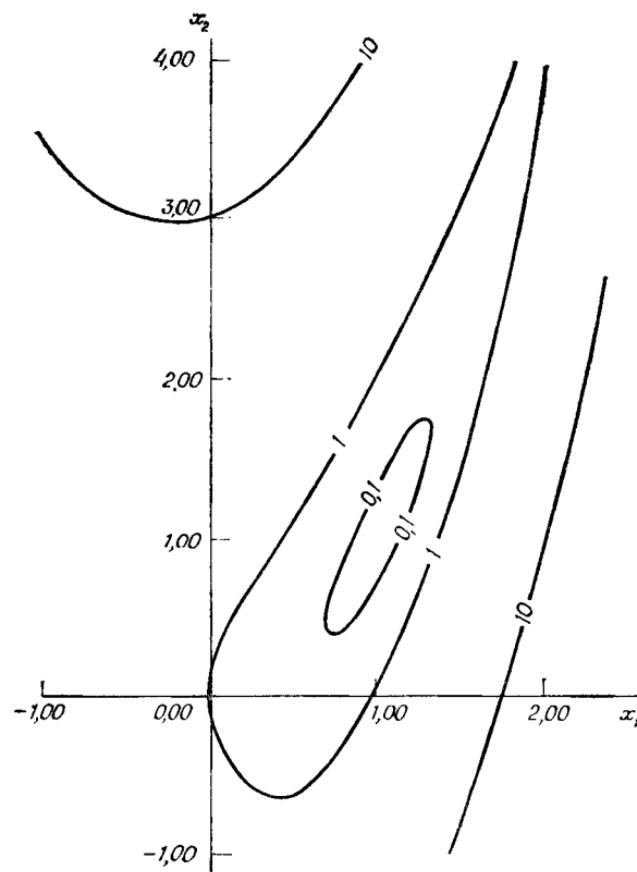
Функции нелинейного программирования при отсутствии ограничений, применяющиеся в качестве тестовых функций

Функция	Литература	Начальный вектор $(x^{(0)})^T$	Значения $f(x)$ в точке минимума	
			$f(x^*)$	x^*
I: $100 (x_2 - x_1^2)^2 + (1 - x_1)^2$	[1]	(-1,2; 1)	0	(1, 1)
II: $(x_2 - x_1^2)^2 + (1 - x_1)^2$	[2]	(-1,2; 1)	0	(1, 1)
III: $(x_2 - x_1^2)^2 + 100 (1 - x_1)^2$	[2]	(-1,2; 1)	0	(1, 1)
IV: $100 (x_2 - x_1^3)^2 + (1 - x_1)^2$	[2]	(-1,2; 1)	0	(1, 1)
V: $[1,5 - x_1 (1 - x_2)]^2 + [2,25 - x_1 (1 - x_2^2)]^2 + [2,625 - x_1 (1 - x_2^3)]^2$	[3]		0	(3, $\frac{1}{2}$)
VI: $100 (x_2 - x_1^2)^2 + (1 - x_1)^2 + 90 (x_4 - x_3^2)^2 + (1 - x_3)^3 + 10,1 (x_2 - 1)^2 + (x_4 - 1)^2 + 19,8 (x_2 - 1) \times (x_4 - 1)$	[4] ¹⁾	(-3, -1, -3, -1)	0	(1, 1, 1, 1)
VII: $(x_1 + 10x_2) + 5 (x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10 (x_1 - x_4)^4$	[5] ²⁾	(-3, -1, 0, 1)	0	(0, 0, 0, 0)
VIII: $(c_1^2 - x_2)^4 + 100 (x_2 - x_3)^6 + \operatorname{tg}^4 (x_3 - x_4) + x_1^8 + (x_4 - 1)^2$	[6]	(1, 2, 2, 2)	0	[0, 1, 1, (1 $\pm n\pi$)]
IX: функция P задачи 10 из приложения А	{ Ссылка на литературу дана в приложении А	Детали изложены в приложении А; описание функции P см. в гл. 7	Детали изложены в приложении А; описание функции P см. в гл. 7	Детали изложены в приложении А; описание функции P см. в гл. 7
X: функция P задачи 18 из приложения А				

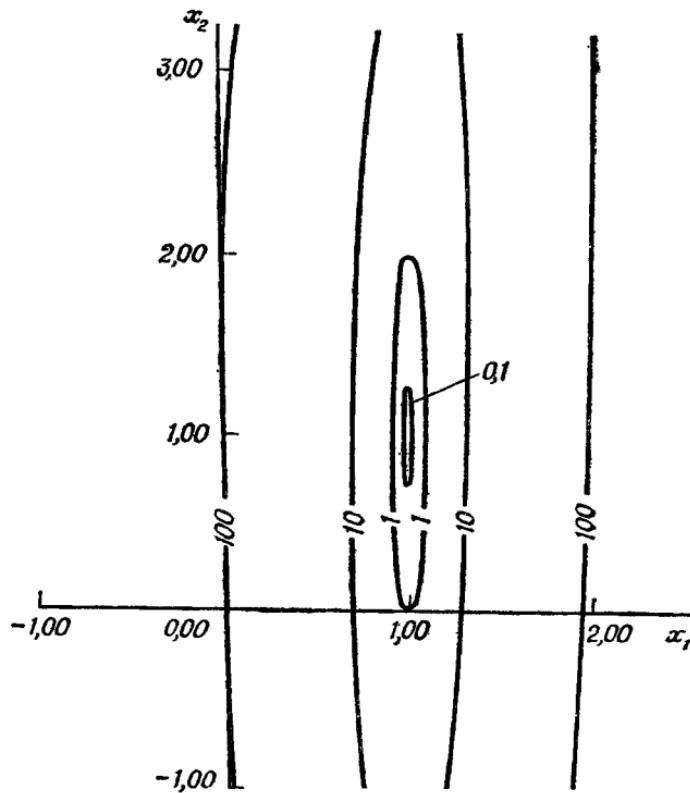
¹⁾ Функция имеет несколько локальных минимумов; это обстоятельство может вызвать преждевременное окончание процесса.²⁾ Заметим, что матрица Гессе этой функции в точке минимума сингулярна.



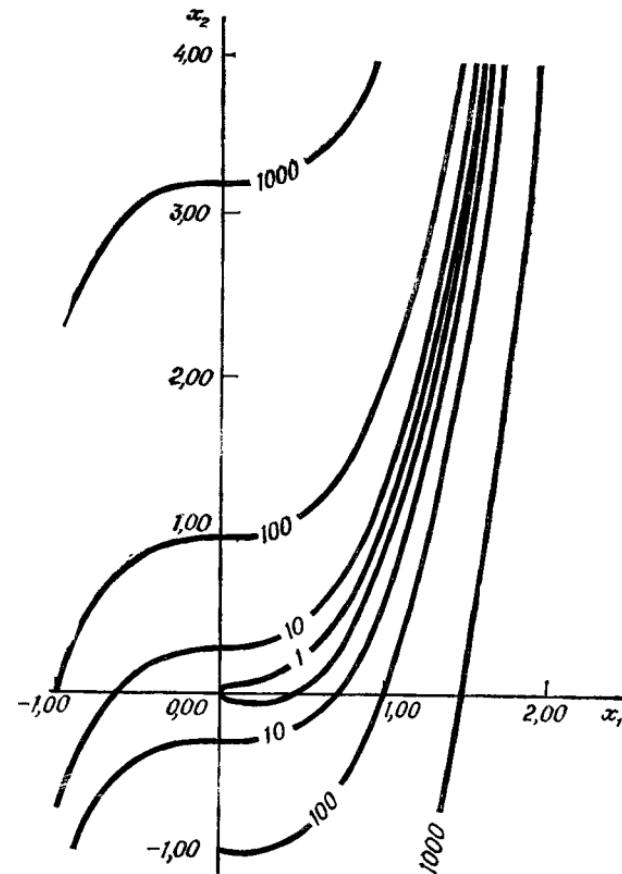
Ф и г. 5.2.1. Функция I: $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$.



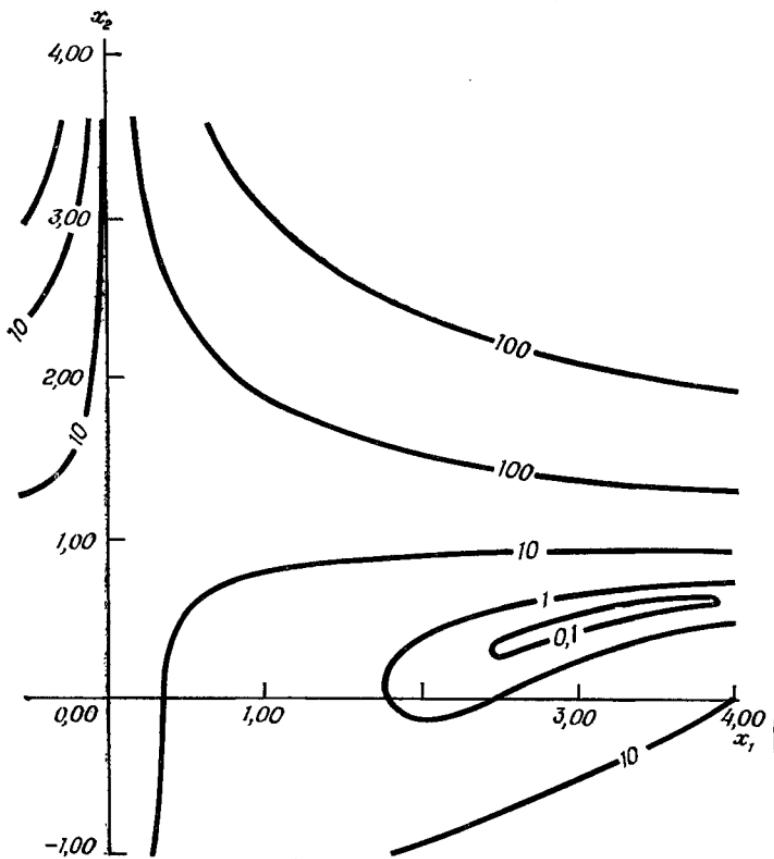
Ф и г. 5.2.2. Функция II: $f(\mathbf{x}) = (x_2 - x_1^2)^2 + (1 - x_1)^2$.



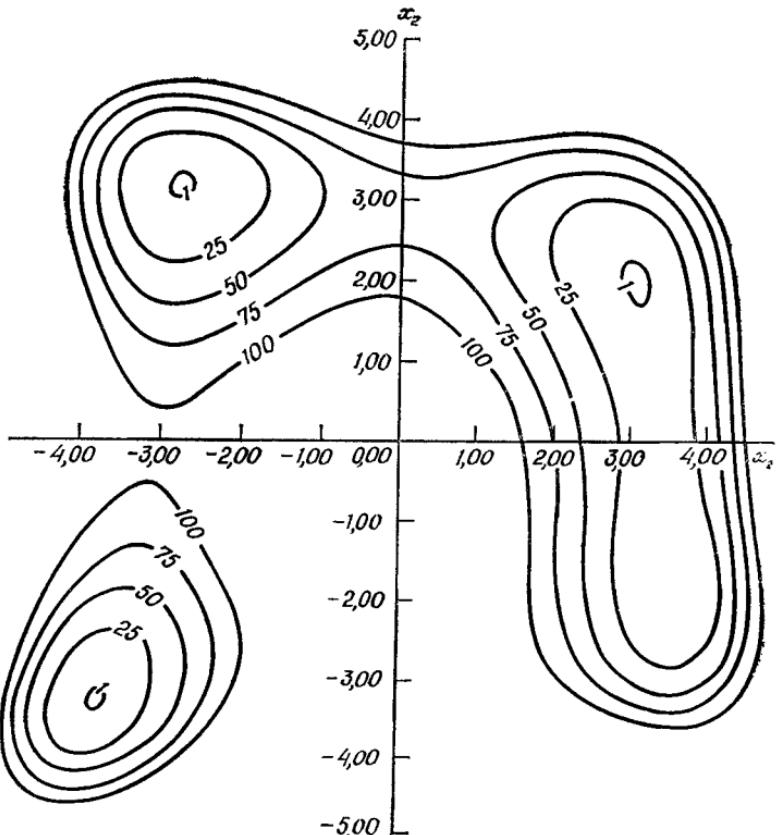
Ф и г. 5.2.3. Функция III: $f(x) = (x_2 - x_1^2) + 100(1 - x_1)^2$.



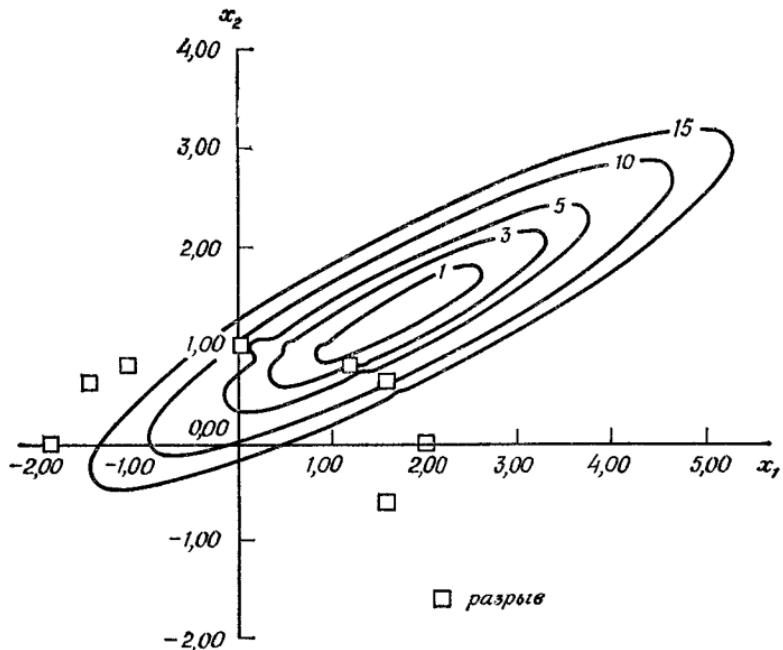
Ф и г. 5.2.4. Функция IV: $f(x) = 100(x_2 - x_1^3)^2 + (1 - x_1)^2$



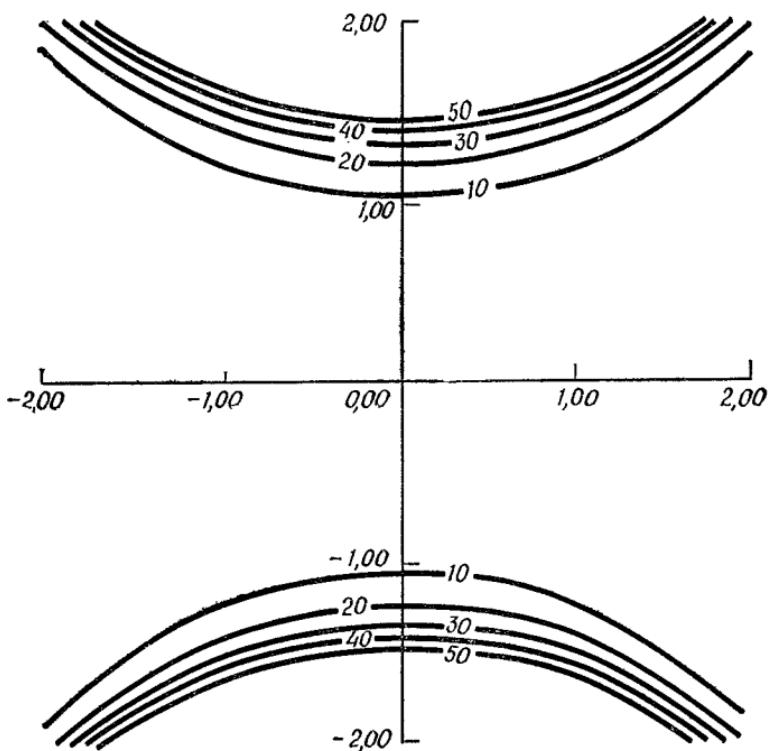
Ф и г 5.2.5. Функция V : $f(x) = [1,5 - x_1(1 - x_2)]^2 + [2,25 - x_1(1 - x_2^2)]^2 + [2,625 - x_1(1 - x_2^3)]^2$.



Ф и г. 5.2.6. Функция 28 приложения А: $f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2$.



Ф и г. 5.2.7. Функция 31 приложения А: $f(x) = (x_1 - 2)^2 + (x_2 - 1)^2 + \frac{0,04}{-(x_1^2/4) - x_2^2 + 1} + \frac{1}{0,2} (x_1 - 2x_2 + 1)^2$.



Ф и г. 5.2.8. Функция 33 приложения А: $f(x) = e^{-(x_1^2-x_2^2)} [2x_1^2 + 3x_2^2]$.

прямой $x_1 = 1$; у функции IV крутая впадина вдоль кривой $x_2 = x_1^3$; функция Биля V также имеет узкую закругленную впадину, приближающуюся к прямой $x_2 = 1$. Функции IX и X представляют собой штрафные функции (они будут описаны в гл. 7).

5.3. ОЦЕНИВАНИЕ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ОТСУТСТВИИ ОГРАНИЧЕНИЙ

Сначала рассмотрим некоторые опубликованные работы, а затем обратимся к неопубликованным результатам и сравним большинство алгоритмов, изложенных в гл. 3 и 4, на единой основе.

В работе Леона [7] среди других алгоритмов сравниваются в частности следующие:

- 1) Метод Дэвидона в том виде, как его запрограммировал Стивенс [8].
- 2) Программа Баера [9]. Эта программа генерирует последовательность ограниченных минимумов и проводит между ними интерполяцию в окрестности $f(x)$.
- 3) Партан-метод наискорейшего спуска.
- 4) Модифицированный партан-метод [10].
- 5) Итерационный партан-метод.

При этом используются первые пять целевых функций, приведенных в табл. 5. 2. 1. Соответствующая таблица результатов (табл. 5.3.1) содержит несколько начальных векторов, конечный вектор x , значение $f(x)$ в конечном x и относительное машинное время, необходимое для выполнения минимизации. Все значения, за исключением относительных значений времени, округлены до третьего десятичного знака. Партан-метод наискорейшего спуска и итерационный партан-метод не включены в табл. 5.3.1, поскольку эти алгоритмы работали очень плохо в том смысле, что в значительной части тестовых решений конечные векторы x отклонялись от известных векторов x^* , соответствующих минимуму, более чем на 5%, а во многих случаях эти отклонения составляли свыше 100%.

Из табл. 5.3.1 следует, что процедура Дэвидона — Флетчера — Паузэлла представляется наилучшей в смысле общего метода нелинейного программирования при отсутствии ограничений в отношении получения правильных значений вектора x в точках минимума целевой функции. Процедура Баера приводила к ошибкам при минимизации целевой функции Биля, если процесс начинался с некоторых определенных начальных векторов, тогда как модифицированный партан-метод оказался неспособным справиться с функцией Розенброка. Поскольку относительные значения времени выполнения алгоритмов были приблизительно одинаковы, метод

Таблица 5.3.1

Результаты минимизации пяти целевых функций различными методами

Функция	Начальные значения $x_1 \ x_2 \ f(x)$	Дэвидон		Баер		Модифицированный партан-метод	
		конечные значения $x_1 \ x_2 \ f(x)$	отно- си- тель- ное вре- мя ¹⁾	конечные значения $x_1 \ x_2 \ f(x)$	отно- си- тель- ное вре- мя ¹⁾	конечные значения $x_1 \ x_2 \ f(x)$	отно- си- тель- ное вре- мя ¹⁾
I	—1,200 1,000 24,200	1,000 1,000 $3,4 \cdot 10^{-11}$	1,00	1,000 1,000 $2,7 \cdot 10^{-12}$	0,50	0,939 0,882 $3,7 \cdot 10^{-2}$	0,50
	—2,000 —2,000 3609,0	1,000 1,000 $6,5 \cdot 10^{-11}$	0,64	1,000 1,000 $4,3 \cdot 10^{-12}$	1,71	1,002 1,003 $4,0 \cdot 10^{-6}$	0,35
	5,621 —3,635 $1,24 \cdot 10^5$	1,000 1,000 $9,3 \cdot 10^{-13}$	0,78	1,000 1,000 $1,0 \cdot 10^{-12}$	1,62	1,000 1,000 $9,6 \cdot 10^{-3}$	0,41
	—0,221 0,639 36,320	1,000 1,000 $7,5 \cdot 10^{-15}$	0,66	1,000 1,000 $1,7 \cdot 10^{-12}$	0,92	0,940 0,883 $3,6 \cdot 10^{-3}$	0,50
II	—2,000 —2,000 45,000	1,000 1,000 $1,4 \cdot 10^{-12}$	0,51	1,000 1,000 $7,2 \cdot 10^{-16}$	1,77	0,999 1,000 $1,0 \cdot 10^{-6}$	0,34
	0,803 —0,251 0,840	1,000 1,000 $5,8 \cdot 10^{-15}$	0,26	1,000 1,000 $9,7 \cdot 10^{-11}$	1,11	0,991 0,979 $9,2 \cdot 10^{-5}$	0,19
	0,211 3,505 12,600	1,000 1,000 $1,5 \cdot 10^{-14}$	0,30	1,000 1,000 $2,9 \cdot 10^{-15}$	0,29	1,000 1,000 $2,2 \cdot 10^{-16}$	0,27
III	2,000 —2,000 136,00	1,000 1,000 $5,5 \cdot 10^{-12}$	0,44	1,000 1,000 $5,6 \cdot 10^{-15}$	0,50	1,000 1,000 $4,7 \cdot 10^{-11}$	0,17
	1,992 —3,222 150,10	1,000 1,000 $1,1 \cdot 10^{-12}$	0,34	1,000 1,000 $2,6 \cdot 10^{-10}$	0,20	1,000 1,000 $1,1 \cdot 10^{-12}$	0,16
	1,986 5,227 98,86	1,000 1,000 $2,7 \cdot 10^{-12}$	0,23	1,000 1,000 $2,2 \cdot 10^{-12}$	0,32	1,000 1,000 $1,0 \cdot 10^{-12}$	0,12
IV	1,200 —2,000 1389,8	1,000 1,000 $2,3 \cdot 10^{-13}$	0,95	1,000 1,000 $2,1 \cdot 10^{-13}$	1,05	1,000 0,999 $9,8 \cdot 10^{-7}$	0,94
	0,248 —3,082 964,54	1,000 1,000 $5,6 \cdot 10^{-13}$	0,92	1,000 1,000 $1,0 \cdot 10^{-8}$	0,90	1,000 1,000 $9,9 \cdot 10^{-13}$	1,16
	—1,200 —1,000 57,840	1,000 1,000 $9,2 \cdot 10^{-13}$	1,00	1,000 1,000 $5,2 \cdot 10^{-14}$	0,98	1,000 1,000 $1,1 \cdot 10^{-13}$	1,15
V	0,000 0,000 14,200	3,000 0,500 $1,8 \cdot 10^{-15}$	0,54	3,000 0,500 $2,3 \cdot 10^{-12}$	0,37	2,999 0,500 $3,3 \cdot 10^{-7}$	1,07
	8,000 0,200 81,700	3,000 0,500 $1,0 \cdot 10^{-11}$	0,67	2,045 0,075 $5,4 \cdot 10^{-1}$	1,21	3,002 0,501 $8,7 \cdot 10^{-7}$	0,40
	5,000 0,800 0,490	3,000 0,500 $2,2 \cdot 10^{-13}$	0,42	3,000 0,500 $3,6 \cdot 10^{-14}$	0,94	3,000 0,500 $3,7 \cdot 10^{-13}$	0,43
	8,000 0,800 2,042	3,000 0,500 $2,8 \cdot 10^{-12}$	0,62	7,996 0,874 $3,8 \cdot 10^{-1}$	0,10	3,000 0,500 $1 \cdot 10^0$	0,37

¹⁾ Время, отнесенное ко времени минимизации функции I методом Дэвидона, начиная с точки $\mathbf{x}(0) = [-1,2 \ 1,0]^T$.

Сравнение нескольких методов миними

Функция и $x^{(0)}$	Метод Дэвидона — Флетчера — Пауэлла							Метод Ньютона		
	Уортман [11]			Флетчер—Пауэлл [12]			(Уортман [11])			
	$f(x^*)$	этапы	число вычислений функции	$f(x^*)$	этапы	число вычислений функции	$f(x^*)$	этапы	число вычислений функции	
I ($-1, 2, 1, 0$)	$1 \cdot 10^{-12}$	23	120	$1 \cdot 10^{-8}$	18	—	$3 \cdot 10^{-13}$	17	98	
I ($-2, 547, 1, 489$)	$6 \cdot 10^{-16}$	16	87				$6 \cdot 10^{-16}$	20	130	
II ($0,211, 3,505$)	$1 \cdot 10^{-11}$	9	36				$2 \cdot 10^{-18}$	8	28	
IV ($-1, 2, 1, 0$)	$3 \cdot 10^{-12}$	21	120				$4 \cdot 10^{-15}$	25	127	
IV ($0, 248, -3, 082$)	$7 \cdot 10^{-15}$	23	115				$2 \cdot 10^{-19}$	16	71	
V ($0, 0$)	$7 \cdot 10^{-14}$	12	39				$2 \cdot 10^{-17}$	9	31	
V ($8, 0, 8$)	$7 \cdot 10^{-14}$	21	119				0,25 ¹⁾	400	1922	
VI ($3, -1, 0, 1$)	$5 \cdot 10^{-10}$	32	149	$2,5 \cdot 10^{-8}$	6	—	$8 \cdot 10^{-12}$	26	95	

1) Матрица Гессе не является положительно определенной, поэтому был использован поиск

Дэвидона — Флетчера — Пауэлла представляется предпочтительнее других.

В другом исследовании, проведенном Уортманом [11], применялись методы Дэвидона — Флетчера — Пауэлла и Ньютона совместно с методами поиска Хука — Дживса и Пауэлла (описанными в гл. 4); последние включались в качестве подпрограмм полных машинных программ решения. Были проверены функции I, II, IV и VII табл. 5.2.1. В табл. 5.3.2 приведены результаты Уортмана, а также некоторые данные, полученные другими авторами, в виде количества необходимых вычислений функции. (В табл. 7.2.1 представлены некоторые соответствующие результаты для функции IX табл. 5.2.1.) В табл. 5.3.2 в столбце «этапы» приведены числа успешных направлений поиска (т. е. число минимумов, найденных при осуществлении линейных поисков по этим направлениям). В процессе линейного поиска осуществлялись последовательный поиск и подгонка кривой.

Используя процедуру Хука и Дживса, Уортман немного модифицировал ее по сравнению с описанной в разд. 4.1 так, что в случае успешного шага следующее значение x_i берется в два раза большим предыдущего x_i , тогда как при неудаче следующее значение x_i берется в два раза меньшим предыдущего x_i . Для поиска по методу Хука и Дживса столбец «этапы» относится к исследованию поиску и, возможно, продвижению по образцу, тогда как в случае поиска по методу Пауэлла этап включает обнаружение минимума

Таблица 5.3.2

зации при отсутствии ограничений

Метод Дэвиса — Свена — Кемпти (Флетчер [13])			Метод Хука и Дживса (Уортман [11])			Метод Паузелла (Уортман [11])			Метод Паузелла [12]		
$f(\mathbf{x}^*)$	этапы	число вычислений функции	$f(\mathbf{x}^*)$	этапы	число вычислений функции	$f(\mathbf{x}^*)$	этапы	число вычислений функции	$f(\mathbf{x}^*)$	этапы	число вычислений функции
$1,5 \cdot 10^{-12}$	21	187	$2 \cdot 10^{-7}$	87	353	$4 \cdot 10^{-14}$	15	192	$7 \cdot 10^{-10}$	13	151
			$2 \cdot 10^{-9}$	142	588	$9 \cdot 10^{-14}$	17	216			
			$2 \cdot 10^{-10}$	40	168	$6 \cdot 10^{-16}$	7	66			
			$3 \cdot 10^{-9}$	106	458	$5 \cdot 10^{-13}$	17	202			
			$3 \cdot 10^{-7}$	44	181	$9 \cdot 10^{-16}$	24	333			
			$1 \cdot 10^{-11}$	43	181	$1 \cdot 10^{-15}$	7	77			
			$4 \cdot 10^{-10}$	56	230	$1 \cdot 10^{-17}$	9	119			
$1,6 \cdot 10^{-11}$	12	196	$2 \cdot 10^{-8}$	100	769	$3 \cdot 10^{-9}$	19	89	$5 \cdot 10^{-9}$	16	235

методом наискорейшего спуска; $\mathbf{x}_{\text{конечное}} = [7,94 \ 0,858]^T$.

с помощью линейного поиска в каждом из N независимых координатных направлений и, возможно, в смешанном направлении.

Все конечные значения элементов вектора независимых переменных отклонялись от истинных значений \mathbf{x}^* не больше, чем на 10^{-5} (10^{-4} для процедуры Хука — Дживса), за исключением особо отмеченных случаев. Приведенное здесь число вычислений функции несколько больше, чем в случае, если бы удалить одну из частей программы, работающей как машинная подпрограмма, и использовать ее отдельно. Тем не менее алгоритм Дэвидона — Флетчера — Паузелла оказывается почти таким же удовлетворительным, как и метод Ньютона. (Когда в этих примерах при использовании алгоритма Ньютона матрица Гессе оказывалась не положительно определенной, матрица направлений заменялась единичной матрицей, т. е. в качестве направления поиска выбиралось направление отрицательного градиента.)

В табл. 5.3.3 для нескольких алгоритмов приводится число этапов, приведенных в сообщении Пирсона, необходимых, чтобы уменьшить $f(\mathbf{x})$ ниже уровня 10^{-18} . Эти числа в сущности не являются количеством проведенных вычислений функции, поскольку на каждом этапе для обнаружения минимума $f(\mathbf{x})$ осуществлялся одномерный поиск Фибоначчи. Изменяя критерий окончания процесса одномерного поиска, получим различное число итераций; следовательно, данные, приведенные в таблице, должны рассматриваться скорее как относительные, чем как абсолютные. Столбец

Таблица 5.3.3

24

Число этапов для уменьшения $f(x^*)$ до значений, меньших 10^{-13} [15]

Алгоритм	Раздел	Функция I по табл. 5.2.1 (функция Розенброка)		Функция VI по табл. 5.2.1 (функция Вуда)		Функция IX по табл. 5.2.1 ¹⁾		Функция X по табл. 5.2.1 ²⁾	
		без перезадания	с перезаданием	без перезадания	с перезаданием	без перезадания	с перезаданием	без перезадания	с перезаданием
Пирсона № 2	3.4.3	18	31	36	47	27 (62)	22 (60)	134 (221)	98 (187)
Пирсона № 3	3.4.3	21	37	46	47	33 (67)	22 (54)	136 (246)	100 (168)
Дэвидона — Флетчера — Паузелла	3.4.2	19	35	40	49	27 (60)	22 (56)	406 (500)	97 (169)
Ньютона	3.2.1	12	—	23	—	11 (22)	—	30 (48)	—
Флетчера — Ривса	3.3.2	—	16	—	30	—	34 (F)	—	>489 (F)
Проективный									
Ньютона (Пирсона)	3.4.3	36	21	58	55	31 (20)	67 (54)	166 (230)	113 (186)
Проективный (Заутендайка)	3.3.4	—	42	—	65	(26)	(70)	(120)	(211)

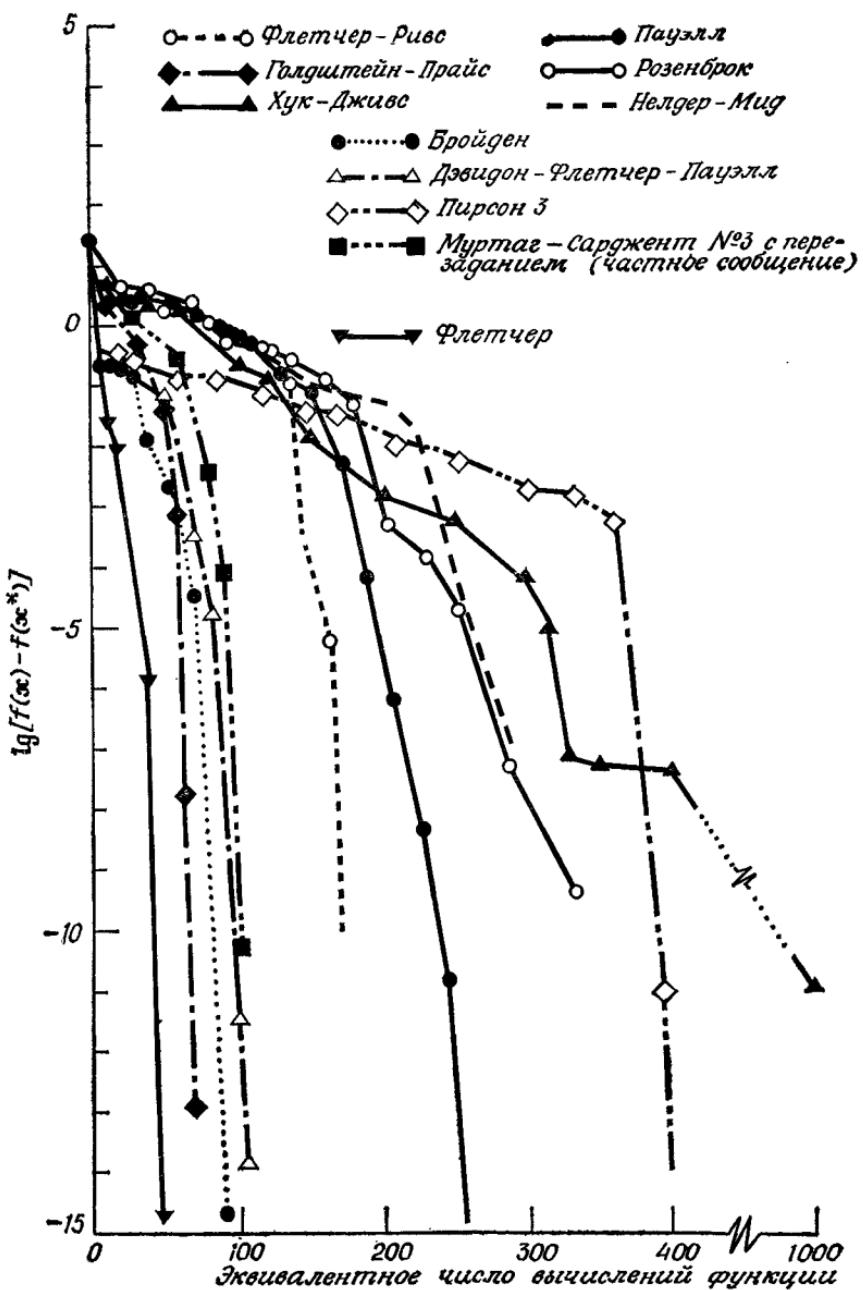
1) Числа без скобок для $r = 1$; со скобками — для $r = 2.44 \cdot 10^{-4}$ (см. гл. 7).2) Числа без скобок для $r = 1$; со скобками — для $r = 0.0625$ (см. гл. 7).

F — неудача.

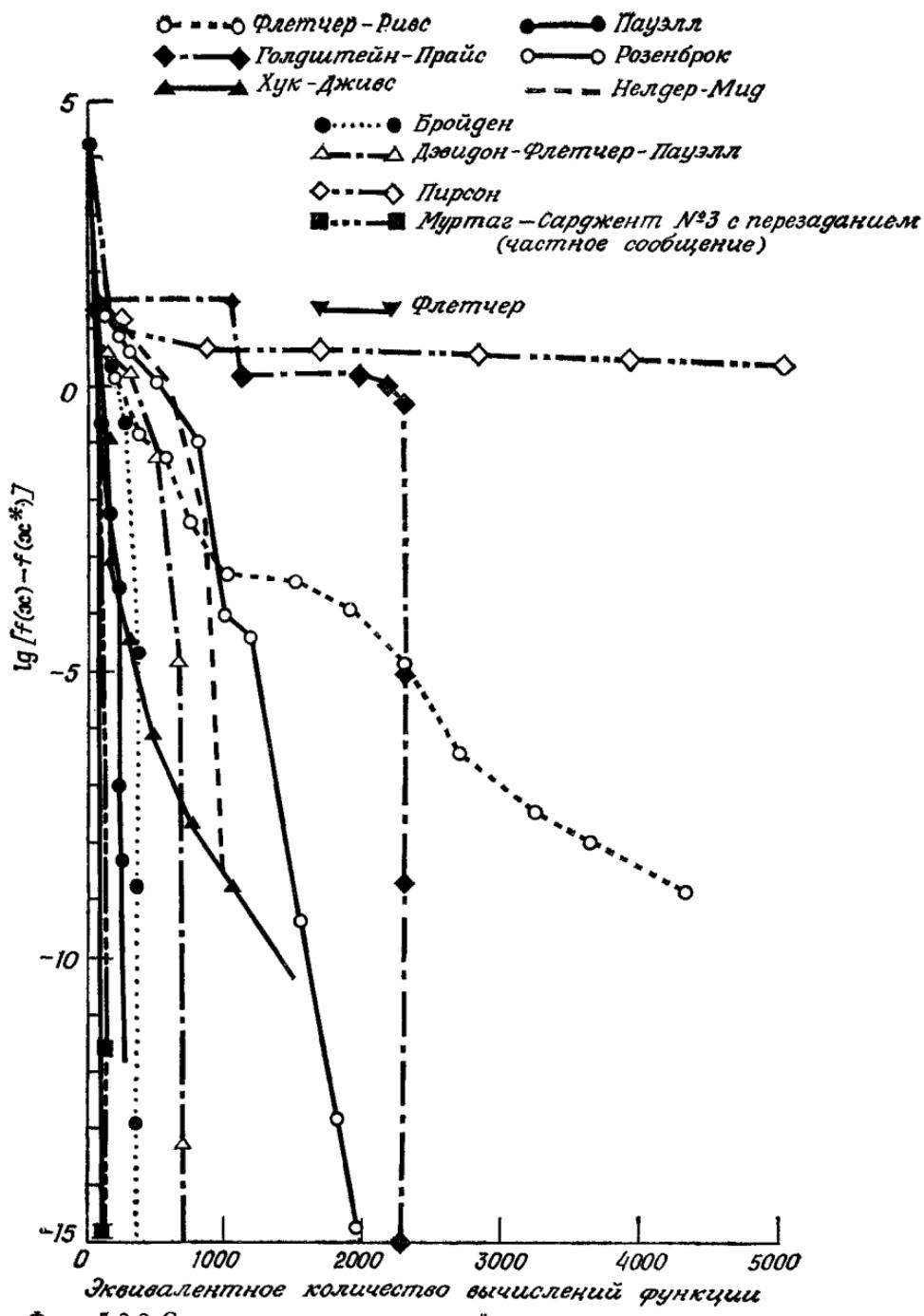
«с перезаданием» указывает, что после $(n + 1)$ -го шага величина η^{n+1} задавалась заново равной $R^{(0)}$ и алгоритм повторялся; в методе Флетчера — Ривса и проективном методе начальная точка всегда должна перезадаваться.

Метод Ньютона всегда оказывался лучшим; что же касается остальных методов, то, за исключением метода Флетчера — Ривса, который не сработал в некоторых тестах, все они оказались приблизительно равной эффективности. Из табл. 5.3.3 видно, что перезадание η дает существенный выигрыш для целевых функций, полученных с использованием понятия штрафной функции, ибо, как только x приближается к активным ограничениям, матрица Гессе для штрафной функции становится очень «плохой». Для других функций задание заново η не приводит к каким-нибудь преимуществам. Мак-Кормик и Пирсон [14] пришли к выводу, что использование методов переменной метрики с перезаданием является наиболее подходящим на ранних этапах оптимизации, тогда как отказ от перезадания предпочтителен на более поздних этапах, когда x оказывается настолько близок к x^* , что эффективными становятся свойства сопряженности алгоритмов переменной метрики.

Рассмотрение приведенных выше результатов тестов дает лишь частичную картину относительной эффективности алгоритмов нелинейного программирования при отсутствии ограничений, поскольку в каждом исследовании использовались различные методы одномерного поиска, разные критерии окончания процесса и различные методы подсчета количества вычислений функции. Более желательным является проведение оценивания с использованием единого набора стандартов и тестовых задач. Для уменьшения объема данных, которые могут быть получены при решении тестовых задач, до разумно достижимого предела мы воспользуемся тем фактом, что все алгоритмы уменьшают (по крайней мере должны уменьшать) целевую функцию монотонно от этапа к этапу. Так, представляется разумным строить графики $|f(x) - f(x^*)|$, $\|f(x)\|$ или, что еще лучше, $\lg |f(x) - f(x^*)|$ в зависимости от количества вычислений функции или времени, т. е. кривые типа изображенных на фиг. 5.3.1 и 5.3.2. Можно было бы взять $\|x - x^*\|$ или другой критерий для сравнения, но эти функции не будут уменьшаться монотонно. Такие кривые дают эмпирическую меру скорости сходимости соответствующих алгоритмов. Заметим, что если заканчивать процесс оптимизации всегда при заданной разумной степени точности в $|f(x) - f(x^*)|$, скажем 10^{-10} , то можно использовать соответствующее этому уровню число вычислений функции или время в качестве единственной меры эффективности алгоритма. Хотя абсолютные значения количества вычислений функции или времени могут не иметь большого смысла, относительное ранжирование алгоритмов по этим величинам оказывается разумным количественным механизмом сравнения.



Ф и г. 5.3.1. Сравнение алгоритмов нелинейного программирования при отсутствии ограничений для функции Розенброка.



Ф и г. 5.3.2. Сравнение алгоритмов нелинейного программирования при отсутствии ограничений для функции Вуда.

Попытки сравнения различных алгоритмов на основе количества вычислений функции могут быть менее удовлетворительными, чем сравнение с использованием времени решения, особенно если последнее может быть определено для одной и той же ЭВМ, используются общие подпрограммы и задачи решаются до одной и той же степени точности. В качестве примера приведем (см. фиг. 5.3.1) одну из причин, почему использование количества вычислений функции в качестве критерия является не очень подходящим. Эквивалентное число вычислений функции в методе Пауэлла представляет собой просто число вычислений целевой функции; в методе Флетчера — Ривса оно представляет собой количество обращений к функциональной подпрограмме, составленной так, что один градиентный вызов (содержащий две производные) эквивалентен одному вызову целевой функции; в алгоритме Голдштейна — Прайса вычисление двух компонент градиента также считается эквивалентным одному вычислению целевой функции. На фиг. 5.3.2 один градиентный вызов эквивалентен двум обращениям к вычислению целевой функции. Однако компоненты градиента в алгоритме Голдштейна — Прайса вычисляются гораздо чаще, чем в алгоритме Флетчера — Ривса, что приводит к некоторому относительному искажению при сравнении по этому критерию указанных алгоритмов из-за произвольного определения эквивалентного количества вычислений функции. Именно поэтому время решения до заданного значения $\lg |f(x) - f(x^*)|$, а не эквивалентное количество вычислений функции было выбрано в качестве единственной наиболее подходящей меры эффективности.

Там, где это было удобно, применялись метод золотого сечения и поиск ДСК — Пауэлла. Эти методы описаны в разд. 2.6. В приложении Б содержатся машинные программы этих процедур и ряда алгоритмов. Критерии окончания процедур одномерного поиска были одними и теми же. Окончание процедур основных алгоритмов тоже проводилось по одним и тем же критериям. Для оценивания времени решения все тестовые задачи решались на одной и той же машине ЭВМ CDC 6600. Время печатания, периферической обработки информации и время работы на пульте исключены из времени, приведенного в табл. 5.3.4, так что эти данные действительно представляют собой число секунд, необходимых для реализации алгоритмов без какого бы то ни было прерывания процесса.

В табл. 5.3.5 дана оценка алгоритмов на основании данных табл. 5.3.4. Все алгоритмы вместе со своими подпрограммами ранжировались в порядке возрастания времени от номера 1 (самый быстрый) до номера 11. Алгоритм Пирсона № 2 не рассматривался из-за повторяющихся неудач. (Оценки некоторых других менее надежных алгоритмов приводились в гл. 3 и 4.) После ранжирования по каждой задаче результаты были усреднены по всем 11

Таблица 5.3.4

Сравнение времени решения (в секундах) для одиннадцати тестовых задач с помощью одиннадцати алгоритмов

(Задачи из приложения А или табл. 5.2.1)												
Алгоритм	Раздел	2 (I ¹)	VI ¹)	26 (VII ¹)	28	29	30	31	32	33	34	35 (V ¹)
Бройдена	3.4.1											
ДСК		0,013 ²⁾	0,089	0,060	0,040	0,031	0,024	0,010	0,125	0,015	0,037	0,013
3		0,016 ²⁾	0,092	0,046	0,012	0,018	0,021	0,006	0,109	0,016	0,132	0,012
Дэвидона — Флетчера — Пауэлла	3.4.2											
ДСК		0,014 ²⁾	0,088	0,104	0,027	0,099	0,027	0,008	0,052	0,015	0,056	0,015
3		0,016 ²⁾	0,113	0,088	0,010	0,046	0,025	0,019	0,121	0,016	0,134	0,010
Пирсона № 2	3.4.3											
ДСК		>1,00	F	F	>1,00	F	F	0,008	F	0,022	F	F
3		>1,00	F	F	>1,00	F	F	0,011	F	F	—	F
Пирсона № 3	3.4.3											
ДСК		0,058	2,900 (0,314) ³⁾	>100	0,028	0,059	0,046	0,008	>10	0,024	>10	0,016
3		0,051	3,012 (1,199) ³⁾	2,763	0,011	0,106	0,065	0,010	>10	0,016	>10	0,013

(Задачи из приложения А или табл. 5.2.1)

Алгоритм	Раздел	2 (I ¹)	VI ¹	26 (VII ¹)	28	29	30	31	32	33	34	35 (V ¹)
Голдштейна — Прайса	3.4.6											
ДСК		0,016 ²)	0,079	0,100	0,052	0,026	0,021	0,096	0,066	0,035	0,158	0,014
З		0,013	0,089	0,057	0,014	0,044	0,029	0,145	0,149	0,020	0,097	0,016
ГП		—	—	—	0,081	—	—	—	0,094	—	0,043	—
Флетчера — Ривса	3.3.2											
ДСК		0,027	25,0 ⁴)	0,144	0,055	0,069	0,019	0,015	F	0,013 ⁵)	0,064	0,012
З		—	0,330	2,330	—	—	—	—	F	—	0,975	—
Флетчера	3.4.5	0,022	0,024	0,050	0,006	0,018	0,012	0,004	0,050	0,010	0,028	0,011
Хука — Дживса	4.1	0,100	0,152	0,067	0,008	0,167	0,018	0,012	0,127 ⁶)	0,054	F	0,009
Нелдерса — Мида	4.2	0,097	0,154	0,072	0,024	0,036	0,148	?	0,685 ⁶)	0,019	0,531	0,014
Розенброка	4.3	0,058	0,378	0,097	0,035	0,125	0,131	0,025	0,202 ⁶)	0,027	0,148	0,025
Паузэлла	4.4											
ДСК		0,035	0,041	0,084	0,006	0,014	0,014	0,005	0,178	0,017	0,025	0,012
З		0,050	0,098	0,174	0,021	0,106	0,028	0,015	0,857	0,018	0,108	0,027

¹) Римские цифры означают номер задачи по табл. 5.2.1.²) Время увеличивается на 70—100%, если поиск проходит по изогнутой впадине.³) С перезаданием.⁴) С перезаданием на ($n + 1$)-м этапе.⁵) Работает только с масштабированием (см. машинную программу в приложении Б).⁶) Закончил решение на плоском плато до достижения минимума.⁷) Поиск стал уходить в направлении глобального минимума на —со.

Обозначения: ДСК — метод ДСК; З — метод золотого сечения; F — неудача; ГП — метод Голдштейна — Прайса.

задачам, причем предполагалось, что задачи имеют равный вес. Конечно, можно было бы считать, что более трудные задачи должны иметь больший вес, чем более легкие, но это не было сделано. Тем не менее возможное различие при применении двух подпрограмм одномерного поиска было исключено путем слияния двух ранжировок в одну общую для каждого алгоритма.

Интересным результатом такого ранжирования было то, что алгоритмы оказалось возможным объединить в группы. В табл.

Таблица 5.3.5

**Оценка алгоритмов нелинейного программирования
при отсутствии ограничений, исходя из времени решения**

Классификация	Алгоритм
Наилучший	{ Флетчера Дэвидона — Флетчера — Пауэлла Бройдена Пауэлла
Хороший	Голдштейна — Прайса Нелдера — Мида
Благоприятный	Розенброка Флетчера — Ривса Хука — Дживса
Ненадежный	Пирсона № 3 Пирсона № 2

5.3.5 приведены в порядке уменьшения ранга (увеличения времени решения) различные алгоритмы. Эти алгоритмы разбиты на группы в соответствии с их качественными характеристиками: «наилучший», «хороший» и «благоприятный». Ввиду ограниченного набора тестовых задач такая классификация представляется более разумной, чем непрерывная классификация. Внутри каждой группы алгоритмы расположены в порядке вычисленного ранга.

Как и следовало ожидать, алгоритмы поиска работают медленнее, чем алгоритмы, использующие производные, но особенно интересно то, что алгоритм Пауэлла имеет высокую эффективность. Алгоритмы Бройдена и Пауэлла в более сложных задачах (таких, как задачи 26 и 32), по-видимому, работают лучше, чем алгоритм Дэвидона — Флетчера — Пауэлла, тогда как для некоторых более простых задач имело место обратное.

Пять алгоритмов попадают в класс «благоприятных» алгоритмов. Каждый из этих алгоритмов обычно требовал больше времени, чем алгоритмы из «наилучшего» класса. Кроме того, эти алгоритмы менее надежны, чем алгоритмы из «наилучшей» группы, поскольку они могут закончиться преждевременно или быть неэффективными

из-за чрезмерно избыточного использования машинного времени. В методах, использующих производные, медленные колебания в процедуре поиска указывают на то, что матрица направлений становится почти сингулярной. Включение в алгоритм подходящей процедуры перезадания начальной точки, восстанавливающей матрицу направлений до положительно определенной формы, возможно, может улучшить некоторые методы переменной метрики.

Алгоритмы Хука — Дживса, Нелдера — Мида и Розенброка оказались довольно слабыми при решении задачи статистической оценки (задача 32). В этой задаче в окрестности минимума целевая функция была нечувствительна к изменениям переменных, т. е. в этой области целевая функция имела вид плато.

Все тестовые задачи, результаты решения которых использованы при построении табл. 5.3.5, содержали всего лишь несколько переменных. Очень важно выяснить, как ведут себя эти алгоритмы с увеличением размерности задачи. К сожалению, в этом отношении имеется очень мало данных. Для выяснения этого вопроса Флетчер и Паузелл (1963 г.) предложили тестовую функцию для задачи минимизации, содержащую регулируемое число переменных:

$$f(\mathbf{x}) = \sum_{i=1}^n \left[E_i - \sum_{j=1}^n (A_{ij} \sin x_j + B_{ij} \cos x_j) \right]^2. \quad (5.3.1)$$

Функция (5.3.1) соответствует решению совместной системы трансцендентных уравнений

$$\sum_{j=1}^n (A_{ij} \sin x_j + B_{ij} \cos x_j) = E_i, \quad i = 1, \dots, n,$$

полученному минимизацией суммы квадратов разностей. Коэффициенты A_{ij} и B_{ij} генерируются в виде псевдослучайных целых величин с равномерным (прямоугольным) распределением вероятностей на интервалах $-100 \leq A_{ij} \leq 100$ и $-100 \leq B_{ij} \leq 100$. Значения E_i были промоделированы путем генерирования различных по величине групп из x_j от $n = 5$ до $n = 100$ как псевдослучайных действительных чисел в интервале от $-\pi$ до π .

Целевая функция $f(\mathbf{x})$ минимизировалась по отношению к вектору $\mathbf{x} = [x_1, \dots, x_n]^T$, начиная с вектора $\mathbf{x}^{(0)} = [x_1 + 0,1\delta_1, \dots, x_n + 0,1\delta_n]^T$, где δ_i — случайные числа на интервале от $-\pi$ до π . В табл. 5.3.6 приведено эквивалентное количество вычислений функции, необходимое для уменьшения изменения каждого x_j до величины, меньшей чем 10^{-4} . Для разных начальных векторов могут быть достигнуты различные минимумы, так как, конечно, такая функция, как (5.3.1), имеет много минимумов; с другой стороны, при некоторых начальных векторах локальный минимум может быть и не найден.

Хотя табл. 5.3.6. может дать только качественное представление относительно эффективности соответствующих алгоритмов при решении задач большой размерности, создается впечатление, что методы Дэвидона — Флетчера — Пауэлла и Пауэлла (1964 г.), грубо говоря, эквивалентны и явно лучше других методов, за исключением метода Ривса.

Таблица 5.3.6

Влияние размерности на алгоритм минимизации
(Средние значения эквивалентного количества вычислений функции¹⁾,
взятые из работы Бокса [16])

Алгоритм	Раздел	Число переменных		
		5	10	20
Флетчера — Ривса	3.3.2	286	1925	8150
Дэвидона — Флетчера — Пауэлла	3.4.2	126	358	1910
Нелдера — Мида	4.2	241	890	9760
Розенброка	4.3	426	1258	7770
Дэвиса — Свенна — Кемпи	4.3	298	1530	6450
Пауэлла (1964)	4.4	103	399	1863
Пауэлла (1965)	[17]	22	35	56

1) Эквивалентное количество вычислений функции для методов, использующих производные, означает, что одно вычисление функции плюс n вычислений производных считаются как $(n + 1)$ эквивалентных вычислений функции.

чением метода Пауэлла (1965 г.). Высокая эффективность алгоритма Пауэлла (1965 г.) при решении задач типа (5.3.1) приводит к вопросу: насколько эффективны рассмотренные алгоритмы при применении их к тому или иному специальному типу задачи минимизации суммы квадратов и, в частности, каковы они по сравнению с классическим методом наименьших квадратов Гаусса?

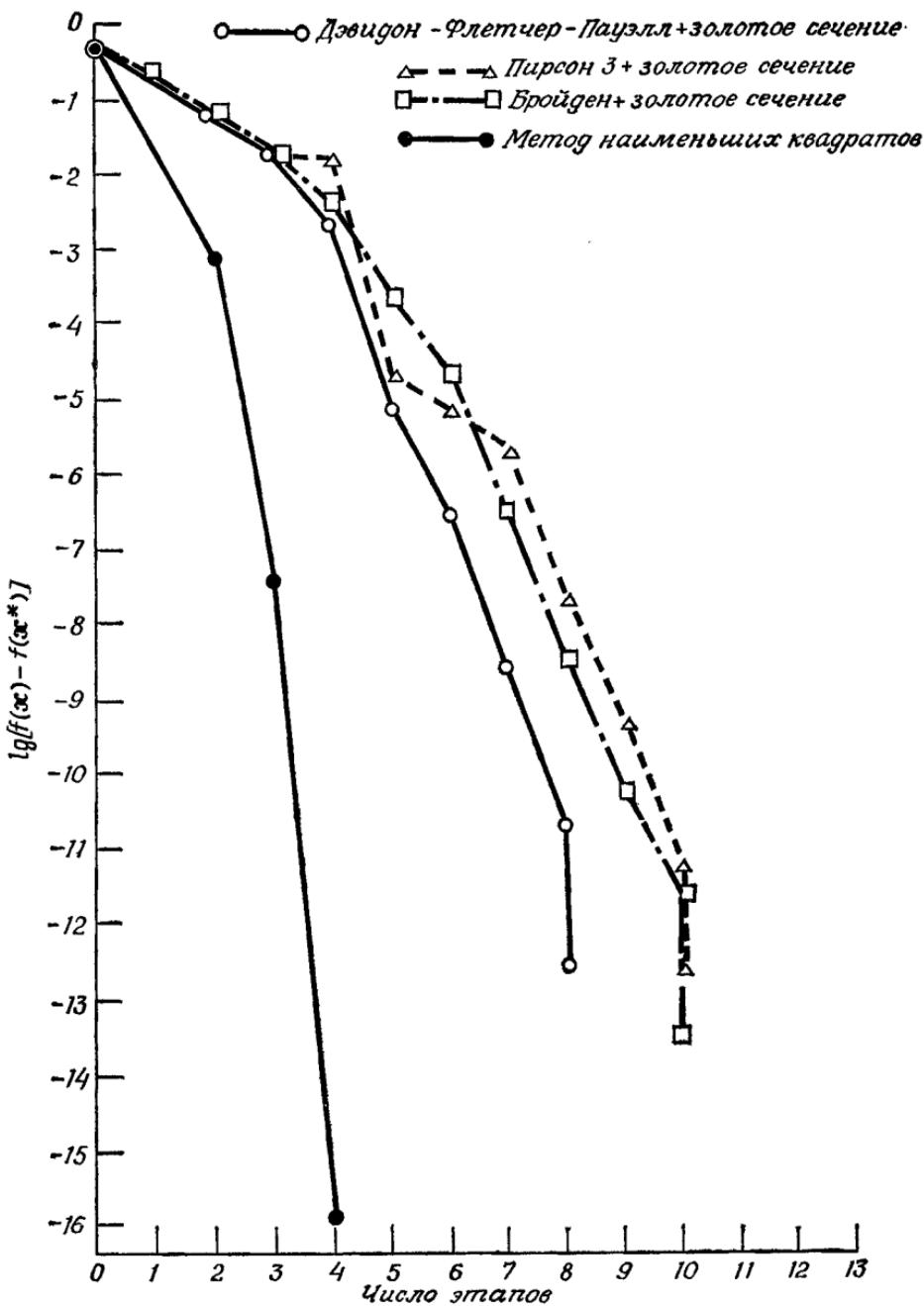
Для целевой функции, представляющей собой сумму квадратов

$$f(\mathbf{x}) = \sum_{i=1}^q \phi_i^2(\mathbf{x}) = \boldsymbol{\phi}^T(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}),$$

$$\nabla f(\mathbf{x}) = 2 \mathbf{J}^T(\mathbf{x}) \boldsymbol{\phi}(\mathbf{x}),$$

где $\mathbf{J}(\mathbf{x})$ — матрица Якоби размерности $q \times n$:

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} -\frac{\partial \phi_1}{\partial x_1} & \dots & \frac{\partial \phi_1}{\partial x_n} \\ \vdots & & \vdots \\ -\frac{\partial \phi_q}{\partial x_1} & \dots & \frac{\partial \phi_q}{\partial x_n} \end{bmatrix},$$



Ф. г. 5.3.3. Сравнение различных методов при использовании их для решения задач, сводящихся к задаче минимизации суммы квадратов (задача 35 из приложения А).

а ϕ — вектор-столбец из q функций. Если градиент в точке $\mathbf{x}^{(k+1)}$ аппроксимируется по формуле

$$\nabla f(\mathbf{x}^{(k+1)}) \approx 2\mathbf{J}^T(\mathbf{x}^{(k)})\phi(\mathbf{x}^{(k+1)}), \quad (5.3.2)$$

а $\phi(\mathbf{x}^{(k+1)})$ в свою очередь аппроксимируется членом первого порядка разложения в ряд Тейлора

$$\phi(\mathbf{x}^{(k+1)}) = \phi(\mathbf{x}^{(k)}) + \mathbf{J}(\mathbf{x}^{(k)}) (\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}), \quad (5.3.3)$$

то подстановка (5.3.3) в (5.3.2) дает приближенное выражение для градиента целевой функции в новой точке. Необходимые условия существования минимума в точке $\mathbf{x}^{(k+1)}$, состоящие в том, что $\nabla f(\mathbf{x}^{(k+1)}) = 0$, приводят к вычислению $\mathbf{x}^{(k+1)}$ по методу наименьших квадратов Гаусса:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - [\mathbf{J}^T(\mathbf{x}^{(k)}) \mathbf{J}(\mathbf{x}^{(k)})]^{-1} \mathbf{J}^T(\mathbf{x}^{(k)}) \phi(\mathbf{x}^{(k)}). \quad (5.3.4)$$

(Модификация Левенберга — Маркуардта, отмеченная в разд. 3.2, заключается в замене $[\mathbf{J}^T(\mathbf{x}^{(k)}) \mathbf{J}(\mathbf{x}^{(k)})]$ на $[\mathbf{J}^T(\mathbf{x}^{(k)}) \mathbf{J}(\mathbf{x}^{(k)}) + \beta I]$.) При использовании метода Гаусса или метода Маркуардта требуется такой алгоритм обращения матриц, который гарантировал бы положительную определенность получаемой обратной матрицы при положительно определенной исходной матрице. К сожалению, такая процедура отсутствует во многих широко используемых машинных вычислительных методах программирования.

Алгоритм Гаусса состоит в следующем. Начиная из точки $\mathbf{x}^{(k)}$:

- 1) определить $\mathbf{J}(\mathbf{x}^{(k)})$ и вычислить $[\mathbf{J}^T(\mathbf{x}^{(k)}) \mathbf{J}(\mathbf{x}^{(k)})]^{-1}$ и $\phi(\mathbf{x}^{(k)})$;
- 2) по формуле (5.3.4) вычислить $\mathbf{x}^{(k+1)}$;
- 3) повторять переход от шага 2 к шагу 1, пока не будет удовлетворен критерий окончания процесса.

На фиг. 5.3.3 сравнивается метод наименьших квадратов Гаусса с другими методами минимизации при решении систем уравнений, но это не слишком показательно, поскольку один этап в методе наименьших квадратов не то же, что один этап в других методах. Аналогично трудно сравнивать количество вычислений функции и количество вычислений производных. В табл. 5.3.7 приводится время решения (на ЭВМ CDC 6600) семи задач методом наименьших квадратов, а также лучшее время, полученное при использовании алгоритмов табл. 5.3.4. Задача 29 из приложения А не была решена из-за возникшей неопределенности матрицы $(\mathbf{J}^T \mathbf{J})$. (В таблицу включена и статистическая задача — задача 32 из приложения А.)

Из табл. 5.3.7 видно, что для задач этого специального типа метод Гаусса может конкурировать с лучшим из алгоритмов

Таблица 5.3.7

Сравнение метода наименьших квадратов с другими методами (лучшими по времени решения) для целевых функций, представляющих собой сумму квадратов

Задача (приложение А)	Время решения ме- тодом наименьших квадратов, с	Лучшее время решения другими методами, с
2	0,012	0,013 (метод Брайдена)
26	0,058	0,046 (метод Брайдена)
28	0,013	0,006 (метод Пауэлла, Флетчера)
30	0,015	0,012 (метод Флетчера)
32	0,061	0,050 (метод Флетчера)
34	0,020	0,025 (метод Пауэлла)
35	0,017	0,010 (метод Дэвидона — Флетчера — Пауэлла)

Таблица 5.3.8

Результаты применения различных алгоритмов при решении задач минимизации суммы квадратов

Группа	Пол- ный ранг	Алгоритм	Средняя сумма рангов	Число попаданий в классы	
				II	III
A	1	Гаусса	26	0	0
	2	Маркуардта (разд. 3.2)	32	0	0
B	3	Брайдена (подразд. 3.4.1) ¹⁾	73	2	2
	4	(подразд. 3.4.6)	77	3	1
C	5	Алгоритм Дэвидона — Флетчера — Пауэлла (подразд. 3.4.2)	97	3	3

1) Здесь собственные значения могут устанавливаться на некоторых абсолютных уровнях, как это следует из процедуры Гринштадта (разд. 3.2).

Таблица 5.3.9

Влияние количества оцениваемых параметров на работу алгоритма при решении задачи минимизации суммы квадратов

Алгоритм	Ранг для указанного числа параметров				
	3	5	6	8	10
Гаусса	1	2	1	2	1
Маркуардта (разд. 3.2)	2	1	2	1	2
Брайдена (подразд. 3.4.1)	3	3	3	5	5
Дэвидона — Флетчера — Пауэлла (подразд. 3.4.2)	5	5	5	4	3

нелинейного программирования при отсутствии ограничений. В других работах [18—21], где рассматривается специальный случай решения системы совместных нелинейных уравнений путем минимизации невязок, показывается, что методы переменной метрики не всегда являются надежными.

При исследовании алгоритмов вычисления нелинейных оценок в статистических задачах Бард [22] проверил пять существенно нелинейных моделей, содержащих от трех до десяти параметров, охарактеризовал работу каждого алгоритма для каждой из задач (ранг 1 определялся как лучший), а затем классифицировал результаты следующим образом:

класс I: лучшие результаты;

класс II: не такие хорошие результаты, как в классе 1, но вполне приемлемые.

класс III: неприемлемые результаты и (или) отсутствие сходимости.

В табл. 5.3.8 приведена средняя сумма рангов для различных алгоритмов и число попаданий каждого алгоритма в классы II и III. С увеличением числа оцениваемых параметров ранги в столбце 2 табл. 5.3.8 сдвигаются ненамного, как это видно из табл. 5.3.9. Хотя методы Гаусса и Маркуардта с увеличением числа параметров все еще представляются лучшими, метод Дэвидона — Флетчера — Пауэлла становится относительно более предпочтительным.

ЛИТЕРАТУРА

1. Rosenbrock H. H., *Computer J.*, 3, 175 (1960).
2. Whittle B. F., Holst W. R., Spring Joint Computer Conf., Washington, D. C., 1964.
3. Beale E. M. L., On an Iterative Method of Finding a Local Minimum of a More Than One Variable, Princeton Univ. Stat. Techn. Res. Group Techn. Rept. 25, Nov. 1958.
4. Wood C. F., Westinghouse Res. Labs.
5. Powell M. J. D., *Computer J.*, 7, 155 (1964).
6. Cragg E. E., Levy A. V., Rice Univ. Aero-Astronautics Rept. 58, Houston, Tex., 1969.
7. Leon A., A Comparison Among Eight Known Optimizing Procedures, in: Recent Advances in Optimization Techniques, Lavi A., Vogl T. P., eds., Wiley, Inc. N. Y., 1966.
8. Stevens D. F., Instructions for the Use of VARMINT, Univ. of California, Lawrence Radiation Lab., Berkeley, June 1961.
9. Baer R. M., *Computer J.*, 5, 193 (1962).
10. Doerfler T. E., Partan Minimization by Method of Parallel Tangents, Iowa State Univ., Ames, Iowa, April 1964.
11. Wortman J. D., NLPROG, Ballistic Res. Lab. Mem. Rept. 1958, Aberdeen Proving Grounds, Md., Jan. 1969.
12. Fletcher R., Powell M. J. D., *Computer J.*, 6, 163 (1963).
13. Fletcher R., *Computer J.*, 8, 33 (1965).
14. McCormick G. P., Pearson J. D., Chap. 21 in: Optimization, Fletcher R., ed., Academic Press, Inc., London, 1969.

15. Pearson J. D., *Computer J.*, **13**, 171 (1969).
16. Box M. J., *Computer J.*, **9**, 67 (1966).
17. Powell M. J. D., *Computer J.*, **7**, 303 (1965).
18. Box M. J., *Computer J.*, **9**, 67 (1966).
19. Barnes J. G. P., *Computer J.*, **8**, 66 (1965).
20. Powell M. J. D., *Computer J.*, **7**, 303 (1965).
21. Spendley W., Chap. 16 in: Optimization, Fletcher R., ed., Academic Press, Inc., London, 1969.
22. Bard Y., Comparison of Gradient Methods for the Solution of Nonlinear Parameter Estimation Problems, IBM N. Y. Sci. Center Rept. 320-2955, 1968; *SIAM J. Numerical Anal.*, **7**, 157 (1970).

Ч а с т ь III

МЕТОДЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ



В области нелинейного программирования с ограничениями¹⁾ методы решения менее разработаны по сравнению с областью нелинейного программирования, охватывающего круг задач, в которых ограничения отсутствуют. При решении задач нелинейного программирования с ограничениями встречаются гораздо большие трудности, чем при решении сопоставимых (т. е. обладающих приблизительно такой же размерностью и такой же «степенью» нелинейности) задач безусловной оптимизации по той причине, что искомое решение должно подчиняться дополнительному требованию, а именно удовлетворять фигурирующим в задаче ограничивающим условиям. Опубликованные вычислительные процедуры решения задачи нелинейного программирования, содержащей ограничения, в большинстве своем опираются на один из следующих подходов:

1. Распространение аппарата линейного программирования на условия нелинейного программирования путем использования процедуры последовательной (повторяемой определенное число раз) линейной аппроксимации.

2. Преобразование задачи нелинейного программирования с ограничениями в эквивалентную ей последовательность задач безусловной оптимизации путем введения в рассмотрение штрафных функций.

3. Использование скользящих допусков, позволяющих оперировать в процессе решения задачи оптимизации как с допустимыми, так и с недопустимыми (но близкими к допустимым) векторами в пространстве решений.

Основные характеристики и потенциальные возможности ряда методов решения задач нелинейного программирования при наличии ограничений в систематизированном виде представлены в

¹⁾ Задача нелинейного программирования при наличии ограничений в общей постановке представлена соотношениями (2.2.1) — (2.2.3).

Таблица А

Характеристики некоторых методов нелинейного программирования при наличии ограничений

Структурные характеристики решаемых задач и характеристики метода	Глава 6 Методы линейной аппроксимации								Глава 7 Методы преобразования целевой функции (методы штрафных функций)			Глава 8 Метод скользящего допуска	
	Метод аппроксимирующего программирования (подразд. 6.1.2)	ПОП (подразд. 6.1.2)	Метод НЛП (разд. 6.2)	Метод Розена (проекции градиента, подразд. 6.3.1)	ОГМОП (подразд. 6.3.2)	Проективный вариант метода Дэвидона — Флетчера — Пауэлла (подразд. 6.3.3)	Метод допустимых направлений (разд. 6.4)	Метод приведенного градиента (разд. 6.5)	Метод Розенброка (подразд. 7.1.2)	Метод Дэвидона — МБП (подразд. 7.1.3)	МПБМ (разд. 7.2)		
Целевая функция Позволяет манипулировать с ограничениями в виде равенств	Л и НЛ Нет	Л и НЛ Нет	Л и НЛ Л и НЛ	Л и НЛ Л	Л и НЛ Л и НЛ	Л и НЛ Л	Л и НЛ Л	Л и НЛ Нет	Л и НЛ Л и НЛ	Л и НЛ Нет	Л и НЛ Нет	Л и НЛ Л	Л и НЛ Л и НЛ
Позволяет манипулировать с ограничениями в виде неравенств	НЛ ¹⁾	Л и НЛ	Л и НЛ	Л и НЛ ²⁾	Л и НЛ ²⁾	Л и НЛ	Л и НЛ	Л и НЛ	Л и НЛ	Л и НЛ	Л и НЛ	Л и НЛ	Л и НЛ
Начальная точка Позволяет решать невыпуклые задачи	Д Да	Д и Н Да	Д и Н Да	Д Да	Д и Н Да	Д Да	Д Да	Д и Н Да	Внутр. Да	Внутр. Да	Внутр. Да	Д и Н Да	
Скорость сходимости Промежуточные решения	Низкая Д	Низкая Д	Низкая Д и Н	Высокая ³⁾ Д	Низкая Д и Н	Средняя Д	Высокая Д	Средняя Д	Средняя Д	Средняя Д	Высокая Внутр.	Низкая Д и Н	

¹⁾ Неэффективен при решении задач с линейными ограничениями.²⁾ Весьма неэффективен при решении задач с нелинейными ограничениями.³⁾ Применим только для решения задач с линейными ограничениями.

Обозначения: Л — линейная (ые), НЛ — нелинейная (ые), Д — допустимая (ые), Н — не является допустимой (не являются допустимыми).

табл. А; более подробно эти методы обсуждаются в гл. 6—8. Следует отметить, что в табл. А включены только те методы, которые удовлетворяют ряду конкретных требований; при этом основными критериями являются:

- 1) относительно высокая эффективность рассматриваемого метода как инструмента, позволяющего получать численные решения задач нелинейного программирования;
- 2) простота логики построения вычислительных процедур;
- 3) наличие машинных программ, позволяющих реализовать рассматриваемый алгоритм на ЭВМ.

Глава 6

ПРОЦЕДУРЫ МИНИМИЗАЦИИ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ: МЕТОДЫ ЛИНЕЙНОЙ АППРОКСИМАЦИИ



Поскольку методы линейного программирования успешно применяются для решения задач большой размерности при ограничениях, записанных как в виде равенств, так и в виде неравенств (разумеется, линейной структуры), способ решения задач нелинейного программирования с помощью процедуры линеаризации, реализуемой по надлежащим образом построенной схеме итерационного процесса, относится к числу наиболее очевидных. Два из рассматриваемых ниже методов (возможно, являющихся в некотором отношении наиболее эффективными) предполагают линеаризацию только ограничений, оставляя целевую функцию нелинейной на всех этапах минимизации. Один из этих методов основан на использовании пресектирующих матриц, позволяющих проектировать вектор, который определяет направление поиска оптимальной точки, на подпространство, связанное с некоторой модифицированной совокупностью активных ограничений; во втором методе используется идея приведенного градиента, определение которого дается в разд. 6.5.

Линейная аппроксимация фигурирующих в задаче [см. соотношения (2.2.1) — (2.2.3)] нелинейных функций достигается путем замены этих функций членами первого порядка в соответствующих разложениях в ряд Тейлора в окрестности рассматриваемой точки $\mathbf{x}^{(k)}$. В результате повторяющегося процесса линеаризации нелинейных функций путем разложения их в ряд Тейлора в окрестности каждого промежуточного решения образуется последовательность $\mathbf{x}^{(0)}, \mathbf{x}^1, \dots, \mathbf{x}^{(k)}$, которая при определенных условиях сходится к оптимальному решению \mathbf{x}^* исходной задачи нелинейного программирования. Чтобы пояснить сущность данного метода, обратимся к задаче нелинейного программирования в ее общей постановке [см. соотношения (2.2.1) — (2.2.3)], т. е. рассмотрим следующую задачу:

минимизировать $f(\mathbf{x}), \mathbf{x} \in E^n$,

при ограничениях

$$\begin{aligned} h_i(\mathbf{x}) &= 0, \quad i = 1, \dots, m, \\ g_i(\mathbf{x}) &\geq 0, \quad i = m + 1, \dots, p. \end{aligned} \tag{6.0.1}$$

Эта задача может быть модифицирована путем замены каждой из фигурирующих в (6.0.1) нелинейных функций двумя первыми членами в соответствующем разложении в ряд Тейлора в окрестности $\mathbf{x}^{(k)}$. В результате вместо (6.0.1) нам потребуется

$$\text{минимизировать } f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}), \quad \mathbf{x} \in E^n,$$

при ограничениях

$$\begin{aligned} h_i(\mathbf{x}^{(k)}) + \nabla^T h_i(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) &= 0, \quad i = 1, \dots, m, \\ g_i(\mathbf{x}^{(k)}) + \nabla^T g_i(\mathbf{x}^{(k)})(\mathbf{x} - \mathbf{x}^{(k)}) &\geq 0, \quad i = m+1, \dots, p. \end{aligned} \quad (6.0.2)$$

Поскольку $f(\mathbf{x}^{(k)})$, $\nabla f(\mathbf{x}^{(k)})$, $h_i(\mathbf{x}^{(k)})$, $\nabla h_i(\mathbf{x}^{(k)})$, $g_i(\mathbf{x}^{(k)})$ и $\nabla g_i(\mathbf{x}^{(k)})$ являются либо постоянными векторами, либо скалярными константами (значения которых вычисляются в точке $\mathbf{x}^{(k)}$), задача (6.0.2) представляет собой задачу линейного программирования.

Чтобы проиллюстрировать метод линеаризации задачи нелинейного программирования в окрестности некоторой точки $\mathbf{x}^{(k)}$, рассмотрим следующую частную задачу:

$$\text{минимизировать } f(\mathbf{x}) = 4x_1 - x_2^2 - 12, \quad \mathbf{x} \in E^n, \quad \text{при ограничениях}$$

$$h_1(\mathbf{x}) = 25 - x_1^2 - x_2^2 = 0,$$

$$g_2(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

Пусть $\mathbf{x}^{(k)} = [2 \ 4]^T$. Замена фигурирующих в данном примере нелинейных функций на аппроксимирующие их (в окрестности $\mathbf{x}^{(k)}$) линейные выражения приводит к следующей задаче линейного программирования:

$$\text{минимизировать } \tilde{f}(\mathbf{x}^{(k)}) = 4x_1 - 8x_2 + 4, \quad \mathbf{x} \in E^n, \quad \text{при ограничениях}$$

$$\tilde{h}_1(\mathbf{x}^{(k)}) = 45 - 4x_1 - 8x_2 = 0,$$

$$\tilde{g}_2(\mathbf{x}^{(k)}) = -14 + 6x_1 + 2x_2 \geq 0,$$

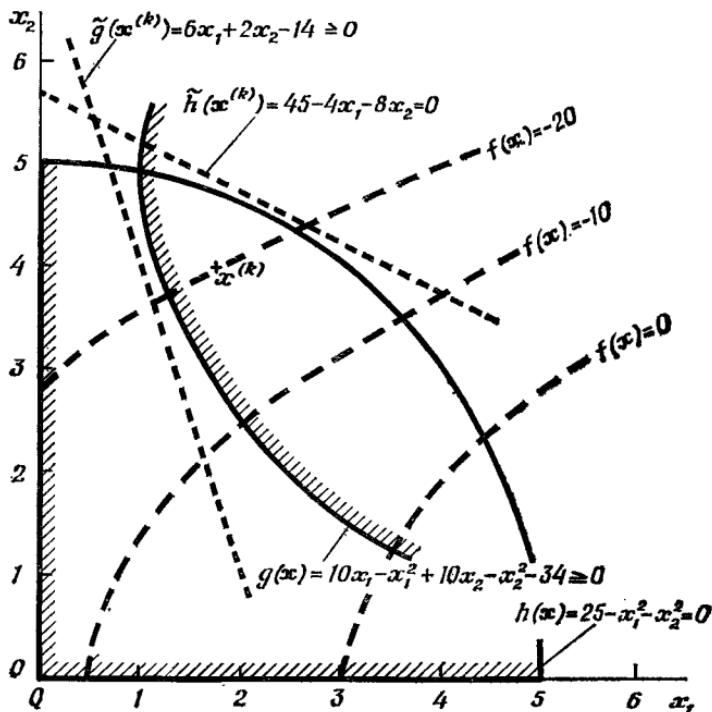
$$\tilde{g}_3(\mathbf{x}^{(k)}) = x_1 \geq 0,$$

$$\tilde{g}_4(\mathbf{x}^{(k)}) = x_2 \geq 0.$$

Чтобы оттенить то обстоятельство, что вместо исходных функций взяты их линейные аппроксимации, мы воспользовались знаком (\sim). Метод оптимизации, требующий линеаризации только

ограничений и не предполагающий линеаризации целевой функции, иллюстрируется геометрически на фиг. 6.0.1.

Таблица 6.0.1, содержащая перечень алгоритмов, описание которых дается в настоящей главе, может служить ориентиром при ознакомлении с основными характеристиками указанных ал-



Фиг. 6.0.1. Пример линеаризации ограничений в окрестности точки, заданной вектором $x^{(k)} = [2 \ 4]^T$.

горитмов. Некоторые другие алгоритмы, являющиеся по сравнению с алгоритмами, приведенными в табл. 6.0.1, менее эффективными или оказывающиеся достаточно эффективными лишь применительно к задачам специального типа (как, например, к задачам выпуклого программирования или к задачам, в которых фигурируют только линейные ограничения), а также алгоритмы, характеристики которых пока не опубликованы, здесь не описываются, хотя ссылки на них можно найти в конце данной главы. Методы, описанию которых посвящена настоящая глава, образуют своего рода «ряд», начинающийся методами с плохими характеристиками и заканчивающийся методами, которые можно отнести к разряду отличных.

Сходимость к искомому решению задачи (6.0.1) гарантируется, если выполняются следующие условия:

1) $f(\mathbf{x}), h_1(\mathbf{x}), \dots, h_m(\mathbf{x}), g_{m+1}(\mathbf{x}), \dots, g_p(\mathbf{x})$ являются непрерывными и дифференцируемыми функциями;

2) функция $f(\mathbf{x})$ выпуклая, а сумма $\sum h_i^2(\mathbf{x})$ выпуклая в R ;

3) $g_{m+1}(\mathbf{x}), \dots, g_p(\mathbf{x})$ представляют собой вогнутые функции;

4) допустимая область непуста, т. е. действительно существуют такие значения составляющих вектора \mathbf{x} , для которых удовлетворяются все условия, определяемые ограничениями задачи (т. е. задача имеет решение);

5) множество R замкнутое и выпуклое;

6) существует $\varepsilon > 0$, такое, что $|h_j(\mathbf{x})| \leq \varepsilon$ ($j = 1, \dots, m$) и $g_i(\mathbf{x}) \geq -\varepsilon$ ($i = m+1, \dots, p$), т. е. фигурирующие в условиях функции ограничены.

Таблица 6.0.1

Характеристики методов линейной аппроксимации

Методы	Раз- дел	Целевая функция	Огра- ничи- ния в виде равенств	Огра- ничи- ния в виде неравенств	Начальный вектор ¹⁾
<i>Аппроксимирующее линейное программирование</i>					
МАП	6.1.1	Л и НЛ	Л и НЛ	Л и НЛ	Д
ПОП	6.1.2	Л и НЛ	—	Л и НЛ	Д и Н
НЛП	6.2	Л и НЛ	Л и НЛ	Л и НЛ	Д и Н
<i>Проективные</i>					
Розена (проекции гра- дiente)	6.3.1	Л и НЛ	Л ²⁾	Л ²⁾	Д
Обобщенного градиентно- го поиска	6.3.2	Л и НЛ	Л и НЛ	Л и НЛ	Д и Н
Дэвидона — Голдфарба — Дэвиса	6.3.3	Л и НЛ	Л	Л и НЛ	Д
Заутендайка (допусти- мых направлений)	6.4	Л и НЛ	—	Л и НЛ	Д
<i>Приведенного градиента</i>					
Обобщенного приведен- ного градиента	6.5	Л и НЛ	Л и НЛ	Л и НЛ	Д и Н

1) Внутренняя точка при необходимости без труда находится путем минимизации суммы квадратов нарушенных ограничений в виде неравенств.

2) Существующие в настоящее время машинные программы обеспечивают лишь решение задач с линейными ограничениями.

Обозначения: Д — допустимый (ая, ые), Н — не являющийся допустимым, Л — линейный (ая, ые), НЛ — нелинейный (ая, ые).

На практике любой из алгоритмов, включенных в табл. 6.0.1, позволяет найти локальный оптимум даже в тех случаях, когда условия 1—6 не выполняются (при этом многое зависит от характера решаемой задачи). Специфические тонкости большинства из описанных в данной главе алгоритмов будут отмечены в гл. 9.

6.1. АППРОКСИМИРУЮЩЕЕ ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ

6.1.1. МЕТОД АППРОКСИМИРУЮЩЕГО ПРОГРАММИРОВАНИЯ (МАП)

Гриффиц и Стюарт (фирма «Шелл-Ойл») [1] описали метод аппроксимирующего программирования, который основан на многократном применении алгоритмов линейного программирования, обеспечивающем в пределе сходимость последовательности решений промежуточных задач линейного программирования к решению задачи нелинейного программирования (6.0.1). (Такой метод иногда называют *градиентным* методом с *малой длиной шага* в отличие от так называемого *крупношагового* градиентного метода, описание которого дано в разд. 6.3.) После выполнения процедуры линеаризации задача формулируется так:

$$\text{минимизировать } f(\mathbf{x}) - f(\mathbf{x}^{(k)}) = \sum_{j=1}^n \frac{\partial f(\mathbf{x}^{(k)})}{\partial x_j} \Delta x_j^{(k)}$$

при условии, что имеют место ограничения в виде равенств

$$\sum_{j=1}^n \frac{\partial h_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta x_j^{(k)} = -h_i(\mathbf{x}^{(k)}), \quad i = 1, \dots, m, \quad (6.1.1)$$

и ограничения в виде неравенств

$$\sum_{j=1}^n \frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta x_j^{(k)} \geq -g_i(\mathbf{x}^{(k)}), \quad i = m+1, \dots, p,$$

где $\Delta x_j^{(k)} = (x_i - x_i^{(k)})$. Метод аппроксимирующего программирования позволяет учитывать все ограничения, записанные в виде неравенств, и этим отличается от некоторых из рассматриваемых ниже методов, которые оперируют лишь активными ограничениями в виде равенств. [В методе Гласса и Купера [2], который подробно здесь не обсуждается, даже в том случае, когда неравенство, полученное в результате линеаризации исходного ограничения, удовлетворяется при одновременном нарушении соответствующего неравенства в (6.0.1) (и, следовательно, \mathbf{x} выходит за пределы допустимой области), направление поиска можно выбрать правильным и, таким образом, вновь оказаться в пределах допустимой области, если к правой части линеаризованного неравенства прибавить константу.]

Метод Гриффица и Стюарта реализуется по следующей схеме. Пусть $\mathbf{x}^{(k)}$ — допустимая точка в R . Произведем в (6.0.1) замену нелинейных функций их линейными аппроксимациями, найден-

ными в окрестности точки $\mathbf{x}^{(k)}$. Это приводит к соотношениям, имеющим вид (6.1.1) [\equiv (6.0.2)]. Затем решим задачу линейного программирования, представленную этими соотношениями, при следующем добавочном условии:

$$\delta_j^k - |x_j^{(k+1)} - x_j^{(k)}| \geq 0, \quad j = 1, \dots, n, \quad (6.1.2)$$

где $|x_j^{(k+1)} - x_j^{(k)}|$ — абсолютное значение приращения x_j , а $\delta_j^{(k)} > 0$ ($j = 1, \dots, n$) есть малая величина, ограничивающая длину шага при перемещении в том или ином направлении и, таким образом, не позволяющая вектору \mathbf{x} выходить за пределы допустимой области задачи (6.1.1). Решение задачи (6.1.1) с учетом (6.1.2) для разности $\mathbf{x} - \mathbf{x}^{(k)}$ при \mathbf{x} , обозначенном теперь как $\tilde{\mathbf{x}}$, дает

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + (\tilde{\mathbf{x}} - \mathbf{x}^{(k)}).$$

Вычислив $\mathbf{x}^{(k+1)}$, мы многократно повторяем указанную выше операцию при постепенном уменьшении $\delta^{(k)}$ (с тем, чтобы довести отклонения элементов \mathbf{x} до величины, определяемой принятым допуском), стремясь достичь такой ситуации, когда минимизирующая поправка к найденному на предыдущем шаге значению $f(\mathbf{x})$ оказывается меньше некоторого наперед заданного (малого) числа.

В задаче (6.1.1) дополнительные ограничения для \mathbf{x} можно ввести и несколько иным способом. Обозначим $\mathbf{x} - \mathbf{x}^{(k)}$ при $\mathbf{x} \geq \mathbf{x}^{(k)}$ через $\Delta^+ \mathbf{x}^{(k)}$, а при $\mathbf{x} \leq \mathbf{x}^{(k)}$ через $\Delta^- \mathbf{x}^{(k)}$. Тогда, прежде чем реализовать очередную аппроксимирующую процедуру, следует учсть ограничения для допустимых перемещений в пространстве решений, определяемые следующими соотношениями:

$$p_j^{(k)} \Delta^+ x_j^{(k)} + q_j^{(k)} \Delta^- x_j^{(k)} \leq m_j^{(k)}, \quad j = 1, \dots, n, \quad (6.1.3)$$

где

$$p_j^{(k)} = \max \left\{ 1, \frac{m_j^{(k)}}{U_j - x_j^{(k)}} \right\};$$

$$q_j^{(k)} = \max \left\{ 1, \frac{m_j^{(k)}}{x_j^{(k)} - L_j} \right\};$$

$m_j^{(k)}$ — максимально допустимое перемещение вдоль j -й оси координат на k -м шаге;

L_j — нижняя граница для x_j ;

U_j — верхняя граница для x_j .

Если $\Delta^- x_j^{(k)} = 0$, то $p_j^{(k)} \Delta^+ x_j^{(k)} \leq m_j^{(k)}$. Когда значение $x_j^{(k)}$ близко к своему верхнему пределу, т. е. $(U_j - x_j^{(k)}) \approx 0$, то в $\{1, m_j^{(k)}/U_j - x_j^{(k)}\}$ максимальным оказывается член $m_j^{(k)}/U_j - x_j^{(k)}$. Следовательно, $x_j - x_j^{(k)} \leq U_j - x_j^{(k)}$, или $x_j \leq U_j$, т. е. мы имеем гарантию, что

x_j не может превысить свое верхнее предельное значение. Аналогичные рассуждения применительно к условиям, когда $\Delta^+ x^{(k)} = 0$, позволяют убедиться в том, что x_j не может принимать значений, меньших L_j .

Соотношения (6.1.1) и (6.1.3) образуют систему линейных соотношений, которая и подлежит анализу с помощью методов линейного программирования. Первый шаг в процессе поиска решения данной задачи линейного программирования с помощью модифицированного симплексного метода заключается во введении в рассмотрение переменных еще двух категорий, а именно так называемых *ослабляющих* и *искусственных* переменных. С помощью ослабляющих переменных ограничения в виде неравенств преобразуются в ограничения в виде равенств. Используя для обозначения ослабляющих переменных символ u_i , можно записать ограничения, которые в (6.1.1) фигурируют как неравенства, в следующем виде:

$$\sum_{j=1}^n \frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^+ x_j^{(k)} - \sum_{j=1}^n \frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^- x_j^{(k)} - u_i^{(k)} = -g_i(\mathbf{x}^{(k)}),$$

$$i = m + 1, \dots, p,$$

где $u_i^{(k)} \geq 0$. Чтобы преобразовать в равенства ограничения (6.3.1), также используются ослабляющие переменные (обозначим их через v_j); в результате вместо (6.3.1) получим

$$p_j^{(k)} \Delta^+ x_j^{(k)} + g_j^{(k)} \Delta^- x_j^{(k)} + v_j^{(k)} = m_j, \quad j = 1, \dots, n.$$

Искусственные переменные w_i вводятся лишь с той целью, чтобы иметь возможность задать в качестве начальной допустимую точку в случае, когда начальный вектор \mathbf{x} оказывается вне пределов допустимой области. Путем включения искусственных переменных в ограничения, записанные в виде равенств, и в ограничения в виде неравенств получаем

$$\sum_{j=1}^n \frac{\partial h_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^+ x_j^{(k)} - \sum_{j=1}^n \frac{\partial h_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^- x_j^{(k)} + w_i^{(k)} = -h_i(\mathbf{x}^{(k)}),$$

$$i = 1, \dots, m,$$

и

$$\sum_{j=1}^n \frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^+ x_j^{(k)} - \sum_{j=1}^n \frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} \Delta^- x_j^{(k)} - u_i^{(k)} + w_i^{(k)} = -g_i(\mathbf{x}^{(k)}),$$

$$i = m + 1, \dots, p,$$

где $w_i \geq 0$. Искусственные переменные отличны от нуля, если условие, определяемое соответствующим ограничением, не удовлетворяется. Применение модифицированного симплексного метода позволяет обратить каждую из искусственных переменных в нуль путем минимизации Σw_i .

Пока на каждом шаге процесса поиска получаемое решение оказывается допустимым, метод аппроксимирующего программирования «работает» весьма быстро. Однако в случае, когда на каком-либо шаге вектор \mathbf{x} , дающий минимизирующую поправку к значению целевой функции $f(\mathbf{x})$, выходит за пределы допустимой области, процесс в значительной степени замедляется. В ходе линейного программирования вначале стремятся удовлетворить ограничения, не позволяющие вектору \mathbf{x} выходить за пределы допустимой области, а затем пытаются улучшить значение целевой функции. При этом не исключена возможность того, что решением (6.1.1) и (6.1.3) на следующем шаге снова будет такой вектор \mathbf{x} , который, обеспечивая минимизирующую поправку к полученному ранее значению $f(\mathbf{x})$, окажется вне допустимой области. Таким образом, как только сразу несколько ограничений становятся активными, МАП начинает работать весьма медленно. Следует отметить, что, поскольку на каждом этапе минимизации осуществляется полная релинейаризация рассматриваемой задачи, вся ранее полученная информация становится бесполезной; следовательно, МАП может быть применен для решения невыпуклых задач. Промежуточные решения при использовании МАП являются либо допустимыми, либо близкими к допустимым.

Олсон [3] показал, каким образом данные относительно осцилирующего поведения алгоритма на предшествующих этапах процесса поиска могут быть использованы с целью увеличения или уменьшения максимальной длины шага m_i . К числу наиболее широко известных в этой связи правил относятся следующие:

Вариант	На k -м шаге	Ретроспективные данные относительно приращений переменных	Изменение длины шага m_j
I	≥ 4	+, -, +, -, +, -	$m_j^{(k+1)} = \alpha_1 m_j^{(k)}$
		-, -, -, +, +, +	$m_j^{(k+1)} = \alpha_2 m_j^{(k)}$
II	≥ 5	+, -, +, -, +, -	$m_j^{(k+1)} = 0,5 m_j^{(k)}$
	< 8	-, -, -, +, +, +	$m_j^{(k+1)} = 0,8 x_{jb}^{(k)} - x_{ib} $

Здесь α_1 есть коэффициент сжатия (или сокращения), значения которого лежат в интервале от 0,5 до 0,8; α_2 — коэффициент расширения (или растяжения), значения которого лежат в интервале от 1,2 до 1,5; x_{jb} — предельная верхняя или нижняя граница x_j в направлении перемещения на k -м шаге.

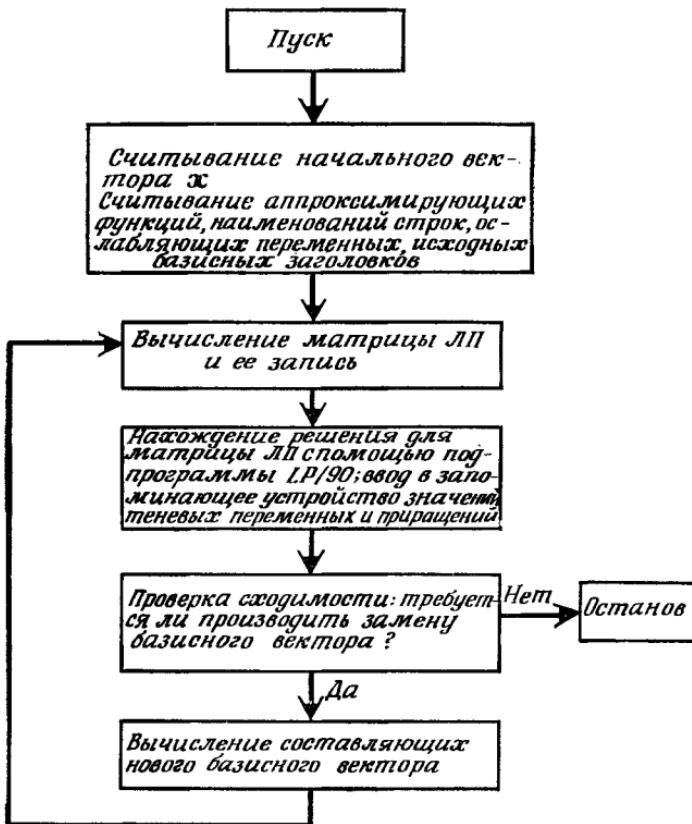
На фиг. 6.1.1 приведена обобщенная блок-схема обработки данных при машинной реализации алгоритма МАП. На основе данного алгоритма Гриффицу и Стюарту удалось решить задачу,

Таблица 6.1.1

Применение МАП при программировании процесса очистки нефти

Этап	1	2	3	...	16	17	18	...	27	28	29
Значение целевой функции	100,0	107,9	116,9	...	122,0	122,1	122,1	...	122,0	122,1	122,0
Число изменений переменных	22	22	21	...	8	5	4	...	4	4	5
Переменные (начальное значение дано в скобках)											
Исходный материал (17,00)	18,00	19,00	20,00	...	21,67	21,67	21,67	...	21,69	21,69	21,70
Температура (920,0)	930,0	925,1	924,9	...	915,4	915,4	915,4	...	916,4	917,2	915,7
Высокооктановый бензин (104,0)	92,8	104,3	108,1	...	103,4	103,5	103,9	...	103,8	103,7	103,8
Обычный бензин (156,0)	172,8	172,8	173,0	...	162,2	162,7	162,1	...	161,1	160,4	161,5
Время, мин											
Запись матрицы	10,4	5,6	4,1	...	4,3	4,2	4,1	...	4,2	4,5	4,4
LP/90	3,1	2,1	2,8	...	2,0	2,0	1,9	...	4,2	2,1	3,4

содержащую примерно 30 переменных, при числе нелинейных ограничений, превышающем 100. Олсон провел анализ затрат машинного времени (применительно к возможностям электронно-вычислитель-



Ф и г. 6.1.1. Блок-схема вычислительного процесса для алгоритма МАП.

ной машины IBM 7094), расходуемого при реализации МАП в интервале между записью матрицы и фазой линейного программирования (см. блок-схему 6.1.1). При этом в нелинейных функциях решаемой задачи фигурировали 27 переменных, а матрица линейного программирования содержала 205 строк. Суммарное время для 29 шагов процесса поиска составило 229 мин. Заметим, что метод Гриффица и Стюарта оказывается относительно малоэффективным при решении задач оптимизации без ограничений (по сравнению с другими методами, предназначенными для решения этих задач) и при решении задач линейного и квадратичного программирования.

Пример 6.1.1. Аппроксимирующее программирование

Метод Гриффица и Стюарта можно проиллюстрировать на следующем примере. Пусть требуется

минимизировать $f(\mathbf{x}) = x_1^2 + x_2^2 - 16x_1 - 10x_2$
при ограничениях

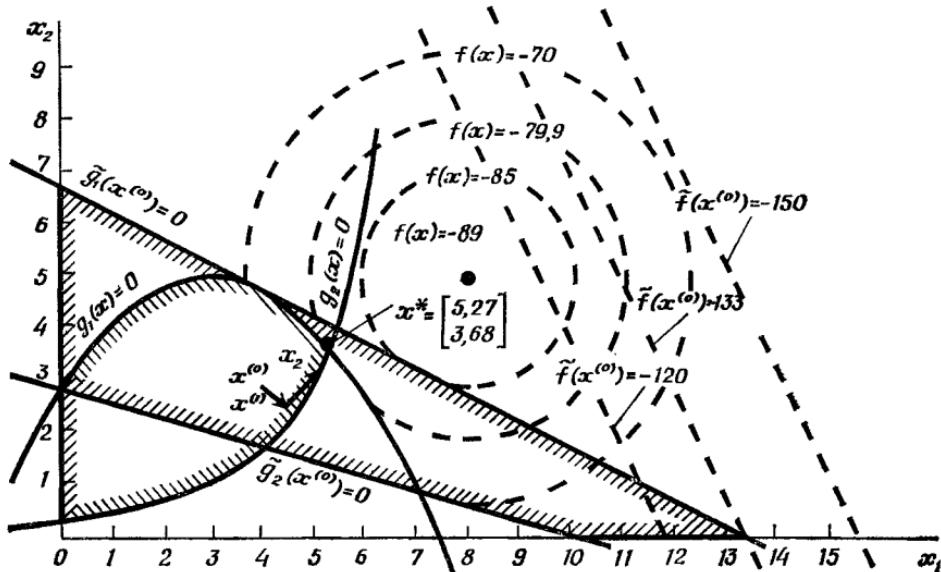
$$g_1(\mathbf{x}) = 11 - x_1^2 + 6x_1 - 4x_2 \geq 0,$$

$$g_2(\mathbf{x}) = x_1x_2 - 3x_2 - e^{x_1-3} + 1 \geq 0, \quad (a)$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

На фиг. П.6.1.1 дано графическое представление (топография) задачи (a); пунктирыми линиями изображены уровни целевой



Фиг. П.6.1.1. Геометрическое представление (топография) задач (a) и (б).

функции $f(\mathbf{x})$, а сплошными кривыми — два первых нелинейных ограничения, соответствующие $g_i(\mathbf{x}) = 0$ ($i = 1, 2$). Оси координат соответствуют уравнениям $g_3(\mathbf{x}) = 0$ и $g_4(\mathbf{x}) = 0$. Решением задачи (a) является $\mathbf{x}^* = [5,27; 3,68]^T$; при этом $f(\mathbf{x}) = -79,9$. Начальные шаги процедуры поиска решения сводятся к следующему:

1. В качестве начальной (допустимой) точки выбирается $\mathbf{x}^{(0)} = [4; 3]^T$, в которой $f(\mathbf{x}^0) = -69$.
2. Линейная аппроксимация задачи (a) в окрестности точки

$\mathbf{x}^{(0)} = [4 \ 3]^T$ достигается путем замены фигурирующих в (а) нелинейных функций их линейными приближениями; линеаризация приводит к следующей задаче:

минимизировать $\tilde{f}(\mathbf{x}^{(0)}) = -8x_1 - 4x_2 - 25$
при ограничениях

$$\begin{aligned}\tilde{g}_1(\mathbf{x}^{(0)}) &= -2x_1 - 4x_2 + 27 \geq 0, \\ \tilde{g}_2(\mathbf{x}^{(0)}) &= 0,28x_1 + x_2 - 2,84 \geq 0, \\ \tilde{g}_3(\mathbf{x}^{(0)}) &= x_1 \geq 0, \\ \tilde{g}_4(\mathbf{x}^{(0)}) &= x_2 \geq 0.\end{aligned}\quad (\text{б})$$

Как уже отмечалось выше, символ \sim означает, что берется линейное приближение рассматриваемой функции. Допустимая область, определяемая линеаризованными ограничениями задачи (а) в окрестности $\mathbf{x}^{(0)} = [4 \ 3]^T$, ограничена на фиг. П.6.1.1 сплошными прямыми, а уровни линеаризованной целевой функции $f(\mathbf{x}^{(0)})$ представлены пунктирными прямыми. Решением задачи (б) является $\mathbf{x}^{(1)} = [13,5 \ 0]^T$; этот вектор допустим по отношению к задаче (б) и не является допустимым по отношению к задаче (а).

Чтобы предотвратить выход \mathbf{x} из допустимой области задачи (а), на \mathbf{x} накладывают ограничения вида (6.1.2), т. е. полагают

$$|x_j^{(1)} - x_j^{(0)}| \leq \delta_j^{(0)}, \quad j = 1, \dots, n. \quad (\text{в})$$

Начальное значение $\delta^{(0)}$ можно выбирать произвольным образом; для задачи (а) подходящим оказывается, например, $\delta^{(0)} = [0,5 \ 0,5]^T$. Заметим, что перемещение от точки $\mathbf{x}^{(0)} = [4 \ 3]^T$ к точке $\mathbf{x}^{(1)} = [13,5 \ 0]^T$ сопряжено с существенным увеличением x_1 и заметным уменьшением x_2 . Учтем теперь условие (в). В полученной точке $\mathbf{x}^{(1)} = [(4+0,5) \ (3-0,5)]^T = [4,5 \ 2,5]^T$ функция $f(\mathbf{x}^{(1)}) = -70,65$.

На следующем шаге ($k = 1$) положим $\delta^{(1)} = 0,8\delta^{(0)} = [0,4 \ 0,4]^T$. Линейный эквивалент задачи (а) в окрестности $\mathbf{x}^{(1)} = [4,5 \ 2,5]^T$ имеет следующий вид:

минимизировать $\tilde{f}(\mathbf{x}^{(1)}) = -7x_1 - 5x_2 - 36$
при ограничениях

$$\begin{aligned}\tilde{g}_1(\mathbf{x}^{(1)}) &= 31,4 - 3x_1 - 4x_2 \geq 0, \\ \tilde{g}_2(\mathbf{x}^{(1)}) &= 5,42 - 1,98x_1 + 1,5x_2 \geq 0, \\ \tilde{g}_3(\mathbf{x}^{(1)}) &= x_1 \geq 0, \\ \tilde{g}_4(\mathbf{x}^{(1)}) &= x_2 \geq 0.\end{aligned}\quad (\text{г})$$

Решением задачи (г) является $\mathbf{x}^{(2)} = [5,45 \quad 3,58]^T$; данная точка расположена весьма близко к допустимой области задачи (а). Наложив дополнительное условие $|x_i^{(2)} - x_i^{(1)}| \leq \delta_i^{(1)}$ ($j = 1, \dots, n$), в качестве допустимой точки $\mathbf{x}^{(2)}$ получаем $\mathbf{x}^{(2)} = [4,9 \quad 2,9]^T$; в этой точке $f(\mathbf{x}^{(2)}) = 75,1$. Следующий шаг ($k = 2$) состоит в выполнении вычислительной процедуры по приведенной выше схеме при $\delta^{(2)} < \delta^{(1)}$ (например, при $\delta^{(2)} = 0,8\delta^{(1)}$). Итерационный процесс заканчивается, как только выполняется условие $|\delta^{(k)}| \leq |\varepsilon|$ (где $|\varepsilon|$ — достаточно малая величина) и $\mathbf{x}^{(k+1)}$ оказывается внутри допустимой области. На каждом шаге при выходе \mathbf{x} из допустимой области, прежде чем продолжать процесс минимизации, методом линейного программирования определяется решение, являющееся допустимым.

6.1.2. МЕТОД ПОП

В 1965 г. Г. Смитом был разработан метод отыскания оптимального решения, основанный (аналогично методу, рассмотренному в разд. 6.1.1) на линеаризации и пригодный для решения задач нелинейного программирования в случае, когда ограничения в виде равенств отсутствуют. Этот метод положен в основу универсальной машинной программы, известной под названием ПОП¹⁾ (программа оптимизации процессов). В соответствии с ПОП процедура линеаризации выполняется на каждом шаге и каждая из промежуточных линейно-оптимизационных задач решается с помощью модифицированного симплексного метода. Повторяющийся процесс линеаризации (релинеаризация) осуществляется в окрестности найденной на предыдущем шаге оптимальной точки, а предельные значения для длины шага устанавливаются с таким расчетом, чтобы не выйти за рамки, вне которых соотношения, полученные в результате линеаризации, не могут быть использованы. Как уже отмечалось выше, с помощью рассматриваемого метода не удается решать задачи с ограничениями, записанными в виде равенств; исключение составляют случаи, когда каждое из равенств может быть представлено в виде двух неравенств, например, по следующей схеме:

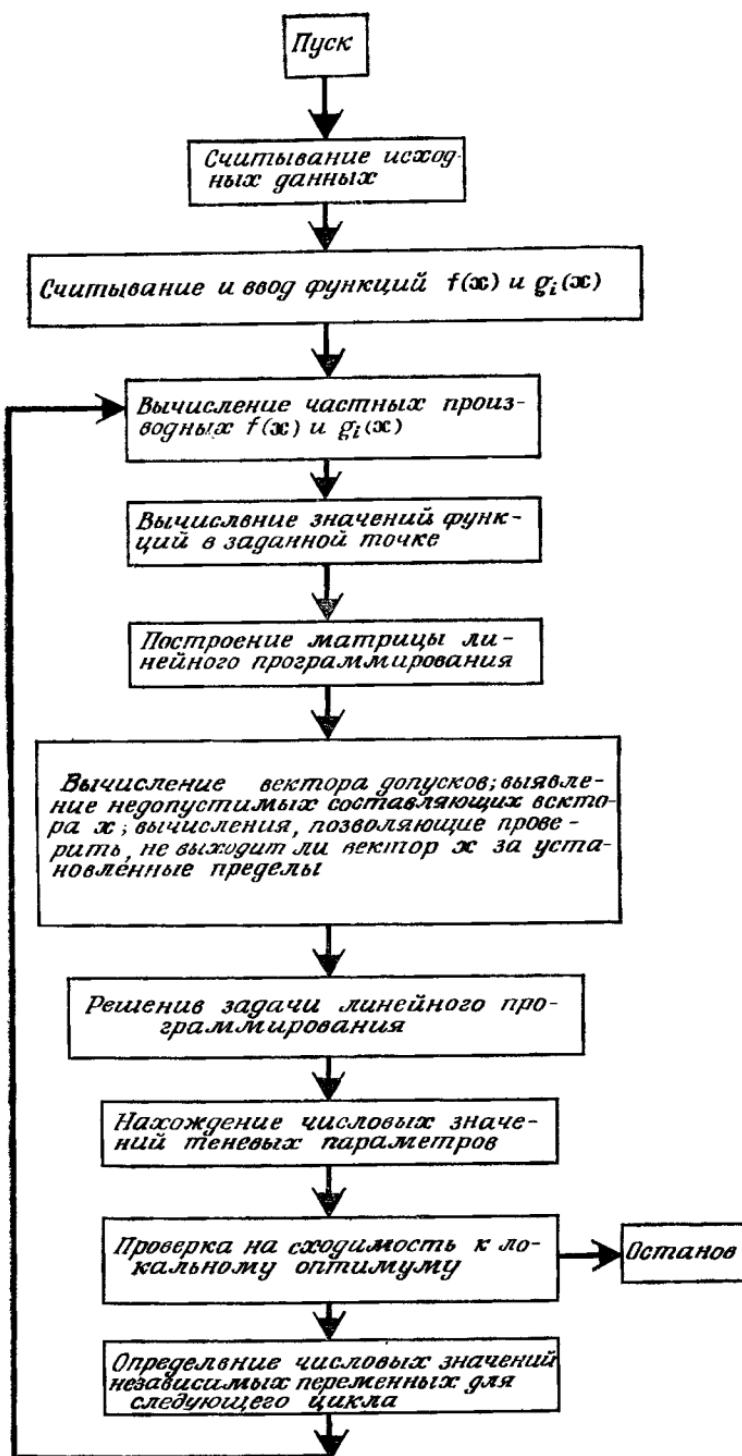
равенство $h(\mathbf{x}) = 0$

заменяется эквивалентной ему парой неравенств

$$h_1(\mathbf{x}) + \xi \geq 0 \text{ и } h_2(\mathbf{x}) - \xi \leq 0.$$

(В подобных ситуациях в процессе поиска оптимального решения значение ξ постепенно уменьшается.) Однако такой способ обращения

¹⁾ Подробный анализ ПОП (в английском варианте POP — Process Optimization Program) содержится в работе [4].



Ф и г. 6.1.2. Блок-схема вычислительного процесса для алгоритма ПОП II.

с ограничениями в виде равенств редко оправдывает себя на практике. Поэтому ПОП, вообще говоря, используется лишь для решения задач следующего вида:

минимизировать $f(\mathbf{x})$, $\mathbf{x} \in E^n$,

при ограничениях

$$\begin{aligned} g_i(\mathbf{x}) &\geq 0, \quad i = 1, \dots, p, \\ L_j &\leq x_j \leq U_j, \quad j = 1, \dots, n. \end{aligned} \quad (6.1.4)$$

При решении локально линеаризованной задачи нелинейного программирования применяют метод линейного программирования таким образом, чтобы последовательность решений промежуточных линеаризованных задач сходилась к решению исходной нелинейной задачи (6.1.4). Получаемые при этом векторы \mathbf{x} могут быть как допустимыми, так и недопустимыми. На фиг. 6.1.2 приведена в общем виде блок-схема вычислительного процесса (или блок-схема обработки данных) при реализации алгоритма Смита.

ПОП автоматически формирует матрицу локального линейного программирования (ЛП) (фиг. 6.1.3), элементы которой выражаются через коэффициенты линеаризованной модели. При этом в строках, номера которых равны или превосходят $(n + 1)$, располагаются значения первых частных производных целевой функции и функций, фигурирующих в ограничениях, по независимым переменным x_j . Числовые значения этих частных производных берутся в точке, задаваемой текущим вектором \mathbf{x} . Следует отметить, что отдельная строка отводится для каждой независимой переменной x_j . Кроме того, каждому активному ограничению соответствует отдельная строка.

Матрица, ассоциированная с алгоритмом линейного программирования, содержит также ряд дополнительных строк и столбцов, в частности следующие столбцы:

1. *Вектор-столбец, задающий предельные значения для перемещений в пространстве решений (вектор допусков).* Элементы этого столбца, расположенные в первых n строках, которые ассоциируются с независимыми переменными, представляют собой наименьшие значения пределов перемещений вдоль соответствующих этим переменным направлений (или, другими словами, наименьшие расстояния до ближайшей границы внутри допустимой области по каждому из упомянутых направлений). Элементы, расположенные в строках с порядковыми номерами, лежащими в интервале от $(n + 1)$ до $(n + m + 4)$, представляют собой либо текущее значение g_i , либо разность между текущим и ближайшим предельным значением g_i .

2. *Вектор-столбец, ассоциированный с вспомогательными операциями.* Этот вектор-столбец используется при выполнении вычислений неявного характера, а также в связи с реализацией над-

лежащих вычислительных процедур в тех случаях, когда исходный (на рассматриваемом шаге) вектор выходит из допустимой области.

Если какая-либо из независимых переменных x_i принимает недопустимое значение, допустимость восстанавливается путем заме-

	Номер столбца	1 2 . . . n	$n + 1$ Вектор-столбец, управляющий вспомогательными операциями	$n + 2$ Вектор допусков	$n + 3$ Рабочее поле
x_j	1 2 . . . n	1 0 0 1			
$\frac{\partial f(x)}{\partial x_j}$	$n + 1$				
$\frac{\partial g_i(x)}{\partial x_j}$	$n + 2$ $n + 3$. . $n + m + 1$				
	$n + m + 2$	Вспомогательная операция	1	100	
	$n + m + 3$	Функциональное уравнение	-500		
	$n + m + 4$	Штрафная по- правка			

Ф и г. 6.1.3. Структура части матрицы линейного программирования.

ны данного значения x_i на наименьшую из разностей между рассматриваемым значением этой переменной и ее предельными значениями. На данном шаге требуется также внести надлежащие изменения в структуру вектора допусков. Возникающие нарушения ограничивающих условий временно устраняются путем комбинирования составляющих получаемого вектора с произвольно выби-раемыми элементами, которым придаются значения 100 и -500 (см. матрицу на фиг. 6.1.3).

Промежуточные задачи линейного программирования решаются с помощью модифицированного симплекс-алгоритма. Для каждого

из промежуточных решений в соответствии с машинной программой вычисляется приращение вектора \mathbf{x} (которое мы обозначим через $\Delta \mathbf{x}$) и осуществляется сложение за тем, чтобы значение целевой функции в точке $(\mathbf{x} + \Delta \mathbf{x})$ улучшилось по сравнению со значением $f(\mathbf{x})$, найденным на предыдущем шаге (т. е. в точке $\mathbf{x}^{(k)}$) при незначительных аппроксимационных погрешностях [см. приведенные ниже соотношения (6.1.5)]. Каждый такой цикл построения матрицы линейного программирования и решения промежуточной линеаризованной задачи называется *петлей*. Петли повторяются в автоматическом режиме до тех пор, пока не будет найдено оптимальное решение или пока число выполненных вычислительных циклов не превысит максимально допустимое значение, установленное пользователем.

1. Нахождение числовых значений частных производных. Как уже отмечалось выше, чтобы сформировать матрицу линейного программирования, требуется знать значения первых частных производных целевой функции и функций, фигурирующих в ограничениях, по каждой из независимых переменных x_i . Для приближенного вычисления этих частных производных в ПОП используется численный метод центрированных разностей. Чтобы определить значение первой частной производной целевой функции по независимой переменной x_i в текущей точке $\mathbf{x}^{(k)}$, по программе находятся значения $f(\mathbf{x})$ в точках $\mathbf{x}^{(k)} + \delta_j$ и $\mathbf{x}^{(k)} - \delta_j$, где вектор δ_j определяется параметрами, устанавливаемыми пользователем. Тогда на k -м шаге числовое значение первой частной производной целевой функции по x_i вычисляется по формуле

$$\frac{\partial f(\mathbf{x}^{(k)})}{\partial x_i} = \frac{f(\mathbf{x}^{(k)} + \delta_j) - f(\mathbf{x}^{(k)} - \delta_j)}{2\delta_j}, \quad j = 1, \dots, n,$$

где

$$\mathbf{x}^{(k)} + \delta_j = [x_1^{(k)}, x_2^{(k)}, \dots, x_j^{(k)} + \delta_j, \dots, x_n^{(k)}]^T$$

$$\text{и } \mathbf{x}^{(k)} - \delta_j = [x_1^{(k)}, x_2^{(k)}, \dots, x_j^{(k)} - \delta_j, \dots, x_n^{(k)}]^T.$$

Аналогично для вычисления первых частных производных $g_i(\mathbf{x})$ используется формула

$$\frac{\partial g_i(\mathbf{x}^{(k)})}{\partial x_j} = \frac{g_i(\mathbf{x}^{(k)} + \delta_j) - g_i(\mathbf{x}^{(k)} - \delta_j)}{2\delta_j}, \\ i = 1, \dots, p, j = 1, \dots, n.$$

На протяжении всего процесса оптимизации выбранные вначале значения δ_j остаются неизменными, причем от выбора этих значений существенно зависит точность численных оценок упомянутых выше частных производных.

2. Адаптивные предельные значения для длины шага. Используемые в ПОП II предельные значения для перемещения в пространстве решений (или длины шага) в направлениях, ассоцииро-

ванных с независимыми переменными x_i , можно получить одним из следующих способов (выбор остается за пользователем). Во-первых, упомянутые предельные значения могут устанавливаться пользователем по его усмотрению и рассматриваться как исходные данные (так, например, длина шага может составлять 10% от длины рассматриваемого диапазона изменений x_j); в этом случае программой предусматривается (после выполнения надлежащего масштабирования, описание которого приводится ниже) правило обращения с такого рода фиксированными входными данными. Возможен и другой вариант, когда вычисление предельных значений для длины шага при реализации каждой петли возлагается на саму программу. Оценивание аддитивных предельных значений для длины шага осуществляется способом, аналогичным способу нахождения числовых оценок первых частных производных $f(x)$ и $g_i(x)$ по x_j . В соответствии с программой для независимых переменных x_j вычисляются максимальные значения перемещений, при которых погрешность, обусловленная линеаризацией (см. ниже), не превышает максимально допустимую погрешность, указанную на входе как для целевой функции, так и для каждого из ограничений. По программе находятся числовые значения $f(x)$ и $g_i(x)$ ($i = 1, \dots, n$) в текущей точке $x^{(k)}$, а также в точках $x^{(k)} + \delta_j$ и $x^{(k)} - \delta_j$. Поскольку оценивание аддитивных предельных значений для длины шага выполняется одновременно с нахождением числовых значений первых частных производных, вычисление $f(x)$ и $g_i(x)$ ($i = 1, \dots, n$) в точках, задаваемых векторами $x + \delta_j$, x и $x - \delta_j$, осуществляется только один раз сразу для обеих из указанных целей. Затем в соответствии с ПОП находится линеаризационная погрешность для целевой функции по формуле

$$E_{j,0} = f(x) - \frac{f(x + \delta_j) + f(x - \delta_j)}{2}, \quad j = 1, \dots, n, \quad (6.1.5a)$$

и для функций $g_i(x)$ с помощью следующих соотношений:

$$\begin{aligned} E_{j,i} = g_i(x) - \frac{g_i(x + \delta_j) + g_i(x - \delta_j)}{2}, \quad & j = 1, \dots, n; i = \\ & = 1, \dots, p. \end{aligned} \quad (6.1.5b)$$

Получаемые числовые погрешности используются для нахождения предельного значения перемещения по x_j . Если исходить из точности линеаризации целевой функции, то предельное значение для длины шага получается в виде

$$\Delta x_{j,0,\max} = \delta_j \times \left(\frac{\text{максимально допустимая линеаризационная погрешность для целевой функции}}{E_{j,0}} \right)^{1/2}, \quad j = 1, \dots, n, \quad (6.1.6a)$$

а если исходить из точности линеаризации ограничений, то

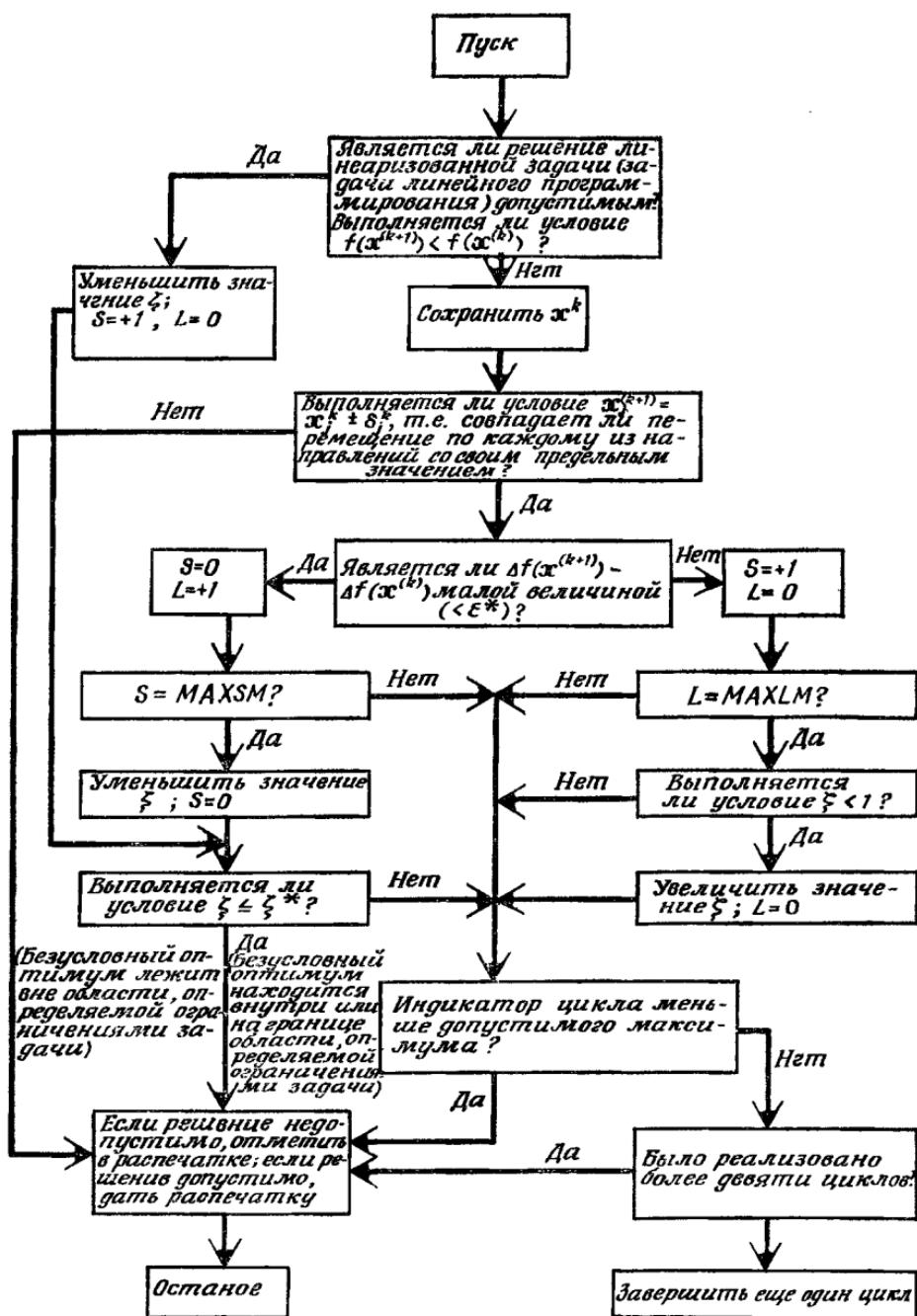
$$\Delta x_{j,i,\max} = \delta_j \left(\frac{\text{максимально допустимая линеаризационная погрешность для функций } g_i(\mathbf{x})}{E_{j,i}} \right)^{1/2},$$

$$j = 1, \dots, n, \quad i = 1, \dots, m. \quad (6.1.66)$$

Для каждой независимой переменной x_j предельное значение длины шага, используемое в процессе решения локальной (полученной методом линеаризации в точке $\mathbf{x}^{(k)}$) подзадачи линейного программирования, равняется (в зависимости от того, что окажется меньшим по своему значению) либо (1) наименьшему из $\Delta x_{j,i}$ ($i = 0, \dots, m$), либо (2) удвоенному предельному значению, установленному априори. Альтернатива (2) может иметь место, если только $f(\mathbf{x})$ и $g_i(\mathbf{x})$ линейны по x_j . Метод нахождения аддитивных предельных значений для длины шага по каждому из направлений в пространстве решений позволяет варьировать упомянутыми предельными значениями с учетом степени кривизны той или иной модели. С другой стороны, предельные значения длины шага могут оказаться слишком малыми; в этом случае процесс оптимизации сильно замедляется. При этом независимо от того, как выбраны предельные значения длины шага, последние умножаются на переменную величину ξ , которая равняется вначале единице, а затем (по мере приближения к оптимуму) постепенно убывает (см. описание соответствующей процедуры в следующем подразделе).

3. Завершение вычислительной процедуры. Алгоритмом, заложенным в ПОП II, предусматриваются два автономных критерия определения оптимума. Если безусловный оптимум (экстремум) находится в точке, которая расположена вне области, определяемой ограничениями задачи, то соответствующий условный экстремум достигается в конце фазы линейного программирования при получении такой ситуации, когда ни одна из компонент вектора $\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$ не превышает соответствующее предельное значение длины шага. Если же безусловный экстремум лежит внутри или на границе области, задаваемой ограничениями исходной задачи, то соответствующий условный экстремум достигается в силу того, что либо точка, определяемая текущим вектором \mathbf{x} , осциллирует относительно оптимальной точки, либо последовательность перемещений в пространстве решений при очень малых оптимизирующих поправках к текущим значениям $f(\mathbf{x})$ приводит к тому, что ξ начинает принимать значения, меньшие предельных значений, предписанных пользователем. Логика тестов на завершение вычислительной процедуры отражена в общих чертах на блок-схеме, приведенной на фиг. 6.1.4.

Если приращение x_j соизмеримо с предельным значением длины



Фиг. 6.1.4. Блок-схема вычислительного процесса, основанного на использовании критериев сходимости в алгоритме ПОП II.

MAXLM — максимально допустимое число последовательных «больших» шагов (перемещений), предшествующих началу возрастаания ξ . (Как правило, $\xi = 3$.) MAXSM — максимально допустимое число «мелких» шагов («малых» перемещений), которое может иметь место до того, как ξ начнет убывать. (Как правило, $\xi = 2$.)

ξ^* — минимальное из допустимых значений ξ . (Чаще всего $\xi^* = 0,05$.)

шага, то весьма вероятной оказывается возможность продолжить оптимизационный процесс путем дополнительных перемещений в указанном направлении. Тест окончания процесса осуществляется с той целью, чтобы установить,— большой или незначительной— можно считать положительную оптимизирующую поправку к текущему значению целевой функции. Оптимизирующая поправка считается большой, если она превышает некоторое фиксированное число, установленное пользователем¹⁾, и незначительной, если это условие не выполняется. Данный классификационный признак используется в связи с заданием в вычислительной программе изменений по счетчикам больших перемещений L и малых перемещений S . На фиг. 6.1.4 случай, когда $S = +1$ и $L = +1$, означает, что показание каждого из счетчиков возрастает на единицу; случай же, когда $L = 0$ и $S = 0$, указывает на то, что счетчики больших и малых перемещений устанавливаются на 0. Смит рекомендует увеличивать предельное значение длины шага путем умножения ξ на 2 и уменьшать это значение в такой же пропорции (т. е. путем деления множителя ξ на 2).

Для компоновки машинной программы, реализующей алгоритм ПОП II, требуется, чтобы пользователь подготовил специальную подпрограмму (называемую моделью), назначение которой состоит в том, чтобы обеспечить нахождение числовых значений целевой функции $f(\mathbf{x})$ и функций $g_i(\mathbf{x})$ для заданного вектора $\mathbf{x}^{(k)}$. Подпрограмма модели должна обеспечивать переоценку значений $f(\mathbf{x})$ и $g_i(\mathbf{x})$ после завершения каждого этапа линейного программирования. Кроме того, пользователь должен задать полный набор входных данных, включая начальный вектор \mathbf{x} , нижнюю и верхнюю границы значений независимых переменных и большое число других параметров оптимизации и контроля за протеканием вычислительного процесса. В саму программу может быть внесено так много различного рода коррекций, что часто остается неясным, какое влияние оказывает тот или иной параметр на ход оптимизационного процесса.

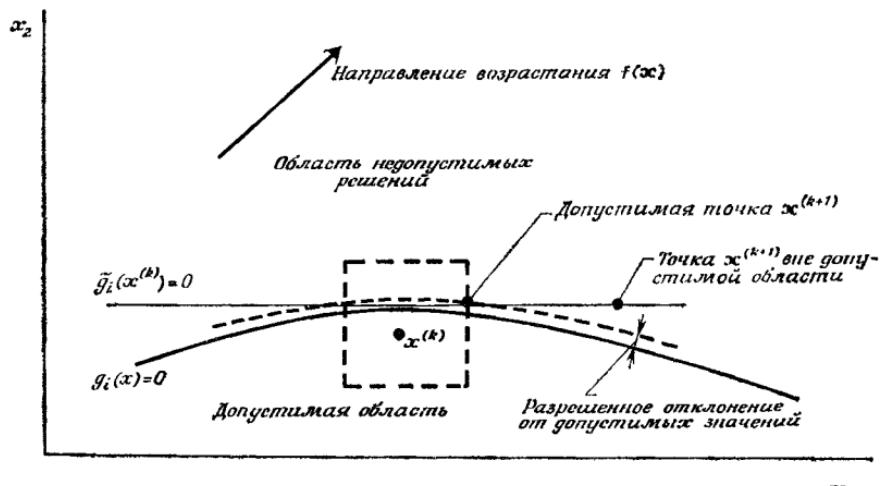
В гл. 9 будет проведено сравнение алгоритма ПОП II с другими оптимизационными алгоритмами; однако подробные примеры в данной связи приводиться не будут, поскольку это было бы со-пряженено с рассмотрением большого числа различных альтернатив, каждая из которых требует проведения серьезного анализа с тем, чтобы стала ясной связь того или иного из возможных вариантов с базовой структурой рассматриваемого алгоритма.

¹⁾ Смит в качестве наиболее типичной пороговой константы называет число 0,25; при этом, однако, следует, иметь в виду, что размерность константы должна совпадать с размерностью целевой функции,

6.1.3. ТРУДНОСТИ, ВОЗНИКАЮЩИЕ ПРИ АППРОКСИМИРУЮЩЕМ ЛИНЕЙНОМ ПРОГРАММИРОВАНИИ

Цварт [5] обратил внимание на следующие четыре трудности, которые возникают при решении задач нелинейного программирования путем многократной линеаризации и использования аппарата линейного программирования:

1. Оптимизация происходит медленно, если стремиться сделать векторы $\mathbf{x}^{(k-1)}, \mathbf{x}^{(k)}, \mathbf{x}^{(k+1)}, \dots$ допустимыми или очень близкими

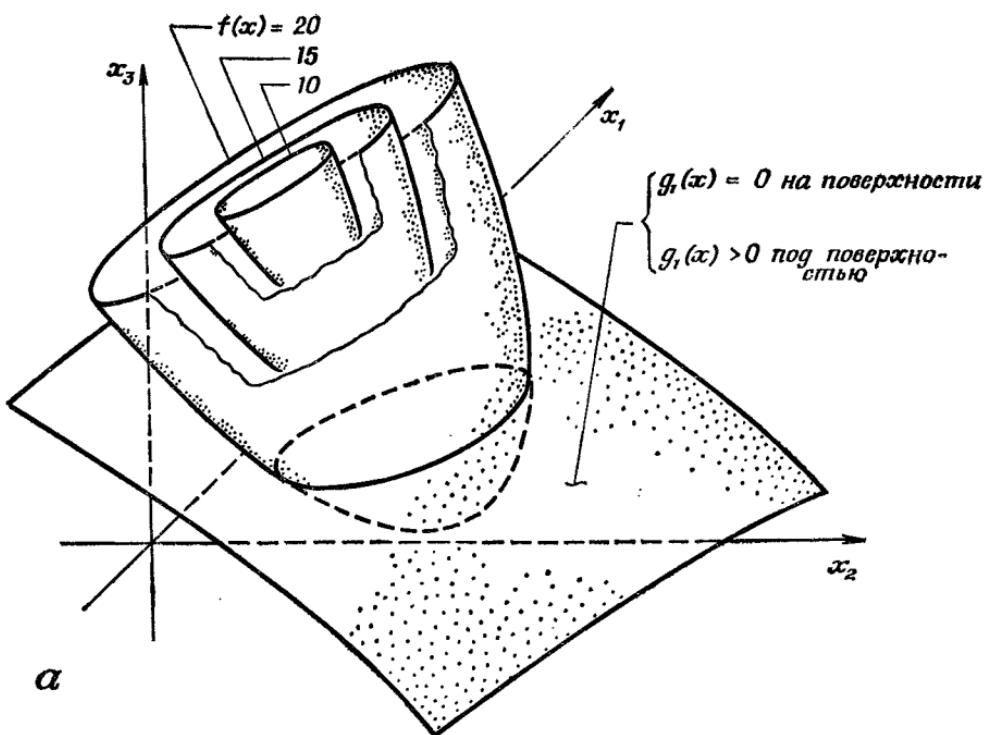


Фиг. 6.1.5. Влияние размера допуска на длину шага в двумерной задаче. Квадрат, изображенный пунктирными линиями, представляет значения $\delta_i^{(k)}$, фигурирующие в соотношении (6.1.2); $\tilde{g}_i(\mathbf{x}^{(k)}) = 0$ представляет собой линеаризованное ограничение.

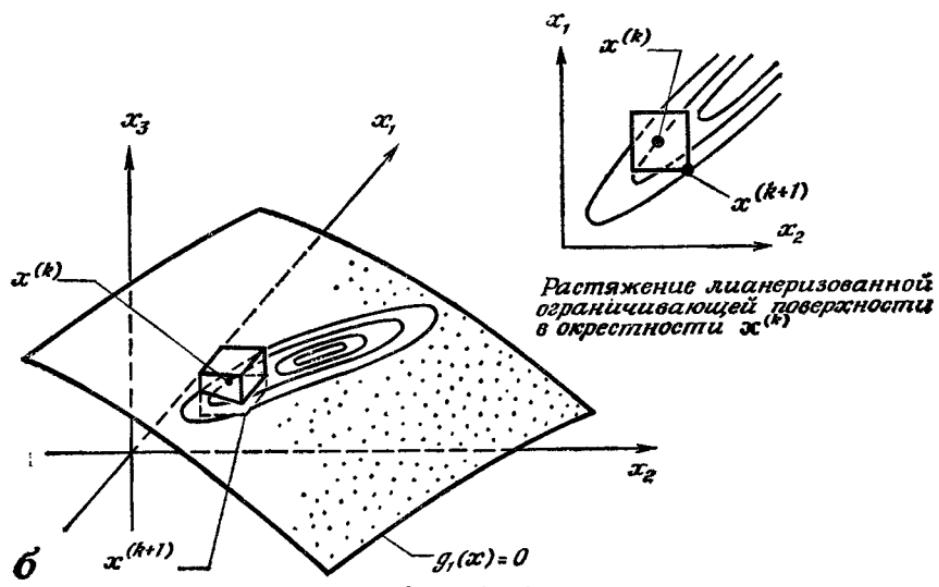
к допустимым. На фиг. 6.1.5 иллюстрируется случай, когда малый допуск при учете ограничений приводит к тому, что поиск оптимального решения оказывается слишком «мелкошаговым», т. е. протекает медленно. При увеличении допуска решение задачи нелинейного программирования может остаться недопустимым, если произойдет останов работы ЭВМ из-за ограничений по времени.

2. Трудно сравнивать один за другим два вектора \mathbf{x} , которые приводят к разным значениям $f(\mathbf{x})$ и в различной степени нарушают некоторое подмножество ограничений. Действительно, допустим, что мы хотим сравнить векторы $\mathbf{x}^{(k)}$ и $\mathbf{x}^{(k+1)}$. Пусть $f(\mathbf{x}^{(k)}) < f(\mathbf{x}^{(k+1)})$, но ограничение $g_1(\mathbf{x}) \geq 0$ для $\mathbf{x}^{(k)}$ нарушается сильнее, нежели ограничение $g_1(\mathbf{x}) > 0$ (или, возможно, другое ограничение $g_2(\mathbf{x}) \geq 0$) для вектора $\mathbf{x}^{(k+1)}$. Можно ли утверждать, что первый из этих векторов предпочтительнее второго?

3. Если целевая функция характеризуется высокой степенью нелинейности, то на основе решения линеаризованной задачи



a



Растяжение линеаризованной ограничивающей поверхности в окрестности $x^{(k)}$

Фиг. 6.1.6.

a — пересечение уровняй целевой функции с поверхностью, заданной активным ограничением; *б* — контуры, получаемые на поверхности, заданной линеаризованным ограничением.

направление поиска оптимальной точки может быть выбрано слишком неточно. Вспомним, что одна из трудностей градиентного метода, изложенного в гл. 3, заключается в том, что градиент оказывается направленным в сторону минимума, если только масштаб по каждой из переменных выбран таким, что уровни целевой функции представляют собой гиперсфера. Линейное программирование не позволяет правильно выбрать направление оптимизирующего поиска также и в том случае, когда поверхность, ограничивающая допустимую область, слишком растянута. На фиг. 6.1.6, а изображена целевая функция и ограничивающая поверхность $g_1(\mathbf{x})$, а на фиг. 6.1.6, б показано направление поиска на ограничивающей поверхности (от $\mathbf{x}^{(k)}$ к $\mathbf{x}^{(k+1)}$), определяемое с помощью линейного программирования для δ_i , соответствующего показанному кубу. Приведенный на фиг. 6.1.6, б дополнительный рисунок содержит более подробную информацию относительно направления оптимизирующего поиска на линеаризованной гиперповерхности, определяемой ограничением $\tilde{g}_1(\mathbf{x}^{(k)}) = 0$. Аналогичная трудность возникает в тех случаях, когда ограничивающая поверхность имеет ребристую структуру, если даже вид целевой функции достаточно близок к линейному.

4. Следует, кроме того, отметить, что при использовании метода аппроксимирующего линейного программирования трудно придумать такую процедуру учета априорных данных, которая бы надлежащим образом ускоряла процесс оптимизации.

6.1.4 КВАДРАТИЧНОЕ ПРОГРАММИРОВАНИЕ И ДРУГИЕ ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Вместо того чтобы аппроксимировать целевую функцию линейной функцией, ее с равным успехом можно аппроксимировать некоторой квадратичной формой. Уилсон [6] и Бил [7] разработали алгоритмы, реализующие подобный подход на практике; однако при этом не всегда оказывается ясным, позволяют ли такого рода более сложные алгоритмы сократить время выполнения вычислительных процедур по сравнению с временем, требуемым для решения исходной задачи методом линейной аппроксимации. В этой связи накоплен незначительный практический опыт. Можно отметить лишь вычислительную программу, предложенную Уилсоном; однако эта программа в конструктивном плане еще не доработана.

Грейвс и Уинстон [8], обнаружив, что при оптимизации без ограничений учет квадратичных членов в разложении функций в ряд ускоряет сходимость, расширили метод линейной аппроксимации путем введения в рассмотрение при решении общей задачи

нелинейного программирования членов второго порядка. При этом были сформулированы условия первого порядка для локальной стационарной точки. Для задачи без ограничений алгоритм указанных выше авторов переходит в обычный алгоритм спуска, учитывающий квадратичные члены. В случае когда ограничения линейны, а целевая функция имеет квадратичную форму, алгоритм Грэйвса и Уинстона становится тождественным алгоритму Франка и Вольфа [9]. Грэйвсом и Уинстоном при решении задач 8, 10, 11 и 18, приведенных в приложении А, установлено, что при использовании метода квадратичной аппроксимации для достижения заданной точности при нахождении оптимума требуется гораздо меньше итераций, нежели в случае метода, основанного на линеаризации исходных функций. Для упомянутых задач, начиная из точки, заданной вектором $x^{(0)} = [3 \ 3 \ 3 \ 3]^T$, получаем:

Номер задачи	При учете только первых производных		При учете вторых производных	
	число итераций	$f(x^*)$	число итераций	$f(x^*)$
10	>200	-32,3163	9	-32,3487
18	>200	-32,8153	16	-32,3478
11	7	-30665,29		
8	50	$2,25 \cdot 10^{-6}$	32	0

Интересно отметить, что, несмотря на различие в необходимом количестве итераций, машинное время, требуемое для поиска оптимального решения методом квадратичного программирования (с использованием матрицы Гессе), составляло 60—90% машинного времени, расходуемого при поиске оптимума методом линейной аппроксимации. Поскольку критерии завершения вычислительных процедур (останова) были для этих методов разными, метод квадратичного программирования, возможно, несколько более эффективен, нежели это можно заключить на основе приведенных выше данных.

6.2. АЛГОРИТМ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

Метод нелинейного программирования (сокращенно — метод НЛП) разработан Барнсом [10]. Он представляет собой обобщение алгоритма, предложенного Дибелла и Стивенсом [11]. Алгоритм Дибелла и Стивенса предназначался для решения задач нелинейного программирования при наличии ограничений только в виде равенств (плюс ограничения, задающие верхний и нижний пределы изменения значений независимых переменных); данный алгоритм основан на

использовании линейной аппроксимации нелинейных функций в окрестности некоторой допустимой или почти допустимой точки $\mathbf{x}^{(k)}$. Идея Дибелла относительно минимизации $f(\mathbf{x})$ путем применения градиентного и линейно-аппроксимативного методов получила дальнейшее развитие у Барнса, которому удалось обобщить упомянутый выше алгоритм на случай, когда в дополнение к ограничениям в виде равенств имеют место записанные в общем виде ограничения в виде неравенств. Таким образом, с помощью алгоритма НЛП можно решить задачу нелинейного программирования (6.0.1).

При реализации данного алгоритма в качестве начального могут использоваться как допустимые векторы, так и векторы, выходящие за пределы допустимой области. Если $\mathbf{x}^{(0)}$ не является допустимым, алгоритм реализует фазу наискорейшего спуска с тем, чтобы получить такой вектор \mathbf{x} , для которого множество ограничивающих условий «почти» удовлетворяется. Затем алгоритм работает на основе метода линейной аппроксимации с применением аппарата линейного программирования и, следовательно, обеспечивает построение надлежащей матрицы локального линейного программирования, элементами которой являются значения первых частных производных целевой функции и функций $g_i(\mathbf{x})$ по независимым переменным. Задача линейного программирования решается с помощью модифицированного симплексного метода; при этом к значению целевой функции добавляется оптимизирующая (минимизирующая) поправка в небольшой области, являющейся окрестностью текущей точки \mathbf{x} . После этого процедура линейного программирования повторяется. На любом этапе вычислительного процесса, когда в качестве текущего решения находится вектор \mathbf{x} , отклоняющийся от допустимой области сверх установленного предела, повторяется фаза наискорейшего спуска.

При обсуждении алгоритма НЛП удобно отдельно рассмотреть фазу наискорейшего спуска и фазу линейного программирования.

Фаза наискорейшего спуска в алгоритме НЛП сопряжена лишь с повышением степени допустимости вектора \mathbf{x} с точки зрения учета подмножеств ограничений $h_i(\mathbf{x}) = 0$ и $g_i(\mathbf{x}) > 0$; на этом этапе об улучшении значения целевой функции не заботятся. Подпрограмма наискорейшего спуска реализует безусловную минимизацию суммы квадратов остаточных погрешностей, т. е. выражения

$$T(\mathbf{x}) = \sum_{i=1}^m h_i^2(\mathbf{x}) + \sum_{i=m+1}^p U_i g_i^2(\mathbf{x}). \quad (6.2.1)$$

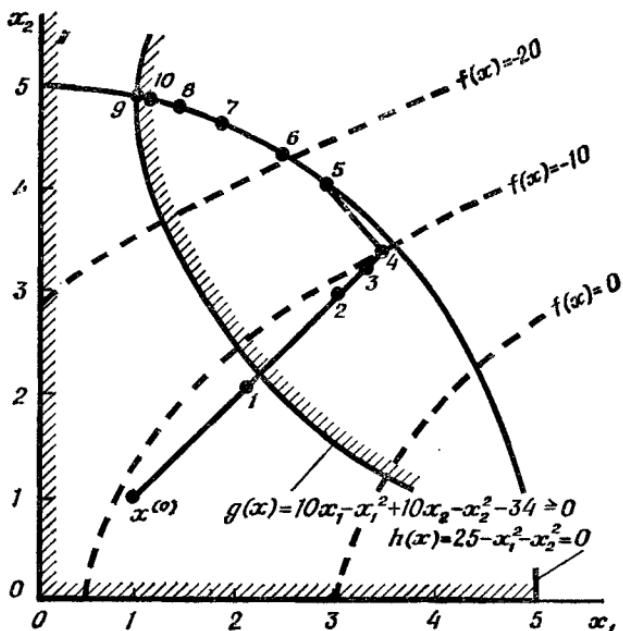
Совершенно очевидно, что для ограничений в виде равенств остаточная погрешность равняется $h_i(\mathbf{x})$ ($i = 1, \dots, m$), а для ограничений, записанных в виде неравенств, остаточная погрешность равняется $g_i(\mathbf{x})$, если $g_i(\mathbf{x}) < 0$, и обращается в нуль за счет

оператора Хевисайда (\mathcal{U}_i), если $g_i(x) > 0$ ($i = m + 1, \dots, p$); другими словами, $\mathcal{U}_i = 0$ при $g_i(x) \geq 0$. Сумма квадратов остаточных погрешностей вычисляется перед началом каждого этапа наискорейшего спуска и каждого этапа линейного программирования независимо от того, принадлежал ли предыдущий текущий вектор x допустимой области.

После вычисления $T(x)$ машинная программа осуществляет тесты, разработанные на основе анализа большого количества практических результатов решения различных «пробных» задач. С помощью этих тестов можно убедиться в том, что точка, заданная вектором x , действительно находится вблизи допустимой области, либо установить обратное. Если рассматриваемая точка находится достаточно близко от допустимой области, вычислительный процесс переходит в стадию линейного программирования, осуществляющего по алгоритму Гриффица и Стюарта (см. разд. 6.1); в противном случае вычисления продолжаются в режиме наискорейшего спуска.

Ускорение или замедление вычислительного процесса на стадии линейного программирования достигается путем варьирования максимального значения величины Δx_j . В тех случаях, когда точка, заданная вектором x , осциллирует относительно «хребта» или «долины», в рассмотрение вводится параметр ρ_j , меняющий знак, как только два последовательных шага Δx_j оказываются противоположными по знаку. Параметр ρ_j воздействует на счетчик λ_j (где вначале устанавливается единица), который используется для того, чтобы определять необходимость изменения предельной длины шага m_j . Значение λ_j определяется числом, которое показывает, сколько раз перемещение по переменной x_j осуществлялось в одном и том же направлении. Таким образом, числовое значение λ_j отражает количество благоприятных (оптимизирующих $f(x)$) приращений x_j в одном и том же направлении; значение λ_j вновь становится равным единице, как только направление перемещения по x_j меняется на противоположное. Если ρ_j меняет знак, то вместо значения m_j , которое использовалось на предыдущем шаге, берется значение $m_j/2$. Если же знак ρ_j не меняется, а числовое значение λ_j превышает 4, значение m_j , использовавшееся на предыдущем шаге, удваивается. Акселерация вычислительного процесса может вывести вектор x за пределы допустимой области; однако так как сумма квадратов остаточных погрешностей ($T(x)$) вычисляется после каждого изменения x , то в случае слишком больших отклонений текущей точки от границы допустимой области начинает работать метод наискорейшего спуска.

После каждого перемещения в пространстве решений на стадии линейного программирования проводится тест, цель которого состоит в том, чтобы выяснить, является ли используемое приращение Δx_j по своему абсолютному значению меньше некоторой за-



Ф и г. 6.2.1. Траектория поиска оптимального решения с помощью алгоритма НЛП для условий задачи, проиллюстрированной на фиг. 6.0.1.

ранее установленной (по усмотрению пользователя) константы ε (данная константа может быть выбрана произвольно, но следует иметь в виду, что она должна быть одной и той же для всех независимых переменных).

Если условие $|x_j^{(k)} - x_j^{(k-1)}| \leq \varepsilon$ выполняется, поиск заканчивается; в противном случае поиск оптимального решения продолжается.

На фиг. 6.2.1 показана траектория поиска применительно к условиям задачи, проиллюстрированной на фиг. 6.0.1.

6.3. ПРОЕКТИВНЫЕ МЕТОДЫ

Проективные методы образуют целое семейство. Иногда их называют методами допустимых направлений или градиентными методами с большой длиной шага; последнее название подчеркивает разницу между алгоритмами данного класса и алгоритмами, рассмотренными в разд. 6.1, которые можно назвать градиентными методами при малой длине шага. Сущность этих методов заключается в следующем: линейные или линеаризованные ограничения образуют линейное многообразие (определенное пересечением ограничений), на которое можно спроектировать выбранное направление поиска s в пространстве решений (см. разд. 6.3.1 и 6.3.2). Один из первых опубликованных методов такой структуры, а именно метод

Розена [12—16], по-видимому, является наиболее хорошо известным, поскольку для его реализации на ЭВМ существуют готовые машинные программы.

Все проективные методы (или методы допустимых направлений) предполагают реализацию на каждом k -м этапе указанной ниже последовательности шагов:

1. Алгоритм начинает работу в допустимой точке $\mathbf{x}^{(k)}$.
2. Определяется допустимое направление $\mathbf{s}^{(k)}$ (точное определение допустимого направления будет дано позднее).
3. В допустимом направлении выбирается шаг длиной $\lambda^{(k)}$, минимизирующий $f(\mathbf{x})$, но в то же время сохраняющий вектор $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)}\mathbf{s}^{(k)}$ допустимым.

Алгоритмы рассматриваемого семейства отличаются друг от друга прежде всего способами определения направления $\mathbf{s}^{(k)}$; однако все они характеризуются тем, что поиск начинается с допустимого решения и развивается (при линеаризованных ограничениях) в направлении, обеспечивающем уменьшение целевой функции при сохранении текущей точки \mathbf{x} внутри допустимой области.

Заметим, что большинство алгоритмов решения задач линейного программирования представляет собой модифицированные алгоритмы обращения матрицы. Так, например, симплексный метод линейного программирования соответствует методу обращения матриц, известному под названием «метод Гаусса — Жордана». Метод Фриша и метод Розена представляют собой два различных подхода к обращению симметрической матрицы $\mathbf{A}^T\mathbf{A}$ (где \mathbf{A} — матрица, элементами которой являются коэффициенты при независимых переменных в соотношениях, задающих ограничения). Однако следует отметить, что проективные методы не требуют, чтобы *все* ограничения в виде неравенств были заменены эквивалентной системой ограничений в виде равенств путем введения в рассмотрение ослабляющих переменных (как это имеет место в линейном программировании); проективные методы на каждом итерационном шаге вовлекают в вычислительный процесс по возможности минимальное количество ограничений в виде неравенств из числа активных ограничений. Прежде чем перейти к рассмотрению алгоритма Розена, поясним более подробно понятие проекции.

То, что здесь подразумевается под проекцией, ассоциируется с понятием проекции вектора, описывающего локальное приращение функции при перемещении в заданном направлении.

Пусть $\mathbf{A}^T = [\mathbf{a}_1 \mathbf{a}_2 \dots \mathbf{a}_n]$ — прямоугольная матрица с линейно независимыми вектор-столбцами, определяющими некоторое пространство R . Допустим, что \mathbf{A} допускает разбиение на две матрицы $\mathbf{A}_l^T = [\mathbf{a}_1 \dots \mathbf{a}_l]$ и $\mathbf{A}_{n-l}^T = [\mathbf{a}_{l+1} \dots \mathbf{a}_n]$. На векторы, входящие в \mathbf{A}^T , натянуто n -мерное пространство, и, следовательно, любой вектор в R может быть однозначно представлен в виде линейной

комбинации векторов \mathbf{a}_j , т. е.

$$\mathbf{x} = \sum_{j=1}^n \tau_j \mathbf{a}_j \text{ или } \mathbf{x} = \mathbf{A}^T \boldsymbol{\tau}. \quad (6.3.1a)$$

(Таким образом, коэффициенты τ_j представляют собой координаты вектора \mathbf{x} относительно базиса \mathbf{A} .) С другой стороны, векторы, составляющие матрицу \mathbf{A}_l^T , задают пространство лишь l -й размерности, и, следовательно, только для некоторого подмножества векторов в R имеет место соотношение

$$\mathbf{x} = \sum_{j=1}^l \tau_j \mathbf{a}_j \text{ или } \mathbf{x} = \mathbf{A}_l^T \boldsymbol{\tau}. \quad (6.3.1b)$$

Уравнение $\mathbf{a}_j^T \mathbf{x} = 0$ определяет в R гиперплоскость, проходящую через начало координат. Обозначим пересечение множества всех гиперплоскостей с $j = 1, 2, \dots, l$ (т. е. систему уравнений $\mathbf{A}_l \mathbf{x} = \mathbf{0}$), формирующих линейное многообразие¹⁾ размерности ($n - l$), через \mathcal{M} . Ортогональная проекция \mathbf{x} на \mathcal{M} (которую обозначим через $\mathbf{x}_{\mathcal{M}}$) обладает тем проективным свойством, что $\mathbf{x}_{\mathcal{M}}$ и $\mathbf{x} - \mathbf{x}_{\mathcal{M}}$ ортогональны, т. е.

$$\mathbf{x}_{\mathcal{M}}^T (\mathbf{x} - \mathbf{x}_{\mathcal{M}}) = 0.$$

Тогда n -мерная квадратная проектирующая матрица \mathbf{P}_l , определяемая условием

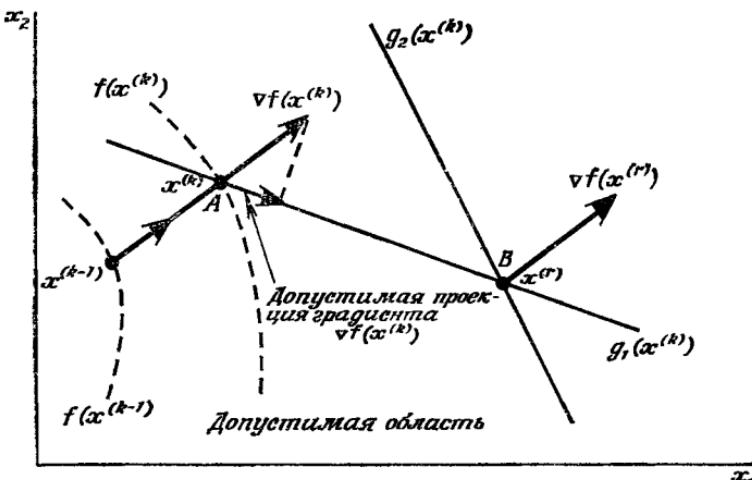
$$\mathbf{x}_{\mathcal{M}} = \mathbf{P}_l \mathbf{x} \quad (6.3.2)$$

и вычисляемая с помощью соотношения [17]

$$\mathbf{P}_l = \mathbf{I} - \mathbf{A}_l^T (\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l, \quad (6.3.3)$$

позволяет найти $\mathbf{x}_{\mathcal{M}}$. В одном предельном случае $\mathbf{P}_0 \equiv \mathbf{I}$. Это означает, что если подмножество l векторов пусто (т. е. $l = 0$), то $\mathbf{x}_{\mathcal{M}} =$

1) Понятие многообразия в E^n почти совпадает с понятием подпространства в E^n . Разница заключается лишь в том, что многообразие не обязано включать в себя начало координат, как это должно быть в случае подпространства. Когда говорят, что линейное многообразие \mathcal{M} имеет размерность $(n - l)$, то имеют в виду то обстоятельство, что если рассмотреть систему линейных уравнений $\mathbf{A}^T \mathbf{x} = \mathbf{b}$, то наибольшее число линейно независимых векторов, формирующих решение, оказывается равным $(n - l)$. Например, если \mathbf{A}^T есть $(l \times n)$ -матрица ($n > l$), причем каждая из l строк определяет линейно независимое уравнение, то $(n - l)$ есть число независимых переменных в системе уравнений, а l соответствует числу ограничений (или числу зависимых переменных). В E^n точка имеет размерность $l = 0$, так что $n - l = n$; гиперплоскость имеет размерность $l = 1$, и, следовательно, $n - l = n - 1$ и т. д. Таким образом, добавление независимого уравнения к заданной системе линейно независимых уравнений сокращает размерность многообразия \mathcal{M} на единицу.



Ф и г. 6.3.1. Проекция градиента на активные ограничения в случае ограничений в виде неравенств.

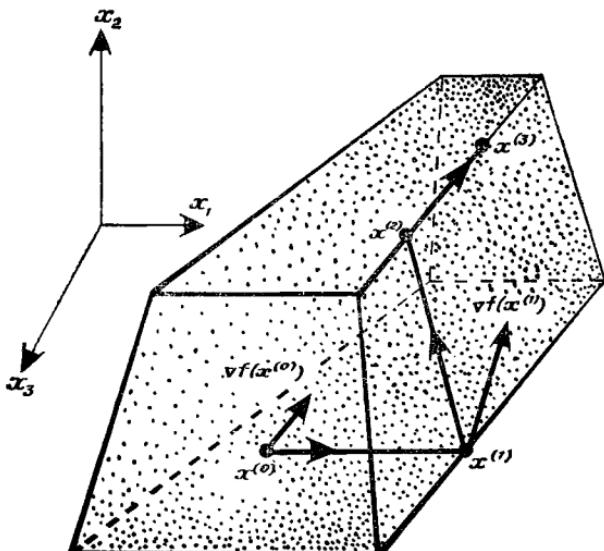
В точке A активно одно ограничение, и, следовательно, $n-l=2-1=1$; в точке B активными являются два ограничения, и, таким образом, $n-l=2-2=0$. В точке $x^{(k-1)}$, являющейся внутренней допустимой точкой, активные ограничения отсутствуют и $P_0=1$, так что проекция градиента $v f(x^{(k-1)})$ совпадает с самим градиентом. Заметим, что на пересечении в точке $x^{(k)}$ двух активных ограничений $P_2 \equiv 0$ и единственной возможной проекцией градиента $v f(x^{(k)})$ является указанная точка.

$= x$; следовательно, вектор x совпадает со своей проекцией. В другом предельном случае $P_n = 0$, т. е. подмножество векторов в R является полным, и, следовательно, $l=n$, а $x_M=0$ (проекция на точку есть нуль-вектор, как показано на фиг. 6.3.1). В случае когда на пересечение подмножества ограничений проектируется градиент целевой функции в точке $x^{(k)}$ (т. е. $\nabla f(x^{(k)})$), метод оптимизации называется *методом проекции градиента*.

6.3.1. МЕТОД ПРОЕКЦИИ ГРАДИЕНТА (МЕТОД РОЗЕНА)

Обратимся теперь к рассмотрению метода Розена. По существу этот метод минимизации представляет собой метод наискорейшего спуска, применяемый в сочетании с ортогональным проектированием отрицательного градиента на линейное многообразие ограничений или их аппроксимаций. В допустимой точке $x^{(k)}$ ограничения, имеющие вид равенств (активные ограничения), линеаризуются (если являются нелинейными) и временно заменяются соответствующими (т. е. касательными к ним в точке $x^{(k)}$) гиперплоскостями. Предполагается, что эти гиперплоскости линейно независимы и, следовательно, нормали к ним также линейно независимы. (Как показано Розеном, метод проекции градиента не позволяет манипулировать ограничениями, представляющими собой линейные

комбинации других ограничений). Эти нормали (соответствующие составляющие как для $\nabla g_i(\mathbf{x}^{(k)})$, так и для $\nabla h_i(\mathbf{x}^{(k)})$) будем обозначать через $\partial g_i(\mathbf{x}^{(k)})/\partial x_j$; используются для составления проектирующей матрицы, которая проектирует градиент целевой функции $\nabla f(\mathbf{x})$, найденный в точке $\mathbf{x}^{(k)}$, на пересечение касательных гиперплоскостей. В пространстве без ограничений n переменных форми-



Ф и г. 6.3.2. Метод проекции градиента при активных ограничениях в трехмерном пространстве.

Допустимая область представляет собой многогранник, состоящий из граней (двумерные многообразия), ребер (одномерные многообразия) и вершин (нульмерные многообразия). Стрелка указывает направление проекции градиента. $\mathbf{x}^{(0)}$ — начальная точка; $\mathbf{x}^{(1)}$ — минимальное смещение в $(\mathbf{x}^{(1)} - \mathbf{x}^{(0)})$ -направлении и т. д.

ируют n -мерную область. Однако, если в точке $\mathbf{x}^{(k)}$ должны удовлетворяться l ограничивающих условий (ограничивающие условия в виде равенств плюс активные ограничивающие условия в виде неравенств), размерность пространства, ассоциированного с линейно независимыми векторами, уменьшается на l , т. е. становится равной $(n - l)$. Понятие *приведенный базис*, или *базис при наличии ограничений*, означает совокупность линейно независимых гиперплоскостей, пересечения которых ограничивают область поиска. Основной особенностью метода проекции градиента является то, что он преобразует (проектирует) градиент целевой функции так, что его составляющие лежат в $(n - l)$ -мерном многообразии, определяющем допустимую область, в которой удовлетворяются l ограничений в виде равенств. Рассмотрим фиг. 6.3.2.

Обозначим через A_l матрицу (размером $n \times l$) частных производных активных ограничений (для экономии места зависимость

\mathbf{A}_l от $\mathbf{x}^{(k)}$ будет предполагаться, но не будет указываться в обозначениях в явном виде):

$$\mathbf{A}_l = \begin{bmatrix} \frac{\partial g_1(\mathbf{x}^{(k)})}{\partial x_1} & \cdots & \frac{\partial g_1(\mathbf{x}^{(k)})}{\partial x_n} \\ \cdot & \cdots & \cdot & \cdots & \cdot \\ \frac{\partial g_l(\mathbf{x}^{(k)})}{\partial x_1} & \cdots & \frac{\partial g_l(\mathbf{x}^{(k)})}{\partial x_n} \end{bmatrix} = \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \cdot & \cdots & \cdot \\ a_{l1} & \dots & a_{ln} \end{bmatrix}.$$

В силу соотношений (6.3.2) и (6.3.3) проекция $\nabla f(\mathbf{x}^{(k)})$ на активные ограничения определяет новое направление поиска $\mathbf{s}^{(k+1)}$:

$$\mathbf{s}^{(k+1)} = \mathbf{P}_l \nabla f(\mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k)}) - \mathbf{A}_l^T (\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l \nabla f(\mathbf{x}^{(k)}). \quad (6.3.4)$$

Выражение $(\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l \nabla f(\mathbf{x}^{(k)}) = \mathbf{u}$ можно рассматривать как вектор-столбец, элементами которого являются множители Лагранжа (так называемые теневые параметры). С его помощью можно определить новое направление

$$\mathbf{s}^{(k+1)} = \nabla f(\mathbf{x}^{(k)}) - \mathbf{A}_l^T \mathbf{u}.$$

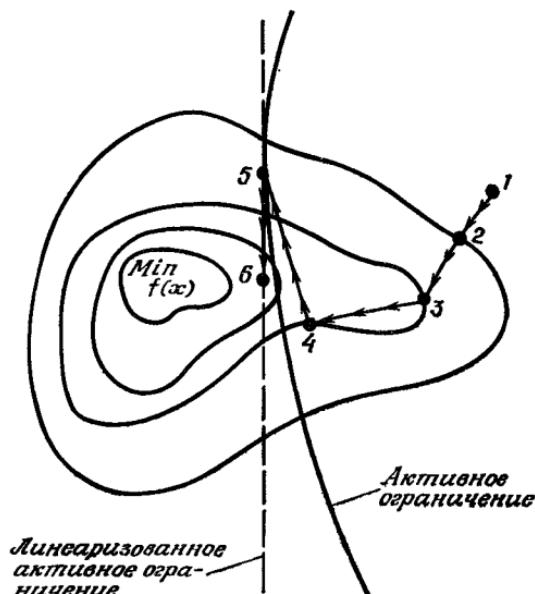
При этом составляющие u_j вектора \mathbf{u} позволяют уточнить, можно ли в ходе поиска по данному направлению получить дополнительные оптимизирующие поправки для целевой функции. Если какая-либо из составляющих u_j отрицательна, соответствующее ограничение в виде неравенства можно исключить из приведенного базиса до того, как будет продолжен оптимизационный поиск. Из-за взаимного пересечения ограничений одновременное удаление из приведенного базиса более одного ограничения в практическом отношении нецелесообразно; при этом удаляется то ограничение, которое соответствует наибольшей по абсолютной величине составляющей u_j .

Если активные ограничения линейны, составляющие вектора нового направления $\mathbf{s}^{(k+1)}$ будут лежать на самих ограничениях (фиг. 6.3.2). Если же активные ограничения нелинейны, как это имеет место в задаче, иллюстрируемой на фиг. 6.3.3, то составляющие вектора нового направления будут лежать на гиперплоскостях, касательных к ограничениям в точке $\mathbf{x}^{(k)}$. Таким образом, в случае, когда у некоторой граничной точки дальнейшее перемещение по градиенту целевой функции сопряжено с выходом за пределы допустимой области, вместо перемещений по составляющим градиента производится перемещение по составляющим проекции градиента. Следовательно, если задача содержит нелинейные ограничения, а $\mathbf{x}^{(k+1)}$ оказывается вне допустимой области, то применяется (в том или ином виде) алгоритм проекции градиента, что позволяет вернуться в ближайшую допустимую точку. После этого работа алгоритма повторяется. Оптимизационный процесс реали-

зуется большими шагами с линейными ограничениями, причем в конце каждого этапа текущий вектор \mathbf{x} непременно является допустимым и либо лежит на поверхности, порождаемой одним из ограничений, либо может быть разложен на составляющие, лежащие в плоскостях, представляющих некоторое подмножество ограничений. В случае задачи оптимизации без ограничений метод Розена сводится к методу наискорейшего спуска, описание которого приведено в гл. 3.

Переходя к подробному описанию алгоритма Розена, мы вместо общей задачи (6.0.1) ограничимся рассмотрением задачи нелинейного программирования с линейными ограничениями, поскольку данными относительно его практического применения в условиях общей постановки задачи (6.0.1) мы пока не располагаем¹⁾. Рассматриваемый алгоритм начинает работу с допустимой точки и продолжает эффективно действовать как метод наискорейшего спуска до тех пор, пока не будет достигнута точка, для которой становятся активными одно или несколько ограничений.

Если задача содержит ограничения в виде равенств, активные ограничения имеют место на каждом этапе вычислительного процесса; если же в задаче фигурируют ограничения только в виде неравенств, то по истечении некоторого времени мы должны оказаться в граничной точке (если решение задачи не представляется внутренней точкой). В граничной точке $\mathbf{x}^{(k)}$ функция $g_i(\mathbf{x}^{(k)}) = 0$ и $h_i(\mathbf{x}^{(k)}) = 0$ для $i = 1, \dots, l$, где $l \leq n$ (n — число переменных). В методе Розена $\nabla f(\mathbf{x}^{(k)})$ проектируется на пересечение l ограничений $\{\mathbf{x}^{(k)} | g_i(\mathbf{x}^{(k)}) = 0 \text{ и } h_i(\mathbf{x}^{(k)}) = 0, i = 1, \dots, l\}$. Пусть $\mathbf{P}_l = \mathbf{I} - \mathbf{A}_l^T (\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l$ — проектирующая матрица. При $l = 0$ (в этом случае $\mathbf{x}^{(k)}$ является внутренней



Фиг. 6.3.3. Метод проекции градиента при активном ограничении. Если активное ограничение в точке $\mathbf{x}^{(5)}$ нелинейно, точка $\mathbf{x}^{(6)}$ может оказаться вне допустимой области.

¹⁾ В подразд. 6.3.3 обсуждается метод аккомодации нелинейных ограничений, прошедший экспериментальную проверку.

точкой) \mathbf{P}_0 представляет собой единичную матрицу ($\mathbf{P}_0 = \mathbf{I}$). Проекция $\nabla f(\mathbf{x}^{(k)})$ на пересечение всех гиперповерхностей $g_i(\mathbf{x}^{(k)}) = 0$ и $h_i(\mathbf{x}^{(k)}) = 0$ ($i = 1, \dots, l$) находится с помощью соотношения (6.3.4). В разд. 2.5 отмечалось, что $\mathbf{x}^{(k)}$ является решением задачи нелинейного программирования тогда и только тогда, когда $\mathbf{P}_l \times \nabla f(\mathbf{x}^{(k)}) = 0$ и $\mathbf{u} = (\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l \nabla f(\mathbf{x}^{(k)}) \geq 0$. С учетом этого обстоятельства рассмотрим отдельно случаи $\mathbf{P}_l \nabla f(\mathbf{x}^{(k)}) \neq 0$ и $\mathbf{P}_l \nabla f(\mathbf{x}^{(k)}) = 0$. (Если решением является внутренняя точка, то $\mathbf{P}_0 \nabla f(\mathbf{x}^{(k)}) = \nabla f(\mathbf{x}^{(k)}) = 0$.)

Условие I: $\mathbf{P}_l \nabla f(\mathbf{x}^{(k)}) \neq 0$.

Пусть $\hat{\mathbf{s}}^{(k)}$ есть единичный вектор в направлении проекции $\nabla f(\mathbf{x}^{(k)})$ на пересечение всех ограничений $g_i(\mathbf{x}^k) = 0$ и $h_i(\mathbf{x}^k) = 0$:

$$\hat{\mathbf{s}}^{(k)} = \frac{\mathbf{P}_l \nabla f(\mathbf{x}^{(k)})}{\|\mathbf{P}_l \nabla f(\mathbf{x}^{(k)})\|}. \quad (6.3.5)$$

Как обычно, воспользуемся соотношением

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \hat{\mathbf{s}}^{(k)}.$$

Путем одномерного перемещения из точки $\mathbf{x}^{(k)}$ вдоль проекции градиента можно определить такое λ , которое минимизирует значение $f(\mathbf{x})$ и одновременно сохраняет $\mathbf{x}^{(k+1)}$ внутри допустимой области. Точнее говоря, в результате одномерного перемещения в направлении $\hat{\mathbf{s}}^{(k)}$ мы определяем $\lambda^* = \max \{\lambda \mid \mathbf{x}^{(k)} + \lambda \hat{\mathbf{s}}^{(k)} \in R\}$, где $R = \{\mathbf{x} \mid g_i(\mathbf{x}) \geq 0 \text{ и } h_i(\mathbf{x}) = 0, i = 1, \dots, p\}$, а затем находим $\lambda^{(k)}$, удовлетворяющее условию $0 \leq \lambda^{(k)} \leq \lambda^*$, так что значение $f(\mathbf{x}^{(k)} + \lambda^{(k)} \hat{\mathbf{s}}^{(k)})$ в направлении $\hat{\mathbf{s}}^{(k)}$ оказывается минимальным. Тогда значение λ^* есть максимальная длина шага, который можно сделать в направлении $\hat{\mathbf{s}}^{(k)}$ из точки $\mathbf{x}^{(k)}$, не выходя при этом за пределы допустимой области R .

Если $\lambda^{(k)} < \lambda^*$, то число ограничений в виде равенств, которые удовлетворяются в точке $\mathbf{x}^{(k+1)}$, не меняется; иначе говоря, $g_i(\mathbf{x}^{(k+1)}) = 0$ и $h_i(\mathbf{x}^{(k+1)}) = 0$ для $i = 1, \dots, l$, и, следовательно элементы a_{ij} матрицы \mathbf{A}_l , используемой на $(k+1)$ -м шаге, остаются без изменений. С другой стороны, если $\lambda^{(k)} = \lambda^*$, то в точке $\mathbf{x}^{(k+1)}$ в дополнение к l ограничениям, которые учитывались в точке $\mathbf{x}^{(k)}$ в виде равенств (и сохраняющим вид равенств при переходе от $\mathbf{x}^{(k)}$ к $\mathbf{x}^{(k+1)}$), должны приниматься во внимание нелинейные ограничения (одно или более), представленные в виде соответствующих равенств. Рассмотрим переход от \mathbf{x}^0 к $\mathbf{x}^{(1)}$, показанный

графически на фиг. 6.3.2. В этом случае производится вычисление элементов новой матрицы \mathbf{A} , которая включает в себя все ограничения, принимающие в точке $\mathbf{x}^{(k+1)}$ вид равенств. В дальнейшем работа алгоритма повторяется.

Условие 2: $\mathbf{P}_l \nabla f(\mathbf{x}^{(k)}) = 0$.

Из формулы (6.3.1а) следует, что $\nabla f(\mathbf{x}^{(k)})$ можно выразить через \mathbf{A}_l^T и \mathbf{u} , а именно

$$\nabla f(\mathbf{x}^{(k)}) = \mathbf{A}_l^T \mathbf{u}. \quad (6.3.6)$$

Умножая (6.3.6) слева на $(\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l$, получаем

$$(\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l \nabla f(\mathbf{x}^{(k)}) = \mathbf{u}.$$

Если $u_i \geq 0$ для всех значений $i = 1, \dots, l$, то $\mathbf{x}^{(k)}$ есть искомое решение. В противном случае подбираем такое $u_m < 0$, для которого значение $\|a_m\| u_m$ отрицательно¹⁾ и максимально по модулю, пренебрегаем соответствующей гиперплоскостью, вычеркиваем в матрице \mathbf{A}_l m -ю строку и возвращаемся к соотношению (6.3.5). Новая проектирующая матрица \mathbf{P}_{l-1} удовлетворяет условию

$$\mathbf{P}_{l-1} \nabla f(\mathbf{x}^{(k)}) = \mathbf{P}_{l-1} \mathbf{A}_l^T \mathbf{u} \neq 0.$$

Таким образом, на каждом этапе вычислительного процесса возможны следующие варианты: 1) совокупность активных ограничений может остаться без изменений; 2) в дополнение к имевшим место может добавиться одно активное ограничение; 3) из совокупности активных ограничений одно ограничение может быть исключено. Случай, когда условию 2 удовлетворяют два или более активных ограничения, называется *вырожденным*. Подробное обсуждение вопросов, связанных с вырождением, можно найти у Кюнци [18] или у Флетчера [19]. В этих работах описываются эффективные способы нахождения $(\mathbf{A}_{l+1} \mathbf{A}_{l+1}^T)^{-1}$ и $(\mathbf{A}_{l-1} \mathbf{A}_{l-1}^T)^{-1}$ при заданной матрице $(\mathbf{A}_l \mathbf{A}_l^T)^{-1}$; методика вычисления $(\mathbf{A}_{l+1} \mathbf{A}_{l+1}^T)^{-1}$ и $(\mathbf{A}_{l-1} \mathbf{A}_{l-1}^T)^{-1}$ рассматривается также в разд. 6.3.3.

6.3.2. ОБОБЩЕННЫЙ ГРАДИЕНТНЫЙ МЕТОД ОПТИМИЗАЦИОННОГО ПОИСКА

Программа, реализующая обобщенный градиентный метод оптимизационного поиска, разработанная Кроссом и Кефартом (Union Carbide Corp., Oak Ridge, Tenn., USA), позволяет одновременно оперировать как линейными, так и нелинейными ограничениями, записанными в виде равенств и неравенств. Эта программа осуществляет оптимизационный поиск методом наискорейшего спуска внутри

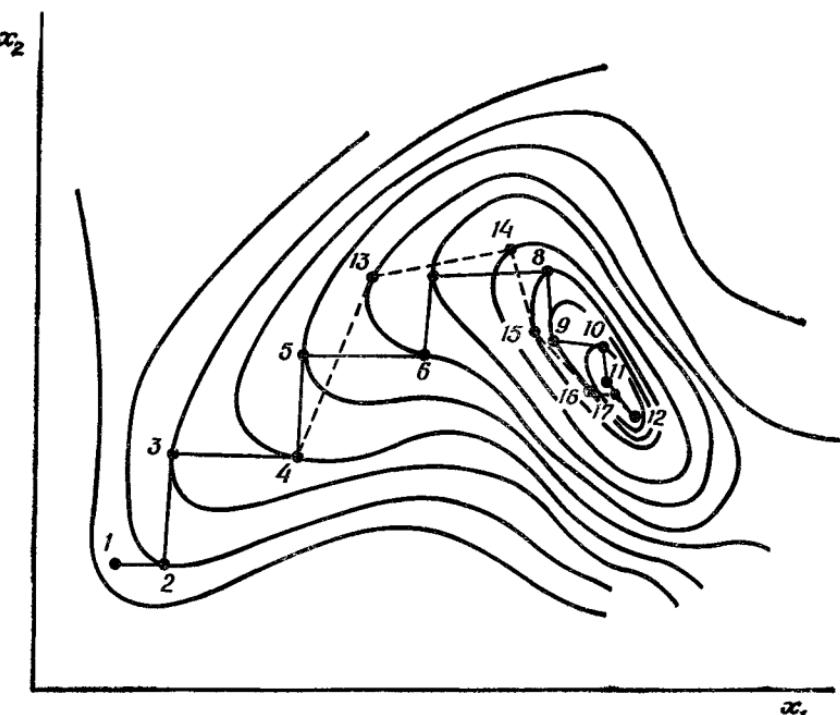
¹⁾ Символ $\| \cdot \|$ здесь используется для обозначения нормы вектора a_m .

допустимой области; при этом численная аппроксимация частных производных целевой функции (по формуле упреждающей разности) и длина шага в заданном направлении на $(k+1)$ -м этапе являются функциями числа результативных шагов на k -м этапе вычислительного цикла. При наличии нетривиальных ограничений (т. е. ограничений, не имеющих вид $L_j \leq x_j \leq U_j$) после их линеаризации применяется метод проекции градиента. При этом необходимо иметь записанные в явном виде аналитические выражения для частных производных функций, задающих нетривиальные ограничения. В окрестности ограничения используется параболическая аппроксимация вектора-градиента, что позволяет определить (приближенно) точку касания ограничивающей гиперплоскости с поверхностью данного ограничения. Машинная программа содержит также ряд специализированных стратегий для манипулирования с «хребтами» и тривиальными ограничениями, а также для ускорения сходимости промежуточных точек к оптимальному решению. В процессе оптимизации для перехода от недопустимой начальной (стартовой) точки к допустимой используются проективные методы.

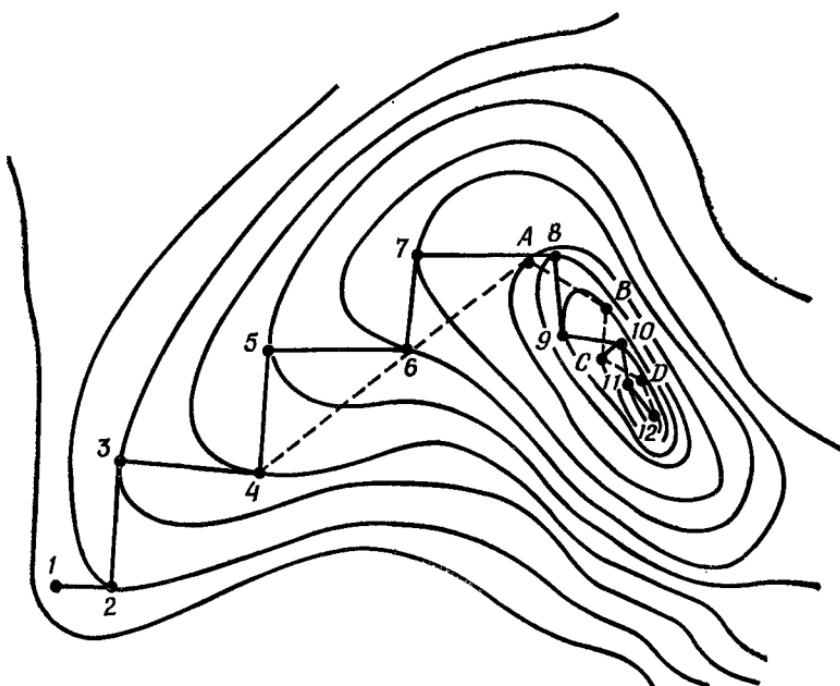
На фиг. 6.3.4 показана полученная градиентным методом траектория оптимизационного поиска в области оврага (являющегося эквивалентом хребта в задаче максимизации). Наличие оврага можно установить, обнаружив изменение знака составляющих градиента целевой функции (например, изменение знака градиента целевой функции по переменной x_2). Чтобы ускорить оптимизационный поиск в области оврага, в точке $\mathbf{x}^{(4)}$ длина шага при перемещении по x_2 сокращается до 0,64 его первоначальной длины, так что траектория проходит по ломаной, представленной на графике пунктирными линиями (не совпадающей с ломаной, представленной сплошными линиями). Если после первого сокращения длины шага знак частной производной продолжает «осциллировать», следует провести дальнейшее уменьшение шага. Если же осцилляция знака прекращается, то длина шага в направлении x_2 умножается на 1,25.

Другой способ ускорения сходимости, используемый при формировании вычислительного процесса на «градиентной» стадии оптимизационного поиска, заключается в применении метода параллельных касательных (см. разд. 3.3.3), с помощью которого после прохождения точек 2 и 4 на фиг. 6.3.5 удается попасть в точку A, минуя точки 5—8.

Когда нарушенным оказывается тривиальное ограничивающее условие (т. е. условие $L_j \leq x_j \leq U_j$), производится сокращение длины шага путем умножения текущего значения длины шага на отношение разности между значением $x_j^{(k)}$ и граничным значением x_j к длине шага, фактически реализованного в направлении x_j . В ситуациях, когда условие $L_j \leq x_j \leq U_j$ нарушается сразу для двух или более переменных, длина шага сокращается в такой



Ф и г. 6.3.4. Корректировка траектории оптимизационного поиска на этапе использования метода проекции градиента.



Ф и г. 6.3.5. Использование партан-метода на некотором этапе использования метода проекции градиента.

степени, чтобы удовлетворить ближайшему ограничивающему условию. Если проекция осуществляется на ограничивающее условие, имеющее вид неравенства, используется другая вычислительная процедура. Составляющая $\frac{\partial f(x^{(k)})}{\partial x_j}$ приравнивается нулю, а при проектировании градиента целевой функции переменная x_j исключается из рассмотрения путем приравнивания нулю $\frac{\partial g_j(x^{(k)})}{\partial x_j}$ для каждого j -го ограничения, принимающего при проектировании градиента вид строгого равенства. Таким образом, операция проектирования выполняется в подпространстве, в котором j -я координата отсутствует.

Когда нарушаются нетривиальные, причем нелинейные, ограничения, для нахождения следующей допустимой точки может потребоваться несколько последовательных итераций. Пусть Δx есть n -мерный вектор-столбец, переводящий вектор x , не являющийся допустимым, в допустимый вектор $x^{(k+1)}$, а Δb представляет собой p -мерный вектор-столбец, показывающий, какие изменения требуется произвести в ограничениях задачи, чтобы они оказались удовлетворенными. В линейном приближении

$$\Delta b = A \Delta x, \quad (6.3.7)$$

где A — матрица первых частных производных по x_j функций, задающих ограничения. Приращение вектора x (т. е. Δx) находится путем умножения A^T на p -мерный вектор $\gamma: \Delta x = A^T \gamma$, так что $\Delta b = AA^T \gamma$. Следовательно, поскольку Δb и A известны,

$$\gamma = (AA^T)^{-1} \Delta b. \quad (6.3.8)$$

Фактически реализуемый шаг находится из соотношения

$$x_j^{(k+1)} = x_j^{(k)} + (\Delta x)^T \left(\frac{\partial h}{\partial x_j} \right), \quad (6.3.9)$$

где h — та часть матрицы, которая ассоциируется с активными ограничениями.

Чтобы знать, какие из ограничений в виде неравенств следует в каждой текущей точке рассматривать как равенства в дополнение к исходной совокупности ограничений в виде равенств для каждого ограничения, записанного в виде неравенства, находится скалярное произведение градиента целевой функции (или отрицательного градиента в случае минимизации) и единичной нормали n_i к соответствующей ограничивающей поверхности:

$$\nabla^T f(x^{(k)}) n_i = \nabla^T f(x^{(k)}) \frac{\nabla g_i(x^{(k)})}{\| \nabla g_i(x^{(k)}) \|}.$$

При этом ограничение, соответствующее наибольшему значению такого скалярного произведения, добавляется к первоначальной совокупности ограничений-равенств также в виде равенства. Затем

$\nabla f(\mathbf{x}^{(k)})$ проектируется на полученное таким способом множество ограничений, после чего этот вычислительный цикл повторяется до тех пор, пока проекция градиента на пересечение ограничений не определит допустимое направление. Так, например, изображенное на фиг. 6.3.6, а скалярное произведение $\nabla f(\mathbf{x}^{(k)})$ и единичной нормали в точке $\mathbf{x}^{(k)}$ указывает на то, что к множеству активных ограничений следует присоединить ограничение, представленное функцией $g_2(\mathbf{x}^{(k)})$. Поскольку проекция $\nabla f(\mathbf{x}^{(k)})$ на $g_2(\mathbf{x}^{(k)}) = 0$ определяет допустимое направление, ограничение, задаваемое функцией $g_1(\mathbf{x}^{(k)})$, в систему активных ограничений включать не следует. Таким образом, несмотря на то что $\nabla f(\mathbf{x}^{(k)})$ может нарушать сразу несколько ограничений-неравенств, не всегда возникает необходимость проектировать $\nabla f(\mathbf{x}^{(k)})$ на все нарушающие ограничения. Если при перемещении в допустимом направлении встречается другое ограничение, то его добавляют к системе активных ограничений, после чего вновь ищется проекция $\nabla f(\mathbf{x})$. Исключение ограничений из группы ограничений в виде равенств осуществляется на стадии, предшествующей вычислению нового градиента целевой функции.

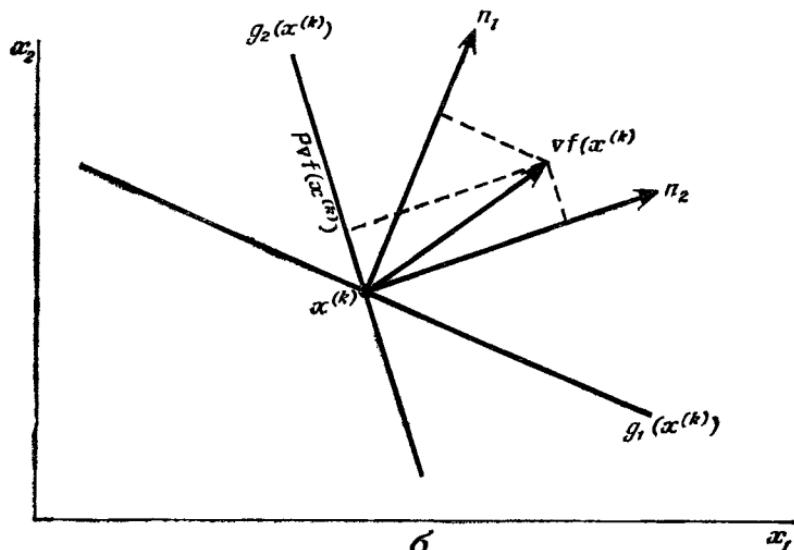
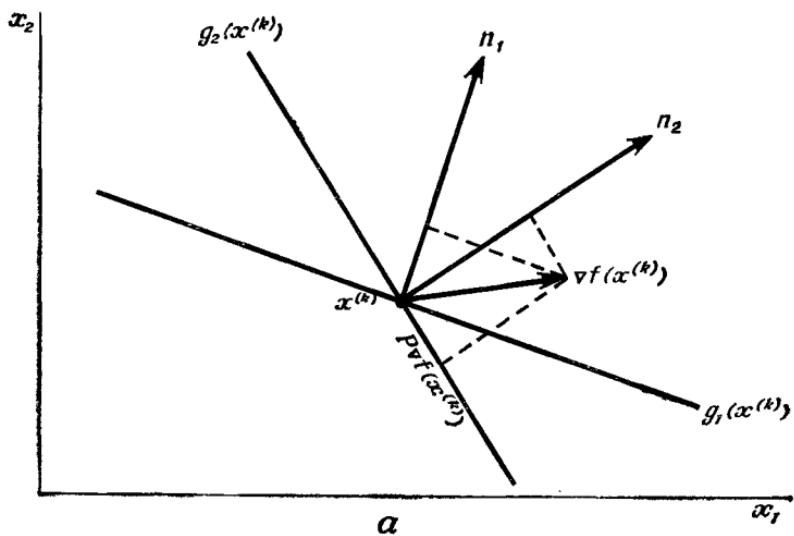
На фиг. 6.3.6, б показано, что происходит в оптимальной точке $\mathbf{x}^{(k)}$. Как и в случае, иллюстрированном на фиг. 6.3.6, а, в группу активных ограничений вводится ограничение, задаваемое функцией $g_2(\mathbf{x}^{(k)})$, поскольку

$$\nabla^T f(\mathbf{x}^{(k)}) \frac{\nabla g_2(\mathbf{x}^{(k)})}{\|\nabla g_2(\mathbf{x}^{(k)})\|} > \nabla^T f(\mathbf{x}^{(k)}) \frac{\nabla g_1(\mathbf{x}^{(k)})}{\|\nabla g_1(\mathbf{x}^{(k)})\|}.$$

Однако при этом проекция $\nabla f(\mathbf{x}^{(k)})$ на $g_2(\mathbf{x}^{(k)})$ нарушает ограничение, представленное функцией $g_1(\mathbf{x}^{(k)})$, и, следовательно, не является допустимым вектором. Поэтому $g_1(\mathbf{x}^{(k)})$ добавляется к системе активных ограничений, и $\nabla f(\mathbf{x}^{(k)})$ проектируется на пересечение ограничений, представленных функциями $g_1(\mathbf{x}^{(k)})$ и $g_2(\mathbf{x}^{(k)})$, что дает нуль-вектор. Оптимизирующий поиск заканчивается, если на двух последовательных этапах вычислительного процесса не удалось определить такие перемещения в пространстве решений, которые вносили бы минимизирующую поправку в значение целевой функции.

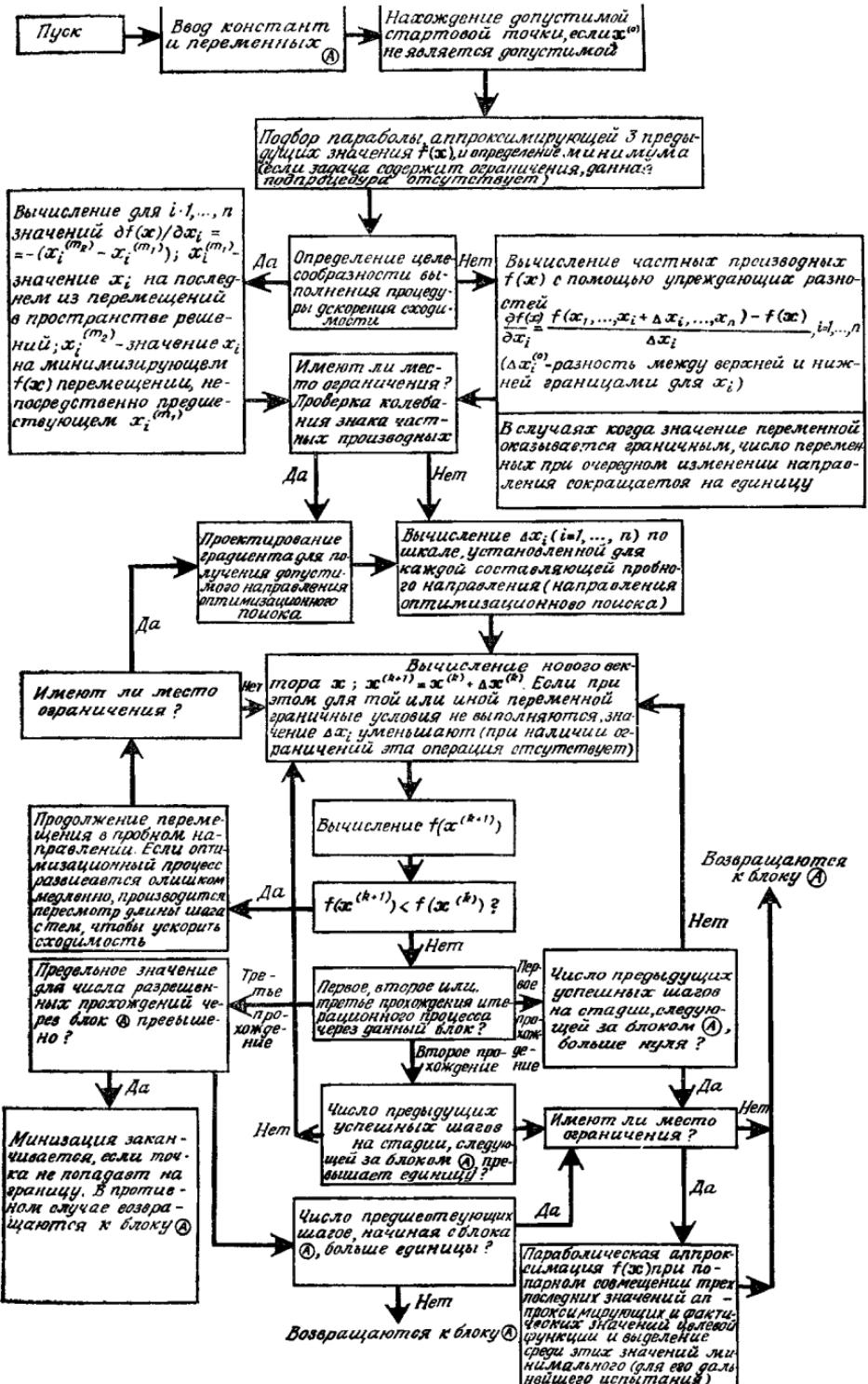
Сравнение обобщенного градиентного метода (основные элементы которого представлены на фиг. 6.3.7) с другими алгоритмами оптимизации проводится в гл. 9. В работе Кросса [20] отмечается, что обобщенным градиентным методом решались задачи нелинейного программирования, содержащие около 50 ограничений при таком же количестве переменных.

Одна из трудностей, с которой сопряжено применение рассмотренного выше метода, состоит в том, что значения элементов



Ф и г. 6.3.6. Включение ограничений в виде неравенств в систему активных ограничений при оптимизационном поиске обобщенным градиентным методом с последующим применением метода проекции градиента,

a — неоптимальная точка; *b* — оптимальная точка.



Фиг. 6.3.7. Блок-схема вычислительного процесса на основе обобщенного градиентного метода

матрицы \mathbf{A} представляют собой лишь аппроксимации их истинных значений, так как частные производные фигурирующих в задаче функций вычисляются в точках, не являющихся допустимыми. Поэтому в тех случаях, когда в ходе итерации из-за слишком грубых нарушений ограничений не удается получить допустимую точку с точностью до установленного ранее допуска, после определенного числа итераций алгоритм вычисляет новый градиент и уменьшает длину шага с тем, чтобы найти в результате допустимую точку. Другая трудность, возникающая в случае нелинейных ограничений, заключается в том, что поверхности линеаризованных ограничений могут в какой-нибудь точке совпадать с граничной поверхностью. Тогда матрица \mathbf{AA}^T становится вырожденной. Вырожденность матрицы \mathbf{AA}^T может быть обусловлена также дополнительным введением слишком большого числа ограничивающих поверхностей в имеющуюся уже систему активных ограничений, так что локальное решение может оказаться «избыточно обусловленным». Вырождение устраняется путем замены j -го диагонального элемента матрицы \mathbf{AA}^T на единицу всякий раз, когда этот элемент равняется нулю. Поскольку j -я составляющая вектора $\mathbf{A}\Delta f(\mathbf{x})$ в таких случаях также равна нулю, параметр λ_j оказывается равным нулю независимо от наличия и вида j -го ограничения. Если перемещение в новом направлении не улучшает значение $f(\mathbf{x})$ (или $\mathbf{P}\nabla f(\mathbf{x}) = 0$), ищется новый градиент $\nabla f(\mathbf{x})$ и реализуется описанная выше операция проектирования $\nabla f(\mathbf{x})$ на подмножество активных ограничений.

Пример 6.3.1. Обобщенный градиентный метод

Рассмотрим следующую задачу:

минимизировать $f(\mathbf{x}) = 4x_1 - x_2^2 - 12$
при ограничениях

$$h_1(\mathbf{x}) = 25 - x_1^2 - x_2^2 = 0,$$

$$g_2(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

Фигурирующие в данной задаче функции изображены графически на фиг. 6.0.1.

Начнем итерационный процесс с недопустимой (но являющейся внутренней) точки $\mathbf{x}^{(0)} = [2 \ 4]^T$ и наложим нижние ограничения $(0 \ 0)$ и произвольные верхние ограничения $(10^6 \ 10^6)$ на x_1 и x_2 соответственно. Частные производные целевой функции и функций,

задающих нетривиальные ограничения, имеют следующий вид:

$$\begin{aligned}\frac{\partial f(\mathbf{x})}{\partial x_1} &= 4, & \frac{\partial f(\mathbf{x})}{\partial x_2} &= -2x_2, \\ \frac{\partial h_1(\mathbf{x})}{\partial x_1} &= -2x_1, & \frac{\partial h_1(\mathbf{x})}{\partial x_2} &= -2x_2, \\ \frac{\partial g_2(\mathbf{x})}{\partial x_1} &= 10 - 2x_1, & \frac{\partial g_2(\mathbf{x})}{\partial x_2} &= 10 - 2x_2.\end{aligned}$$

Значения частных производных в начальной точке (в приведенном выше порядке) запишем в виде матрицы

$$\begin{bmatrix} 4 & -8 \\ -4 & -8 \\ 6 & 2 \end{bmatrix}.$$

Вычислим прежде всего матрицу \mathbf{A} (в точке $\mathbf{x}^{(0)} = [2 \ 4]^T$ активным является лишь ограничение $h_1(\mathbf{x}) = 0$), воспользовавшись соотношением

$$a_{ik}^{(k)} = \sum_{j=1}^n \left(\frac{\partial h_i(\mathbf{x}^{(k)})}{\partial x_j} \right) \left(\frac{\partial h_k(\mathbf{x}^{(k)})}{\partial x_j} \right), \quad i, k = 1, \dots, n^*,$$

где n^* — суммарное число активных ограничений из числа ограничений, записанных в исходной формулировке задачи в виде неравенств, и ограничений, имеющих по условию задачи вид равенств. В рассматриваемом случае (при $n^* = 1$)

$$\mathbf{A}^{(0)} = a_{11}^{(0)} = -4(-4) + [-8(-8)] = 80.$$

После этого вычисляется матрица $\Delta \mathbf{b}$, которая показывает, какие приращения должны получить фигурирующие в задаче функции, задающие ограничения (в виде равенств и активных неравенств), чтобы соответствующие ограничения удовлетворялись при «старте» из точки $\mathbf{x}^{(0)} = [2 \ 4]^T$; в рассматриваемом случае $\Delta \mathbf{b}$ содержит всего один элемент

$$b_1^{(0)} = -5,$$

так как значение $h_1(\mathbf{x})$ в точке $\mathbf{x} = [2 \ 5]^T$ равняется пяти. Из формулы (6.3.7) следует, что

$$\Delta \mathbf{x} = \mathbf{A}^{-1} \Delta \mathbf{b}.$$

Таким образом, $\Delta \mathbf{x} = 1/80(-5) = -0,0625$.

На первом шаге ($k = 1$) вектор $\mathbf{x}^{(1)}$ находится с помощью формулы (6.3.9):

$$x_j^{(1)} = x_j^{(0)} + (\Delta x) \frac{\partial h_1(\mathbf{x}^{(0)})}{\partial x_j},$$

$$x_1^{(1)} = 2,00 - 0,0625(-4) = 2,25,$$

$$x_2^{(1)} = 4,00 - 0,0625(-8) = 4,50.$$

Последовательно реализуя шаги итерационного процесса, получаем (числа округлены):

Порядковый номер шага k	x_1	x_2	$\frac{\partial h_1(x^{(k)})}{\partial x_1}$	$\frac{\partial h_1(x^{(k)})}{\partial x_2}$	$h(x^{(k)})$	Δx
1	2,250	4,500	-4,500	-9,000	$-3,125 \cdot 10^{-1}$	$3,083 \cdot 10^{-3}$
2	2,236	4,472	-4,472	-8,944	$-9,645 \cdot 10^{-4}$	$9,644 \cdot 10^{-6}$
3	2,236	4,472	-4,472	-8,944	$-9,302 \cdot 10^{-9}$	

После третьего шага условие допуска для $h_1(x^{(k)})$ оказывается выполненным и значения независимых переменных последовательно смещаются на величину $\alpha(U_j - L_j)$, где $\alpha = 0,005$ (значение этой константы можно выбирать произвольным образом по усмотрению пользователя), а U_j и L_j ($j = 1, 2$) представляют собой верхние и нижние предельные значения (границы) для x_1 и x_2 соответственно. В рассматриваемом примере $\alpha(U_j - L_j) = 0,005 \cdot 10^6 = 5 \cdot 10^3$ как для x_1 , так и для x_2 , так что смещения текущих значений этих переменных имеют следующий вид:

$x_1^{(3)} + \delta x_1$	$x_2^{(3)} + \delta x_2$	$f(x)$ или $f(x + \delta x)$
$2,236 + 0$	$4,472 + 0$	$-23,05$
$2,236 + 5 \cdot 10^3$	$4,472 + 0$	$1,99 \cdot 10^4$
$2,136 + 0$	$4,472 + 5 \cdot 10^3$	$-2,50 \cdot 10^7$

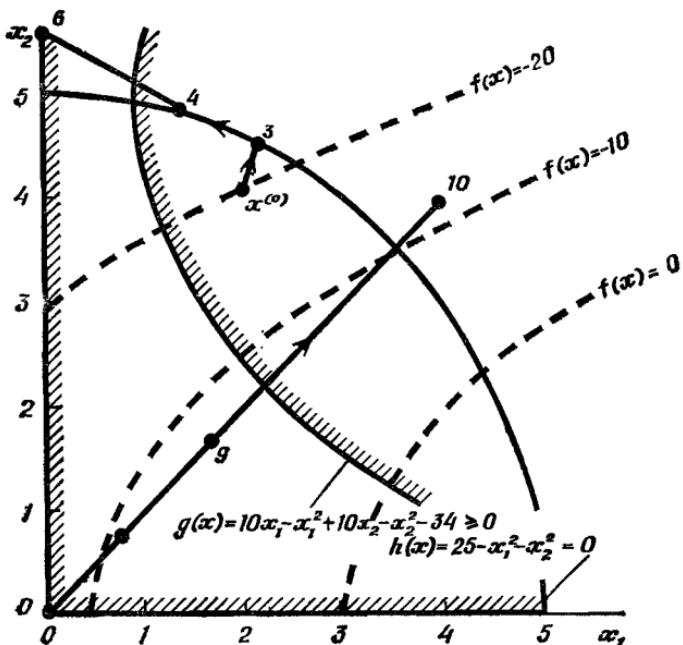
Эти смещения используются при вычислении новых составляющих градиента функции $f(x)$:

$$\frac{\partial f(x)}{\partial x_j} \approx \frac{f(x_1, x_2, \dots, x_j + \delta x_j, \dots, x_n) - f(x)}{\delta x_j},$$

$$\frac{\partial f(x^{(3)})}{\partial x_1} \approx \frac{1,99 \cdot 10^4 - (-23,05)}{5 \cdot 10^3} = 4,00,$$

$$\frac{\partial f(x^{(3)})}{\partial x_2} \approx \frac{-2,50 \cdot 10^7 - (-23,05)}{5 \cdot 10^8} = -5,009 \cdot 10^3.$$

(Старые значения составляющих градиента целевой функции заменяются на вновь вычисленные.) Следовательно, на шаге 4 ($k = 4$) поиск осуществляется сравнительно большими перемещениями и на шаге 6 приводит к вектору $x^{(6)} = [0 \ 5,590]^T$, для которого $h_1(x) = -6,250$, а $g_2(x) = -9,348$.



Фиг. П.6.3.1. Траектория оптимизационного поиска обобщенным градиентным методом.

Теперь, поскольку условие $g_2(\mathbf{x}) \geq 0$ не удовлетворяется, наряду с ограничением $h_1 = 0$ необходимо учитывать также и ограничение, заданное функцией $g_2(\mathbf{x})$. При этом имеем

$$\mathbf{A} = \begin{bmatrix} 1,250 \cdot 10^2 & 1,319 \cdot 10^1 \\ 1,319 \cdot 10^1 & 1,393 \end{bmatrix}, \quad \Delta \mathbf{b} = \begin{bmatrix} 6,250 \\ 9,348 \end{bmatrix},$$

$$\Delta \mathbf{x} = \begin{bmatrix} -6,545 \cdot 10^{13} \\ 6,113 \cdot 10^{14} \end{bmatrix}.$$

На следующем шаге матрица \mathbf{A} превращается в единичную матрицу. Для ряда последующих шагов итерационного процесса получаем

Номер шага k	$x_1^{(k)}$	$x_2^{(k)}$
7	0	0
8	0	0
9	1,70	1,70
10	4,00	4,00

На десятом шаге ($k = 10$) матрица A , к сожалению, становится вырожденной:

$$A = \begin{bmatrix} 128 & -32 \\ -32 & 8 \end{bmatrix},$$

так что работа алгоритма преждевременно прекращается. Траектория оптимизационного поиска показана на фиг. П.6.3.1.

6.3.3. ОБОБЩЕННЫЙ МЕТОД ДЭВИДОНА

Прежде всего изложим методы сопряженных направлений и переменной метрики при наличии линейных ограничений, а затем покажем, каким образом можно подойти к решению проблемы согласования и учета нелинейных ограничений. В данном подразделе в основном используются результаты Дэвидона [21], который впервые сформулировал существо метода, результаты Голдфарба [22, 23], проанализировавшего метод с использованием алгебры матриц, а также работы Дэвиса [24] и Муртага и Сарджента [25], которым удалось модифицировать метод Дэвидона и разработать машинную программу, позволяющую реализовать предлагаемый метод на ЭВМ. Хотя и предполагается, что на каждом этапе оптимизационного процесса для определения направлений пробного поиска работает алгоритм Дэвидона — Флетчера — Пауэлла или алгоритм Брайдена, следует иметь в виду, что вместо них можно успешно использовать почти любой из методов, описание которых дано в гл. 3.

Дэвидон в своей первой статье по данному вопросу утверждал, что его метод можно распространить на случай нелинейного программирования при линейных ограничениях как в виде равенств, так и в виде неравенств путем понижения ранга матрицы $\eta^{(k)}$ на единицу для каждого активного ограничения и переопределения $\eta^{(k)}$ с тем, чтобы каждая составляющая вектора $x^{(k+1)} = x^{(k)} + \Delta x^{(k)}$ удовлетворяла всем активным ограничениям [см. (3.4.1)]. Используя при этом подход, представленный соотношениями (6.3.2) и (6.3.3), можно определить так называемую обобщенную проектирующую матрицу. Обозначим через x_M ортогональную проекцию вектора x [26], связанную с данной матрицей Q :

$$x_M = \hat{P}_I x, \quad (6.3.10)$$

где матрица \hat{P}_I определяется соотношением

$$\hat{P}_I = I - A_I^T (A_I Q A_I^T)^{-1} A_I Q. \quad (6.3.11)$$

Сравнивая это выражение с соотношениями (6.3.2) и (6.3.3), мы видим, что в последних $\mathbf{Q} = \mathbf{I}$. Кроме того,

$$\mathbf{x}_M^T \mathbf{Q} (\mathbf{x} - \mathbf{x}_M) = 0.$$

Таким образом, $\hat{\mathbf{P}}_l$ есть обобщенная проектирующая матрица, позволяющая проектировать \mathbf{x} на многообразие M в соответствии с метрикой \mathbf{Q} . Затем положим $\mathbf{Q} = \eta^{(k)}$, где $\eta^{(k)}$ — оценка обращенной матрицы Гессе, описание которой дано в разд. 3.4, посвященном оптимизации при отсутствии ограничений.

Напомним, что направление оптимизационного поиска в случае, когда задача нелинейного программирования не содержит ограничений, может быть найдено методом Дэвидона (см. разд. 3.4):

$$\mathbf{s}^{(k)} = -\eta_0^{(k)} \nabla f(\mathbf{x}^{(k)}).$$

Здесь $\eta_0^{(k)}$ есть η в точке $\mathbf{x}^{(k)}$ при отсутствии ограничений. Если имеется l активных ограничений ($A_l \mathbf{x} = b$), то, как уже установлено с помощью (6.3.4),

$$\mathbf{s}^{(k)} = -\hat{\mathbf{P}}_l \nabla f(\mathbf{x}^{(k)}).$$

Один из способов комбинированного использования понятия проектирующей матрицы и метода Дэвидона позволяет определить направление оптимизационного поиска в виде

$$\mathbf{s}^{(k)} = -\eta_0^{(k)} [\hat{\mathbf{P}}_l \nabla f(\mathbf{x}^{(k)})] = -\eta^{(k)} \nabla f(\mathbf{x}^{(k)}), \quad (6.3.12)$$

где $\hat{\mathbf{P}}_l$ задается соотношением (6.3.11)¹⁾ при $\mathbf{Q} = \eta_0^{(k)}$, а

$$\eta^{(k)} = \eta_0^{(k)} \hat{\mathbf{P}}_l = \eta_0^{(k)} - \eta_0^{(k)} A_l^T (A_l \eta_0^{(k)} A_l^T)^{-1} A_l \eta_0^{(k)}. \quad (6.3.13)$$

В принятых здесь обозначениях $\eta_l^{(k)}$ есть матрица η в точке $\mathbf{x}^{(k)}$ при l активных ограничениях, имеющих место в точке $\mathbf{x}^{(k)}$. Например, накладывая ограничение $a_j^T \mathbf{x} = b$ на целевую функцию задачи без ограничений, с помощью (6.3.13) получаем

$$\eta_l^{(k)} = \eta_0^{(k)} - \frac{\eta_0^{(k)} a_j a_j^T \eta_0^{(k)}}{a_j^T \eta_0^{(k)} a_j}. \quad (6.3.13a)$$

Голдфарб показал, что в случае квадратичной целевой функции n переменных при l ограничениях в виде равенств использование

¹⁾ Если определить $\mathbf{s}^{(k)}$ с помощью соотношения $\mathbf{s}^{(k)} = -\hat{\mathbf{P}}_l^* [\eta_0^{(k)} \nabla f(\mathbf{x}^{(k)})]$, в котором предполагается, что проектирующая матрица проектирует шаг в условиях соответствующей задачи без ограничений на ограничивающую поверхность исходной задачи, то $\hat{\mathbf{P}}_l^*$ запишется (по Голдфарбу) в виде

$$\hat{\mathbf{P}}_l^* = \mathbf{I} - \mathbf{Q} \mathbf{A}_l^T (\mathbf{A}_l \mathbf{Q} \mathbf{A}_l^T)^{-1} \mathbf{A}_l,$$

для определения направлений оптимизационного поиска соотношения (6.3.12) в комбинации с поэтапной линеаризацией обеспечивает сходимость последовательности промежуточных решений к оптимальному за $(n - l)$ итераций. Матрица η обновляется (корректируется) способом, предложенным Брайденом (алгоритм единичного понижения ранга); при этом следует иметь в виду, что эффективность рассматриваемого здесь метода связана с тем, что главная задача состоит не в нахождении проектирующей матрицы \hat{P}_l (или \hat{P}_l^*) в явном виде на каждом этапе вычислительной процедуры, а в определении поправок к \hat{P}_l по отношению к предыдущему этапу.

Если имеется одно или несколько ограничений, которые в ходе оптимизационного поиска в заданном направлении оказываются нарушенными, то эти ограничения добавляются к совокупности ограничений в виде равенств и учитываются при формировании $\eta^{(k)}$. С другой стороны, ограничения, записанные в исходной формулировке задачи в виде неравенств, могут исключаться из системы активных ограничений, если направление оптимизационного поиска оказывается таким, что эти ограничения из группы m ограничений-неравенств вида

$$\mathbf{B}_m \mathbf{x} \geq \mathbf{c}$$

при этом более не нарушаются. Голдфарб установил следующее соотношение, позволяющее исключать ограничение в виде неравенства из группы, состоящей из l активных ограничений:

$$\eta_{l-1}^{(k)} = \eta^{(k)} + \frac{\mathbf{P}_{m-1} \mathbf{b}_j^T \mathbf{b}_j \hat{\mathbf{P}}_{m-1}}{\mathbf{b}_j \mathbf{P}_{m-1} \mathbf{b}_j^T}, \quad (6.3.14)$$

где $\mathbf{b}_j = [b_{j1} \ b_{j2} \ \dots \ b_{jn}]$, т. е. \mathbf{b}_j представляет собой строку матрицы \mathbf{B}_m , соответствующую j -му ограничению, подлежащему исключению из совокупности активных ограничений, а \mathbf{P}_{m-1} находится с помощью (6.3.3) при исключении j -го ограничения в виде неравенства из множества m ограничений-неравенств (являющихся на предыдущем шаге активными):

$$\mathbf{P}_{m-1} = \mathbf{I} - \mathbf{B}_{m-1}^T (\mathbf{B}_{m-1} \mathbf{B}_{m-1}^T)^{-1} \mathbf{B}_{m-1}. \quad (6.3.15)$$

Теперь остается лишь ответить на следующий вопрос: как можно определить, какое из ограничений в виде неравенств становится активным в точке $\mathbf{x}^{(k)}$? Как уже отмечалось в разд. 6.3.1, это позволяют сделать так называемые теневые параметры, т. е. составляющие вектора \mathbf{u} , определение которого дано в связи с рассмотрением соотношения (6.3.4):

$$\mathbf{u} = (\mathbf{A}_l \mathbf{A}_l^T)^{-1} \mathbf{A}_l \nabla f(\mathbf{x}^{(k)}).$$

Отрицательное значение u_j соответствует ограничению, которое можно исключить из базисной совокупности ограничений, причем для этого выбирается ограничение с наибольшим абсолютным значением $|u_j|$. Поскольку из совокупности активных ограничений можно исключить лишь ограничения в виде неравенств (ограничения, записанные в виде равенств, всегда активны), можно избежать вычисления u , ограничившись вычислением лишь вектора

$$\tilde{u} = (\mathbf{B}_m \mathbf{B}_m^T)^{-1} \mathbf{B}_m \nabla f(\mathbf{x}^{(k)}). \quad (6.3.16)$$

Кроме того, как отмечалось в разд. 6.3.1, существуют более эффективные способы вычисления $(\mathbf{B}_m \mathbf{B}_m^T)^{-1}$ при заданных $(\mathbf{B}_{m+1} \mathbf{B}_{m+1}^T)^{-1}$ и $(\mathbf{B}_{m-1} \mathbf{B}_{m-1}^T)^{-1}$, нежели прямой способ перемножения матриц с последующим выполнением операции нахождения обратной матрицы.

Используя $\eta_i^{(k)}$, \hat{P}_i и \tilde{u} , определение которых дано выше, можно рассматривать метод Дэвидона как метод решения задач нелинейного программирования при линейных ограничениях. Начальная допустимая точка $\mathbf{x}^{(0)}$ предполагается известной; в противном случае для того, чтобы алгоритм начал работать, ее нужно отыскать. Если $\mathbf{x}^{(0)}$ — внутренняя точка R и ограничения в виде равенств отсутствуют, $\eta_0^{(0)}$ принимается равной I . Если $\mathbf{x}^{(0)}$ лежит на l линейно независимых гиперплоскостях (т. е. удовлетворяет l ограничениям в виде равенств), то $\eta_0^{(0)}$ находится путем l -кратного применения (6.3.13а) при начальном условии $\eta_0(0) = I$. Таким образом, множество независимых ограничений в виде равенств оказывается включенным в исходную совокупность базисных ограничений, и, следовательно, только ограничения в виде неравенств либо вводятся в базис, либо из него исключаются.

Предполагается, что на k -м этапе оптимизационного поиска мы знаем $\mathbf{x}^{(k)}$, $f(\mathbf{x}^{(k)})$, $\nabla f(\mathbf{x}^{(k)})$ и $\eta_l^{(k)}$; будем также считать, что $\mathbf{x}^{(k)}$ принадлежит R и многообразию M , задаваемому пересечением l линейно независимых гиперплоскостей, представляющих активные ограничения (базисную совокупность ограничений). Схема дальнейшего развития вычислительного процесса, предложенная Дэвидом, включает следующие операции:

1. Выделение активных ограничений из числа ограничений, имеющих вид неравенств. Вычисляют $s^{(k)} = -\eta_l^{(k)}$, $\nabla f(\mathbf{x}^{(k)})$ и с помощью формулы (6.3.16) \tilde{u} . Если $\|s^{(k)}\| = 0$ и $\tilde{u}_j \geq 0$ при любом значении j , оптимум \mathbf{x}^* найден и работа алгоритма заканчивается; в противном случае осуществляется переход к шагу 2.

2. Перестройка базиса ограничений. Из базисной совокупности ограничений исключается то ограничение в виде неравенства, которое соответствует наибольшему по модулю отрицательному значению \tilde{u}_j ; эта операция выполняется с помощью формулы (6.3.14).

Вычисляется $s^{(k)} = -\eta_l^{(k)} \nabla f(\mathbf{x}^{(k)})$; затем, как и в случае применения алгоритма проекции градиента (алгоритма Розена), убеждаются в том, что вновь определенный вектор $s^{(k)}$ задает допустимую (по отношению ко всем неактивным ограничениям) точку. Если точка, задаваемая вектором $s^{(k)}$, не является допустимой, используют соотношение (6.3.13а) и прибавляют к базисной системе ограничений те ограничения в виде неравенств, которые при этом нарушаются. После этого с помощью формулы (6.3.12) находится новое значение $s^{(k)}$. Эта процедура выполняется (от начала рассматриваемого шага) столько раз, сколько необходимо для получения допустимого направления. Если для некоторого k справедливо равенство $\|s_{\text{допуст}}^{(k)}\| = 0$, возвращаются к шагу 1; в противном случае переходят к шагу 3.

3. Одномерный оптимизационный поиск. По схеме, описание которой дано в разд. 3.4 и 2.6, реализуется одномерный оптимизационный поиск в допустимом направлении. Если при единичном перемещении в процессе одномерного оптимизационного поиска нарушается одно из ограничений, переходят к шагу 5. В противном случае фиксируется точка, соответствующая наименьшему значению $f(\mathbf{x})$.

4. Переход к новой матрице. Если в процессе линейного оптимизационного поиска группа активных ограничений не меняется, осуществляется переход от матрицы $\eta_l^{(k)}$ к матрице $\eta_l^{(k+1)}$ с использованием при этом либо метода Дэвидона — Флетчера — Паузелла, либо метода Броддена (см. разд. 3.4). Затем возвращаются к шагу 1.

5. Формирование новой группы базисных ограничений. Группа базисных ограничений расширяется за счет введения в нее нового ограничения, а именно того, которое оказалось нарушенным на шаге 3. Затем возвращаются к шагу 1.

Для определения направления оптимизационного поиска Муртаг и Сарджент использовали проектирующую матрицу $\hat{\mathbf{P}}_l^*$, т. е. полагали

$$\begin{aligned}\Delta \mathbf{x}^{(k)} &= -\lambda^{(k)} \hat{\mathbf{P}}_l^{*(k)} \eta_0^{(k)} \nabla f(\mathbf{x}^{(k)}) = \\ &= -\lambda^{(k)} [\mathbf{I} - \eta_0^{(k)} \mathbf{A}_l^T (\mathbf{A}_l \eta_0^{(k)} \mathbf{A}_l^T)^{-1} \mathbf{A}_l] \eta_0^{(k)} \nabla f(\mathbf{x}^{(k)}) = \\ &= -\lambda^{(k)} \eta_0^{(k)} [\nabla f(\mathbf{x}^{(k)}) - \mathbf{A}_l^T \mathbf{u}_l^{(k)}],\end{aligned}$$

где $\mathbf{u}_l^{(k)} \equiv (\mathbf{A}_l \eta_0^{(k)} \mathbf{A}_l^T)^{-1} \mathbf{A}_l \eta_0^{(k)} \nabla f(\mathbf{x}^{(k)})$ можно рассматривать как вектор, составляющими которого являются ассоциированные с активными ограничениями множители Лагранжа; с их помощью можно определить, какие ограничения следует исключить из базиса. Выражение $[\nabla f(\mathbf{x}^{(k)}) - \eta_0^{(k)} \mathbf{u}_l^{(k)}]$ интерпретируется как разность между градиентом $f(\mathbf{x})$ в точке $\mathbf{x}^{(k)}$ и соответствующим градиентом $\eta_0^{(k)} \mathbf{u}_l^{(k)}$ в найденной стационарной точке.

Алгоритм Муртага — Сарджента весьма похож на алгоритм Дэвиса. На k -м этапе реализуются следующие шаги:

1. Вычисляется $\mathbf{u}_l^{(k)} = \mathbf{M}_l^{(k)} \mathbf{A}_l \boldsymbol{\eta}_0^{(k)} \nabla f(\mathbf{x}^{(k)})$, где $\mathbf{M}_l^{(k)}$ — аппроксимация $(\mathbf{A}_l \mathbf{A}_l^T)^{-1}$.

2. Вычисляется $\Delta \mathbf{x}^{(k)} = -\boldsymbol{\eta}_0^{(k)} [\nabla f(\mathbf{x}^{(k)}) - \mathbf{A}_l^T \mathbf{u}_l^{(k)}]$.

3. Вычисляется

$$\beta = \max_{\substack{j \mid u_j > 0 \\ j=1, \dots, l}} \left\{ \frac{\frac{1}{2} u_j}{m_{jj}} \right\},$$

где m_{jj} — j -й диагональный элемент матрицы $\mathbf{M}_l^{(k)}$.

4. Если $\|\Delta \mathbf{x}^{(k)}\| < \epsilon$ и $\beta < \epsilon$, вычислительный процесс завершается; в противном случае осуществляется переход к следующему шагу.

5. Вносятся изменения в базисную группу ограничений (если в этом имеется необходимость):

а) к базису добавляется некоторое ограничение. Если точка $\mathbf{x}^{(k)}$ оказывается на поверхности, связанной с новым базисным ограничением, то $g_{l+1}(\mathbf{x}^{(k)}) = 0$. Если же $(\Delta \mathbf{x}^{(k)})^T \nabla g_{l+1}(\mathbf{x}^{(k)}) > 0$, то в систему базисных ограничений вводят $g_{l+1}(\mathbf{x}^{(k)})$, а затем возвращаются к шагу 1;

б) некоторое ограничение исключается из базиса. Если $\|\Delta \mathbf{x}^{(k)}\| < \beta$, то выбранное j -е ограничение исключают из группы базисных ограничений, а затем возвращаются к шагу 1.

При включении в базис нового ограничения перестройка матрицы $\mathbf{M}_{l+1}^{(k)}$ осуществляется следующим образом:

$$\mathbf{M}_{l+1}^{(k)} = \begin{bmatrix} \mathbf{M}_{11} & \mathbf{M}_{12} \\ \mathbf{M}_{21} & \mathbf{M}_{22} \end{bmatrix},$$

где

$$\mathbf{M}_{11} = \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1} \mathbf{A}_{12} \mathbf{A}_0^{-1} \mathbf{A}_{21} \mathbf{A}_{11}^{-1} = \frac{\mathbf{M}_{12} \mathbf{M}_{12}^T}{\mathbf{M}_{22}},$$

$$\mathbf{M}_{12} = \mathbf{M}_{21}^T = -\mathbf{A}_{11}^{-1} \mathbf{A}_{12} \mathbf{A}_0^{-1},$$

$$\mathbf{M}_{22} = \mathbf{A}_0^{-1} \text{ (скаляр),}$$

$$\mathbf{A}_{11}^{-1} = \mathbf{M}_l^{(k)},$$

$$\mathbf{A}_{12} = \mathbf{A}_{21}^T = \mathbf{A}_l \boldsymbol{\eta}_0^{(k)} \nabla g_{l+1}(\mathbf{x}^{(k)}),$$

$$\mathbf{A}_{22} = \nabla g_{l+1}^T(\mathbf{x}^{(k)}) \boldsymbol{\eta}_0^{(k)} \nabla g_{l+1}(\mathbf{x}^{(k)}),$$

$$\mathbf{A}_0 = \mathbf{A}_{22} - \mathbf{A}_{21} \mathbf{A}_{11}^{-1} \mathbf{A}_{12} \text{ (скаляр).}$$

При исключении ограничения из базиса матрица $M_l^{(k)}$ заменяется матрицей

$$M_{l-1}^{(k)} = M_{11} - M_{12}M_{22}^{-1}M_{21}.$$

6. Осуществляется оптимизационный поиск в направлении проекции:

$$\Delta x^{(k)} = -\lambda \eta_0^{(k)} [\nabla f(x^{(k)}) - A_l^T u_l^{(k)}].$$

Параметр λ подбирается так, чтобы значение $f(x)$ уменьшалось, а вектор x оставался допустимым; при этом используется одномерный поиск.

7. Вычисляется $\Delta g^{(k)} = \nabla f(x^{(k+1)}) - \nabla f(x^{(k)})$.

8. Перестраивается матрица η при использовании метода Бройдена:

$$\eta_0^{(k+1)} = \eta_0^{(k)} + \frac{(\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)}) (\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})^T}{(\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})^T \Delta g^{(k)}}.$$

Осуществляется проверка с тем, чтобы убедиться в положительной определенности матрицы $\eta^{(k+1)}$. Если $\eta^{(k+1)}$ не является положительно определенной, $\eta_0^{(k+1)}$ приравнивается $\eta_0^{(k)}$.

9. Перестраивается матрица $M_l^{(k)}$:

$$M_l^{(k+1)} = M_l^{(k)} + \frac{[M_l^{(k)} A_l (\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})] [M_l^{(k)} A_l (\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})]^T}{(\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})^T \Delta g^{(k)} + [\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)}]^T A_l^T [M_l^{(k)} A_l (\Delta x^{(k)} - \eta_0^{(k)} \Delta g^{(k)})]}.$$

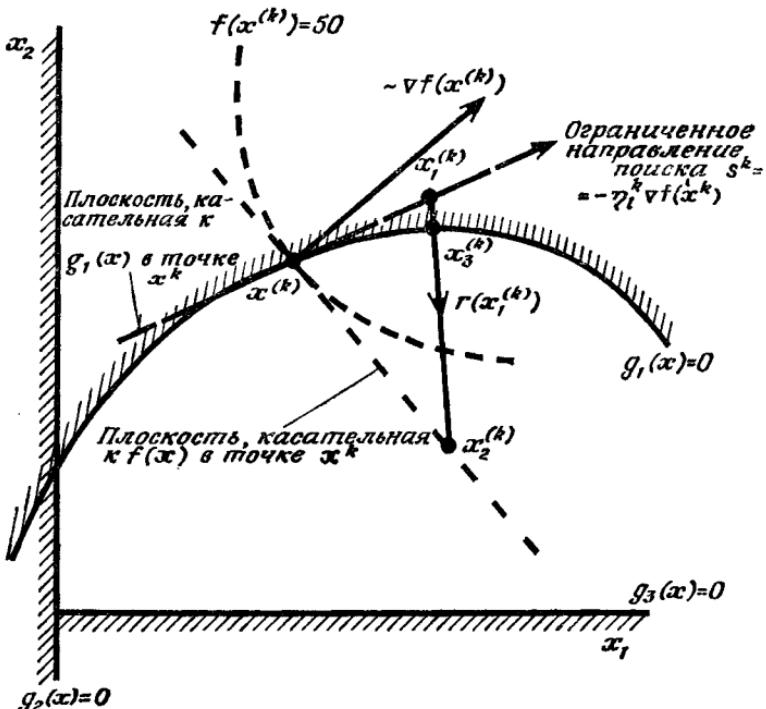
Затем возвращаются к шагу 1.

Как обобщить метод Дэвидона на случай *нелинейных ограничений*? Если ограничения локально линеаризованы, то выбранный метод минимизации должен обеспечить возвращение из точки, не являющейся допустимой, в допустимую область (как, например, это имело место для $x^{(6)}$ на фиг. 6.3.3) и гарантировать, что при этом значение $f(x)$ действительно улучшится. Приведенный ниже алгоритм такого обобщенного метода предложен Дэвисом [24] и основан на результатах, полученных Розеном [14]. На фиг. 6.3.8 графически представлены активное ограничение, заданное функцией $g_1(x)$; два неактивных ограничения, заданных функциями $g_2(x)$ и $g_3(x)$; линии уровней целевой функции $f(x)$ в точке $x^{(k)}$; плоскость, касательная к $f(x^{(k)})$ в точке $x^{(k)}$; плоскость, касательная к $g_1(x^{(k)})$ в точке $x^{(k)}$ (последняя представляет собой линеаризованное ограничение $\tilde{g}_1(x^{(k)}) = 0$). Активным нелинейным ограничением будем считать ограничение, линейная аппроксимация которого является активной в соответствии с соотношением (6.3.15). Вектор «возврата» $r(x^{(k)})$ соединяет недопустимую текущую точку с точкой, принад-

лежащей допустимой области:

$$\mathbf{r}(\mathbf{x}^{(k)}) = \mathbf{B}^T (\mathbf{B}\mathbf{B}^T)^{-1} \phi^{(k)}, \quad (6.3.17)$$

где \mathbf{B} — матрица, строки которой состоят из элементов, представляющих собой найденные в точке $\mathbf{x}^{(k)}$ значения частных производ-



Ф и г. 6.3.8. Метод Дэвидона, обобщенный на случай нелинейных ограничений.

ных активных ограничений в виде неравенств по каждой из переменных, а ϕ — вектор-столбец, элементами которого являются абсолютные значения функций, задающих активные ограничения, вычисленные в точках, лежащих на $s^{(k)}$ в окрестности $x^{(k)}$. Наконец, вектор, задающий направление оптимизационного поиска $s^{(k)}$, есть проекция вектора $-\eta^{(k)} \nabla f(x^{(k)})$ на пересечение всех активных линейных ограничений и активных ограничений, полученных из нелинейных ограничений с помощью линеаризации (последние представляют собой гиперплоскости, касательные к соответствующим нелинейным ограничениям).

Метод, предложенный Дэвисом, имеет следующую структуру ¹⁾:

Шаг 1. По аналогии с шагом 1 алгоритма, используемого в случае линейных ограничений, выявляются активные ограничения в

¹⁾ Ограничения в виде равенств, по-видимому, также могут быть включены в рассмотрение; однако никаких специальных правил Дэвисом по этому поводу указано не было.

виде неравенств путем рассмотрения линеаризованных (в окрестности точки $\mathbf{x}^{(k)}$) ограничений, заменяющих соответствующие нелинейные ограничения.

Шаг 2. Поскольку при использовании аппроксимирующих функций на каждом шаге итерации происходит изменение базиса ограничений, вычисляется матрица $\eta_j^{(k)}$ для j активных линейных ограничений в виде неравенств [вместо того, чтобы в ходе «согласования» активных линеаризованных ограничений применять для понижения и повышения ранга матрицы η требующие больших временных затрат соотношения (6.3.13а) и (6.3.14)]. Матрица $\eta_l^{(k)}$ вводится в память ЭВМ; затем понижается ранг дубликата $\eta_j^{(k)}$ путем последовательного применения соотношения (6.3.13а) для каждого активного линеаризованного ограничения, в результате чего формируется матрица $\eta_l^{(k)}$, где l — суммарное число активных ограничений. Наконец, находится

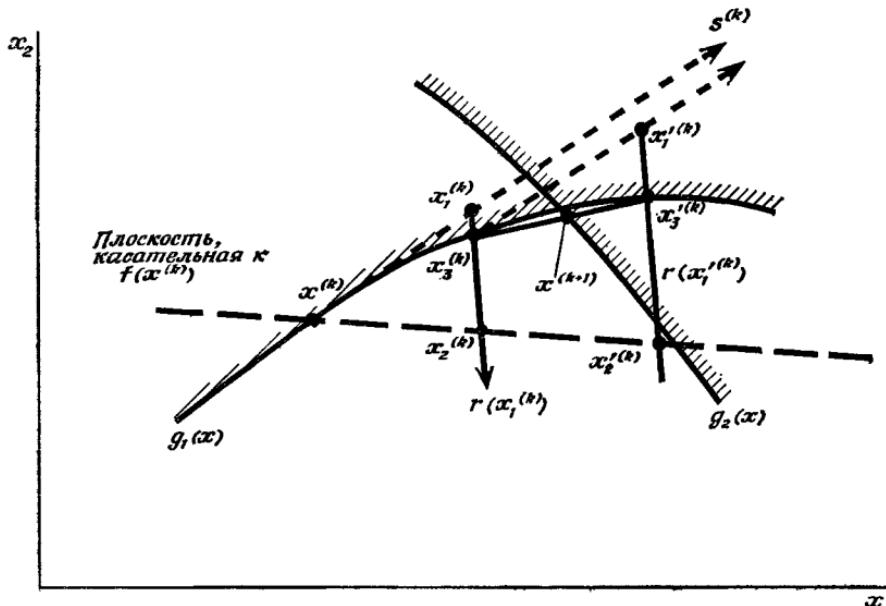
$$\mathbf{s}^{(k)} = -\eta_l^{(k)} \nabla f(\mathbf{x}^{(k)}).$$

Шаг 3. Чтобы начать оптимизационный поиск в точке $\mathbf{x}^{(k+1)}$, осуществляется одномерное перемещение вдоль направления $\mathbf{s}^{(k)}$. Это перемещение реализуется следующим образом. Длина шага $\lambda^{(k)}$ выбирается равной меньшему из чисел α и β , где α — удвоенное перемещение в сторону минимума по алгоритму Ньютона, а β — расстояние до ближайшего линеаризованного ограничения вдоль $\mathbf{s}^{(k)}$. Данная процедура гарантирует, что минимизация превращается в безусловную (т. е. в минимизацию при отсутствии ограничений) и что нерациональный выбор длины шага будет при этом исключен. Далее находится $\mathbf{x}_1^{(k)} = \mathbf{x}^{(k)} + \lambda^{(k)} \mathbf{s}^{(k)}$ (фиг. 6.3.8) и осуществляется проверка с целью выяснения, не нарушается ли какое-либо из нелинейных ограничений. Если какое-нибудь из ограничений, задаваемых функциями $g_i(\mathbf{x}_1^{(k)})$, нарушается, то по направлению вектора $\mathbf{r}(\mathbf{x}^{(k)})$ производится перемещение из $\mathbf{x}_1^{(k)}$ к допустимой области. Чтобы избежать излишних операций, связанных с вычислением значений $g_i(\mathbf{x}^{(k)})$, указанное выше перемещение всегда осуществляется до точки пересечения $\mathbf{r}(\mathbf{x}^{(k)})$ с плоскостью, касательной к контуру $f(\mathbf{x}^{(k)})$, т. е. в точку $\mathbf{x}_2^{(k)}$. (Эта точка является внутренней, так как она находится в области, лежащей между плоскостью, касательной к контуру целевой функции $f(\mathbf{x}^{(k)})$, и ограничивающими поверхностями.) После установления отрезка $\mathbf{x}_1^{(k)} - \mathbf{x}_2^{(k)}$ проводится интерполяция с целью отыскания точки $\mathbf{x}_3^{(k)}$ на поверхности, определяющей ближайшее к $\mathbf{x}_2^{(k)}$ ограничение с точностью до установленного допуска.

Если точка $\mathbf{x}_2^{(k)}$ не удовлетворяет активным ограничивающим условиям, то из точки $\mathbf{x}^{(k)}$ производится новое перемещение при

сокращенной длине шага и, таким образом, находится новая точка $x_2^{(k)}$.

После отыскания граничной точки $x_3^{(k)}$ производится проверка, цель которой заключается в том, чтобы установить, существует ли минимум на дуге, соединяющей $x^{(k)}$ и $x_3^{(k)}$ (см. в этой связи шаг 4).



Фиг. 6.3.9. Экстраполяция при поиске граничной точки в случае, когда нарушаются новые ограничения.

Если существование минимума установлено, производится уточнение его положения путем надлежащей интерполяции. Этот минимум обозначим через $x^{(k+1)}$. Если существование минимума установить не удается, то из точки $x_3^{(k)}$ предпринимается шаг длиной λ в направлении $s^{(k)}$ и повторяется шаг 1, в результате чего определяется точка $x_3'^{(k)}$ (фиг. 6.3.9).

Если в точке $x_3^{(k)}$ оказывается нарушенным какое-либо другое из нелинейных ограничений (на фиг. 6.3.9 этот момент проиллюстрирован на примере ограничения, заданного функцией $g_2(x)$), граничная точка $x^{(k+1)}$ определяется с помощью экстраполяции вдоль линии, соединяющей $x_3^{(k)}$ и непосредственно предшествующую ей точку $x_3'^{(k)}$ (см. шаг 4). После этого осуществляется переход к шагу 5.

Шаг 4. Прекращение оптимизационного поиска в окрестности $x^{(k+1)}$ может иметь место при разных условиях. За исключением случая, когда оказываются нарушенными m новых ограничений ($m \geq 1$), поиск обычно прекращается, если оказывается, что

$f(\mathbf{x}_3^{(k)}) \geq f(\mathbf{x}_3^{(k)})$ или $\nabla^T f(\mathbf{x}_3^{(k)}) s^{(k)} \geq 0$, так как минимум функции $f(\mathbf{x})$ находится на прямой $\mathbf{x}_3^{(k)} - \mathbf{x}_3^{(k)}$. При этом производится интерполяция с целью более точного определения положения минимума функции $f(\mathbf{x})$ на той прямой (точки $\mathbf{x}_4^{(k)}$). После этого из точки

Таблица 6.3.1

Сравнение четырех методов

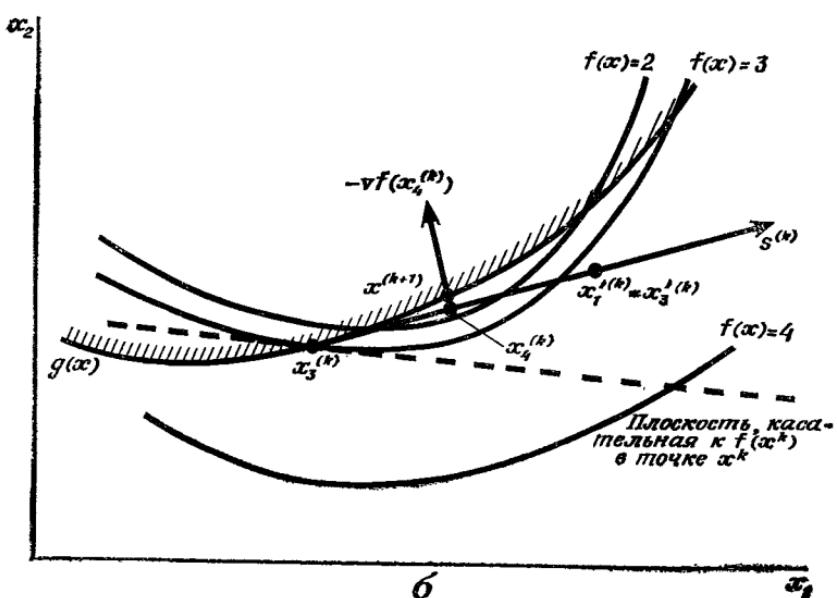
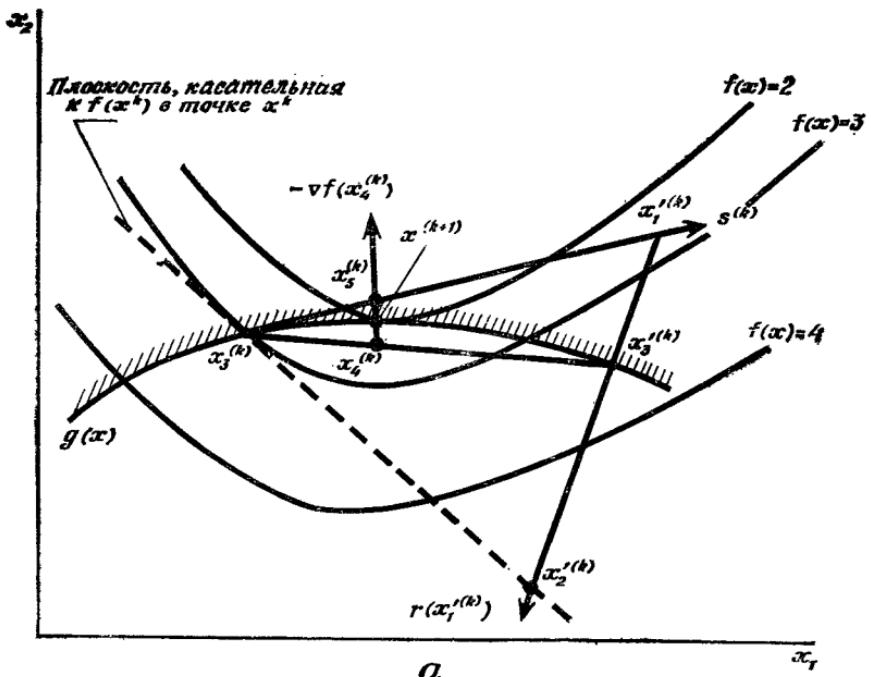
Задача 1)	Метод	Число ограничений, активных в минимуме	Количество вычислений $f(\mathbf{x})$	Число плавильных ограничений
10	а	5	120	0
	б		12	0
	в		8	0
	г		7	
11	а	5	109	114
	б		94	111
	в		10	22
	г		9	
18	а	11	712	805
	б		657	794
	в		227	421
	г		124	

1) В качестве начальной выбирается точка, лежащая в допустимой области.

$\mathbf{x}_4^{(k)}$ предпринимается шаг в направлении «наискорейшего спуска» к ближайшей границе и находится точка $\mathbf{x}^{(k+1)}$. Дэвисом установлено, что для выпуклых функций-ограничений положение $\mathbf{x}^{(k+1)}$ можно определить путем нахождения пересечения $-\nabla f(\mathbf{x}_4^{(k)})$ и $s^{(k)}$ в точке $\mathbf{x}_5^{(k)}$ (не являющейся допустимой) с последующей интерполяцией или экстраполяцией (для вогнутых ограничений) между точками $\mathbf{x}_4^{(k)}$ и $\mathbf{x}_5^{(k)}$. Данная процедура для двумерного случая иллюстрируется на фиг. 6.3.10. Заметим, что при вогнутом ограничении $\mathbf{x}^{(k+1)}$ следует находить путем линейного поиска вдоль $-\nabla f(\mathbf{x}_4^{(k)})$, так как точки $\mathbf{x}_4^{(k)}$ и $\mathbf{x}_5^{(k)}$ совпадают.

Если в процессе поиска минимума $f(\mathbf{x})$ путем перемещения вдоль линии $\mathbf{x}_3^{(k)} - \mathbf{x}_3^{(k)}$ значение $f(\mathbf{x})$ не получается меньшим $f(\mathbf{x}_3^{(k)})$, то полагают $\mathbf{x}^{(k+1)} = \mathbf{x}_3^{(k)}$.

Шаг 5. Как только положение точки $\mathbf{x}^{(k+1)}$ оказывается установленным, по аналогии с шагом 5 оптимизационной процедуры при линейных ограничениях к $\eta_i^{(k)}$ добавляются новые активные линеаризованные ограничения, после чего возвращаются к шагу 1.



Ф и г. 6.3.10. Определение $x^{(k+1)}$.

Шаг 6. Если группа активных ограничений остается без изменений, осуществляется переход от матрицы $\eta^{(k)}$ к новой матрице $\eta^{(k+1)}$ по схеме, представленной шагом 4 оптимизационной процедуры при линейных ограничениях.

Описанная выше процедура воспринимается пока лишь как «красивая абстракция». Поскольку относительно эффективности метода Дэвидона при нелинейных ограничениях известно очень мало, мы предлагаем читателю ознакомиться с табл. 6.3.1, в которой проводится сравнение:

- комбинации метода Дэвидона с МБП¹⁾ (описание метода см. в разд. 7.1.3);
- комбинации метода Голдфарба с МБП;
- метода Дэвидона — Дэвиса (описание метода дано выше);
- метода Муртага и Сарджента (аналогичного методу Дэвиса).

При этом анализировались результаты, полученные при рассмотрении задач 10, 11 и 18, приведенных в приложении А, а также и ряда более простых задач.

Применительно к задачам, содержащим только линейные ограничения (в таблице ссылки на такого рода задачи отсутствуют), сочетание метода Голдфарба с МБП оказывается гораздо эффективнее сочетания метода Дэвидона и МБП; однако при решении задач с нелинейными ограничениями эти комбинированные методы почти равноценны. С точки зрения эффективности оба они значительно уступают методу Дэвидона — Дэвиса.

6.4. МЕТОД ДОПУСТИМЫХ НАПРАВЛЕНИЙ (МЕТОД ЗАУТЕНДАЙКА)

Как уже отмечалось выше, для каждой допустимой точки $x^{(k)}$ в R может существовать множество различных допустимых направлений оптимизационного поиска [27]. Сущность методов допустимых направлений сводится к следующему: поиск начинается в допустимой точке пространства решений и реализуется (при линеаризованных ограничениях) по траектории, обеспечивающей улучшение значений целевой функции и вместе с тем никогда не выходящей за пределы допустимой области. В этом смысле предложенный Розеном метод проекции градиента представляет собой одну из модификаций метода допустимых направлений. Однако необходимо обратить внимание на следующее: проекция градиента $\nabla f(x^{(k)})$ на подмножество ограничений, содержащее $x^{(k)}$, задает направление поиска, соответствующее (относительно евклидовой метрики) направлению наискорейшего спуска для целевой функции, тогда

¹⁾ МБП — метод барьерных поверхностей.

как в методе Заутендайка используется другая метрика, а именно метрика, определяемая соотношением

$$\|\Delta \mathbf{x}\| = \max \{|\Delta x_1|, \dots, |\Delta x_n|\}.$$

Данная метрика приводит к тому, что в качестве допустимого выбирается такое направление, которое обеспечивает наибольшую минимизирующую поправку к значению целевой функции без нарушения какого-либо из ограничений.

Метод Заутендайка позволяет оперировать как с линейными, так и с нелинейными ограничениями; однако с его помощью не удается решать задачи с ограничениями в виде равенств. Задача, рассмотрению которой посвящен данный раздел, по существу представляет собой задачу (6.1.1) без ограничений в виде равенств. Линеаризация фигурирующих в этой задаче функций путем их разложения в ряд Тейлора в окрестности точки $\mathbf{x}^{(k)}$ приводит к следующей задаче:

$$\text{минимизировать } f(\mathbf{x}^{(k)}) + \nabla^T f(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}), \quad \mathbf{x} \in E^n,$$

при ограничениях

$$g_i(\mathbf{x}^{(k)}) + \nabla g_i(\mathbf{x}^{(k)}) (\mathbf{x} - \mathbf{x}^{(k)}) \geq 0, \quad i = 1, \dots, p. \quad (6.4.1)$$

Процедура Заутендайка устанавливает наиболее благоприятные¹⁾ из возможных направлений оптимизационного поиска при отправной точке $\mathbf{x}^{(k)}$ путем решения некоторой (связанной с исходной задачей) подзадачи, а именно путем решения задачи (6.4.5), структура которой указана ниже.

Перемещение из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$, как обычно, задается соотношением $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \mathbf{s}^{(k)}$. Подстановка в задаче (6.4.1) вместо вектора \mathbf{x} его текущего значения $\mathbf{x}^{(k+1)}$ приводит к задаче минимизации $f(\mathbf{x}^{(k)}) + \lambda^{(k)} \nabla^T f(\mathbf{x}^{(k)}) \mathbf{s}^{(k)}$

при ограничениях

$$g_i(\mathbf{x}^{(k)}) + \lambda^{(k)} \nabla g_i(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} \geq 0, \quad i = 1, \dots, p. \quad (6.4.2)$$

Поскольку $f(\mathbf{x}^{(k)})$ и $g_i(\mathbf{x}^{(k)}) \geq 0$ являются константами, необходимые и достаточные условия реализации перемещения из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$ в связи с решением задачи (6.4.2) [причем такого перемещения, когда точка $\mathbf{x}^{(k+1)}$ остается допустимой с точки зрения

¹⁾ В смысле получения наибольшей минимизирующей поправки к значению $f(\mathbf{x})$ при перемещении из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$ без нарушения какого-либо из ограничений.

условий задачи (6.4.1)] имеют следующий вид:

$$\nabla^T f(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} < 0 \quad (6.4.3)$$

и

$$\nabla^T g_i(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} \geq 0, \quad i = 1, \dots, p. \quad (6.4.4)$$

Любой вектор $\mathbf{s}^{(k)}$, удовлетворяющий неравенствам (6.4.3) и (6.4.4), одновременно является вектором допустимого направления.

Метод Заутендайка выделяет допустимое направление, обеспечивающее получение наибольшей минимизирующей поправки к значению целевой функции $f(\mathbf{x})$ на шаге от точки $\mathbf{x}^{(k)}$ к точке $\mathbf{x}^{(k+1)}$ путем решения следующей задачи линейного программирования:

минимизировать $\nabla^T f(\mathbf{x}^{(k)}) \mathbf{s}^{(k)}$, $\mathbf{s}^{(k)} \in E^n$,

при ограничениях

$$\nabla^T g_i(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} \geq 0, \quad i = 1, \dots, p. \quad (6.4.5)$$

Решая задачу (6.4.5), мы определяем составляющие вектора допустимого направления, вдоль которого следует перемещаться из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$.

Каждый из этапов реализации метода Заутендайка имеет следующую алгоритмическую структуру:

1. Пусть $\mathbf{x}^{(k)}$ — некоторая допустимая точка задачи (6.0.1), в которой отсутствуют, однако, ограничения в виде равенств.

2. Вычисляются градиенты функции $f(\mathbf{x})$ и функций $g_i(\mathbf{x})$ ($i = 1, \dots, p$) в точке $\mathbf{x}^{(k)}$ и решается задача линейного программирования (6.4.5), в результате чего находится допустимое направление $\mathbf{s}^{(k)}$. Решение задачи (6.4.5) находится с помощью любого из алгоритмов линейного программирования.

3. Если $\nabla^T f(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} < 0$, то определяется максимальная длина шага λ^* , осуществляемого в направлении $\mathbf{s}^{(k)}$ без выхода за пределы допустимой области задачи (6.0.1); λ^* получается из условия $\lambda^* = \max \{ \lambda | \mathbf{x}^{(k)} + \lambda \mathbf{s}^{(k)} \in R \}$. Значение λ^* может быть найдено с помощью одномерного поиска в направлении $\mathbf{s}^{(k)}$, начиная из точки $\mathbf{x}^{(k)}$. Затем определяется такое значение $\lambda^{(k)}$, удовлетворяющее условию $0 \leq \lambda^{(k)} \leq \lambda^*$, для которого $f(\mathbf{x}^{(k)} + \lambda^{(k)} \mathbf{s}^{(k)})$ есть минимальное значение целевой функции в направлении $\mathbf{s}^{(k)}$. Новая точка определяется из $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \mathbf{s}^{(k)}$. Теперь возвращаются к шагу 2 с тем, чтобы начать оптимизационный поиск на $(k+1)$ -м этапе вычислительного процесса.

4. Если $\nabla^T f(\mathbf{x}^{(k)}) \mathbf{s}^{(k)} = 0$, оптимизационный поиск заканчивается, так как дальнейшее уменьшение значения $f(\mathbf{x})$ не представляется возможным.

Заутендайком дано также описание модифицированного метода допустимых направлений [28], в рассмотрении которого пока нет

особой необходимости. Опубликованные сведения относительно машинных программ, которые позволяли бы реализовать метод Заутендейка на ЭВМ, крайне бедны; тем не менее некоторые результаты известны [28]. Рассмотренный выше метод Заутендейка по сравнению с другими методами, применяемыми при аналогичной постановке задачи нелинейного программирования, является более «быстroredействующим». Кроме того, он имеет то преимущество, что позволяет оперировать как с линейными, так и с нелинейными ограничениями, имеющими вид неравенств. Однако нет достаточно данных, доказывающих эффективность данного метода при решении задач с большим числом нелинейных ограничений.

6.5. МЕТОД ОБОБЩЕННОГО ПРИВЕДЕННОГО ГРАДИЕНТА (МОПГ)

Алгоритм обобщенного приведенного градиента [30—32] представляет собой модификацию алгоритма Вольфа [33], которая может быть использована для решения задач при нелинейном характере и целевой функции и ограничений. Структура данного метода предполагает реализацию (по отношению к нелинейным функциям) линейно-аппроксимирующих процедур; определение новых переменных, ортогональных к некоторым из ограничений; приведение градиента целевой функции к преобразованному таким способом базису. (Вольфом подробно рассматривается связь метода приведенного градиента в его первоначальной формулировке и симплексного метода линейного программирования.) Хотя задача, решаемая методом приведенного градиента, формируется в общем виде как задача

минимизации $f(\mathbf{x})$, $\mathbf{x} \in E^n$,

при ограничениях

$$h_i(\mathbf{x}) = 0, \quad i = 1, \dots, m, \quad (6.5.1)$$

$$L_j \leq x_j \leq U_j, \quad j = 1, \dots, n,$$

ограничения в виде неравенств удается включить в оптимизационную схему путем вычитания из левых частей ограничений, имеющих вид неравенств, неотрицательных ослабляющих переменных, что превращает ограничения-неравенства в ограничения-равенства вида

$$h_i(\mathbf{x}) = g_i(\mathbf{x}) - v_i^2 = 0$$

при неограниченных предельных значениях переменных v_i , т. е. $-\infty \leq v_i \leq \infty$. (Переменные v_i добавляются к n переменным исходной задачи.)

Если в соответствии с принятым предположением выполняется условие невырожденности, в структуре обобщенного алгоритма

приведенного градиента различают две системы переменных: m базисных (зависимых) переменных x_i формируют подмножество I , а $(n - m)$ небазисных (независимых) переменных x_K образуют подмножество K . При этом зависимые переменные неявным образом определяются через независимые переменные; следовательно, $f(x)$ есть функция лишь $(n - m)$ независимых переменных. Поясним используемые в данном разделе специальные обозначения:

$$\mathbf{h} \equiv \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_m(\mathbf{x}) \end{bmatrix} - (m \times 1)\text{-мерная матрица},$$

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}_I} \equiv \begin{bmatrix} \frac{\partial h_1(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_1(\mathbf{x})}{\partial x_m} \\ \vdots & \ddots & \vdots & \ddots \\ \frac{\partial h_m(\mathbf{x})}{\partial x_1} & \cdots & \frac{\partial h_m(\mathbf{x})}{\partial x_m} \end{bmatrix} - (m \times m)\text{-мерная квадратная матрица («базисная» матрица)},$$

$$\nabla_{\mathbf{x}_I}^T f = \left[\frac{\partial f(\mathbf{x})}{\partial x_{m+1}} \cdots \frac{\partial f(\mathbf{x})}{\partial x_n} \right] - [1 \times (n - m)]\text{-мерная матрица},$$

$$\nabla_{\mathbf{x}_K}^T f = \left[\frac{\partial f(\mathbf{x})}{\partial x_1} \cdots \frac{\partial f(\mathbf{x})}{\partial x_m} \right] - (1 \times m)\text{-мерная матрица},$$

$$\frac{df(\mathbf{x})}{d\mathbf{x}_K} = \left[\frac{df(\mathbf{x})}{dx_{m+1}} \cdots \frac{df(\mathbf{x})}{dx_n} \right] = z - [1 \times (n - m)]\text{-мерная матрица}$$

(матрица приведенного градиента),

$$\frac{d\mathbf{x}_I}{d\mathbf{x}_K} = \begin{bmatrix} \frac{dx_1}{dx_{m+1}} & \cdots & \frac{dx_1}{dx_n} \\ \vdots & \ddots & \vdots & \ddots \\ \frac{dx_m}{dx_{m+1}} & \cdots & \frac{dx_m}{dx_n} \end{bmatrix} - [m \times (n - m)]\text{-мерная матрица},$$

$$\frac{d\mathbf{h}}{d\mathbf{x}_K} = \begin{bmatrix} \frac{dh_1(\mathbf{x})}{dx_{m+1}} & \cdots & \frac{dh_1(\mathbf{x})}{dx_n} \\ \vdots & \ddots & \vdots & \ddots \\ \frac{dh_m(\mathbf{x})}{dx_{m+1}} & \cdots & \frac{dh_m(\mathbf{x})}{dx_n} \end{bmatrix} - [m \times (n - m)]\text{-мерная матрица},$$

6.5.1. ПРИВЕДЕНИЙ ГРАДИЕНТ

Поскольку ограничения в виде равенств отражают зависимость между переменными задачи (6.5.1) лишь в неявной форме, непосредственное сокращение размерности этой задачи, вообще говоря, неосуществимо. Другими словами, уравнения, задающие упомянутые выше ограничения, не могут быть разрешены относительно зависимых переменных с тем, чтобы их можно было путем соответ-

ствующих подстановок исключить из структуры целевой функции и, таким образом, выразить целевую функцию только через независимые переменные. Однако метод ограниченных вариаций позволяет сократить размерность задачи и делает возможным использование приведенного градиента в качестве одного из критериев при установлении оптимальности. Чтобы проиллюстрировать эту идею, рассмотрим частный случай задачи (6.5.1), предположив, что целевая функция зависит всего от двух переменных при единственном ограничении в виде равенства, а именно рассмотрим следующую задачу:

минимизировать $f(x_1, x_2)$

при ограничении

$$h(x_1, x_2) = 0.$$

При бесконечно малых приращениях x_1 и x_2 имеем

$$df(x) = \frac{\partial f(x)}{\partial x_1} dx_1 + \frac{\partial f(x)}{\partial x_2} dx_2.$$

Кроме того,

$$dh(x) = \frac{\partial h(x)}{\partial x_1} dx_1 + \frac{\partial h(x)}{\partial x_2} dx_2 = 0.$$

Приведенные выше выражения линейны относительно dx_1 и dx_2 , так что из выражения для $df(x)$ либо dx_1 , либо dx_2 можно исключить. На фиг. 6.5.1 показано, что единственными допустимыми перемещениями являются перемещения вдоль ограничения $h(x_1, x_2) = 0$.

Решим уравнение $dh(x) = 0$ относительно dx_2 :

$$dx_2 = -\frac{\partial h(x)/\partial x_1}{\partial h(x)/\partial x_2} dx_1.$$

Полученное решение подставим в выражение для $df(x)$:

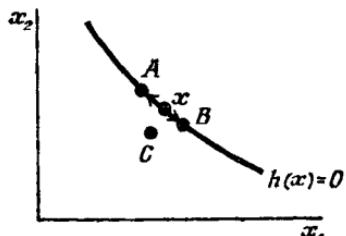
$$df(x) = \left(\frac{\partial f(x)}{\partial x_1} - \frac{\partial f(x)}{\partial x_2} \frac{\partial h(x)/\partial x_1}{\partial h(x)/\partial x_2} \right) dx_1.$$

Отсюда получим следующее выражение для *приведенного градиента*:

$$\frac{df(x)}{dx_1} = \frac{\partial f(x)}{\partial x_1} - \frac{\partial f(x)}{\partial x_2} \left[\frac{\partial h(x)}{\partial x_2} \right]^{-1} \frac{\partial h(x)}{\partial x_1}.$$

Необходимым условием минимума $f(x)$ является равенство нулю дифференциала $df(x)$ (т. е. $df(x) = 0$). По аналогии со случаем минимизации без ограничений можно записать это условие в виде

$$\frac{df(x)}{dx_1} = 0.$$



Фиг. 6.5.1. Допустимые перемещения в случае, когда минимизация $f(x)$ осуществляется при единственном ограничении в виде равенства $h(x) = 0$; A и B — допустимые точки, тогда как точка C не является допустимой.

Возвращаясь к общей формулировке задачи (6.5.1), выразим обобщенный приведенный градиент через компоненты градиента целевой функции, матрицу, обратную по отношению к базисной, и якобиан для ограничений, имеющих вид равенств. Для $d\hat{f}(\mathbf{x})$ получаем

$$d\hat{f}(\mathbf{x}) = \nabla_{\mathbf{x}_K}^T \hat{f} d\mathbf{x}_K + \nabla_{\mathbf{x}_I}^T \hat{f} d\mathbf{x}_I.$$

Путем перемножения матричных элементов нетрудно показать, что приведенный градиент можно записать в виде

$$\frac{d\hat{f}(\mathbf{x})}{d\mathbf{x}_K} = \nabla_{\mathbf{x}_K}^T \hat{f} + \nabla_{\mathbf{x}_I}^T \hat{f} \frac{d\mathbf{x}_I}{d\mathbf{x}_K}. \quad (6.5.2)$$

(Следует помнить, что такие производные, как $d\mathbf{x}_{m+1}/d\mathbf{x}_{m+2}$ или $d\mathbf{x}_{m+1}/d\mathbf{x}_n$, обращаются в нуль, так как переменные x_i ($i = m+1, \dots, n$) являются независимыми.) Чтобы исключить из соотношения (6.5.2) неудобную для дальнейшего анализа матрицу $d\mathbf{x}_I/d\mathbf{x}_K$, заметим, что

$$dh_i(\mathbf{x}) = \nabla_{\mathbf{x}_K}^T h_i(\mathbf{x}) d\mathbf{x}_K + \nabla_{\mathbf{x}_I}^T h_i(\mathbf{x}) d\mathbf{x}_I = 0, \quad i = 1, \dots, m,$$

$$\frac{dh}{d\mathbf{x}_K} = \frac{\partial h}{\partial \mathbf{x}_K} + \left(\frac{\partial h}{\partial \mathbf{x}_I} \right) \left(\frac{d\mathbf{x}_I}{d\mathbf{x}_K} \right) = 0, \quad (6.5.3a)$$

так что

$$\frac{d\mathbf{x}_I}{d\mathbf{x}_K} = - \left(\frac{\partial h}{\partial \mathbf{x}_I} \right)^{-1} \left(\frac{\partial h}{\partial \mathbf{x}_K} \right). \quad (6.5.36)$$

Подставляя (6.5.36) в (6.5.2), получаем для обобщенного приведенного градиента

$$\frac{d\hat{f}(\mathbf{x})}{d\mathbf{x}_K} = \nabla_{\mathbf{x}_K}^T \hat{f} - \nabla_{\mathbf{x}_I}^T \hat{f} \left(\frac{\partial h}{\partial \mathbf{x}_I} \right)^{-1} \left(\frac{\partial h}{\partial \mathbf{x}_K} \right). \quad (6.5.4)$$

Заметим, что число составляющих вектора приведенного градиента равняется числу независимых переменных (по одной составляющей на каждую независимую переменную).

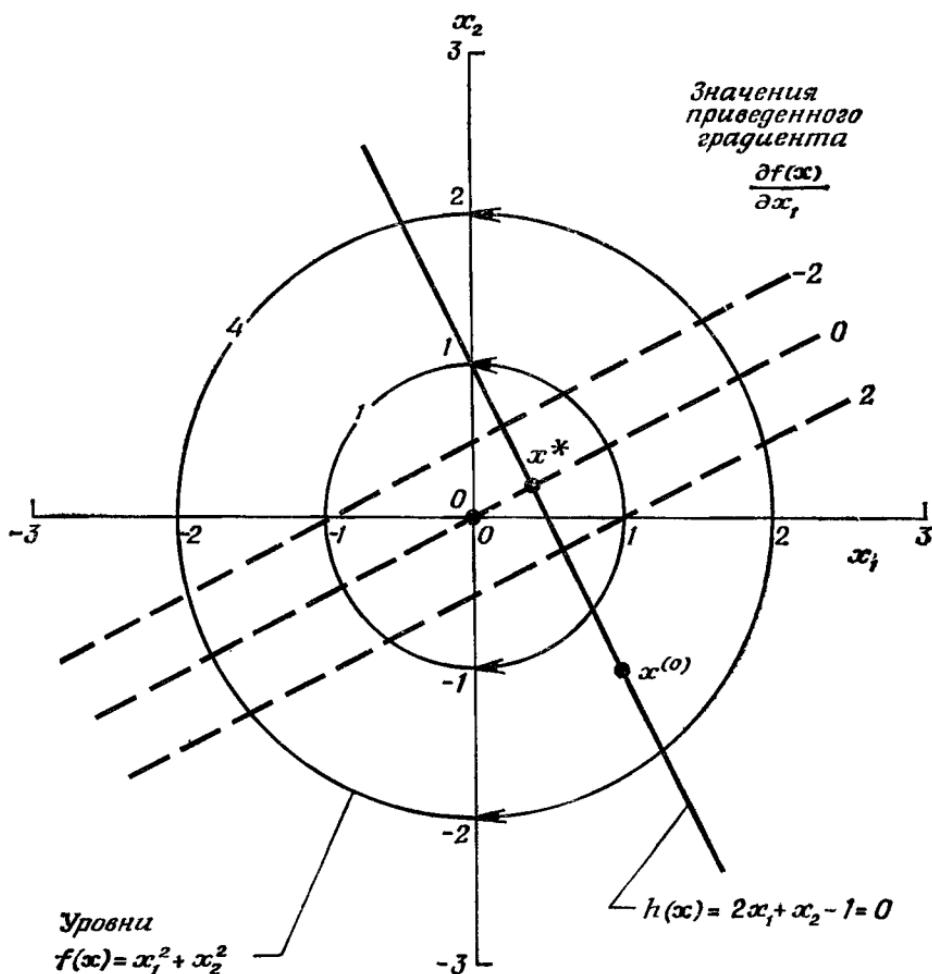
На фиг. 6.5.2 дано графическое представление приведенного градиента для следующей задачи квадратичного программирования, включающей одно ограничение в виде равенства:

минимизировать $f(\mathbf{x}) = x_1^2 + x_2^2$

при ограничении

$$h(\mathbf{x}) = 2x_1 + x_2 - 1 = 0.$$

Пусть x_1 есть независимая (небазисная) переменная, а x_2 — зависимая (базисная) переменная. Частные производные функций



Фиг. 6.5.2. Приведенный градиент: условный минимум находится на линии $h(\mathbf{x}) = 0$ в точке $\mathbf{x}^* = [0,4 \ 0,2]^T$ (в этой точке приведенный градиент обращается в нуль).

$f(\mathbf{x})$ и $h(\mathbf{x})$ имеют следующий вид:

$$\frac{\partial f(\mathbf{x})}{\partial x_1} = 2x_1, \quad \frac{\partial f(\mathbf{x})}{\partial x_2} = 2x_2,$$

$$\frac{\partial h(\mathbf{x})}{\partial x_1} = 2, \quad \frac{\partial h(\mathbf{x})}{\partial x_2} = 1.$$

Для обобщенного приведенного градиента получаем

$$\begin{aligned} \frac{df(\mathbf{x})}{dx_1} &= \frac{\partial f(\mathbf{x})}{\partial x_1} - \frac{\partial f(\mathbf{x})}{\partial x_2} \left[\frac{\partial h(\mathbf{x})}{\partial x_2} \right]^{-1} \frac{\partial h(\mathbf{x})}{\partial x_1} = \\ &= 2x_1 - 2x_2 \cdot 1 \cdot 2 = 2x_1 - 4x_2. \end{aligned}$$

Перемещение из любой допустимой точки вдоль ограничения $h(x) = 0$ до тех пор, пока $df(x)/dx_1$ не обратится в нуль, обеспечивает минимизацию $f(x)$.

Можно интерпретировать приведенный градиент и другим образом, опираясь на понятие двойственной по отношению к (6.5.1) задачи математического программирования¹⁾. Нетрудно показать, что условия Куна — Таккера (см. подразд. 2.5.4) для двойственной по отношению к (6.5.1) задачи (если она является задачей выпуклого программирования) имеют следующий вид:

$$\nabla_{x_K} f(x) - v \frac{dh(x)}{dx_K} = z, \quad (6.5.5a)$$

$$\nabla_{x_I} f(x) - v \frac{dh(x)}{dx_I} = 0, \quad (6.5.5b)$$

$$z_j \leq 0, \text{ если } x_j = L_j,$$

$$z_j \geq 0, \text{ если } x_j = U_j, j = m + 1, \dots, n, \quad (6.5.5b)$$

$$z_j = 0, \text{ если } L_j \leq x_j \leq U_j,$$

где $z = [z_{m+1} \dots z_n]$, а $v = [v_1 \dots v_m]$.

Если положить $z \equiv (df(x)/dx_K)$, то подстановка (6.5.5b) в (6.5.5a) приводит к (6.5.4). Таким образом, значения составляющих приведенного градиента служат ориентиром при отыскании оптимального вектора x (поиск состоит в отыскании точки, в которой приведенный градиент целевой функции обращается в нуль).

6.5.2. НАПРАВЛЕНИЕ ОПТИМИЗАЦИОННОГО ПОИСКА

Алгоритм обобщенного приведенного градиента начинает работу с допустимой точки. Если же относительно условий рассматриваемой задачи вектор x не является допустимым и, следовательно, возникает необходимость вводить в рассмотрение (при записи ограничений) искусственные переменные, то значения последних постепенно сводят к нулю путем добавления к целевой функции штрафного члена (см. гл. 7), что в итоге делает вектор x допустимым. С этой целью можно также использовать и сам алгоритм обобщенного приведенного градиента, минимизируя (или максимизируя) значение каждой искусственной переменной (или суммы абсолютных значений этих переменных). Если приведенный градиент ни на одном из этапов вычислительной процедуры не обращается в нуль, произ-

¹⁾ В математическом программировании понятие двойственности отражает то обстоятельство, что если существует допустимый оптимальный вектор x , являющийся решением задачи минимизации (которую называют исходной), то этот же вектор является решением соответствующей задачи максимизации (которую называют двойственной).

водится замена текущего вектора по стандартной формуле

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \lambda^{(k)} \Delta^{(k)}, \quad (6.5.6)$$

где $\lambda \geq 0$, а n элементов вектор-столбца $\Delta^{(k)}$ задают направление оптимизационного поиска по алгоритму обобщенного приведенного градиента. При этом компоненты Δ_j , соответствующие независимым (небазисным) переменным, определяются способом, который отличается от способа определения Δ_j для зависимых (базисных) переменных. Начнем с обсуждения способа определения составляющих Δ_j , соответствующих независимым переменным.

Индекс K , обозначающий подмножество независимых переменных, ниже всюду (за исключением случаев, когда окажется необходимым различать \mathbf{x}_I и \mathbf{x}_K) будет опускаться. Направления оптимизационного поиска Δ_j ($j = m+1, \dots, n$), соответствующие независимым переменным, определяются через значения составляющих приведенного градиента следующим образом:

$$\Delta_j^{(k)} = 0 \begin{cases} \text{если } x_j^{(k)} = U_j \text{ и } z_j^{(k)} > 0, \\ \text{если } x_j^{(k)} = L_j \text{ и } z_j^{(k)} < 0, \end{cases}$$

$$\Delta_j^{(k)} = -z_j^{(k)}, \text{ если } L_j < x_j^{(k)} < U_j,$$

где z_j , согласно приведенному выше определению [см. соотношения (6.5.5в)], представляют собой составляющие приведенного градиента (т. е. $\mathbf{z} = df(\mathbf{x})/d\mathbf{x}_K$). Так, например, на графике, приведенном на фиг. 6.5.2, в точке $\mathbf{x}^{(0)} = [1 \ -1]^T$ имеем $\Delta_1^{(0)} = -z_1^{(0)} = -(df \times (\mathbf{x})/dx_1) = -6$; следовательно, $x_1^{(1)} = 1 - 6\lambda$, а в точке $\mathbf{x}^{(0)} = \left[-\frac{1}{2} \ 2 \right]^T$ имеем $-z_1^{(0)} = 9$ и $x_1^{(1)} = 1 + 9\lambda$.

Если ограничения линейны, то в силу (6.5.6) метод обобщенного приведенного градиента совпадает с методом приведенного градиента, предложенным Вольфом. В соответствии с данным алгоритмом составляющие $\Delta_j^{(k)}$ при этом также обращаются в нуль, когда значение x_j становится очень близким либо к нижнему предельному значению L_j , либо к верхнему предельному значению U_j ; при этом прилагательное «близкий» понимается в том смысле, что разница между x_j и L_j (или U_j) не превосходит некоторое произвольное малое число ε .

Ниже рассматриваются два других варианта алгоритма обобщенного приведенного градиента, с помощью которых можно выбирать $\Delta_j^{(k)}$ для независимых переменных.

1. Симплекс-модификация метода обобщенного приведенного градиента. Пусть

$$|z_j^{(k)}| = \max |z_j^{(k)}|$$

по значениям j , для которых $L_j < x_j^{(k)} < U_j$. Положим

$$\Delta_j^{(k)} = \begin{cases} 0, & \text{если } j \neq l, \\ -z_i^{(k)}, & \text{если } j = l. \end{cases}$$

Если целевая функция и ограничения линейны, симплекс-модификация метода обобщенного приведенного градиента эквивалентна симплексному методу линейного программирования. Симплекс-модификация метода обобщенного приведенного градиента обладает тем достоинством, что с ее помощью удается сократить количество вычислительных операций, необходимых для определения направлений оптимизационного поиска $\Delta^{(k)}$, причем последние оказываются независимыми от того, в каких единицах измеряются составляющие вектора \mathbf{x} .

2. Циклический вариант метода обобщенного приведенного градиента. В рамках данного варианта составляющие вектора $\Delta^{(k)}$ определяются по следующей схеме:

Номер итерации	$\Delta_j^{(k)}$
1	$\Delta_j^{(1)} = \begin{cases} -z_1^{(1)}, & j = 1 \\ 0, & j \neq 1 \end{cases}$
2	$\Delta_j^{(2)} = \begin{cases} -z_2^{(2)}, & j = 2 \\ 0, & j \neq 2 \end{cases}$
и т. д.	

После выполнения n такого рода операций вычислительный цикл повторяется. Рассматриваемая модификация метода обобщенного приведенного градиента также «индифферентна» по отношению к выбору единиц измерения составляющих вектора \mathbf{x} .

Наконец, направления оптимизационного поиска для независимых переменных можно (при желании) вводить в блок-схему алгоритма обобщенного приведенного градиента и другими способами, отличающимися от только что рассмотренных, например с помощью метода сопряженных градиентов (см. подразд. 3.3.2). Вычислительные эксперименты показывают, что, прежде чем говорить о преимуществах методов определения оптимизационных направлений, не совпадающих с симплекс-модификацией и циклическим вариантом метода обобщенного приведенного градиента, необходимо вначале накопить определенный опыт логического и экспериментального характера.

Перейдем теперь к рассмотрению вопроса о выборе направлений оптимизационного поиска для зависимых переменных. При этом направляющий вектор $\Delta_{\mathbf{x}_l}$ для определения составляющих вектора

\mathbf{x} , входящих в подмножество \mathbf{x}_I , выбирается иначе, чем для независимых составляющих этого вектора, входящих в подмножество \mathbf{x}_K . Мы знаем, что если ограничения линейны, то их можно разрешить относительно базисных (зависимых) переменных, выразив последние в явном виде через независимые (небазисные) переменные. В случае же нелинейных ограничений сделать это не всегда удается. Поэтому при наличии нелинейных ограничений вектор направления оптимизационного поиска для подмножества \mathbf{x}_I определяется по существу с помощью линеаризации ограничений по схеме, заданной соотношением (6.5.3а) или (6.5.3б):

$$d\mathbf{x}_I = - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}_I} \right)^{-1} \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}_K} \right) d\mathbf{x}_K,$$

или в разностном виде

$$\Delta_{\mathbf{x}_I}^{(k)} = - \left(\frac{\partial \mathbf{h}(\mathbf{x}^{(k)})}{\partial \mathbf{x}_I} \right)^{-1} \left(\frac{\partial \mathbf{h}(\mathbf{x}^{(k)})}{\partial \mathbf{x}_K} \right) d\mathbf{x}_K^{(k)}. \quad (6.5.7)$$

Так, например, обращаясь к иллюстрации, приведенной на фиг. 6.5.2, мы видим, что по отношению к точке $\mathbf{x}^{(0)}$ перемещение $\Delta_1^{(0)} = -6$, и, следовательно, $\Delta_2^{(0)} = -(1)(2)(-6) = 12$.

6.5.3. МЕТОД ОПТИМИЗАЦИОННОГО ПОИСКА В ЗАДАННОМ НАПРАВЛЕНИИ С ЦЕЛЬЮ НАХОЖДЕНИЯ ДОПУСТИМОЙ ТОЧКИ

Опишем сначала процедуру завершения поиска длины шага, уменьшающего $f(\mathbf{x})$, с использованием как зависимых, так и независимых переменных, а затем поясним, каким образом удается подобрать значения зависимых переменных так, чтобы текущее значение $f(\mathbf{x})$ получило дополнительную минимизирующую поправку при сохранении вектора \mathbf{x} внутри или на границе допустимой области. Прежде всего определим, какими должны быть значения $\lambda^{(k)}$ в соотношении (6.5.6). Как обычно, значение целевой функции в точке $(\mathbf{x}^{(k)} + \lambda \Delta^{(k)})$ минимизируется при помощи параметра λ на k -м шаге (т. е. λ^k), значение которого определяется в процессе одномерного дихотомического поиска. Значения $\lambda^{(k)}$ лежат в интервале, определяемом неравенствами $0 \leq \lambda^{(k)} \leq \lambda_m$, где

$$\lambda_m = \min \{ \lambda_1, \lambda_2 \}, \quad (6.5.8)$$

и для $j = 1, \dots, n$ имеют место соотношения

$$\lambda_1 = \min \left\{ \min \left\{ \frac{x_j^{(k)} - L_j}{-\Delta_j^{(k)}} \mid \Delta_j < 0 \right\}, \min \left\{ \frac{U_j - x_j^{(k)}}{\Delta_j^{(k)}} \mid \Delta_j > 0 \right\} \right\},$$

$$\lambda_2 = \left\{ \max \frac{\lambda_3^{(i)}}{\nabla^T f(\mathbf{x}^{(i)}) \Delta^{(i)} / \| \Delta^{(i)} \|^2} \right\} \left\{ \frac{\nabla^T f(\mathbf{x}^{(k)}) \Delta^{(k)}}{\| \Delta^{(k)} \|^2} \right\},$$

где $\lambda_3^{(i)}$ — наибольшее из возможных значений λ , для которого переход в допустимую точку реализуется при улучшении значения целевой функции $f(\mathbf{x})$ без изменения подмножества базисных переменных x_i , которые использовались на p предыдущих итерациях. Индекс k означает порядковый номер текущего шага, а индекс i — порядковый номер итерации, связанной с λ_3 . Величина λ_2 выбирается с учетом данных, полученных на k -м шаге. После выбора λ путем минимизации $f(\lambda)$ может обнаружиться, что разность

$$f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})$$

окажется меньше некоторого заранее заданного критического значения. В этом случае значение λ можно постепенно увеличивать, скажем вдвое, до тех пор, пока не будет выполняться условие

$$\left| \frac{f(\mathbf{x}^{(k+1)}) - f(\mathbf{x}^{(k)})}{f(\mathbf{x}^{(k)})} \right| > \varepsilon.$$

Таким путем от неудовлетворительного (слишком малого) выбора λ_m осуществляется переход к более оптимальным значениям этого параметра; однако при этом возникают дополнительные трудности. Абади и Гигу проанализировали ряд других факторов, влияющих на выбор λ . Оказалось, что в любом случае в каждом из направлений $\Delta_j^{(k)}$ реализуется перемещение с шагом длиной $\lambda^{(k)} \Delta_j^{(k)}$, за исключением тех ситуаций, когда для той или иной независимой переменной граничное условие, задаваемое предельным значением L_j (или U_j), не выполняется; в таких случаях граничным значением соответствующей переменной становится ее значение на $(k + 1)$ -м шаге, т. е. $x_j^{(k+1)}$.

Если λ определяется диахотомическим поиском, то, подставляя найденное значение этого параметра в (6.5.6) и учитывая (6.5.7), мы обнаруживаем, что некоторые из составляющих $\mathbf{x}_j^{(k)}$ оказываются недопустимыми (обозначим их через $\tilde{\mathbf{x}}_j^{(k)}$), причем такая ситуация при нелинейных ограничениях вполне объяснима. В таких случаях для нахождения допустимого вектора $\mathbf{x}_j^{(k+1)}$ преобразуются лишь базисные (зависимые) переменные. Предположим, что в точке $(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_j^{(k+1)})$ выполняется условие $\mathbf{h}(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_j^{(k+1)}) \neq 0$. Если ограничения линеаризуются путем замены нелинейных функций усеченным рядом Тейлора, можно определить такой вектор $\mathbf{x}_j^{(k+1)}$, для которого $\mathbf{h}(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_j^{(k+1)})$ обращается в нуль, т. е.

$$\begin{aligned} \mathbf{h}(\mathbf{x}_K^{(k+1)}, \mathbf{x}_j^{(k+1)}) \approx \mathbf{h}(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_j^{(k+1)}) + \\ + \frac{\partial \mathbf{h}(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_j^{(k+1)})}{\partial x_j} (\mathbf{x}_j^{(k+1)} - \tilde{\mathbf{x}}_j^{(k+1)}) = 0, \end{aligned}$$

откуда следует, что

$$\mathbf{x}_I^{(k+1)} - \tilde{\mathbf{x}}_I^{(k+1)} = - \left(\frac{\partial \mathbf{h}}{\partial \mathbf{x}_I} \right)^{-1} \mathbf{h}(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_I^{(k+1)}). \quad (6.5.9)$$

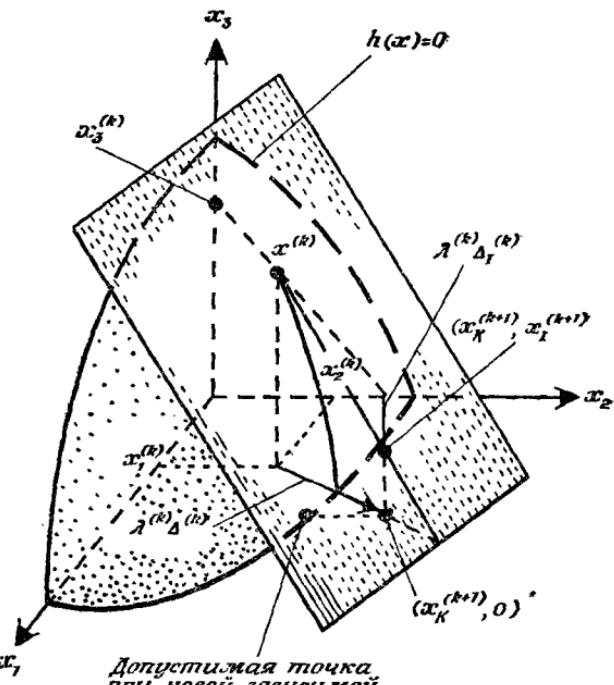
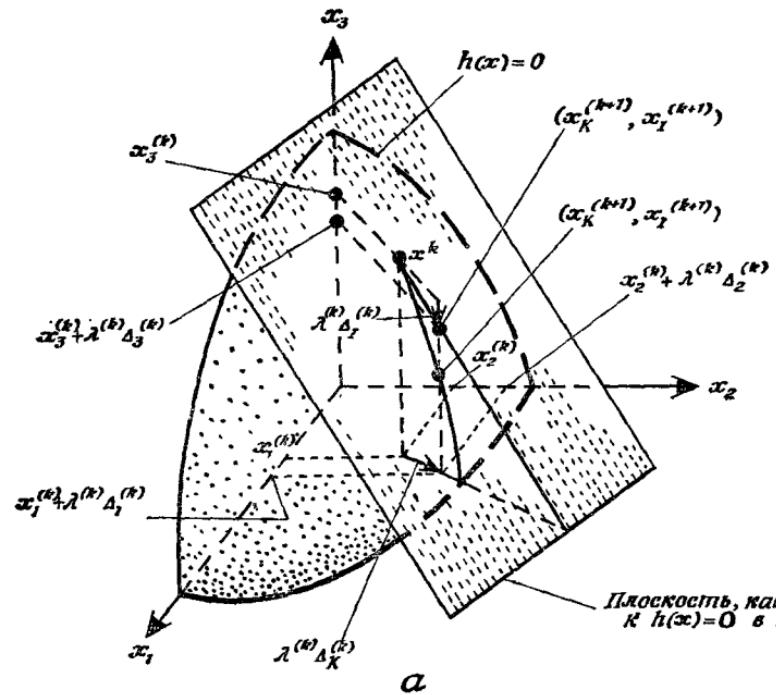
С соотношением (6.5.9) связывают понятие «итерация методом Ньютона»¹⁾. Построенный с помощью этого метода итерационный процесс продолжается до тех пор, пока не будет получен один из приведенных ниже результатов.

Заметим, что если точка $(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_I^{(k+1)})$ является допустимой в пределах заданного интервала, как это показано, например, на фиг. 6.5.3 а, то $\tilde{\mathbf{x}}_I^{(k+1)}$ превращается в $\mathbf{x}_I^{(k+1)}$. (1) Если $f(\mathbf{x}_K^{(k+1)}, \mathbf{x}_I^{(k+1)}) < f(\mathbf{x}_K^{(i)}, \mathbf{x}_I^{(i)})$, где индекс i обозначает последний из допустимых текущих векторов, то итерационный процесс по методу Ньютона заканчивается и оптимизационный поиск продолжают, беря в качестве отправного соотношение (6.5.6). (2) Если точка $\tilde{\mathbf{x}}_I^{(k+1)}$ является внутренней или граничной и тем не менее²⁾ $f(\mathbf{x}_K^{(k+1)}, \tilde{\mathbf{x}}_I^{(k+1)}) > f(\mathbf{x}_K^{(i)}, \mathbf{x}_I^{(i)})$ или если итерационный процесс с помощью (6.5.9) не обеспечивает сходимость за определенное (фиксированное) число шагов (итераций), то производится уменьшение (в некоторой заданной пропорции, например вдвое или в десять раз) значения λ и повторяется итерация по схеме, вытекающей из соотношения (6.5.9). (3) Если первые две из описанных выше ситуаций не реализуются и последняя из найденных с помощью (6.5.9) точек не является ни внутренней, ни граничной, то вносятся надлежащие изменения в базис. Отрезок прямой, соединяющий точку $\tilde{\mathbf{x}}_I^{(k+1)}$ с точкой, полученной из (6.5.9), пересекает поверхность параллелотропа, заданного входящими в задачу (6.5.1) векторами \mathbf{L} и \mathbf{U} , в точке, где одна из переменных (обозначим ее через x_r) подмножества x_I принимает либо свое верхнее предельное значение U_r , либо свое нижнее предельное значение L_r . Рассмотрим фиг. 6.5.3, б; здесь в роли такой переменной x_r выступает x_3 . Переменная x_r , принимающая свое граничное значение, исключается из базиса и заменяется переменной x_s из подмножества x_K . После этого применительно к новому набору независимых переменных выполняется итерация по Ньютону.

Правила выбора переменной, заменяющей исключаемую из базиса переменную x_r , могут быть различными. Значение переменной, вводимой в базисный набор переменных, не должно совпадать с

¹⁾ Если базис содержит ослабляющие переменные, появляющиеся из-за наличия ограничений в виде неравенств, то вместо (6.5.9) используется более эффективное рекуррентное соотношение, описание структуры которого можно найти в соответствующих публикациях (см. цитируемую литературу).

²⁾ При этом осуществляются также некоторые дополнительные контрольно-вычислительные тесты.



Ф и г. 6.5.3. Оптимизационный поиск методом обобщенного приведенного градиента и преобразование зависимых переменных.

Здесь x_3 — зависимая переменная, а x_1 и x_2 — независимые переменные. Имеет место нелинейное ограничение в виде равенства $h(x) = 0$, а также требуется, чтобы выполнялись условия $0 \leq x_j \leq 100$ ($j = 1, 2, 3$). На фиг. а показано положение новой допустимой точки. На фиг. б представлена схема изменения базиса путем исключения из него x_3 и введения в базис переменной x_2 .

соответствующим граничным значением (L_s или U_s), причем предпочтение отдается такой ситуации, когда это несовпадение оказывается значительным.

Пусть

K' — совокупность индексов переменных из множества K , которые могут быть введены в базис в ходе итерационного процесса;

Ω_r — r -я строка матрицы $(\partial h / \partial x_i)^{-1}$, где r — индекс переменной, подлежащей исключению из базиса;

$\partial h / \partial x_k$ — k -й столбец матрицы $(\partial h / \partial x_i)$;

$$d_k = \min \{ (x_k - L_k), (U_k - x_k) \};$$

$Y_j = (x_j^{(k+2)} - x_j^{(k)})$ при допустимых $x_j^{(k+2)}$ и $x_j^{(k)}$;
 ϵ — произвольное малое число.

Тогда, если через s обозначить индекс переменной, подлежащей включению в базис, то значение этого индекса определяется из следующих критериев:

Критерий 1:

$$\left| \Omega_r \frac{\partial h}{\partial x_s} \right| d_s = \max_{K'} \left\{ \left| \Omega_r \frac{\partial h}{\partial x_k} \right| d_k, \left| \Omega_r \frac{\partial h}{\partial x_k} \right| > \epsilon \right\}. \quad (6.5.10)$$

Если (6.5.10) не позволяет определить переменную x_s , например, по той причине, что $|\Omega_r (\partial h / \partial x_s)| < \epsilon$, то для этой цели используется критерий, формулировка которого приводится ниже.

Критерий 2:

а) Если $x_r = L_r$ (т. е. если x_r принимает свое нижнее предельное значение), то

$$\Omega_r \frac{\partial h}{\partial x_s} = \max \left\{ \Omega_r \frac{\partial h}{\partial x_k}, \Omega_r \frac{\partial h}{\partial x_k} Y_k > 0 \right\}. \quad (6.5.11a)$$

б) Если $x_r = U_r$ (т. е. если x_r принимает свое верхнее предельное значение), то

$$\Omega_r \frac{\partial h}{\partial x_s} = \max \left\{ \left| \Omega_r \frac{\partial h}{\partial x_s} \right|, \Omega_r \frac{\partial h}{\partial x_s} Y_k < 0 \right\}. \quad (6.5.11b)$$

Если при выполнении ньютоновской итерации для определения x_r получается $\lambda = 0$, то используются некоторые другие критерии поиска; подробное описание этих критериев можно найти у Абади и Гигу. Схема итераций по Ньютону может быть улучшена, если при переходе от одного вычислительного этапа к другому не допускать слишком больших приращений для элементов базисной матрицы. Например, можно рассматривать отношение Δ_{x_i} к норме Δ_K и в случаях, когда это отношение превышает некоторое заранее установленное критическое значение, исключать переменную x_i из базиса, заменяя ее другой переменной по указанному выше правилу. В особых ситуациях, когда на первых итерациях все базисные переменные являются ослабляющими, при реализации итерационной

процедуры по Ньютону находят практическое применение некоторые специальные правила, позволяющие избегать слишком больших изменений в группе базисных переменных.

6.5.4. КРИТЕРИИ ЗАВЕРШЕНИЯ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

В экстремальной точке составляющие вектора Δ обращаются в нуль. Первый тест, состоящий в исследовании на сходимость, заключается в том, чтобы проверить, выполняется ли условие $|\Delta_j^{(k)}| < \varepsilon |\Delta_j^{(l)}|$, $j = m + 1, \dots, n$, где ε — некоторое малое число. Второй тест состоит в проверке выполнения условия $\Gamma < \Gamma_{\max}$, где

$$\Gamma = \max_K \left\{ \Delta_j^{(k)} \underbrace{(U_j - x_j^{(k)})}_{\Delta_j^{(k)} > 0}, \Delta_j^{(k)} \underbrace{(L_j - x_j^{(k)})}_{\Delta_j^{(k)} < 0} \right\},$$

Γ_{\max} — некоторый (произвольным образом выбранный) параметр.

6.5.5. МЕТОДЫ СОКРАЩЕНИЯ ВРЕМЕНИ РЕАЛИЗАЦИИ ВЫЧИСЛИТЕЛЬНОГО ПРОЦЕССА

Вместо того чтобы заново вычислять $(\partial h / \partial x_I)^{-1}$ каждый раз, когда данная матрица должна использоваться в вычислительной процедуре, оказывается возможным аппроксимировать $(\partial h / \partial x_I)^{-1}$ с помощью метода, который впервые был предложен Билем. Предположим, что в точке $x^{(k)}$ матрица $(\partial h / \partial x_I)^{-1}$ найдена. Тогда в соответствии с методикой, предложенной Абади и Гигоу, на некотором последующем l -м шаге

$$\left(\frac{\partial h(x^{(l)})}{\partial x_I} \right)^{-1} \approx 2 \left(\frac{\partial h(x^{(k)})}{\partial x_I} \right)^{-1} - \left(\frac{\partial h(x^{(k)})}{\partial x_I} \right)^{-1} \left(\frac{\partial h(x^{(l)})}{\partial x_I} \right) \left(\frac{\partial h(x^{(k)})}{\partial x_I} \right).$$

Данная аппроксимация еще подлежит тщательному анализу; вместе с тем можно утверждать, что она выглядит весьма многообещающей. В тех случаях, когда приближение оказывается слишком грубым, с помощью специальной машинной «подпроцедуры» удается заново вычислить элементы матрицы, обратной по отношению к базисной. Абади и Гигоу предлагают для определения погрешностей, возникающих при аппроксимации, ряд соответствующих тестов.

Оптимизационный поиск можно также осуществлять (только внутри допустимой области) с помощью метода наискорейшего спуска, находя при этом ряд следующих одна за другой точек $x^{(k)}$, $x^{(k+1)}$, ..., $x^{(k+p)}$ и перемещаясь в направлении $x^{(k+p)} - x^{(k)}$.

Основываясь на эмпирических данных, Абади и Гигоу рекомендуют принимать $p = 2$.

Более поздние машинные (переведенные на язык кодов) варианты алгоритма обобщенного приведенного градиента, ориентированные, правда, на решение задач с линейными ограничениями, можно найти в специализированной библиотеке VIM Libragy¹⁾ по номенклатуре E4 EDF PHIMAX и E4 EDF PHIMAQ. Имеются сведения о том, что метод обобщенного приведенного градиента успешно применялся Вольфом при решении задач с нелинейными целевыми функциями при наличии от 200 до 300 линейных ограничений и при числе переменных, равном 1000 [34]. Качество работы машинной программы, составленной в 1969 г. на основе алгоритма обобщенного приведенного градиента, проанализировано в гл. 9 путем рассмотрения ряда тестовых задач.

Пример 6.5.1. Метод обобщенного приведенного градиента

Рассмотрим следующую частную задачу:

$$\text{минимизировать } f(\mathbf{x}) = 4x_1 - x_2^2 - 12$$

при ограничениях

$$h_1(\mathbf{x}) = 25 - x_1^2 - x_2^2 = 0,$$

$$g_2(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

Функции $f(\mathbf{x})$, $h_1(\mathbf{x})$, $g_2(\mathbf{x})$, $g_3(\mathbf{x})$ и $g_4(\mathbf{x})$ графически изображены на фиг. 6.0.1.

Поскольку начальная точка $\mathbf{x}^{(0)} = [2 \ 4]^T$ не является допустимой и так как нетривиальным компонентом задачи является ограничение $g_2(\mathbf{x}) \geq 0$, мы вводим аддитивным способом искусственную переменную x_3 в ограничение в виде равенства и вычитаем из левой части ограничения $g_2(\mathbf{x}) \geq 0$ ослабляющую переменную x_4 . Частные производные целевой функции и функций, задающих ограничения, по переменным x_1 и x_2 имеют следующий вид:

$$\frac{\partial f(\mathbf{x})}{\partial x_1} = 4, \quad \frac{\partial f(\mathbf{x})}{\partial x_2} = -2x_2,$$

$$\frac{\partial h_1(\mathbf{x})}{\partial x_1} = -2x_1, \quad \frac{\partial h_1(\mathbf{x})}{\partial x_2} = -2x_2,$$

$$\frac{\partial g_2(\mathbf{x})}{\partial x_1} = 10 - 2x_1, \quad \frac{\partial g_2(\mathbf{x})}{\partial x_2} = 10 - 2x_2.$$

¹⁾ Поступили в указанную библиотеку 9 мая 1968 г.

Итак, в задаче фигурируют четыре переменные x_1, x_2, x_3 и x_4 и два ограничения в виде равенств, выраженные функциями $h_1(x)$ и $g_2(x)$; следовательно, две переменные будут независимыми и две — зависимыми. Начнем вычисления, рассматривая в качестве независимых переменных x_1 и x_2 , а в качестве зависимых (или базисных) — переменные x_3 и x_4 . Приступая к поиску допустимого решения, вычтем из $f(x)$ величину $10^5 x_3$ и будем считать,

что $-10^{10} < x_3 < 0$ и $0 < x_4 < 10^{10}$. Тогда $\frac{\partial f(x)}{\partial x_1} = 4$, $\frac{\partial f(x)}{\partial x_2} = 8$, $\frac{\partial f(x)}{\partial x_3} = -10^5$.

Для формирования приведенного градиента нужны матрицы $[\partial h(x^{(0)})/\partial x_I]^{-1}$ и $[\partial h(x^{(0)})/\partial x_K]$; другими словами, необходим якобиан, вычисленный в $x^{(0)}$ по каждому из подмножеств переменных:

$$\frac{\partial h(x^{(0)})}{\partial x_I} = \begin{bmatrix} \frac{\partial h_1(x^{(0)})}{\partial x_3} & \frac{\partial h_1(x^{(0)})}{\partial x_4} \\ \frac{\partial g_2(x^{(0)})}{\partial x_3} & \frac{\partial g_2(x^{(0)})}{\partial x_4} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Из (6.5.4) для приведенного градиента получаем

$$\begin{aligned} \frac{\partial f(x^{(0)})}{\partial x_K} &= \left[\frac{\partial f(x^{(0)})}{\partial x_1} \quad \frac{\partial f(x^{(0)})}{\partial x_2} \right] = [4 \quad -8] - \\ &- [-10^5 \quad 0] \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}^{-1} \begin{bmatrix} -4 & -8 \\ 6 & 2 \end{bmatrix} = [-3,99996 \cdot 10^5 \quad \\ &\quad -8,00008 \cdot 10^5]. \end{aligned}$$

Далее вычисляется норма приведенного градиента с тем, чтобы проверить, не оказывается ли она ниже числового значения критерия, определяющего условия завершения вычислительного цикла при данной текущей точке и с учетом граничных значений переменных; выясняется, что норма градиента равняется $8,944 \cdot 10^5$, что превышает значение упомянутого выше критерия.

После этого находятся направления оптимизационного поиска для независимых переменных; для этого определяются

$$\Delta_1^{(0)} = -\frac{\partial f(x^{(0)})}{\partial x_1} = -3,99996 \cdot 10^5$$

и

$$\Delta_2^{(0)} = -\frac{\partial f(x^{(0)})}{\partial x_2} = -8,00008 \cdot 10^5.$$

По завершении этой вычислительной процедуры ищутся направления поиска для зависимых переменных; используя (6.5.7), по-

лучаем

$$\begin{aligned}\Delta_{\mathbf{x}_I}^{(0)} &= - \left[\frac{\partial \mathbf{h}(\mathbf{x}^{(0)})}{\partial \mathbf{x}_I} \right]^{-1} \left[\frac{\partial \mathbf{h}(\mathbf{x}^{(0)})}{\partial \mathbf{x}_K} \right] \Delta_{\mathbf{x}_K}^{(0)} = \\ &= - \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} -4 & -8 \\ 6 & 2 \end{bmatrix} \begin{bmatrix} -3,99996 \cdot 10^5 \\ -8,00008 \cdot 10^5 \end{bmatrix} = - \begin{bmatrix} 8,000 \cdot 10^6 \\ 4,000 \cdot 10^6 \end{bmatrix}.\end{aligned}$$

Определив направления поиска, найдем λ , минимизирующее $f(\mathbf{x})$ [см. (6.5.6)]:

Переменные и целевая функция	$\mathbf{x}^{(0)}$	Этап поиска		
		1	2	последний
x_1	2	2,250	2,125	2,250
x_2	4	4,500	4,250	4,500
x_3	-5	$-2,842 \cdot 10^{-14}$	-2,500	0
x_4	6	8,500	7,250	8,500
$f(\mathbf{x})$	$\sim 5 \cdot 10^5$	-23,250	$24,997 \cdot 10^5$	-23,250

Окончательное значение λ равняется $6,250 \cdot 10^{-7}$.

Теперь используем последовательно (6.5.9), стремясь подобрать значения зависимых переменных таким образом, чтобы вектор \mathbf{x} стал допустимым:

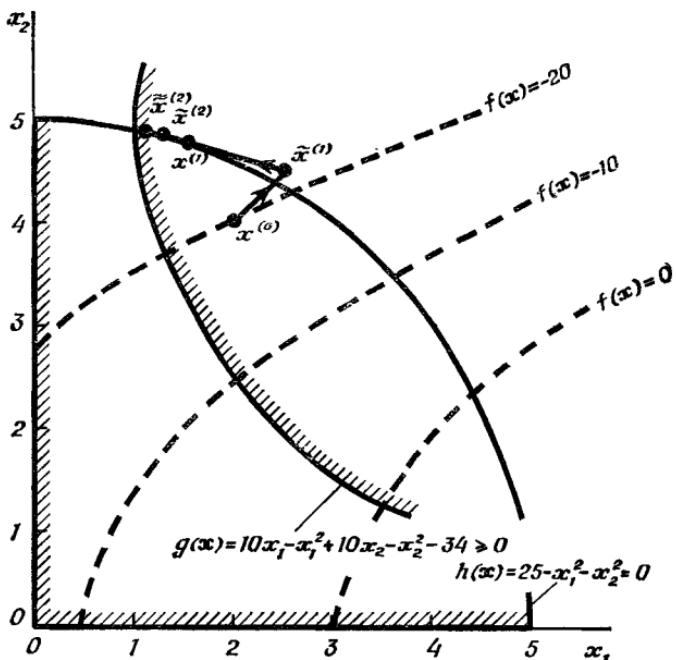
$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 8,500 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} -0,3125 \\ -0,3125 \end{bmatrix} = \begin{bmatrix} 0,3125 \\ 8,1875 \end{bmatrix}.$$

Поскольку вектор $\tilde{\mathbf{x}} = [2,250 \ 4,500 \ 0,3125 \ 8,187]^T$ не был допустимым, осуществляется поиск нового значения λ ; в результате выполнения пяти итераций методом Ньютона находим допустимую точку:

$$\mathbf{x}^{(1)} = [1,568 \ 4,748 \ 0 \ 4,161]^T.$$

При этом и x_1 и x_2 продолжают оставаться независимыми переменными, а x_3 обращается в нуль.

Теперь переходим к вычислению составляющих нового редуцированного градиента и, следовательно, опять приступаем к выполнению описанной выше процедуры. Последовательность значений составляющих вектора \mathbf{x} указана на фиг. П.6.5.1. Приведенный градиент вычисляется при этом еще только раз, а все остальные вычислительные операции проводятся с целью нахождения допустимого вектора и определения момента завершения вычислительного цикла. В общей сложности значения функций, задающих ограничения, вычислялись 103 раза, целевая функция 110 раз, а градиент



Ф и г. П.6.5.1. Траектория поиска методом обобщенного приведенного градиента.

целевой функции 27 раз. При этом замена базисных переменных осуществлялась дважды (фактически это сводилось к перенумерации переменных).

ЗАДАЧИ¹⁾

6.1. Выполните линеаризацию фигурирующих в задачах 6.17, 6.22, 6.28 и 6.29 целевых функций и ограничений в окрестности выбранной точки. Для одномерных и двумерных задач изобразите графически исходные целевые функции и их линейные аппроксимации.

6.2. Можете ли вы линеаризовать целевую функцию задачи 6.20?

6.3. Проверьте, выполняются ли для задач 6.17, 6.18, 6.20, 6.23, 6.28 и 6.29 условия, гарантирующие сходимость к оптимальному решению (см. стр. 242—243).

6.4. Выполните два этапа алгоритма Гриффица и Стюарта для задачи, приведенной в примере 6.2.1 (используя вычислительную процедуру, приведенную в связи с рассмотрением примера 6.1.1). Для получения допустимого вектора \mathbf{x} либо воспользуйтесь

¹⁾ Ряд задач, имеющих отношение к содержанию данной главы, можно найти также в конце гл. 7 и в приложении А.

машинной программой для алгоритма линейного программирования, либо выполните все вычисления вручную.

6.5. Повторите задачу 6.4 применительно к условиям задачи 6.25.

6.6. Составьте машинную программу, реализующую алгоритм ПОП II. Решите задачу, приведенную в примере 6.2.1. Ответьте на следующие вопросы:

а) Значения каких параметров необходимо задать (из рекомендуемого набора), чтобы получить искомое решение?

б) Каким образом сравниваются значения производных, полученные численным методом, с соответствующими значениями этих производных, найденными в аналитическом виде?

в) Какие трудности возникают при решении рассматриваемой вами задачи и как они выглядят по сравнению с трудностями, описанными в разд. 6.1.3?

г) Какие усовершенствования алгоритма могли бы вы предложить?

6.7. Составьте матрицу приращений, структура которой приведена на фиг. 6.1.3, необходимую при использовании алгоритма ПОП, для задачи, приведенной в примере 6.1.1 (за исключением столбцов с порядковым номером $j \geq n + 1$ и строк с порядковым номером $i > n + m + 2$). Положите $\delta_j = 0,001$. Вычислите также погрешности за счет линеаризации.

6.8. Найдите проекцию градиента целевой функции

$$f(x) = 5x_1 - 3x_2 + 6x_3$$

на $(x_1 - x_2)$ -плоскость (данная плоскость соответствует ограничению $x_3 = 0$). Изобразите графически $\nabla f(x)$ и найденную вами проекцию градиента на $(x_1 - x_2)$ -плоскость.

6.9. Повторите задачу 6.8, взяв в качестве целевой функции

$$f(x) = 2x_1^2 + x_2^2 + 2x_3^2.$$

При этом вычислите градиент этой функции в точке $(1, -1, 1)$ и спроектируйте его на $(x_1 - x_2)$ -плоскость.

6.10. Предложите процедуру минимизации, основанную на методе проекции градиента, с тем чтобы найти минимум функции

$$f(x) = x_1^2 + x_2^2 + x_3^2 - 2x_1x_2$$

при ограничениях

$$h_1(x) = 2x_1 + x_2 - 4 = 0$$

и

$$h_2(x) = 5x_1 - x_3 - 8 = 0.$$

Реализуйте практически три этапа предложенной вами процедуры. На каждом этапе укажите составляющие градиента целевой функции, структуру проектирующей матрицы, составляющие вектора независимых переменных и значение $f(x)$.

6.11. Определите проекцию градиента в точке $\mathbf{x} = [1 \ 1]^T$ для следующей задачи:

$$\text{минимизировать } f(\mathbf{x}) = 5x_1^2 - 3x_2^2$$

при ограничениях

$$x_1 \geq 0,$$

$$x_2 \geq 0.$$

6.12. Определите проекции градиента целевой функции

$$f(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 2x_1x_2 - 2x_1x_3 - 2x_2x_3$$

на ограничения

$$h_1(\mathbf{x}) = 2x_1 + x_2 - 6 = 0,$$

$$g_2(\mathbf{x}) = x_1 - x_3 - 8 > 0$$

из точки

$$\mathbf{x} = [1 \ 1 \ 1]^T.$$

6.13. Должна ли проектирующая матрица быть обязательно положительно определенной?

6.14. Покажите, что соотношение (6.3.11) справедливо и для обобщенной проектирующей матрицы.

6.15. Докажите, что при исключении ограничения в виде неравенства из группы активных ограничений правомерно использовать соотношение (6.3.14).

6.16. Подготовьте машинную программу, реализующую алгоритм, предложенный Дэвисом (см. подразд. 6.3.3). Вначале составьте вариант программы, пригодный для решения задач с линейными ограничениями, а затем видоизмените подготовленную вами программу применительно к условиям нелинейных ограничений. Каким образом вы считаете возможным эффективно учитывать в ходе вычислений ограничения в виде равенств?

6.17. Рассмотрите следующую задачу (вытекающую из математической модели функционирования сушильного аппарата):

$$\text{минимизировать } f(\mathbf{x}) = 0,0064x_1 [\exp(-0,184x_1^{0,3}x_2 - 1)]$$

при ограничениях

$$g_1(\mathbf{x}) = 1,2 \cdot 10^{13} - (3000 + x_1)x_1^2x_2 > 0,$$

$$g_2(\mathbf{x}) = 4,1 - \exp(0,184x_1^{0,3}x_2) > 0,$$

где x_1 — скорость газа, а x_2 — глубина ванны. Начиная поиск из точки $\mathbf{x} = [31000 \ 0,345]^T$, покажите, что локальный оптимум соответ-

ствует точке $\mathbf{x}^* = [31800 \ 0,342]^T$. Насколько строго в этой точке удовлетворяются условия, представленные указанными выше ограничениями? Является ли локальный минимум истинным минимумом? Является ли локальный минимум глобальным минимумом? (См. гл. 2.) Можно ли прийти в ту же точку \mathbf{x}^* , выбрав в качестве отправной точку $\mathbf{x}^{(0)} = [0 \ 0]^T$? Точку $\mathbf{x}^{(0)} = [1 \ 1]^T$?

6.18. Суммарные затраты, связанные с сооружением ректификационной колонны, можно записать в виде

$$C = C_p AN + C_s HAN + C_f + C_d + C_b + C_l + C_x, \quad (a)$$

где C — суммарные затраты, долл;

C_p — стоимость одного квадратного метра горизонтального перекрытия монолитными плитами, долл/фут²;

A — площадь поперечного сечения колонны, фут²;

N — число монолитных плит (равняется числу горизонтальных перекрытий), N_{\min} — минимальное число горизонтальных перекрытий (монолитных плит);

C_s — удельные затраты, связанные с сооружением фермы, долл/фут³;

H — расстояние между горизонтальными перекрытиями (между монолитными плитами), фут;

C_f — стоимость (плюс затраты на монтаж) подающего насоса (насоса накачки), долл;

C_d — стоимость (плюс затраты на монтаж) системы насосов, обеспечивающих ректификационный процесс, долл;

C_b — стоимость (плюс затраты на монтаж) насоса откачки низких фракций, долл;

C_l — стоимость (плюс затраты на монтаж) насоса, обеспечивающего повторную перегонку промежуточных фракций, долл;

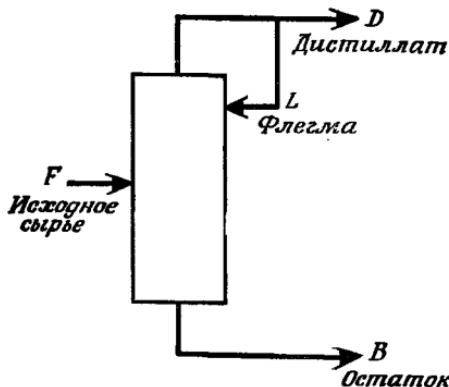
C_x — другие фиксированные затраты, долл.

Задача заключается в минимизации суммарных затрат при заданных спецификационных характеристиках выходного продукта и пропускной способности ректификационной колонны, а также при фиксированных стоимостных характеристиках, относящихся к функционированию всех видов насосных установок (т. е. C_f , C_s , C_d и C_b фиксированы). После подбора строительного материала становятся фиксированными C_p , C_s и C_x .

Переменные, описывающие технологический процесс, связаны двумя (установленными эмпирически) соотношениями:

$$\frac{L}{D} = \left[\frac{1}{1 - (N_{\min}/N)} \right]^\alpha \left(\frac{L}{D} \right)_{\min}, \quad (b)$$

$$A = K(L + D)^\beta. \quad (b)$$



Для простоты положим $\alpha = \beta = 1$; тогда

$$\frac{L}{D} = \left[\frac{1}{1 - (N_{\min}/N)} \right] \left(\frac{L}{D} \right)_{\min}, \quad (\delta')$$

$$A = K(L + D). \quad (\delta')$$

Некоторые из типовых ректификационных колонн характеризуются следующими значениями указанных выше параметров:

$$C_p = 30, \quad C_x = 8000,$$

$$C_s = 10, \quad F = 1500,$$

$$H = 2, \quad D = 1000,$$

$$C_f = 4000, \quad N_{\min} = 5,$$

$$C_d = 3000, \quad \frac{L}{D} = 1,$$

$$C_b = 2000, \quad K = \frac{1}{100} \frac{\text{ч} \cdot \text{фут}^3}{\text{фунт}}.$$

Затраты на перекачку, обеспечивающую повторную перегонку, выражаются формулой

$$C_L = 5000 + 0,7L. \quad (\gamma)$$

а) Определите подходящее решение для этого процесса, т. е. значения независимых переменных. Какие из переменных являются зависимыми?

б) Определите минимальные суммарные затраты и найдите соответствующие им числовые значения переменных.

6.19. Майлендер [35] показал, что сформулированную Боксом [36] задачу, заключающуюся в минимизации

$$f(\mathbf{x}) = b_0 + a_{01}x_1 + \left(\sum_{j=2}^5 a_{0j}x_j \right)x_1$$

при ограничениях

$$0 \leq a_{11}x_1 + \left(\sum_{j=2}^5 a_{ij}x_j \right) x_1 \leq b_1, \quad i = 1, 2, 3,$$

$$x_1 \geq 0, \quad 1,2 \leq x_2 \leq 2,4, \quad 20,0 \leq x_3 \leq 60, \quad 9,0 \leq x_4 \leq 9,3, \\ 6,5 \leq x_5 \leq 7,0,$$

можно преобразовать в задачу линейного программирования с помощью подстановок $y_i = x_1 x_i$ ($i = 2, 3, 4, 5$) и $y_1 = x_1$. При этом получается следующая задача линейного программирования:

$$\text{минимизировать } g(y) = b_0 + \sum_{i=1}^5 a_{0i}y_i$$

при ограничениях

$$0 \leq \sum_{i=1}^5 a_{ij}y_i \leq b_i, \quad i = 1, 2, 3, \\ y_i \geq 0, \quad i = 1, 5,$$

$$y_2 - 1,2 y_1 \geq 0, \quad 2,4y_1 - y_2 \geq 0,$$

$$y_3 - 20,0y_1 \geq 0, \quad 60,0y_1 - y_3 \geq 0,$$

$$y_4 - 9,0y_1 \geq 0, \quad 9,3y_1 - y_4 \geq 0,$$

$$y_5 - 6,5y_1 \geq 0, \quad 7,0y_1 - y_5 \geq 0.$$

Используя метод нелинейного программирования (НЛП) для исходной задачи, подтвердите оптимальность решения, полученного с помощью линейного программирования, а именно убедитесь, что

$$g = 5\,280\,344,9,$$

$$y_1 = 4,53743, \quad y_2 = 10,88983, \quad y_3 = 272,24584,$$

$$y_4 = 42,19811, \quad y_5 = 31,76202,$$

т. е. если вернуться к исходным переменным, то будем иметь

$$f = -5\,280\,344,9,$$

$$x_1 = 4,53743, \quad x_2 = 2,40000, \quad x_3 = 60,00000,$$

$$x_4 = 9,30000, \quad x_5 = 7,00000.$$

Интересно отметить, что исходная нелинейная задача не является выпуклой: допустимая область невыпукла, целевая функция также не является выпуклой. Тем не менее полученная в результате указанной выше замены переменных линейная задача (эквивалентная исходной) оказывается выпуклой.

Ниже указаны значения фигурирующих в задаче постоянных коэффициентов:

$$\begin{aligned}
 a_{01} &= -8\,720\,288,795, & a_{23} &= 12,9492, \\
 a_{02} &= -150\,512,524, & a_{24} &= 10\,236,8839, \\
 a_{03} &= -156,695, & a_{25} &= 13\,176,7859, \\
 a_{04} &= -476\,470,319, & a_{31} &= -326\,669,5059, \\
 a_{05} &= -729\,482,825, & a_{32} &= 7390,6840, \\
 a_{11} &= -145\,421,4004, & a_{33} &= -27,8987, \\
 a_{12} &= 2931,1506, & a_{34} &= 16\,643,0759, \\
 a_{13} &= -40,4279, & a_{35} &= 30\,988,1459, \\
 a_{14} &= 5\,106,1920, & b_0 &= -24\,345,0, \\
 a_{15} &= 15\,711,3600, & b_1 &= 294\,000,0 \\
 a_{21} &= -155\,011,1055, & b_2 &= 294\,000,0, \\
 a_{22} &= 4\,360,5334, & b_3 &= 277\,200,0.
 \end{aligned}$$

6.20. Найдите минимальное значение интеграла

$$f(x) = \int_0^1 y dx$$

при ограничении

$$\int_0^1 \left[1 + \left(\frac{dy}{dx} \right)^2 \right]^{1/2} dx = 4.$$

6.21. Определите максимальное и минимальное расстояния от начала координат кривой:

$$5x_1^2 + 6x_1x_2 + 5x_2^2 = 8.$$

6.22. Найдите максимальное значение функции

$$\begin{aligned}
 f(x) &= 20,21 - 46,38x_1 + 59,42x_2 + 16,30x_1x_2 + \\
 &\quad + 8,34x_1^2 + 4,26x_2^2
 \end{aligned}$$

на ограничивающей окружности с радиусом, равным 3. Какова структура вектора x ?

6.23. Фирма, выпускающая химическую продукцию, продает три вида продукции. Этой фирмой установлено, что функция дохода имеет вид $f = 10x + 4,4y^2 + 2z$, где x , y и z представляют собой месячные нормы выработки продукции первого, второго и третьего вида соответственно. На основе анализа графиков распределения уровней спроса на выпускаемую продукцию фирмой выявлены следующие предельные условия норм выработки упомянутых выше видов

продукции:

$$\begin{aligned}x &> 2, \\ \frac{1}{2} z^2 + y^2 &> 3.\end{aligned}$$

Кроме того, фирма должна иметь в виду, что объемы имеющихся в наличии сырьевых материалов ограничены; это приводит к следующим ограничениям, которые должны учитываться при составлении производственного графика:

$$\begin{aligned}x + 4y + 5z &< 32, \\ x + 3y + 2z &< 29.\end{aligned}$$

Составьте наиболее рациональный для данной фирмы производственный график и определите максимальное значение функции суммарного дохода фирмы.

6.24. Минимизируйте половину квадрата расстояния от точки $\mathbf{x} = [2 \ 3 \ -1]^T$ до тетраэдра, заданного соотношениями

$$\begin{aligned}w = -x_1 - x_2 - x_3 + 3 &\geq 0, \\ x_1, x_2, x_3 &\geq 0.\end{aligned}$$

Примечание. Целевая функция имеет вид

$$f(\mathbf{x}) = \frac{1}{2} [(x - 2)^2 + (y - 3)^2 + (z + 1)^2].$$

6.25. При определении условий химического равновесия сталкиваются со следующей задачей:

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^n x_i \left(w_i + \ln P + \ln \frac{x_i}{\sum_{i=1}^n x_i} \right)$$

при ограничениях

$$\begin{aligned}x_1 + 2x_2 + 2x_3 + x_6 + x_{10} &= 2, \\ x_4 + 2x_5 + x_6 + x_7 &= 1, \\ x_3 + x_7 + x_8 + 2x_9 + x_{10} &= 1.\end{aligned}$$

Пусть $P = 750$, а w_i имеет следующие значения:

i	w_i	i	w_i
1	-10,021	6	-18,918
2	-21,096	7	-28,032
3	-37,986	8	-14,640
4	-9,846	9	-30,594
5	-28,653	10	-26,111

Определите \mathbf{x}^* и $f(\mathbf{x}^*)$.

6.26. Система, состоящая из трех последовательно соединенных теплообменников (см. схему на стр. 329), реализует обычный поэтапный процесс теплообмена без повторных циклов. Обозначения для температур, характеризующих каждый из этапов теплообмена, приведены на схеме (предполагается, что температуры измеряются по шкале Фаренгейта). Процесс теплообмена осуществляется следующим простым способом: холодная жидкость нагревается в каждом из теплообменников горячей жидкостью, поток которой перпендикулярен потоку нагреваемой жидкости. Допустим, что каждый из теплообменников можно описать следующей стационарной макромоделью :

$$WC_p(T_n - T_{n-1}) = U_n A_n (t_{n1} - T_n).$$

Предположим, что скорости всех потоков одинаковы и равняются W , а также будем считать, что теплоемкость жидкости в любом узле системы постоянна и равна C_p . Задача заключается в определении площади поверхности теплообмена A для каждого из теплообменников, так чтобы суммарная площадь, обеспечивающая теплообмен в системе, была минимальной. Ниже приводятся входные (исходные) данные для фигурирующих в приведенной модели постоянных коэффициентов:

$$\begin{aligned} T_0 &= 100 \text{ }^{\circ}\text{F}, & U_1 &= 120 \text{ БЕТ/ч} \cdot (\text{фут})^2 \cdot \text{ }^{\circ}\text{F}, \\ T_3 &= 500 \text{ }^{\circ}\text{F}, & U_2 &= 80 \text{ БЕТ/ч} \cdot (\text{фут})^2 \cdot \text{ }^{\circ}\text{F}, \\ t_{11} &= 300 \text{ }^{\circ}\text{F}, & U_3 &= 40 \text{ БЕТ/ч} \cdot (\text{фут})^2 \cdot \text{ }^{\circ}\text{F}, \\ t_{21} &= 400 \text{ }^{\circ}\text{F}, \\ t_{31} &= 600 \text{ }^{\circ}\text{F}, \\ WC_p &= 10^5 \text{ БЕТ/}^{\circ}\text{F}. \end{aligned}$$

(Примечание. Данная задача нередко решается с помощью динамического программирования.)

6.27. Минимизируйте

$$f(x) = \frac{2}{x_1 + 0,5} + \frac{1}{x_2 + 0,2} + \frac{3}{x_3 + 0,5}$$

при ограничениях

$$4x_1 + 7x_2 + 3x_3 \leq 10,$$

$$3x_1 + 4x_2 + 5x_3 \leq 8,$$

$$x_1, x_2, x_3 \geq 0.$$

6.28. Определите глобальный минимум и глобальный максимум функции

$$f(x) = x_2 \sin x_2 - 4x_1$$

при ограничениях

$$g_1(\mathbf{x}) = x_2 \sin x_2 - x_1^3 - x_1 = 0,$$

$$g_2(\mathbf{x}) = x_1 > 0,$$

$$g_3(\mathbf{x}) = x_2 > 0,$$

$$g_4(\mathbf{x}) = x_1 < 4,$$

$$g_5(\mathbf{x}) = x_2 < 4.$$

(Примечание. Приведенная здесь целевая функция имеет несколько минимумов и максимумов.)

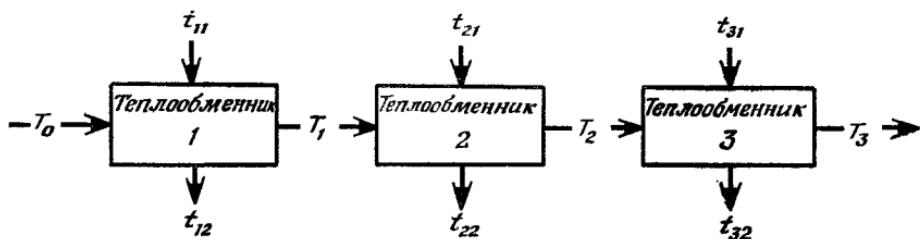
6.29. Определите минимум целевой функции

$$f(\mathbf{x}) = x_1^2 + x_2^2 - 3x_1 + 15x_2$$

при ограничении

$$(x_1 + x_2)^2 - 4(x_1 - x_2) = 0.$$

(Примечание. Данная целевая функция имеет два локальных минимума.)



ЛИТЕРАТУРА

1. Griffith R. E., Stewart R. A., *Management Sci.*, 7, 379 (1961).
2. Glass H., Cooper L., *J. Assoc. Computer Mach.*, 12, 71 (1965).
3. Olson F. A., ACM SIGMAP Workshop, IBM Data Processing Div., June 14—15, 1966.
4. Smith H. V., A Process Optimization Program for Nonlinear Systems: POP II, IBM Gen. Program Library 7090 H9 IBM 0021, 1965.
5. Zwart P. B., SIGMAP Workshop on Nonlinear Programming, Yorktown Heights, N. Y., 1967.
6. Wilson R. B., Ph. D. Dissertation, Harvard Univ. Graduate School of Business Administration, Boston, 1963.
7. Beale E. M. L., Numerical Methods, in: Nonlinear Programming, Abadie J., ed., Interscience Publ., N. Y., 1967.
8. Graves G. W., Whinston A. B., Univ. Calif. Western Management Sci. Inst. Paper 108, Los Angeles, Sept. 1966.
9. Frank M., Wolfe P., *Naval Res. Logistics Quart.*, 3, 95 (1956).
10. Barnes G. K., M. S. Thesis, Univ. of Texas, Austin, Tex., 1967.
11. DiBella C. W., Stevens W. F., *Ind. Eng. Chem. Process Design Develop.*, 4, 16 (1965).
12. Frisch R., The Multiple Method for Linear Programming, Mem. Univ. Socialokon Inst., Oslo, Oct. 1955.

13. Zoutendijk G., Methods of Feasible Directions, Elsevier Publ. Co., Amsterdam, 1960.
14. Rosen J. B., *J. Soc. Ind. Appl. Math.*, 8, 181 (1960); 9, 514 (1961).
15. Rosen J. B., Merrill R. P., Gradient Projection — GP90, Share Program 7090-H2-3430GP90.
16. Murtagh B. A., Sargent R. W. H., Chap. 14 in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
17. Householder A. S., The Theory of Matrices in Numerical Analysis, Blaisdell Publ. Co., Waltham, Mass., 1964, p. 8.
18. Künzi H. P., Krelle W., Nonlinear Programming, Blaisdell Publ. Co., Waltham, Mass., 1966.
19. Fletcher R., *J. Inst. Math. Appl.*, 5, 2 (1969).
20. Cross K. E., AEC Doc. K-1746, May 30, 1968.
21. Davidon W. C., AEC Doc. ANL-5990 (rev.), 1959.
22. Goldfarb D., Ph. D. Dissertation, Princeton Univ., Princeton, N. J., 1966.
23. Goldfarb D., Lapidus L., *Ind. Eng. Chem. Fundamentals*, 7, 142 (1968).
24. Davies D., The Use of Davidon's Method in Nonlinear Programming, ICI Ltd. Rept. MSDH/68/110, Aug. 1968; Doc. N69-33235 from CFSTI, Springfield, Va.
25. Murtagh B. A., Sargent R. W. H., Chap. 14 in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
26. Householder A. S., The Theory of Matrices in Numerical Analysis, Blaisdell Publ. Co., Waltham, Mass., 1964, p. 10.
27. Zoutendijk G., Methods of Feasible Directions, Elsevier Publ. Co., Amsterdam, 1960.
28. Zoutendijk G., *SIAM J. Control*, 4, 194 (1966).
29. Wolfe P., Recent Developments in Nonlinear Programming, Rand Corp. Rept. R-401-PR, 1962.
30. Abadie J., Carpentier J., Généralisation de la méthode du gradient réduit de Wolfe au cas de contraintes nonlinéaires, Proc. IFORS Conf.; Chap. 4 in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
31. Faure P., Huard P., Rev. Française Recherche Opérationnelle, 9, 167 (1965).
32. Abadie J., Guigou J., Gradient réduit généralisé, Électricité de France Note HI 069/02, April 15, 1969.
33. Wolfe P., *Notices Am. Math. Soc.*, 9 (4), 308 (1962); Methods of Nonlinear Programming, in: Recent Advances in Mathematical Programming, Graves R. L., Wolfe P., eds., McGraw-Hill, N. Y., 1963, pp. 76—77.
34. Künzi H. P., *Unternehmensforschung*, 12, 1 (1968).
35. Mylander W. C., *Computer J.*, 8, 391 (1965).
36. Box M. J., *Computer J.*, 8, 42 (1965).

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

ОБЩИЕ ВОПРОСЫ

- Abadie J., Numerical Experiments with the GRG Method, in: Integer and Nonlinear Programming, Abadie J., ed., North Holland Publ. Co., Amsterdam, 1970.
- Bellmore M., Greenberg H. J., Jarvis J. J., Generalized Penalty-function Concepts in Mathematical Optimization, *Operations Res.*, 18, 193 (1970).
- Carpentier J., Abadie J., Généralisation de la méthode du gradient réduit de Wolfe au cas de contraintes nonlinéaires, Proc. IFORS Congr., Cambridge, Mass., Aug. 29 — Sept. 2, 1966.
- Charnes A., Cooper W. W., Nonlinear Power of Adjacent Extreme Point Methods in Linear Programming, *Econometrica*, 25, 132 (1957).
- Davies D., Some Practical Methods of Optimization: Notes for the NATO Summer

- School, on Integer and Nonlinear Programming, Academic Press, N. Y., June 18—20, 1969.
- Davies D., Review of Constrained Optimization, Clearinghouse for Federal Scientific and Technical Information, Document N 69-36898, Sept. 30, 1968.
- Dennis J. B., Mathematical Programming and Electrical Networks, MIT, Cambridge, Mass., 1959.
- Fauré P., Huard P., Résolution de programmes mathématiques à fonction nonlinéaire par la méthode du gradient réduit, *Rev. Franc. Recherche Opérationnelle*, № 36, 167 (1965).
- Fletcher R., Clearinghouse for Federal Scientific and Technical Information, Document N 69-37016, Sept. 30, 1968.
- Griffith R. E., Stewart R. A., A Nonlinear Programming Technique for Optimization of Continuous Processing Systems, *Management Sci.*, 7, 379 (1961).
- Kleinbohm K., Ein Verfahren zur approximativen Lösung von konvexen Programmen, Ph. D. dissertation, Univ. of Zurich, 1966.
- Leviton E. S., Polyak B. T., Constrained Minimization Methods, *USSR Computational Math. and Math. Phys.*, 6, 1 (1966).
- Rosen J. B., The Gradient Projection Method for Nonlinear Programming, Part I, *J. Soc. Ind. Appl. Math.*, 8, 181 (1960); Part II, 9, 514 (1961); IBM Share Program 1399.
- Wolfe P., Methods of Nonlinear Programming, in: Nonlinear Programming, Abadie J., ed., North Holland Publ. Co., Amsterdam, 1967.
- Zoutendijk G., Methods of Feasible Directions, American Elsevier, N. Y., 1960.

СХОДИМОСТЬ АЛГОРИТМОВ

- Topkis D. M., Veinott A. E., On the Convergence of Some Feasible Direction Algorithms for Nonlinear Programming, *J. SIAM Control.*, 5, 268 (1967).
- Zangwill W. I., Convergence Conditions for Nonlinear Programming Algorithms, *Management Sci.*, 16, 1 (1969).

ДРУГИЕ МЕТОДЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ, ИСПОЛЬЗУЮЩИЕ ЛИНЕАРИЗАЦИЮ

- Cheney E. W., Goldstein A. A., Newton's Method for Convex Programming and Tchebycheff Approximation, *Numerical Math.*, 4, 253 (1959).
- De Remus L. V., Nonlinear Partition Programming, SIGMAP Workshop on Nonlinear Programming, IBM Corporation, Yorktown Heights, N. Y., 1968.
- DiBella C. W., Stevens W. F., Process Optimization by Nonlinear Programming, *Ind. Eng. Chem. Process Design Develop.*, 4, 16 (1965).
- Glass H., Cooper L., Sequential Search: A Method for Solving Constrained Optimization Problems, *J. ACM*, 12, 71 (1965).
- Graves G. W., Whinston A. B., The Application of a Nonlinear Programming Algorithm to a Second Order Representation of the Problem, Univ. of California at Los Angeles, Western Management Sci. Inst. Paper 108, Sept. 1966 (AD641196).
- Hartley H. O., Hocking R. R., Convex Programming by Tangential Approximation, *Management Sci.*, 9, 600 (1963).
- Hartley H. O., et al., Convex; A Computer Program for Solving Convex Programs, Techn. Rep. № 23, Texas A and M Univ., College Station, Texas, July 1970.
- Hilleary R. R., The Tangent Search Method of Constrained Minimization, U. S. Naval Postgraduate School Techn. Rept. Res. Paper 59, March 1966 (AD 632121).

- Kelley J. E., The Cutting-plane Method for Solving Convex Programs, *J. Soc. Ind. Appl. Math.*, 8, 703 (1960); Method of Gradients, Chap. 6 in: Optimization Techniques with Applications, Leitmann G., ed., Academic Press, N. Y., 1962.
- Künzi H. P., The Duoplex Method in Nonlinear Programming, *J. SIAM Control*, 4, 130 (1966).
- McGuire S. W., Hocking R. R., Hartley H. O., Spherical Programming: A Convex Programming Algorithm, Techn. Rep. № 5, Inst. of Statistics, Texas A and M Univ., College Station, Texas, Oct. 1968.
- Miele A., Huang H. Y., Heideman J. C., Sequential Gradient-restoration Algorithm for Minimization of Constrained Functions — Ordinary and Conjugate Gradient Methods, *J. Optimization Theory Appl.*, 4, 213 (1969).
- Mills D. H., Extending Newton's Method to Systems of Inequalities, Proc. 6th Intern. Symp. on Math. Programming, Princeton, N. J., Aug. 1967.
- Muggele R. A., A Program for Optimal Control of Nonlinear Processes, *IBM Systems J.*, 1, 2 (1962).
- Pinsker I. S., The Alternance Method (for Solution of Problems in Nonlinear Programming), *Automatic Remote Control*, 25, 280 (1964).
- Shanno D. F., An Accelerated Gradient Projection Method for Linearly Constrained Nonlinear Estimation, *SIAM J. Appl. Math.*, 18, 322 (1970).

Глава 7

ПРОЦЕДУРЫ МИНИМИЗАЦИИ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ: МЕТОДЫ ШТРАФНЫХ ФУНКЦИЙ



В публикациях по нелинейному программированию *методы штрафных функций* представлены в различных вариантах, которые, однако, имеют одну общую черту: во всех этих методах осуществляется преобразование задачи нелинейного программирования при наличии ограничений либо в одну (эквивалентную исходной) задачу без ограничений, либо в эквивалентную последовательность задач без ограничений. Пусть, например, мы хотим найти минимум функции

$$f(x) = (x_1 - 3)^2 + (x_2 - 2)^2$$

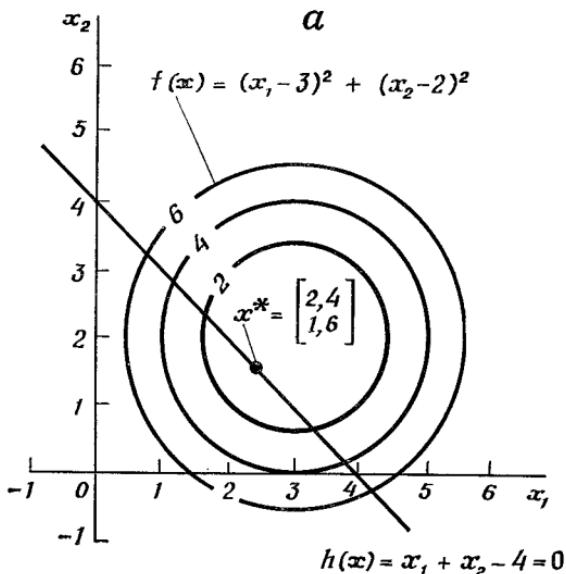
при ограничении

$$h(x) = x_1 + x_2 - 4 = 0.$$

Прибавим $h^2(x)$ к целевой функции $f(x)$, в результате чего получим новую целевую функцию

$$P(x) = (x_1 - 3)^2 + (x_2 - 2)^2 + (x_1 + x_2 - 4)^2,$$

значения которой будем считать свободными от каких-либо ограничений [$h^2(x)$ здесь выступает в роли «штрафа»]. В процессе минимизации $P(x)$ «штрафная добавка» к $f(x)$ способствует тому, чтобы вектор x в некоторой степени удовлетворял исходному ограничивающему условию ($h(x) = 0$). Совершенно очевидно, что, пока условие $h(x) = 0$ удовлетворяется (в пределах установленного допуска), значение штрафного члена пренебрежимо мало, и $P(x^*) \rightarrow f(x^*)$ при $x \rightarrow x^*$. Преимущество, которое мы получаем за счет перехода от задачи минимизации при наличии ограничений к задаче минимизации в отсутствие ограничений, состоит в том, что в последнем случае минимизация может осуществляться с помощью гораздо более простых (по сравнению с первым случаем) алгоритмов. При использовании методов штрафных функций получается максимальный оптимизирующий эффект за счет постоянного компромисса между необходимостью удовлетворения ограничений и процессом минимизации $f(x)$, который достигается путем присвоения надлежащих весов целевой функции и функциям, задающим ограничения. На фиг. 7.0.1 графически представлена преобразованная целевая функция, полученная путем добавления к $f(x)$ штрафной функции (с весовым коэффициентом, равным единице).



Ф и г. 7.0.1.

a — графическое представление исходной задачи; *б* — уровни штрафной функции.

Методы штрафных функций можно разделить на два класса: 1) параметрические методы и 2) непараметрические методы. Параметрические методы характеризуются наличием одного или нескольких надлежащим образом подобранных параметров, входящих в структуру штрафной функции (которая строится с помощью функций-ограничений) в качестве весовых коэффициентов. К числу ти-

личных параметрических методов относится метод последовательной безусловной минимизации (МПБМ), предложенный Фиакко и Мак-Кормиком [1], а также метод Зангвилла [2]. С другой стороны, в непараметрических методах, таких, как «метод центров» Гуарда [3] и предложенный Фиакко и Мак-Кормиком непараметрический МПБМ [4], целевая функция рассматривается как функция, задающая дополнительное искусственное ограничение, постепенно «уплотняемое» по мере получения информации в ходе решения задачи.

Параметрические методы распадаются на три категории: 1) методы внутренней точки; 2) методы внешней точки и 3) комбинированные методы. При использовании методов внутренней точки уровень целевой функции удерживается в отдалении от границы допустимой области (т. е. точка $\mathbf{x}^{(k)}$ постоянно находится внутри допустимой области) с помощью штрафной функции. Методы внешней точки (такие, как методы Зангвилла, Петрижковского [5]) и метод Фиакко и Мак-Кормика [6], наоборот, генерируют последовательность точек, которые выходят за пределы допустимой области, но дают в пределе допустимое решение. Штрафная функция не позволяет вектору \mathbf{x} слишком удаляться от границы допустимой области. В комбинированных методах (использование которых особенно необходимо в случае, когда ограничения имеют вид равенств) в ходе минимизации одни из ограничивающих условий удовлетворяются, а другие не удовлетворяются; однако все условия в пределах заданного допуска оказываются удовлетворенными при достижении исходного решения. МПБМ является комбинированным методом. Для полноты классификационной картины отметим, что в работе [6] перечислены и кратко описаны непараметрические методы внешней точки. Хороший исторический обзор большинства методов штрафных функций можно найти в монографии Фиакко и Мак-Кормика [6].

Итак, в основу методов штрафных функций в области нелинейного программирования положена идея преобразования общей нелинейной задачи (2.2.1) — (2.2.3) в последовательность задач без ограничений путем добавления к целевой функции одной или нескольких функций, задающих ограничения, с тем, чтобы ограничения, как таковые, в задаче оптимизации не фигурировали. Формально преобразование задачи, представленной соотношениями (2.2.1) — (2.2.3), в задачу минимизации без ограничений осуществляется путем перехода от (2.2.1) — (2.2.3) к задаче минимизации

$$P(\mathbf{x}^{(k)}, \rho^{(k)}) = f(\mathbf{x}^{(k)}) + \sum_{i=1}^m \rho_i^{(k)} H(h_i(\mathbf{x}^{(k)})) + \sum_{i=m+1}^p \rho_i^{(k)} G(g_i(\mathbf{x}^{(k)})). \quad (7.0.1)$$

Будем называть $P(\mathbf{x}^{(k)}, \rho^{(k)})$ обобщенной присоединенной функцией¹⁾ или же просто штрафной функцией. В формуле (7.0.1) $\rho_i^{(k)} \geq 0$

¹⁾ Иногда $P(\mathbf{x}^{(k)}, \rho^{(k)})$ называют расширенной функцией.— Прим. перев.

представляют собой весовые коэффициенты, $H(h_i(\mathbf{x}^{(k)}))$ и $G(g_i(\mathbf{x}^{(k)}))$ являются функционалами соответственно $h_i(\mathbf{x}^{(k)})$ и $g_i(\mathbf{x}^{(k)})$ [эти функционалы выбираются с учетом ряда конкретных требований (см. ниже)], а $k = 0, 1, \dots$ есть число завершенных этапов вычислительного оптимизационного процесса.

Типичными требованиями, из которых исходят при выборе функционала $G(g_i(\mathbf{x}^k))$, являются следующие:

1. $G(g_i(\mathbf{x})) \rightarrow +\infty$ при $g_i(\mathbf{x}) \rightarrow 0^+$, для чего необходимо, чтобы точка, определяемая вектором \mathbf{x} , всегда была внутренней, т. е. чтобы $\{\mathbf{x} | g_i(\mathbf{x}) > 0 \ (i = m+1, \dots, p)\}$.

2. $G_2(g_i(\mathbf{x})) \rightarrow 0$ при $g_i(\mathbf{x}) \rightarrow 0^-$. При таком выборе функционала G оперируют только с внешними точками: $\{\mathbf{x} | g_i(\mathbf{x}) < 0\}$.

3. $G_3(g_i(\mathbf{x})) > 0$ при $g_i(\mathbf{x}) < 0$ и $G_4(g_i(\mathbf{x})) = 0$ при $g_i(\mathbf{x}) \geq 0$. При таком выборе функционала G не заботятся о том, чтобы ограничивающие условия удовлетворялись на промежуточных этапах вычислительного процесса, хотя, естественно, требуют, чтобы эти условия удовлетворялись в точке, определяющей искомое решение.

В случае когда ограничения имеют вид равенств, как правило, требуют, чтобы $H(h_i(\mathbf{x})) \rightarrow 0$ при $h_i(\mathbf{x}) \rightarrow 0$. При этом обычно полагают $H(h_i(\mathbf{x})) = h_i^2(\mathbf{x})$.

При любом (из указанных выше) выборе функционалов $H(h_i(\mathbf{x}))$ и $G(g_i(\mathbf{x}))$ требуют, чтобы

$$\begin{aligned} \lim_{k \rightarrow \infty} \sum_{i=m+1}^p \rho_i^{(k)} G(g_i(\mathbf{x}^{(k)})) &= 0, \\ \lim_{k \rightarrow \infty} \sum_{i=1}^m \rho_i^{(k)} H(h_i(\mathbf{x}^{(k)})) &= 0, \\ \lim_{k \rightarrow \infty} |P(\mathbf{x}^{(k)}, \rho^{(k)}) - f(\mathbf{x}^{(k)})| &= 0. \end{aligned} \quad (7.0.2)$$

Другими словами, влияние входящих в $P(\mathbf{x}^{(k)}, \rho^{(k)})$ функций-ограничений на значение данной присоединенной функции постепенно по мере развития процесса оптимизационного поиска ослабевает, а в пределе полностью исчезает, так что последовательность промежуточных значений $P(\mathbf{x}^{(k)}, \rho^{(k)})$ сходится к тому же значению, что и соответствующая последовательность значений $f(\mathbf{x}^{(k)})$, и, следовательно, экстремум $P(\mathbf{x})$ совпадает с экстремумом $f(\mathbf{x})$.

В данной главе будет проведено краткое обсуждение некоторых вариантов методики штрафных функций, а затем дано несколько более детальное описание метода последовательной безусловной минимизации.

7.1. МЕТОДЫ ШТРАФНЫХ ФУНКЦИЙ СПЕЦИАЛЬНОЙ СТРУКТУРЫ

7.1.1. ИСПОЛЬЗОВАНИЕ МНОЖИТЕЛЕЙ ЛАГРАНЖА

Методы, основанные на использовании множителей Лагранжа, относятся к категории параметрических методов штрафных функций, поскольку для них характерно то, что функции-ограничения вводятся в структуру модифицированной целевой функции совместно с некоторым переменным параметром. Чтобы обобщить метод множителей Лагранжа, ограничения в виде неравенств следует преобразовать в ограничения, имеющие вид равенств, путем введения надлежащих ослабляющих переменных (на каждое ограничение неравенство по одной ослабляющей переменной). Задача нелинейного программирования в общей постановке [см. соотношения (2.2.1) — (2.2.3)] принимает при этом следующий вид:

минимизировать $f(\mathbf{x})$, $\mathbf{x} \in E^n$,

при ограничениях

$$\begin{aligned} h_i(\mathbf{x}) &= 0, \quad i = 1, \dots, m, \\ g_i(\mathbf{x}) - v_i^2 &= 0, \quad i = m+1, \dots, p. \end{aligned} \tag{7.1.1}$$

Если вычесть v_i^2 из $g_i(\mathbf{x})$ ($i = m+1, \dots, p$), то можно гарантировать, что ограничивающее условие, имеющее в исходной постановке задачи вид неравенства, действительно выполняется. Тогда можно определить обычным образом функцию Лагранжа

$$P(\mathbf{x}, \omega) = f(\mathbf{x}) + \sum_{i=1}^m \omega_i h_i(\mathbf{x}) + \sum_{i=m+1}^p \omega_i [g_i(\mathbf{x}) - v_i^2], \tag{7.1.2}$$

где ω_i ($i = 1, \dots, p$) — неотрицательные и не зависящие от \mathbf{x} весовые коэффициенты, которые можно отождествить с множителями Лагранжа. Для того чтобы \mathbf{x}^* было решением общей задачи нелинейного программирования (2.2.1) — (2.2.3), необходимо и достаточно [8], чтобы: 1) функция $f(\mathbf{x}^*)$ была выпуклой, 2) в окрестности \mathbf{x}^* ограничения задачи были выпуклы и 3) в точке \mathbf{x}^* удовлетворялась следующая система уравнений [определенная стационарное решение (7.1.2)]:

$$\begin{aligned} \frac{\partial P(\mathbf{x}^*)}{\partial x_j} &= 0 && \text{при } j = 1, \dots, n, \\ \frac{\partial P(\mathbf{x}^*)}{\partial \omega_i} &= 0 && \text{при } i = 1, \dots, p, \\ \frac{\partial P(\mathbf{x}^*)}{\partial v_i} &= 2\omega_i v_i = 0 \text{ при } i = m+1, \dots, p, \\ \omega_i &\geq 0, && i = 1, \dots, p. \end{aligned} \tag{7.1.3}$$

Короче говоря, условный минимум $f(\mathbf{x})$ имеет место в стационарной точке для $P(\mathbf{x}, \omega, \mathbf{v})$ и, в частности, в седловой точке $(\mathbf{x}, \omega, \mathbf{v})$ -пространства, так что задача с ограничениями превращается в задачу определения седловой точки в отсутствие ограничений.

Пример 7.1.1. Использование множителей Лагранжа

Рассмотрим в качестве примера следующую задачу:

минимизировать $y = x_1 x_2$, $\mathbf{x} \in E^n$,

при ограничении

$$g_1(\mathbf{x}): 25 - x_1^2 - x_2^2 \geq 0. \quad (\text{а})$$

Подлежащая минимизации присоединенная функция имеет вид

$$P(\mathbf{x}, \omega) = x_1 x_2 - \omega_1 (25 - x_1^2 - x_2^2 - v_1^2). \quad (\text{б})$$

Необходимые условия существования экстремума:

$$\begin{aligned} \frac{\partial P}{\partial x_1} &= x_2 + 2\omega_1 x_1 = 0, \\ \frac{\partial P}{\partial x_2} &= x_1 + 2\omega_1 x_2 = 0, \\ \frac{\partial P}{\partial \omega_1} &= 25 - x_1^2 - x_2^2 - v_1^2 = 0, \\ \frac{\partial P}{\partial v_1} &= 2\omega_1 v_1 = 0. \end{aligned} \quad (\text{в})$$

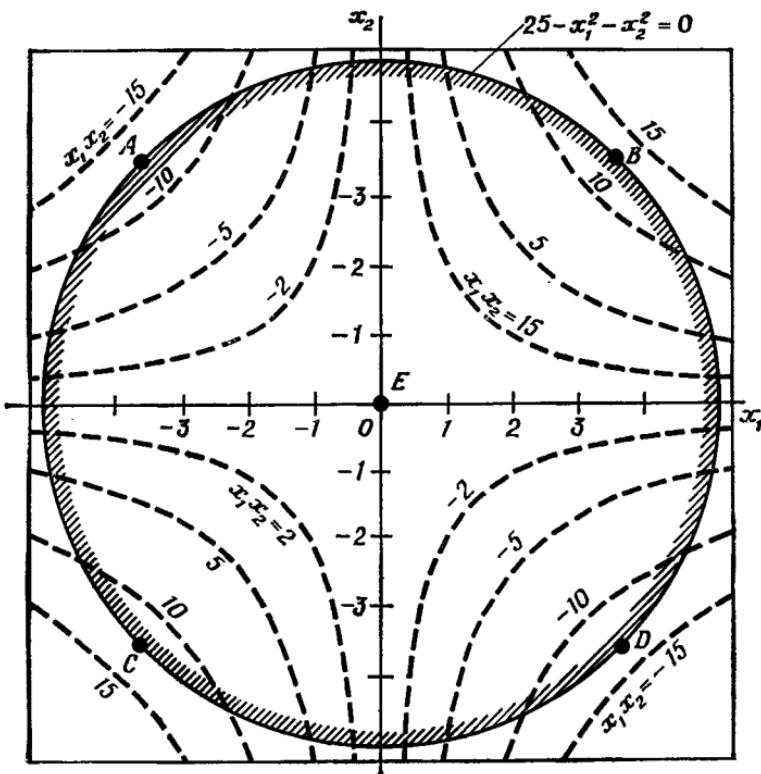
Совместные решения уравнений (в) при $\omega_1 = 0$ и $\omega_1 \neq 0$ приведены в табл. П.7.1.1.

Таблица П.7.1.1

Решение задачи (а) с помощью метода множителей Лагранжа

ω	x_1	x_2	Точка	v_1	$f(\mathbf{x})$	Примечание
0	0	0	E	5	0	Седловая точка
0,5	{+3,54 -3,54}	{-3,54 +3,54}	D A	0 0	-12,5 -12,5	Минимум Минимум
-0,5	{+3,54 -3,54}	{+3,54 -3,54}	B C	0 0	+12,5 +12,5	Максимум Максимум

Векторы \mathbf{x}^* являются стационарными решениями задачи (а). Заметим, что решения при $\omega_1 > 0$ являются минимумами, при $\omega_1 < 0$ — максимумами, а при $\omega_1 = 0$ — седловой точкой задачи (а). Функции, фигурирующие в задаче (а), представлены графически на



Фиг. П.7.1.1.

фиг. П.7.1.1. Уровни целевой функции (гиперболы) изображены пунктирными кривыми, а допустимая область заштрихована. Эта область ограничена окружностью $g_1(\mathbf{x}) = 0$. Точки A и D соответствуют двум (указанным в табл. П.7.1.1) минимумам, точки B и C — двум (также указанным в табл. П.7.1.1) максимумам, а E является седловой точкой целевой функции $f(\mathbf{x})$.

Метод множителей Лагранжа интенсивно изучался многими специалистами (см. список литературы, приведенный в конце главы). Этот метод может оказаться непригодным в случае невыпуклых задач, для решения которых успешно применяются другие методы штрафных функций. Метод множителей Лагранжа мало эффективен в случае задач нелинейного программирования большой размерности, поскольку совместное решение системы уравнений, аналогичной (в) в примере П.7.1.1, требует применения численных методов, и соответствующие вычислительные процедуры реализовать не легче, чем алгоритмы оптимизации. Более того, многократно повторяемые процедуры решения упомянутых выше уравнений должны быть

выделены из общей программы. Для данного метода и соответствующих вычислительных процедур было составлено несколько машинных программ, но ни одна из них не оказалась достаточно эффективной при решении задачи нелинейного программирования общего вида.

7.1.2. МЕТОД РОЗЕНБРОКА ДЛЯ РЕШЕНИЯ ЗАДАЧ ОПТИМИЗАЦИИ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ

Задача минимизации $f(\mathbf{x})$, $\mathbf{x} \in E^n$,

при ограничениях

$$g_i(\mathbf{x}) \geq 0, \quad i = 1, \dots, p, \quad (7.1.4)$$

была преобразована Розенброком таким образом, что оказался применимым для ее решения метод, описание которого дано в разд. 4.3. Решение задачи (7.1.4) эквивалентно определению минимума не связанный никакими ограничениями присоединенной функции

$$P(\mathbf{x}^{(k)}, \mathcal{U}) = f(\mathbf{x}^{(k)}) \prod_{i=1}^p \mathcal{U}_i(\mathbf{x}^{(k)}) g_i(\mathbf{x}^{(k)}), \quad (7.1.5)$$

где $\mathcal{U}_i = 0$ при $g_i(\mathbf{x}^{(k)}) < 0$ и $\mathcal{U}_i = 1$ при $g_i(\mathbf{x}^{(k)}) \geq 0$ ($i = 1, \dots, p$). Таким образом, присоединенная функция, определяемая соотношением (7.1.5), вне допустимой области обращается в нуль. Предполагается, что в каждой точке допустимой области целевая функция принимает отрицательное значение, т. е. $\{f(\mathbf{x}) < 0 | \mathbf{x} \in R\}$. Если бы в некоторых или даже во всех точках $\mathbf{x} \in R$ значения $f(\mathbf{x})$ оказались положительными, мы могли бы вычесть из $f(\mathbf{x})$ большое постоянное число K , так что имело бы место условие $\tilde{f}(\mathbf{x}) = [f(\mathbf{x}) - K] < 0$. Ясно, что характер задачи (минимизировать $\tilde{f}(\mathbf{x})$) при этом остался бы прежним. При решении сформулированной выше задачи численный оптимизационный поиск может начинаться не только с внутренней точки $\mathbf{x}^{(0)}$, но также и с точки, слегка выходящей за пределы допустимой области. Для минимизации $P(\mathbf{x})$ используется разработанная Розенброком процедура безусловной минимизации в том виде, в каком она представлена в разд. 4.3. Последовательность точек $\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \dots, \mathbf{x}^{(k)}$, получаемая при этом, является допустимой.

Чтобы увидеть взаимосвязь между функцией (7.1.5) и присоединенной функцией типа (7.0.1), требуется лишь прологарифмировать правую и левую части (7.1.5), изменив предварительно их знаки и перейдя от минимизации к максимизации. В результате придем к задаче максимизации

$$\ln [-P(\mathbf{x}^{(k)})] = \ln [-f(\mathbf{x}^{(k)})] + \sum_{i=1}^p (\ln \mathcal{U}_i) + \sum_{i=1}^p \ln g_i(\mathbf{x}^{(k)}). \quad (7.1.6)$$

[Умножение (7.1.5) на -1 необходимо по той причине, что $P(\mathbf{x}^{(k)})$ и $f(\mathbf{x}^{(k)})$ здесь являются отрицательными величинами.] Совершенно очевидно, что минимизация $f(\mathbf{x}) < 0$ эквивалентна максимизации $\ln[-f(\mathbf{x})] > 0$. Поскольку $\sum_{i=1}^p \ln \mathcal{U}_i$ на любой стадии вычислительного процесса является константой (хотя значение этой константы, возможно, и не полностью конкретизировано), можно положить $\rho_i = 1$ и считать, что $\ln g_i(\mathbf{x}^{(k)}) = G(g_i(\mathbf{x}^{(k)}))$ ($i = 1, \dots, p$), в результате чего соответствие между (7.1.5) и (7.0.1) оказывается установленным.

Вычислительная практика с применением машинных программ, построенных на основе метода Розенброка, показывает, что этот метод не позволяет оперировать ограничениями в виде равенств и не представляется удовлетворительным для решения задач, содержащих нелинейные ограничения в виде неравенств (см. гл. 9).

7.1.3. МЕТОД ВНУТРЕННЕЙ ТОЧКИ (ОБЪЕДИНЕНИЕ МЕТОДА БАРЬЕРНЫХ ПОВЕРХНОСТЕЙ И МЕТОДА ДЭВИДОНА)

Бокс, Дэвис и Свен [9] объединили метод Дэвидона с предложенным Кэрроллом [10] методом барьерных поверхностей (МБП), построив таким образом своеобразный параметрический метод внутренней точки, позволяющий эффективно решать задачи нелинейного программирования при наличии ограничений. Метод Кэрролла основывался на использовании штрафной функции вида

$$P(\mathbf{x}, r) = f(\mathbf{x}) + r \sum_{i=1}^p \frac{\omega_i}{g_i(\mathbf{x})}, \quad (7.1.7)$$

где r — параметр, значения которого убывают с каждым циклом, а ω_i ($i = 1, \dots, p$) — положительные весовые коэффициенты (веса). При приближении к границе изнутри, т. е. как только $g_i(\mathbf{x}) \rightarrow -\infty$, «штраф» становится большим. Соотношение (7.1.7) преобразует задачу нелинейного программирования при наличии ограничений в задачу без ограничений с гораздо более сложной (по сравнению с исходной задачей) структурой, но зато характеризующуюся наличием весьма сильных барьеров вдоль границы допустимой области.

Построив штрафную функцию и определив внутреннюю точку, приступаем к реализации процедуры минимизации $P(\mathbf{x}, r)$ при заданном начальном значении $r^{(0)}$. Тогда конечная для первого этапа вычислительной процедуры точка \mathbf{x} становится исходной (стартовой) точкой для минимизации функции P при уменьшенном значении r и т. д. Завершающий этап минимизации реализуется при очень малом значении r , так что результирующая точка \mathbf{x} с точностью до

установленного допуска может (если требуется) оказаться либо на одной из ограничивающих поверхностей, либо сразу на нескольких поверхностях, заданных ограничениями задачи. На каждом этапе безусловной минимизации Бокс, Дэвис и Свен использовали метод Дэвидона — Флетчера — Пауэлла. Доказательство корректности данного подхода приведено в подразд. 7.2.2.

На эффективность МБП существенно влияют и выбор начального значения r ($r^{(0)}$), и метод сокращения значений r в ходе минимизации, и выбор весовых коэффициентов. Если в $P(x, r)$ значение $r^{(0)}$ выбирается слишком малым, на начальной стадии процесса минимизации мы придем к минимуму функции $f(x)$, который вряд ли окажется вблизи условного минимума в точке x^* . Следовательно, движение вдоль траектории, которая должна привести к x^* , будет связано со значительными временными затратами. С другой стороны, если значение $r^{(0)}$ слишком велико, то на первых этапах вычислительного процесса текущая точка неизбежно окажется слишком далеко за пределами допустимых границ и поиск из-за необходимости возврата в пределы допустимой области также окажется весьма затяжным. Бокс, Дэвис и Свен [9] считают, что на начальной стадии оптимизационного процесса угол между $\nabla f(x^{(0)})$ и $\nabla P(x^{(0)}, r^{(0)})$ должен быть острым, а значение r должно увеличиваться или уменьшаться путем умножения или деления начального значения этого параметра на некоторое число до тех пор, пока угол между $\nabla f(x^{(k)})$ и $\nabla P(x^{(k)}, r^{(k)})$ не станет тупым. Кроме того, должно выполняться условие $r^{(0)} > 10^{-4}$. (На практике упомянутые выше авторы чаще всего просто полагают $r^{(0)} = 50$.) Они рекомендуют также при переходе от одного этапа вычислительного процесса к последующему умножать текущее значение r на 0,1 (или на число, лежащее в интервале от 0,02 до 0,1), что обеспечивает постепенное уменьшение этого параметра. Фиакко и Мак-Кормик (1963 г.) предъявляют к выбору r несколько иные требования; подробное обсуждение их рекомендаций содержится в подразд. 7.2.1.

Относительно ω_i рекомендации Бокса, Дэвиса и Свена сводятся к следующему: значения ω_i должны выбираться с таким расчетом, чтобы все слагаемые в (7.1.7) после надлежащего масштабирования оказывались по возможности величинами одного порядка. Данными авторами разработан эмпирический метод «подгонки» значений ω_i , основанный на том, что ω_i для часто нарушаемых ограничений слишком малы и, следовательно, в этих случаях значения ω_i следует увеличивать.

Дэвис [11] считает, что в случае, когда ограничения неравенства приводят к P -функциям вида

$$P_1 = f(x) + k \sum_{i=1}^m \frac{\omega_i}{g_i(x)}, \quad (7.1.8a)$$

$$P_2 = f(\mathbf{x}) + k \sum_{i=1}^m \frac{\omega_i}{g_i^2(\mathbf{x})}, \quad (7.1.86)$$

$$P_3 = f(\mathbf{x}) + k \sum_{i=1}^m \omega_i \log [-g_i(\mathbf{x})], \quad k > 0, \quad (7.1.8b)$$

ω_i следует полагать равными нулю до тех пор, пока не окажется нарушенным хотя бы одно из ограничений. Для активных ограничений ω_i полагаются равными единице. В качестве значения k Дэвис рекомендует брать значение множителя Лагранжа, ассоциированного с первым из нарушенных ограничений. Так, например, в приведенном выше выражении для P_1 (применительно к активным ограничениям $g_i(\mathbf{x}) \geq 0$)

$$k = \frac{g_i^2(\mathbf{x}) [\nabla^T f(\mathbf{x}) \nabla g_i(\mathbf{x})]}{\nabla^T g_i(\mathbf{x}) \nabla g_i(\mathbf{x})}, \quad (7.1.9)$$

где значения градиентов $f(\mathbf{x})$ и $g(\mathbf{x})$ вычислены в последней из текущих допустимых точек. Вместо того чтобы вычислять k_i для каждого дополнительного ограничения и пересматривать значения k_i на каждом этапе вычислительного процесса, представляется целесообразным (эта идея также принадлежит Дэвису) величину k , определенную с помощью (7.1.9), при переходе от одного этапа итерационного процесса к следующему уменьшать путем умножения на число, лежащее в интервале от 10^{-2} до 10^{-4} . Если задача содержит ограничения в виде равенства, то (по Дэвису) к P -функции (вида P_1 , P_2 или P_3) добавляется выражение, соответствующее второму слагаемому в правой части (7.0.1).

Фиакко и Мак-Кормик, используя МПБМ, полагали все ω_i равными единице, хотя и обсуждали в своей книге (см. список литературы в конце данной главы) другие варианты выбора весовых коэффициентов. Некоторые результаты, полученные при решении проблемных задач путем комбинированного применения МБП и метода Дэвидона, приведены в табл. 9.3.4.

7.1.4. МЕТОД ВЕЙСМАНА

Вейсман [12] разработал структуру комплексного метода (известного под названием «МИНИМАЛ»), позволяющего реализовать в рамках одной машинной программы сразу три алгоритма, а именно алгоритм прямого поиска, предложенный Хуком и Дживсом, алгоритм случайного поиска и алгоритм, основанный на использовании штрафной функции. Штрафная функция строится следующим образом:

$$P(\mathbf{x}, r) = f(\mathbf{x}) + \sum_{i=1}^p \delta_i r_i g_i^2(\mathbf{x}), \quad (7.1.10)$$

где $\delta_i = (1 - U_i)$ равняется нулю, если ограничивающие условия удовлетворяются, и единице, если ограничения оказываются нарушенными. При этом ограничения в виде равенств преобразуются в ограничения-неравенства, т. е. записываются в виде

$$g_i(\mathbf{x}) = |h_i(\mathbf{x})| - \varepsilon_i \leqslant 0,$$

где ε_i — величина допуска, ассоциированного с соответствующим исходным ограничением в виде равенства.

Минимизация P осуществляется методом Хука и Дживса для некоторой поэтапно возрастающей последовательности значений r_i . Поиск заканчивается либо когда оказываются удовлетворенными все ограничивающие условия, либо когда абсолютная разность между значениями $g_i(\mathbf{x})$ в начале и конце поиска оказывается меньше некоторого заранее установленного доверительного числа (например, 10^{-4}). Вейсман выбирал начальные значения r_i так, чтобы

$$r_i^{(0)} = \frac{0,02}{p^* g_i(\mathbf{x}^{(0)}) f(\mathbf{x}^{(0)})},$$

где p^* — число ограничений. В конце каждого этапа значения $r_i^{(k)}$ для каждого нарушенного (на k -м этапе) ограничения умножаются на 8; полученное в результате новое значение r используется на $(k+1)$ -м этапе.

В конце каждого этапа оптимизационного поиска методом Хука и Дживса реализуется, кроме того, случайный (рандомизированный) поиск. Пространство E^n делится на 10 гиперсфер с центром в точке \mathbf{x} , оказавшейся наиболее предпочтительной в результате поиска методом Хука и Дживса. В n случайно выбранных точках внутри каждой гиперсферы (число n задается заранее) находятся значения целевой функции, и, если попадается точка, дающая минимизирующую поправку менее 10% предыдущего значения $f(\mathbf{x})$ (т. е. значения $f(\mathbf{x})$, найденного в предыдущей точке \mathbf{x}), поиск методом Хука и Дживса повторяется. На следующем $(k+1)$ -м этапе стартовой (исходной) является точка, в которой значение $f(\mathbf{x})$ оказывается наименьшим среди всех значений, полученных на предыдущих k этапах оптимизационного поиска по любому из упомянутых выше алгоритмов.

7.1.5. ДРУГИЕ МЕТОДЫ ШТРАФНЫХ ФУНКЦИЙ

1. **Метод компенсирующих констант (МКК)**¹⁾. Фиакко и Мак-Корник в работе [13] описали метод, в структуре которого с неравенствами можно обращаться так же, как и с равенствами, за счет использования положительных ослабляющих констант (примерно в том же

¹⁾ Этот метод представляет собой одну из модификаций метода внешней точки.

смысле, в каком это делалось в связи с рассмотрением (7.1.1)). Этот метод предполагает использование штрафной функции

$$P(x, r^{(k)}, c) = f(x) + (r^{(k)})^{-1} \sum_{i=m+1}^p [g_i(x^{(k)}) - c_i]^2, \quad (7.1.11)$$

где $c_i \geq 0$. Штрафная функция такого вида достигает экстремума в результате последовательных перемещений в направлении искомой точки со стороны недопустимой области и имеет то преимущество, что поиск допустимой точки в начале вычислительного процесса можно не проводить. Относительно эффективности использования такого рода штрафной функции вряд ли можно сделать достаточно аргументированное заключение, так как «экспериментальных» данных по этому вопросу явно недостаточно.

2. Метод внутренней точки. Фиакко и Мак-Кормик [13] предложили другой вид штрафной функции в рамках алгоритма внутренней точки:

$$P(x^{(k)}, x^{(k-1)}) = [f(x^{(k-1)}) - f(x^{(k)})]^{-1} + \sum_i [g_i(x^{(k)})]^{-1}. \quad (7.1.12)$$

Ограничения в виде равенств могут быть включены в схему алгоритма по аналогии с (7.0.1). Функция (7.1.12), однако, исследована еще недостаточно.

3. Метод центров (разновидность метода внутренней точки). Гуард [14] описал алгоритм оптимизации, основанный на использовании следующей модификации целевой функции:

$$P(x^{(k)}, x^{(k-1)}) = [f(x^{(k-1)}) - f(x^{(k)})] \prod_i g_i(x^{(k)}). \quad (7.1.13)$$

Несмотря на то что функция (7.1.13) не является выпуклой, Гуарду удалось доказать, что получаемый на k -м этапе локальный оптимум является также глобальным оптимумом целевой функции. Следует, однако, заметить, что метод центров на практике недостаточно хорошо зарекомендовал себя, поэтому логика построения вычислительной процедуры и подробное описание метода центров на языке машинной программы здесь не приводятся (читатель может обратиться по этому вопросу к оригинальному источнику).

7.2. МЕТОД ПОСЛЕДОВАТЕЛЬНОЙ БЕЗУСЛОВНОЙ МИНИМИЗАЦИИ (КОМБИНИРОВАННЫЙ МЕТОД ШТРАФНЫХ ФУНКЦИЙ)

Алгоритм нелинейного программирования, называемый сокращенно МПБМ¹⁾ (метод последовательной безусловной минимизации), является обобщением метода барьерных поверхностей, который

¹⁾ Этот алгоритм разработан в одном из научно-исследовательских учреждений США (Research Analysis Corp., McLean, Va.).

был предложен Кэрролом [15]. Фиакко и Мак-Кормик [16] развили этот метод, доказали его эффективность и распространили его алгоритмическую структуру на случай, когда среди ограничений задачи имеют место ограничения в виде равенств. Соответствующие машинные программы имеются в Корпорации по научным исследованиям и разработкам (США), а также в лабораториях баллистических исследований (США) [17]. Алгоритм МПБМ разработан для решения задачи нелинейного программирования вида (2.2.1) — (2.2.3), в которой $f(\mathbf{x})$ и $g_i(\mathbf{x})$ ($i = m + 1, \dots, p$) могут быть нелинейными функциями независимых переменных, а $h_i(\mathbf{x})$ ($i = 1, \dots, m$) должны быть линейными функциями независимых переменных. При таких условиях гарантируется сходимость последовательности промежуточных решений к оптимальному решению задачи нелинейного программирования.

В основном МПБМ сводится к следующему: ищется решение некоторой последовательности задач без ограничений, причем в пределе находится минимум исходной задачи нелинейного программирования. В переложенном на язык машинных программ варианте МПБМ, относящемся к 1967 г., задача нелинейного программирования преобразуется в последовательность задач без ограничений путем построения P -функции следующего вида:

$$P(\mathbf{x}^{(k)}, r^{(k)}) = f(\mathbf{x}^{(k)}) + (r^{(k)})^{-1/2} \sum_{i=1}^m h_i^2(\mathbf{x}^{(k)}) + r^{(k)} \sum_{i=m+1}^p \frac{1}{g_i(\mathbf{x}^{(k)})}, \quad (7.2.1)$$

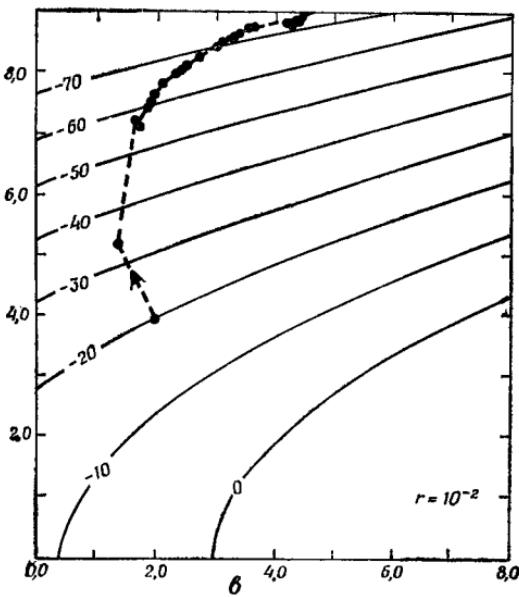
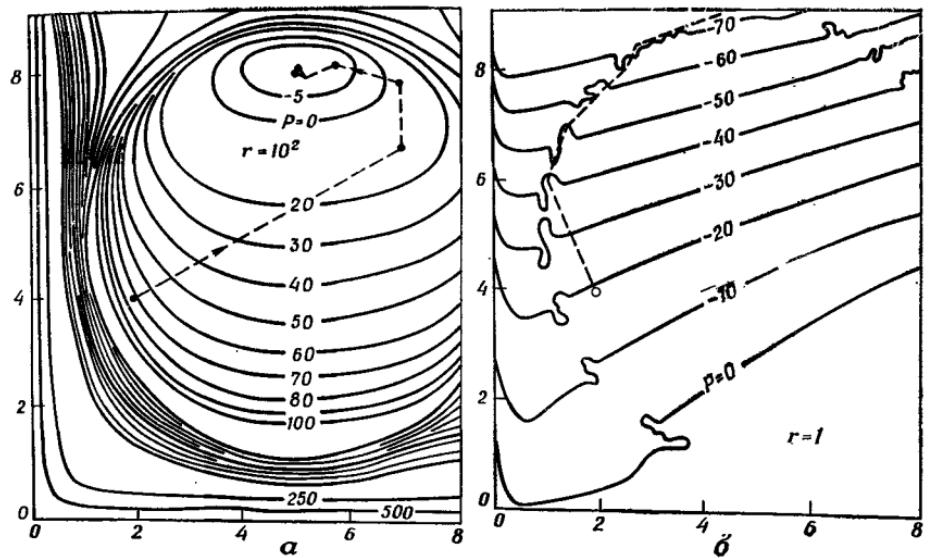
где значения весовых коэффициентов r положительны и образуют монотонно убывающую последовательность $\{r\} | r^{(0)} > r^{(1)} > \dots > > 0\}$. На фиг. 7.2.1 графически представлена P -функция, являющаяся частным случаем (7.2.1), а именно

$$P(\mathbf{x}) = (4x_1 - x_2^2 - 12) + r \left(\frac{1}{10x_1 - x_1^2 + 10x_2 - x_2^2 - 34} + \frac{1}{x_1} + \frac{1}{x_2} \right),$$

для трех различных значений r ; в этом случае целевую функцию исходной задачи нелинейного программирования и функции, задающие ограничения, легко выделить путем почлененного сопоставления данной P -функции с (7.2.1). Пунктирной кривой изображена траектория поиска минимума $P(\mathbf{x}, r)$.

Заметим, что вначале Фиакко и Мак-Кормик предпочитали задавать G -функционал ограничений-неравенств в виде своего рода «барьера», который в задачах минимизации добавлялся к целевой функции, а в задачах максимизации вычитался из $f(\mathbf{x})$. Этот функционал имел следующий вид:

$$G(g(\mathbf{x}^{(k)})) = \sum_{i=1}^p \frac{1}{g_i(\mathbf{x}^{(k)})}.$$



Ф и г. 7.2.1. Уровни P -функции.

Как только хотя бы одна из функций $g_i(x^{(k)}) \rightarrow 0$ внутри допустимой области, $G(g(x^{(k)})) \rightarrow \infty$; отсюда и происходит понятие «барьер». При уменьшении $r^{(k)}$ влияние барьера уменьшается и точку x удается приблизить к границе, ассоциированной с ограничениями-неравенствами.

Как уже отмечалось выше, можно выбрать G -функционал иным способом, например в виде

$$G(g(x^{(k)})) = \sum_{i=1}^p \min \{0, g_i(x^{(k)})\}^2$$

или

$$G(g(x^{(k)})) = - \sum_{i=1}^p \ln(g_i(x^{(k)})) = \sum_{i=1}^p \ln \frac{1}{g_i(x^{(k)})}.$$

Переложенный на язык машинных программ в 1970 г. вариант МПБМ основывался на использовании штрафной функции

$$P(x^{(k)}, r^{(k)}) = f(x) + \frac{1}{r^{(k)}} \sum_{i=1}^m h_i^2(x^{(k)}) - r^{(k)} \sum_{i=m+1}^p \ln g_i(x^{(k)}). \quad (7.2.1a)$$

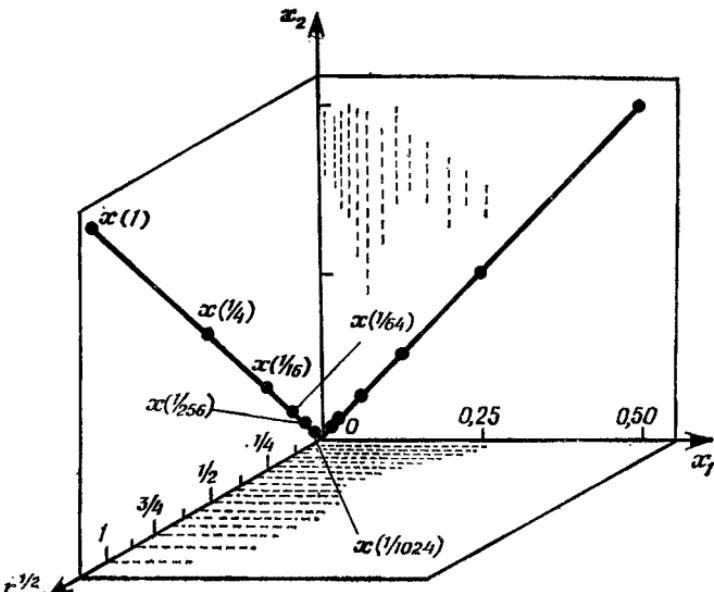
Как в этом варианте МПБМ, так и в варианте, описание которого приведено выше, функционал $H(h(x^{(k)}))$ представлял собой просто сумму квадратов $h_i(x)$, так что при $r^{(k)} \rightarrow 0$ ограничивающие условия в виде равенств удовлетворялись с непрерывно возрастающей точностью. Хотя каждое из ограничений в виде равенства можно в принципе разбить на два неравенства и затем иметь дело с неравенствами, на практике такой прием оказывался весьма неудовлетворительным — он в значительной степени замедляет оптимизационный поиск и порождает тенденцию к преждевременному прекращению вычислительного процесса.

Процедура минимизации функций (7.2.1) и (7.2.1a) начинается с внутренней (или граничной) точки, т. е. с точки $x^{(0)}$, в которой все ограничивающие условия в виде неравенств удовлетворены. После вычисления $r^{(0)}$ точка $x^{(1)}$ определяется путем минимизации $P(x, r^{(0)})$. Затем вычисляется $r^{(1)}$ и путем минимизации $P(x, r^{(1)})$ находится точка $x^{(2)}$ и т. д.

При выполнении описанной выше процедуры возникает ряд помех. Во-первых, ассоциированная с P -функцией матрица Гессе по мере приближения к экстремуму начинает вести себя все хуже; поэтому направления оптимизационного поиска могут в этих условиях стать ошибочными. Во-вторых, скорость сходимости зависит от начального выбора $r^{(0)}$ и от способа редуктирования данного параметра. Наконец, данные относительно «топологии» $f(x)$ и $P(x, r)$ в большинстве случаев при переходе от одного этапа вычислений к другому оказываются обесцененными (даже в тех случаях, когда алгоритм предполагает выполнение экстраполяции того или иного вида).

7.2.1. МЕТОДИКА ОПРЕДЕЛЕНИЯ ПАРАМЕТРА r

В первых вариантах МПБМ в качестве весового коэффициента перед $\sum_{i=1}^m h_i^2(x^{(k)})$ использовался множитель $(r^{(k)})^{-1/2}$, поскольку, как показал Фиакко (1966 г.) путем сравнения P -функции и $f(x)$ вбли-



Фиг. 7.2.2. Траектория минимума функции

$$P(x, r^{(k)}) = (4x_1 + x_2) + r^{(k)} \left(\frac{1}{x_1} + \frac{1}{x_2} \right).$$

зи экстремума при очень малых r , вектор x в окрестности искомого минимума $P(x, r)$ можно представить в виде некоторого полинома по $r^{1/2}$ (фиг. 7.2.2). В последующих вариантах МПБМ для суммы $\sum_{i=1}^m h_i^2(x^{(k)})$ использовался весовой коэффициент $(r^{(k)})^{-1}$. Весовой коэффициент для той компоненты P -функции, которая обусловлена наличием ограничений-неравенств, в обоих вариантах выбирался равным $r^{(k)}$.

Фиакко и Мак-Кормик предложили три способа выбора начального значения $r^{(0)}$:

а) $r^{(0)} = 1$. (7.2.2)

б) Выбирается такое значение $r^{(0)}$, которое минимизирует норму градиента P -функции [т. е. функции $P(x^{(0)}, r^{(0)})$ относительно r], что дает

$$r^{(0)} = \frac{-\nabla^T f(x^{(0)}) \nabla R(x^{(0)})}{\|\nabla R(x^{(0)})\|^2}, \quad (7.2.3)$$

где $\mathbf{x}^{(0)}$ — внутренняя допустимая точка, $R(\mathbf{x}^{(0)}) = \sum_{i=m+1}^p 1/g_i(\mathbf{x}^{(0)})$, а $\nabla R(\mathbf{x}^{(0)})$ — вектор-столбец, составленный из вычисленных в точке $\mathbf{x}^{(0)}$ первых частных производных $R(\mathbf{x})$ по \mathbf{x} . Мы видим, что при построении $R(\mathbf{x}^{(0)})$ учитываются только ограничения, которые имеют вид неравенств.

в) Значение $r^{(0)}$ выбирается так, чтобы минимизировать превышение функции $P(\mathbf{x}, r^{(0)})$ относительно его минимального значения (этую величину называют метрически взвешенным градиентом P -функции); при этом $r^{(0)}$ задается следующим соотношением¹⁾:

$$r^{(0)} = \left\{ \frac{\nabla^T f(\mathbf{x}^{(0)}) [\nabla^2 R(\mathbf{x}^{(0)})]^{-1} \nabla f(\mathbf{x}^{(0)})}{\nabla^T R(\mathbf{x}^{(0)}) [\nabla^2 R(\mathbf{x}^{(0)})]^{-1} \nabla R(\mathbf{x}^{(0)})} \right\}^{1/2}, \quad (7.2.4)$$

где $\nabla^2 R(\mathbf{x}^{(0)})$ — матрица Гессе для $R(\mathbf{x}) = \sum_{i=1}^p 1/g_i(\mathbf{x})$, вычисленная в точке $\mathbf{x}^{(0)}$. Соотношение (7.2.4) может быть применено лишь в том случае, когда $\nabla R(\mathbf{x}^{(0)}) \neq 0$, так что $r^{(0)} \neq 0$. При слишком больших значениях $r^{(0)}$ минимизационный процесс начинается во внутренних точках, лежащих слишком далеко от границы допустимой области, тогда как при слишком малых $r^{(0)}$ начальное решение оказывается чрезмерно близким к границе, задаваемой ограничениями задачи. В обоих случаях, для того чтобы оказаться в нужном месте допустимой области, приходится тратить дополнительное машинное время.

Из указанных выше трех способов задания $r^{(0)}$ наиболее удобным с практической точки зрения оказывается первый (в большинстве машинных программ, составленных на основе МПБМ, полагают $r^{(0)} = 1$); именно этот вариант рассматривается в гл. 9. Однако, чтобы достичь при решении задачи наибольшей точности, предпочтение следует отдать выражению (7.2.4), хотя расход машинного времени из-за слишком больших начальных значений $r^{(0)}$ будет при этом большим. Соотношение (7.2.3) применяется в тех случаях, когда отсутствуют данные о вторых частных производных функции R по \mathbf{x} . Если в результате вычислений по формуле (7.2.3) или по формуле (7.2.4) окажется, что $r^{(0)} \leq 0$, оптимизационный поиск начинается в точке $\mathbf{x}^{(0)}$ одним из методов, описанных в гл. 3. При этом $r^{(0)}$ вычисляется в каждой новой точке $\mathbf{x}^{(k)}$, пока не будет получено положительное значение $r^{(0)}$ (в этом случае оптимизационный поиск продолжается с помощью МПБМ) или пока не будет достигнут безусловный минимум целевой функции задачи (2.2.1) — (2.2.3). В последнем случае \mathbf{x}^* есть внутренняя точка, и поиск, таким обра-

¹⁾ Этим соотношением $r^{(0)}$ задается точно, если P -функция является квадратичной.

зом, заканчивается. Фиакко и Мак-Кормиком экспериментально было показано, что после определения $r^{(0)}$ одним из указанных выше способов эффективность алгоритма не изменится в значительной степени, если последовательность $r^{(1)}, r^{(2)}, \dots, r^{(k)}$ индуцировать простым соотношением $r^{(k)} = r^{(k-1)}/c$, где $c > 1$ есть константа (обычно полагают $c = 4$). Такой простой способ определения каждого последующего значения $r^{(k)}$ в ходе итерационного процесса существенно упрощает процедуру экстраполяции, описание которой дано в подразд. 7.2.3.

7.2.2. УСЛОВИЯ СХОДИМОСТИ МПБМ

Можно доказать, что при определенных условиях последовательность безусловных минимумов $P(\mathbf{x}, r^{(k)})$ будет стремиться к решению задачи нелинейного программирования (2.2.1) — (2.2.3) по мере приближения к нулю значений $r^{(k)}$. Существенным является условие выпуклости P -функции. Условия сходимости (ср. их с условиями, перечисленными на стр. 242—243) выглядят следующим образом:

Условие 1. Объединение множества, содержащего все точки \mathbf{x} , удовлетворяющие ограничивающим условиям в виде равенств $h_i(\mathbf{x}) = 0$ ($i = 1, \dots, m$), и множества S^* , содержащего все точки \mathbf{x} , удовлетворяющие ограничивающим условиям $g_i(\mathbf{x}) \geq 0$ ($i = m+1, \dots, p$), должно быть непустым. Другими словами, задача должна иметь допустимую область.

Условие 2. Функции $f(\mathbf{x}), h_1(\mathbf{x}), \dots, h_m(\mathbf{x}), g_{m+1}(\mathbf{x}), \dots, g_p(\mathbf{x})$ должны быть дважды непрерывно дифференцируемыми, если возникает необходимость применять метод, основанный на использовании вторых производных (см. подразд. 7.2.3, шаг 3).

Условие 3. Для любого конечного q и любого $r^{(k)} > 0$ множество точек \mathbf{x} , удовлетворяющих неравенству

$$f(\mathbf{x}) + (r^{(k)})^{-1/2} \sum_{i=1}^m h_i^2(\mathbf{x}) \leq q$$

и принадлежащих множеству S^* , должно быть ограниченным.

Условие 4. Целевая функция $f(\mathbf{x})$ должна быть выпуклой, а функции $h_i(\mathbf{x})$ ($i = 1, \dots, m$) должны быть линейными (точнее, сумма $\sum h_i^2(\mathbf{x})$ должна быть выпуклой).

Условие 5. Функции $g_i(\mathbf{x})$ ($i = m+1, \dots, p$) должны быть вогнутыми.

Условие 6. Матрица Гессе для P -функции относительно \mathbf{x} (т. е. матрица, составленная из вторых частных производных P -функции по независимым переменным) не должна обращаться в нуль ни для одной из точек, принадлежащих множеству S^* .

Выполнение условий 1—6 полностью гарантирует сходимость к решению задачи нелинейного программирования, представленной соотношениями (2.2.1) — (2.2.3); однако сходимость может иметь место даже и тогда, когда эти условия полностью не выполняются.

Опираясь на условия 1—6, Фиакко и Мак-Кормик доказали следующую теорему сходимости:

Теорема 1 (сходимость для исходной задачи)

Если задача нелинейного программирования удовлетворяет условиям 1—6, то:

a) каждая функция $P(x, r^{(k)})$ имеет минимум в некоторой точке $x(r^{(k)})$ множества S^* и

b) $\lim_{r^{(k)} \rightarrow 0} P(x(r^{(k)}), r^{(k)}) = \min f(x) = f(x^*)$, т. е. последовательность

безусловных минимумов $P(x(r^{(k)}), r^{(k)})$ стремится к решению $f(x^*)$ исходной задачи нелинейного программирования при $k \rightarrow \infty$.

Кроме того, при выполнении условий 1—6 P -функция есть выпуклая функция в S^* .

С задачей нелинейного программирования (2.2.1) — (2.2.3) ассоциированы две двойственные задачи: одна из них относится к случаю, когда функции $h_i(x)$ ($i = 1, \dots, m$) нелинейны, а другая — к случаю, когда эти функции линейны. Если функции $h_i(x)$ нелинейны, то нелинейную задачу (2.2.1) — (2.2.3) необходимо (путем записи каждого из равенств в виде двух неравенств) переформулировать следующим образом:

Задача A:

минимизировать $f(x)$, $x \in E^n$,

при ограничениях

$$\begin{aligned} h_i^2(x) &\geq 0, & i &= 1, \dots, m, \\ g_i(x) &\geq 0, & i &= m+1, \dots, p. \end{aligned} \tag{7.2.5}$$

Двойственной по отношению к задаче A является задача A':

$$\text{максимизировать } E(x, u, w) = f(x) - \sum_{i=m+1}^m u_i g_i(x) + \sum_{i=1}^m w_i h_i^2(x)$$

при ограничениях

$$\begin{aligned} \nabla_x E(x, u, w) &= 0, \\ u_i &\geq 0, & i &= m+1, \dots, p, \\ w_i &\geq 0, & i &= 1, \dots, m, \end{aligned} \tag{7.2.6}$$

где $\nabla_x E(x, u, w)$ есть вектор, составляющими которого являются первые частные производные $E(x, u, w)$ по независимым переменным. Следует иметь в виду следующее: когда функции $h_i(x)$ нелинейны, нет гарантии того, что решение задачи A' совпадает с $f(x^*)$, т. е. с решением задачи A. (Опыт говорит о том, что это весьма маловероятно.)

Если функции $h_i(x)$ линейны, задача А может быть представлена в эквивалентной форме В:

Задача В:

минимизировать $f(x)$, $x \in E^n$,

при ограничениях

$$\begin{aligned} h_i(x) &\geq 0, & i = 1, \dots, m, \\ -h_i(x) &\geq 0, & i = 1, \dots, m, \\ g_i(x) &\geq 0, & i = m+1, \dots, p. \end{aligned} \quad (7.2.7)$$

Задача, двойственная по отношению к задаче В, формулируется следующим образом:

Задача В':

$$\begin{aligned} \text{максимизировать } E(x, u, w, w') &= f(x) - \sum_{i=m+1}^p u_i g_i(x) + \\ &+ \sum_{i=1}^m w_i h_i(x) - \sum_{i=1}^m w'_i h'_i(x) \end{aligned}$$

при ограничениях

$$\begin{aligned} \nabla_x E(x, u, w, w') &= 0, \\ u_i &\geq 0, & i = m+1, \dots, p, \\ w_i &\geq 0, & i = 1, \dots, m, \\ w'_i &\geq 0, & i = 1, \dots, m. \end{aligned} \quad (7.2.8)$$

Двойственная задача В' имеет решение в некоторой точке (x^*, u^*, w^*, w'^*) , где $E(x^*, u^*, w^*, w'^*) = f(x^*)$, т. е. совпадает с решением задачи В. Фиакко и Мак-Кормик доказали следующую теорему сходимости применительно к условиям двойственной задачи:

Теорема 2 (сходимость для двойственной задачи)

При выполнении условий (1—6) МПБМ генерирует точки $\{x(r^{(k)}), u(r^{(k)}), w(r^{(k)}), w'(r^{(k)})\}$, являющиеся допустимыми и обладающими следующим свойством:

$$\lim_{k \rightarrow \infty} E(x(r^{(k)}), u(r^{(k)}), w(r^{(k)}), w'(r^{(k)})) = f(x^*).$$

Таким образом, наряду с решением задачи А (или В), которую мы называем исходной, алгоритм, основанный на МПБМ, генерирует также последовательность точек, приводящую к решению задачи А' (или В'), двойственной по отношению к А (или В). Поскольку $f(x^*)$ есть максимальное значение $E(x, u, w, w')$, то, как только условия 1—6 оказываются выполненными, имеет место следующее неравенство:

$$E(x(r^{(k)}), u(r^{(k)}), w(r^{(k)}), w'(r^{(k)})) \leq f(x^*) \leq P(x(r^{(k)})). \quad (7.2.9)$$

Неравенство (7.2.9) можно использовать [если задача удовлетворяет условиям (1—6)] для определения момента, когда в процессе реализации вычислительного процесса на ЭВМ сходимость последовательности промежуточных решений к решению исходной задачи нелинейного программирования оказывается практически достигнутой.

7.2.3. ВЫЧИСЛИТЕЛЬНАЯ ПРОЦЕДУРА

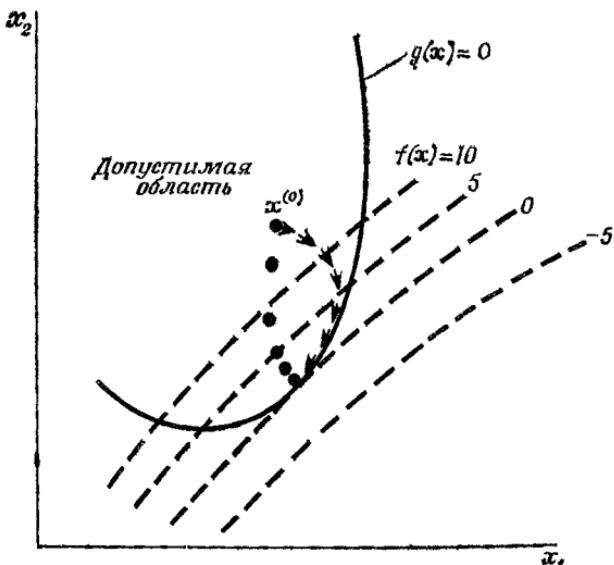
В общем случае вычислительный алгоритм реализует следующие шаги (фиг. 7.2.3)¹⁾:

Шаг 1. Пользователь выбирает точку $\mathbf{x}^{(0)}$, принадлежащую множеству S^* (т. е. такую точку $\mathbf{x}^{(0)}$, для которой $g_i(\mathbf{x}^{(0)}) > 0$, $i = m + 1, \dots, p$). Другими словами, начальная точка должна быть внутренней точкой (в соответствии с определением, приведенным в гл. 2). Поскольку P -функция может иметь несколько минимумов, то, начиная поиск с точки, не являющейся внутренней, можно прийти к минимуму, не являющемуся допустимым. Если требуемая внутренняя точка неизвестна, ее можно найти повторным применением метода МПБМ. В варианте МПБМ, относящемся к 1967 г., фиксировалось первое нарушенное ограничение-неравенство, причем функция, соответствующая этому ограничению, рассматривалась как временная целевая функция и с помощью МПБМ минимизировалось отрицательное значение этой функции при учете совокупности удовлетворенных ограничивающих условий в виде неравенств. Как только все эти ограничивающие условия оказываются удовлетворенными, упомянутое выше выделенное ограничение переводится в разряд обычных ограничений и процедура повторяется относительно другого нарушенного ограничения, т. е. фиксируется новая временная целевая функция. В варианте МПБМ, относящемся к 1970 г., для определения внутренней точки осуществлялась минимизация взятой с отрицательным знаком суммы всех $g_i(\mathbf{x})$, для которых соответствующие ограничения оказывались нарушенными. В обоих из указанных выше вариантах МПБМ ограничивающие условия в виде равенств удовлетворяются лишь на заключительном этапе решения исходной задачи.

Следует отметить, что если оказываются нарушенными сразу несколько ограничений с высокой «степенью» нелинейности, то определение допустимой начальной (стартовой) точки может потребовать значительных временных затрат. Поэтому всякий раз, когда допустимую начальную точку удается «увидеть» без дополнительных вычислений, следует именно с нее и начинать итерационный процесс.

¹⁾ Фиг 7.2.3 позволяет провести сравнение траектории последовательности минимумов $P(\mathbf{x}, r)$, найденных с помощью МПБМ, с траекторией поиска, полученной проективным методом.

Шаг 2. После определения стартовой точки в соответствии с алгоритмом находится положение минимума P -функции [определенной формулами (7.2.1) или (7.2.1а)] для текущего значения $r^{(k)}$. Вероятно, большая часть машинного времени потребуется на отыскание с помощью МПБМ последовательности внутренних точек, минимизирующих P -функцию для каждого значения $r^{(k)}$. Направление



Ф и г. 7.2.3. Сравнение траектории оптимизационного поиска, полученной методом штрафных функций, с траекторией минимизации, найденной проективным методом.

$g(x) \geq 0$ — нелинейное ограничение в виде неравенства. Стрелками изображена траектория, полученная проективным методом; точками изображена траектория, полученная методом штрафных функций.

оптимизационного поиска находится с помощью описанного в разд. 3.3 метода Ньютона путем предварительного умножения градиента функции $P(x, r^{(k)})$ на матрицу, являющуюся обратной по отношению к матрице вторых частных производных $P(x, r^{(k)})$ по независимым переменным, или (как альтернативный вариант) с помощью метода Брайдена (см. подразд. 3.4.1), аппроксимирующего $[\nabla^2 P(x, r^{(k)})]^{-1}$:

$$\mathbf{s}^{(k)} = -[\nabla^2 P(x^{(k)}, r^{(k)})]^{-1} \nabla P(x^{(k)}, r^{(k)}) \quad (7.2.10)$$

или

$$\mathbf{s}^{(k)} = -\eta_p \nabla P(x^{(k)}, r^{(k)}). \quad (7.2.10a)$$

Длина шага, как обычно, определяется путем минимизации модифицированной P -функции. Как только направление поиска оказывается установленным, в рамках МПБМ производится поиск методом Фибоначчи с тем, чтобы определить положение минимума

$P(x, r^{(k)})$ на расстоянии $\Delta x^{(k)}$ от $x^{(k)}$ в направлении поиска [тогда как в структуре метода НЛП предполагается сочетание поиска и экстраполяции (подгонки)] для того, чтобы перейти к минимизации методом линейной аппроксимации.

Применение метода Ньютона сопряжено с определенной «опасностью», поскольку поведение матрицы, обратной к матрице Гессе для P -функции, может оказаться весьма неудовлетворительным. Мюррей [18] показал, что даже в случае задач с хорошо выбранными масштабами для переменных, матрица Гессе для $P(x, r^{(k)})$ ведет себя неудовлетворительно. Новая информация, полученная на том или ином этапе вычислительного процесса, может оказаться недостаточной для того, чтобы скомпенсировать увеличение погрешности в информации, полученной на предыдущем этапе.

Флетчер и Мак-Канн [19] рассмотрели отношение наибольшего из собственных значений матрицы Гессе для P -функции к наименьшему собственному значению этой матрицы в качестве некоторого критерия; ими было установлено, что применение метода Ньютона приводит к неудовлетворительному поведению матрицы Гессе, тогда как метод Дэвидона — Флетчера — Паузелла приводит хотя и к приближенной, но удовлетворительно ведущей себя матрице, обратной по отношению к матрице Гессе для P -функции. Флетчером и Мак-Канном предложена определенная стратегия ускорения процесса минимизации $f(x)$ путем использования наряду с r собственных значений $P(x, r)$ для выявления на каждом этапе вычислительного процесса (когда значение r уменьшается) наиболее удовлетворительной структуры матриц, задающих направление оптимизационного поиска. Как показало решение пяти пробных (тестовых) задач, включая задачи 10, 11 и 18, приведенные в приложении А, экстраполяция позволяет сократить (по сравнению с методом Ньютона) количество вычислительных операций (определение направления поиска + линейный поиск) на 15—50%.

Из-за необходимости приведения (в рамках МПБМ) матрицы Гессе для P -функции к регулярному виду (если эта матрица не является положительно определенной) берется направление поиска, совпадающего с отрицательным градиентом, т. е. матрица, обратная к матрице Гессе для P -функции, полагается равной единичной матрице.

Хотя Фиакко и Мак-Кормик утверждают, что практическое применение методов «первых производных» и метод Дэвидона при минимизации P -функции не было в такой же степени успешным, как использование метода Ньютона, Уортман [17] считает, что минимизация методом Дэвидона — Флетчера — Паузелла требует наименьших затрат времени. В табл. 7.2.1 приведены некоторые результаты, полученные Уортманом для задачи 18 из приложения А с помощью различных минимизационных подпрограмм. Метод Ньютона имел при этом структуру, описание которой дано в гл. 3, если не считать,

что, когда $\nabla^2 P$ не был положительно определенным, $\nabla^2 P$ заменялось на единичную матрицу I , т. е. направление поиска осуществлялось вдоль отрицательного градиента. Небольшие отличия имели место между МПБМ и методом НЛП в отношении условий завершения вычислительных операций; однако в целом эти методы равнозначны.

Таблица 7.2.1

Сравнение различных методов поиска при минимизации P -функции задачи 18, приведенной в приложении А

	МПБМ 1)	Метод Ньютона	Метод Дэвидона — Флетчера — Пауэлла	Метод Пауэлла	Метод Хука и Дживса
$f(x^*)$	32,3519	32,3488	32,3488	32,3488 455	32,3526 1363
Полное число этапов					
Одномерный линейный поиск (число шагов)	92	88	272	7164	
Количество вычислений значений целевой функции		292	1086	27 987	37 485
Количество вычислений значений P -функции		487	1331	27 458	34 054
Время, мин		1,14	0,73	4,71	4,72

1) Фнакко и Мак-Кормик.

В методе Хука и Дживса вычислительный этап состоит из структурно-аналитического и по необходимости моделирующего поиска. В методе Пауэлла на каждом этапе находится линеаризованный минимум в n независимых направлениях, и не исключается возможность того, что на каком-либо из этапов реализуется поиск в «составном» направлении (см. гл. 4). За исключением случаев применения методов Хука и Дживса, составляющие вектора x определялись с точностью до $10^{-5} - 10^{-4}$ от их истинных значений (см. в этой связи приложение А). Во всех случаях все ограничивающие условия в виде неравенств были полностью удовлетворены. Как метод Дэвидона — Флетчера — Пауэлла, так и метод Ньютона позволили найти минимум целевой функции быстро и с большой степенью точности.

В одном из своих исследований Табак [20] приводит данные относительно временных затрат при минимизации P -функции на ИБМ 360/91. Эти данные содержатся в табл. 7.2.2. Они относятся к следующей задаче:

минимизировать $f(\mathbf{x}) = cx_1 + x_3 + x_5$
при ограничениях

$$(2x_3^2 - 1)x_4^2 + (2x_5^2 - 1)x_6^2 \geq 0,$$

$$x_1[4x_4^2x_6^2(4x_3^2x_5^2 - 2x_3^2 - 2x_5^2 + 1) + x_4^4 + x_6^4] + 1 - 2x_2(x_3x_4 + x_5x_6) \geq 0,$$

$$2x_1x_4^2x_6^2[(2x_3^2 - 1)x_6^2 + (2x_5^2 - 1)x_4^2] + 2x_2x_4x_6(x_3x_6 + x_4x_5) - x_4^2 - x_6^2 - 4x_3x_4x_5x_6 \geq 0,$$

$$x_{3,\min} \leq x_3 \leq x_{3,\max},$$

$$x_{5,\min} \leq x_5 \leq x_{5,\max},$$

$$x_4 \leq x_{4,\max},$$

$$x_6 \leq x_{6,\max}.$$

Тестовые решения задач 1, 2, 4, 5 и 11 из приложения А (соответствующие результаты приведены в табл. 7.2.3) показывают, что

Таблица 7.2.2
Минимизация P -функции различными методами

Метод	Машинное время, с	Метод	Машинное время, с
Ньютона	1,40	Бройдена	6,63
Проекции приведенного градиента	2,10	Флетчера — Ривса	7,66
Проективный Ньютона — Пирсона	2,93	Пирсона 2	32,24
Пирсона 3	6,13	Наискорейшего спуска	57,53

перечисленные методы минимизации с применением машинной программы «МПБМ-1970» в грубом приближении эквивалентны, если используется аналитическая запись частных производных, но оказываются малоэффективными в условиях поиска без использования производных в их аналитической записи (т. е. в условиях «чистого» поиска).

Завершение поиска минимального значения P -функции в МПБМ определяется на каждом этапе вычислительного процесса одним из трех критериев сходимости; выбор критерия осуществляется пользователем. Работа программы заканчивается, если выполняется одно из следующих условий:

a) $|\nabla^T P(\mathbf{x}^{(k)}, \mathbf{r}^{(k)}) [\nabla^2 P(\mathbf{x}^{(k)}, \mathbf{r}^{(k)})]^{-1} \nabla P(\mathbf{x}^{(k)}, \mathbf{r}^{(k)})| < \epsilon,$

Другими словами [см. соотношение (7.2.10)], останов имеет место, если

$$\frac{\partial P(x, r^{(k)})}{\partial x_i} (\Delta x_i) < \varepsilon, \quad i = 1, \dots, n.$$

б) $|\nabla^T P(x^{(k)}, r^{(k)}) [\nabla^2 P(x^{(k)}, r^{(k)})]^{-1} \nabla P(x^{(k)}, r^{(k)})| < \frac{P(x^{(k-1)}) - P(x^{(k)})}{5}.$

Данный критерий останавливает минимизационный поиск, если

$$\frac{\partial P(x^{(k)}, r^{(k)})}{\partial x_i} (\Delta x_i) < \frac{P(x^{(k-1)}, r^{(k-1)}) - P(x^{(k)}, r^{(k)})}{5}, \quad i = 1, \dots, n.$$

в) $\left| \frac{\partial P(x^{(k)}, r^{(k)})}{\partial x_i} \right| < \varepsilon, \quad i = 1, \dots, n,$

где вертикальные линии показывают, что берется абсолютное значение величины $\partial P / \partial x_i$. Критерий «а» использовался исключительно

Таблица 7.2.3

Временные затраты¹⁾ на решение некоторых задач нелинейного программирования с помощью МПБМ (1970 г.)

Номер задачи	1		2		4		5		11
	1	2	1	2	1	2	1	2	1
Методы минимизации Ньютона с производными в аналитической записи									
Бройдена	0,53	0,67	0,76	0,68	2,05	2,27	0,99	1,33	0,98
Только вычисление значений функций, Фиакко и Мак-Корник	0,56	1,01	0,85	0,88	4,39	4,80	1,01	1,30	1,65
	0,94	1,23			Решение найти не удалось				3,40

1) Основное время машинной обработки данных в секундах на CDC 6600.

2) См. подразд. 7.2.3 (шаг 5).

В связи с решением задач, рассматриваемых в гл. 9 (при $\varepsilon = 10^{-7}$). В табл. 7.2.4 приведены некоторые данные применительно к условиям задачи 1 из приложения А; эти данные позволяют провести сравнение эффективности сформулированных выше критериев останова.

Шаг 3. В МПБМ вектор $x^{(k+1)}$ задается соотношением

$$x^{(k+1)} = x^{(k)} + \lambda^{(k)} s^{(k)},$$

где $s^{(k)}$ определяется либо формулой (7.2.10), либо (7.2.10a). Чтобы избежать манипуляции с точками, лежащими вне допустимой области, необходимо располагать некоторым методом выявления

ситуаций, когда оказывается нарушенным то или иное ограничение в виде неравенства.

Шаг 4. Рассмотрим теперь ускорение процесса поиска решения путем экстраполяции. Без акселерационного шага МПБМ обеспечивает сходимость к условному экстремуму слишком медленно

Таблица 7.2.4

Результаты применения МПБМ (1970 г.) при различных критериях останова в связи с решением задачи 1 из приложения А¹⁾

Останов при минимизации <i>P</i> -функции ²⁾	Останов по завершении всего минимизационного процесса ³⁾		
	Критерий 1	Критерий 2	Критерий 3
Метод Ньютона			
Критерий а	0,529 (32)	0,670 (38)	0,606 (38)
Критерий б	0,703 (106)	0,832 (129)	0,815 (129)
Критерий в	0,604 (93)	0,884 (127)	0,785 (127)
Метод Бройдена	Критерий 1	Критерий 2	Критерий 3
Критерий а	0,563 (49)	1,013 (64)	0,967 (64)
Критерий б	0,729 (49)	1,190 (61)	0,955 (61)
Критерий в	0,597 (49)	0,974 (64)	1,112 (64)
Только вычисление значений функции	Критерий 1	Критерий 2	Критерий 3
Критерий а	0,944 (112)	1,234 (146)	1,231 (164)
Критерий б	1,316 (187)	1,679 (223)	1,462 (214)
Критерий в	0,703 (58)	1,002 (91)	1,051 (91)

1) Чистое машинное время в секундах; в скобках указано число вычислительных этапов.

2) См. подразд. 7.2.3 (шаг 2).

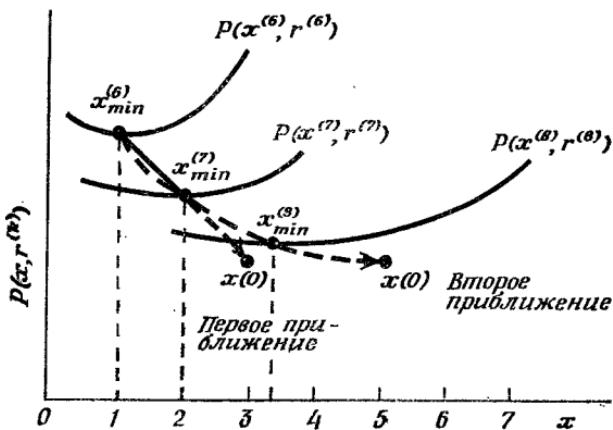
3) См. подразд. 7.2.3 (шаг 5).

(так как вблизи границы процесс минимизации проходит медленно). Именно для преодоления этой трудности используется прием последовательного уменьшения весового коэффициента r с тем, чтобы получить последовательность минимумов $P(\mathbf{x}, r^{(k)})$ и соответствующих значений $\mathbf{x}^{(k)}$. После этого по трем точкам $\mathbf{x}^{(k-1)}$, $\mathbf{x}^{(k)}$ и $\mathbf{x}^{(k+1)}$ можно найти приближенный экстремум. Данная процедура оказывается эффективной при решении многих задач, но становится явно неудовлетворительной при сильном «сгущении» текущих точек.

После того как удается отыскать несколько минимумов функции $P(\mathbf{x}^{(k)}, r^{(k)})$, МПБМ экстраполирует значения составляющих вектора \mathbf{x} путем их представления в виде полинома по степеням $r^{1/2}$. (Если функция $P(\mathbf{x}^{(k)}, r^{(k)})$ строго выпукла, то для каждого $r^{(k)}$ существует только один минимум $P(\mathbf{x}^{(k)}, r^{(k)})$). Экстраполяционные процедуры, основанные на использовании двух или трех последовательных значений \mathbf{x} , соответствующих минимальным значениям

P -функции, называются процедурами первого и второго приближений соответственно (фиг. 7.2.4).

Укажем, что Фиакко и Мак-Кормик доказали, что траектория, проходящая через точки $\mathbf{x}(r^{(k)})$ (каждая из которых является решением подзадачи безусловной минимизации $P(\mathbf{x}, r^{(k)})$), прибли-



Ф и г. 7.2.4. Метод экстраполяции для ускорения оптимизационного поиска с помощью МПБМ.

зительно линейна по $(r^{(k)})^{1/2}$ при $r^{(k)} \rightarrow 0$ (т. е. для малых значений $r^{(k)}$):

$$\mathbf{x}(r^{(k)}) \approx \mathbf{x}(0) + a(r^{(k)})^{1/2} \quad (7.2.11)$$

и

$$\mathbf{x}\left(\frac{r^{(k)}}{c}\right) \approx \mathbf{x}(0) + a\left(\frac{r^{(k)}}{c}\right)^{1/2}, \quad (7.2.12)$$

где a — некоторая константа, а $\mathbf{x}(0)$ — значение $\mathbf{x}(r^{(k)})$ при $r^{(k)} \rightarrow 0$. Таким образом, приближение первого порядка к решению задачи нелинейного программирования находится путем решения уравнений (7.2.11) и (7.2.12) относительно $\mathbf{x}(0)$, в результате чего имеем

$$\mathbf{x}(0) \approx \frac{c^{1/2}\mathbf{x}(r^{(k)}/c) - \mathbf{x}(r^{(k)})}{c^{1/2} - 1}. \quad (7.2.13)$$

Для получения экстраполяционных оценок во втором приближении полагаем

$$\begin{aligned} \mathbf{x}(r^{(k)}) &= \mathbf{x}(0) + a_1(r^{(k)})^{1/2} + a_2(r^{(k)}), \\ \mathbf{x}\left(\frac{r^{(k)}}{c}\right) &= \mathbf{x}(0) + a_1\left(\frac{r^{(k)}}{c}\right)^{1/2} + a_2\left(\frac{r^{(k)}}{c}\right), \\ \mathbf{x}\left(\frac{r^{(k)}}{c^2}\right) &= \mathbf{x}(0) + a_1\left(\frac{r^{(k)}}{c^2}\right)^{1/2} + a_2\left(\frac{r^{(k)}}{c^2}\right). \end{aligned} \quad (7.2.14)$$

Разрешив (7.2.14) относительно $\mathbf{x}(0)$, получаем

$$\mathbf{x}(0) \approx \frac{\mathbf{x}(r^{(k)}) - (c + c^{1/2}) \mathbf{x}(r^{(k)}/c) + c^{3/2} \mathbf{x}(r^{(k)}/c^2)}{(c - 1)(c^{1/2} - 1)}. \quad (7.2.15)$$

Соотношения (7.2.13) и (7.2.15) используются в МПБМ для ускорения процесса оптимизационного поиска и, следовательно, для экономии машинного времени.

Шаг 5. Итак, тесты на сходимость выполнены. Алгоритм содержит три критерия завершения оптимизационного поиска (выбор критерия предоставляется пользователю). Машинная программа реализует останов (прекращение итерационного процесса), если удовлетворяется один из следующих критериев:

1) $\frac{f(\mathbf{x}(r^{(k)}))}{E(\mathbf{x}(r^{(k)}), \mathbf{u}(r^{(k)}), \mathbf{w}(r^{(k)}), \mathbf{w}'(r^{(k)}))} - 1 \leq \theta_0,$

где функция E определена в подразд. 7.2.2.

2)

$$r^{(k)} \sum_{i=m+1}^p \frac{1}{g_i(\mathbf{x}(r^{(k)}))} \leq \theta_0.$$

[Следует заметить, что этот критерий не содержит члена, который позволял бы проверить, удовлетворяются ли условия $h_i(\mathbf{x}(r^{(k)})) = 0$, $i = 1, \dots, m$.]

3)

$$\frac{\text{Значение } f(\mathbf{x}^*) \text{ в первом приближении}}{E(\mathbf{x}(r^{(k)}), \mathbf{u}(r^{(k)}), \mathbf{w}(r^{(k)}), \mathbf{w}'(r^{(k)}))} - 1 \leq \theta_0,$$

где θ_0 — некоторая константа, значение которой выбирается пользователем. Критерии 1 и 3 являются логическим развитием (7.2.9). Значения целевых функций исходной и двойственной задач при уменьшении $r^{(k)}$ обнаруживают тенденцию к совпадению (по крайней мере в задачах с линейными ограничениями в виде равенств), а функции E и P стремятся к $f(\mathbf{x}^*)$. Вычисления заканчиваются, если выполняется выбранный критерий. В противном случае алгоритм реализует переход к шагу 6. Ряд результатов применительно к условиям задачи 1 из приложения А (с использованием критериев 1—3) приведен в табл. 7.2.4; в гл. 9 критерий 1 применяется при реализации вычислительных процедур.

Шаг 6. Как отмечалось выше, машинная программа обеспечивает уменьшение значения $r^{(k)}$.

Шаг 7. Машинная программа обеспечивает приближенное вычисление минимума P -функции для уменьшенного значения $r^{(k)}$ с помощью экстраполяционных формул, приведенных в связи с описанием шага 4.

Шаг 8. Исходя из результата, полученного на шаге 7, программа реализует переход к шагу 3, и итерационный процесс продолжается в установленном выше порядке.

В варианте МПБМ, относящемся к 1967 г., кроме задания целевой функции и ограничений, выраженных через независимые переменные, от пользователя требовалось также ввести в программу аналитические выражения первых и вторых частных производных $f(\mathbf{x})$ и функций, задающих ограничения, по независимым переменным задачи. В варианте МПБМ 1970 г. пользователю предоставляется возможность определить программным способом приближенные значения вторых частных производных с помощью разностных соотношений, содержащих аналитическое представление первых частных производных; эта особенность машинной программы не была, однако, в достаточной степени опробована. Кроме того, МПБМ 1970 г. позволяет применять алгоритм Бройдена с использованием аналитической записи первых частных производных или метод оптимизационного поиска, в котором частные производные вообще не используются. Пользователь должен также задать начальный вектор \mathbf{x} , который не обязательно является допустимым, а также заранее установить значения некоторых из фигурирующих в программе параметров и констант.

Пример 7.2.1. Метод последовательной безусловной минимизации (МПБМ)

Рассмотрим задачу, которая уже решалась в предыдущей главе (здесь она будет решена с помощью МПБМ):

минимизировать $f(\mathbf{x}) = 4x_1 - x_2^2 - 12$
при ограничениях

$$h_1(\mathbf{x}) = 25 - x_1^2 - x_2^2 = 0,$$

$$g_2(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_3(\mathbf{x}) = x_1 \geq 0,$$

$$g_4(\mathbf{x}) = x_2 \geq 0.$$

Функции $f(\mathbf{x})$, $h_1(\mathbf{x})$, $g_2(\mathbf{x})$, $g_3(\mathbf{x})$ и $g_4(\mathbf{x})$ изображены графически на фиг. 6.0.1.

Начнем оптимизационный поиск с точки $\mathbf{x}^{(0)} = [1 \ 1]^T$, в которой $f(\mathbf{x}^0) = -9$; однако, поскольку точка $\mathbf{x}^{(0)}$ является внешней, в соответствии с МПБМ определим прежде всего внутреннюю точку \mathbf{x} (не являющуюся допустимой по отношению к ограничению $h_1(\mathbf{x}) = 0$) путем минимизации $-g_2(\mathbf{x})$ при ограничениях $g_3(\mathbf{x}) \geq 0$ и $g_4(\mathbf{x}) \geq 0$, которые не должны при этом нарушаться. Построим первую из вспомогательных P -функций в виде

$$P' = -g_2(\mathbf{x}) + r \sum_{i=3}^4 \frac{1}{g_i(\mathbf{x})} = -(10x_1 - x_1^2 + 10x_2 - x_2^2 - 34) + \\ + (1) \left(\frac{1}{x_1} + \frac{1}{x_2} \right). \quad (a)$$

В точке $\mathbf{x}^{(0)} = [1 \ 1]^T$ имеем $-g_2(\mathbf{x}) = 16$ и $[(1/x_1) + (1/x_2)] = 2$, так что $P'(\mathbf{x}^{(0)}) = 18$. Присоединенная (расширенная) целевая функция (которая подлежит максимизации) двойственной задачи имеет следующий вид:

$$E = -g_2(\mathbf{x}) - r \sum_{i=3}^4 \frac{1}{g_i(\mathbf{x})} = 14. \quad (6)$$

Минимизация функции P' выполняется методом Ньютона (см. разд. 3.3). Для частных производных функции P' имеем

$$\frac{\partial P'}{\partial x_1} = -10 + 2x_1 - \frac{1}{x_1^2},$$

$$\frac{\partial P'}{\partial x_2} = -10 + 2x_2 - \frac{1}{x_2^2},$$

$$\frac{\partial^2 P'}{\partial x_1^2} = 2 + \frac{2}{x_1^3},$$

$$\frac{\partial^2 P'}{\partial x_2^2} = 2 + \frac{2}{x_2^3},$$

$$\frac{\partial^2 P'}{\partial x_1 \partial x_2} = 0.$$

Таким образом, направление поиска и длина шага в случае минимизации P' задаются соотношением

$$\mathbf{s} = -[\nabla^2 P']^{-1} \nabla P' = -\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}^{-1} \begin{bmatrix} -9 \\ -9 \end{bmatrix} = \begin{bmatrix} 2,25 \\ 2,25 \end{bmatrix}. \quad (7)$$

Шаг длиной 2,25 реализуется вдоль каждой оси координат, в результате чего получаем внутренний вектор \mathbf{x} следующего вида:

$$\mathbf{x}^{(0')} = \mathbf{x}^{(0)} + \mathbf{s} = \begin{bmatrix} 3,25 \\ 3,25 \end{bmatrix}. \quad (8)$$

Для этого вектора удовлетворяются все ограничивающие условия в виде неравенств, а $f(\mathbf{x}) = -20,25$.

Построим теперь основную P -функцию:

$$P = f(\mathbf{x}) + \frac{h_1^2(\mathbf{x})}{\sqrt{r}} + r \left(\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{10x_1 - x_1^2 + 10x_2 - x_2^2 - 34} \right), \quad (9)$$

В точке $\mathbf{x} = [3,25 \ 3,25]^T$ при $r = 1$ эта функция принимает значение 529,716. Подлежащая максимизации целевая функция двойственной задачи имеет вид

$$E = f(\mathbf{x}) - r \left(\frac{1}{x_1} + \frac{1}{x_2} + \frac{1}{10x_1 - x_1^2 + 10x_2 - x_2^2 - 34} \right) + 2rh_1^2(\mathbf{x})$$

и принимает в точке $\mathbf{x} = [3,25 \ 3,25]^T$ значение 1047,72.

Вычислим прежде всего частные производные P -функции в точке $\mathbf{x} = [3,25 \ 3,25]^T$:

$$\begin{aligned}\frac{\partial P}{\partial x_1} &= -46,505, \quad \frac{\partial^2 P}{\partial x_1^2} = 69,084, \quad \frac{\partial^2 P}{\partial x_1 \partial x_2} = 84,525, \\ \frac{\partial P}{\partial x_2} &= -57,006, \quad \frac{\partial^2 P}{\partial x_2^2} = 69,084, \quad \frac{\partial^2 P}{\partial x_2 \partial x_1} = 84,525.\end{aligned}$$

Матрица Гессе для функции P не является положительно определенной, так как при положительном значении определяющего элемента

$$\det \begin{bmatrix} 69,084 & 84,525 \\ 84,525 & 69,084 \end{bmatrix} < 0.$$

Поэтому составляющие вектора, определяющего направление поиска, направлены вдоль составляющих отрицательного градиента функции P . Вектор $\mathbf{x}^{(0)}$ изменяется следующим образом:

$$\mathbf{x} = \mathbf{x}^{(0)} + \Delta \mathbf{x},$$

и для начального шага теперь имеем

$$\begin{bmatrix} 49,755 \\ 60,256 \end{bmatrix} = \begin{bmatrix} 3,25 \\ 3,25 \end{bmatrix} + \begin{bmatrix} 46,505 \\ 57,006 \end{bmatrix}.$$

Далее процесс минимизации P -функции реализуется методом Фибоначчи; поиск продолжается до тех пор, пока не будет достигнута точка $\mathbf{x} = [3,516 \ 3,577]^T$ (в действительности результаты поиска оказываются вычисленными с точностью до пятого десятичного знака).

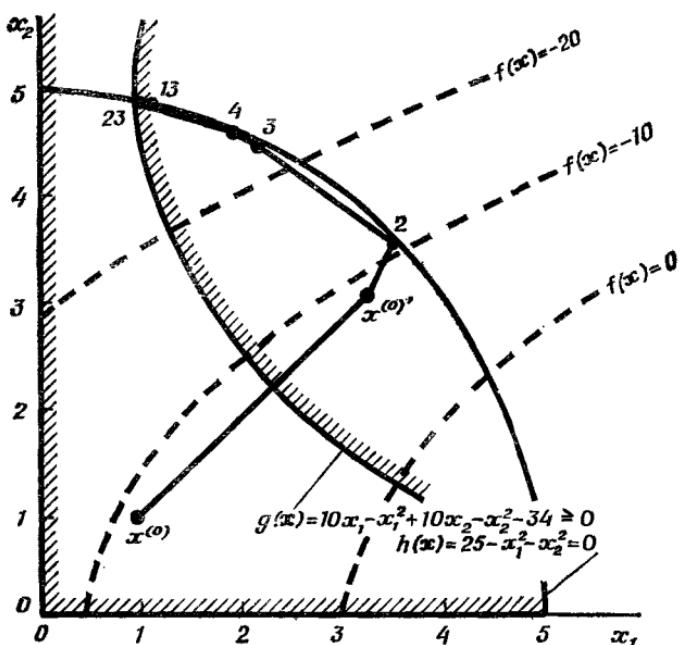
Теперь вновь вычислим частные производные P -функции:

$$\begin{aligned}\frac{\partial P}{\partial x_1} &= 6,119, \quad \frac{\partial^2 P}{\partial x_1^2} = 99,613, \quad \frac{\partial^2 P}{\partial x_1 \partial x_2} = 100,628, \\ \frac{\partial P}{\partial x_2} &= 2,160, \quad \frac{\partial^2 P}{\partial x_2^2} = 101,024, \quad \frac{\partial^2 P}{\partial x_2 \partial x_1} = 100,628.\end{aligned}$$

При этом матрица Гессе для функции P опять не является положительно определенной, так как

$$\det \begin{bmatrix} 99,613 & 100,628 \\ 100,628 & 101,024 \end{bmatrix} < 0.$$

Таким образом, поиск методом Фибоначчи вновь реализуется в направлении отрицательного градиента; этот поиск осуществляется до тех пор, пока не будет достигнута точка $\mathbf{x} = [2,137 \ 4,702]^T$. Минимизация функции P продолжается следующим образом. В точке



Ф и г. П.7.2.1. Траектория оптимизационного поиска с помощью МПБМ (числа означают порядковый номер вычислительного этапа).

\mathbf{x} , для которой матрица Гессе для P -функции положительно определена, используется соотношение (7.2.10).

После 23 итераций вдоль соответствующих направлений оптимизационного поиска значение параметра r (начальное значение $r = 1$) начинают вчетверо уменьшать. Значения переменных x_1 и x_2 , а также функций $f(\mathbf{x})$, $P(\mathbf{x}, r)$ и $E(\mathbf{x}, r)$, вычисленные при уменьшении r , приведены ниже:

Номер этапа	r	$E(\mathbf{x}, r)$	$f(\mathbf{x})$	$P(\mathbf{x}, r)$	x_1	x_2	$h_1(\mathbf{x})$
13	1	-32,990	-31,583	-29,400	1,150	4,918	$-1,01 \cdot 10^{-1}$
23	1/4	-32,270	-31,807	-30,959	1,073	4,909	$-2,53 \cdot 10^{-1}$
33	1/16	-32,065	-31,902	-31,547	1,037	4,904	$-1,26 \cdot 10^{-1}$
42	1/64	-32,011	-31,948	-31,788	1,019	4,902	$-6,34 \cdot 10^{-2}$
51	1/256	-31,997	-31,970	-31,895	1,010	4,900	$-3,17 \cdot 10^{-2}$
55	1/1024	-31,993	-31,944	-31,944	1,006	4,899	$-1,58 \cdot 10^{-2}$
62	1/4096	-31,993	-31,987	-31,969	1,004	4,899	$-7,94 \cdot 10^{-3}$
68	1/16384	-31,992	-31,990	-31,981	1,002	4,899	$-3,95 \cdot 10^{-3}$

Отметим, что имеет место типичная сходимость функций P и E к минимальному значению функции $f(\mathbf{x})$. Обратим также внимание на верхнюю границу, определяемую функцией P , и нижнюю границу, задаваемую функцией E . Количество вычислений, связанных с получением оценок указанных выше функций и их частных производных, на разных этапах оказывается различным и варьируется от 11 на первом этапе до 4 на последнем этапе (число вычислений упомянутого характера зависит, разумеется, от требуемой точности результатов вычислений при одномерном поиске). Траектория оптимизационного поиска показана на фиг. П.7.2.1.

ЗАДАЧИ ¹⁾

7.1. Рассмотрите следующую задачу:

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + 6x_1 + x_2^2 + 9$$

при ограничениях

$$g_i(\mathbf{x}): \quad x_i \geqslant 0, \quad (i = 1, 2).$$

Пусть минимизация осуществляется с помощью МПБМ, начиная из точки $\mathbf{x}^{(0)} = [1 \ 0,5]^T$.

а) Вычислите $r^{(0)}$, используя соотношения (7.2.2) — (7.2.4). Какой способ задания $r^{(0)}$, с вашей точки зрения, является наиболее предпочтительным?

б) Постройте функцию $P(\mathbf{x}, r^{(0)})$.

в) Найдите $\mathbf{x}^{(1)}$ и $f(\mathbf{x}^{(1)})$ с помощью любой оптимизационной процедуры в рамках МПБМ. Каким образом можно избежать опасности выхода за пределы допустимой области?

г) Реализуйте некоторую последовательность этапов оптимизационного процесса.

д) Какие из шести условий, сформулированных в подразд. 7.2.2, выполняются для рассматриваемой задачи?

е) Выполните проверку на сходимость на каждом из вычислительных циклов, используя критерии останова, сформулированные в разд. 7.2. Удовлетворяются ли критерии останова, если P -функция выпуклая?

7.2. Можно ли сформулировать задачу, двойственную по отношению к задаче 7.1? При положительном ответе сформулируйте соответствующую двойственную задачу и реализуйте несколько итерационных циклов при ее решении. Исследуйте тенденции, которые обнаруживают решения исходной и двойственной задач, и проверьте, выполняются ли неравенства (7.2.9).

¹⁾ Ряд задач, имеющих отношение к содержанию данной главы, можно найти также в конце гл. 6 и в приложении А.

7.3. Повторите пункты задачи 7.1 применительно к условиям следующих задач:

а) минимизировать $f(\mathbf{x}) = x_1^2 + x_2^2$
при ограничении

$$h(\mathbf{x}) = x_1^2 + x_2^2 - 9x_2 + 4,25 = 0;$$

б) минимизировать $f(\mathbf{x}) = e^{x_1} + e^{x_2}$
при ограничениях

$$h(\mathbf{x}) = x_1^2 + x_2^2 - 9 = 0,$$

$$g_1(\mathbf{x}) = x_1 + x_2 - 1 \geq 0,$$

$$g_2(\mathbf{x}) = x_1 \geq 0,$$

$$g_3(\mathbf{x}) = x_2 \geq 0.$$

7.4. Прокомментируйте структурные особенности P -функций, построенных для решения следующей задачи:

минимизировать $f(\mathbf{x})$
при ограничениях

$$g_i(\mathbf{x}) \geq 0, \quad i = 1, 2, \dots, m,$$

начиная с допустимой стартовой (начальной) точки. Упомянутые выше P -функции имеют следующий вид:

а) $P(\mathbf{x}, \mathbf{x}^{(k)}) = \frac{1}{f(\mathbf{x}^{(k)}) - f(\mathbf{x})} + \sum_{i=1}^m \frac{1}{g_i(\mathbf{x})},$

б) $P(\mathbf{x}^{(k)}) = f(\mathbf{x}^{(k)}) - r^{(k)} \sum_{i=1}^m \ln g_i(\mathbf{x}^{(k)}).$

Какие преимущества имеют эти функции по сравнению с (7.2.1)?
Какие неудобства возникают при их использовании?

7.5. Изобразите графически уровни штрафной функции Фиакко — Мак-Кормика для следующей двумерной задачи:

минимизировать $f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 2)^2$

при ограничении

$$h(\mathbf{x}) = x_1 + x_2 - 4 = 0.$$

Используйте различные значения параметра r .

7.6. Задача оптимального размещения в трехмерном пространстве единицы оборудования относительно конфигурации, уже содержащей $(n - 1)$ фиксированных (в смысле дислокации) единиц оборудования, формулируется следующим образом:

минимизировать

$$S = \sum_{j=1}^n c_j [(x_{1j} - x_1)^2 + (x_{2j} - x_2)^2 + (x_{3j} - x_3^2)]^{1/2},$$

где S — суммарные затраты, x_1, x_2, x_3 — координаты местонахождения новой единицы оборудования относительно «нулевой точки», x_{1j}, x_{2j}, x_{3j} — координаты уже имеющегося оборудования.

Ограничения имеют вид

$$3x_1 + 3x_3 \leq 30 \text{ и } x_1, x_2, x_3 \geq 0.$$

Значения c_j и x_{ij} указаны в приведенной ниже таблице:

i	c_j	x_{1j}	x_{2j}	x_{3j}
1	1	0	0	0
2	1	10	0	0
3	1	10	10	0
4	1	0	10	0
5	1	0	0	10
6	1	10	0	10
7	1	10	10	10
8	1	0	10	10

а) Покажите, что S — выпуклая функция. (Замечание. Алгебраическая сумма выпуклых функций есть также выпуклая функция, поэтому требуется лишь убедиться в том, что $[(x_1 - a_1)^2 + (x_2 - a_2)^2 + (x_3 - a_3)^2]^{1/2}$ представляет собой выпуклую функцию. Используйте в ходе доказательства неравенство Коши — Шварца)

$$\left[\left(\sum_{i=1}^3 x_i^2 \right) \left(\sum_{i=1}^3 y_i^2 \right) \right]^{1/2} \geq \sum_{i=1}^3 x_i y_i$$

и соотношение (2.4.1), прибавив к правой и левой его частям $\theta^2 \sum_{i=1}^3 x_i^2 + (1 - \theta)^2 \sum_{i=1}^3 y_i^2$. Можно также воспользоваться свойствами «а» и «б», ассоциированными с (2.4.2).)

б) Покажите, что функция, задающая ограничение, является выпуклой.

в) Докажите, что для рассматриваемой задачи выполняются условия 1—6, сформулированные в подразд. 7.2.2.

г) Постройте P -функцию.

д) Покажите, что для $r^{(0)} = 100$ при стартовой (начальной) точке $\mathbf{x}^{(0)} = [0 \ 2 \ 0]^T$ и $S = 86\,540$ первый этап оптимизационного поиска вдоль градиента функции $P(\mathbf{x})$ приводит в точку $\mathbf{x}^{(1)} = [1,03 \ 1,03 \ 1,03]^T$, для которой $P(\mathbf{x}) = 97,237$ и $S(\mathbf{x}) = 82,782$.

е) Каким должно быть очередное значение r ?

ж) Сколько потребуется итераций для того, чтобы понизить значения составляющих вектора $\nabla P(\mathbf{x})$ до числа, меньшего 10^{-2} ? 10^{-4} ?

з) Покажите, что решение рассматриваемой задачи имеет следующий вид:

$$\mathbf{x}^* = [1/3 \quad 1/3 \quad 1/3]^T, \quad P(\mathbf{x}) = 71,817, \quad S(\mathbf{x}) = 71,816.$$

7.7. Проверьте, удовлетворяют ли задачи 6.17, 6.18, 6.20, 6.25 и 6.27 условиям сходимости МПБМ.

7.8. Рассмотрите следующую задачу:

$$\text{минимизировать } f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$$

при ограничениях

$$g_1(\mathbf{x}) = -\frac{x_1^2}{4} - x_2^2 + 1 \geq 0,$$

$$h_2(\mathbf{x}) = x_1 - 2x_2 + 1 = 0.$$

а) Постройте P -функцию Фиакко — Мак-Кормика при $r = 0,04$. Изобразите построенную P -функцию графически (если сможете найти надлежащий способ графического изображения уровня этой функции).

б) Определите направление оптимизационного поиска из начальной внутренней точки $\mathbf{x}^{(0)} = [0,7489 \quad 0,5485]^T$.

в) Найдите вектор $\mathbf{x}^{(1)}$.

г) Может ли одна из точек $\mathbf{x}^{(k)}$ оказаться вне допустимой области? Ответ поясните.

7.9. Преобразуйте задачи, указанные в упражнении 7.7, в задачи минимизации путем использования надлежащих P -функций.

7.10. Сравните (для некоторого числа итераций) скорость минимизационного процесса для задачи, приведенной в подразд. 7.2.1, со скоростью минимизации, если в качестве обобщенной присоединенной функции (вместо рассмотренной в подразд. 7.2.1) взять следующие функции:

а) $P(\mathbf{x}^{(k)}, r^{(k)}) = f(\mathbf{x}) + \frac{1}{r^{(k)}} \sum_i [g_i(\mathbf{x}^{(k)}) - c]^2 + \frac{1}{r^{(k)}} \sum_i [h_i(\mathbf{x}^{(k)}) - c]^2$;

б) функцию, фигурирующую в (7.2.1а).

Придайте константе c некоторое конкретное значение.

7.11. Сравните P -функцию, изображенную графически на фиг. 7.2.1, с P -функциями, упомянутыми в упражнении 7.10. Начертите линии уровней этих P -функций при таких же значениях x_1 и x_2 , как в случае, проиллюстрированном на фиг. 7.2.1. Ответьте для каждого из вариантов выбора P -функции на следующие вопросы:

а) Сохраняется ли разрывность на границе области внутренних точек?

б) Как повлияет на траекторию поиска выбор стартовой (начальной) точки? По-прежнему ли необходимо выбирать в качестве стартовой (начальной) внутреннюю точку?

7.12. Рассмотрите следующую задачу:

$$\text{минимизировать } f(\mathbf{x}) = 4x_1 - x_2^2 - 12$$

при ограничениях

$$g_1(\mathbf{x}) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_2(\mathbf{x}) = x_1 \geq 0,$$

$$g_3(\mathbf{x}) = x_2 \geq 0,$$

выбрав в качестве начальной точку $\mathbf{x}^{(0)} = [2 \ 4]^T$.

Изобразите графически уровни $f(\mathbf{x})$ в области

$$0 < x_1 < 6,$$

$$0 < x_2 < 6$$

(пусть это будет график А). Наложите на полученное вами графическое изображение кривую, заданную уравнением $g_1(\mathbf{x}) = 0$. Начертите уровни постоянных значений для функции (7.1.7) при $\omega_t = 1$ и значениях r , равных $100, 1, 10^{-2}, 10^{-4}$ (т. е. требуется начертить четыре графика: В1, В2, В3, В4). Для каждого значения r минимизируйте $P(\mathbf{x}, r)$, начав оптимизационный поиск в точке $\mathbf{x}^{(0)}$ и используя алгоритм Дэвидона — Флетчера — Пауэлла. Изобразите графически траектории поиска отдельно для случаев В1, В2 и В3 и наложите графики этих траекторий на график А.

Какие выводы можно сделать относительно выбора $r^{(0)}$?

7.13. Мощность, потребляемая миксером-нагревателем промышленного назначения, выражается формулой

$$P = 2,63 \cdot 10^{-3} F_s L^5 \left(\frac{n}{60} \right)^3,$$

где L — диаметр импеллера, фут; n — скорость вращения импеллера, об/мин; μ — вязкость; s — удельный вес; а F — эмпирическое выражение.

Для чисел Рейнольдса¹⁾, превышающих 150, $F = (\mu/3750snL^2)^{0.05}$, тогда как для чисел Рейнольдса, меньших 150, $F = 33,3/(3750snL^2/\mu)^{0.75}$.

Определите минимальную мощность, требуемую для функционирования миксера, вмещающего 6000 галлонов жидкости, удельный вес которой равняется 0,8, а вязкость — 200.

¹⁾ Числа Рейнольдса выражаются через диаметр импеллера, скорость вращения импеллера, удельный вес перемешиваемой жидкости и ее вязкость.

Исходные данные. Для простоты будем считать, что зазор между импеллером и внутренней поверхностью бака, в которую заливается жидкость, равняется 8 дюймам. Число Рейнольдса определяется по формуле

$$\text{Re} = \frac{3750snL^2}{\mu}.$$

Стоимость бака находится из расчета 100 долл на 1 фунт металла, а толщина стенки бака должна быть не менее 0,25 дюйма при высоте до 6 футов и не менее 0,375 дюйма при высоте, превышающей 6 футов. Ежегодные расходы, связанные с обслуживанием и капитальным ремонтом миксера, составляют 25% стоимости бака; затраты на энергоснабжение 0,01 долл за 1 кВт·ч. Суммарное время эксплуатации миксера в течение года составляет 6000 ч.

7.14. Разработка оптимальной модели функционирования химического завода сопряжена с необходимостью решения следующей задачи:

максимизировать

$$f(x) = \frac{8400z - 2,2s - 1041,6(0,3x_5 + 0,0068d)}{6} - 10,$$

где

$$z = 0,3x_5 - 0,02x_1 - 0,03x_2 - 0,01y_5 + 0,0068d,$$

$$s = y_1 + y_2 + y_3 + y_4 + y_5 + y_6,$$

$$x_5 = y_6 - 0,1y_4,$$

$$d = (s - x_5 - y_5)x_3,$$

при ограничениях

$$x_1 - x_3y_1 - \frac{c_1y_1y_2}{s^2} = 0,$$

$$x_2 - x_3y_2 - \frac{c_1y_1y_2}{s^2} - \frac{c_2y_2y_3}{s^2} = 0,$$

$$- x_3y_3 + 2 \frac{c_1y_1y_2}{s^2} - 2 \frac{c_2y_2y_3}{s^2} - \frac{c_3y_3y_6}{s^2} = 0,$$

$$- x_3y_4 + 2 \frac{c_2y_2y_3}{s^2} = 0,$$

$$- y_5 + 1,5 \frac{c_3y_3y_6}{s^2} = 0,$$

$$0,1y_4(1 - x_3) - y_6 + \frac{c_3y_3y_6}{s^2} - 0,5 \frac{c_3y_3y_6}{s^2} = 0,$$

где

$$c_1 = 5,9755 \cdot 10^9 \cdot e^{-12 \cdot 10^3/(x_4+460)},$$

$$c_2 = 2,5962 \cdot 10^{12} \cdot e^{-15 \cdot 10^3/(x_4+460)},$$

$$c_3 = 9,6283 \cdot 10^{15} \cdot e^{-20 \cdot 10^3/(x_4+460)}.$$

7.15. Галлер и Готас [21] сформулировали следующую задачу нелинейного программирования, связанную с проблемой оптимизации затрат на эксплуатацию фильтровальной установки (подробности, относящиеся к постановке задачи, см. в работе [21]):

минимизировать

$$f(\mathbf{x}) = \left(\frac{2c_1\pi W x_1 x_2}{27} + \frac{2c_1\pi W F x_1}{27} + \frac{c_1\pi W^2 x_2}{27} + c_2\pi x_1^2 + \right. \\ \left. + \frac{c_3\pi x_1^2 x_2}{27} + 2c_4 x_1 + \frac{x_3}{c_6 + c_7 x_3} \right) + \frac{c_5 S \times 8,34 \times 365 \times x_3 (x_1 + 1)}{2,65 p}$$

при ограничениях

$$x_1 = \left[\frac{43560(S + x_3)}{\pi Q} \right]^{1/2},$$

$$K_1 \frac{(S L_i + x_3 L_e)^{1,19}}{(S + x_3)^{0,78} (x_2 + 1)^{0,67} (x_1)^{0,25}} - L_e = 0,$$

$$0 < x_3 \leq 4S,$$

$$3 \leq x_2 \leq 10,$$

$$\max \left\{ 10, \left(\frac{43560(S + x_3)}{\pi Q} \right)^{1/2} \right\} \leq x_1 \leq 100.$$

Найдите минимум $f(\mathbf{x})$ и определите \mathbf{x}^* , основываясь на следующих данных:

$$c_1 = 80, \quad K_1 = 0,219,$$

$$c_2 = 4, \quad W = 1,$$

$$c_3 = 10, \quad F = 1,$$

$$c_4 = 53, \quad S = 1,$$

$$c_5 = 3,2 \cdot 10^{-4}, \quad p = 0,7,$$

$$c_6 = 5,55, \quad \pi = 3,14159,$$

$$c_7 = 0,01, \quad L_i = 200,$$

$$L_e = 30.$$

7.16. Клейн и Климпел в работе [22] описали задачу нелинейного программирования, связанную с поиском оптимального решения относительно выбора мест для строительства промышленных предприятий и определения размера этих предприятий в каждом из выбранных мест; предполагается, что речь идет о выработке решения для некоторого заранее установленного интервала времени. Основной и оборотный капитал можно записать в следующем виде:

$$\text{Основной капитал} = a_0 + a_1 S^{a_2},$$

$$\text{Оборотный капитал} = b_0 + b_1 P + b_2 S^{a_4}.$$

Здесь S — размер предприятия, P — проектная мощность предприятия (объем продукции, выпускаемой за год), a_0, b_0, a_1, b_1 и b_2 — известные константы, определяемые эмпирическим способом.

Затраты в течение года (переменная величина) определяются с помощью следующей формулы:

$$\text{Затраты} = P(c_1 + c_2S + c_3S^2).$$

Предполагается, что при заданном пункте отправления и фиксированном пункте назначения транспортные расходы пропорциональны объему перевозок.

Целевой функцией является чистый дисконтированный капитал (ЧДК), т. е. сумма дисконтированных капиталовложений; при этом норма дисконтирования полагается равной 10%.

Предполагается, что движение всех денежных средств, за исключением вкладов на капитальное строительство, происходит в течение года равномерно; оборотный капитал учитывается (либо со знаком плюс, либо со знаком минус) дискретно в начале каждого года, а основной капитал учитывается (со знаком плюс) лишь в начале планового периода (т. е. в начале «нулевого» года).

В непрерывном представлении коэффициенты дисконтирования выглядят следующим образом:

1. Для одновременных вкладов

$$F_i = e^{-ry},$$

где r — процентная ставка на размещенный капитал, y — длительность периода в годах.

2. Для равномерно (во времени) реализуемых вкладов

$$F_u = \frac{e^r - 1}{r} e^{-ry}.$$

Переменная y может принимать как положительные (после начала отсчета времени), так и отрицательные значения (до начала отсчета); в конце года, являющегося началом отсчета времени, $y = 0$.

Поскольку цены и статьи дохода Клейном и Климпелом не рассматривались, максимизация ЧДК эквивалентна минимизации себестоимости.

Обозначим через P_{ijk} объем продукции, доставленной с i -го предприятия ($i = 1, 2, 3, 4$) на рынок сбыта j ($j = 1, 2, 3$) в течение k -го года ($k = 0, 1, 2, 3$). Пусть S_t и \bar{S}_t представляют собой соответственно размер i -го промышленного предприятия и переменную, принимающую значение 0 или 1 в зависимости от того, равняется или не равняется нулю величина S_t . Обозначим через M_{0jk} уровень спроса в j -м пункте сбыта в k -м году. Наконец, для удобства обозначим через P_{i0k} суммарный объем продукции, выпускаемой i -м предприятием в k -м году.

Таблица А

Чистый дисконтированный капитал

1. Доля основного капитала (предприятие № 1)

Год	Основной капитал	Коэффициент дисконтирования	Дисконтированный основной капитал
0 (1967)	$0,7 \bar{S}_1 + 1,5 S_1^{0,6}$	1,0517	$-0,7362 \bar{S}_1 - 1,5775 S_1^{0,6}$

2. Доля оборотного капитала (предприятие № 1)

Конец года	Оборотный капитал	Коэффициент дисконтирования	Дисконтированный оборотный капитал при норме дисконтирования, равной 10%
0	$0,4 \bar{S}_1 + 0,2 P_{101} + 0,05 S_1^{0,6}$	1,000	$-0,4 \bar{S}_1 - 0,2 P_{101} - 0,05 S_1^{0,6}$
1	$0,2 (P_{102} - P_{101})$	0,9048	$-0,1810 P_{102} + 0,1810 P_{101}$
2	$0,2 (P_{103} - P_{102})$	0,8187	$-0,1637 P_{103} + 0,1637 P_{102}$
3	$-0,4 \bar{S}_1 - 0,2 P_{103} - 0,05 S_1^{0,6}$	0,7408	$+0,2963 \bar{S}_1 + 0,1482 P_{103} + 0,0370 S_1^{0,6}$

3. Учет эксплуатационных и прочих затрат (предприятие № 1)

а. Таблица затрат (без учета транспортных расходов)

Год	Показатель	Амортизация ¹⁾	Другие затраты
1	P_{101}	$0,4667 \bar{S}_1 + 1,0 S_1^{0,6}$	$0,03 \bar{S}_1 - 0,01 S_1 + 0,05 S_1^{0,45} + 0,07 S_1^{0,6} + 0,1 P_{101} - 0,005 P_{101} S_1 + 0,4 P_{101} S_1^{-0,55}$
2	P_{102}	$0,1167 \bar{S}_1 + 0,25 S_1^{0,6}$	$0,03 S_1 - 0,01 S_1 + 0,05 S_1^{0,45} + 0,07 S_1^{0,6} + 0,095 P_{102} - 0,0048 P_{102} S_1 + 0,38 P_{102} S_1^{-0,55}$
3	P_{103}	$0,1166 \bar{S}_1 + 0,25 S_1^{0,6}$	$0,03 S_1 - 0,01 S_1 - 0,05 S_1^{0,45} + 0,07 S_1^{0,6} + 0,0903 P_{103} - 0,0045 P_{103} S_1 + 0,361 P_{103} S_1$

6. Результаты дисконтирования затрат (предприятие № 1)¹⁾

Год	Коэффициент дисконтирования	Результаты дисконтирования при норме дисконтирования, равной 10%
1	0,9516	$0,1983 S_1 + 0,0049 \bar{S}_1 - 0,0247 S_1^{0,45} + 0,4221 S_1^{0,6} - 0,0495 P_{101} + 0,0025 P_{101}S_1 - 0,1979 P_{101}S_1^{-0,55}$
2	0,8611	$0,0348 S_1 + 0,0045 \bar{S}_1 - 0,0224 S_1^{0,45} + 0,0720 S_1^{0,6} - 0,0425 P_{102} + 0,0020 P_{102}S_1 - 0,1702 P_{102}S_1^{-0,55}$
3	0,7791	$0,0315 S_1 + 0,0041 \bar{S}_1 - 0,0203 S_1^{0,45} + 0,0651 S_1^{0,6} - 0,0366 P_{103} + 0,0017 P_{103}S_1 - 0,1463 P_{103}S_1^{0,55}$

4. Учет транспортных расходов (при перевозках с предприятия № 1)

Год	Коэффициент дисконтирования	Транспортные расходы	Результаты дисконтирования при норме дисконтирования, равной 10%
1	0,9516	$0,8 P_{121} + 0,5 P_{131}$	$-0,396 P_{121} - 0,247 P_{131}$
2	0,8611	$0,7 P_{122} + 0,45 P_{132}$	$-0,313 P_{122} - 0,201 P_{132}$
3	0,7791	$0,6 P_{123} + 0,4 P_{133}$	$-0,243 P_{123} - 0,162 P_{133}$

¹⁾ Приведенные здесь результаты получены с помощью методов так называемых иаклонных и горизонтальных дисконтирующих сечений.

Таблица Б

Целевая функция

$$\begin{aligned}
Z_{\max} = & -0,5753 \bar{S}_1 - 1,0313 S_1^{0,6} - 0,0685 P_{101} - 0,0597 P_{102} - 0,0522 P_{103} + \\
& + 0,0135 S_1 - 0,0674 S_1^{0,45} + 0,0025 P_{101} S_1 + 0,0020 P_{102} S_1 + \\
& + 0,0017 P_{103} S_1 - 0,1979 P_{101} S_1^{-0,55} - 0,1702 P_{102} S_1^{-0,55} - \\
& - 0,1463 P_{103} S_1^{-0,55} - 0,396 P_{121} - 0,247 P_{131} - 0,313 P_{122} - \\
& - 0,202 P_{132} - 0,243 P_{123} - 0,162 P_{133} - 0,3428 \bar{S}_2 - 0,8920 S_2^{0,6} + \\
& - 0,0685 P_{201} - 0,0597 P_{202} - 0,0522 P_{203} + 0,0135 S_2 - 0,0809 S_2^{0,45} + \\
& + 0,0025 P_{201} S_2 + 0,0020 P_{202} S_2 + 0,0017 P_{203} S_2 - 0,2227 P_{201} S_2^{-0,55} - \\
& - 0,1914 P_{202} S_2^{-0,55} - 0,1645 P_{203} S_2^{-0,55} - 0,396 P_{211} - 0,495 P_{231} - \\
& - 0,313 P_{212} - 0,448 P_{232} - 0,243 P_{213} - 0,405 P_{233} - 0,3164 \bar{S}_3 - \\
& - 1,2987 S_3^{0,6} - 0,0942 P_{301} - 0,0819 P_{302} - 0,0712 P_{303} - 0,0539 S_3^{0,45} + \\
& + 0,0030 P_{301} S_3 + 0,0026 P_{302} S_3 + 0,0022 P_{303} S_3 - 0,2227 P_{301} S_3^{-0,55} - \\
& - 0,1914 P_{302} S_3^{-0,55} - 0,1645 P_{303} S_3^{-0,55} - 0,247 P_{311} - 0,495 P_{321} - \\
& - 0,202 P_{312} - 0,448 P_{322} - 0,162 P_{313} - 0,405 P_{323} - 0,2441 \bar{S}_4 - \\
& - 1,3707 S_4^{0,6} - 0,0577 P_{401} - 0,0504 P_{402} - 0,0440 P_{403} + 0,0020 P_{401} S_4 + \\
& + 0,0017 P_{402} S_4 + 0,0015 P_{403} S_4 - 0,1484 P_{401} S_4^{-0,55} - \\
& - 0,1276 P_{402} S_4^{-0,55} - 0,1097 P_{403} S_4^{-0,55} - 0,495 P_{411} - 0,099 P_{421} - \\
& - 0,040 P_{431} - 0,448 P_{412} - 0,090 P_{422} - 0,040 P_{432} - 0,405 P_{413} - \\
& - 0,088 P_{423} - 0,041 P_{433}
\end{aligned}$$

Таблица Б

Коэффициенты

- | | | |
|---|--|-----------------------------|
| (1) $S_1 + S_2 + S_3 + S_4 = 10$ | (2) $P_{111} + P_{211} + P_{311} + P_{411} = 1$ | |
| (3) $P_{112} + P_{212} + P_{312} + P_{412} = 4$ | (4) $P_{113} + P_{213} + P_{313} + P_{413} = 5$ | |
| (5) $P_{121} + P_{221} + P_{321} + P_{421} = 2$ | (6) $P_{122} + P_{222} + P_{322} + P_{422} = 3$ | |
| (7) $P_{123} + P_{223} + P_{323} + P_{423} = 2$ | (8) $P_{131} + P_{231} + P_{331} + P_{431} = 4$ | |
| (9) $P_{132} + P_{232} + P_{332} + P_{432} = 3$ | (10) $P_{133} + P_{233} + P_{333} + P_{433} = 2$ | |
| (11) $P_{101} - S_2 \leq 0$ | (12) $P_{102} - S_1 \leq 0$ | (13) $P_{103} - S_1 \leq 0$ |
| (14) $P_{201} - S_2 \leq 0$ | (15) $P_{202} - S_2 \leq 0$ | (16) $P_{203} - S_2 \leq 0$ |
| (17) $P_{301} - S_3 \leq 0$ | (18) $P_{302} - S_3 \leq 0$ | (19) $P_{303} - S_3 \leq 0$ |
| (20) $P_{401} - S_4 \leq 0$ | (21) $P_{402} - S_4 \leq 0$ | (22) $P_{403} - S_4 \leq 0$ |

Задача нелинейного программирования заключается в том, чтобы определить S_i и P_{ijk} , максимизирующие \sum_i ЧДК (с учетом транспортных расходов) при ограничениях

$$\begin{aligned} \sum_i P_{ijk} &= M_{0jk}, \\ S_i &\geq 0, \quad P_{ijk} \geq 0. \end{aligned}$$

В табл. А показан способ определения ЧДК для предприятия № 1; для других предприятий ЧДК находится аналогичным образом. В табл. Б приведена полная структура целевой функции, а в табл. В указаны 22 ограничения: одно ограничение в виде равенства, лимитирующее полную производственную мощность предприятия (10 миллионов фунтов в год), девять равенств, обусловленных требованием полного удовлетворения спроса на трех рынках сбыта в течение года, и 12 неравенств, ограничивающих превышение парциальных производственных мощностей рассматриваемого предприятия. Кроме того, следует иметь в виду условия неотрицательности для всех переменных (число переменных равняется 40). Таким образом, рассматриваемая задача содержит 10 ограничений в виде равенств и 52 ограничения в виде неравенств.

ЛИТЕРАТУРА

1. Fiacco A. V., McCormick G. P., *Nonlinear Programming*, Wiley, N. Y., 1968.
2. Zangwill W. I., *Management Sci.*, **13**, 344 (1967).
3. Huard P., Resolution de programmes mathématiques à contraintes nonlinéaires par la méthode des centres, Note Electricité de France, HR 5690/3/317, 1964; см. также The Method of Centers in: Course A., *Nonlinear Programming*, North Holland Publ. Co., Amsterdam, 1965.
4. Fiacco A. V., McCormick G. P., *Operations Res.*, **16**, 820 (1968).
5. Pietrzykowski T., Application of the Steepest Descent Method to Concave Programming, Proc. IFIPS Congr., Munich, North Holland Publ. Co., Amsterdam, 1962.
6. Fiacco A. V., McCormick G. P., *J. Soc. Ind. Appl. Math.*, **15**, 505 (1967).
7. Fiacco A. V., Ph. D. Dissertation, Northwestern Univ., Evanston, Ill., 1967.
8. Dennis J. B., *Mathematical Programs and Electrical Networks*, Wiley, N. Y., 1959.
9. Box M. J., Davies D., Swann W. H., *Nonlinear Optimization Techniques*, ICI Monograph of Mathematics and Statistics, № 5, Oliver and Boyd Ltd., London, 1969.
10. Carroll C. W., *Operations Res.*, **9**, 169 (1961).
11. Davies D., Some Practical Methods for Optimization, Notes for the NATO Summer School on Integer and Nonlinear Programming, June 8—20, 1969.
12. Weisman J., Ph. D. Dissertation, Univ. of Pittsburgh, Pa., 1968.
13. Fiacco A. V., McCormick G. P., *SIAM J. Appl. Math.*, **15**, 505 (1967).
14. Huard P., p. 209 in: *Nonlinear Programming*, Abadie J., ed., North Holland Publ., Amsterdam, 1967.

15. Carroll C. W., *Operations Res.*, 9, 169 (1961); Ph. D. Dissertation, Inst. of Paper Chemistry, Appleton, Wis., 1959.
16. Fiacco A. V., McCormick G. P., *Management Sci.*, 10, 360, 601 (1964); 12, 816 (1966).
17. Wortman J. D., BRL 1958 (NLPROG), Jan. 1969.
18. Murray W., Proc. 6th Intern. Symp. on Mathematical Programming, Princeton, N. J., 1967.
19. Fletcher R., McCann A. P., Chap. 13 in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
20. Tabak D., *IEEE Trans. Automatic Control*, AC14, 572 (1969).
21. Galler W. S., Gotas H. B., *J. Sanit. Eng. Div., Am. Soc. Civil Engr.*, SA1, 163 (1966).
22. Klein M., Klimpel R. R., *J. Indus. Eng.*, 18, 90 (1967).

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

МЕТОДЫ МНОЖИТЕЛЕЙ ЛАГРАНЖА

- Arrow K. J., Hurwicz L., Gradient Methods for Constrained Optimization, *J. Operations Res. Soc.*, 5, 258 (1957).
- Bard Y., Greenstadt J. L., A Modified Newton Method for Optimization with Equality Constraints, in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
- Dorn W. S., On Lagrange Multipliers and Inequalities, *J. Operations Res. Soc.*, 9, 95 (1961).
- Everett H., Generalized Lagrange Multiplier Methods for Solving Problems of Optimal Allocation of Resources, *Operations Res.*, 11, 399 (1969).
- Falk J. E., Lagrange Multipliers and Nonconvex Programs, Res. Analysis Corp. Techn. Paper RAC-TP-335, Nov. 1968; Lagrange Multipliers and Nonlinear Programming, *J. Math. Analysis Appl.*, 19, 141 (1967).
- Klein B., Direct Use of Extremal Principles in Solving Certain Optimizing Problems Involving Inequalities, *J. Operations Res. Soc.*, 3, 169 (1955).
- Kuhn H. W., Tucker A. W., Nonlinear Programming, Proc. 2nd Berkeley Symp. on Math. Statist. Prob., Univ. California Press, Berkeley, Calif., 1951, p. 481.
- Takahashi I., Variable Separation Principle in Mathematical Programming, *J. Operations Res. Japan*, Vol. 6 (1964).
- Zwart, *J. Opt. Theory and Appl.*, 6, 150 (1970).

МПБМ

- Bracken J., McCormick G. P., Selected Applications of Nonlinear Programming, Wiley, N. Y., 1968.
- Fiacco A. V., McCormick G. P., Nonlinear Programming: Sequential Unconstrained Minimization Techniques, Wiley, N. Y., 1968.
- Lootsma F. A., Logarithmic Programming: A Method of Solving Nonlinear Programming Problems, *Phillips Res. Rept.*, 22, 329 (1967); Constrained Optimization via Penalty Functions, *Phillips Res. Rept.*, 23, 408 (1968).
- Pomentale T., A New Method for Solving Conditioned Maxima Problems, *J. Math. Analysis Appl.*, 10, 216 (1965).
- Schmit L. A., An Integrated Approach to Structural Analysis and Synthesis, *AIAA J.*, 3, 1104 (1964).

ДРУГИЕ МЕТОДЫ

- Allran R. R., Johnsen S. E. J., *Computer J.*, 13, 171 (1970).
- Bellmore M., Greenberg H. J., Jarvis J. J., *Oper. Res.*, 17, 229 (1969).

- Fletcher R., McCann A. P., Acceleration Techniques for Nonlinear Programming, in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
- Haarhoff P. C., Buys J. D., Computer J., 13, 78 (1970).
- Huard P., Resolution of Mathematical Programming with Nonlinear Constraints by the Method of Centers, in: Nonlinear Programming, Abadie J., ed., North Holland Publ. Co., Amsterdam, 1967.
- Kelley H. J., Denham W. G., Johnson I. L., Wheatley P. O., An Accelerated Gradient Method for Parameter Optimization with Nonlinear Constraints, *J. Astronautical Sci.*, 13, 166 (1966).
- Kowalik J., Osborne M. R., Ryan D. M., A New Method for Constrained Optimization Problems, *Operations Res.*, 17, 973 (1969).
- Lasdon L. S., An Efficient Algorithm for Minimizing Barrier and Penalty Functions, Techn. Memorandum № 210, Operations Res. Depart., Case Western Reserve Univ., Dec. 1970.
- Lootsma F. A., Boundary Properties of Penalty Functions for Constrained Minimization, Nat. Techn. Inform. Service, Document N70-33412, 1970.
- Morrison D. D., Optimization by Least Squares, *SIAM J. Numerical Analysis*, 5, 83 (1968).
- Murray W., Constrained Optimization, Nat. Phys. Lab. Rept. № NA 79, Aug. 1969.
- Murray W., Behavior of Hessian Matrices of Barrier and Penalty Functions Arising in Optimization, Nat. Phys. Lab. Rept. № NA 77, April 1969.
- Powell M. J. D., A Method for Nonlinear Constraints in Minimization Problems, in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
- Schwartz L. E., Large Step Gradient Methods, Chap. 8 in: Optimization, Fletcher R., ed., Academic Press, London, 1969.
- Siddall J. N., McDonald J. F., OPTIPAC: The Designers Optimization Problem Solver, 2 vols., McMaster Univ., Mech. Engineering Depart., Hamilton, Ont., Canada, Oct. 1969.

Глава 8

ПРОЦЕДУРЫ МИНИМИЗАЦИИ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ: МЕТОД СКОЛЬЗЯЩЕГО ДОПУСКА

•

Напомним, что в общей постановке задача нелинейного программирования формулируется следующим образом:

минимизировать $f(x)$, $x \in E^n$,

при ограничениях

$$\begin{aligned} h_i(x) = 0, \quad i = 1, \dots, m, \\ g_i(x) \geq 0, \quad i = m + 1, \dots, p, \end{aligned} \tag{8.0.1}$$

где функции $f(x)$, $h_i(x)$ и $g_i(x)$ могут быть как линейными, так и нелинейными¹⁾. При практической реализации на ЭВМ многих методов нелинейного программирования значительная доля машинного времени тратится на то, чтобы обеспечить строгое выполнение требований допустимости. Алгоритм скользящего²⁾ допуска [1] позволяет улучшить значения целевой функции как за счет информации, получаемой в допустимых точках пространства решений, так и за счет информации, которую удается получить при прохождении через некоторые точки, лежащие вне допустимой области, но являющиеся близкими к допустимым; для краткости будем называть их *почти допустимыми* точками. Интервалы, в пределах которых точки можно считать почти допустимыми, в ходе оптимизационного поиска постепенно сокращаются, так что в пределе (по мере приближения к искомому решению задачи нелинейного программирования) в системе соотношений (8.0.1) учитываются только допустимые точки x . При такой стратегии оптимизационного поиска задачу (8.0.1) можно заменить более простой (но имеющей то же самое решение) задачей

минимизации $f(x)$, $x \in E^n$,

при ограничении

$$\Phi^{(k)} - T(x) \geq 0, \tag{8.0.2}$$

где $\Phi^{(k)}$ — значение критерия скользящего допуска на k -м этапе поиска [этот критерий определяется соотношениями (8.1.1)], а $T(x)$

¹⁾ Как уже отмечалось в разд. 2.1, при линейной структуре *всех* фигурирующих в (8.0.1) функций мы имеем дело с задачей *линейного программирования*.

²⁾ Иногда говорят «нежесткого допуска».

представляет собой положительно определенный функционал над множеством всех функций, задающих ограничения (как в виде равенств, так и в виде неравенств) в задаче (8.0.1). Функционал $T(\mathbf{x})$ является мерой степени нарушения ограничений рассматриваемой задачи [определение $T(\mathbf{x})$ дано с помощью соотношения (8.1.5) в разд. 8.1]. В разд. 8.2 приводится стратегия алгоритма. Разд. 8.3 посвящен описанию процедуры определения необходимой совокупности как допустимых, так и почти допустимых точек, а в разд. 8.4 обсуждается вычислительная процедура в начале оптимизационного поиска. Отметим, что хотя при решении задач безусловной минимизации используется так называемый метод деформируемого многогранника, предложенный Нелдером и Мидом (см. разд. 4.2), так как он оказывается исключительно эффективным, при решении задачи безусловной минимизации можно применять и другие методы, не связанные со стратегией скользящего допуска. Таким образом, метод Нелдера и Мида можно заменить любым другим методом определения безусловного минимума, если при этом гарантируется надлежащая степень эффективности вычислительных процедур. Получаемая при этом последовательность векторов \mathbf{x} будет просто представлять собой последовательность точек в E^n , а не вершины специально построенного многогранника.

8.1. ОПРЕДЕЛЕНИЕ Φ , $T(\mathbf{x})$ И ПОЧТИ ДОПУСТИМЫХ ТОЧЕК

8.1.1. КРИТЕРИЙ ДОПУСКА Φ

В качестве критерия допуска Φ выбирается положительно определенная убывающая функция координат точек, являющихся вершинами деформируемого многогранника в E^n , т. е. $\Phi^{(k)} = \Phi^{(k)}(\mathbf{x}_1^{(k)}, \mathbf{x}_2^{(k)}, \dots, \mathbf{x}_{r+1}^{(k)}, \mathbf{x}_{r+2}^{(k)})$. Функция Φ служит критерием допуска для нарушения ограничений решаемой задачи на протяжении всего процесса оптимизационного поиска и, кроме того, является критерием, позволяющим определить момент прекращения процедуры оптимизации. Варианты конкретного выбора Φ многочисленны; здесь мы рассмотрим лишь одну из возможностей, представив функцию Φ в следующем виде:

$$\Phi^{(k)} = \min \left\{ \Phi^{(k-1)}, \frac{m+1}{r+1} \sum_{i=1}^{r+1} \|\mathbf{x}_i^{(k)} - \mathbf{x}_{r+2}^{(k)}\| \right\}, \quad (8.1.1)$$

$$\Phi^{(0)} = 2(m+1)t,$$

где

t — величина, характеризующая размер исходного многогранника;

m — число ограничений в виде равенств;

$\mathbf{x}_i^{(k)}$ — вектор, задающий положение i -й вершины многогранника в E^n ;

$r = (n - m)$ — число степеней свободы целевой функции $f(\mathbf{x})$ в задаче (8.0.1);

$\mathbf{x}_{r+2}^{(k)}$ — вектор, задающий положение вершины, которая соответствует «центру тяжести» рассматриваемого многогранника при $n = r$ [см. соотношение (4.2.1)];

$k = 0, 1, \dots$ — индекс, указывающий число полностью законченных этапов вычислительного процесса;

$\Phi^{(k-1)}$ — значение Φ на $(k - 1)$ -м этапе оптимизационного поиска.

Обозначим второй член в фигурных скобках соотношения (8.1.1) через $\theta^{(k)}$, т. е. положим

$$\begin{aligned}\theta^{(k)} &= \frac{m+1}{r+1} \sum_{i=1}^{r+1} \|\mathbf{x}_i^{(k)} - \mathbf{x}_{r+2}^{(k)}\| = \\ &= \frac{m+1}{r+1} \left\{ \sum_{i=1}^{r+1} \sum_{j=1}^n (x_{ij}^{(k)} - x_{r+2,j}^{(k)})^2 \right\}^{1/2},\end{aligned}\quad (8.1.2)$$

где $x_{ij}^{(k)}$ ($j = 1, \dots, n$) — координаты i -й вершины нежесткого многогранника в E^n . Заметим, что $\theta^{(k)}$ представляет собой среднее расстояние от точек $\mathbf{x}_i^{(k)}$ ($i = 1, \dots, r + 1$) до центра тяжести $\mathbf{x}_{r+2}^{(k)}$ выбранного многогранника в E^n . Чтобы понять поведение функции $\Phi^{(k)}$, необходимо прежде всего рассмотреть величину θ . Очевидно, что θ зависит от размеров упомянутого выше многогранника, которые могут оставаться неизменными, увеличиваться или уменьшаться в зависимости от того, какая из четырех описанных в разд. 4.2 операций используется для перехода от $\mathbf{x}_i^{(k)}$ к $\mathbf{x}_i^{(k+1)}$. Таким образом, $\Phi^{(k)}$ ведет себя как положительно определенная убывающая функция \mathbf{x} , хотя $\theta^{(k)}$ может в ходе оптимизационного поиска либо возрастать, либо убывать, а по мере приближения к оптимальной точке и $\theta^{(k)}$, и $\Phi^{(k)}$ устремляются к нулю, т. е. образуется последовательность

$$\Phi^{(0)} \geq \Phi^{(1)} \geq \dots \geq \Phi^{(k)} \geq 0. \quad (8.1.3)$$

Если при использовании метода Нелдера и Мида улучшение значений $f(\mathbf{x})$ с помощью (4.2.2) оказывается невозможным, вершины деформируемого многогранника постепенно приближают к той вершине, которая соответствует наименьшему значению целевой функции. В пределе имеет место полное вырождение деформируемого многогранника в точку, соответствующую стационарному

значению $f(\mathbf{x})$. Таким образом, по мере приближения к стационарному значению $f(\mathbf{x})$ величина $\theta^{(k)}$, определяемая соотношением (8.1.2), постепенно уменьшается, так как среднее расстояние между вершинами многогранника и его центром тяжести сокращается до нуля. Поскольку на k -м этапе оптимизационного поиска $\Phi^{(k)}$ получается равной наименьшему из значений либо $\Phi^{(k-1)}$, либо $\theta^{(k)}$, величина $\Phi^{(k)}$ также уменьшается и в пределе имеем

$$\lim_{\mathbf{x} \rightarrow \mathbf{x}^*} \Phi^{(k)} = 0. \quad (8.1.4)$$

8.1.2. КРИТЕРИИ $T(\mathbf{x})$ КАК «МЕРА» СТЕПЕНИ НАРУШЕНИЯ ОГРАНИЧЕНИЙ

Рассмотрим теперь функционал $T(\mathbf{x})$ над множеством ограничений задачи (8.0.1):

$$T(\mathbf{x}) = + \left[\sum_{i=1}^m h_i^2(\mathbf{x}) + \sum_{i=m+1}^p \mathcal{U}_i g_i^2(\mathbf{x}) \right]^{1/2}, \quad (8.1.5)$$

где \mathcal{U}_i — оператор Хевисайда, обладающий следующим свойством: $\mathcal{U}_i = 0$ при $g_i(\mathbf{x}) \geq 0$ и $\mathcal{U}_i = 1$ при $g_i(\mathbf{x}) < 0$. Таким образом, функционал $T(\mathbf{x})$ представляет собой взятый с положительным знаком квадратный корень из суммы квадратов функций, задающих полную совокупность нарушенных ограничений задачи (8.0.1). Заметим, что $T(\mathbf{x}) \geq 0$ при любом $\mathbf{x} \in E^n$. В частности, обратим внимание на то, что если сумма $\sum_{i=1}^m h_i^2(\mathbf{x})$ является выпуклой, а функции $g_i(\mathbf{x})$ ($i = m+1, \dots, p$) вогнуты, то $T(\mathbf{x})$ есть выпуклая функция, обладающая глобальным минимумом $T(\mathbf{x}) = 0$ для любого допустимого вектора \mathbf{x} , т. е. для любого $\{\mathbf{x} | h_i(\mathbf{x}) = 0, g_i(\mathbf{x}) \geq 0 (i = 1, \dots, p)\}$. Кроме того, $T(\mathbf{x}) > 0$ для любого вектора \mathbf{x} , не являющегося допустимым. Для заданного $\mathbf{x}^{(k)} \in E^n$ вычисленное с помощью формулы (8.1.5) значение $T(\mathbf{x})$ можно использовать для различия допустимых и недопустимых точек. Если $T(\mathbf{x}^{(k)}) = 0$, то точка $\mathbf{x}^{(k)}$ допустима; если $T(\mathbf{x}) > 0$, то точка $\mathbf{x}^{(k)}$ не является допустимой. При этом существенным является следующее обстоятельство: если значение $T(\mathbf{x}^{(k)})$ мало, то это означает, что точка $\mathbf{x}^{(k)}$ расположена относительно недалеко от границы допустимой области; при большом же значении $T(\mathbf{x}^{(k)})$ точка $\mathbf{x}^{(k)}$ лежит на значительном расстоянии от границы допустимой области.

8.1.3. ПОЧТИ ДОПУСТИМЫЕ ТОЧКИ (КВАЗИДОПУСТИМОСТЬ)

Почти допустимыми точками называются такие точки $\mathbf{x} \in E^n$, которые не являются допустимыми, но в то же время почти допустимы в указанном ниже смысле. Чтобы установить четкое различие между допустимыми, почти допустимыми и недопустимыми точками, рассмотрим значение Φ на k -м этапе оптимизационного поиска, т. е. значение Φ в точке $\mathbf{x}^{(k)} \in E^n$. Говорят, что вектор $\mathbf{x}^{(k)}$ является:

- 1) допустимым, если $T(\mathbf{x}^{(k)}) = 0$;
- 2) почти допустимым, если $0 \leq T(\mathbf{x}^{(k)}) \leq \Phi^{(k)}$;
- 3) недопустимым, если $T(\mathbf{x}^{(k)}) > \Phi^{(k)}$.

Таким образом, область квазидопустимости определяется соотношением

$$\Phi^{(k)} - T(\mathbf{x}) \geq 0. \quad (8.1.6)$$

Любое перемещение из точки $\mathbf{x}^{(k)}$ в точку $\mathbf{x}^{(k+1)}$ называется допустимым, если $T(\mathbf{x}^{(k+1)}) = 0$, почти допустимым, если $0 \leq T(\mathbf{x}^{(k+1)}) \leq \Phi^{(k)}$, и недопустимым, если $T(\mathbf{x}^{(k+1)}) > \Phi^{(k)}$. Отметим, что значение Φ на $(k+1)$ -м этапе оптимизационного поиска находится только после того, как определяется, что точка $\mathbf{x}^{(k+1)}$ является либо допустимой, либо почти допустимой.

8.2. СТРАТЕГИЯ АЛГОРИТМА СКОЛЬЗЯЩЕГО ДОПУСКА

В этом разделе мы покажем, что задачу нелинейного программирования в общей формулировке (8.0.1) можно заменить более простой задачей минимизации $f(\mathbf{x})$ при единственном «укрупненном» ограничении в виде неравенства, а именно задачей следующей структуры:

минимизировать $f(\mathbf{x})$, $\mathbf{x} \in E^n$,

при ограничении

$$\Phi^{(k)} - T(\mathbf{x}) \geq 0. \quad (8.2.1)$$

Поиск методом деформируемого многогранника (т. е. методом Нелдера и Мида) удобен и эффективен, хотя и не относится к числу основных методов при минимизации $f(\mathbf{x})$ в отсутствие ограничений, т. е. когда ограничение в (8.2.1) не является активным. Однако этот метод используется и для минимизации $T(\mathbf{x})$ (см. разд. 8.3) при ограничении, фигурирующем в (8.2.1), если это ограничение активно. Общая схема работы алгоритма выглядит следующим образом: по мере развития оптимизационного поиска уменьшается

значение $\Phi^{(k)}$, что приводит к сужению области квазидопустимости, и процедура минимизации $f(x)$ отделяется от этапов, служащих для выполнения ограничения, указанного в (8.2.1). При заданном $\Phi^{(k)}$ в точке $x^{(k+1)}$ имеет место один из следующих вариантов:

1) $T(x^{(k+1)}) \leq \Phi^{(k)}$. В этом случае точка $x^{(k+1)}$ является либо допустимой, либо почти допустимой; соответствующее перемещение можно считать разрешенным.

2) $T(x^{(k+1)}) > \Phi^{(k)}$. В этом случае точка $x^{(k+1)}$ классифицируется как недопустимая; необходимо отыскать вместо $x^{(k+1)}$ другую точку, которая либо лежала бы ближе к границе допустимой области, либо принадлежала допустимой области. Один из способов перемещения точки $x^{(k+1)}$ в сторону допустимой области состоит в уменьшении значения $T(x^{(k+1)})$ в соответствии с (8.1.5) до тех пор, пока не будет выполнено условие $T(x^{(k+1)}) \leq \Phi^{(k)}$.

Чтобы убедиться в том, что решение задачи (8.2.1) эквивалентно решению задачи (8.0.1), достаточно проанализировать поведение $\Phi^{(k)}$. Поскольку $\Phi^{(k)}$ есть положительно определенная невозрастающая функция, так что $\Phi^{(k)} = 0$ лишь в том случае, когда дальнейшее улучшение значения $f(x)$ для задачи (8.2.1) невозможно, область квазидопустимости, определяемая соотношением (8.1.6), постепенно уменьшается по мере того, как оптимизационный поиск приближает нас к решению задачи (8.2.1). В пределе, когда все вершины $x_i^{(k)}$ ($i = 1, \dots, r + 1$) деформируемого многогранника в E^n стягиваются в одну точку x^* , $\Phi^* = 0$ и условию (8.1.6) могут удовлетворять лишь допустимые точки x , т. е. точки $\{x | h_i(x) = 0, g_i(x) \geq 0 \text{ } (i = 1, \dots, p)\}$. Другими словами, если $\Phi^{(k)} = 0$, то, поскольку $T(x)$ не может принимать отрицательных значений, единственным возможным значением $T(x)$ является $T(x) = 0$, что эквивалентно требованию, чтобы были удовлетворены все ограничивающие условия в задаче (8.0.1).

Поскольку критерий допустимости Φ есть положительно определенная невозрастающая функция над последовательностью точек $x^{(0)}, x^{(1)}, \dots, x^{(k)}, \dots, x^*$, которые генерируются в ходе оптимизационного поиска, а также в силу того, что Φ не зависит ни от $f(x)$, ни от функций, задающих ограничения задачи, и в пределе $\Phi^* = 0$, сходимость алгоритма гарантируется по следующим причинам:

1. Невозрастающее поведение критерия допустимости обеспечивается самим способом построения Φ [см. формулу (8.1.1)]. Если бы функция Φ могла неограниченно возрастать, появилась бы возможность улучшать значение $f(x)$, все более удаляясь от границы допустимой области.

2. Когда решением задачи (8.0.1) является внутренняя точка (при отсутствии ограничений в виде равенств), сходимость алгоритма гарантируется в силу свойства деформируемого многогран-

ника вырождаться в точку лишь при достижении оптимума $f(\mathbf{x})$ в задаче (8.0.1). При этих обстоятельствах $T(\mathbf{x})$ не влияет на сходимость алгоритма, так как на заключительных этапах поиска точки $\mathbf{x}_i^{(k)}$ ($i = 1, \dots, r+1$) являются внутренними и, таким образом, $T(\mathbf{x}_i^{(k)}) = 0$, откуда следует, что в (8.1.6) ограничивающие условия в виде неравенств удовлетворяются для любых $\mathbf{x}_i^{(k)}$ и задача (8.0.1) не содержит активных ограничений.

3. Когда оптимум задачи (8.0.1) достигается в точке, не являющейся внутренней [либо потому, что \mathbf{x}^* является граничной точкой, либо потому, что задача (8.0.1) содержит только ограничения в виде равенств], сходимость гарантируется за счет условия (8.1.6), т. е. в силу того, что $\Phi^{(k)} - T(\mathbf{x}) \geq 0$. Деформируемый многогранник не начнет вырождаться в точку до тех пор, пока не будет найдено такое улучшенное значение $f(\mathbf{x}_i^{(k)})$, для которого $\Phi^{(k)} - T(\mathbf{x}_i^{(k)}) \geq 0$.

Пусть $\mathbf{x}_i^{(k)}$ — вершины деформируемого многогранника, такие, что $\Phi^{(k)} - T(\mathbf{x}_i^{(k)}) \geq 0$, а $f(\mathbf{x}_i^{(k)})$ — наилучшее значение $f(\mathbf{x})$, полученное на k -м этапе оптимизационного поиска. Обозначим через $\mathbf{x}_i^{(k+1)}$ вершины, полученные путем построения точек, симметричных точкам $\mathbf{x}_i^{(k)}$ относительно центра тяжести многогранника, так чтобы $\Phi^{(k)} - T(\mathbf{x}_i^{(k+1)}) \geq 0$. Если $f(\mathbf{x}_i^{(k+1)}) > f(\mathbf{x}_i^{(k)})$ для любого симметричного преобразования относительно центра тяжести, $\Phi^{(k)}$ будет убывать вследствие сжатия элементов многогранника. Таким образом, значения $\Phi^{(k)}$ уменьшаются, и $\mathbf{x}_i^{(k)}$ должны удовлетворять ограничивающим условиям задачи (8.0.1) с постоянно возрастающей точностью, пока оптимизационный поиск не прекратится из-за того, что окажется выполненным условие $\Phi^{(k)} \leq \varepsilon$.

С другой стороны, если $f(\mathbf{x}_i^{(k+1)}) \leq f(\mathbf{x}_i^{(k)})$, то значение $\Phi^{(k)}$ не уменьшится, поскольку никаких сжатий многогранника не будет иметь места, и $\mathbf{x}_i^{(k)}$ заменяется на более подходящую вершину. Построение симметричных (относительно центра тяжести) точек и рас-tяжение элементов деформируемого многогранника осуществляются в ходе поиска допустимой или почти допустимой точки [для которой $f(\mathbf{x}_i^{(k+1)}) \leq f(\mathbf{x}_i^{(k)})$] столько раз, сколько это необходимо. Таким образом, преждевременное прекращение поиска в окрестности нелокального оптимума не имеет места, так как деформируемый многогранник не вырождается в точку, если существует такой вектор $\mathbf{x}_i^{(k+1)}$, для которого $\Phi^{(k)} - T(\mathbf{x}_i^{(k+1)}) \geq 0$ и $f(\mathbf{x}_i^{(k+1)}) \leq f(\mathbf{x}^* \pm \varepsilon)$.

Одно из преимуществ стратегии скользящего допуска, нашедшее отражение в структуре задачи (8.2.1), заключается в том, что степень нарушения ограничений, содержащихся в задаче (8.0.1),

по мере приближения к искомому решению этой задачи постепенно уменьшается. Поскольку на первых этапах поиска ограничения задачи (как в виде равенств, так и в виде неравенств) должны удовлетворяться весьма приблизительно и лишь при поиске непосредственно в окрестности искомого решения задачи (8.2.1) требуется большая точность, полный объем вычислений в процессе оптимизации по сравнению с другими методами существенно сокращается.

Другим преимуществом стратегии скользящего допуска является то, что оказывается удобным использовать $\Phi^{(k)}$ в качестве критерия окончания процесса поиска. Во всех возникающих на практике ситуациях оптимизационный поиск достаточно продолжать до тех пор, пока $\Phi^{(k)}$ не станет меньше некоторого произвольным образом выбранного положительного числа ε . На заключительных этапах поиска $\Phi^{(k)}$ является также мерой среднего расстояния от вершин деформируемого многогранника $x_i^{(k)}$ ($i = 1, \dots, r + 1$) до его центра тяжести $x_{r+2}^{(k)}$. Если $\Phi^{(k)} \leq \varepsilon$, то значительное число вершин $x_i^{(k)}$ содержится внутри гипосферы радиуса ε . (Если бы последний из рассматриваемых многогранников был правильным, внутрь гиперсферы радиуса ε попали бы все вершины $x_i^{(k)}$; в силу же того, что многогранник является неправильным, некоторые из вершин выходят за пределы упомянутой гиперсферы.) Таким образом, если $\Phi^{(k)} \leq \varepsilon$, не исключается возможность того, что значение $f(x)$ не удастся улучшить без дальнейшего уменьшения размеров деформируемого многогранника. Отсюда, в частности, следует, что замена $\varepsilon \rightarrow 2\varepsilon$ в точке $x_i^{(k)}$, соответствующей наилучшему значению $f(x)$ (т. е. в точке $x_i^{(k)}$), не приведет к улучшению целевой функции. Следовательно, к моменту прекращения оптимизационного поиска выполняется следующее условие:

$$f(x_i^{(k)}) \leq f(x^* \pm \varepsilon). \quad (8.2.2)$$

Поскольку условие (8.1.6) удовлетворяется при любом перемещении в пространстве решений, если $\Phi^{(k)} \leq \varepsilon$, то условие $\varepsilon - T(x_i^{(k)}) \geq 0$, очевидно, также выполняется, и мы имеем

$$T(x_i^{(k)}) = \left[\sum_{i=1}^m h_i^2(x) + \sum_{i=m+1}^p U_i(x) g_i^2(x) \right]^{1/2} \leq \varepsilon. \quad (8.2.3)$$

Из соотношения (8.2.3) следует, что при прекращении поиска суммарное значение функций, ассоциированных с нарушенными ограничениями, не превышает ε . Само собой разумеется, что и значение каждой отдельно взятой функции, ассоциированной с тем или иным из нарушенных ограничений, также не может превысить ε .

Пример 8.2.1. Стратегия скользящего допуска

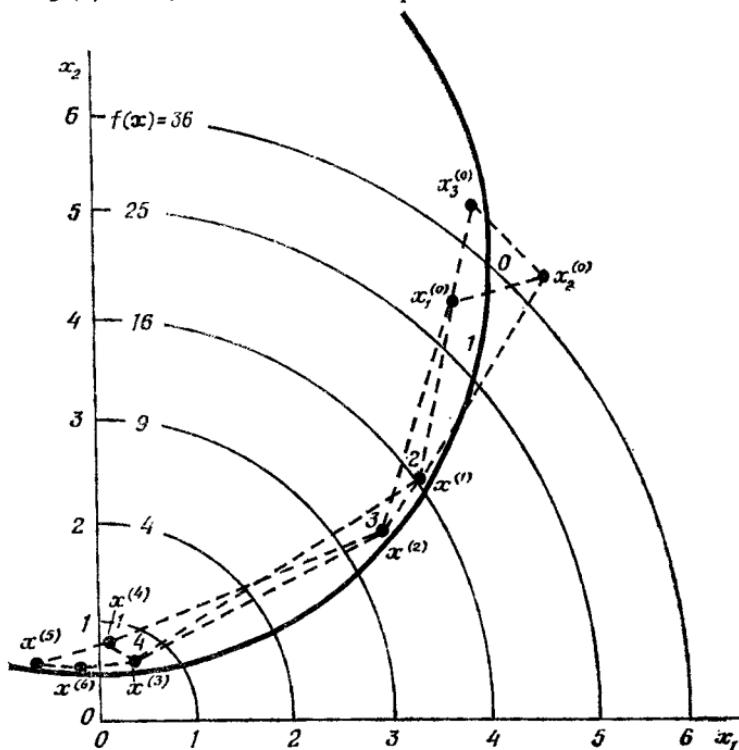
Рассмотрим следующую задачу:

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + x_2^2, \quad \mathbf{x} \in E^n,$$

при ограничении

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 - 9x_2 + 4,25 = 0. \quad (\text{а})$$

Уровни целевой функции $f(\mathbf{x})$ и геометрическое место точек, для которых $h_1(\mathbf{x}) = 0$, показаны на фиг. П.8.2.1. Решением задачи



Фиг. П.8.2.1. Минимизация при наличии ограничений с помощью алгоритма скользящего допуска.

(а) является точка $\mathbf{x}^* = [0 \ 0,5]^T$, в которой $f(\mathbf{x}^*) = 0,25$ и $h_1(\mathbf{x}^*) = 0$. В соответствии с методом скользящего допуска задача (а) преобразуется в задачу

$$\text{минимизировать } f(\mathbf{x}) = x_1^2 + x_2^2, \quad \mathbf{x} \in E^n,$$

при ограничении

$$\Phi^{(k)} - T(\mathbf{x}) \geqslant 0,$$

где $T(\mathbf{x}) = [(x_1^2 + x_2^2 - 9x_2 + 4,25)^2]^{1/2}$. Заметим, что в рассматриваемом примере $r = (n - m) = 1$, так что при отыскании минимума $f(\mathbf{x})$ можно использовать $r + 1 = 2$ вершин. Однако в силу того,

что методика интерполяции (см. разд. 8.3.2) требует три опорные точки, число вершин q всегда должно удовлетворять условию $q \geq 3$. Именно поэтому в данном примере рассматриваются три вершины деформируемого многогранника. При десяти переменных и пяти ограничениях будет использовано только пять вершин.

Для иллюстрации работы алгоритма скользящего допуска ниже приводится подробное описание вычислительных операций на первых двух этапах решения задачи (б). В качестве начального выбирается вектор $\mathbf{x}^{(0)} = [4 \ 4,5]^T$, а величина, характеризующая размер исходного многогранника, $t = 1$ (при этом мы будем иметь многогранник с единичными ребрами и центром тяжести в точке $\mathbf{x}^{(0)}$). В силу (8.1.1) при $t = 1$ и $m = 1$ имеем $\Phi^{(0)} = 4$. Вершины исходного многогранника расположены в точках $\mathbf{x}_1^{(0)} = [3,592 \ 4,092]^T$, $\mathbf{x}_2^{(0)} = [4,558 \ 4,351]^T$ и $\mathbf{x}_3^{(0)} = [3,85 \ 5,06]^T$. Для соответствующих значений $f(\mathbf{x})$ имеем $f(\mathbf{x}_1^{(0)}) = 29,64$, $f(\mathbf{x}_2^{(0)}) = 40,4$ и $f(\mathbf{x}_3^{(0)}) = 40,4$.

НУЛЕВОЙ ЭТАП ОПТИМИЗАЦИОННОГО ПОИСКА ($k = 0$)

Вершина, соответствующая наименьшему (на данном этапе) значению $f(\mathbf{x})$, находится в точке $\mathbf{x}_l^{(0)} = \mathbf{x}_1^{(0)}$ [в этой точке $f(\mathbf{x}_l^{(0)}) = 29,64$], а вершина, соответствующая наибольшему (на данном этапе) значению $f(\mathbf{x})$, находится в точке $\mathbf{x}_h^{(0)} = \mathbf{x}_3^{(0)}$ [в этой точке $f(\mathbf{x}_h^{(0)}) = 40,4$]. Значение $T(\mathbf{x}) = [(x_1^2 + x_2^2 - 9x_2 + 4,25)^2]^{1/2}$ в точке $\mathbf{x}_l^{(0)}$ равняется $T(\mathbf{x}_l^{(0)}) = 2,65$, и, поскольку $\Phi^{(0)} - T(\mathbf{x}_l^{(0)}) > 0$, неравенство (8.1.6) удовлетворяется и точка $\mathbf{x}_l^{(0)}$ может рассматриваться как почти допустимая. Нет никакой необходимости в том, чтобы неравенство (8.1.6) удовлетворялось для двух других вершин, поскольку в процессе оптимизации эти вершины будут заменены другими вершинами с улучшенными значениями $f(\mathbf{x})$. Согласно (4.2.1), центр тяжести для оставшихся вершин находится в точке $\mathbf{x}_4^{(0)} = [4,07 \ 4,22]^T$. В соответствии с (4.2.2) при $\alpha = 1$ точка, симметричная точке $\mathbf{x}_h^{(0)}$ относительно $\mathbf{x}_4^{(0)}$, задается вектором $\mathbf{x}_5^{(0)} = [4,28 \ 3,37]^T$.

В точке $\mathbf{x}_5^{(0)}$ функционал $T(\mathbf{x})$ принимает значение $T(\mathbf{x}_5^{(0)}) = 3,75$. Поскольку $\Phi^{(0)} - T(\mathbf{x}_5^{(0)}) > 0$, точка $\mathbf{x}_5^{(0)}$ является почти допустимой; при этом $f(\mathbf{x}_5^{(0)}) = 29,6$. Вследствие того что $f(\mathbf{x}_5^{(0)}) < f(\mathbf{x}_l^{(0)})$, согласно (4.2.3), при $\gamma = 2$ следующий шаг состоит в выполнении операции растяжения, что дает $\mathbf{x}_6^{(0)} = [4,49 \ 2,52]^T$; в этой точке $T(\mathbf{x}_6^{(0)}) = 8$. Поскольку $\Phi^{(0)} - T(\mathbf{x}_6^{(0)}) < 0$, точка $\mathbf{x}_6^{(0)}$ не является почти допустимой и поэтому отбрасывается. Таким образом, минимизация $T(\mathbf{x})$ начинается из точки $\mathbf{x}_6^{(0)}$ и продолжается до тех пор, пока не будет удовлетворено неравенство (8.1.6).

Получающаяся в результате минимизации $T(\mathbf{x})$ новая вершина $\mathbf{x}_6^{(0)}$ задается вектором $\mathbf{x}_6^{(0)} = [3,32 \ 2,40]^T$, для которого $T(\mathbf{x}_6^{(0)}) = 0,59$, и, следовательно, точка $\mathbf{x}_6^{(0)}$ почти допустима. Теперь нетрудно убедиться, что в почти допустимой точке $\mathbf{x}_6^{(0)}$ функция $f(\mathbf{x}_6^{(0)}) = 16,8$. Поскольку $f(\mathbf{x}_6^{(0)}) < f(\mathbf{x}_i^{(0)})$, точка $\mathbf{x}_h^{(0)} = \mathbf{x}_3^{(0)}$ заменяется на $\mathbf{x}_6^{(0)}$; на этом нулевой этап ($k = 0$) оптимизационного поиска заканчивается. Как показано на фиг. П.8.2.1, вершина $\mathbf{x}_6^{(0)}$ отождествляется с точкой $\mathbf{x}^{(1)}$.

ПЕРВЫЙ ЭТАП ОПТИМИЗАЦИОННОГО ПОИСКА ($k = 1$)

Вершинами деформируемого многогранника в начале первого ($k = 1$) этапа оптимизационного поиска являются точки $\mathbf{x}_1^{(0)} = [3,592 \ 4,092]^T$, $\mathbf{x}_2^{(0)} = [4,558 \ 4,351]^T$ и $\mathbf{x}^{(1)} = [3,32 \ 2,40]^T$. Обратим внимание на то, что при переходе от этапа к этапу происходит замена только одной вершины многогранника. С помощью формулы (8.1.1) для критерия допуска получаем $\Phi^{(1)} = 0,745$. Вершины, в которых $f(\mathbf{x})$ принимает наибольшее и наименьшее (на данном этапе) значения, находятся соответственно в точках $\mathbf{x}_h^{(1)} = \mathbf{x}_i^{(0)}$ и $\mathbf{x}_l^{(0)} = \mathbf{x}^{(1)}$ [при этом имеем $f(\mathbf{x}^{(1)}) = 16,8$ и $T(\mathbf{x}^{(1)}) = 0,59$]. Поскольку $\Phi^{(1)} - T(\mathbf{x}^{(1)}) > 0$, неравенство (8.1.6) оказывается выполненным, и поиск продолжается. Центр тяжести находится в точке $\mathbf{x}_4^{(1)} = [3,45, 3,29]^T$. Симметричное отображение точки $\mathbf{x}_h^{(1)} = \mathbf{x}_2^{(1)}$ относительно точки $\mathbf{x}_4^{(1)}$ дает точку $\mathbf{x}_5^{(1)} = [2,34 \ 2,23]^T$, в которой $T(\mathbf{x}_5^{(1)}) = 5,50$. В силу того что $\Phi^{(1)} - T(\mathbf{x}_5^{(1)}) < 0$, точка $\mathbf{x}_5^{(1)}$ не является допустимой; поэтому осуществляется минимизация $T(\mathbf{x})$, исходя из точки $\mathbf{x}_5^{(1)}$.

В новой точке $\mathbf{x}_5^{(1)} = [3,1 \ 2,1]^T$ имеем $T(\mathbf{x}_5^{(1)}) = 0,64$. Поскольку $\Phi - T(\mathbf{x}_5^{(1)}) > 0$, производится вычисление $f(\mathbf{x})$ в точке $\mathbf{x}_5^{(1)}$. Нетрудно убедиться, что $f(\mathbf{x}_5^{(1)}) = 14,01$. В силу того что $f(\mathbf{x}_5^{(1)}) < f(\mathbf{x}_i^{(1)})$, в соответствии с (4.2.3) выполняется операция растяжения, в результате чего получаем $\mathbf{x}_6^{(1)} = [2,75 \ 09]^T$. В этой точке $T(\mathbf{x}_6^{(1)}) = 4,5$. Поскольку точка $\mathbf{x}_6^{(1)}$ не является допустимой, производится минимизация $T(\mathbf{x})$ при стартовой точке $\mathbf{x}_6^{(1)}$ и находится новая точка $\mathbf{x}_6^{(1)} = [2,92 \ 1,90]^T$, в которой $T(\mathbf{x}_6^{(1)}) = 0,71$, так что эта новая точка $\mathbf{x}_6^{(1)}$ оказывается почти допустимой. В силу того что $f(\mathbf{x}_6^{(1)}) < f(\mathbf{x}_i^{(1)})$, $\mathbf{x}_h^{(1)} = \mathbf{x}_2^{(0)}$ заменяется на $\mathbf{x}_6^{(1)}$; на этом первый этап ($k = 1$) оптимизационного поиска заканчивается. Точка $\mathbf{x}_6^{(1)}$ отождествляется (см. фиг. П.8.2.1) с точкой $\mathbf{x}^{(2)}$.

Вычислительная процедура, выполняемая по схеме, проиллюстрированной для первых двух ($k = 0$ и $k = 1$) этапов оптимиза-

ционного поиска, повторяется на последующих этапах $k = 2, 3, \dots$; при этом на каждом этапе вершина, соответствующая наибольшему значению $f(x)$, заменяется на новую вершину, получаемую путем выполнения соответствующих операций по методу Нелдера и Мида. Эти операции повторяются до тех пор, пока найденное с помощью (8.1.1) значение $\Phi^{(k)}$ не будет удовлетворять условию $\Phi^{(k)} \leq \varepsilon$, где ε — заранее выбранное число. При выполнении указанного выше условия оптимизационный поиск заканчивается.

Подробно каждый шаг алгоритма скользящего допуска описан в разд. 8.4, где содержится также блок-схема алгоритма.

8.3. ПРОЦЕДУРА ОТЫСКАНИЯ ДОПУСТИМЫХ И ПОЧТИ ДОПУСТИМЫХ ТОЧЕК

В предыдущем разделе было установлено, что в процессе оптимизационного поиска при улучшении значений целевой функции $f(x)$ используются только допустимые или почти допустимые точки. Если какая-либо точка $x^{(k)}$ оказывается с точки зрения критерия Φ недопустимой, необходимо определить другую точку, которую можно было бы квалифицировать либо как допустимую, либо как почти допустимую. Отыскание либо допустимой, либо почти допустимой точки осуществляется за счет минимизации $T(x^{(k)})$ над множеством всех точек пространства решений; при этом процесс минимизации $T(x^{(k)})$ продолжается до тех пор, пока не окажется выполненным неравенство (8.1.6).

Для минимизации $T(x)$ методом Нелдера и Мида необходимо построить новый многогранник в окрестности недопустимой точки $x^{(k)}$. Чтобы не возникло путаницы, будем обозначать вершины многогранников, рассматриваемых в связи с процедурой улучшения значения $f(x)$, через $\hat{x}_i^{(k)}$ ($i = 1, \dots, r+1$), а вершины многогранников, рассматриваемых при отыскании допустимых или почти допустимых точек путем минимизации $T(x)$, — через $\hat{x}_i^{(s)}$ ($i = 1, \dots, n+1$). Последовательность векторов, генерируемых в процессе минимизации $T(x)$, для каждого недопустимого вектора $x_i^{(k)}$ будет, таким образом, представляться последовательностью $\hat{x}_i^{(0)}, \hat{x}_i^{(1)}, \dots, \hat{x}_i^{(s)}$ ($i = 1, \dots, n+1$), где индекс s обозначает число полностью завершенных этапов процесса минимизации $T(x)$. В любой процедуре минимизации $T(x)$ в качестве начальной всегда берется точка $\hat{x}_i^{(0)} = x_i^{(k)}$, где $x_i^{(k)}$ — недопустимая точка на k -м этапе процесса минимизации $f(x)$. Последняя вершина последовательности $\hat{x}_i^{(0)}, \hat{x}_i^{(1)}, \dots, \hat{x}_i^{(s)}$ считается достигнутой в том случае, когда для некоторого $\hat{x}_i^{(s)}$ выполняется

условие $T(\hat{\mathbf{x}}_i^{(s)}) \leq \Phi^{(k)}$. При этом недопустимая вершина $\hat{\mathbf{x}}_i^{(k)}$ заменяется допустимой или почти допустимой вершиной $\hat{\mathbf{x}}_i^{(s)}$, для которой $\Phi^{(k)} - T(\hat{\mathbf{x}}_i^{(k)}) \geq 0$, где $\hat{\mathbf{x}}_i^{(k)} = \hat{\mathbf{x}}_i^{(s)}$.

Процедура отыскания допустимых или почти допустимых точек будет описана в подразд. 8.3.3, а до этого рассмотрим:

- 1) метод получения $(n + 1)$ вершин исходного многогранника в E^n (подразд. 8.3.1);
- 2) метод интерполяции между внутренними и внешними точками (подразд. 8.3.2).

8.3.1. МЕТОД ПОЛУЧЕНИЯ $(n + 1)$ ВЕРШИН ИСХОДНОГО МНОГОГРАННИКА

Пусть $\hat{\mathbf{x}}^{(0)} = \mathbf{x}_i^{(k)}$ — недопустимая точка в E^n . Чтобы приступить к поиску в связи с минимизацией значения $T(\mathbf{x})$ методом Нелдера и Мида, требуется задать $(n + 1)$ исходных точек $\hat{\mathbf{x}}_i^{(0)} (i = 1, \dots, n + 1)$, которые могут (либо не могут) образовывать правильный многогранник в E^n . Эти $(n + 1)$ вершин должны быть выбраны таким образом, чтобы n векторов, соответствующих любому подмножеству n вершин, были линейно независимы. На практике всегда оказывается наиболее удобным строить правильный многогранник, беря в качестве базовой точки $\hat{\mathbf{x}}^{(0)}$. Вершины исходного многогранника в E^n [число вершин, как уже отмечалось, равняется $(n + 1)$] находятся с помощью соотношения

$$\hat{\mathbf{x}}_i^{(0)} = \hat{\mathbf{x}}^{(0)} + \mathbf{D}_i, \quad i = 1, \dots, n + 1, \quad (8.3.1)$$

где \mathbf{D}_i — вектор-столбец, составляющими которого являются элементы i -го столбца $[n \times (n + 1)]$ -мерной матрицы. Определение этой матрицы дано в разд. 4.2.

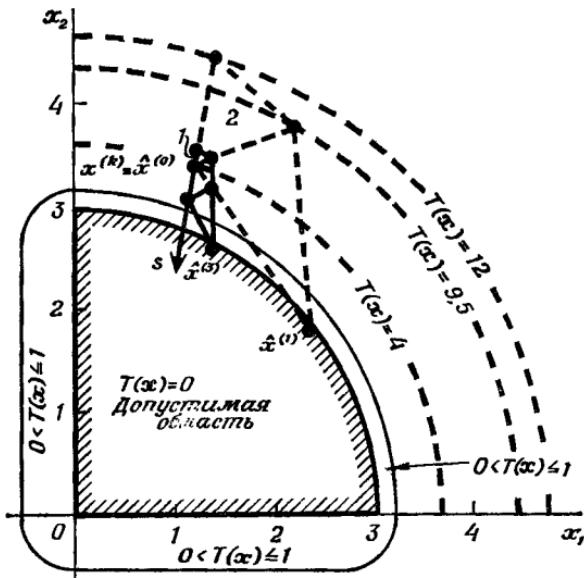
Траектория поиска, получаемая путем минимизации $T(\mathbf{x})$ методом Нелдера и Мида, зависит от размера и ориентации исходного многогранника в пространстве решений. В случае когда размеры многогранника малы по сравнению с размерами допустимой области, траектория поиска почти не зависит от ориентации этого многогранника в E^n . При малых значениях t траектория поиска (по крайней мере на первых этапах поиска) мало отличается от траектории, получаемой методом наискорейшего спуска.

Чтобы проиллюстрировать влияние размеров многогранника на ход оптимизационного поиска на начальных этапах, рассмотрим совокупность ограничений (8.3.2), которые графически представлены

на фиг. 8.3.1:

$$\begin{aligned} g_1(x) : 9 - x_1^2 - x_2^2 \geq 0, \\ g_2(x), g_3(x) : x_1, x_2 \geq 0. \end{aligned} \quad (8.3.2)$$

Допустимая область указана здесь штрихами по ее периметру. В любой точке допустимой области $T(x) = 0$, тогда как в любой точке, лежащей вне допустимой области, $T(x) > 0$. Уровни $T(x) =$



Фиг. 8.3.1. Влияние размеров исходного деформируемого многогранника на ход минимизации $T(x)$.

$= 12$, $T(x) = 9,5$ и $T(x) = 4$, выходящие за пределы указанной допустимой области, показаны на фиг. 8.3.1 пунктирными кривыми. На фиг. 8.3.1 изображены также два равносторонних треугольника, имеющих общую вершину в недопустимой точке $x^{(k)} = [1,2 \ 3,4]^T$, в которой $T(x^{(k)}) = 4$. Если применить методы Нелдера и Мида, то потребуется всего один этап для того, чтобы установить допустимую точку $\hat{x}^{(1)}$, начав с рассмотрения большего из упомянутых двух треугольников. Если начать рассмотрение с меньшего треугольника, то допустимую точку $\hat{x}^{(3)}$ удастся определить лишь по завершении трех этапов. Направление наискорейшего спуска для $T(x)$ в точке $x^{(k)}$ задается вектором $s = [-0,4 \ 0,9]^T$. (Это направление также показано на фиг. 8.3.1.) Чем меньше размеры исходного многогранника, тем ближе проходит траектория поиска к траектории наискорейшего спуска. Чтобы предотвратить возможность осцилляции с пересечением границы допустимой области, весьма желательна такая ситуация, когда

траектория поиска при минимизации $T(\mathbf{x})$ не слишком сильно удаляется от траектории наискорейшего спуска для $T(\mathbf{x})$.

В алгоритме скользящего допуска величина, характеризующая размер исходного многогранника, при минимизации $T(\mathbf{x})$ на k -м этапе определяется с помощью эмпирической формулы

$$t = 0,05 \Phi^{(k)}, \quad (8.3.3)$$

где $\Phi^{(k)}$ — значение критерия допуска, вычисленное с помощью (8.1.1) на k -м этапе. Следует помнить, что размеры многогранника, используемого при минимизации $f(\mathbf{x})$, фиксируются в начале этой вычислительной процедуры и сокращаются только, когда векторы \mathbf{x} не приводят к улучшению значения $f(\mathbf{x})$.

На первых этапах поиска почти допустимые точки $\mathbf{x}^{(k)}$ расположены дальше от границы допустимой области, нежели на этапах, завершающих оптимизационный процесс. На фиг. 8.3.1 множество точек, для которых $T(\mathbf{x}) = 0$, образует допустимую область, а множество точек, для которых $0 < T(\mathbf{x}) \leq \Phi^{(k)}$, образует квазидопустимую область. Квазидопустимая область при $\Phi^{(k)} = 1$ представлена на фиг. 8.3.1 узкой полосой, обрамляющей допустимую область. Следует отметить, что эта полоса в случае линейных ограничений оказывается более широкой, чем в случае нелинейных ограничений.

8.3.2. МЕТОД ИНТЕРПОЛЯЦИИ МЕЖДУ ВНУТРЕННИМИ И ВНЕШНИМИ ТОЧКАМИ

В случае когда задача (8.0.1) содержит только ограничения в виде неравенств, метод Нелдера и Мида может оказаться неэффективным. Проиллюстрируем это, рассмотрев следующую задачу:

минимизировать $f(\mathbf{x}) = -x_1 - x_2$

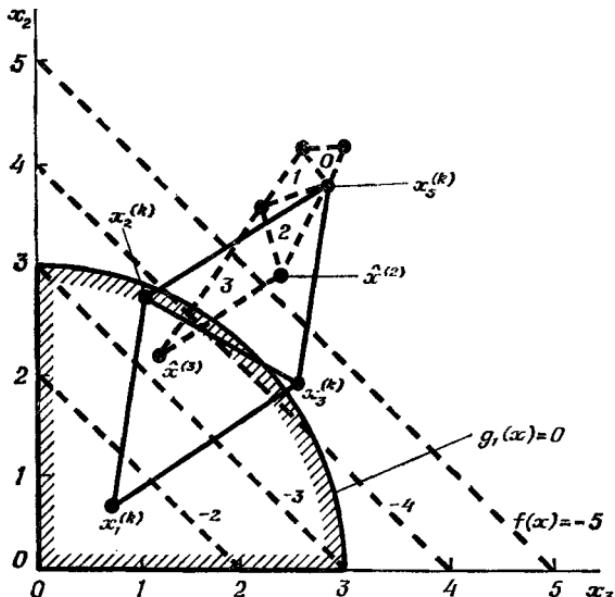
при ограничениях

$$\begin{aligned} g_1(\mathbf{x}): 9 - x_1^2 - x_2^2 &\geq 0, \\ g_2(\mathbf{x}), \quad g_3(\mathbf{x}): x_1, \quad x_2 &\geq 0. \end{aligned} \quad (8.3.4)$$

На фиг. 8.3.2 допустимая область задачи (8.3.4) показана заштрихованной кромкой на внутренней пограничной части контура, заданного уравнениями $g_i(\mathbf{x}) = 0$ ($i = 1, 2, 3$), а уровни целевой функции $f(\mathbf{x})$ изображены наклонными пунктирными линиями. Допустим, что на k -м этапе поиска вершины многогранника (фактически треугольника), рассматриваемого в связи с минимизацией $f(\mathbf{x})$, находятся в точках $\mathbf{x}_1^{(k)} = [0,7 \ 0,7]^T$, $\mathbf{x}_2^{(k)} = [1 \ 2,7]^T$ и $\mathbf{x}_3^{(k)} = [2,5 \ 1,8]^T$ (фиг. 8.3.2). В этих точках $f(\mathbf{x})$ принимает соответственно значения $f(\mathbf{x}_1^{(k)}) = -1,4$, $f(\mathbf{x}_2^{(k)}) = -3,7$ и $f(\mathbf{x}_3^{(k)}) = -4,3$. Симметричное отображение точки $\mathbf{x}_1^{(k)}$ относительно центра тяжести

точек $x_2^{(k)}$ и $x_3^{(k)}$ (соответствующая процедура обсуждалась в разд. 4.2) дает точку $x_5^{(k)} = [2,8 \ 3,8]^T$, лежащую вне допустимой области.

В результате безусловной минимизации $T(x)$ методом Нелдера и Мида находится допустимая точка $\hat{x}^{(3)} = [1,15 \ 2,15]^T$ (фиг. 8.3.2).



Фиг. 8.3.2. Пример «выброса» при нарушенных ограничениях (метод Нелдера и Мида).

Траектория поиска, в результате которого удалось найти точку $\hat{x}^{(3)}$, представлена пунктирной ломаной линией 0, 1, 2, 3. Поскольку точка $x_5^{(k)}$ находилась на относительно большом расстоянии от границы допустимой области, точка $\hat{x}^{(3)}$ оказалась глубоко внутри допустимой области задачи (8.3.4).

После замены $x_5^{(k)}$ на $\hat{x}^{(3)}$ (последнюю точку для удобства будем также обозначать через $x_5^{(k)}$) получаем $f(x_5^{(k)}) = -3,3$. Заметим, однако, что вершине $x_5^{(k)}$ теперь соответствует наибольшее [по сравнению со значениями $f(x)$ в двух других вершинах] значение $f(x)$. Действительно, $f(x_5^{(k)}) = -3,3$, тогда как $f(x_2^{(k)}) = -3,7$ и $f(x_3^{(k)}) = -4,3$. Отсюда следует, что точка $x_5^{(k)}$ подлежит симметричному отображению относительно центра тяжести точек $x_2^{(k)}$ и $x_3^{(k)}$, в результате чего получается новая точка, лежащая вне допустимой области. Такого рода осцилляция относительно границы допустимой области может продолжаться достаточно долго без существен-

ного улучшения значения $f(x)$. Чтобы в какой-то мере исправить ситуацию (которая, как правило, возникает при решении задач, содержащих только ограничения-неравенства), по отношению к внутренним и внешним точкам осуществляется квадратичная интерполяция, с помощью которой находится точка x , лежащая вблизи от границы, заданной нарушенными ограничениями. На фиг. 8.3.2 нарушенным является ограничение, задаваемое функцией $g_1(x)$.

Пусть $\hat{x}^{(s)}$ — внутренняя точка, а $\hat{x}^{(s-1)}$ — ближайшая к ней внешняя точка, найденная путем минимизации $T(x)$. Если вновь обратиться к фиг. 8.3.2, то нетрудно убедиться, что $\hat{x}^{(s)} = \hat{x}^{(3)} = [1,15 \ 2,15]^T$, а, $\hat{x}^{(s-1)} = \hat{x}^{(2)} = [2,35 \ 2,9]^T$. Любая точка на отрезке между точками $\hat{x}^{(s)}$ и $\hat{x}^{(s-1)}$ задается соотношением

$$\hat{x} = \hat{x}^{(s)} + \lambda^{(s)} \hat{s} \text{ при } 0 \leq \lambda^{(s)} \leq \lambda^*, \quad (8.3.5)$$

где $\lambda^* = \left[\sum_{j=1}^n (\hat{x}_j^{(s-1)} - \hat{x}_j^{(s)})^2 \right]^{1/2}$ есть расстояние от точки $\hat{x}^{(s)}$ до точки $\hat{x}^{(s-1)}$, а $\hat{s} = (\hat{x}^{(s-1)} - \hat{x}^{(s)}) / \lambda^*$ представляет собой единичный вектор в направлении $\hat{x}^{(s-1)} - \hat{x}^{(s)}$. Пусть $Z(x) = \sum_{i=1}^p g_i^2(x)$, где \hat{p} — суммарное число ограничений в виде неравенств, оказавшихся нарушенными в точке $\hat{x}^{(s-1)}$. Вычислим $Z(x)$ в точках $\hat{x}^{(s)}$, $\hat{x}^{(s)} + 0,5\lambda^* \hat{s}$ и $\hat{x}^{(s-1)} = \hat{x}^{(s)} + \lambda^* \hat{s}$ и положим $z_1 = Z(\hat{x}^{(s)})$, $z_2 = Z(\hat{x}^{(s)} + 0,5\lambda^* \hat{s})$ и $z_3 = Z(\hat{x}^{(s-1)})$. Таким образом, z_1 , z_2 и z_3 представляют собой значения $Z(x)$ в трех одинаково отстоящих друг от друга точках, расположенных вдоль вектора $\hat{x}^{(s)} - \hat{x}^{(s-1)}$. Желательно найти такую точку \hat{x}^* , в которой $Z(\hat{x}^*)$ почти не отличается от нуля. Такая точка определяется соотношением

$$\hat{x}^* = \hat{x}^{(s)} + \left(\frac{\beta + \sqrt{\beta^2 - 8\alpha z_1}}{4\alpha} \right) \lambda^* \hat{s}, \quad (8.3.6)$$

где $\alpha = z_1 - 2z_2 + z_3$, а $\beta = 3z_1 - 4z_2 + z_3$. Соотношение (8.3.6) можно получить, записывая приближение $Z(x)$ с точностью до второго порядка в интервале, определяемом λ^* . При этом рассматриваются только положительные вещественные корни $(\beta^2 - 8\alpha z_1)^{1/2}$.

В примере, проиллюстрированном на фиг. 8.3.2, $Z(x) = (9 - x_1^2 - x_2^2)^2$, $\hat{x}^{(s)} = [1,15 \ 2,15]^T$ и $\hat{x}^{(s-1)} = [2,35 \ 2,9]^T$. Следовательно, $\lambda^* = 1,37$, $\hat{s} = [0,837 \ 0,548]^T$ и $\hat{x}^{(s)} + 0,5\lambda^* \hat{s} = [1,72 \ 2,53]^T$. При этом получаем $z_1 = Z(\hat{x}^{(s)}) = 9,3$, $z_2 = Z(\hat{x}^{(s)} + 0,5\lambda^* \hat{s}) = 0,36$ и

$z_3 = Z(\hat{\mathbf{x}}^{(s-1)}) = 22$. Отсюда следует, что $\alpha = 30,6$ и $\beta = 48,8$. С помощью формулы (8.3.6) находим

$$\hat{\mathbf{x}}^* = \begin{bmatrix} 1,15 \\ 2,15 \end{bmatrix} + 0,47 \times 1,37 \begin{bmatrix} 0,835 \\ 0,548 \end{bmatrix} = \begin{bmatrix} 1,68 \\ 2,50 \end{bmatrix}.$$

Значение $Z(\mathbf{x})$ в точке $\hat{\mathbf{x}}^*$ равняется $Z(\hat{\mathbf{x}}^*) = 0,005$, и, следовательно, точка $\hat{\mathbf{x}}^*$ может рассматриваться по существу как граничная точка. В чрезмерно точном определении положения $\hat{\mathbf{x}}^*$ нет никакой необходимости. Если значение $T(\hat{\mathbf{x}}^*)$ в точке $\hat{\mathbf{x}}^*$, найденное с помощью (8.1.5), не удовлетворяет требованию допустимости (8.1.6), то осуществляется переход к новой точке $\hat{\mathbf{x}}^*$ путем перемещения вдоль s по направлению к $\hat{\mathbf{x}}^{(s)}$; при этом процедура пошагового перемещения в указанном направлении продолжается до тех пор, пока не будет выполнено условие $T(\hat{\mathbf{x}}^*) \leq \Phi^{(k)}$.

8.3.3. ПРОЦЕДУРА НАХОЖДЕНИЯ ДОПУСТИМЫХ И ПОЧТИ ДОПУСТИМЫХ ТОЧЕК

Процедура, в результате которой удается получить либо допустимую, либо почти допустимую точку, выглядит следующим образом:

1. Пусть $\hat{\mathbf{x}}^{(0)} = \mathbf{x}_i^{(k)}$ есть недопустимая точка в E^n , $\Phi^{(k)}$ — значение критерия допустимости, найденное с помощью соотношения (8.1.1) на k -м этапе процедуры оптимизационного поиска. Пусть $t=0,05$ $\Phi^{(k)}$ есть параметр размера исходного многогранника, ассоциированного с минимизацией $T(\mathbf{x})$, начиная из точки $\hat{\mathbf{x}}^{(0)}$. С помощью процедуры, описание которой дано в подразд. 8.3.1, и определяют $(n+1)$ вершин $\hat{\mathbf{x}}_i^{(0)}$ ($i = 1, \dots, n+1$), требуемых для выполнения начального шага в связи с минимизацией $T(\mathbf{x})$. С помощью соотношения (8.1.5) вычисляется значение $T(\mathbf{x})$ в каждой из $(n+1)$ вершин, т. е. находится $T(\hat{\mathbf{x}}_i^{(0)})$ при $i = 1, \dots, n+1$.

2. При $\alpha = 1$, $\beta = 0,5$ и $\gamma = 2$ с помощью процедуры Нелдера и Мида минимизируется $T(\mathbf{x})$. В конце каждого s -го этапа наименьшее из значений $T(\hat{\mathbf{x}}_i^{(s)})$ ($i = 1, \dots, n+1$), т. е. $T(\hat{\mathbf{x}}_i^{(s)})$, сравнивается с $\Phi^{(k)}$.

3. Если $T(\hat{\mathbf{x}}_i^{(s)}) \geq \Phi^{(k)}$, найденная точка является либо допустимой, либо почти допустимой. Если $T(\hat{\mathbf{x}}_i^{(s)}) > 0$, недопустимая точка $\mathbf{x}_i^{(k)}$ заменяется на точку $\hat{\mathbf{x}}_i^{(s)}$; при этом точка $\hat{\mathbf{x}}_i^{(k)} = \hat{\mathbf{x}}_i^{(s)}$ окажется ли-

бо допустимой, либо почти допустимой и минимизация $T(\mathbf{x})$ заканчивается. Если же $T(\hat{\mathbf{x}}_l^{(s)}) = 0$ и $m = 0$, переходят к шагу 7, описание которого дано ниже.

4. Если $T(\hat{\mathbf{x}}_l^{(s)}) \geq \Phi^{(k)}$, вычисляется величина

$$\mathcal{A}^{(s)} = \frac{1}{n+1} \left\{ \sum_{i=1}^{n+1} [T(\hat{\mathbf{x}}_i^{(s)}) - T(\hat{\mathbf{x}}_{n+2}^{(s)})]^2 \right\}^{1/2},$$

где $T(\hat{\mathbf{x}}_{n+2}^{(s)})$ — значение $T(\mathbf{x})$ в центре тяжести многогранника на s -м этапе минимизации $T(\mathbf{x})$.

5. Если $\mathcal{A}^{(s)} > 10^{-7}$, возвращаются к шагу 2 и продолжается минимизация $T(\mathbf{x})$ на $(s+1)$ -м этапе.

6. Если $\mathcal{A}^{(s)} \leq 10^{-7}$, деформируемый многогранник близок к вырождению в точку, тогда как допустимую (или почти допустимую) точку так и не удалось найти. При $\mathcal{A}^{(s)} \leq 10^{-7}$ и при наличии большого числа нелинейных ограничений (как в виде равенств, так и в виде неравенств), формирующих структуру $T(\mathbf{x})$ и, следовательно, определяющих значения $T(\mathbf{x})$ в вершинах $\hat{\mathbf{x}}_i^{(s)}$, в ходе выполнения процедуры Нелдера и Мида могут возникнуть серьезные затруднения. В этом случае $T(\mathbf{x})$ оказывается весьма сложной функцией в недопустимой области E^n . Пусть $\hat{\mathbf{x}}_l^{(s)}$ есть вершина, соответствующая наименьшему значению $T(\mathbf{x})$, найденному с помощью процедуры Нелдера и Мида. Вместо того чтобы прекратить поиск в точке $\hat{\mathbf{x}}_l^{(s)}$, потеряв возможность определения допустимой или почти допустимой точки, алгоритм продолжает работу, реализуя поиск вдоль каждого из направлений, параллельных осям координат, и осуществляется поиск минимума $T(\mathbf{x})$ по следующей схеме. Пусть $\hat{\mathbf{x}}_j^* (j = 1, \dots, n)$ — точки, соответствующие наименьшим среди всех значений $T(\mathbf{x})$, найденным на траекториях, параллельных осям координат. Начиная из $\hat{\mathbf{x}}_l^{(s)}$, определяется $\hat{\mathbf{x}}_1^*$, соответствующее минимальному значению $T(\mathbf{x})$ при перемещении в направлении, параллельном координатной оси x_1 ; затем, начиная из $\hat{\mathbf{x}}_1^*$, определяется $\hat{\mathbf{x}}_2^*$ и т. д. Этот процесс продолжается до тех пор, пока не будут определены $\hat{\mathbf{x}}_j^*$ для всех n значений индекса j . Используемая при этом методика заключается в определении такого интервала I_0 , который содержал бы точку с минимальным значением $T(\mathbf{x})$ в выбранном направлении. После этого осуществляется одномерный поиск методом золотого сечения [2]. Этот поиск продолжается до тех пор, пока длина интервала, содержащего точку $\hat{\mathbf{x}}_i^*$, не уменьшится до $0,01 \Phi^{(k)}$. Цель такого одномерного поиска состоит в том,

чтобы найти новую точку, не совпадающую с $\hat{x}_i^{(s)}$, и повторить вычислительные операции, начиная с шага 1, при больших размерах исходного многогранника.

В конце каждого одномерного поиска в направлениях, параллельных осям координат, производится проверка с целью выяснения, выполняется ли для нового значения $T(x_i^*)$ условие $T(\hat{x}_i^{(s)}) \leq \Phi^{(k)}$. Если это условие выполняется, $x_i^{(k)}$ заменяется на $\hat{x}_i^{(s)}$ и процедура минимизации $T(x)$ заканчивается. Если после проведения поиска в каждом из координатных направлений допустимую (или почти допустимую) точку определить все же не удалось, алгоритм реализует переход к шагу 1 и все операции вновь повторяются по схеме, предусмотренной методом Неллера и Мида. При этом в качестве начальной вновь выбирается точка \hat{x}_n^* , т. е. точка, в которой $T(x)$ принимает минимальное значение в ходе перемещения в направлении, параллельном n -й оси координат. Если в результате трехкратного выполнения всей процедуры от шага 1 до шага 6 допустимую или почти допустимую точку найти не удается, минимизационный поиск прекращается и квалифицируется как безрезультативный.

7. Если $T(\hat{x}_i^{(s)}) = T(x_i^{(k)}) = 0$, то, прежде чем вернуться к процедуре минимизации $f(x)$, осуществляется интерполяция, описание которой дано в подразд. 5.4.2. В результате интерполяции добиваются того, чтобы точка $x_i^{(k)} = \hat{x}_i^{(s)}$ не была слишком удалена от границ, задаваемых теми ограничениями, которые были нарушены непосредственно перед тем, как была найдена точка $\hat{x}_i^{(s)}$.

8.4. НАЧАЛО И ОКОНЧАНИЕ ПОИСКА

В данном разделе приводится описание процедуры начала поиска с целью минимизации $f(x)$. Напомним, что при минимизации $T(x)$ использовались все $(n + 1)$ вершин многогранника, где n — суммарное число переменных (как независимых, так и зависимых) задачи (8.0.1), тогда как при минимизации $f(x)$ рассматриваются лишь $(r + 1)$ вершин, где $r = (n - m)$ есть число «степеней свободы» целевой функции. Если $m = 0$ [т. е. если задача (8.0.1) не содержит ограничений в виде равенств], то $r = n$, и при поиске минимума $f(x)$ мы имеем такое же число степеней свободы, как и при минимизации $T(x)$.

Приступая к поиску минимума целевой функции $f(x)$ с помощью алгоритма скользящего допуска, мы должны знать начальную точку $x^{(0)}$, t , $\Phi^{(0)}$ и r . Для того чтобы поиск минимума $f(x)$ был начат при правильном выборе размеров деформируемого многогранника,

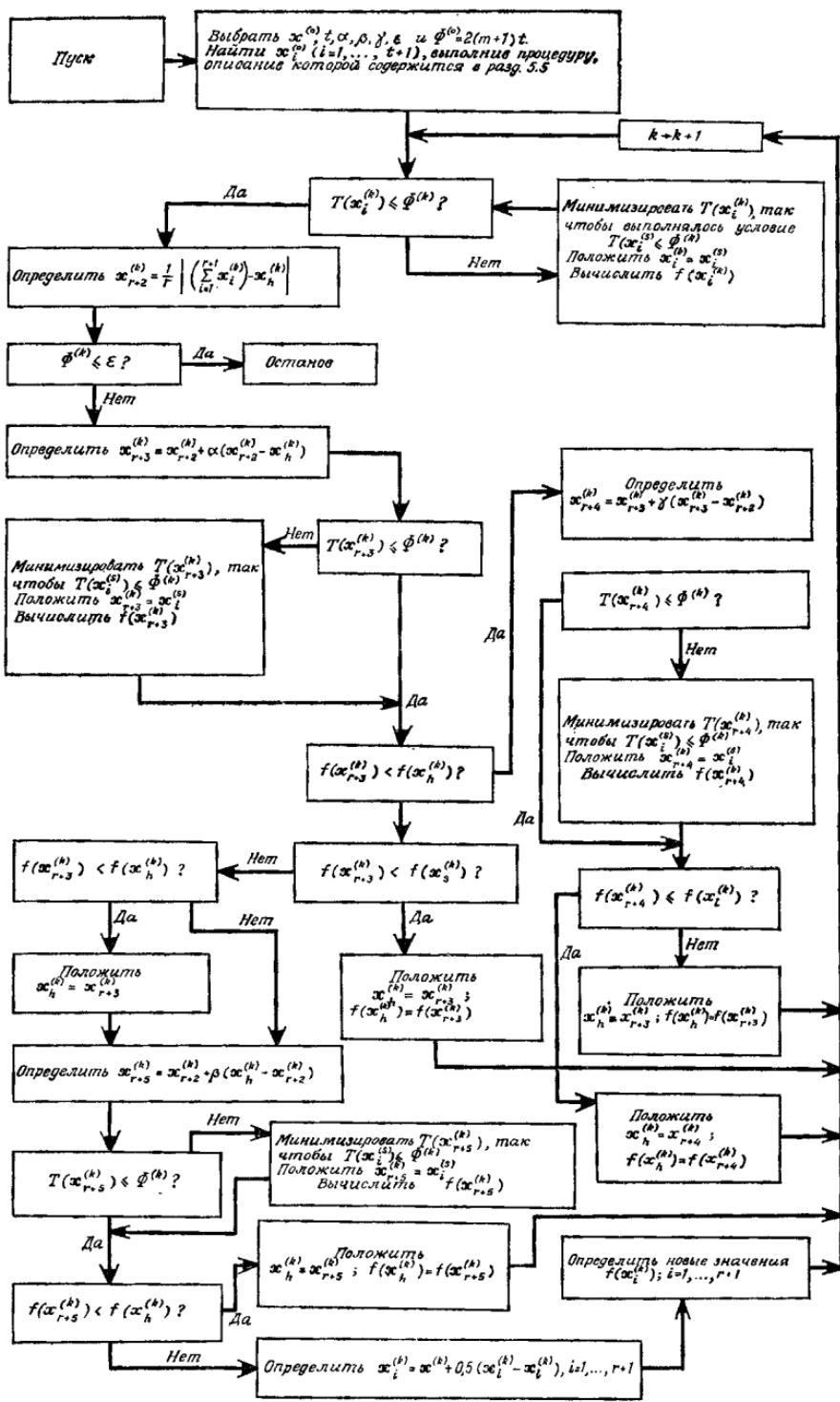
параметр t должен быть задан как функция интервала ожидаемых вариаций переменных \mathbf{x} рассматриваемой задачи. Как правило, оказываются известными нижняя и верхняя границы изменения \mathbf{x} ; тогда для оценки наиболее рационального значения t можно воспользоваться следующей формулой:

$$t = \min \left\{ \left[-\frac{0.2}{n} \sum_{i=1}^n (U_i - L_i) \right], (U_1 - L_1), \dots, (U_n - L_n) \right\}, \quad (8.4.1)$$

где $(U_i - L_i)$ — разность между верхним и нижним предельными значениями, которые может принимать переменная x_i . Таким образом, если U_i и L_i ($i = 1, \dots, n$) известны, установить подходящее исходное значение t удается относительно легко.

В рамках рассматриваемого алгоритма локальные минимумы исключаются проще, если исходный многогранник охватывает весьма большую область пространства решений. Стратегия алгоритма не зависит ни от локальных свойств $f(\mathbf{x})$, ни от сочетаний характеристик $f(\mathbf{x})$ и функций, задающих ограничения (с противоположной ситуацией сталкиваются, например, при использовании методов проекции градиента). На каждом этапе оптимизационного поиска методом скользящего допуска информация, необходимая для реализации очередного перемещения в пространстве решений, получается за счет рассмотрения $(r + 1)$ вершин деформируемого многогранника в E^n . Таким образом, алгоритм скользящего допуска имеет весьма важное преимущество, которое заключается в том, что в самом начале поиска удается получить существенный объем информации относительно $f(\mathbf{x})$ за счет рассмотрения большого числа вершин деформируемого многогранника. При этом увеличивается вероятность того, что некоторые из найденных $x_i^{(k)}$ будут соответствовать локальному оптимуму, лучшему любого другого локального оптимума. Практика решения задач, характеризующихся наличием многочисленных локальных оптимумов, подтверждает эффективность алгоритма скользящего допуска при исключении из рассмотрения побочных локальных оптимумов. Разумеется, никакая из реализуемых на ЭВМ вычислительных процедур не может гарантировать глобальность найденного экстремума при решении задачи, целевая функция которой обладает множеством локальных экстремумов.

Представляется также целесообразным выбирать в качестве точки $\mathbf{x}^{(0)}$, относительно которой строится исходный многогранник, допустимую или почти допустимую точку. Если исходный многогранник строить относительно точки, лежащей далеко за пределами допустимой области, то придется производить замену $r + 1$ вершин на другие вершины, расположенные ближе к границе допустимой области.



Ф и г. 8.4.1. Блок-схема алгоритма скользящего допуска.

Процедура отыскания вершин $x_i^{(0)} (i = 1, \dots, r+1)$ реализуется по следующей схеме. С помощью (8.1.1) вычисляется $\Phi^{(0)} = 2(m+1)t$ и находится значение $T(x)$ в исходной точке $x^{(0)}$. Если $T(x^{(0)}) < \Phi^{(0)}$, то $x^{(0)}$ является либо допустимой, либо почти допустимой точкой, и исходные вершины $x_i^{(0)} (i = 1, \dots, r+1)$ находятся с помощью процедуры, описание которой дано в подразд. 8.3.1. Если $T(x^{(0)}) > \Phi^{(0)}$, то $T(x)$ минимизируется до тех пор, пока не будет найдена допустимая или почти допустимая точка x ; именно эта точка и выбирается в качестве базовой при построении исходного многогранника.

Работа алгоритма заканчивается при двух обстоятельствах:

1. Когда $\Phi^{(k)} \leq \varepsilon$. В этом случае поиск считается завершенным и квалифицируется как успешный (именно такая ситуация возникает в подавляющем большинстве случаев).

2. Когда с помощью процедуры, описание которой приведено в разд. 8.3, не удается найти допустимую или почти допустимую точку. В этом случае поиск заканчивается, производится замена стартовой точки $x^{(0)}$ и/или осуществляется переход к другому набору значений параметров α, β, γ, t и ε . В обычных условиях рекомендуется принимать $\alpha = 1, \beta = 0,5, \gamma = 2$ и $\varepsilon = 10^{-5}$.

На фиг. 8.4.1 представлена блок-схема, дающая общее представление о логической схеме алгоритма скользящего допуска. Список соответствующих машинных команд на языке ФОРТРАН приведен в приложении Б.

Пример 8.4.1. Метод скользящего допуска

В качестве примера рассмотрим следующую задачу:

$$\text{минимизировать } f(x) = 4x_1 - x_2^2 - 12$$

при ограничениях

$$h_1(x) = 25 - x_1^2 - x_2^2 = 0,$$

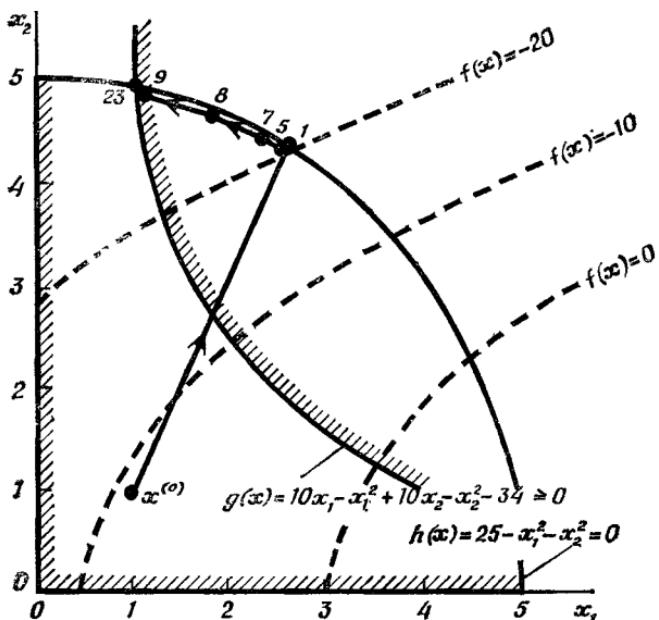
$$g_2(x) = 10x_1 - x_1^2 + 10x_2 - x_2^2 - 34 \geq 0,$$

$$g_3(x) = x_1 \geq 0,$$

$$g_4(x) = x_2 \geq 0.$$

Целевая функция и ограничения этой задачи изображены графически на фиг. 6.0.1 и П.8.4.1. В качестве начальной (стартовой) точки возьмем недопустимую точку $x_0^{(0)} = [1 \ 1]^T$, а начальную величину параметра размера многогранника положим равным $t = 0,30$. Тогда, согласно (8.1.1), для критерия допуска на старте будем иметь

$$\Phi^{(0)} = 2(m+1)t = 2 \times (1+1) \times 0,30 = 1,20.$$



Фиг. П.8.4.1. Траектории поиска с помощью алгоритма скользящего допуска (числа указывают порядковые номера этапов оптимизационного процесса).

В точке $\mathbf{x}_0^{(0)} = [1 \ 1]^T$ значения функций, задающих нарушенные ограничения, таковы:

$$h_1(\mathbf{x}^{(0)}) = 23 \text{ и } g_2(\mathbf{x}^{(0)}) = -16,$$

так что значение $T(\mathbf{x}_0^{(0)})$, вычисленное с помощью соотношения (8.1.5), равняется

$$T(\mathbf{x}_0^{(0)}) = [(23)^2 + (-16) \times 2]^{1/2} = 28,02.$$

Мы видим, что $T(\mathbf{x}_0^{(0)}) > \Phi^{(0)}$. Следовательно, на первом цикле работы алгоритма осуществляется поиск почти допустимого вектора.

Чтобы начать процедуру минимизации $\hat{T}(\mathbf{x})$, построим равносторонний треугольник, каждая сторона которого равняется 0,06:

Вершина	\hat{x}_1	\hat{x}_2
1	1,000	1,000
2	1,057	1,015
3	1,015	1,057

Теперь методом деформируемого многогранника (обсуждение данного метода см. в разд. 4.2) осуществляется минимизация

$$T(\hat{\mathbf{x}}) = [h_1^2(\hat{\mathbf{x}}) + \mathcal{U}_2 g_2^2(\hat{\mathbf{x}}) + \mathcal{U}_3 g_3^2(\hat{\mathbf{x}}) + \mathcal{U}_4 g_4^2(\hat{\mathbf{x}})]^{1/2},$$

где \mathcal{U}_i — оператор Хевисайда. В результате получается последовательность вершин, указанных в приведенной ниже таблице:

Новая вершина	\hat{x}_1	\hat{x}_2	Удовлетворяются ли ограничивающие условия в виде неравенств?		
			$g_2(\hat{\mathbf{x}})$	$g_3(\hat{\mathbf{x}})$	$g_4(\hat{\mathbf{x}})$
4	1,110	1,110	Нет	Да	Да
5	1,072	1,221	»	»	»
6	1,243	1,381	»	»	»
7	1,253	1,683	»	»	»
8	1,600	2,154	»	»	»
7	1,794	2,993	Да	»	»
8	1,426	1,918	Нет	»	»
9	2,584	4,356	Да	»	»

(Напомним, что $\mathcal{U}_i = 0$, если соответствующее ограничивающее условие удовлетворено.) В точке $\hat{\mathbf{x}}_9^{(0)}$ удовлетворяются все ограничения-неравенства, так что единственным нарушенным ограничением в этой точке является ограничение $h(\hat{\mathbf{x}}_9^{(0)}) = 0,648$. Отсюда следует, что

$$T(\hat{\mathbf{x}}_9^{(0)}) = 0,648 < \Phi^{(0)} \text{ и } \mathbf{x}_0^{(1)} = \hat{\mathbf{x}}_9^{(0)}.$$

Теперь алгоритм переключается на минимизацию целевой функции $f(\mathbf{x})$. В точке $\mathbf{x}_0^{(1)} = [2,584 \ 4,356]^T$ имеем $f(\mathbf{x}_0^{(1)}) = 2,063$. Далее строится новый симплекс при следующих координатах вершин:

Новая вершина	x_1	x_2
0	2,584	4,356
1	2,559	4,331
2	2,617	4,347
3	2,574	4,389

На этом этапе оптимизационного поиска значение критерия допуска $\Phi^{(1)}$, найденное с помощью (8.1.1), равняется

$$\Phi^{(1)} = \min \{1,20; 0,0447\} = 0,0447, \text{ т. е.}$$

$\Phi^{(1)} < T(\mathbf{x}_3^{(1)})$. Следовательно, необходимо вновь перейти к поиску почти допустимой точки, начиная с $\hat{\mathbf{x}}_0^{(2)} = \mathbf{x}_3^{(1)} = [2,574 \quad 4,389]^T$.

Все последующие этапы поиска в точности воспроизводят описанные выше процедуры, и мы их подробно не рассматриваем. Результаты

Таблица П.8.4.1

Этап	x_1	x_2	$f(\mathbf{x})$	Φ	$h_1(\mathbf{x})$
1	2,574	4,389	-20,966	$4,47 \cdot 10^{-2}$	$-8,94 \cdot 10^{-1}$
2	2,600	4,274	-19,867	$7,13 \cdot 10^{-3}$	$-3,01 \cdot 10^{-2}$
3	2,599	4,270	-19,836	$6,33 \cdot 10^{-3}$	$6,68 \cdot 10^{-3}$
4	2,578	4,284	-20,041	$1,22 \cdot 10^{-3}$	$-6,74 \cdot 10^{-4}$
5	2,549	4,301	-20,029	$6,33 \cdot 10^{-3}$	$6,33 \cdot 10^{-4}$
6	2,457	4,355	-21,137	$6,33 \cdot 10^{-3}$	$-2,06 \cdot 10^{-4}$
7	2,278	4,451	-22,703	$6,33 \cdot 10^{-3}$	$-5,42 \cdot 10^{-3}$
8	1,814	4,659	-26,457	$6,22 \cdot 10^{-3}$	$-6,05 \cdot 10^{-3}$
9	1,059	4,886	-31,636	$6,33 \cdot 10^{-3}$	$4,79 \cdot 10^{-3}$
10	1,015	4,896	-31,916	$6,33 \cdot 10^{-3}$	$-3,64 \cdot 10^{-3}$
11	1,015	4,896	-31,916	$6,33 \cdot 10^{-3}$	$-3,64 \cdot 10^{-3}$
12	1,003	4,898	-31,983	$6,33 \cdot 10^{-3}$	$-1,77 \cdot 10^{-3}$
13	1,003	4,898	-31,983	$6,33 \cdot 10^{-3}$	$-1,77 \cdot 10^{-3}$
14	1,003	4,898	-31,983	$6,33 \cdot 10^{-3}$	$-1,77 \cdot 10^{-3}$
15	1,002	4,899	-31,989	$2,24 \cdot 10^{-3}$	$-2,03 \cdot 10^{-4}$
16	1,001	4,899	-31,993	$1,11 \cdot 10^{-3}$	$-1,43 \cdot 10^{-3}$
17	1,001	4,899	-31,992	$2,43 \cdot 10^{-4}$	$-7,01 \cdot 10^{-4}$
18	1,001	4,899	-31,991	$2,43 \cdot 10^{-4}$	$6,59 \cdot 10^{-5}$
19	1,001	4,898	-31,991	$3,95 \cdot 10^{-5}$	$7,69 \cdot 10^{-5}$
20	1,001	4,898	-31,992	$3,95 \cdot 10^{-5}$	$-3,34 \cdot 10^{-5}$
21	1,001	4,898	-31,992	$3,95 \cdot 10^{-5}$	$-3,50 \cdot 10^{-5}$
22	1,001	4,898	-31,992	$1,00 \cdot 10^{-5}$	$-3,50 \cdot 10^{-5}$
23	1,001	4,989	-31,992	$8,82 \cdot 10^{-6}$	$5,53 \cdot 10^{-6}$

результаты, полученные в ходе оптимизационного поиска с помощью алгоритма скользящего допуска, приведены в табл. П.8.4.1 (эти результаты округлены до третьего десятичного знака). Мы видим, что на десяти последних этапах минимизирующие поправки для \mathbf{x} и $f(\mathbf{x})$ незначительны. Однако в результате реализации этих этапов достигается более строгое выполнение ограничения-равенства. Траектория поиска изображена на фиг. П.8.4.1. Следует обратить внимание на то, как в процессе поиска происходит постепенный переход к точкам, обеспечивающим более строгое выполнение условия $h_1(\mathbf{x}) = 0$.

8.5. МЕТОДЫ РЕШЕНИЯ ЗАДАЧ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ С ЗОНАЛЬНОЙ НЕОПРЕДЕЛЕННОСТЬЮ

Известны примеры задач нелинейного программирования, в которых целевая функция и, возможно, некоторые из ограничений в некоторых зонах (или областях) E^n не определены. Чтобы не допустить в таких случаях преждевременного прекращения поиска, необходимо переформулировать задачу, введя запрет на те векторы \mathbf{x} , для которых хотя бы одна из функций $f(\mathbf{x})$, $h_i(\mathbf{x})$ и $g_i(\mathbf{x})$ однозначно не определена или теряет физический смысл. Для иллюстрации этой идеи рассмотрим следующую задачу:

$$\begin{aligned} \text{минимизировать } f(\mathbf{x}) = & (x_1^2 + x_2^2 - 4x_2 - 45)^{1/2} + \\ & + [\ln x_1 (x_2 - 2)]^2, \quad \mathbf{x} \in E^n, \end{aligned}$$

при ограничении

$$g_1(\mathbf{x}) = 64 - x_1^2 - x_2^2 \geqslant 0. \quad (8.5.1)$$

Поскольку квадратный корень из отрицательного числа и логарифм отрицательного числа не имеют смысла, целевая функция задачи (8.5.1) не определена в точках, для которых

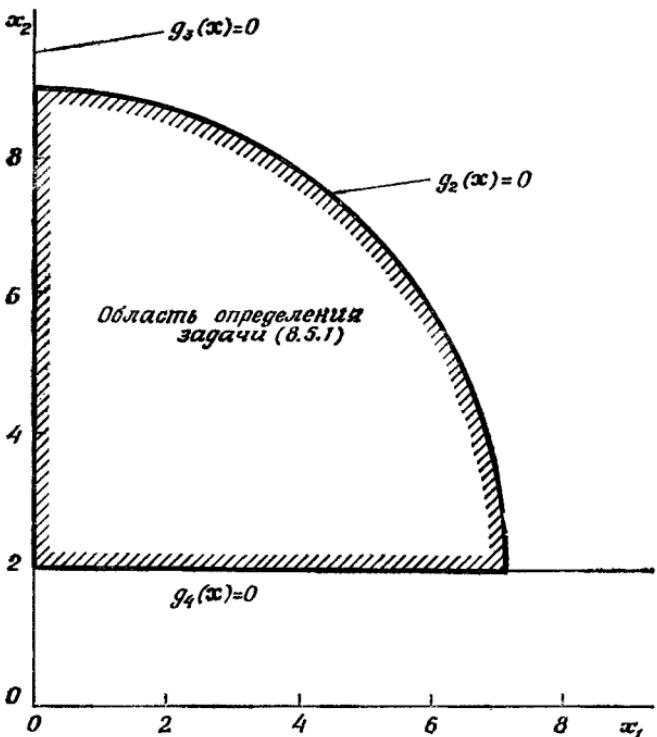
$$(x_1^2 + x_2^2 - 4x_2 - 45) < 0, \quad x_1 < 0 \text{ или } (x_2 - 2) < 0.$$

В области, показанной на фиг. 8.5.1 заштрихованной кромкой по внутренней стороне контура, образованного дугой окружности, которая задается уравнением $g_2(\mathbf{x}) = x_1^2 + x_2^2 - 4x_2 - 45 = 0$, и отрезками прямых, описываемых уравнениями $g_3(\mathbf{x}) = x_1 = 0$ и $g_4(\mathbf{x}) = -x_2 - 2 = 0$, все функции, фигурирующие в задаче (8.5.1), определены. Обратим внимание на то, что значительная часть допустимой области, определяемая условием $g_1(\mathbf{x}) \geqslant 0$, по указанным выше причинам исключается из рассмотрения. Другими примерами, в которых некоторые из функций, фигурирующих в формулировке задачи, зонально не определены, являются задачи 4, 9 и 17 в приложении А.

Если алгоритм скользящего допуска используется для решения задачи нелинейного программирования в общей постановке, то эту задачу необходимо привести к такому виду, когда заведомо устраняется возможность манипулирования с точками \mathbf{x} , лежащими за пределами области определения рассматриваемой задачи. Один из способов устранения из рассмотрения зон, в которых задача не определена, заключается в надлежащей замене переменных. Например, замена переменных x_1 и x_2 в задаче (8.5.1) по формулам

$$\begin{aligned} x_1 &= 7 + e^{z_1}, \\ x_2 &= 2 + e^{z_2}, \end{aligned}$$

где \mathbf{z} — новый вектор, приводит к следующей задаче:



Ф и г. 8.5.1. Графическое изображение области определения для задачи (8.5.1).

минимизировать $f(\mathbf{z}) = (e^{2z_1} + 14e^{z_1} + e^{2z_2})^{1/2} + [\ln(8 + e^{z_1}) e^{z_2}]^2$,
 $\mathbf{z} \in E^n$,

при ограничении

$$g_1(\mathbf{z}) = 11 - 14e^{z_1} - e^{2z_1} - 4e^{z_2} - e^{2z_2} \geq 0,$$

где $f(\mathbf{z})$ определена теперь для любого вектора \mathbf{z} . В результате преобразования (8.5.1) в (8.5.2) формулировка задачи оказывается полностью корректной и допускает применение алгоритма скользящего допуска в его обычном виде. После того как будет найдено оптимальное решение \mathbf{z}^* задачи (8.5.2), значение \mathbf{x}^* нетрудно вычислить, используя соотношения, с помощью которых производилась замена переменных.

Когда в некоторых зонах области изменения \mathbf{x} не определена *лишь* целевая функция, для каждого члена в выражении для $f(\mathbf{x})$, приводящего к неопределенности в указанном выше смысле, необходимо ввести вспомогательное ограничение в виде неравенства

$$g_{p+i}(\mathbf{x}): F_{p+i}(\mathbf{x}) - \Phi^{(k)} \geq 0, \quad i = p+1, \dots, p+q, \quad (8.5.3)$$

где $F_{p+i}(\mathbf{x})$ — i -я компонента в структуре целевой функции, за счет которой $f(\mathbf{x})$ теряет определенность в отдельных зонах E^n , а q —

число таких компонент в выражении для $f(\mathbf{x})$. Дополнительные ограничения вида (8.5.3) добавляются к p исходным ограничениям задачи нелинейного программирования, и «уточненная» задача решается обычным способом. Путем вычитания $\Phi^{(k)}$ из $F_{p+i}(\mathbf{x})$ ($i = p+1, \dots, p+q$) мы исключаем из рассмотрения те векторы \mathbf{x} , для которых $F_{p+i}(\mathbf{x}) < 0$ в случаях, когда выполняется условие (8.1.6). Таким образом мы добиваемся того, чтобы неравенство (8.1.6) выполнялось лишь при условии $F_{p+i}(\mathbf{x}) \geq 0$ ($i = p+1, \dots, p+q$).

Переходя к более подробному рассмотрению, допустим вначале, что одна из компонент $F_{p+i}(\mathbf{x})$ в текущей точке \mathbf{x} такая, что $-F_{p+i}(\mathbf{x}) > 0$, т. е. $F_{p+i}(\mathbf{x}) < 0$. Поскольку по определению $\Phi^{(k)} \geq 0$, из (8.5.3) следует, что $g_{p+i}(\mathbf{x}) = F_{p+i}(\mathbf{x}) - \Phi^{(k)} < 0$, т. е. в текущей точке \mathbf{x} функция $g_{p+i}(\mathbf{x})$ принимает отрицательное значение. Предположим, кроме того, что для всех остальных ограничений $h_i(\mathbf{x}) = 0$ ($i = 1, \dots, m$), $g_i(\mathbf{x}) \geq 0$ ($i = m+1, \dots, p+i-1, p+i+1, \dots, p+q$). Тогда будем иметь $T(\mathbf{x}) = +[(-F_{p+i}(\mathbf{x}) - \Phi^{(k)})^2]^{1/2} = F_{p+i}(\mathbf{x}) + \Phi^{(k)}$. Поскольку $T(\mathbf{x}) > \Phi^{(k)}$, условие (8.1.6) не выполняется, и, следовательно, необходимо минимизировать $T(\mathbf{x})$ до тех пор, пока не будет выполняться условие $F_{p+i}(\mathbf{x}) \geq 0$.

В силу определения $T(\mathbf{x})$ очевидно, что для любой компоненты F_{p+i} , удовлетворяющей условию $0 \leq F_{p+i}(\mathbf{x}) < \Phi^{(k)}$, неравенство (8.1.6) выполняется даже в том случае, когда $g_{p+i}(\mathbf{x}) < 0$. При $F_{p+i}(\mathbf{x}) \geq \Phi^{(k)}$ имеет место неравенство $g_{p+i}(\mathbf{x}) > 0$, что никак не оказывается на поведении $T(\mathbf{x})$. Аналогичные рассуждения справедливы и тогда, когда в точке \mathbf{x} оказываются нарушенными сразу несколько ограничений в виде равенств и/или в виде неравенств, так как в структуру $T(\mathbf{x})$ входят все нарушаемые ограничения. Добавление к исходным ограничениям общей задачи нелинейного программирования (8.0.1) ограничений (8.5.3) приводит к следующему выражению для $T(\mathbf{x})$:

$$T(\mathbf{x}) = + \left[\sum_{i=1}^m h_i^2(\mathbf{x}) + \sum_{i=m+1}^{p+q} U_i g_i^2(\mathbf{x}) \right]^{1/2}. \quad (8.5.4)$$

Формула (8.1.5) используется при определении $T(\mathbf{x})$ в тех случаях, когда все функции, фигурирующие в формулировке задачи (8.0.1), определены для всех значений $\mathbf{x} \in E^n$; в случае же, когда некоторые из упомянутых выше функций зонально не определены (т. е. определены не для всех $\mathbf{x} \in E^n$), следует использовать формулу (8.5.4).

Чтобы проиллюстрировать описанную выше процедуру, преобразуем задачу (8.5.1) в следующую задачу:

$$\text{минимизировать } f(\mathbf{x}) = (x_1^2 + x_2^2 - 4x_2 - 45)^{1/2} + [\ln x_1 (x_2 - 2)]^2 \quad (8.5.5)$$

при ограничениях

$$g_1(\mathbf{x}) = 64 - x_1^2 - x_2^2 \geq 0,$$

$$g_2(\mathbf{x}) = x_1^2 + x_2^2 - 4x_2 - 45 - \Phi^{(k)} \geq 0,$$

$$g_3(\mathbf{x}) = x_1 - \Phi^{(k)} \geq 0,$$

$$g_4(\mathbf{x}) = x_2 - 2 - \Phi^{(k)} \geq 0.$$

Функции $g_i(\mathbf{x})$ ($i = 1, 2, 3, 4$) определены для любого $\mathbf{x} \in E^n$; однако, поскольку вычисление $f(\mathbf{x})$ возможно лишь при выполнении неравенства (8.1.6), отрицательные значения $(x_1^2 + x_2^2 - 4x_2 - 45)$, x_1 и $(x_2 - 2)$ оказываются недопустимыми. Путем использования рассмотренной вычислительной схемы были успешно решены, в частности, задачи 4, 9 и 17 из приложения А.

Следует, однако, еще раз отметить, что описанная выше процедура оказывается неприменимой в случае, когда зонально (т. е. для некоторых $\mathbf{x} \in E^n$) не определены одновременно и целевая функция, и функции, задающие ограничения задачи.

ЛИТЕРАТУРА

1. Paviani D., Himmelblau D. M., *Operations Res.*, 17 (1969).
2. Wilde D. J., *Optimum Seeking Methods*, Prentice-Hall, Inc., Englewood Cliffs, N. J., 1962, p. 32.

Глава 9

ОЦЕНКА ЭФФЕКТИВНОСТИ МЕТОДОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ

Целесообразность выбора того или иного из рассмотренных выше алгоритмов в целях его практического использования определяется эффективностью этого алгоритма при решении конкретного класса задач нелинейного программирования с помощью ЭВМ. Как и следовало ожидать, ни один из алгоритмов нелинейного программирования, рассмотрению которых посвящены гл. 6—8, не может считаться наилучшим по сравнению с другими алгоритмами при решении любых задач нелинейного программирования и при любых обстоятельствах. Прежде чем перейти к оцениванию эффективности конкретных алгоритмов нелинейного программирования при наличии ограничений, рассмотрим вопрос относительно критериев, которые следует иметь при этом в виду. После обсуждения используемых критериев эффективности приводятся результаты сравнительного анализа различных алгоритмов. Наконец, излагаются некоторые общие соображения и выводы, которые могут послужить своего рода ориентиром для тех, кто намерен применять алгоритмы нелинейного программирования на практике.

9.1. КРИТЕРИИ, ИСПОЛЬЗУЕМЫЕ ПРИ ОЦЕНКЕ ЭФФЕКТИВНОСТИ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ

На вопрос о том, какой из критериев оценки нелинейного программирования наилучший, можно дать лишь «комплексный» (т. е. развернутый) ответ. Ответ на этот вопрос в значительной степени зависит от того, к какому классу (или типу) относится рассматриваемая задача, от глубины предварительного анализа структурных особенностей задачи и от осведомленности пользователя относительно конфигурации и размеров допустимой области, ассоциированной с решаемой задачей. К числу важных критериев, используемых при оценивании «качества» того или иного алгоритма, относятся следующие:

- 1) время, необходимое для реализации серии вычислительных процедур (число операций и время их выполнения);

- 2) степень сложности задачи (размерность, число ограничений в виде неравенств, число ограничений в виде равенств);
- 3) точность решения по отношению к оптимальному значению x^* и (или) по отношению к $f(x^*)$, $h(x^*)$, $g(x^*)$ и $\nabla f(x^*)$;
- 4) простота практического использования алгоритма (время, необходимое для ввода исходных данных и записи функций в память ЭВМ);
- 5) простота машинной программы, реализующей рассматриваемый алгоритм.

Наконец, важное требование, предъявляемое к алгоритму нелинейного программирования, заключается в том, чтобы

- 6) он позволял решать задачи, которые представляют практический интерес (нельзя считать, что какой-либо из алгоритмов окажется эффективным при решении любой задачи, и требовать, чтобы он решал «патологические» задачи, т. е. задачи, которые специально «подобраны», чтобы создать трудности для данного алгоритма).

Следует отметить, что указанные выше критерии носят глобальный, а не «локальный» характер в том смысле, что они относятся ко всем этапам процесса оптимизации (начиная с первого и кончая последним), а не к каким-либо отдельным этапам оптимизационного поиска.

Наиболее широко используемыми критериями при оценивании относительной эффективности машинных программ, предназначенных для решения задач нелинейного программирования, являются количество вычислений значений функций, требуемое для получения оптимального решения той или иной тестовой задачи с заданной степенью точности, и (или) затраты машинного времени, сопряженные с решением рассматриваемой тестовой задачи. Количество вычислений значений функций показывает, сколько раз возникает необходимость в определении числовых значений целевой функции и (или) той или иной функции из совокупности функций, задающих ограничения задачи (а также числовых значений производных упомянутых выше функций), прежде чем будет найдено решение рассматриваемой задачи нелинейного программирования. Этот критерий является менее значимым в тех случаях, когда задача содержит большое число ограничений при сравнительно небольшом числе переменных, так как временные затраты, требуемые для определения точки, в которой следует вычислить значения $f(x)$ и (или) $h_i(x)$ и $g_i(x)$, нередко в несколько раз превышают временные затраты, связанные с нахождением самих значений указанных выше функций.

Таким образом, машинное время, требуемое для выполнения последовательности процедур оптимизации, является наиболее часто используемым критерием, позволяющим сравнивать эффективность различных алгоритмов нелинейного программирования. Вопрос о затратах машинного времени при использовании про-

грамм, реализующих различные вычислительные алгоритмы, исследовался Стоккером [1], Хольцманом [2] и Колвиллом [3]. Поскольку разные ЭВМ имеют различные характеристики и неодинаковое быстродействие, разработаны стандартные таймер-программы для перерасчета временных затрат при переходе от одного типа ЭВМ к другому. В результате можно проводить сравнение значений времени по единой (приведенной) шкале, которую

Таблица 9.1.1

Машинное время, требуемое для реализации стандартной таймер-программы

Автор	Тип ЭВМ	Программа	Время, с
Абади	IBM 7094	МОПГ	63,0
	CDC 6400	МОПГ (модифицированный)	20,5
Боас		Оптим	119,7
Колвилл	IBM 360/50	ПОП II	168,0
Дэвис	English Electric KDF9	Разн.	362,0
Кефарт	IBM 7094	ОГМОП	128,2
Мак-Кормик		МПБМ	599,0
Хольцман	IBM 360/50		140,0
Стоккер	CDC 6600		22,0

называют шкалой стандартизированного машинного времени. Типичная стандартная таймер-программа, составленная на языке ФОРТРАН Колвиллом (см. приложение Г), сводится к простому десятикратному повторению операции инверсии по отношению к матрице размерности 40×40 . В табл. 9.1.1 приведены временные затраты по реализации таймер-программы; указанные временные характеристики для различных типов ЭВМ заимствованы у различных авторов (Колвилл, Дэвис, Стоккер и др.).

Необходимо, однако, подчеркнуть, что сравнительные оценки стандартизованных временных затрат нельзя считать достаточно точными показателями при оценивании эффективности различных машинных программ. Оказывается, что при решении одной и той же тестовой задачи на различных ЭВМ в оценках стандартизованных временных затрат имеют место значительные расхождения. Для иллюстрации в табл. 9.1.2 дано сравнение стандартизованных временных затрат при решении тестовых задач 10, 11, 15 и 19 из приложения А (с помощью метода МПБМ); содержащиеся в этой таблице данные заимствованы у Колвилла и Стоккера (оба автора пользовались одной и той же программой). Данные, приведенные в табл. 9.1.2, показывают, что сравнение

эффективности машинных программ на основе оценок затрат машинного времени по приведенной (стандартизированной) шкале вводит в некоторой степени в заблуждение. «Корректной» стандартной таймер-программой могла бы быть лишь такая программа, в которой принимались бы в расчет полиморфные факторы вычислительной логики, структурные и емкостные характеристики памяти ЭВМ, конфигурационные и временные характеристики вычислительного комплекса (центрального процессора и периферийных устройств), объем распечатываемой информации и т. д., имею-

Таблица 9.1.2

Сравнение стандартизованных временных затрат при решении задач 10, 11, 15 и 19 приложения А

Метод	Номер задачи			
	10	11	15	19
Колвилла	0,0162	0,0282	0,1511	0,238
Стоккера	0,127	0,048	0,253	0,719

щие место в различных типах ЭВМ и разных видах программного обеспечения.

Чистое машинное время¹⁾, требуемое для решений той или иной задачи, существенно зависит от того, с какой точностью нужно определить оптимальное решение, а также от величины допуска при выполнении ограничивающих условий на этапах, непосредственно предшествующих завершению оптимизационного поиска. Чтобы критерий, на основании которого производится сравнение эффективности используемых при программировании алгоритмов, «работал» однозначно, при решении тестовой задачи необходимо стремиться к достижению одной и той же степени точности. Вообще говоря, не исключено, что машинная программа, позволяющая решить задачу быстро, но в грубом приближении, окажется предпочтительнее какой-либо другой машинной программы, с помощью которой та же самая задача решается гораздо дольше, хотя и с большей точностью определения оптимальных значений x и $f(x)$ и (или) выполнения ограничивающих условий $h_t(x) = 0$ и $g_t(x) \geq 0$. К сожалению, критерии останова, используемые в разных программах, различны, и, следовательно, точность решения одной и той же задачи с помощью разных программ также оказывается неодинаковой. В разд. 9.3 вместо попытки унифицировать критерии завершения работы программы (унифи-

1) «Чистое» машинное время не включает время на подготовку, считывание и распечатку, время задержки данных в системе разделения времени в периферийном контуре и т. п.

кация такого рода сильно отразилась бы на всей схеме вычислительного процесса) проводится сравнение программ другим способом (по методике, предложенной автором данной книги).

Другим важным критерием, используемым при сравнении машинных программ, является показатель, характеризующий степень простоты (или сложности) подготовки задачи к решению пользователем. Хотя этот аспект анализа и носит в известной степени качественный характер, два существенных фактора, влияющих на решение вопроса о целесообразности использования той или иной программы, требуют особого рассмотрения. Один из них — это фактор, обусловленный возможностью возникновения ошибок в процессе подготовки данных человеком («вручную»). Программы, требующие выполнения громоздких и трудных подготовительных операций, более подвержены влиянию ошибок человека, нежели программы, подготовительные операции для которых просты. Другой фактор является чисто экономическим. Затраты, связанные с решением задачи математического программирования, складываются из затрат на подготовительные операции и стоимости машинного времени, расходуемого на решение задачи. Таким образом, решение задачи с использованием менее эффективной программы, которая, однако, не требует больших трудозатрат на подготовительном этапе, может обойтись дешевле, нежели решение той же самой задачи с помощью высокоеффективной программы, требующей большого объема подготовительных работ. Такого рода экономические расчеты с трудом поддаются сравнению, поскольку упомянутые выше расходы сильно варьируются в зависимости от конкретной ситуации; тем не менее экономические соображения представляются весьма существенными.

Некоторые машинные программы содержат большое число параметров, удачный выбор которых повышает эффективность применяемого алгоритма. Все упомянутые выше разработчики программ предлагают придавать входящим в программы параметрам и константам их средние значения; однако при этом максимально возможная эффективность программы при решении той или иной конкретной задачи может оказаться недостигнутой. В некоторых программах содержится настолько большое число переменных параметров, что в итоге определяющий программу вычислительный алгоритм можно рассматривать как функцию некоторого набора параметров. Если программа оказывается малоэффективной при каком-либо конкретном наборе значений параметров, ее можно подправить путем перехода к другому набору значений этих параметров. Наличием чрезвычайно большого числа подобных параметров особенно отличается ПОП II.

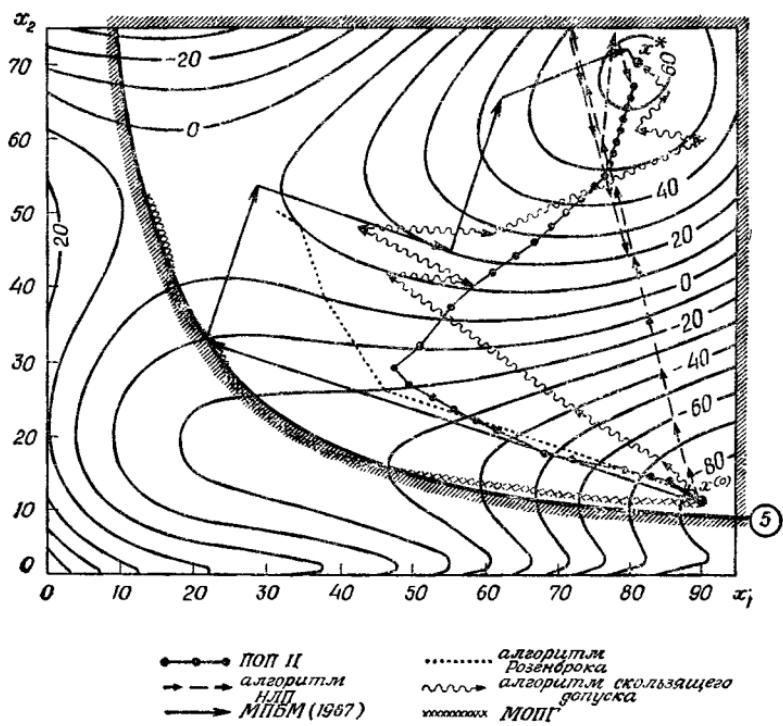
Необходимость учета всех упомянутых выше факторов в определенной степени затрудняет интерпретацию результатов оценивания эффективности алгоритмов путем решения тестовых задач.

В частности, оказывается, что эффективность того или иного алгоритма, оцениваемая на основе требуемого машинного времени при решении тестовых задач (и в несколько меньшей степени — оценка, основанная на количестве тестовых задач, решенных с помощью рассматриваемого алгоритма), является в значительно большей степени качественной, чем это представляется на первый взгляд.

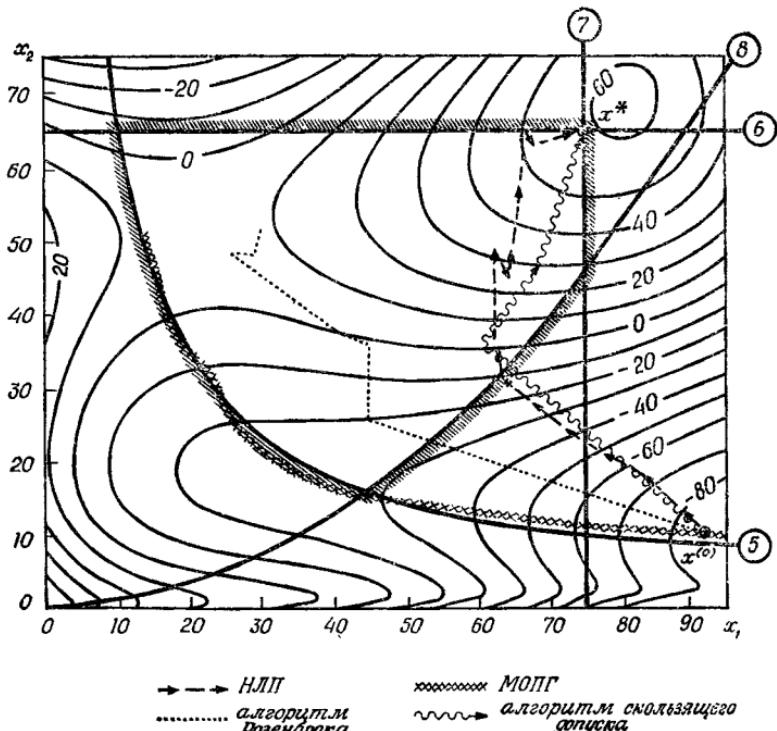
Учитывая изложенные выше соображения, проведем анализ эффективности различных алгоритмов нелинейного программирования, исходя из следующих критериев. Первый и наиболее важный критерий сводится к ответу на вопрос: удается ли вообще решить с помощью рассматриваемой программы поставленную задачу? Этот критерий выбран потому, что для пользователя наиболее ценным качеством машинной программы является ее способность обеспечивать решение самых разнообразных задач нелинейного программирования. Вторым критерием является требуемое количество машинного времени. Наконец, используется и третий критерий, характеризующий трудоемкость подготовительных процедур, осуществляемых вручную на этапе, предшествующем реализации той или иной программы на ЭВМ. Этот критерий приобретает весьма важное значение при сравнении временных затрат на подготовительном этапе применительно к схеме оптимизации методами прямого поиска с соответствующими временными затратами в случае, когда используются методы, требующие знания аналитического вида частных производных.

9.2. СРАВНЕНИЕ НЕКОТОРЫХ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ: ДВУМЕРНЫЕ ЗАДАЧИ

Чтобы проиллюстрировать поведение траектории поиска экстремума задачи нелинейного программирования при наличии ограничений в двумерном пространстве, с помощью множественной регрессии была построена целевая функция специальной структуры (табл. 9.2.1). Эта функция в заданном интервале значений двух независимых переменных имеет один пик и одну седловую точку. Рассмотрен ряд конкретных задач, каждая из которых сводится к максимизации целевой функции при наличии некоторого множества ограничений-неравенств, приведенных в табл. 9.2.2. Пять таких задач, конкретизированных путем выбора подмножества ограничений-неравенств, приведены в табл. 9.2.3, а целевая функция, ограничения и соответствующие траектории, исходящие из недопустимой начальной точки $x^{(0)} = [95 \ 10]^T$, изоб-



Ф и г. 9.2.1. Задача 1.



Ф и г. 9.2.2. Задача 2.

ражены графически на фиг. 9.2.1 — 9.2.5. Пик целевой функции находится в точке с координатами $x_1 = 81,154841$ и $x_2 = 69,135588$; в этой точке целевая функция принимает значение 61,9059345. Условный максимум находится в точке с координатами $x_1 = 75,000000$ и $x_2 = 65,000000$; в этой точке значение целевой функции равняется 58,9034360.

Таблица 9.2.1

Целевая функция

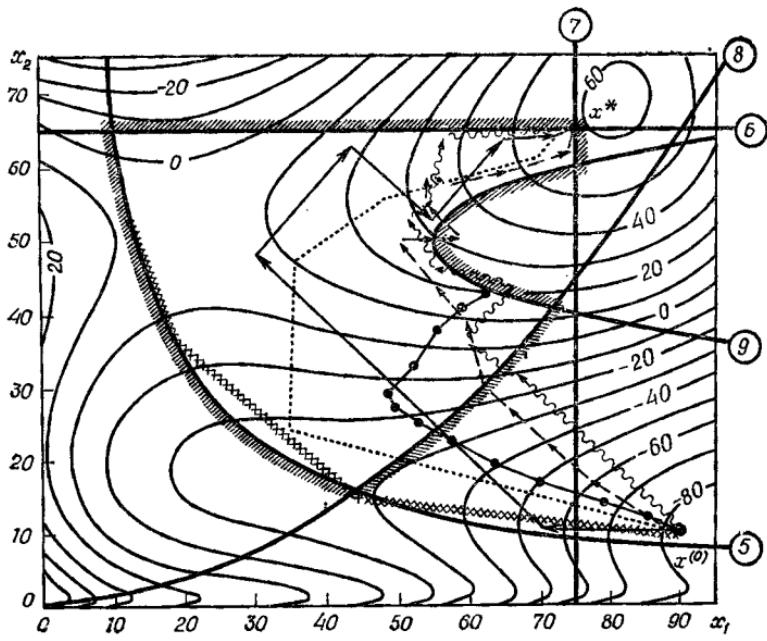
$$f(x) = B_1 + B_2(x_1) + B_3(x_1)^2 + B_4(x_1)^3 + B_5(x_1)^4 + B_6(x_2) + B_7(x_1)(x_2) + B_8(x_1)^2(x_2) + \\ + B_9(x_1)^3(x_2) + B_{10}(x_1)^4(x_2) + B_{11}(x_2)^2 + B_{12}(x_2)^3 + B_{13}(x_2)^4 + B_{14}\left[\frac{1.0}{x_2 + 1.0}\right] + \\ + B_{15}(x_1)^2(x_2)^2 + B_{16}(x_1)^3(x_2)^2 + B_{17}(x_1)^8(x_2)^3 + B_{18}(x_1)(x_2)^2 + B_{19}(x_1)(x_2)^3 + \\ + B_{20}\{\exp[0.0005(x_1)(x_2)]\}$$

$B_1 = 75,1963666677$	$B_{11} = 0,2564581253$
$B_2 = -3,8112755343$	$B_{12} = -0,0034604030$
$B_3 = 0,1269366345$	$B_{13} = 0,0000135139$
$B_4 = -0,0020567665$	$B_{14} = -28,1064434908$
$B_5 = 0,0000103450$	$B_{15} = -0,0000052375$
$B_6 = -6,8306567613$	$B_{16} = -0,0000000063$
$B_7 = 0,0302344793$	$B_{17} = 0,0000000007$
$B_8 = -0,0012813448$	$B_{18} = 0,0003405462$
$B_9 = 0,0000352559$	$B_{19} = -0,0000016638$
$B_{10} = -0,0000002266$	$B_{20} = -2,8673112392$

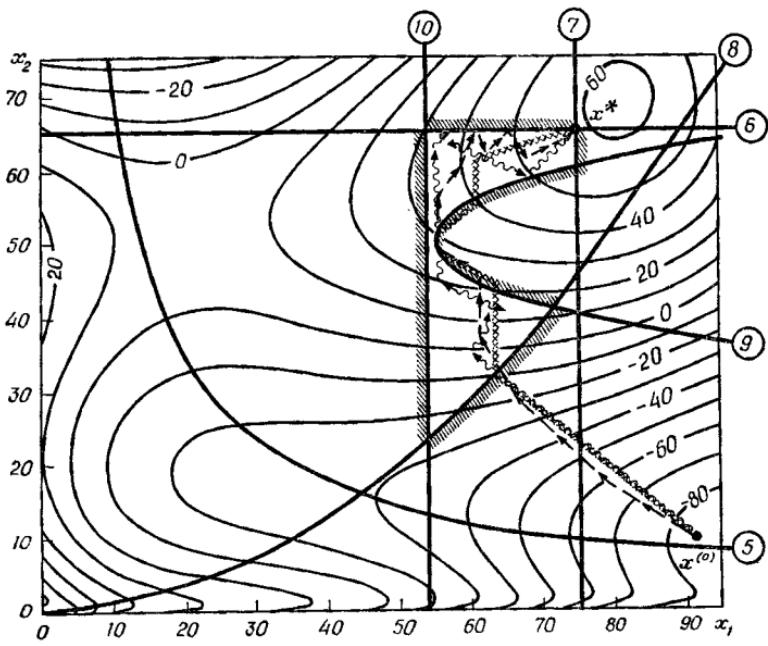
Задача 3 (фиг. 9.2.3) позволяет наглядно представить ряд типичных характеристик машинных программ, к обсуждению которых мы и переходим.

1. Алгоритм скользящего допуска. Каждый из векторов, изображенных на фиг. 9.2.3, соединяет следующие одна за другой наилучшие точки, генерируемые в процессе оптимизационного поиска. Напомним, что в методе скользящего допуска используется симплекс с $(n + 1)$ вершинами, где n — число независимых переменных. Следовательно, на каждом этапе поиска наилучшей точкой является та вершина рассматриваемого на данном этапе симплекса, в которой целевая функция принимает наибольшее (по отношению к остальным вершинам) значение.

Поскольку начальная точка не является допустимой, программа реализует поиск новой точки, удовлетворяющей исходному критерию допуска. Этот шаг поиска изображен графически первым вектором, проведенным из точки с координатами $x_1 = 90,0$ и $x_2 = 10,0$ в точку с координатами $x_1 = 68,787$ и $x_2 = 31,213$. Следует отметить, что поиск точки, удовлетворяющей критерию допуска, реализуется по *наицратачайшему* пути вблизи границы допустимой области. Тот факт, что траектория поиска, кроме того,



Ф и г. 9.2.3. Задача 3.



Ф и г. 9.2.4. Задача 4.

Таблица 9.2.2

Ограничения-неравенства, используемые для построения двумерных моделей при заданной целевой функции (см. табл. 9.2.1)

$$g_1(x): x_1 \geq 0$$

$$g_2(x): x_2 \geq 0$$

$$g_3(x): 95,0 - x_1 \geq 0$$

$$g_4(x): 75,0 - x_2 \geq 0$$

$$g_5(x): x_1 x_2 - 700,0 \geq 0$$

$$g_6(x): 75,0 - x_1 \geq 0$$

$$g_7(x): 65,0 - x_2 \geq 0$$

$$g_8(x): x_2 - 5,0 \left\{ \frac{x_1}{25,0} \right\}^2 \geq 0$$

$$g_9(x): (x_2 - 50,0)^2 - 5,0 (x_1 - 55,0) \geq 0$$

$$g_{10}(x): x_1 - 54,0 \geq 0$$

$$g_{11}(x): \frac{30,0}{20,0} (x_2 - 45,0) - (x_1 - 45,0) \geq 0$$

$$g_{12}(x): x_1 - 35,0 - \frac{40,0}{25,0} (x_2 - 40,0) \geq 0$$

перпендикулярна линиям уровней целевой функции, является простой случайностью.

Четыре последующих шага в процессе оптимизационного поиска имеют длину, превышающую линейный размер допустимой области, так как длина ребра первоначального симплекса принималась равной пяти единицам. Если бы размеры исходного симплекса были меньше указанных выше, траектория поиска проходила бы ближе к границе, определяемой ограничением 9, и, следовательно, затраты машинного времени были бы более значительными.

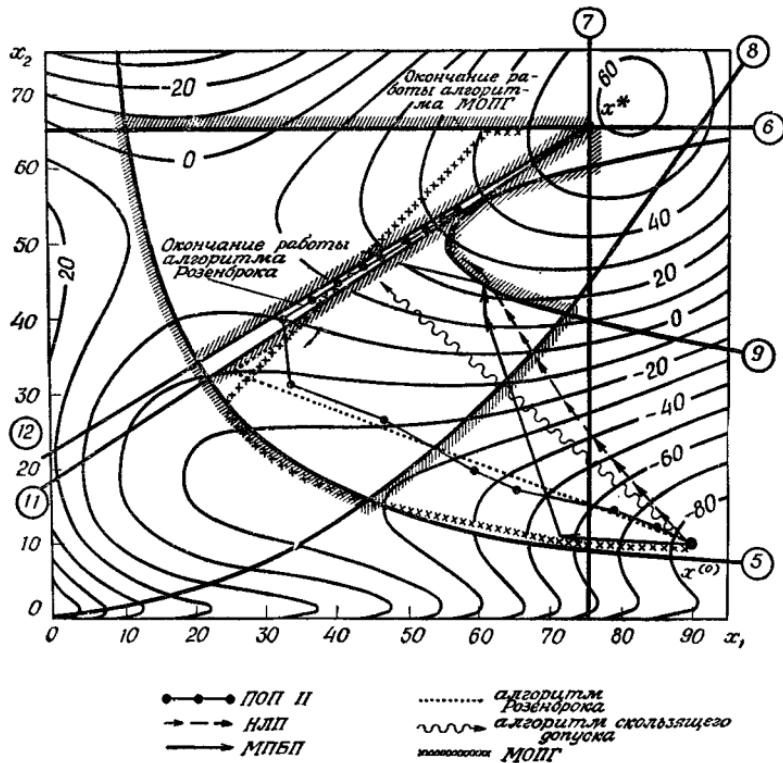
Пять первых шагов оптимизационного поиска позволяют определить точку вблизи оптимума; однако окончательная сходи-

Таблица 9.2.3

Варианты двумерных задач нелинейного программирования при наличии ограничений

Номер задачи	Номер фигуры	Номера ограничений
1	9.2.1	1, 2, 3, 4, 5
2	9.2.2	5, 6, 7, 8
3	9.2.3	5, 6, 7, 8, 9
4	9.2.4	5, 6, 7, 8, 9, 10
5	9.2.5	5, 6, 7, 8, 9, 11, 12

мость к оптимальному решению достигается лишь в ходе реализации дополнительных 39 итерационных этапов, что объясняется постепенным уменьшением размеров симплекса на завершающих стадиях вычислительного процесса. (Чтобы избежать чрезмерной громоздкости графического изображения, на фиг. 9.2.3



Ф и г. 9.2.5. Задача 5.

эти 39 шагов работы алгоритма не представлены.) Вектор x осциллирует в окрестности точки, лежащей на расстоянии около пяти единиц от оптимума; процесс осцилляции продолжается до тех пор, пока размеры симплекса не уменьшатся до такой степени, когда станет возможным получение дополнительных минимизирующих поправок к значению целевой функции (т. е. дальнейшее перемещение в направлении к искомой оптимальной точке).

2. Алгоритм НЛП. Каждый из векторов, формирующих траекторию алгоритма, соединяет следующие одна за другой точки, получаемые в ходе оптимизационного поиска. На пяти первых шагах от исходной точки реализуются перемещения по алгоритму наискорейшего спуска, поскольку получаемые при этом точки лежат далеко за пределами допустимой области. Шестой шаг, предпринимаемый из точки, лежащей вне допустимой области,

переводит траекторию поиска внутрь допустимой области и осуществляется по алгоритму линейного программирования. Все последующие шаги реализуются методом линейного программирования. Перемещения с помощью линейного программирования производятся вдоль ограничивающих поверхностей (отметим, в частности, что последние шаги реализуются вдоль ограничений 9 и 7).

3. МПБМ (вариант, относящийся к 1967 г.). При использовании МПБМ вначале осуществляется поиск внутренней точки; схема поиска выглядит следующим образом. При заданной начальной точке нарушаются ограничения 7 и 8; поэтому с помощью МПБМ прежде всего минимизируется взятая со знаком минус функция, задающая ограничения 7 (при этом должны удовлетворяться ограничивающие условия 5, 6 и 9), после чего минимизируется взятая со знаком минус функция, задающая ограничение 8 (при этом должны удовлетворяться ограничивающие условия 5, 6, 7 и 9). В результате получается внутренняя точка, лежащая в окрестности точки с координатами $x_1 = 31$ и $x_2 = 48$. Таким образом, первые два вектора, соединяющие первоначальную недопустимую стартовую точку с внутренней (допустимой) точкой, получаются в результате большого числа итерационных этапов (на рисунке микроструктура итерационного процесса не представлена).

Завершающая часть оптимизационного поиска сводится к решению девяти сопутствующих подзадач (см. описание МПБМ в гл. 8), в результате чего обеспечивается сходимость к условному экстремуму, являющемуся решением исходной задачи. Заключительные шаги поиска для удобства показаны на фиг. 9.2.3 в сгруппированном виде. При использовании варианта МПБМ, относящегося к 1967 г., вид траектории оптимизационного поиска зависит от порядка рассмотрения ограничений, тогда как в варианте МПБМ, относящемся к 1970 г., такая зависимость не имеет места, так как алгоритм минимизирует сразу сумму всех функций, ассоциированных с нарушенными ограничениями. Интересно отметить, что МПБМ-1970 в случае задачи 1 заканчивает работу возле седловой точки; попытки применить данный метод для решения других задач оказались безуспешными.

4. Алгоритм Розенброка. За исключением первого вектора траектории поиска, который просто соединяет исходную точку с внутренней (допустимой) точкой решаемой задачи, каждый из последующих векторов, получаемых с помощью алгоритма Розенброка, ассоциируется с новой минимизирующей поправкой к предшествующему значению целевой функции. Машинная программа обеспечивает поиск допустимой начальной точки путем максимизации суммы значений функций, задающих нарушенные ограничения (в результате последовательной минимизации эта сумма

обращается в нуль); другими словами, все ограничивающие условия оказываются выполненными. Для простоты на фиг. 9.2.3 показан лишь вектор, соединяющий исходную точку с первой из допустимых точек рассматриваемой задачи. Поскольку в направлении поиска допустимой точки методом Розенброка длина первоначального шага полагается равной одной десятой исходного значения каждой из независимых переменных, длина каждого из шагов на начальной стадии поиска составляла девять единиц в направлении x_1 и одну единицу в направлении x_2 . Так как при этом значения функций, задающих нарушенные ограничения, улучшались, поиск продолжался почти в одном и том же направлении (длина каждого из шагов в направлении x_1 значительно превышала длину каждого из шагов в направлении x_2).

Дальнейший поиск оптимальной точки при старте из первой допустимой точки включал дополнительно 136 этапов, по завершении которых вычислительный процесс заканчивался. Точка, полученная в результате реализации первых шести этапов, оказалась весьма близкой к условно-оптимальной точке $x = [74,72669\ 64,99136]^T$. На фиг. 9.2.3 микроструктура работы алгоритма на протяжении 130 этапов не представлена. (Интересно отметить, что алгоритм Розенброка при решении задач 1, 2 и 5 привел к седловой точке, а в условиях задачи 4 оказался не в состоянии определить допустимую точку.)

5. ПОП II. Наиболее примечательной особенностью ПОП II является то, что оптимизационный поиск осуществляется при малой длине шага. Линейно программирующая стадия ПОП II генерировала траекторию поиска, проходящую почти вдоль ограничений 9 и 7. (На фиг. 9.2.3 точки, попадающие на ограничивающую поверхность 9, не показаны, так как они расположены слишком близко друг от друга.)

6. Алгоритм ОГМОП (вариант, разработанный фирмой «Юньон Карбайд»). Машинальная программа, реализующая этот алгоритм, оказалась не в состоянии решить задачу 3. При решении задачи 1 была найдена точка $x = [13,8\ 50,6]^T$, лежащая неподалеку от седловой точки. В процессе решения задач 2—5 на одной из стадий оптимизационного поиска после выполнения операции проектирования частные производные функций, задающих ограничения, оказывались равными нулю.

7. Алгоритм МОПГ. Метод обобщенного приведенного градиента относится к числу методов, индуцирующих траекторию поиска, проходящую весьма близко к ограничивающей поверхности, которая определяется ограничением в виде неравенства. В силу этого обстоятельства при решении задачи 3 «останов» имеет место не при достижении глобального оптимума, а при «попадании» в локальный оптимум, лежащий в окрестности точки с координатами $x_1 = 12$ и $x_2 = 53$. Следует отметить, что результат поиска

зависит от стартовой точки. При решении задачи 4 глобальный оптимум был достигнут при заданной стартовой точке по той причине, что локальный (неглобальный) оптимум и седловая точка задачи оказались вне допустимой области из-за наличия ограничения 10.

9.3. СРАВНЕНИЕ НЕКОТОРЫХ АЛГОРИТМОВ ОПТИМИЗАЦИИ ПРИ НАЛИЧИИ ОГРАНИЧЕНИЙ В СЛУЧАЕ БОЛЕЕ СЛОЖНЫХ ЗАДАЧ

Для оценки эффективности некоторых алгоритмов нелинейного программирования при наличии ограничений как с вычислительной, так и с других точек зрения Стоккер [1] применял эти алгоритмы при решении тестовых задач на ЭВМ CDC 6600. Колвилл [3] исследовал еще большее (по сравнению со Стоккером) чис-

Таблица 9.3.1

Ограничения в виде неравенств	Ограничения в виде равенств		
	нелинейные	линейные	отсутствуют
Нелинейные	A1	B1	C1
Линейные	A2	B2	D
Отсутствуют	A3	C2	E

ло алгоритмов, хотя и приводит меньше подробностей относительно их достоинств. В дополнение к результатам Стоккера и Колвилла приведем в данном разделе результаты, которые не были опубликованы. Большинство тестовых задач, содержащихся в приложении А, заимствовано из опубликованных источников; остальные тестовые задачи приложения А подготовлены нами специально. Некоторые из этих задач можно считать типичными задачами нелинейного программирования в том смысле, что они (или их аналоги) часто встречаются при решении конкретных практических проблем. Эти тестовые задачи представляют различные группы (классы) задач нелинейного программирования, т. е. это задачи с различными типами входящих в них функций, с разными структурными характеристиками, с неодинаковой степенью сложности и различным количеством независимых переменных.

Практически во всех рассматриваемых задачах целевые функции нелинейны. Системы же ограничений варьируются: наиболее тривиальными являются системы ограничений, состоящие из одних лишь линейных ограничений в виде неравенств, а наиболее полными (в структурно-аналитическом отношении) — си-

Таблица 9.3.2

Классификация машинных программ, предложенная Колвиллом

	Производные и форма их представления	Ссылка на раздел (подраздел)
Методы прямого поиска (МПП) Оптим (фирма «Мобил Ойл») Последовательный поиск (фирма «Гласс и Купер») Комбинированный поиск Розенброка Клингмана и Химмельблау	Отсутствуют » » » Аналитическая запись	4.5.2 6 ¹⁾ 4.5.1 7.1.2 <i>J. Accos. Computer Math.</i> , 11, 400 (1964)
Симплексный поиск (фирма «Шелл Дивелопмент») Проб Метод скользящего допуска ²⁾	Отсутствуют » »	6 ¹⁾ 8.1
Мелкошаговые градиентные методы (МГМ) ПОП/360 (IBM) Рикошет (Гринштадт)	Числовые значения Аналитическая запись	6.1.2 <i>J. SIAM Appl. Math.</i> , 14, 3 (1966)
Программа «Карбайд» ОГМОП Метод аппроксимирующего программирования Подъем с уклонением (фирма «Шелл-Дивелопмент»). НЛП ²⁾	Числовые значения Аналитическая запись Числовые значения » Аналитическая запись	6.3.2 6.1.1 6.2
Крупношаговые градиентные методы (КГМ) МОПГ ³⁾ Метод допустимых направлений (IBM) Сочетание метода Дэвидона с МБП Выпуклое программирование (IBM, Франция) Метод сопряженных градиентов (Голдфарб) Проективный метод с переменной метрикой (Муртаг) ²⁾ Метод проекции градиента («Шелл Дивелопмент») Улучшенный метод проекции градиента (институт Паскаля) Модифицированный метод приведенного градиента (институт Паскаля) Модифицированный метод допустимых направлений	Аналитическая запись То же » » » » » » » » » »	6.5 6.4 7.1.3 6.3.3 6.3.3 6.3.1
Методы вторых производных (МВП) (IBM, ФРГ) Гаусса—Ньютона—Кэролла МПБМ Солвера	» » Аналитическая запись или числовые значения Аналитическая запись	6.4 5.3 7.2 6.1.4

¹⁾ Ссылки на опубликованные работы см. в конце указанной главы.²⁾ В обзор Колвилла (1958 г.) не включен.

стемы, содержащие нелинейные ограничения как в виде равенств, так и в виде неравенств. В табл. 9.3.1 представлена своего рода классификационная схема, согласно которой задачи нелинейного программирования в зависимости от типа системы ограничений делятся на пять основных классов. Последовательность А, В, С, Д и Е (и индексы подклассов 1, 2 и 3) соответствует порядку уменьшения сложности задачи, хотя и не исключено, что задача с единственным нелинейным ограничением в виде равенства (класс А) окажется менее сложной, нежели задача, содержащая линейные ограничения в виде равенств и нелинейные ограничения в виде неравенств (класс В).

Таблица 9.3.3

Среднее стандартизированное время при использовании программ классов А, В, С, Д и Е¹⁾

Номер задачи (приложение А)	Класс	Категория (тип) ²⁾			
		МПП	МГМ	КГМ	МВП
15	A2	0,179	0,060	0,049	—
19	B2	—	0,305	0,148	0,238
7	C1	0,089	0,054	0,033	—
11	C1	0,295	0,017	0,023	0,019
14	C1	—	—	—	—
18	C1	—	0,326	0,220	0,151
10	D	0,384	0,055	0,027	0,023
8	E	0,026	0,049	0,025	0,025

1) Задано Колвиллом (1958).

2) См. табл. 9.3.2.

Колвилл собрал данные об эффективности более тридцати различных машинных программ, предназначенных для решения задач нелинейного программирования; он проанализировал «работу» этих программ для восьми стандартных тестовых задач, а именно задач 7, 8, 10, 11, 14, 15, 18 и 19, приведенных в приложении А. Анализ проводился методом «коллективных усилий», предполагающим участие в испытаниях программ нескольких специалистов. Каждый из участников испытаний должен был решить совокупность тестовых задач, выбирая по своему усмотрению метод решения, машинную программу и тип ЭВМ. Колвилл сгруппировал все программы таким образом, что получилось четыре больших класса; классификация Колвилла приведена в табл. 9.3.2. Эффективность каждого из этих классов можно оценить с единой точки зрения, взяв за основу среднее стандартизированное время, требуемое для решения каждой из тестовых задач (табл. 9.3.3).

Поскольку многие из машинных программ оказались слишком неэффективными при решении тех или иных задач и не учитывался такой показатель, как число случаев, когда задача в установленное время не была решена, а условия распечатки и останова были в разных испытаниях различными (не говоря уже о том, что имели место и другие трудности в установлении оценочных показателей¹⁾), среднее стандартизированное время представляет собой лишь грубую характеристику эффективности каждой из групп алгоритмов. В работе Колвилла отсутствуют данные о среднем стандартизированном времени, которое тратится на решение задачи 14, так как в этом случае имеет место большое число локальных оптимумов. Кроме того, Колвилл не приводит данных относительно того, удается ли решить методом прямого поиска задачи 18 и 19. Как и следовало ожидать (см. табл. 9.3.3), метод прямого поиска (МПП) оказывается наиболее медленно действующим, а крупношаговые градиентные методы (КГМ) и методы «вторых производных» (МВП) — наиболее быстродействующими. Некоторое удивление вызывает тот факт, что мелкошаговые градиентные методы (МГМ) работают так же быстро, как и КГМ и МВП.

Время, расходуемое на выполнение подготовительных операций, представляет собой другой существенный критерий эффективности машинной программы, предназначеннай для решения задач нелинейного программирования. Соответствующие данные для типов программ, перечисленных в табл. 9.3.2, приведены в табл. 9.3.4. Указанные в табл. 9.3.4 временные затраты на выполнение подготовительных операций в известной степени оценены по минимуму и не включают затрат времени, требуемых для ознакомления с самой программой (т. е. предполагается, что пользователю структура программы известна), а также возможные потери времени в тех случаях, когда подготовительные операции приходится выполнять заново из-за неполадок в работе программы при первых ее «испытаниях». Временные затраты, связанные с выполнением подготовительных операций пользователями, которые не знакомы с МПБМ, ПОП II и алгоритмом НЛП, оказываются в 2—5 раз выше соответствующих оценочных данных, приведенных в табл. 9.3.4. Представляется маловероятным, чтобы оценки временных затрат для методов, основанных на использовании аналитической записи вторых производных (например, МПБМ), оказались справедливыми и для других задач, поскольку, например, в задаче с девятью нелинейными ограничениями (при десяти переменных) требовалось бы вычислить (причем без ошибок) 100 первых частных производных и 550 вторых частных производных. За исключением тех случаев, когда большинство частных производных равно нулю или является константами, на

¹⁾ См. разд. 5.1 и 9.1.

выполнение этих операций потребуется определенное время. С другой стороны, если использовать дополнительную программу «символического типа», время, требуемое на вычисление производных, возможно, удалось бы сократить. Следует отметить, что

Таблица 9.3.4

Время (в часах), расходуемое на подготовительные операции при решении тестовых задач с помощью некоторых из алгоритмов, приведенных в табл. 9.3.1

Алгоритм	Производные	Номер задачи (см. приложение А)							
		7	8	10	11	14	15	18	19
Скользящего допуска ¹⁾	О	0,3	0,1	0,5	0,1	0,4	0,1	0,5	0,5
Оптим ²⁾	О	1,5	0,1	1,0	1,0	2,0	—	—	—
ПОП 360 ²⁾	Ч	1,0	0,5	2,0	1,0	2,0	2,0	2,0	2,0
МАП ²⁾	Ч	—	1,0	—	—	—	—	—	—
НЛП ¹⁾	А	1,5	1,0	1,0	0,8	1,5	1,3	1,2	0,8
ОГМОП ¹⁾	А	—	0,5	0,5	0,4	1,5	1,2	0,7	2,0
ОГМОП ²⁾	Ч	6,0	3,0	5,0	4,0	—	6,0	4,0	6,0
МОПГ ¹⁾ (1970)	А	6,0	0,4	1,0	1,0	1,0	1,5	1,0	3,5
МОПГ ²⁾	А	5,0	1,5	2,0	2,0	—	4,0	2,0	4,0
Дэвидона — МБП ²⁾	А	6,0	1,0	2,0	3,0	—	—	4,0	4,0
Минимал ²⁾	О	0,8	0,5	1,5	1,0	—	1,0	—	1,0
МПБМ (1967)	А ³⁾	4,0 ¹⁾	1,5 ¹⁾	3,5 ²⁾	4,0 ²⁾	—	—	6,0 ²⁾	3,0 ²⁾
Розенброка ¹⁾	О	0,3	—	0,8	0,8	—	—	—	—
Проективный с переменной метрикой ²⁾	А	6,0	2,0	2,0	3,0	—	—	3,0	4,0
Куранта ²⁾	А	4,0	1,0	3,0	2,0	—	3,0	2,0	4,0

1) Неопубликованные данные.

2) Занимствовано у Колвилла (1968 г.).

3) Требуется вычисление вторых частных производных.

Обозначения: А — аналитическая запись, Ч — числовые значения, О — производные не используются.

у малоопытного специалиста лишь программа скользящего допуска не отнимает много времени в процессе подготовки задачи к решению на ЭВМ; в этом случае требуется перенести на перфосетели лишь данные относительно целевой функции, ограничений и стартовой точки $x^{(0)}$.

Стоккер оценил эффективность тех же самых программ, что и в работе Колвилла; кроме того, Стоккер получил соответствующие данные для алгоритма НЛП и алгоритма скользящего допуска.

В табл. 9.3.5 указано стандартизированное время для различных типов программ и разных задач (из числа задач, приве-

Таблица 9.3.5

Стандартизированное время, требуемое при решении восьми тестовых задач с помощью различных алгоритмов

Алгоритм	Номер раздела	Номер задачи (см. приложение А)							
		7	8	10	11 ^a)	14 ^a)	15	18	19
Оптим	4.5	0,142 ^b	0,010 ^b	0,100 ^b	0,014 ^b	0,250 ^b	e, c	e, b	e, c
ПОП 360 ^b	6.1	0,044	0,011	0,037	0,016	0,067	0,090	0,168	0,313
ПОП II ^d	6.1	0,078	0,135	e	e	e	f	e	f
МАП ^b	6.1	e	0,086						
НЛП ^d	6.2	0,073	0,638	0,074	0,110	g	0,353	4,15	2,52
Проекция градиента ^b	6.3.1	f	0,040	0,0127	f	f	f	f	0,0936
ОГМОП ^b	6.3.2	0,094	0,062	0,023	0,016	0,031	0,049	0,686	0,328
ОГМОП ^c	6.3.2		e	e	0,104	f	e	e	0,589
МОПГ (1970) ^b	6.5	0,022	0,010	0,008	0,006		0,008	0,084	0,018
МОПГ (1970) ^c	6.5	0,176	0,072	0,069	e	e	0,415	0,242	0,132
Дэвидсона — МБП ^b	7.1.3	0,022	0,006	0,039	0,015	f	f	0,384	0,272

Алгоритм	Номер раздела	Номер задачи (см. приложение А)							
		7	8	10	11 а)	14 а)	15	18	19
Минимал ^h	7.1.4	0,118	0,087	0,412	0,103	e	0,184	e	0,875
Метод центров ^b	7.1.5				0,186				
МПБМ (1968)									
b	7.2			0,016	0,028	f	0,151	0,238	
d	7.2	f	0,082	0,127	0,048	g	f	2,53	0,719
МПБМ (1970) ^c	7.2					f	0,159		
Розенброка									
b	4.3.6			0,424	0,358	f	f	e	f
d	7.1.2	0,045	0,032	e	0,078	f	f	e	f
Скользящего допуска ⁱ	8.3	0,193	0,023	0,344	0,121	1,31	0,214	22,8	6,45
Проективный с переменной метрикой ^b	6.3.3	0,025	0,016	0,006	0,006	f	0,574	0,062	
Куранта ^b		0,036	0,004	0,026	0,025		0,072	0,380	0,209

Обозначения

a — при допустимом начальном векторе; b — Colville A. R., A Comparative Study on Nonlinear Programming Codes, IBM N.Y., Sci. Center Rept. 320-2949, June 1968, Supplement, 1969; c — не опубликовано; d — Stocker D. C., M. S. Thesis, Univ. of Texas, Austin, Tex., 1969; e — не позволяет найти решение задачи; f — к данной задаче алгоритм неприменим; g — решение не получено вследствие ошибок при вычислениях производных; h — Holzman A. G., Comparative Analysis of Nonlinear Programming Codes with the Weismann Algorithm, SRCC Rept. 113, Univ. of Pittsburgh, Pittsburgh, Pa., Nov. 1969; i — Pavian D. A., Ph. D. Dissertation, Univ. of Texas, Austin, Tex., 1969; j — останов при погрешности 1% в f(x) из-за слишком большого времени, требуемого для вычислений.

денных в приложении А). Содержащиеся в таблице данные заимствованы у Колвилла и Стоккера, а также взяты из некоторых неопубликованных источников. Следует обратить внимание на то, что данные, взятые из разных источников, не всегда оказываются одинаковыми. Тот факт, что та или иная ячейка таблицы у Колвилла оказывается незаполненной, как правило, означает, что алгоритм не смог справиться с решением соответствующей задачи (по крайней мере решение задачи не известно, хотя попытка решить эту задачу и имела место). На длительность подготовительных процедур оказывают влияние все факторы, о которых шла речь в разд. 9.1, так что практически невозможно сделать какой-либо общий вывод относительно эффективности рассмотренных алгоритмов, даже располагая данными, представленными в табл. 9.3.5. Однако можно утверждать (по крайней мере в результате предварительного ознакомления с приведенными выше результатами), что для выявления возможностей практического использования метода скользящего допуска, ПОП, алгоритма НЛП, метода МОПГ, проективного метода с переменной метрикой, метода Куранта и МПБМ требуется проведение дополнительных исследований.

С этой целью Стоккер провел глубокий анализ эффективности пяти указанных выше алгоритмов, применяя их при решении пятнадцати тестовых задач; результаты анализа (вместе с данными, относящимися к двум другим алгоритмам и пяти дополнительным тестовым задачам) приведены в табл. 9.3.6. Таким образом, табл. 9.3.6 содержит операционные характеристики (время, решения) семи машинных программ, полученные путем анализа их возможностей в связи с решением 20 тестовых задач. Вначале обсудим (разд. 9.3.1) рабочие характеристики некоторых машинных программ, полученные при решении задач, представляющих наибольший практический интерес, а затем (разд. 9.3.2) подведем итоги относительно возможностей каждой программы в отдельности.

9.3.1. РАБОЧИЕ ХАРАКТЕРИСТИКИ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ, ПОЛУЧЕННЫЕ ПРИ РЕШЕНИИ ЗАДАЧ, ПРЕДСТАВЛЯЮЩИХ ОСОБЫЙ ИНТЕРЕС

Задача 11¹⁾. Данную задачу, характеризующуюся квадратичной целевой функцией и содержащую шесть квадратичных ограничений в виде неравенств при пяти независимых переменных (значения которых могут меняться в пределах фиксированных интервалов), удается решить с помощью любой из рассмотренных выше машинных программ, за исключением ПОП II. Алгоритм Розенброка оказался не в состоянии обеспечить такую же точность, какая

¹⁾ Рассматриваемые здесь задачи приведены в приложении А.

Машинное время (в секундах), расходуемое при решении задач нелинейного известных

Класс (см. фиг. 9.3.1)	A				B			
	A1	A2		B1	B2			
Номер задачи (см. приложение A)	20	15	5	13 ^a	1	4 ^b	6	19
Число переменных	24	3	3	12	2	10	45	16
Число ограничений в виде равенств линейных нелинейных	2 12	2 4	1 1	4 3	1	3	12	8
Число ограничений в виде неравенств линейных нелинейных	6				1			
Верхняя и нижняя грани- цы	24	12	3	16		10	45	32
Время, с								
МОПГ (1969—1970)	4,98 ^f	9,14	1,21		1,06	1,45	<i>h</i>	2,92
Скользящий допуск	511	4,71	0,84	<i>c</i>	0,43	27,9	<i>c</i>	142
НЛП	<i>e</i>	7,76	0,33		0,07	5,37 ^f		55,4
МПБМ (1968)	<i>i</i>	<i>i</i>	<i>a</i>	<i>j</i>	0,51	2,44	29,4	15,6
МПБМ (1970), лучшие варианты					0,22			
ПОП II	<i>i</i>	<i>i</i>	<i>c</i>	<i>i</i>	<i>i</i>	<i>i</i>	<i>i</i>	<i>i</i>
Розеиброка	<i>i</i>	<i>i</i>	<i>c</i>	<i>i</i>	4,52	<i>i</i>	<i>i</i>	<i>i</i>
ОГМОП	<i>e</i>	<i>e</i>	<i>e</i>	<i>c</i>	5,18	1,15 (6,63)	<i>e</i>	13,06

Машинное время указано применительно к ЭВМ CDC 6600 (стандартизированное время 22,0 с). Исключением задачи 20) нелинейные. Незаполненные ячейки означают, что соответствующая задача

Обозначения

a— получено недопустимое решение (ограничивающие условия не удовлетворены); *b* — в скобках линейной сходимости; *d* — ограничивающие условия надлежащим образом не удовлетворены; *e* — резкстремум; *h* — получены отрицательные значения аргумента логарифмической целевой функции; слишком большими трудностями; *k* — не способен обеспечить решение (возможно, из-за ошибок).

Таблица 9.3.6

программирования различной степени сложности с помощью некоторых алгоритмов

C										D	E		
CI													
3	7	9	11 ^b	12	14	16	18 ^b	13 ^a	10	17	2	8	
2	3	4	5	5	6	9	15	5	5	10	2	4	
3	14	1	6	35	4	13	5	3	10				
4	6	5	10	10		1	15	10	5	20	0	8	
<i>g</i>	3,83	<i>e</i>	{ 3,57 6,51}		<i>e</i>	<i>e</i>	5,38	2,25	1,54	0,35 ^l	0,99	1,59	
0,34	4,23	3,63	2,67 (13,9)	103,3	29,1	59,6	<i>c</i>	72,1	7,58	2,92 ^d	0,86	0,50	
<i>g</i>	1,60		2,31 (0,64)	4,88		<i>e</i>	91,4 (84,1)		1,62	<i>h</i>	0,18	14,04	
0,82	<i>k</i>	<i>j</i>	1,05	<i>k</i>	<i>j</i>	<i>j</i>	5,58 (6,71)		2,79 1,99	1,32	0,11	1,80	
	12,66		(2,98)	76,0									
0,63	1,72		<i>c</i>	<i>e</i>	<i>e</i>	<i>e</i>	<i>e</i>		<i>c</i>	0,65	<i>e</i>	3,93	
1,31	0,52	<i>e</i>	1,71 (3,00)	<i>c</i>	<i>e</i>	<i>c</i>	<i>e</i>		<i>c</i>	<i>h</i>	0,21	0,70	
<i>e</i>		17,37	2,30		<i>e</i>	<i>e</i>	<i>e</i>		<i>e</i>	2,08	4,52	<i>e</i>	
<i>e</i>													

Классы задач приведены в порядке уменьшения сложности (от А до Е). Все целевые функции (за указанным в левой колонке методом не решалась).

указано время, расходуемое при старте из недопустимой точки; *c* — решение не получено из-за медленное отсутствует (не получено); *f* — для задачи 4а; *g* — достигается не глобальный, а локальный *i* — к данной задаче алгоритм не применим; *j* — вычисление вторых производных сопряжено со допускаемых при вычислении вторых производных); *l* — относится к 1970 г.

достигается при использовании других машинных программ. Это объясняется тем, что оптимизационный поиск методом Розенброка на последних этапах вычислительного процесса прекращается в силу слишком медленного изменения минимизирующих поправок к текущим значениям целевой функции в ходе перемещений вдоль ограничений в окрестности оптимума. ПОП II приводит к минимуму, лежащему ниже минимальных значений целевой функции, получаемых другими методами. Однако точка, соответствующая экстремуму, находится за пределами допустимой области. Алгоритм скользящего допуска, алгоритм НЛП, МПБМ, алгоритм Розенброка и метод МОПГ приводят к оптимальному решению задачи при старте из точки, не являющейся допустимой.

Задача 7. Это типичная задача из категории решаемых с помощью ПОП II. Фигурирующие в ней функциональные связи образуют «самосогласованную» модель, легко описываемую на языке команд программы. Интересующиеся способом построения моделей такого типа, и в частности данной модели, могут обратиться к работе [4].

Аналитический вид фигурирующих в задаче функций в явном виде не определен, так что нахождение частных производных этих функций оказывается весьма затруднительным. На выполнение подготовительных работ при решении задачи 7 с помощью алгоритма НЛП уходит 1,5 ч; чтобы получить аналитическую запись вторых частных производных, требуется дополнительно 2,5 ч; таким образом, суммарные временные затраты при подготовке задачи к решению с помощью МПБМ составляют 4 ч. Тем не менее неоднократные попытки найти решение задачи 7 с применением МПБМ заканчивались неудачей: программа прекращала работу в условиях, когда имела место «сходимость» к ошибочному решению. Эти неудачи, безусловно, объясняются ошибками, допускаемыми при отыскании вручную вторых частных производных целевой функции и функций, задающих ограничения, по независимым переменным. Помимо отмеченных выше, значительные временные затраты сопряжены с безуспешными попытками установить источник неполадок. Следует отметить, что методы прямого поиска, метод скользящего допуска, алгоритм Розенброка и ПОП II, не требующие вычисления частных производных фигурирующих в задаче функций, характеризуются гораздо меньшими затратами времени на подготовительные операции. Эти затраты при использовании любого из указанных методов составляют приблизительно 15 мин.

Задача 18. Эта задача, являющаяся двойственной по отношению к задаче 10, содержит 15 независимых переменных. Она характеризуется кубической целевой функцией при пяти кубических ограничениях. Для значений всех 15 переменных установлены нижние границы. Эту задачу удалось решить лишь с помощью алго-

ритмов МОПГ, НЛП и МПБМ, которые позволяют определить максимальное значение целевой функции при старте как из допустимой точки, так и из точки, не являющейся допустимой. При решении задачи 18 преимущества МПБМ и алгоритма МОПГ менее очевидны. Отметим, однако, что данная задача обладает такой структурой целевой функции и такими ограничениями, при которых вычисление частных производных не представляет особой трудности. На подготовительные операции при использовании МПБМ требуется всего 30 мин. Оказалось, что алгоритм скользящего допуска обеспечивает очень медленную сходимость к оптимальному решению, приводя к чрезвычайно большим затратам машинного времени. При этом, несмотря на то что оптимальное решение целевой функции в процессе поиска было почти установлено, вектор \mathbf{x} , соответствующий наилучшему из найденных значений $f(\mathbf{x})$, значительно отличался от оптимального вектора \mathbf{x} , полученного с помощью алгоритма НЛП и с помощью МПБМ. Алгоритмы Розенброка, ОГМОП и ПСП II оказались не в состоянии обеспечить сходимость к искомому решению.

Задача 12. Эта задача представляет собой модель функционирования гипотетического целлюлозно-бумажного комбината. Она обладает существенно нелинейной целевой функцией, характеризующейся сложной зависимостью от пяти независимых переменных, значения которых ограничены как снизу, так и сверху. Задача содержит три линейных ограничения в виде равенств и 35 нелинейных ограничений, записанных в виде неравенств. Решить эту задачу удалось лишь с помощью алгоритмов скользящего допуска и НЛП (разумеется, с применением ЭВМ). Прямой поиск, реализованный в рамках алгоритма Розенброка, привел к преждевременному останову из-за чрезвычайно медленного перемещения вдоль одного из ограничений, в результате чего скорость увеличения значений целевой функции была явно недостаточной. Программа ПОП II не справилась с решением задачи по не совсем понятным причинам. Неоднократные попытки решить задачу 12, используя МПБМ, также не привели к желаемому результату из-за невыявленных ошибок, допущенных при вычислении 315 частных производных. Задача 12 по степени сложности целевой функции и функций, задающих ограничения, напоминает задачу 7. Мы вновь подчеркиваем то обстоятельство, что в процессе нахождения вручную аналитических выражений частных производных целевой функции и функций, задающих ограничения, по независимым переменным не исключены ошибки. Хотя сравнение затрат машинного времени при использовании алгоритмов скользящего допуска и НЛП показывает, что вычислительная эффективность алгоритма НЛП выше вычислительной эффективности алгоритма скользящего допуска, следует иметь в виду, что машинная программа, составленная на основе метода НЛП, требует

нескольких часов предварительной подготовки задачи к решению на ЭВМ, тогда как временные затраты на выполнение подготовительных операций при использовании алгоритма скользящего допуска составляют менее 30 мин.

Задача 1. Это простейшая из задач, относящихся к классу В. Она характеризуется квадратичной целевой функцией двух независимых переменных и содержит одно линейное ограничение в виде равенства и одно нелинейное ограничение в виде неравенства. Напомним, что алгоритмы Розенброка и ПОП II не могут оперировать непосредственно с ограничениями, записанными в виде равенств. Однако, как уже отмечалось выше, ограничения в виде равенств можно включить в систему ограничений задачи в преобразованном виде, т. е. после замены равенства $h_i(\mathbf{x}) = 0$ ($i = 1, \dots, m$) на два неравенства

$$h_i(\mathbf{x}) \geq -\xi_i \text{ и } h_i(\mathbf{x}) \leq \xi_i, \quad i = 1, \dots, m.$$

Ограничения, записанные в таком виде, поддаются адекватному учету при использовании программы Розенброка и ПОП II, если пользователем установлены верхняя и нижняя границы изменения функций $h_i(\mathbf{x})$, т. е. если заданы значения $-\xi_i$ и ξ_i .

Указанный выше способ обращения с ограничениями в виде равенств пытались применить в рамках алгоритма Розенброка и ПОП II для решения задачи 1; при этом полагали $\xi_i = 10^{-4}$, так как ограничивающие условия для значений $h_i(\mathbf{x})$, имеющие вид $-10^{-4} < h_i(\mathbf{x}) < 10^{-4}$, вполне удовлетворительны. С решением задачи 1 программа Розенброка справилась успешно, хотя найденное при этом оптимальное решение несколько отличается от более точного решения, полученного методами скользящего допуска, НЛП, ОГМОП, МОПГ и с помощью МПБМ. Кроме того, следует отметить, что машинное время, требуемое для решения задачи 1 на основе алгоритма Розенброка, сильно отличается от затрат машинного времени, имеющих место при решении этой задачи другими (эффективными в условиях рассматриваемой задачи) методами. Отметим также, что ПОП II не обеспечил сходимости к строго оптимальному решению, а в точке, соответствующей минимуму, функция в ограничении-равенстве принимала значение 10^{-1} (вместо нуля).

Задача 4. Эта задача (так же, как и задача 6) представляет собой пример оптимизации химического состава жидкой смеси, которая должна находиться в состоянии химического равновесия. Смесь химических компонентов, поддерживаемая при постоянной температуре и при постоянном давлении, достигает состояния химического равновесия одновременно с уменьшением значения целевой функции (т. е. свободной энергии жидкой смеси) до минимума. Нелинейная целевая функция характеризуется логариф-

мической зависимостью от десяти независимых переменных при наличии трех линейных ограничений в виде равенств; нижней границей для каждой из независимых переменных является нуль.

Поскольку и алгоритм НЛП, и алгоритм скользящего допуска не исключают некоторых отклонений текущего вектора x от его допустимых значений, возникает вероятность того, что независимые переменные задачи 4 примут в ходе итерационного процесса отрицательные значения. Это в свою очередь может привести к тому, что некоторые из натуральных логарифмов в выражении для целевой функции потеряют смысл и, таким образом, оптимизационный поиск может оборваться. При решении задачи 11 алгоритм скользящего допуска никогда не приводил к отрицательным значениям никакой из десяти независимых переменных; поэтому именно с помощью этого алгоритма и была решена рассматриваемая задача (см. приложение А). Поиск же методом НЛП был преждевременно прерван, как только значение одной из независимых переменных стало отрицательным.

Чтобы преодолеть такого рода трудность, независимые переменные были «переопределены». Обозначим новые переменные через x'_j . Формула перехода от x_j к x'_j имела следующий вид:

$$x'_j = \ln x_j, \quad j = 1, \dots, 10.$$

Переопределенная таким способом задача (т. е. задача, сформулированная через новые переменные) приведена в приложении А (см. задачу 4а). Следует отметить, что задача 4а содержит три нелинейных ограничения в виде равенств вместо трех линейных ограничений-равенств исходной задачи и, следовательно, с полным основанием может быть отнесена к классу А.

После перехода к новым переменным алгоритм НЛП оказался в состоянии решить задачу 4; в результате получилось приблизительно такое же оптимальное значение целевой функции, как и в случае применения МПБМ и алгоритма МОПГ. Соответствующий оптимальному решению вектор x , найденный Брэккеном и Мак-Кормиком [5] (которые тоже использовали МПБМ), был в точности воспроизведен с помощью программы МОПГ. Однако соответствующие оптимальному решению векторы x , полученные с помощью алгоритмов скользящего допуска, НЛП и ОГМОП, несколько отличались от вектора x , полученного упомянутыми выше авторами (см. приложение А). Отсюда следует сделать вывод, что целевая функция задачи 4 обладает своего рода «широкополосным» минимумом (т. е. в точке экстремума кривизна функции выражена весьма слабо).

Задача 6. Задача 6 также связана с проблемой поиска условий химического равновесия. Однако эта задача содержит 45 независимых переменных и 16 линейных ограничений в виде равенств.

Таким образом, ее размерность значительно выше размерности задачи 4. Из используемых для решения задачи 6 четырех алгоритмов (скользящий допуск, НЛП, ОГМОП и МПБМ) три первых столкнулись с трудностью, связанной с появлением отрицательных значений выражений, стоящих под знаком логарифма (как и в задаче 4). Предпринималась попытка перейти к новым независимым переменным при использовании алгоритма НЛП (как и при решении задачи 4), а также вводились дополнительные ограничения в связи с применением алгоритма скользящего допуска (описание этого приема см. в разд. 8.6). Оказалось, что успешное решение рассматриваемой задачи достигается лишь с помощью МПБМ, хотя получаемая при этом оптимальная точка \mathbf{x} и не совпадает с оптимальной точкой, найденной Джонсом (формулировка задачи 6 принадлежит именно Джонсу); все эти решения указаны (для сравнения) в приложении А. Кроме того, следует отметить, что найденный Джонсом оптимальный вектор \mathbf{x} не удовлетворяет ограничивающим условиям, записанным в виде равенств. Минимальное значение целевой функции Джонсом не указано; однако, подставляя полученные им оптимальные значения независимых переменных в выражение для целевой функции, мы получаем $f(\mathbf{x}^*) = -79,108$. Минимальное значение $f(\mathbf{x}^*)$, найденное с помощью МПБМ, равняется $-1910,446$, т. е. является по отношению к результату, полученному Джонсом, существенно улучшенным. Это значение $f(\mathbf{x}^*)$ определено при старте из трех точек: а) из точки $x_{jk} = 10^{-1}$; б) из точки $x_{jk} = 10^{-10}$; в) из оптимальной точки, полученной Джонсом.

Для решения задачи 6 ни алгоритм Розенброка, ни ПОП II не применялись.

Задача 5. Данная задача характеризуется квадратичной целевой функцией трех независимых переменных и содержит одно нелинейное ограничение в виде равенства и одно линейное ограничение также в виде равенства. Каждая из независимых переменных может принимать лишь положительные значения. Таким образом, задача 5 относится к классу А. Для решения задач нелинейного программирования при нелинейных ограничениях, записанных в виде равенств, подходят лишь алгоритмы скользящего допуска, ОГП, МОПГ и НЛП. Именно эти четыре алгоритма и оказались в состоянии обеспечить решение задачи 5.

МПБМ не смог обеспечить решение данной задачи в том смысле, что он приводит в точку, где ограничивающие условия в виде равенств с требуемой точностью не выполняются (см. приложение А). Несколько дальнейших попыток удовлетворить ограничениям-равенствам с большей точностью также не увенчались успехом. Неспособность МПБМ обеспечить решение задачи 5 была неожиданной; однако при использовании МПБМ сходимость к оптимальному решению задачи нелинейного программирования

при ограничениях в виде равенств, вообще говоря, и не гарантируется (см. гл. 7).

Алгоритм Розенброка и ПОП II применялись для решения задачи 5 после преобразования ограничений-равенств в ограничения, имеющие вид неравенств (см. задачу 1). Ни один из этих алгоритмов не смог обеспечить сходимость к оптимальному решению. Алгоритм Розенброка прекратил работу в условиях, когда минимизационный процесс стал протекать крайне медленно (причем в стационарном режиме). ПОП II обеспечил сходимость к значению целевой функции, лежащему ниже оптимальных значений $f(x)$, полученных в случае применения алгоритмов скользящего допуска и НЛП; однако при этом ограничения-равенства оказались нарушенными.

Задача 20. Целевая функция данной задачи линейна, а число независимых переменных равняется 24. Задача содержит двенадцать нелинейных ограничений в виде равенств, два линейных ограничения, также имеющих вид равенств, и шесть нелинейных ограничений в виде неравенств. Независимые переменные могут принимать только положительные значения. Из всех задач, приведенных в приложении А, данная задача, возможно, является наиболее трудной. Никаких попыток применить алгоритм Розенброка или ПОП II для решения задачи 20 не предпринималось.

Алгоритмы скользящего допуска, НЛП, ОГМОП и МПБМ оказались в состоянии улучшить значение целевой функции по сравнению со значением $f(x)$ в стартовой точке; однако сходимость к решению задачи была достигнута лишь при использовании алгоритмов скользящего допуска и МОПГ (1970). Ни алгоритм НЛП, ни МПБМ сходимость к искомому решению обеспечить не смогли и продолжали «осциллировать» относительно некоторой наилучшей из текущих точек, которую они смогли определить, пока не было исчерпано отведенное на решение задачи машинное время. Более того, наилучшие решения, полученные с помощью алгоритмов НЛП и МПБМ, оказались менее точными, нежели решения, полученные с помощью алгоритма скользящего допуска и МОПГ. Кроме того, решение, найденное с помощью МПБМ, не удовлетворяло ограничениям-равенствам с достаточной степенью точности.

9.3.2. ОЦЕНКА ЭФФЕКТИВНОСТИ АЛГОРИТМОВ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ НА ОСНОВЕ АНАЛИЗА РЕЗУЛЬТАТОВ ИХ ПРИМЕНЕНИЯ ПРИ РЕШЕНИИ ТЕСТОВЫХ ЗАДАЧ

В данном разделе дается характеристика каждого из рассмотренных выше алгоритмов нелинейного программирования с учетом всех данных, полученных в ходе применения этих алгоритмов для решения тестовых задач, перечисленных в табл. 9.3.5.

1. МОПГ, вариант 1969 г. С точки зрения вычислительной эффективности результаты, полученные при решении методом МОПГ задач нелинейного программирования (особенно наиболее трудных задач, относящихся к категории А), представляются весьма обнадеживающими (заметим, что задача 20 была решена с помощью невырожденного варианта программы МОПГ, относящегося к 1970 г.). Попытки решить с помощью алгоритма МОПГ задачи класса С (задачи 9, 11, 14 и 16) оказались безуспешными из-за преждевременного останова; тем не менее задачи 11 и 14 были решены позднее Абади. Во всех случаях, когда оптимум был найден, временные затраты, связанные с решением каждой из указанных выше задач, были значительно ниже соответствующих временных затрат, имевших место при использовании других методов; поэтому машинная программа МОПГ является одной из самых популярных и получила широкое распространение.

2. Алгоритм скользящего допуска. Данный алгоритм оказался в состоянии обеспечить решение задач всех пяти классов (А, В, С, Д и Е); исключение составили лишь две задачи, решение которых с помощью алгоритма скользящего допуска получить не удалось. Задачи 6 и 18 обсуждались в подразд. 9.3.1. Следует подчеркнуть, что при использовании алгоритма скользящего допуска было затрачено слишком много времени на поиск точки, удовлетворяющей ограничениям-равенствам, так что при решении этих задач нельзя было уложиться в приемлемое время. Поскольку речь идет о методе прямого поиска, для реализации алгоритма на ЭВМ находить производные (фигурирующих в задачах функций) в аналитическом виде не требуется. Поэтому временные затраты на выполнение подготовительных операций для всех задач оказываются весьма незначительными. Более того, в процессе выполнения подготовительных работ ошибки почти исключаются. С другой стороны, затраты машинного времени при решении задач большой размерности (например, задач 18, 19 и 20) оказываются более значительными, чем в случае, когда оптимизация осуществляется с помощью алгоритмов МОПГ и НЛП.

3. Алгоритм НЛП. С помощью данного алгоритма удалось решить большинство тестовых задач классов В, С, Д и Е; исключение составили задачи 6, 14 и 16. Однако следует отметить, что после перехода в задаче 6 к новым переменным (в результате которого 16 линейных ограничений-равенств превратились в 16 ограничений-равенств нелинейной структуры) ее все же удалось решить. В этой связи следует заметить, что, строго говоря, задачу 6 нужно рассматривать как задачу класса А. Работу алгоритма НЛП в процессе решения задач класса А нельзя назвать достаточно эффективной; сходимость в случае задачи 20 алгоритмом НЛП не была обеспечена. Как известно, при использовании алгоритма НЛП требуется знать частные производные первого

порядка для $f(x)$, $h_i(x)$ и $g_i(x)$ в аналитической записи. Это приводит к тому, что временные затраты на выполнение подготовительных операций оказываются большими и повышается частота возникновения ошибок при подготовке задачи к решению (когда подготовительные операции выполняются вручную). Временные затраты, связанные с реализацией самого вычислительного процесса на ЭВМ, ниже по сравнению со случаем использования метода скользящего допуска (хотя и имеются некоторые исключения).

4. МПБМ (вариант, относящийся к 1968 г.). Результаты, полученные с помощью МПБМ, вполне соответствуют структуре математического аппарата, характерного для метода последовательной безусловной минимизации (см. гл. 7). С помощью МПБМ удалось решить все тестовые задачи классов В, С, D и E, за исключением тех задач, при подготовке которых к решению получаются ошибки при вычислении вторых частных производных $f(x)$, $h_i(x)$ и $g_i(x)$, и, таким образом, окончательное решение оказывается неправильным. МПБМ не смог, однако, отыскать оптимальное решение ни одной из задач класса А. Более того, разрывный характер первых производных в задаче 15 сделал вообще невозможным применить МПБМ для отыскания оптимального решения. Но в этой связи следует напомнить, что МПБМ для задач класса А не гарантирует сходимость к оптимальному решению (см. гл. 7). Необходимость в вычислении первых и вторых частных производных функций $f(x)$, $h_i(x)$ и $g_i(x)$, возникающая в случае применения МПБМ, приводит к значительному количеству ошибок, допускаемых в процессе выполнения вычислительных операций вручную, и увеличивает временные затраты на стадии подготовки задачи к решению на ЭВМ. К счастью, целевые функции некоторых из тестовых задач симметричны по независимым переменным (например, задачи 4, 6, 10, 18 и 19), так что проблема нахождения первых и вторых частных производных оказывается существенно упрощенной. Затраты машинного времени при решении с помощью МПБМ задач, относящихся к классам В, С, D и E, вообще говоря, ниже затрат машинного времени при поиске решения прямыми методами.

5. Алгоритм Розенброка. Данный алгоритм справился с решением лишь некоторых задач, принадлежащих к классам С, D и E. Причина имевших место неудач объясняется прежде всего преждевременным прекращением вычислительного процесса из-за слишком медленной сходимости в ходе перемещений вдоль ограничений задачи.

Зарегистрирован один случай, когда алгоритм Розенброка смог обеспечить решение задачи класса В (а именно задачи 1). Однако тот факт, что даже при решении такой простой задачи (число переменных задачи равняется двум) потребовалось слишком много машинного времени, наводит на мысль о том, что

попытки решать с помощью алгоритма Розенброка задачи класса В бесполезны. Неудачная попытка применения метода Розенброка при решении задачи 5, являющейся простейшей задачей класса А, говорит о том, что алгоритм Розенброка вообще плохо приспособлен для решения задач указанного класса.

6. ПОП II. Попытки применить ПОП II при решении тестовых задач приложения А свидетельствуют о его незначительных воз-

Таблица 9.3.7

Сравнение машинных программ, основанных на использовании различных алгоритмов нелинейного программирования

Алгоритм (и соответствующая машинная программа)	Класс задачи				
	A	B	C	D	E
МОПГ (1970)	Да	Да	Да	Да	Да
Скользящий допуск	Да ¹⁾	Да	Да	Да	Да
НЛП	?	Да	Да	Да	Да
МПБМ (1968)	Нет	Да	Да	Да	Да
Розенброка	Нет	Нет	Нет	Нет	Да
ПОП II ²⁾	Нет	Нет	Нет	Нет	Да
ОГМОП	Нет	?	Нет	?	Да

1) За исключением случаев, когда задача содержит большое число ограничений в виде равенств.

2) Работу алгоритма можно улучшить путем выбора соответствующих значений параметров.

möglichkeiten: с помощью ПОП II удалось решить лишь четыре из двенадцати задач. Эти результаты совершенно отличаются от результатов, опубликованных Колвиллом. Поскольку ПОП II не предназначался для решения задач, относящихся к классам В и А, неспособность данной машинной программы обеспечить решение задач указанных классов не была неожиданностью. Низкая же эффективность ПОП II при решении задач классов D и C вызвала, однако, некоторое недоумение.

7. ОГМОП. Данный алгоритм, как правило, прекращал работу преждевременно; причины преждевременных остановов были самыми различными (на подробном описании этих причин мы здесь не останавливаемся). Отметим лишь, что нередко составляющие вектора, задающего направление поиска, или же составляющие градиента оказывались равными нулю либо в силу случайного стечения обстоятельств, либо в результате округления получаемых числовых значений.

Данные относительно возможностей семи машинных программ, предназначенных для решения задач нелинейного программирования, приведены в табл. 9.3.7. Эта таблица может оказать некоторую помощь тем, кто сталкивается с проблемой выбора типа программы в связи с решением той или иной задачи нелинейного

программирования. Если в ячейке табл. 9.3.7 стоит «да», то это означает, что соответствующий алгоритм можно в принципе считать способным решать задачи нелинейного программирования указанного (в соответствующем столбце) класса. Если же в той или иной ячейке стоит «нет», то это означает, что соответствующий алгоритм не рекомендуется использовать при решении задач, относящихся к тому классу, который указан в соответствующем столбце. Знак вопроса означает, что относительно эффективности соответствующего алгоритма (в связи с его применением при решении задач соответствующего класса) нельзя пока сделать определенного вывода.

ЛИТЕРАТУРА

1. Stocker D. C., A Comparative Study of Nonlinear Programming Codes, M. S. Thesis, The Univ. of Texas, Austin, Tex., 1969.
2. Holzman A. G., Comparative Analysis of Nonlinear Programming Codes with the Weisman Algorithm, SRCC Rept. 113, Univ. of Pittsburgh, Pittsburgh, Pa., Nov. 1969.
3. Colville A. R., IBM N. Y. Sci. Center Rept. 320-2949, June 1968.
4. Sauer R. N., Colville A. R., Burwick C. W., *Hydrocarbon Process. Petrol. Refiner*, **43**, 85 (1964).
5. Bracken J., McCormick G. P., Selected Applications of Nonlinear Programming, Wiley, Inc., N. Y., 1968.

ДОПОЛНИТЕЛЬНАЯ ЛИТЕРАТУРА

- Rastrigin L. A., Criteria for Comparing Methods of Seeking an Extremum (English Trans.), *Zavod Lab.*, **32**, 1248, 1529 (1966).
Rosen J. B., Suzuki S., Construction of Nonlinear Programming Test Problems, *Commun. ACM*, **8**, 113 (1965).

Приложение А

ЗАДАЧИ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ И ИХ РЕШЕНИЯ¹⁾

•

Задача 1 [1].

Число переменных равняется двум.

Задача содержит ограничение в виде равенства и одно ограничение в виде неравенства.

Требуется

минимизировать $f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$
при ограничениях

$$h_1(\mathbf{x}) = x_1 - 2x_2 + 1 = 0,$$

$$g_1(\mathbf{x}) = -\frac{x_1^2}{4} - x_2^2 + 1 \geq 0.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [2 \ 2]^T$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = 1$.

Решение:

$$f(\mathbf{x}^*) = 1,393,$$

$$x_1^* = 0,823,$$

$$x_2^* = 0,911.$$

Задача 2 [2].

Число переменных равняется двум.

Ограничения отсутствуют.

Требуется

минимизировать $f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$.

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [-1,2 \ 1]^T$. В этой точке $f(\mathbf{x}^{(0)}) = 24,20$.

Решение:

$$\mathbf{x}^* = [1 \ 1]^T, \quad f(\mathbf{x}^*) = 0.$$

Задача 3 [3].

Число переменных равняется двум.

Задача содержит три нелинейных ограничения в виде неравенств и четыре предельных ограничения для значений, которые могут принимать независимые переменные.

¹⁾ Источники, из которых заимствованы рассматриваемые ниже задачи, указаны в конце приложения.

Требуется

$$\begin{aligned}
 & \text{максимизировать } f(\mathbf{x}) = 75,196 - 3,8112x_1 + 0,12694x_1^2 - \\
 & - 2,0567 \cdot 10^{-3}x_1^3 + 1,0345 \cdot 10^{-5}x_1^4 - 6,8306x_2 + \\
 & + 0,030234x_1x_2 - 1,28134 \cdot 10^{-3}x_2x_1^2 + 3,5256 \cdot 10^{-5}x_2x_1^3 - \\
 & - 2,266 \cdot 10^{-7}x_2x_1^4 + 0,25645x_2^2 - 3,4604 \cdot 10^{-3}x_2^3 + \\
 & + 1,3514 \cdot 10^{-5}x_2^4 - \frac{28,106}{x_2 + 1} - 5,2375 \cdot 10^{-6}x_1^2x_2^2 - 6,3 \times \\
 & \times 10^{-8}x_1^3x_2^2 + 7 \cdot 10^{-10}x_1^3x_2^3 + 3,4054 \cdot 10^{-4}x_1x_2^2 - 1,6638 \times \\
 & \times 10^{-6}x_1x_2^3 - 2,8673 \exp(0,0005x_1x_2)
 \end{aligned}$$

при ограничениях

$$0 \leq x_1 \leq 75,$$

$$0 \leq x_2 \leq 65,$$

$$x_1x_2 - 700 \geq 0,$$

$$x_2 - 5\left(\frac{x_1}{25}\right)^2 \geq 0,$$

$$(x_2 - 50)^2 - 5(x_1 - 55) \geq 0.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [90 \ 10]^T$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = -82,828$.

Решение:

$$\mathbf{x}^* = [75 \ 65]^T, \quad f(\mathbf{x}^*) = 58,903.$$

Задача 4 [1].

Число переменных равняется десяти.

Задача содержит три линейных ограничения в виде равенств и десять предельных ограничений для значений, которые могут принимать независимые переменные.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right),$$

где $c_1 = -6,089$, $c_2 = -17,164$, $c_3 = -34,054$, $c_4 = -5,914$, $c_5 = -24,721$, $c_6 = -14,986$, $c_7 = -24,100$, $c_8 = -10,708$, $c_9 = -26,662$, $c_{10} = -22,179$.

Ограничения имеют следующий вид:

$$h_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0,$$

$$h_2(\mathbf{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0,$$

$$h_3(\mathbf{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0,$$

$$x_i \geq 0, \quad i = 1, \dots, 10.$$

В качестве начальной берется точка $x_i^{(0)} = 0,1$ ($i = 1, \dots, 10$), лежащая вне допустимой области. В этой точке $f(x^{(0)}) = -20,961$.

Решение: см. задачу 4а.

Задача 4а.

Число независимых переменных равняется десяти.

Задача содержит три нелинейных ограничения в виде равенств. Требуется

$$\text{минимизировать } f(x') = \sum_{i=1}^{10} \left\{ e^{x'_i} \left[c_i + x'_i - \ln \left(\sum_{i=1}^{10} e^{x'_i} \right) \right] \right\}$$

при ограничениях

$$h_1(x') = e^{x'_1} + 2e^{x'_2} + 2e^{x'_3} + e^{x'_6} + e^{x'_{10}} - 2 = 0,$$

$$h_2(x') = e^{x'_4} + 2e^{x'_5} + e^{x'_6} + e^{x'_7} - 1 = 0,$$

$$h_3(x') = e^{x'_3} + e^{x'_7} + e^{x'_8} + 2e^{x'_9} + e^{x'_{10}} - 1 = 0.$$

В качестве начальной берется точка $x_i = -2,3$ ($i = 1, \dots, 10$), лежащая вне допустимой области.

Решение:

	НЛП	Метод скользящего допуска	ОГМОП	МОПГ	МПБМ
$f(x)$	-47,751	-47,736	-47,656	-47,761	-47,761
x_1	0,0350	0,0128	0	0,0406	0,0407
x_2	0,1142	0,1433	0,1695	0,1477	0,1477
x_3	0,8306	0,8078	0,7536	0,7832	0,7832
x_4	0,0012	0,0062	0	0,0014	0,0014
x_5	0,4887	0,4790	0,5000	0,4853	0,4853
x_6	0,0005	0,0033	0	0,0007	0,0007
x_7	0,0209	0,0324	0	0,0274	0,0274
x_8	0,0157	0,0281	0	0,0180	0,0180
x_9	0,0289	0,0250	0,0464	0,0375	0,0373
x_{10}	0,0751	0,0817	0,1536	0,0969	0,0969
$h_1(x)$	3.E - 12	3.E - 05	0	1.E - 06	-8.E - 08
$h_2(x)$	3.E - 12	2.E - 05	0	1.E - 06	-1.E - 07
$h_3(x)$	2.E - 11	9.E - 05	0	1.E - 06	-1.E - 07

Задача 5 [4].

Число переменных равняется трем.

Задача содержит одно нелинейное ограничение в виде равенства, одно линейное ограничение в виде равенства и три предельных ограничения для значений, которые могут принимать независимые переменные.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

при ограничениях

$$h_1(\mathbf{x}) = x_1^2 + x_2^2 + x_3^2 - 25 = 0,$$

$$h_2(\mathbf{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0,$$

$$x_i \geq 0, \quad i = 1, 2, 3.$$

В качестве начальной выбирается точка $x_i^{(0)} = 2$; другой вариант: $x_i^{(0)} = 10$ ($i = 1, 2, 3$). Обе точки лежат вне допустимой области. В любой из этих точек $f(\mathbf{x}^{(0)}) = 976$.

Решение:

$$f(\mathbf{x}^*) = 961,715, \quad x_1 = 3,512, \quad x_2 = 0,217, \quad x_3 = 3,552,$$

$$h_1(\mathbf{x}^*) = 0, \quad h_2(\mathbf{x}^*) = 0.$$

Задача 6 [5].

Число переменных равняется 45.

Задача содержит 16 линейных ограничений в виде равенств. Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{k=1}^7 \left[\sum_{j=1}^{n_k} x_{jk} \left(c_{jk} + \ln \frac{x_{jk}}{\sum_{l=1}^{n_k} x_{jl}} \right) \right]$$

при ограничениях

$$h_i(\mathbf{x}) = \sum_{k=1}^7 \left(\sum_{j=1}^{n_k} E_{ijk} x_{jk} \right) - b_i = 0, \quad i = 1, \dots, 16,$$

$$x_{jk} \geq 0, \quad j = 1, \dots, n_k, \quad k = 1, \dots, 7.$$

(Примечание. Значения b_i и c_{jk} указаны в приведенной ниже таблице.) В качестве начальной выбирается точка $x_{jk} = 0,1$, $j = 1, \dots, n_k$, $k = 1, \dots, 7$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = -30,958$.

Решение:

	НЛП	МПБМ	МПБМ (Джонса)
$f(\mathbf{x}^*)$	-1909,740	-1910,361	-79,108
x_{11}^*	7,854E - 07	6,599E - 06	6,440E - 01
x_{21}^*	8,078E - 02	2,512E - 01	2,590E - 01
x_{31}^*	3,706E - 00	3,705E - 00	3,705E - 00

	НЛП	МПВМ	МПВМ (Джонса)
x_{41}^*	8,855E — 02	2,535E — 01	2,997E — 01
x_{12}^*	6,894E — 01	6,529E — 01	5,617E — 05
x_{22}^*	3,020E — 02	1,235E — 03	6,880E — 04
x_{32}^*	1,398E — 04	3,667E — 04	2,062E — 04
x_{42}^*	1,626E — 04	2,794E — 06	1,101E — 06
x_{52}^*	0	5,441E — 06	2,433E — 06
x_{62}^*	2,782E — 02	7,363E — 02	5,715E — 02
x_{72}^*	7,950E — 02	8,791E — 02	7,938E — 02
x_{82}^*	3,421E — 02	3,542E — 02	3,231E — 03
x_{92}^*	2,486E + 01	4,458E + 01	2,839E — 01
$x_{10,2}^*$	3,873E — 02	2,669E — 02	1,388E — 02
$x_{11,2}^*$	1,500E — 04	7,709E — 06	3,283E — 06
$x_{12,2}^*$	1,170E — 05	3,764E — 05	1,738E — 05
$x_{13,2}^*$	1,550E — 02	1,550E — 02	1,155E — 02
x_{13}^*	0	9,900E — 07	5,956E — 05
x_{23}^*	2,649E — 02	5,077E — 05	4,419E — 04
x_{33}^*	1,251E — 04	3,107E — 05	2,205E — 04
x_{43}^*	1,064E — 01	1,546E — 06	1,095E — 06
x_{53}^*	0	3,102E — 06	1,852E — 06
x_{63}^*	5,253E — 02	6,416E — 03	2,291E — 02
x_{73}^*	8,710E — 03	2,202E — 04	8,751E — 03
x_{83}^*	1,471E — 02	1,287E — 02	4,506E — 02
x_{93}^*	4,735E — 02	2,165E — 00	1,832E — 01
$x_{10,3}^*$	9,208E — 02	2,675E — 00	6,396E — 03
$x_{11,3}^*$	3,119E — 04	3,437E — 06	2,855E — 06
$x_{12,3}^*$	1,560E — 02	1,400E — 05	7,806E — 06
$x_{13,3}^*$	2,421E — 02	1,927E — 02	2,113E — 02
$x_{14,3}^*$	2,448E — 03	1,855E — 03	7,429E — 06

	НЛП	МПБМ	МПБМ (Джонса)
$x_{15,3}^*$	8,398E — 03	3,264E — 06	3,017E — 05
$x_{16,3}^*$	5,285E — 03	7,579E — 07	5,056E — 05
$x_{17,3}^*$	0	3,510E — 07	4,871E — 05
$x_{18,3}^*$	1,601E — 03	2,513E — 07	2,142E — 03
x_{14}^*	4,968E — 07	0	2,337E — 06
x_{24}^*	1,978E — 02	4,200E — 07	1,821E — 04
x_{34}^*	6,271E — 03	7,063E — 06	8,583E — 05
x_{15}^*	5,328E — 02	0	2,355E — 05
x_{25}^*	0	0	1,251E — 03
x_{35}^*	0	1,305E — 06	7,573E — 03
x_{16}^*	2,510E — 02	1,465E — 05	3,038E — 04
x_{26}^*	1,220E — 06	1,382E — 05	3,902E — 05
x_{17}^*	0	2,872E — 06	2,879E — 02
x_{27}^*	0	2,476E — 06	1,499E — 03
$h_1(x^*)$	5,118E — 02	2,529E — 07	—4,800E — 07
$h_2(x^*)$	2,407E — 03	2,263E — 07	1,592E — 06
$h_3(x^*)$	2,559E — 05	1,917E — 07	2,631E — 06
$h_4(x^*)$	4,493E — 02	1,112E — 06	—4,624E + 01
$h_5(x^*)$	2,389E — 02	—4,518E — 07	—4,624E + 01
$h_6(x^*)$	3,100E — 04	—3,946E — 07	1,340E — 01
$h_7(x^*)$	6,692E — 06	6,771E — 07	1,362E — 06
$h_8(x^*)$	6,376E — 04	5,101E — 07	1,362E — 06
$h_9(x^*)$	0	—1,869E — 07	—3,948E — 03
$h_{10}(x^*)$	2,082E — 02	—1,950E — 07	2,280E — 03
$h_{11}(x^*)$	—2,273E — 03	—2,273E — 03	—2,273E — 03
$h_{12}(x^*)$	—1,380E — 02	—5,583E — 07	—2,699E — 04
$h_{13}(x^*)$	6,946E — 04	6,001E — 07	—8,575E — 03
$h_{14}(x^*)$	5,331E — 02	1,169E — 06	8,847E — 03
$h_{15}(x^*)$	7,789E — 06	2,180E — 07	—5,529E — 07
$h_{16}(x^*)$	3,528E — 09	1,284E — 07	—3,330E — 07

Значения b_i и c_{jk} в задаче 6

i	b_i	j	k	c_{jk}	j	k	c_{jk}
1	0,6529581	1	1	0,0	6	3	0,0
2	0,281941	2	1	-7,69	7	3	2,2435
3	3,705233	3	1	-11,52	8	3	0,0
4	47,00022	4	1	-36,60	9	3	-39,39
5	47,02972	1	2	-10,94	10	3	-21,49
6	0,08005	2	2	0,0	11	3	-32,84
7	0,08813	3	2	0,0	12	3	6,12
8	0,04829	4	2	0,0	13	3	0,0
9	0,0155	5	2	0,0	14	3	0,0
10	0,0211275	6	2	0,0	15	3	-1,9028
11	0,0022725	7	2	0,0	16	3	-2,8889
12	0,0	8	2	2,5966	17	3	-3,3622
13	0,0	9	2	-39,39	18	3	-7,4854
14	0,0	10	2	-21,35	1	4	-15,639
15	0,0	11	2	-32,84	2	4	0,0
16	0,0	12	2	6,26	3	4	21,81
		13	2	0,0	1	5	-16,79
		1	3	10,45	2	5	0,0
		2	3	0,0	3	5	18,9779
		3	3	-0,50	1	6	0,0
		4	3	0,0	2	6	11,959
		5	3	0,0	1	7	0,0
					2	7	12,899

Данные для определения E_{ijk} в задаче 6

$i \backslash j$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x_{11}	1															
x_{21}		1														
x_{31}			1													
x_{41}				1	1											
x_{12}	1															
x_{22}		1														
x_{32}			1													
x_{42}				1												
x_{52}					1											
x_{82}						1										
x_{72}							1									
x_{82}								1								

Продолжение табл.

x_{jk}	t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
x_{92}						1	1										
$x_{10,2}$		1				1									—1		
$x_{11,2}$		1			1	1											
$x_{12,2}$		1			—1	1									—2		
$x_{13,3}$										1					—1		
x_{13}	1		1														
x_{23}				1													
x_{33}					1												
x_{43}						1											
x_{53}							1										
x_{63}								1									
x_{73}									1								
x_{83}										1							
x_{93}						1	1										
$x_{10,3}$		1				1											
$x_{11,3}$		1				1	1										
$x_{12,3}$		1			—1	1											
$x_{13,3}$												1			—4		
$x_{14,3}$													1		—3		
$x_{15,3}$	1												1		—1		
$x_{16,3}$	2												1		—2		
$x_{17,3}$	3												1		—1		
$x_{18,3}$	4							1					1		—4		
x_{14}									1							1	
x_{24}										—1						1	
x_{34}										1						1	
x_{15}											—1				1		
x_{25}											1				1		
x_{35}											—1				1		
x_{16}												—1			1		
x_{26}		1				—1									1		
x_{17}		1				—1										1	
x_{27}		1				—1									1		

Задача 7 [6].

Число переменных равняется трем.

Задача содержит 14 нелинейных ограничений в виде неравенств и шесть предельных ограничений для значений, которые могут принимать независимые переменные.

Это одна из типичных задач, при решении которых целевые функции и функции, задающие ограничения, описываются с помощью автономной машинной подпрограммы.

Требуется

максимизировать $f(x) = 0,063y_2y_5 - 5,04x_1 - 3,36y_3 - 0,035x_2 - 10x_3$

при ограничениях

$$\begin{aligned}0 &\leq x_1 \leq 2000, \\0 &\leq x_2 \leq 16000, \\0 &\leq x_3 \leq 120, \\0 &\leq y_2 \leq 5000, \\0 &\leq y_3 \leq 2000, \\85 &\leq y_4 \leq 93, \\90 &\leq y_5 \leq 95, \\3 &\leq y_6 \leq 12, \\0,01 &\leq y_7 \leq 4, \\145 &\leq y_8 \leq 162.\end{aligned}$$

Описание процедуры вычисления y_2 , y_3 , y_4 , y_5 , y_6 , y_7 и y_8 на языке ФОРТРАН выглядит следующим образом:

```

Y(2) = 1.6*X(1)
10 Y(3) = 1.22*Y(2) - X(1)
      Y(6) = (X(2) + Y(3))/X(1)
      Y2CALC = X(1)*(112. + 13.167*Y(6) - 0.6667*Y(6)**2)/100.
      IF(ABS(Y2CALC - Y(2)) - 0.001) 30,30,20
20 Y(2) = Y2CALC
      GO TO 10
30 CONTINUE
      Y(4) = 93.
100 Y(5) = 86.35 + 1.098*Y(6) - 0.038*Y(6)**2 + 0.325*(Y(4) - 89.)
      Y(8) = -133. + 3.*Y(5)
      Y(7) = 35.82 - 0.222*Y(8)
      Y4CALC = 98000.*X(3)/(Y(2)*Y(7) + X(3)*1000.)
      IF(ABS(Y4CALC - Y(4)) - 0.0001) 300,300,200
200 Y(4) = Y4CALC
      GO TO 100
300 CONTINUE
```

В качестве начальной выбирается допустимая точка $\mathbf{x}^{(0)} = [1745 \ 12000 \ 110]^T$. В этой точке $f(\mathbf{x}^{(0)}) = 868,6458$.

Решение:

$$\mathbf{x}^* = [1728,37 \ 16\,000 \ 98,13]^T,$$

$$f(\mathbf{x}^*) = 1162,036.$$

Задача 8 [7].

Число переменных равняется четырем.

Задача содержит восемь предельных ограничений для значений, которые могут принимать независимые переменные, и характеризуется наличием неоптимальной стационарной точки при $f(\mathbf{x}) \approx 8$, обеспечивающей быструю сходимость.

По условию задачи требуется

$$\text{минимизировать } f(\mathbf{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + \\ + (1 - x_3)^2 + 10,1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19,8(x_2 - 1)(x_4 - 1)$$

при ограничениях

$$-10 \leq x_i \leq 10, \quad i = 1, 2, 3, 4.$$

В качестве начальной берется допустимая точка

$$\mathbf{x}^{(0)} = [-3 \ -1 \ -3 \ -1]^T,$$

в которой $f(\mathbf{x}^{(0)}) = 19\,192$.

Решение:

$$\mathbf{x}^* = [1 \ 1 \ 1 \ 1]^T,$$

$$f(\mathbf{x}^*) = 0.$$

Задача 9 [8].

Число переменных равняется четырем.

Задача содержит одно нелинейное ограничение в виде неравенства.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^{19} (y_{i,\text{выч}} - y_{i,\text{набл}})^2,$$

$$y_{i,\text{выч}} = \frac{x_3 \beta^{x_2} \left(\frac{x_2}{6,2832} \right)^{1/2} \left(\frac{c_i}{7,658} \right)^{x_2-1} \exp \left(x_2 - \beta \frac{c_i x_2}{7,658} \right)}{1 + \frac{1}{12x_2}} + \\ + \frac{(1 - x_3) \left(\frac{\beta}{x_4} \right)^{x_1} \left(\frac{x_1}{6,2832} \right)^{1/2} \left(\frac{c_i}{7,658} \right)^{x_1-1} \exp \left(x_1 - \beta \frac{c_i x_1}{7,658 x_4} \right)}{1 + \frac{1}{12x_1}},$$

где $\beta = x_3 + (1 - x_3)x_4$. (Примечание. Значения c_i и $y_{i,\text{набл}}$ указаны в приведенной ниже таблице.)

Ограничения задачи имеют следующий вид:

$$x_3 + (1 - x_3)x_4 \geq 0,$$

$$x_4 \geq 0,$$

$$x_3 \leq 1,$$

$$i = 1, \dots, 4.$$

В качестве начальной выбирается точка

$$\mathbf{x}^{(0)} = [2 \quad 4 \quad 0,04 \quad 2]^T,$$

в которой $f(\mathbf{x}^{(0)}) = 4,8024$.

Решение:

$$\mathbf{x}^* = [12,277 \quad 4,632 \quad 0,313 \quad 2,029]^T,$$

$$f(\mathbf{x}) = 0,0075.$$

Значения c_i и $y_{i,\text{набл}}$ в задаче 9

i	o	$y_i, \text{набл}$	t	c	$y_{t,\text{набл}}$
1	0,1	0,00189	11	10	0,702
2	1	0,1038	12	11	0,528
3	2	0,268	13	12	0,385
4	3	0,506	14	13	0,257
5	4	0,577	15	14	0,159
6	5	0,604	16	15	0,0869
7	6	0,725	17	16	0,0453
8	7	0,898	18	17	0,01509
9	8	0,947	19	18	0,00189
10	9	0,845			

Задача 10 [9].

Число переменных равняется пяти.

Задача содержит десять линейных ограничений в виде неравенств и пять предельных условий для значений, которые могут принимать независимые переменные.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{j=1}^5 e_j x_j + \sum_{i=1}^5 \sum_{j=1}^5 c_{ij} x_i x_j + \sum_{j=1}^5 d_j x_j^3$$

при ограничениях

$$\sum_{j=1}^5 a_{ij}x_j - b_i \geq 0, \quad i = 1, \dots, 10,$$

$$x_j \geq 0, \quad j = 1, \dots, 5.$$

(Примечание. Значения e_j , c_{ij} , d_j , a_{ij} и b_i указаны в приведенной ниже таблице.)

В качестве начальной выбирается допустимая точка $x^{(0)} = [0\ 0\ 0\ 0\ 1]^T$, в которой $f(x^{(0)}) = 20$.

Решение:

$$x^* = [0,3000 \quad 0,3335 \quad 0,4000 \quad 0,4285 \quad 0,224]^T,$$

$$f(x^*) = -32,349.$$

Исходные данные для задач 10 и 18

$j \backslash i$	1	2	3	4	5
e_j	-15	-27	-36	-18	-12
c_{1j}	30	-20	-10	32	-10
c_{2j}	-20	39	-6	-31	32
c_{3j}	-10	-6	10	-6	-10
c_{4j}	32	-31	-6	39	-20
c_{5j}	-10	32	-10	-20	30
d_j	4	8	10	6	2
a_{1j}	-16	2	0	1	0
a_{2j}	0	-2	0	0,4	2
a_{3j}	-3,5	0	2	0	0
a_{4j}	0	-2	0	-4	-1
a_{5j}	0	-9	-2	1	-2,8
a_{6j}	2	0	-4	0	0
a_{7j}	-1	-1	-1	-1	-1
a_{8j}	-1	-2	-3	-2	-1
a_{9j}	1	2	3	4	5
a_{10j}	1	1	1	1	1
b_1	-40				
b_2	-2				
b_3	-0,25				
b_4	-4				
b_5	-4				
b_6	-1				
b_7	-40				
b_8	-60				
b_9	5				
b_{10}	1				

Задача 11 [10].

Число переменных равняется пяти.

Задача содержит шесть нелинейных ограничений в виде неравенств и десять предельных ограничений для значений, которые могут принимать независимые переменные.

Заметим, что в структуру $f(x)$ переменные x_2 и x_4 не входят.

Требуется

$$\begin{aligned} \text{минимизировать } f(x) = & 5,3578547x_3^2 + 0,8356891x_1x_5 + \\ & + 37,293239x_1 - 40792,141 \end{aligned}$$

при ограничениях

$$\begin{aligned} 0 \leqslant & 85,334407 + 0,0056858x_2x_5 + 0,0006262x_1x_4 - \\ & - 0,0022053x_3x_5 \leqslant 92, \end{aligned}$$

$$90 \leqslant 80,51249 + 0,0071317x_2x_5 + 0,0029955x_1x_2 + 0,0021813x_3^2 \leqslant 110,$$

$$\begin{aligned} 20 \leqslant & 9,300961 + 0,0047026x_3x_5 + 0,0012547x_1x_3 + \\ & + 0,0019085x_3x_4 \leqslant 25, \end{aligned}$$

$$78 \leqslant x_1 \leqslant 102,$$

$$33 \leqslant x_2 \leqslant 45,$$

$$27 \leqslant x_3 \leqslant 45,$$

$$27 \leqslant x_4 \leqslant 45,$$

$$27 \leqslant x_5 \leqslant 45.$$

В качестве начальной выбирается допустимая точка

$$x^{(0)} = [78,62 \quad 33,44 \quad 31,07 \quad 44,18 \quad 35,22]^T,$$

в которой $f(x^{(0)}) = -30\,367$.

Решение:

$$x^* = [78,000 \quad 33,000 \quad 29,995 \quad 45,000 \quad 36,776]^T,$$

$$f(x^*) = -30\,665,5.$$

В качестве начальной можно взять точку $x^{(0)} = [78 \quad 33 \quad 27 \quad 27 \quad 27]^T$, лежащую вне допустимой области. В этой точке $f(x^{(0)}) = -32\,217$.

При этом также приходим к решению

$$x^* = [78,000 \quad 33,000 \quad 29,995 \quad 45,000 \quad 36,776]^T,$$

$$f(x^*) = -30\,665,5.$$

Задача 12 [3].

Число переменных равняется пяти.

Задача содержит 4 линейных ограничения в виде неравенств, 34 нелинейных ограничений в виде неравенств (причем оказывается, что некоторые из них можно исключить из условия задачи) и 10 ограничений, задающих нижний и верхний пределы изменения независимых переменных.

Требуется

$$\begin{aligned} \text{максимизировать } f(x) = & 0,0000005843y_{17} - 0,000117y_{14} - 0,1365 - \\ & - 0,00002358y_{13} - 0,000001502y_{16} - 0,0321y_{12} - 0,004324y_5 - \\ & - 0,0001 \frac{c_{15}}{c_{18}} - 37,48 \frac{y_2}{c_{12}}. \end{aligned}$$

Величины y_i и c_i определяются следующими соотношениями:

$$y_1 = x_2 + x_3 + 41,6,$$

$$c_1 = 0,024x_4 - 4,62,$$

$$y_2 = \frac{12,5}{c_1} + 12,0,$$

$$c_2 = 0,0003535x_1^2 + 0,5311x_1 + 0,08705y_2x_1,$$

$$c_3 = 0,052x_1 + 78 + 0,002377y_2x_1,$$

$$y_3 = \frac{c_2}{c_3},$$

$$y_4 = 19y_3,$$

$$c_4 = 0,04782(x_1 - y_3) + \frac{0,1956(x_1 - y_3)^2}{x_3} + 0,6376y_4 + 1,594y_3,$$

$$c_5 = 100x_2,$$

$$c_6 = x_1 - y_3 - y_4,$$

$$c_7 = 0,950 - \frac{c_4}{c_6},$$

$$y_5 = c_6c_7,$$

$$y_6 = x_1 - y_5 - y_4 - y_3,$$

$$c_8 = (y_5 + y_4)0,995,$$

$$y_7 = \frac{c_8}{y_1},$$

$$y_8 = \frac{c_8}{3798},$$

$$c_9 = y_7 - \frac{0,0663y_7}{y_8} - 0,3153,$$

$$y_9 = \frac{96,82}{c_9} + 0,321y_1,$$

$$y_{10} = 1,29y_5 + 1,258y_4 + 2,29y_3 + 1,71y_6,$$

$$y_{11} = 1,71x_1 - 0,452y_4 + 0,580y_8,$$

$$c_{10} = \frac{12,3}{752,3},$$

$$c_{11} = (1,75y_2)(0,995x_1),$$

$$c_{12} = 0,995y_{10} + 1998,$$

$$y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}},$$

$$y_{13} = c_{12} - 1,75y_2,$$

$$y_{14} = 3623 + 64,4x_2 + 58,4x_3 + \frac{146,312}{y_9 + x_5},$$

$$c_{13} = 0,995y_{10} + 60,8x_2 + 48x_4 - 0,1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}},$$

$$y_{16} = 148\,000 - 331\,000y_{15} + 40y_{13} - 61y_{15}y_{13},$$

$$c_{14} = 2324y_{10} - 28\,740\,000y_2,$$

$$y_{17} = 14\,130\,000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}},$$

$$c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0,52},$$

$$c_{16} = 1,104 - 0,72y_{15},$$

$$c_{17} = y_9 + x_5.$$

Ограничения имеют следующий вид:

$$y_4 - \frac{0,28}{0,72}y_5 \geqslant 0,$$

$$1,5x_2 - x_3 \geqslant 0,$$

$$21,0 - 3496 \frac{y_2}{c_{12}} \geqslant 0,$$

$$\frac{62\,212}{c_{17}} - 110,6 - y_1 \geqslant 0,$$

$$213,1 \leqslant y_1 \leqslant 405,23,$$

$$17,505 \leqslant y_2 \leqslant 1053,6667,$$

$$11,275 \leqslant y_3 \leqslant 35,03,$$

$$214,228 \leqslant y_4 \leqslant 665,585,$$

$$7,458 \leqslant y_5 \leqslant 584,463,$$

$$0,961 \leqslant y_6 \leqslant 265,916,$$

$$1,612 \leqslant y_7 \leqslant 7,046,$$

$$0,146 \leqslant y_8 \leqslant 0,222,$$

$$107,99 \leqslant y_9 \leqslant 273,366,$$

$$922,693 \leqslant y_{10} \leqslant 1286,105,$$

$$926,832 \leqslant y_{11} \leqslant 1444,046,$$

$$18,766 \leqslant y_{12} \leqslant 537,141,$$

$$1072,163 \leqslant y_{13} \leqslant 3247,039,$$

$$8961,448 \leqslant y_{14} \leqslant 26\,844,086,$$

$$\begin{aligned}
 0,063 &\leq y_{15} \leq 0,386, \\
 71\,084,33 &\leq y_{16} \leq 140\,000, \\
 2\,802\,713 &\leq y_{17} \leq 12\,146\,108, \\
 704,4148 &\leq x_1 \leq 906,3855, \\
 68,6 &\leq x_2 \leq 288,88, \\
 0 &\leq x_3 \leq 134,75, \\
 193 &\leq x_4 \leq 287,0966, \\
 25 &\leq x_5 \leq 84,1988.
 \end{aligned}$$

В качестве начальной берется допустимая точка $\mathbf{x}^{(0)} = [900 \ 80 \ 115 \ 267 \ 27]^T$, в которой $f(\mathbf{x}^{(0)}) = 0,939$.

Решение:

$$\begin{aligned}
 \mathbf{x}^* &= [705,060 \ 68,600 \ 102,900 \ 282,341 \ 35,627]^T, \\
 f(\mathbf{x}^*) &= 1,905.
 \end{aligned}$$

Задача 13 [11].

Эту задачу можно рассматривать либо (а) как задачу, содержащую 12 переменных, 7 ограничений в виде равенств и 16 ограничений, задающих нижний и верхний пределы изменения независимых переменных, либо (что проще) (б) как задачу, содержащую 5 переменных, 3 нелинейных ограничения в виде неравенств и 10 ограничений, задающих нижний и верхний пределы изменения независимых переменных.

Задача 13 иллюстрирует тот случай, когда требуется определять параметры в высокой степени нелинейных дифференциальных уравнений, основываясь на экспериментальных данных. Целевая функция фактически представляет собой сумму квадратов разностей между экспериментальными данными и решениями системы дифференциальных уравнений, полученными методами численного интегрирования.

По условию задачи требуется

максимизировать $f(\mathbf{x}) = [50y_1 + 9,583y_2 + 20y_3 + 15y_4 - 852\,960 - 38\,100(x_2 + 0,01x_3) + k_{31} + k_{32}x_2 + k_{33}x_3 + k_{34}x_4 + k_{35}x_5]x_1 - 24345 + 15x_6$. Значения y_i ($i = 1, 2, 3, 4$), x_6 , x_7 и x_8 находятся путем решения следующей системы уравнений:

$$\begin{aligned}
 x_6 &= (k_1 + k_2x_2 + k_3x_3 + k_4x_4 + k_5x_5)x_1, \\
 y_1 &= k_6 + k_7x_2 + k_8x_3 + k_9x_4 + k_{10}x_5, \\
 y_2 &= k_{11} + k_{12}x_2 + k_{13}x_3 + k_{14}x_4 + k_{15}x_5,
 \end{aligned}$$

$$\begin{aligned}y_3 &= k_{16} + k_{17}x_2 + k_{18}x_3 + k_{19}x_4 + k_{20}x_5, \\y_4 &= k_{21} + k_{22}x_2 + k_{23}x_3 + k_{24}x_4 + k_{25}x_5, \\x_7 &= (y_1 + y_2 + y_3)x_1, \\x_8 &= (k_{26} + k_{27}x_2 + k_{28}x_3 + k_{29}x_4 + \\&\quad + k_{30}x_5)x_1 + x_6 + x_7,\end{aligned}$$

где

$$\begin{aligned}k_1 &= -145421,402, & k_{19} &= 329,574, \\k_2 &= 2931,1506, & k_{20} &= -2882,082, \\k_3 &= -40,427932, & k_{21} &= 74095,3845, \\k_4 &= 5106,192, & k_{22} &= -306,262544, \\k_5 &= 15711,36, & k_{23} &= 16,243649, \\k_6 &= -161622,577, & k_{24} &= -3094,252, \\k_7 &= 4176,15328, & k_{25} &= -5566,2628, \\k_8 &= 2,8260078, & k_{26} &= -26,237, \\k_9 &= 9200,476, & k_{27} &= 99, \\k_{10} &= 13160,295, & k_{28} &= -0,42, \\k_{11} &= -21686,9194, & k_{29} &= 1300, \\k_{12} &= 123,56928, & k_{30} &= 2100, \\k_{13} &= -21,1188894, & k_{31} &= 925548,252, \\k_{14} &= 706,834, & k_{32} &= -61968,8432, \\k_{15} &= 2898,573, & k_{33} &= 23,3088196, \\k_{16} &= 28298,388, & k_{34} &= -27097,648, \\k_{17} &= 60,81096, & k_{35} &= -50843,766. \\k_{18} &= 31,242116,\end{aligned}$$

Ограничения имеют следующий вид:

$$0 \leqslant x_1 \leqslant 5,$$

$$1,2 \leqslant x_2 \leqslant 2,4,$$

$$20 \leqslant x_3 \leqslant 60,$$

$$9 \leqslant x_4 \leqslant 9,3,$$

$$6,5 \leqslant x_5 \leqslant 7,$$

$$0 \leqslant x_6 \leqslant 294\,000,$$

$$0 \leqslant x_7 \leqslant 294\,000,$$

$$0 \leqslant x_8 \leqslant 277\,200.$$

В качестве начальной выбирается точка

$$\mathbf{x}^{(0)} = [2,52 \quad 2 \quad 37,5 \quad 9,25 \quad 6,8]^T,$$

в которой $f(\mathbf{x}^{(0)}) = 2\ 351\ 243,5$.

Решение:

$$\mathbf{x}^* = [4,538 \quad 2,400 \quad 60,000 \quad 9,300 \quad 7,000]^T,$$

$$f(\mathbf{x}^*) = 5\ 280\ 254,$$

$$\mathbf{x}_6^* = 75\ 570,$$

$$\mathbf{x}_7^* = 198\ 157,$$

$$\mathbf{x}_8^* = 277\ 200.$$

Задача 14 [12].

Число переменных равняется шести.

Задача содержит четыре нелинейных ограничения в виде неравенств.

Задача 14 возникла в связи с реальной проблемой «централизации теплоснабжения нефтеочистительных заводов» и характеризуется наличием большого числа локальных оптимумов, каждый из которых имеет свою интерпретацию.

По условию задачи требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^4 c(x_i) + \sum_{i=5}^6 100 c(x_i)$$

при ограничениях

$$t_3 - 300 \geq 0,$$

$$t_4 - 300 \geq 0,$$

$$280 - T_5 \geq 0,$$

$$250 - T_6 \geq 0.$$

Для вычисления $c(x_i)$, t_i и T_i используются следующие соотношения:

$$c(x_i) = 2,7x_i + 1300 \quad \left(\text{наименьшее целое число} \geq \frac{x_i}{2000} \right),$$

$$T_1 = \frac{0,0285x_1 + 300}{1 + 0,0001425x_1}, \quad T_4 = \frac{t_2' + (70 - t_2)e^{-\alpha_4}}{1 - 0,8e^{-\alpha_4}},$$

$$t_1 = 500 - T_1, \quad t_4 = 350 + (t_2 - T_4)e^{\alpha_4},$$

$$\alpha_2 = -0,0001665x_2, \quad T_{j2} = 0,8T_3 + 0,2T_4,$$

$$T_2 = \frac{200 - 350e^{-\alpha_2}}{1 - 1,5e^{-\alpha_2}}, \quad \alpha_5 = 0,000375x_5,$$

$$\begin{aligned}
 t_2 &= 300 + (200 - T_2) e^{\alpha_2}, & T_5 &= 80 + (T_{j2} - 80) e^{-\alpha_5}, \\
 \alpha_3 &= 0,085 \cdot 9,36 \cdot 10^{-5} x_3, & T_{j1} &= 0,7T_1 + 0,3T_2, \\
 T_3 &= \frac{t_1 + (29,75 - t_1) e^{-\alpha_3}}{1 - 0,915e^{-\alpha_3}}, & \alpha_6 &= 0,0003x_6, \\
 t_3 &= 350 + (t_1 - T_3) e^{\alpha_3}, & T_6 &= 80 + (T_{j1} - 80) e^{-\alpha_6}, \\
 \alpha_4 &= 0,00025 x_4.
 \end{aligned}$$

В качестве начальной выбирается точка

$$\mathbf{x}^{(0)} = [8000 \quad 3000 \quad 14\,000 \quad 2000 \quad 300 \quad 10]^T,$$

в которой $f(\mathbf{x}^{(0)}) = 459\,100$.

Решение, полученное методом скользящего допуска, имеет следующий вид:

$$\mathbf{x}^* = [11\,884 \quad 3288 \quad 20\,000 \quad 4000 \quad 114,18 \quad -155,03]^T,$$

$$f(\mathbf{x}^*) = 250\,799,9.$$

Колвилл приводит следующие значения $f(\mathbf{x})$ в оптимальной точке \mathbf{x} :

При старте из допустимой начальной точки	Алгоритм, положенный в основу машинной программы	При старте из точки, не являющейся допустимой ¹⁾
255303,5	Алгоритм обобщенного градиента	266754,0
389858,0	ПОП-360	—
132518,0	Оптим (название специальной машинной программы)	125578,0

¹⁾ Имеется в виду точка $\mathbf{x}^{(0)} = [8000 \quad 3000 \quad 10\,000 \quad 2000 \quad 200 \quad 10]^T$.

Задача 15 [13].

Число переменных равняется шести, из которых только две являются независимыми.

Задача содержит 4 нелинейных ограничения в виде равенств, 2 ограничения на производные (в виде разрывных функций) и 6 ограничений, задающих нижнюю и верхнюю границы изменения переменных. (Задача оптимизации электрической цепи.)

Требуется

$$\text{минимизировать } f(\mathbf{x}) = f_1(x_1) + f_2(x_2)$$

при ограничениях

$$f_1(x_1) = \begin{cases} 30x_1, & 0 \leq x_1 < 300, \\ 31x_1, & 300 \leq x_1 < 400, \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2, & 0 \leq x_2 < 100, \\ 29x_2, & 100 \leq x_2 < 200, \\ 30x_2, & 200 \leq x_2 < 1000, \end{cases}$$

$$x_1 = 300 - \frac{x_3 x_4}{131,078} \cos(1,48577 - x_6) + \frac{0,90798 x_3^2}{131,078} \cos 1,47588,$$

$$x_2 = -\frac{x_3 x_4}{131,078} \cos(1,48477 + x_6) + \frac{0,90798 x_4^2}{131,078} \cos 1,47588,$$

$$x_3 = -\frac{x_3 x_4}{131,078} \sin(1,48477 + x_6) + \frac{0,90798 x_4^2}{131,078} \sin 1,47588,$$

$$200 - \frac{x_3 x_4}{131,078} \sin(1,48477 - x_6) + \frac{0,90798}{131,078} x_3^2 \sin 1,47588 = 0,$$

$$0 \leq x_1 \leq 400,$$

$$0 \leq x_2 \leq 1000,$$

$$340 \leq x_3 \leq 420,$$

$$340 \leq x_4 \leq 420,$$

$$-1000 \leq x_5 \leq 1000,$$

$$0 \leq x_6 \leq 0,5236.$$

В качестве начальной берется точка

$\mathbf{x}^{(0)} = [390 \quad 1000 \quad 419,5 \quad 340,5 \quad 198,175 \quad 0,5]^T$, в которой $f(\mathbf{x}^{(0)}) = 9074,14$.

В зависимости от степени точности, с которой определяется точка \mathbf{x} , результаты оказываются различными. Наличие разрывов производных $f_1(x_1)$ и $f_2(x_2)$ приводит к скачкообразному изменению $f(\mathbf{x})$ и \mathbf{x}^* . Это подтверждается данными, содержащимися в приведенной ниже таблице.

	Высокая точность	Умеренная точность
x_1^*	107,81	201,78
x_2^*	196,32	100,00
x_3^*	373,83	383,07
x_4^*	420,00	420,00
x_5^*	21,31	-10,907
x_6^*	0,153	0,07314
$f(\mathbf{x}^*)$	8297,5888	8853,44 или 8953,40

Задача 16 [14].

Число переменных равняется девятыи.

Задача содержит 13 нелинейных ограничений в виде неравенств и одно ограничение, задающее верхний предел изменения одной из переменных.

Данная задача связана с максимизацией площади шестиугольника, максимальный линейный размер которого (диаметр) равен единице.

Задача сводится к

$$\text{максимизация } f(\mathbf{x}) = 0,5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$$

при ограничениях

$$1 - x_3^2 - x_4^2 \geq 0,$$

$$1 - x_9^2 \geq 0,$$

$$1 - x_5^2 - x_6^2 \geq 0,$$

$$1 - x_1^2 - (x_2 - x_9)^2 \geq 0,$$

$$1 - (x_1 - x_5)^2 - (x_2 - x_6)^2 \geq 0,$$

$$1 - (x_1 - x_7)^2 - (x_2 - x_8)^2 \geq 0,$$

$$1 - (x_3 - x_5)^2 - (x_4 - x_6)^2 \geq 0,$$

$$1 - (x_3 - x_7)^2 - (x_4 - x_8)^2 \geq 0,$$

$$1 - x_7^2 - (x_8 - x_9)^2 \geq 0,$$

$$x_1x_4 - x_2x_3 \geq 0,$$

$$x_3x_9 \geq 0,$$

$$-x_5x_9 \geq 0,$$

$$x_5x_8 - x_6x_7 \geq 0,$$

$$x_9 \geq 0.$$

В качестве начальной выбирается точка $x_i^{(0)} = 1, i = 1, \dots, 9$, в которой $f(\mathbf{x}^{(0)}) = 0$.

Решение:

$$\begin{aligned} \mathbf{x}^* = & [0,9971 - 0,0758 \quad 0,5530 \quad 0,8331 \quad 0,9981 \\ & - 0,0623 \quad 0,5642 \quad 0,8256 \quad 0,0000024]^T, \end{aligned}$$

$$f(\mathbf{x}^*) = 0,8660.$$

Задача 17 [4].

Число переменных равняется десяти.

Задача содержит 20 ограничений, задающих нижнюю и верхнюю границы изменения переменных.

Вне допустимой области целевая функция не определена.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^{10} \{[\ln(x_i - 2)]^2 + [\ln(10 - x_i)]^2\} - \\ - \left(\prod_{i=1}^{10} x_i \right)^{0.2}$$

при ограничениях

$$2,001 < x_i < 9,999, \quad i = 1, \dots, 10.$$

В качестве начальной выбирается точка $x_i^{(0)} = 9, i = 1, \dots, 10.$

Решение:

$$\mathbf{x}^* = [9,351 \quad 9,351 \\ 9,351 \quad 9,351]^T,$$

$$f(\mathbf{x}^*) = -45,778.$$

Задача 18 [9].

Число переменных равняется 15.

Задача содержит пять нелинейных ограничений в виде неравенств и 15 ограничений, определяющих границы изменения независимых переменных.

Данная задача является двойственной по отношению к задаче 10.

Требуется

$$\text{максимизировать } f(\mathbf{x}) = \sum_{i=1}^{10} b_i x_i - \sum_{j=1}^5 \sum_{i=1}^5 c_{ij} x_{10+i} x_{10+j} - \\ - 2 \sum_{j=1}^5 d_j x_{10+j}^3$$

при ограничениях

$$2 \sum_{i=1}^5 c_{ij} x_{10+i} + 3d_j x_{10+j}^2 + e_j - \sum_{i=1}^{10} a_{ij} x_i \geq 0, \quad j = 1, \dots, 5,$$

$$x_i \geq 0, \quad i = 1, \dots, 15.$$

(Примечание. Значения e_j , c_{ij} , d_j , a_{ij} и b_i содержатся в таблице, приведенной в связи с рассмотрением задачи 10.)

В качестве начальной выбирается допустимая точка

$$x_i^{(0)} = 0,0001, \quad i = 1, \dots, 15, \quad i \neq 7,$$

$$x_7^{(0)} = 60,$$

в которой

$$f(\mathbf{x})^{(0)} = -2400,01.$$

Решение:

$$\mathbf{x}^* = [0,0000 \quad 0,0000 \quad 5,1740 \quad 0,0000 \quad 3,0611 \quad 11,8395 \quad 0,0000 \\ 0,0000 \quad 0,1039 \quad 0,0000 \quad 0,3000 \quad 0,3335 \quad 0,4000 \quad 0,4283 \\ 0,2240]^T,$$

$$f(\mathbf{x}^*) = -32,386.$$

К такому же решению можно прийти, начав оптимизационный поиск в точке

$$\begin{aligned}x_i^{(0)} &= b_i^{(0)}, \quad i = 1, \dots, 10, \\x_i^{(0)} &= 0, \quad i = 11, \dots, 14, \\x_i^{(0)} &= 1, \quad i = 15,\end{aligned}$$

которая не является допустимой и в которой

$$f(\mathbf{x}^{(0)}) = 6829,06.$$

Задача 19 [15].

Число переменных равняется 16.

Задача содержит восемь линейных ограничений в виде равенств и 32 ограничения, которые определяют нижнюю и верхнюю границы изменения переменных.

Требуется

$$\text{максимизировать } f(\mathbf{x}) = -\sum_{i=1}^{16} \sum_{j=1}^{16} a_{ij} (x_i^2 + x_i + 1) (x_j^2 + x_j + 1)$$

при ограничениях

$$\begin{aligned}\sum_{j=1}^{16} b_{ij} x_j &= c_i, \quad i = 1, \dots, 8, \\0 < x_i < 5, \quad j &= 1, \dots, 16.\end{aligned}$$

(Примечание. Значения a_{ij} , b_{ij} и c_i указаны в приведенной ниже таблице.)

В качестве начальной выбирается точка $x_i^{(0)} = 10$, $i = 1, \dots, 16$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = 209,457$.

Решение:

$$\mathbf{x}^* = [0,040 \quad 0,792 \quad 0,203 \quad 0,844 \quad 1,270 \quad 0,935 \quad 1,682 \quad 0,155 \\ 1,568 \quad 0,000 \quad 0,000 \quad 0,000 \quad 0,660 \quad 0,000 \quad 0,674 \quad 0,000]^T,$$

$$f(\mathbf{x}^*) = -244,900.$$

Данные для задачи 19

Задача 20 [4].

Число переменных равняется 24.

Задача содержит 12 нелинейных ограничений в виде равенств, 2 линейных ограничения в виде равенств, 6 нелинейных ограничений в виде неравенств и 24 ограничения, задающих верхние границы интервалов изменения независимых переменных.

Данная задача связана с минимизацией затрат на приготовление многокомпонентных жидких смесей.

Задача состоит в том, чтобы

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^{24} a_i x_i$$

при ограничениях

$$h_i(\mathbf{x}) = \frac{\sum_{j=1}^{24} \frac{x_j}{b_j}}{b_{i+12} \sum_{j=13}^{24} \frac{x_j}{b_j}} - \frac{c_i x_i}{40 b_i \sum_{j=1}^{12} \frac{x_j}{b_j}} = 0, \quad i = 1, \dots, 12,$$

$$h_{13}(\mathbf{x}) = \sum_{i=1}^{24} x_i - 1 = 0,$$

$$h_{14}(\mathbf{x}) = \sum_{i=1}^{12} \frac{x_i}{d_i} + f \sum_{i=13}^{24} \frac{x_i}{b_i} - 1,671 = 0,$$

где

$$f = (0,7302) \cdot (530) \cdot \left(\frac{14,7}{40} \right),$$

$$\frac{-[x_i + x_{i+12}]}{\sum_{j=1}^{24} x_j + e_i} \geq 0, \quad i = 1, 2, 3,$$

$$\frac{-[x_{i+3} + x_{i+15}]}{\sum_{j=1}^{24} x_j + e_i} \geq 0, \quad i = 4, 5, 6.$$

$$x_i \geq 0, \quad i = 1, \dots, 24.$$

(Примечание. Значения a_i , b_i , c_i , d_i и e_i указаны ниже в таблицах.)

В качестве начальной берется точка $x_i^{(0)} = 0,04$, $i = 1, \dots, 24$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = 0,14696$.

Решения, получаемые методом скользящего допуска, методом НЛП и методом МПБМ, имеют следующий вид:

	Метод скользящего допуска	НЛП	МПБМ
$f(\mathbf{x})$	0,05700	0,09670	0,07494
x_1^*	7,804E — 03	9,537E — 07	9,109E — 03
x_2^*	1,121E — 01	0	3,739E — 02
x_3^*	1,136E — 01	4,215E — 03	8,961E — 02
x_4^*	0	1,039E — 04	1,137E — 02
x_5^*	0	0	4,155E — 03
x_6^*	0	0	4,184E — 03
x_7^*	6,609E — 02	2,072E — 01	5,980E — 02
x_8^*	0	5,979E — 01	1,554E — 02
x_9^*	0	1,298E — 01	1,399E — 02
x_{10}^*	0	3,350E — 02	8,780E — 03
x_{11}^*	1,914E — 02	1,711E — 02	1,231E — 02
x_{12}^*	6,009E — 03	8,427E — 03	1,153E — 02
x_{13}^*	5,008E — 02	4,657E — 10	7,570E — 02
x_{14}^*	1,844E — 01	0	7,997E — 02
x_{15}^*	2,693E — 01	0	2,797E — 01
x_{16}^*	0	0	1,168E — 01
x_{17}^*	0	0	2,347E — 02
x_{18}^*	0	0	6,368E — 03
x_{19}^*	1,704E — 01	2,868E — 04	2,028E — 01
x_{20}^*	0	1,193E — 03	7,451E — 03
x_{21}^*	0	8,332E — 05	4,547E — 03
x_{22}^*	0	1,239E — 04	1,010E — 02
x_{23}^*	8,453E — 01	2,070E — 05	1,220E — 03
x_{24}^*	1,980E — 04	1,829E — 07	1,810E — 03
$h_1(\mathbf{x}^*)$	0	4,908E — 07	-1,182E — 03
$h_2(\mathbf{x}^*)$	0	0	-4,329E — 04
$h_3(\mathbf{x}^*)$	0	0	3,467E — 03
$h_4(\mathbf{x}^*)$	0	0	2,217E — 04
$h_5(\mathbf{x}^*)$	0	0	-2,550E — 04
$h_6(\mathbf{x}^*)$	0	0	-7,368E — 04
$h_7(\mathbf{x}^*)$	0	-2,209E — 08	1,982E — 03
$h_8(\mathbf{x}^*)$	0	-8,521E — 08	-2,334E — 05
$h_9(\mathbf{x}^*)$	0	-5,854E — 09	1,629E — 03

	Метод скользящего допуска	НЛП	МПБМ
$h_{10} (x^*)$	0	8,137E - 08	-4,397E - 04
$h_{11} (x^*)$	0	-2,596E - 08	9,431E - 04
$h_{12} (x^*)$	0	5,766E - 08	1,853E - 03
$h_{13} (x^*)$	0	0	-1,741E - 02
$h_{14} (x^*)$	N	0	8,743E - 03

Данные, относящиеся к задачам 10 и 18

i	a_i	b_i	c_i	d_i	e_i
1	0,0693	44,094	123,7	31,244	0,1
2	0,0577	58,12	31,7	36,12	0,3
3	0,05	58,12	45,7	34,784	0,4
4	0,20	137,4	14,7	92,7	0,3
5	0,26	120,9	84,7	82,7	0,6
6	0,55	170,9	27,7	91,6	0,3
7	0,06	62,501	49,7	56,708	
8	0,10	84,94	7,1	82,7	
9	0,12	133,425	2,1	80,8	
10	0,18	82,507	17,7	64,517	
11	0,10	46,07	0,85	49,4	
12	0,09	60,097	0,64	49,1	
13	0,0693	44,094			
14	0,0577	58,12			
15	0,05	58,12			
16	0,20	137,4			
17	0,26	120,9			
18	0,55	170,9			
19	0,06	62,501			
20	0,10	84,94			
21	0,12	133,425			
22	0,18	82,507			
23	0,10	46,07			
24	0,09	60,097			

Задача 21 [16].

Число независимых переменных равняется трем.

Задача содержит 6 ограничений, задающих нижнюю и верхнюю границы изменения независимых переменных.

Требуется

$$\text{минимизировать } f(x) = \sum_{i=1}^{99} \left\{ \exp \left[-\frac{(u_i - x_2)^{x_3}}{x_1} \right] - 0,01i \right\}^2, \quad u_i =$$

$$= 25 + (-50 \ln 0,01i)^{1/1,5}$$

при ограничениях

$$0,1 \leqslant x_1 \leqslant 100,0, \quad 0,0 \leqslant x_2 \leqslant 25,6, \quad 0,0 \leqslant x_3 \leqslant 5,0.$$

В качестве начальной выбирается допустимая точка $x^{(0)} = [100,0 \ 12,5 \ 3,0]^T$.

Решение:

$$x^* = [50,0 \ 25,0 \ 1,5]^T,$$

$$f(x^*) = 0,0.$$

Задача 22 [17].

Число независимых переменных равняется шести.

Задача содержит 4 нелинейных ограничения в виде неравенств и 12 ограничений, задающих нижние и верхние границы изменения независимых переменных.

Требуется

$$\text{минимизировать } f(x) = 4,3x_1 + 31,8x_2 + 63,3x_3 + 15,8x_4 +$$

$$+ 68,5x_5 + 4,7x_6$$

при ограничениях

$$17,1x_1 + 38,2x_2 + 204,2x_3 + 212,3x_4 + 623,4x_5 + 1495,5x_6 -$$

$$- 169x_1x_3 - 3580x_3x_5 - 3810x_4x_5 - 18500x_4x_6 - 24300x_5x_6 \geq b_1,$$

$$17,9x_1 + 36,8x_2 + 113,9x_3 + 169,7x_4 + 337,8x_5 + 1385,2x_6 -$$

$$- 139x_1x_3 - 2450x_4x_5 - 16600x_4x_6 - 17200x_5x_6 \geq b_2,$$

$$- 273x_2 - 70x_4 - 819x_5 + 26000x_4x_5 \geq b_3,$$

$$159,9x_1 - 311x_2 + 587x_4 + 391x_5 + 2198x_6 - 14000x_1x_6 \geq b_4,$$

$$0 \leqslant x_1 \leqslant 0,31, \quad 0 \leqslant x_4 \leqslant 0,042,$$

$$0 \leqslant x_2 \leqslant 0,046, \quad 0 \leqslant x_5 \leqslant 0,028,$$

$$0 \leqslant x_3 \leqslant 0,068, \quad 0 \leqslant x_6 \leqslant 0,0134.$$

Ниже приводится решение задачи для различных значений параметров.

Для				Решение						
b_1	b_2	b_3	b_4	x_1	x_2	x_3	x_4	x_5	x_6	$f(x^*)$
4,97	-1,88	- 29,08	- 78,02	0	0	0	0	0	0,00333	0,0156
4,97	-1,88	- 69,08	-118,02	0	0	0	0	0	0,00332	0,0156
32,97	25,12	- 29,08	- 78,02	0	0	0,0633	0	0	0,0134	4,070
32,97	25,12	-124,08	-173,02	0	0	0,0633	0	0	0,0134	4,070

Задача 23 (распределение орудий по целям) [1].

Число независимых переменных равняется 100.

Задача содержит 12 линейных ограничений и 100 ограничений, задающих нижние границы для значений, которые могут принимать независимые переменные.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = \sum_{i=1}^{20} u_i \left(\prod_{j=1}^5 a_{ij}^{x_{ij}} - 1 \right)$$

при ограничениях¹⁾

$$\begin{aligned} \left(\sum_{i=1}^5 x_{ij} \right) - b_j &\geq 0, \quad j = 1, 6, 10, 14, 15, 16, 20, \\ - \left(\sum_{j=1}^{20} x_{ij} \right) - c_i &\geq 0, \quad i = 1, \dots, 5. \end{aligned}$$

Задача 24 [1].

Число независимых переменных равняется двум.

Задача содержит одно нелинейное ограничение в виде неравенства и одно линейное ограничение также в виде неравенства.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2$$

при ограничениях

$$g_1(\mathbf{x}) = -x_1^2 + x_2 \geq 0,$$

$$g_2(\mathbf{x}) = -x_1 - x_2 + 2 \geq 0.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [2 \ 2]^T$, лежащая вне допустимой области. В этой точке $f(\mathbf{x}^{(0)}) = 1$.

Решение:

$$f(\mathbf{x}^*) = 1,$$

$$\mathbf{x}^* = [1 \ 1]^T.$$

Задача 25.

Число независимых переменных равняется двум.

Ограничения отсутствуют.

См. таблицу на стр. 471.

Данные для задачи 23
 α_{ij} — вероятность того, что j -е орудие не поразит j -ю цель

$i \backslash j$	i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	c_i (количество орудий типа i)
1	1	0,95	1	1	1	0,85	0,90	0,85	0,80	1	1	1	1	1	1	1	1	0,95	1	1	200	
2	0,84	0,83	0,85	0,84	0,85	0,81	0,81	0,82	0,80	0,86	1	0,98	1	0,88	0,87	0,88	0,85	0,84	0,85	0,85	100	
3	0,96	0,95	0,96	0,96	0,96	0,90	0,92	0,91	0,92	0,95	0,99	0,98	0,99	0,98	0,97	0,98	0,95	0,92	0,93	0,92	300	
4	1	1	1	1	1	1	1	1	1	0,96	0,91	0,92	0,91	0,92	0,98	0,93	1	1	1	1	150	
5	0,92	0,94	0,92	0,95	0,95	0,98	0,98	1	1	0,90	0,95	0,96	0,91	0,98	0,99	0,99	1	1	1	1	250	

b_j — минимальное число орудий, выделяемых для поражения j -й цели

30 100 40 50 70 35 10

u_j — степень важности (с военной точки зрения) j -й цели

60 50 50 75 40 60 35 30 25 150 30 45 125 200 200 130 100 100 150

Допустимые решения, получаемые при решении задачи распределения орудий по целям x_{ij}

Орудие типа i	Номер цели i																				Итого
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	
1		24				32	37	28	22						5			52			200
		(16)				(100)	(38)	(26)	(20)												(200)
2	1	8	2	18	11										29	9	21				99
				(23)	(20)										(25)	(31)	(1)				(100)
3		9			29	62									35		17	25	62	60	299
															(45)		(76)	(56)	(62)	(61)	(300)
4										9	39		58		44						150
											(39)	(50)	(57)		(4)						(150)
5	47	5	36	12		6				50	42		51		1						250
	(50)	(46)	(47)							(50)	(57)										(250)
Итого	48	46	38	30	40	100	37	28	22	50	51	39	51	58	70	53	38	77	62	60	
	(50)	(62)	(47)	(23)	(20)	(100)	(38)	(26)	(20)	(50)	(57)	(39)	(50)	(57)	(70)	(35)	(77)	(56)	(62)	(61)	

Примечание: Числа без скобок взяты у Хольцмана, числа в скобках заимствованы у Брэкена и Мак-Кормика.
 $f(\mathbf{x}^*) = 1732$.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = 4(x_1 - 5)^2 + (x_2 - 6)^2.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [8 \ 9]^T$, в которой $f(\mathbf{x}^{(0)}) = 45$.

Решение:

$$\begin{aligned}\mathbf{x}^* &= [5 \ 6]^T, \\ f(\mathbf{x}^*) &= 0.\end{aligned}$$

Задача 26 [18].

Число независимых переменных равняется четырем.

Ограничения отсутствуют.

Требуется

$$\begin{aligned}\text{минимизировать } f(\mathbf{x}) &= (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + \\ &+ 10(x_1 - x_4)^4.\end{aligned}$$

В качестве начальной выбирается либо точка $\mathbf{x}^{(0)} = [3 \ -1 \ 0 \ 1]^T$, в которой $f(\mathbf{x}^{(0)}) = 215$, либо точка $\mathbf{x}^{(0)} = [1 \ 1 \ 1 \ 1]^T$, в которой $f(\mathbf{x}^{(0)}) = 125$.

Решение:

$$\begin{aligned}\mathbf{x}^* &= [0 \ 0 \ 0 \ 0]^T, \\ f(\mathbf{x}^*) &= 0.\end{aligned}$$

Задача 27.

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = (x_1 x_2)^2 (1 - x_1)^2 [1 - x_1 - x_2 (1 - x_1)^5]^2.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [-1,2 \ 1]^T$, в которой $f(\mathbf{x}^{(0)}) = 26\,656$.

Решение:

точкой \mathbf{x}^* является либо точка $[1, \text{неограниченное значение}]^T$, либо точка $[0, \text{неограниченное значение}]^T$, либо точка $[\text{неограниченное значение}, 0]^T$,
 $f(\mathbf{x}^*) = 0$.

Задача 28 (связана с решением системы уравнений).

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2.$$

В качестве начальной берется точка $\mathbf{x}^{(0)} = [1 \ 1]^T$, в которой $f(\mathbf{x}^{(0)}) = 106$.

Решение:

a) $\mathbf{x}^* = [3,58443 - 1,84813]^T$; б) $\mathbf{x}^* = [3 \ 2]^T$, $f(\mathbf{x}^*) = 0$.

(Примечание. Все машинные программы при старте из точки $\mathbf{x}^{(0)} = [1 \ 1]^T$ приводят ко второму решению.)

Задача 29.

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

минимизировать $f(\mathbf{x}) = (x_1^2 + 12x_2 - 1)^2 + (49x_1^2 + 49x_2^2 + 84x_1 + 2324x_2 - 681)^2$.

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [1 \ 1]^T$, в которой $f(\mathbf{x}^{(0)}) = 3,3306 \cdot 10^6$.

Решение:

a) $\mathbf{x}^* = [0,28581 \ 0,27936]^T$, $f(\mathbf{x}^*) = 5,9225$;

б) $\mathbf{x}^* = [-21,026653 \ -36,7660090]^T$, $f(\mathbf{x}^*) = 0$.

Задача 30.

Число независимых переменных равняется трем.

Ограничения отсутствуют.

Требуется

минимизировать $f(\mathbf{x}) = 100 \left[x_3 - \left(\frac{x_1 + x_2}{2} \right)^2 \right]^2 + (1 - x_1)^2 + (1 - x_2)^2$.

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [-1,2 \ 2 \ 0]^T$, в которой $f(\mathbf{x}^{(0)}) = 8,40$.

Решение:

$$\begin{aligned} \mathbf{x}^* &= [1 \ 1 \ 1]^T, \\ f(\mathbf{x}^*) &= 0. \end{aligned}$$

Задача 31 [1].

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

минимизировать $f(\mathbf{x}) = (x_1 - 2)^2 + (x_2 - 1)^2 + \frac{0,04}{g_1(\mathbf{x})} + \frac{h_1^2(\mathbf{x})}{0,2}$,

при ограничениях

$$g_1(\mathbf{x}) = -\frac{x_1^2}{4} - x_2^2 + 1,$$

$$h_1(\mathbf{x}) = x_1 - 2x_2 + 1.$$

В качестве начальной берется точка $x^{(0)} = [2 \ 2]^T$, в которой $f(x^{(0)}) = 5,99$.

Решение:

$$x^* = [1,7954 \ 1,3779]^T \text{ (локальный минимум),}$$

$$f(x^*) = 0,16904.$$

(Примечание. Другие варианты выбора начальной точки приводят к другим решениям. При проверке не было получено ни решение, приведенное в указанном выше источнике, ни данное решение при указанном начальном векторе. Представляется важным следующее обстоятельство: при достаточно больших начальных перемещениях (т. е. при начальных перемещениях с большой длиной шага) локальный минимум $f(x)$ удается обойти, в результате чего оптимизационный поиск сразу приводит к глобальному минимуму $f(x)$ при $g_1(x) = -0$, когда $f(x) \rightarrow -\infty$.)

Задача 32.

Данная задача возникает в связи с оцениванием экспериментальных данных методом наименьших квадратов.

Число независимых переменных равняется четырем.

Ограничения отсутствуют.

Требуется

$$\text{минимизировать } f(x) = 10^4 \sum_{i=1}^7 \left(\frac{\frac{x_1^2 + x_2^2 a_i + x_3^2 a_i^2}{1 + x_4^2 a_i} - b_i}{b_i} \right).$$

В качестве начальной выбирается точка

$$x^{(0)} = [2,7 \ 90 \ 1500 \ 10]^T,$$

в которой $f(x^{(0)}) = 2,905 \cdot 10^4$.

Решение:

$$x^* = [2,714 \ 140,4 \ 1707 \ 31,51]^T,$$

$$f(x^*) = 318,572.$$

Ниже указаны значения констант a_i и b_i :

i	a_i	b_i
1	0,0	7,391
2	0,000428	11,18
3	0,00100	16,44
4	0,00161	16,20
5	0,00209	22,20
6	0,00348	24,02
7	0,00525	31,32

Задача 33.

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

$$\text{максимизировать } f(\mathbf{x}) = e^{-x_1 - x_2} (2x_1^2 + 3x_2^2).$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [2,5 \ 2,5]^T$, в которой $f(\mathbf{x}^{(0)}) = 2,3299 \cdot 10^{-5}$.

Решение:

$$\mathbf{x}^* = [0 \ 1]^T \text{ или } \mathbf{x}^* = [0 \ -1]^T,$$

$$f(\mathbf{x}^*) = 1,1036.$$

Задача 34 [19].

Число независимых переменных равняется трем.

Ограничения отсутствуют.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = 100 \{[x_3 - 10 \theta(x_1, x_2)]^2 + [(x_1^2 + x_2^2)^{1/2} - 1]^2\} + x_3^2,$$

$$\theta(x_1, x_2) = \begin{cases} \frac{1}{2\pi} \arctg \frac{x_2}{x_1}, & x_1 > 0, \\ \frac{1}{2} + \frac{1}{2\pi} \arctg \frac{x_2}{x_1}, & x_1 < 0. \end{cases}$$

В качестве начальной берется точка

$$\mathbf{x}^{(0)} = [-1 \ 0 \ 0]^T.$$

Решение:

$$\mathbf{x}^* = [1 \ 0 \ 0]^T,$$

$$f(\mathbf{x}^*) = 0.$$

Задача 35.

Число независимых переменных равняется двум.

Ограничения отсутствуют.

Требуется

$$\text{минимизировать } f(\mathbf{x}) = u_1^2 + u_2^2 + u_3^2,$$

где

$$u_i = c_i - x_1(1 - x_2^i), \quad c_1 = 1,5, \quad c_2 = 2,25, \quad c_3 = 2,625.$$

В качестве начальной выбирается точка $\mathbf{x}^{(0)} = [2 \ 0,2]^T$, в которой $f(\mathbf{x}^{(0)}) = 0,52978$.

Решение:

$$\mathbf{x}^* = [3,0000 \ 0,5000]^T,$$

$$f(\mathbf{x}^*) = 0.$$

ЛИТЕРАТУРА

1. Bracken J., McCormick G. P., Selected Applications of Nonlinear Programming, Wiley, N. Y., 1968.
2. Rosenbrock H. H., An Automatic Method for Finding the Greatest and Least Value of a Function, *Computer J.*, 3, 175 (1960).
3. Barnes G. K., M. S. Thesis, Univ. of Texas, Austin, Tex., 1967.
4. Paviani D. A., Ph. D. Dissertation, Univ. of Teras, Austin, Tex., 1969.
5. Jones A. P., The Chemical Equilibrium Problem: An Application of SUMT, Res. Analysis Corp., McLean, Va., RAC-TP-272, 1967.
6. Colville A. R., A Comparative Study on Nonlinear Programming Codes, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968, p. 31.
7. Wodd C. F., Westinghouse Res. Lab. (cited in Colville, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968).
8. Himmelblau D. M., Yates R. V., A New Method of Flow Routing, Water Resources Res., 4, 1193 (1968).
9. Shell Development Co. (cited in Colville, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968, p. 21).
10. Proctor and Gamble Co. (cited in Colville, IBM N. Y., Sci. Center Rept. 320-2949, June, 1968, p. 24).
11. Box M. J., A New Method of Constrained Optimization and a Comparison with Other Mehods, *Computer J.*, 8, 42 (1965).
12. Efroymson M. A., Esso Res. and Engineering Co. (cited in Colville, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968, p. 26).
13. Huard P., Electricité de France, Directions des Études et Recherches (cited in Colville, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968, p. 28).
14. Pearson J. D., On Variable Metric Methods of Minimization, Res. Analysis Corp. Rept. RAC-TP-302, McLean, Va., May, 1968.
15. Gauthier J. M., IBM France (cited in Colville, IBM N. Y. Sci. Center Rept. 320-2949, June, 1968, p. 29).
16. Holzman A. G., SRCC Rept. 113, Univ. of Pittsburgh, Pittsburgh, Pa., 1969.
17. U. S. Steel Co (cited by Holzman, SRCC Rept. 113, 1969).
18. Powell M. J. D., *Computer J.*, 5, 147 (1962).
19. Fletcher R., Powell M. J. D., *Computer J.*, 6, 33 (1963).

Приложение Б

ПРОГРАММЫ НА ЯЗЫКЕ ФОРТРАН, НЕПОСТАВЛЯЕМЫЕ КОММЕРЧЕСКИ



Для решения задач нелинейного программирования при *отсутствии ограничений* представленные программы используют следующие алгоритмы:

- 1) алгоритм Бройдена;
- 2) алгоритм Дэвидона — Флетчера — Пауэлла;
- 3) алгоритм Пирсона 2;
- 4) алгоритм Пирсона 3;
- 5) алгоритм Флетчера — Ривса;
- 6) алгоритм Ньютона;
- 7) проективный алгоритм Ньютона;
- 8) алгоритм Голдштейна — Прайса;
- 9) алгоритм Пауэлла;
- 10) алгоритм Нелдера — Мида.

При решении задач нелинейного программирования при *наличии ограничений* используется алгоритм *скользящего допуска*.

Б.1. АЛГОРИТМЫ НЕЛИНЕЙНОГО ПРОГРАММИРОВАНИЯ ПРИ ОТСУТСТВИИ ОГРАНИЧЕНИЙ, ОСНОВАННЫЕ НА ВЫЧИСЛЕНИИ ПРОИЗВОДНЫХ

Приведенные ниже инструкции являются руководством для использования программ¹⁾, основанных на применении в сочетании с методом золотого сечения и методом ДСК — Пауэлла (Коггина) следующих алгоритмов:

- 1) алгоритма Дэвидона — Флетчера — Пауэлла;
- 2) алгоритма Пирсона 2;
- 3) алгоритма Пирсона 3;
- 4) алгоритма Бройдена;
- 5) проективного алгоритма Ньютона;
- 6) алгоритма Флетчера — Ривса;
- 7) алгоритма Ньютона;
- 8) алгоритма Голдштейна — Прайса.

1. Структура программы

1.1. Вызывающая программа EXEC. Эта программа запускает INDIC и IPRINT и измеряет время их исполнения, исключая

¹⁾ Автор приносит благодарность М. Анденбергу за подготовку программ 1—8.

время запуска программы. Следует задать $IPRINT = 1$, чтобы получить полную распечатку каждого шага, и $IPRINT = 2$, если необходимо распечатать только окончательные значения целевой функции и вектора x . Программа EXEC вызывает подпрограмму MINI. (Правила использования INDIC приводятся в описании подпрограммы SEARCH.)

1.2. Подпрограмма MINI. Эта подпрограмма управляет выбором используемого алгоритма. Она вызывает DER, FUN, ETA и CONVRG. Имеются разные варианты MINI для следующих групп алгоритмов:

- а) Дэвидона — Флетчера — Пауэлла, Бродена, Пирсона 2, Пирсона 3, проективного Ньютона;
- б) Ньютона;
- в) Голдштейна — Прайса;
- г) Флетчера — Ривса.

Подпрограмма MINI также обеспечивает распечатку информации на каждом шаге при условии использования IPRINT.

1.3. Подпрограмма ETA. Эта подпрограмма обеспечивает вычисление матрицы выбора направлений для первых пяти из указанных выше алгоритмов (т. е. имеется пять различных вариантов подпрограммы ETA).

1.4. Подпрограмма CONVRG. Эта подпрограмма обеспечивает вычисление критерия сходимости, определяемого в конце каждого шага, на основе значений $f(x)$, x и производных от $f(x)$.

1.5. Подпрограмма SEARCH. Эта подпрограмма включает несколько взаимозаменяемых групп команд одномерного поиска.

1.6. Подпрограмма FUN. Данная подпрограмма содержит функцию, подлежащую минимизации. Чтобы обеспечить минимизацию функции $f(x)$, необходимо ее представить как

$$FX = \dots$$

1.7. Подпрограмма DER. Эта подпрограмма обеспечивает вычисление первых частных производных $f(x)$ в следующей последовательности:

$GX(1)$ — частная производная по x_1 ,

$GX(2)$ — частная производная по x_2 и т. д.

1.8. BLOCK DATA. Подпрограмма обеспечивает присвоение начальных значений составляющим x и фиксирует число независимых переменных N (N принимает целочисленные значения).

2. Данные, подготавливаемые пользователем

Параметры в программу не вводятся. Однако это можно осуществить, если добавить соответствующие подпрограммы.

2.1. Чтобы провести линейный поиск (методом ДСК — Пауэлла (Коггина) или методом золотого сечения), необходимо использовать соответствующие подпрограммы SEARCH и MINI, определив их при помощи комментариев за заголовком подпрограммы.

2.2. Чтобы выбрать необходимую подпрограмму ETA, соответствующую группе алгоритмов 1—5, необходимо прочесть комментирующие перфокарты, следующие после заголовка. Когда в колоду перфокарт вставляются карты с кодирограммой ETA, необходимо убедиться, что ранее используемый вариант этой кодирограммы из колоды изъят.

2.3. В подпрограмме BLOCK DATA необходимо подготовить исходные предполагаемые значения $x^{(0)}$ вектора x и указать размерность вектора x и число переменных. В соответствующей строке после слова DATA нужно разместить вводимые значения x и размерности N (целое) так, как показано в приводимом ниже примере подпрограммы¹⁾.

```

PROGRAM EXEC(INPUT,OUTPUT)
COMMON /THREE/ N,NFUNC,NDRV,ITER,INDIC,IPRINT
NFUNC=0
NDRV=0
INDIC=2
IPRINT = 1
CALL SECOND(TIME)
PRINT 2000, TIME
CALL MINI
CALL SECOND(TIME)
PRINT 2000, TIME
CALL EXIT
2000 FORMAT(*OTIME IS NOW*,F10.3,* SECONDS.*,,/)
END

SUBROUTINE CONVRG(GY,IPASS)
COMMON /ONE/ X{10},Y{10},S{10},FX,FY
COMMON /THREE/ N,NFUNC,NDRV,ITER,INDIC,IPRINT
DIMENSION GY{10}
XTOL=0.00001

```

¹⁾ Рассмотренные программы для решения задач нелинейного программирования написаны на языке ФОРТРАН IV. Здесь процедура присвоения тех или иных значений переменным осуществляется в момент загрузки программы при помощи предложения DATA и подпрограммы BLOCK DATA. Эти методы присвоения значений переменным являются отличительной чертой языка ФОРТРАН IV. Однако приводимые программы можно реализовать и с использованием обычного транслятора с ФОРТРАНом, изменив соответствующим образом процедуры присвоения значений переменным.— Прим. перев.

```

FTOL=0.00001
GTOL=0.0001
C CHECK FUNCTION VALUES
  IF(ABS(FX).LE.FTOL) GOTO10
  IF(ABS((FX-FY)/FX).GT.GTOL) GO TO 60
  GO TO 20
10   IF(ABS(FX-FY).GT.FTOL) GO TO 60
C CHECK TEST POINT
20   DO 40 I=1,N
      IF(ABS(X(I)).LE.XTOL) GO TO 30
      IF(ABS((X(I)-Y(I))/X(I)).GT.XTOL) GO TO 60
      GO TO 40
30   IF(ABS(X(I)-Y(I)).GT.XTOL) GO TO 60
40   CONTINUE
C CHECK GRADIENT
  DO 50 I=1,N
50   IF(ABS(GY(I)).GT.GTOL) GO TO 60
C ALL CONVERGENCE CRITERIA SATISFIED
  IPASS=1
  RETURN
C CONVERGENCE NOT ACHIEVED
60   IPASS=2
  RETURN
END

BLOCK DATA
COMMON /ONE/ X(10),Y(10),S(10),FX,FY
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DATA {X(I),I=1,4},N/-3.,-1.,-3.,-1.,4/
END

SUBROUTINE DER(Z,GX)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION Z(10)
DIMENSION TM(6)
DIMENSION GX(10)
TM(1)=Z(2)-Z(1)*Z(1)
TM(2)=1-Z(1)
TM(3)=Z(4)-Z(3)*Z(3)
TM(4)=1-Z(3)
TM(5)=Z(2)-1
TM(6)=Z(4)-1
GX(1)=-400.*Z(1)*TM(1)-2.*TM(2)
GX(2)=200.*TM(1)+20.*TM(5)+19.8*TM(6)
GX(3)=-360*Z(3)*TM(3)-2.*TM(4)
GX(4)=180.*TM(3)+20.*TM(6)+19.8*TM(5)
NDRV=NDRV+4
RETURN
END

SUBROUTINE FUN(Z,FX)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION Z(10)
DIMENSION TM(6)
TM(1)=Z(2)-Z(1)*Z(1)
TM(2)=1-Z(1)
TM(3)=Z(4)-Z(3)*Z(3)
TM(4)=1-Z(3)
TM(5)=Z(2)-1
TM(6)=Z(4)-1
FX=100.*TM(1)*TM(1)+TM(2)*TM(2)+90.*TM(3)*TM(3)+TM(4)*TM(4)+*
A10.*TM(5)*TM(5)+TM(6)*TM(6))+19.8*TM(5)*TM(6)
NFUNCT=NFUNCT+1
RETURN
END

```

```

SUBROUTINE MINI
C NEWTON METHOD
C THE ETA SUBROUTINE MUST CALCULATE THE ACTUAL HESSIAN MATRIX AND
C RETURN THE INVERSE IN THE H ARRAY.
    COMMON /ONE/ X(10),Y(10),S(10),FX,FY
    COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
    COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
    ITER=0
C EVALUATE INITIAL POINTY
    CALL FUN(X,FX)
    KOUNTS=KOUNTS+1
    PRINT 2000, ITER,FX,(X(I),I=1,N)
    CALL DER(X,GX)
C ITERATE FOR SOLUTION
10   ITER=ITER+1
    CALL ETA
    DO 30 I=1,N
        S(I)=0.
    DO 20 J=1,N
20   S(I)=S(I)-H(I,J)*GX(J)
30   Y(I)=X(I)+S(I)
    CALL FUN(Y,FY)
    KOUNTS=KOUNTS+1
C TEST FOR CONVERGENCE
    CALL DER(Y,GX)
    CALL CONVRG(GX,IPASS)
    IF(IPASS.EQ.1) GO TO 50
C CONVERGENCE CRITERIA NOT SATISFIED
    FX=FY
    DO 40 I=1,N
40   X(I)=Y(I)
    IF(IPRINT.EQ.1) PRINT 2000, ITER,FX,(X(I),I=1,N)
    GO TO 10
C CONVERGENCE CRITERIA SATISFIED
50   PRINT 2000, ITER,FY,(Y(I),I=1,N)
2000 FORMAT(1X,16,E16.8,(5E16.8))
    RETURN
END

```

```

SUBROUTINE MINI
COMMON /ONE/ X(10),Y(10),S(10),FX,FY
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION GY(10)
C EXECUTIVE PROGRAM FOR METHODS WHICH APPROXIMATE THE INVERSE OF THE
C HESSIAN MATRIX INCLUDING
C 1 DAVIDON-FLETCHER,POWELL
C 2 BROYDEN
C 3 PEARSON 2
C 4 PEARSON 3
    ITER=0
C EVALUATE THE INITIAL POINT
    CALL FUN(X,FX)
    CALL DER(X,GX)
    PRINT 2000, ITER,NFUNCT,NDRV,FX,(X(I),I=1,N)
C SET UP THE IDENTITY MATRIX
5   DO 20 I=1,N
      DO 10 J=1,N
10   H(I,J)=0.0
20   H(I,I)=1.0
C TAKE GRADIENT STEP
    IF(IPRINT.EQ.1) PRINT 2100
2100 FORMAT(* GRADIENT STEP*)
    DO 30 I=1,N
30   S(I)=-GX(I)
C FIND NEXT POINT
40   CALL SEARCH

```

```

C CHECK WHETHER SEARCH WAS A SUCCESS. IF NOT, RESET H AND TAKE GRADIENT STEP
  IF(FY.GE.FX) GO TO 5
  ITER=ITER+1
  CALL DER(Y,GY)
  CALL CONVRG(GY,IPASS)
  IF(IPASS.EQ.1) GO TO 70
C CONVERGENCE CRITERIA NOT SATISFIED. FIND A NEW DIRECTION MATRIX
  IF(IPRINT.EQ.1) PRINT 2000, ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
  DO 50 I=1,N
    DELG(I)=GY(I)-GX(I)
    DELX(I)=Y(I)-X(I)
    GX(I)=GY(I)
50   X(I)=Y(I)
    FX=FY
    CALL ETA
C SET UP NEW SEARCH DIRECTION
  DO 60 I=1,N
    S(I)=0.
    DO 60 J=I,N
60   S(I)=S(I)+H(I,J)*GY(J)
    GO TO 40
C CONVERGENCE CRITERIA SATISFIED
70   PRINT 2000, ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
2000 FORMAT(1X,3I7,E16.8,(5E16.8))
  RETURN
  END

```

```

SUBROUTINE MINI
C FLETCHER-REEVES CONJUGATE GRADIENT METHOD, SECTION 3.3-2
COMMON /ONE/ X(10),Y(10),S(10),FX,FY
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
  ITER=0
  IRESET=N+1
  INDEX=IRESET
C EVALUATE STARTING POINT
  CALL FUN(X,FX)
  CALL DER(X,GX)
  PRINT 2000, ITER,NFUNCT,NDRV,FX,(X(I),I=1,N)
C CALCULATE SQUARED NORM OF GRADIENT
10   SQNOR1=0.
  DO 20 I=1,N
20   SQNOR1=SQNOR1+GX(I)*GX(I)
  IF(INDEX.NE.IRESET) GO TO 50
C SET SEARCH DIRECTION TO NEGATIVE GRADIENT
30   IF(IPRINT.EQ.1) PRINT 2100
2100 FORMAT(* GRADIENT STEP*)
  INDEX=0
  DO 40 I=1,N
40   S(I)=-GX(I)
  GO TO 70
C SET SEARCH DIRECTION USING RATIO OF SQUARED NORMS
50   DO 60 I=1,N
60   S(I)=-GX(I)+S(I)*SQNOR1/SQNOR2
C FIND NEXT POINT
70   CALL SEARCH
C CHECK WHETHER SEARCH WAS A SUCCESS. IF NOT TAKE A GRADIENT STEP.
  IF(FY.GE.FX) GO TO 30
  CALL DER(Y,GX)
  INDEX=INDEX+1
  ITER=ITER+1
  CALL CONVRG(GX,IPASS)
  IF(IPASS.EQ.1) GO TO 90
C CONVERGENCE CRITERIA NOT SATISFIED. CONTINUE SEARCH.
  IF(IPRINT.EQ.1) PRINT 2000, ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
C SAVE INFORMATION FOR NEXT STAGE
  DO 80 I=1,N

```

```

      DELX(I)=Y(I)-X(I)
80    X(I)=Y(I)
      FX=FY
      SQNOR2=SQNOR1
      GO TO 10
C CONVERGENCE CRITERIA SATISFIED
90    PRINT 2000, ITER,NFUNCT,NDRV,FY,{Y(I),I=1,N}
2000 FORMAT(1X,3I7,E16.8,(5E16.8))
      RETURN
END

```

```

C SUBROUTINE ETA
C DAVIDON, FLETCHER-POWELL METHOD, SECTION 3.4-2
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION HDG(10),OGH(10)
DXDG=0.
DGHDG=0.
DO 20 I=1,N
HDG(I)=DGH(I)=0.
DO 10 J=1,N
HDG(I)=HDG(I)-H(I,J)*DELG(J)
10   DGH(I)=DGH(I)+DELG(J)*H(J,I)
DXDG=DXDG+DELX(I)*DELG(I)
20   DGHDG=DGHDG+DGH(I)*DELG(I)
DO 30 I=1,N
DO 30 J=1,N
30   H(I,J)=H(I,J)+DELX(I)*DELX(J)/DXDG+HDG(I)*DGH(J)/DXDG
      RETURN
END

```

```

C SUBROUTINE ETA
C BROYDEN METHOD, SECTION 3.4-1
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION DXHDG(10)
DGTERM=0.
DO 20 I=1,N
DXHDG(I)=DELX(I)
DO 10 J=1,N
10   DXHDG(I)=DXHDG(I)-H(I,J)*DELG(J)
20   DGTERM=DGTERM+DXHDG(I)*DELG(I)
DO 30 I=1,N
DO 30 J=1,N
30   H(I,J)=H(J,I)*H(I,J)+DXHDG(I)*DXHDG(J)/DGTERM
      RETURN
END

```

```

C SUBROUTINE ETA
C PEARSON 2' METHOD, SECTION 3.4
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION DXHDG(10)
DXDG=D.
DO 20 I=1,N
DXHDG(I)=DELX(I)
DO 10 J=1,N
10   DXHDG(I)=DXHDG(I)-H(I,J)*DELG(J)
20   DXDG=DXDG+DELX(I)*DELG(I)
DO 30 I=1,N
DO 30 J=1,N
30   H(I,J)=H(I,J)+DXHDG(I)*DELX(J)/DXDG
      RETURN
END

```

SUBROUTINE ETA

```

C PEARSON 3 METHOD, SECTION 3.4
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION DXHDG(10),DGH(10)
DGHDG=0.
DO 20 I=1,N
DXHDG(I)=DELX(I)
DGH(I)=0.
DO 10 J=1,N
DXHDG(I)=DXHDG(I)-H(I,J)*DELG(J)
10 DGH(I)=DGH(I)+DELG(J)*H(J,I)
20 DGHDG=DGHDG+DGH(I)*DELG(I)
DO 30 I=1,N
DO 30 J=1,N
H(I,J)=H(I,J)+DXHDG(I)*DGH(J)/DGHDG
30 RETURN
END

```

SUBROUTINE ETA

```

C PROJECTED NEWTON METHOD, SECTION 3.4 {SAME AS ZOUTENDEK PROJECTION
C METHOD, SECTION 3.3-4}
C THE H MATRIX MUST RESET TO THE IDENTITY MATRIX EVERY N STEPS
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION HDG(10),DGH(10)
DGHDG=0.
DO 20 I=1,N
HDG(I)=0.
DGH(I)=0.
DO 10 J=1,N
HDG(I)=HDG(I)+H(I,J)*DELG(J)
10 DGH(I)=DGH(I)+DELG(J)*H(J,I)
20 DGHDG=DGHDG+DELG(I)*HDG(I).
DO 30 I=1,N
DO 30 J=1,N
30 H(I,J)=H(I,J)-DGH(I)*HDG(J)/DGHDG
RETURN
END

```

SUBROUTINE MINI

```

C GOLDSTEIN-PRICE METHOD, SECTION 3.4
C VERSION 2. THE SEARCH DIRECTION (PHI) IS SET TO THE GRADIENT ONLY ON
C THE FIRST STEP AND WHEN Q IS SINGULAR
COMMON /ONE/ X(10),Y(10),PHI(10),FX,FY
COMMON /TWO/ Q(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
DIMENSION GY(10)
R=0.0001
THETA=R
ITER=0
C EVALUATE STARTING POINT
CALL FUN(X,FX)
CALL OER(X,GX)
C PRINT POINT
PRINT 2400, ITER,NFUNCT,NDRV,FX,(X(I),I=1,N)
GO TO 70
C GENERATE THE Q MATRIX
10 DO 30 J=1,N
XSAVE=X(J)
X(J)=X(J)+THETA
CALL OER(X,GY)
DO 20 I=1,N
20 Q(I,J)=(GY(I)-GX(I))/THETA
30 X(J)=XSAVE
C INVERT Q

```

```

CALL SYMINV(Q,N,10,ISF)
IF((ISF.NE.1) GO TO 40
1F((IPRINT.EQ.1) PRINT 2200
2200 FORMAT(1X,*Q IS SINGULAR*)
GO TO 70
C FORM THE PHI VECTOR USING Q
40 DO 50 I=1,N
PHI(I)=0.
DO 50 J=1,N
50 PHI(I)=PHI(I)+Q(I,J)*GX(J)
GO TO 110
70 IF ((IPRINT.EQ.1) PRINT 2100
2100 FORMAT(1X,*GRADIENT STEP*)
C SET PHI TO THE GRADIENT
DO 80 I=1,N
80 PHI(I)=GX(I)
110 CONTINUE
CALL SEARCH
IF(FY.GE.FX) GO TO 70
C CALCULATE THETA FOR NEXT ITERATION
PHNORM=0.
DO 120 I=1,N
120 PHNORM=PHNORM+PHI(I)*PHI(I)
PHNORM=SQRT(PHNORM)
THETA=R*PHNORM
C TEST FOR CONVERGENCE
ITER=ITER+1
CALL DER(Y,GX)
CALLCONVRG(GX,IPASS)
IF(IPASS.EQ.1) GO TO 140
C CONVERGENCE CRITERIA NOT SATISFIED
1F((IPRINT.EQ.1) PRINT 2000, ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
FX=FY
DO 130 I=1,N
DELX(I)=Y(I)-X(I)
130 X(I)=Y(I)
GO TO 10
C CONVERGENCE CRITERIA SATISFIED
140 PRINT 2000, ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
2000 FORMAT(1X,317,E16.8,(5E16.8))
RETURN
END
SUBROUTINE SYMINV(A,NC,ND,ISF)
DIMENSION A(ND,ND),T(20),Q(20),R(20)
ZERO=0.0 $ ONE=1.0 $ ISF=0 $ DO 21 M=1,NC
21 R(M)=ONE
DO 38 M=1,NC $ BIG=ZERO
DO 24 L=1,NC $ AB=ABS(A(L,L)) $ IF(AB-BIG)24,24,22
22 IF(R(L))23,24,23
23 BIG=AB $ K=L
24 CONTINUE $ IF(BIG)26,25,26
25 PRINT 13 $ ISF=1 $ RETURN
13 FORMAT(10X,23HMATRIX INVERSION FAILED)
26 R(K)=ZERO $ Q(K)=ONE/A(K,K) $ T(K)=ONE $ A(K,K)=ZERO $ KM1=K-1
IF(KM1.EQ.0)31,27
27 DO 30 L=1,KM1 $ T(L)=A(L,K) $ IF(R(L))29,28,29
28 Q(L)=A(L,K)*Q(K) $ GO TO 30
29 Q(L)=-A(L,K)*Q(K)
30 A(L,K)=ZERO
31 CONTINUE $ KP1=K+1 $ IF(KP1.GT.NC)37,32
32 DO 36 L=KP1,NC $ IF(R(L))33,34,33
33 T(L)=A(K,L) $ GO TO 35
34 T(L)=-A(K,L)
35 Q(L)=-A(K,L)*Q(K)
36 A(K,L)=ZERO
37 CONTINUE
DO 38 L=1,NC $ DO 38 K=L,NC
38 A(L,K)=A(L,K)+T(L)*Q(K) $ M=NC+1 $ L=NC

```

```

      DO 39 K=2,NC $ M=M-1 $ L=L-1 $ DO 39 J=1,L
39   A(M,J)=A(J,M)           $      END
      RETURN

      SUBROUTINE SEARCH
C COGGIN METHOD OF UNIVIMENSIONAL SEARCH
      COMMON /ONE/ X(10),Y(10),S(10),FX,FY
      COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
      COMMON /THREE/ N,NFUNC,NDRV,ITER,INDIC,IPRINT
C *** THE INITIAL VARIABLE VALUES ARE IN X, AND THE CORRESPONDING
C *** FUNCTION VALUE IS FX.
C *** THE SEARCH DIRECTION VECTOR IS S, AND THE INITIAL STEP SIZE STEP.
      IEXIT=0
      NTOL=0
      FTOL=.001
      FTOL2=FTOL/100.
      FA=FB=FC=FX
      DA=DB=DC=0.
      K=-2
      M=0
      STEP=1.0
      D=STEP
C USE THE PARAMETER INDIC TO INDICATE HOW THE SEARCH VECTOR LENGTH
C SHOULD BE SCALED.
C     INDIC=2 DU NOT SCALE. TAKE LENGTH GIVEN BY MINI CALCULATION.
C     INDIC=1 SCALE ONLY IF THE LENGTH OF THE LAST STEP WAS SHORTER THAN
C             THE LENGTH OF THE SEARCH VECTOR. SCALE TO LENGTH OF LAST STEP.
C     INDIC=ANYTHING BUT 1 OR 2 RESULTS IN SCALING TO LENGTH OF LAST STEP.
      IF(INDIC.EQ.2.OR.ITER.EQ.0) GO TO 1
C FIND NORM OF S AND NORM OF DELX
      DXNORM=0.
      SNORM=0.
      DO 102 I=1,N
      DXNORM=DXNORM+DELX(I)*DELX(I)
102   SNORM=SNORM+S(I)*S(I)
      IF(INDIC.EQ.1.AND.DXNORM.GE.SNORM) GO TO 1
      RATIO=DXNORM/SNORM
      STEP=SQRT(RATIO)
      D=STEP
C *** START THE SEARCH THE BOUND THE MINIMUM
      1 DO 2 I=1,N
      2 Y(I)=X(I)+D*S(I)
      CALL FUN(Y,F)
      K=K+1
      IF(F-FA) 5,3,6
C *** NO CHANGE IN FUNCTION VALUE. RETURN WITH VECTOR CORRESPONDING TO
C FUNCTION VALUE OF FA, BECAUSE IF THE FUNCTION VALUE IS INDEPENDENT
C OF THIS SEARCH DIRECTION, THEN CHANGES IN THE VARIABLE VALUES MAY
C UPSET THE MAIN PROGRAM CONVERGENCE TESTING.
      3 DO 4 I=1,N
      4 Y(I)=X(I)+DA*S(I)
      FY=FA
      IF(IPRINT.EQ.1) PRINT 2100
2100  FORMAT(* SEARCH FAILED. FUNCTION VALUE INDEPENDENT OF SEARCH DIRE
ACTION*)
      GO TO 326
C *** THE FUNCTION IS STILL DECREASING. INCREASE THE STEP SIZE BY
C DOUBLE THE PREVIOUS INCREASE IN STEP SIZE.
      5 FC=FB $ FB=FA $ FA=F
      DC=DB $ DB=DA $ DA=D
      D=2.*D+STEP
      GO TO 1
C *** MINIMUM IS BOUNDED IN AT LEAST ONE DIRECTION.
      6 IF(K) 7,8,9
C     MINIMUM IS BOUNDED IN ONE DIRECTION ONLY. REVERSE THE SEARCH
C     DIRECTION AND RECYCLE.

```

```

7 FB=F
DB=D $ D=-D $ STEP=-STEP
GO TO 1
C MINIMUM IS BOUNDED IN BOTH DIRECTIONS AFTER ONLY TWO FUNCTION
C EVALUATIONS !ONE EITHER SIDE OF THE ORIGINZ. PROCEED TO THE
C PARABOLIC INTERPOLATION.
8 FC=FB $ FB=FA $ FA=F
DC=DB $ DB=DA $ DA=D
GO TO 21
C THE MINIMUM IS BOUNDED AFTER AT LEAST TWO FUNCTION EVALUATIONS IN
C THE SAME DIRECTION. EVALUATE THE FUNCTION AT STEP SIZE=(DA+DB)/2.
C THIS WILL YIELD 4 EQUALLY SPACED POINTS BOUNDING THE MINIMUM.
9 DC=DB $ DB=DA $ DA=D
FC=FB $ FB=FA $ FA=F
10 D=0.5*(DA+DB)
DO 11 I=1,N
11 Y(I)=X(I)+D*S(I)
CALL FUN(Y,F)
C *** NOW HAVE THAT FA>FOFC AND THAT FA<FOFC ASSUMING THAT THE
C FUNCTION IS UNIMODAL. REMOVE EITHER POINT A OR POINT B IN SUCH A
C WAY THAT THE FUNCTION IS BOUNDED AND FA>FOFC !THE CORRESPONDING
C STEP SIZES ARE DA>DB>DC OR DA<DB>DC Z.
12 IF((DC-D)*(D-DB)) 15,13,18
C *** LOCATION OF MINIMUM IS LIMITED BY ROUNDING ERRORS. RETURN WITH B.
13 DO 14 I=1,N
14 Y(I)=X(I)+DB*S(I)
FY=FB
IF(IEXIT.EQ.1) GO TO 32
IF(IPRINT.EQ.1) PRINT 2200
2200 FORMAT(* SEARCH FAILED. LOCATION OF MINIMUM LIMITED BY ROUNDING*)
GO TO 325
C *** THE POINT D IS IN THE RANGE DA TO DB.
15 IF(F>FB) 16+13,17
16 FC=FB $ FB=F
DC=DB $ DB=D
GO TO 21
17 FA=F
DA=D
GO TO 21
C *** THE POINT D IS IN THE RANGE DB TO DC
18 IF(F<FB) 19,13,20
19 FA=FB $ FB=F
DA=DB $ DB=D
GO TO 21
20 FC=F
DC=D
C *** NOW PERFORM THE PARABOLIC INTERPOLATION.
21 A=FA*(DB-DC)+FB*(DC-DA)+FC*(DA-DB)
IF(A) 22,30,22
22 D=0.5*((DB*DB-DC*DC)*FA+(DC*DC-DA*DA)*FB+(DA*DA-DB*DB)*FC)/A
C CHECK THAT THE POINT IS GOOD. IF SO, EVALUATE THE FUNCTION.
IF((DA-D)*(D-DC)) 13,13,23
23 LO 24 I=1,N
24 Y(I)=X(I)+D*S(I)
CALL FUN(Y,F)
C *** CHECK FOR CONVERGENCE. IF NOT ACHIEVED, RECYCLE.
IF(ABS(FB)-FTOL2) 25,25,26
25 A=1.0 $ GO TO 27
26 A=1.0/FB
27 IF((ABS(FB-F)*A)-FTOL) 28,28,12
C *** CONVERGENCE ACHIEVED. RETURN WITH THE SMALLER OF F AND FB.
28 IEXIT=1
IF(F>FB) 29,13,13
29 FY=F
GO TO 32
C *** THE PARABOLIC INTERPOLATION WAS PREVENTED BY THE DIVISOR BEING
C ZERO. IF THIS IS THE FIRST TIME THAT IT HAS HAPPENED, TRY AN
C INTERMEDIATE STEP SIZE AND RECYCLE; OTHERWISE GIVE UP AS IT LOOKS

```

```

C   LIKE A LOST CAUSE*
30 IF(M> 31,31,13
31 M=M+1
GO TO 10
32 DO 99 I=1,N
IF(Y(I)).NE.X(I)) GO TO 325
99 CONTINUE
GO TO 33
325 IF(INTOL.NE.0.AND.IPRINT.EQ.1) PRINT 3000, NTOL
3000 FORMAT(1X,*TOLERANCE REDUCED *,I1,* TIME(S)*)
326 IF(FY.LT.FX) RETURN
IF(S(I).NE.-GX(I)).OR.(FY.LT.FX)) RETURN
PRINT5000
5000 FORMAT(* SEARCH FAILED ON A GRADIENT STEP.  JOB TERMINATED.*)
PRINT 5100, ITER,NFUNC,NDRV,FY,(Y(I),I=1,N)
5100 FORMAT(1X,3I7,E16.8,(5E16.8))
STOP
33 IF(INTOL.EQ.5) GO TO 34
IEXIT=0
NTOL=NTOL+1
FTOL=FTOL/10.
GO TO 12
34 IF(IPRINT.EQ.1) PRINT 2000
2000 FORMAT(* A POINT BETTER THAN THE ENTERING POINT CANNOT BE FOUND.*)
RETURN
END

```

```

SUBROUTINE SEARCH
C UNIDIMENSIONAL SEARCH USING GOLDEN SECTION. VERSION 2, MOD 4.
COMMON /ONE/ X(10),Y(10),S(10),FX,FY
COMMON /TWO/ H(10,10),DELX(10),DELG(10),GX(10)
COMMON /THREE/ N,NFUNC,NDRV,ITER,INDIC,IPRINT
DIMENSION Z(10),W(10),P(10),R(10),DIFF(10),SS(10)
DATA F1/0.618033989/
C P=OLDEST OF LAST THREE POINTS
C Z=MIDDLE POINT
C W=CURRENT POINT
NTRIES=0
NTOL=0
C TUL=SQUARED NORM OF VECTOR FROM W TO Z WHICH MUST BE ACHIEVED FOR
C CONVERGENCE
TOL=0.000001
NTIMES=0
C USE THE PARAMETER INDIC TO INDICATE HOW THE SEARCH VECTOR LENGTH
C SHOULD BE SCALED.
C INDIC=2 DO NOT SCALE. TAKE LENGTH GIVEN BY MINI CALCULATION
C INDIC=1 SCALE ONLY IF THE LENGTH OF THE LAST STEP WAS SHORTER THAN
C THE LENGTH OF THE SEARCH VECTOR. SCALE TO LENGTH OF LAST STEP.
C INDIC=ANYTHING BUT 1 OR 2 RESULTS IN SCALING TO LENGTH OF LAST STEP.
IF(INDIC.EQ.2.OR.ITER.EQ.0) GO TO 4
C NORMALIZE THE SEARCH VECTOR TO THE LENGTH USED ON THE PREVIOUS STEP.
DXNORM=0.
SNORM=0.
DO 2 I=1,N
DXNORM=DXNORM+DELX(I)*DELX(I)
2 SNORM=SNORM+S(I)*S(I)
IF(INDIC.EQ.1. AND.DXNORM.GE.SNORM) GO TO 4
DXNORM=SQRT(DXNORM)
SNORM=SQRT(SNORM)
RATIO=DXNORM/SNORM
DO 3 I=1,N
3 SS(I)=S(I)*RATIO
GO TO 10
C MAINTAIN THE INTEGRITY OF THE SEARCH VECTOR BY CONSTRUCTING AN IDENTICAL
C VECTOR AND OPERATING ON IT.
4 DO 5 I=1,N
5 SS(I)=S(I)

```

```

C***BRACKET THE MINIMUM IN THE S DIRECTION
C TAKE STEP FROM ORIGINAL POINT
10   DO 20 I=1,N
      Z(I)=X(I)
20   W(I)=X(I)+SS(I)
      FZ=FX
      NTIMES=NTIMES+1
      CALL FUN(W,FW)
      IF(FW-FZ) 30,70,50
C CONTINUE SEARCH IN SAME DIRECTION
30   DO 40 I=1,N
      P(I)=Z(I)
      Z(I)=W(I)
      SS(I)=2.*SS(I)
40   W(I)=W(I)+SS(I)
      FP=FX
      FZ=FW
      NTIMES=NTIMES+1
      CALL FUN(W,FW)
      IF(FW-FZ) 30,70,120
C FW.GT.FZ, DECIDE WHETHER TO REVERSE SEARCH DIREC
50   IF(NTIMES.NE.1) GO TO 120
C REVERSE SEARCH DIRECTION
      DO 60 I=1,N
      SS(I)=-SS(I)
60   P(I)=W(I)
      FP=FW
      GO TO 10
C FZ=FW, CHECK MIDPOINT
70   DO 80 I=1,N
80   R(I)=(Z(I)+W(I))/2.
      NTIMES=NTIMES+1
      CALL FUN(R,FR)
      MIN=1
      IF(FR-FZ) 140,300,90
90   IF(NTIMES.NE.2) GO TO 110
C REVERSE SEARCH DIRECTION
      DO 100 I=1,N
      SS(I)=-SS(I)
100  P(I)=R(I)
      FP=FR
      GO TO 10
C R AND P BRACKET Z AND THE MINIMUM
110  DO 115 I=1,N
      W(I)=R(I)
      R(I)=Z(I)
115  Z(I)=P(I)
      MIN=1
      FW=FR
      FR=FZ
      FZ=FP
      GO TO 140
C P AND W BRACKET Z AND THE MINIMUM
120  DO 130 I=1,N
      R(I)=Z(I)
130  Z(I)=P(I)
      MIN=1
      FR=FZ
      FZ=FP
C***GOLDEN SEARCH) Z AND W BRACKET THE MINIMUM
140  WZNORM=0.
      DO 145 I=1,N
      DIFF(I)=W(I)-Z(I)
145  WZNORM=WZNORM+DIFF(I)*DIFF(I)
      IF(WZNORM.LT.TOL) GO TO 290
146  DO 150 I=1,N
      SECT=F1*DIFF(I)
      P(I)=Z(I)+SECT

```

```

150   R(I)=W(I)-SECT
      CALL FUN(P,FP)
      CALL FUN(R,FR)
160   IF(FR-FP) 170,230,200
C   REPLACE W BY P AND P BY R
170   WZNORM=0.
      DO 180 I=1,N
      W(I)=P(I)
      P(I)=R(I)
      DIFF(I)=W(I)-Z(I).
      WZNORM=WZNORM+DIFF(I)*DIFF(I)
180   R(I)=W(I)-F1*DIFF(I)
      FW=FP
      FP=FR
      IF(WZNORM.LT.TOL) GO TO 320
      CALL FUN(R,FR)
      GO TO 160
C   REPLACE Z BY R AND R BY P
200   WZNORM=0.
      DO 210 I=1,N
      Z(I)=R(I)
      R(I)=P(I)
      DIFF(I)=W(I)-Z(I)
      WZNORM=WZNORM+DIFF(I)*DIFF(I)
210   P(I)=Z(I)+F1*DIFF(I)
      FZ=FR
      FR=FP
      IF(WZNORM.LT.TOL) GO TO 300
      CALL FUN(P,FP)
      GO TO 160
C   FP=FR, CHECK MIDPOINT
230   DO 240 I=1,N
240   Y(I)=(P(I)+R(I))/2.
      CALL FUN(Y,FY)
      IF(FY-FP) 250,340,270
C   P AND R BRACKET THE MINIMUM (Y IS BRACKETED)
250   DO 260 I=1,N
      Z(I)=R(I)
      W(I)=P(I)
260   R(I)=Y(I)
      FZ=FR
      FW=FP
      FR=FY
      MIN=1
      GO TO 140
C   THERE ARE TWO MINIMA BETWEEN Z AND W.  ARBITRARILY PICK THE INTERVAL
C   BETWEEN Y AND W (WHICH INCLUDES P)
270   DO 280 I=1,N
280   Z(I)=Y(I)
      FZ=FY
      MIN=2
      GO TO 140
C   BRACKET ON THE MIN IS SUFFICIENTLY SMALL
290   GO TO (300,320), MIN
C   R IS THE POINT INSIDE THE BRACKET
300   DO 310 I=1,N
310   Y(I)=R(I)
      FY=FR
      GO TO 340
C   P IS THE POINT INSIDE THE BRACKET
320   DO 330 I=1,N
330   Y(I)=P(I)
      FY=FP
340   CONTINUE
      DO 345 I=1,N
345   IF(X(I).NE.Y(I)) GO TO 346
      GO TO 350
346   IF(FY.GE.FX) GO TO 370

```

```

IF(IPRINT,NE,1) RETURN
IF(NTOL,NE,0) PRINT 3000,NTOL
IF(INTRIES,NE,0) PRINT 3100
3000 FORMAT(1X,*TOLERANCE REDUCED #,1I,* TIME(S)*)
3100 FORMAT(1X,*SECOND TRY.*)
RETURN

C## TAKE CARE OF PATHOLOGICAL CONDITIONS
C AT THE PRESENT TOLERANCE LEVEL NO POINT CAN BE FOUND WHICH
C IS BETTER THAN THE ENTERING POINT. REDUCE TOL BY A FACTOR OF 100.
350 IF(NTOL,EQ,5) GO TO 360
NTOL=NTOL+1
TOL=TOL/100.
GO TO 146

C PRINT MESSAGE AND RETURN
360 IF(IPRINT,NE,1) GO TO 376
IF(INTRIES,NE,0) GO TO 375
PRINT 2400,TOL
2000 FORMAT(1X,*THE TOLERANCE HAS BEEN REDUCED 5 TIMES TO A CURRENT VAL
AUE OF#.E15.8,*#,/,#
B1X. *A POINT BETTER THAN THE ENTERING POINT CANNOT BE FOUND AT TH
CIS LEVEL OF TOLERANCE. THE ENTERING POINT IS BEING RETURNED.*)
GO TO 376

C## FY.GT.FX, FIND A BRACKET EXCLUDING THE VALLEY CONTAINING Y.
370 IF(INTRIES,EQ,0) GO TO 380
IF(NTOL,LT,5) GO TO 350
375 PRINT 2100
PRINT 3000,NTOL
2100 FORMAT(1X,*A POINT WAS FOUND SUCH THAT FY WAS GREATER THAN FX. A
BSECOND ATTEMPT TO FIND A POINT WITH A FUNCTION VALUE LESS THAN FX
CFAILED.*)
376 IF(S(1),NE,-GX(1).OR.(FY,LT,FX)) RETURN
PRINT 2200
2200 FORMAT(* SEARCH FAILED ON A GRADIENT STEP. JOB TERMINATED*)
PRINT 2300,ITER,NFUNCT,NDRV,FY,(Y(I),I=1,N)
2300 FORMAT(1X,3I7,E16.8,(5E16.8))
STOP

C LOOK FOR A BRACKET NEAR X ON THE SIDE OPPOSITE FROM Y.
380 NTRIES=1
DO 390 I=1,N
Z(I)=X(I)
SS(I)=(X(I)-Y(I))/20.
390 W(I)=X(I)+SS(I)
FZ=FX
CALL FUN(W,FW)
IF(FW-FZ) 30,400,450
C FZ=FW, CHECK MIDPOINT
400 DO 410 I=1,N
410 P(I)=(Z(I)+W(I))/2.
CALL FUN(P,FP)
IF(FP-FZ) 420,320,430

C FZ.GT.FP AND FW.GT.FP. W AND Z FORM A BRACKET
420 MIN=2
GO TO 140
C FP.GT.FZ, P IS A BRACKET WITH THE MINIMUM BETWEEN P AND Y
430 DO 440 I=1,N
440 W(I)=P(I)
FW=FP

C FW.GT.FZ, W IS A BRACKET WITH THE MINIMUM BETWEEN W AND Y
C CHECK MIDPOINT OF Y AND W
450 DO 460 I=1,N
Z(I)=Y(I)
460 P(I)=(Z(I)+W(I))/2.
FZ=FY
470 CALL FUN(P,FP)
IF(FP,LE,FZ) GO TO 490

C TRY AGAIN
DO 480 I=1,N
Z(I)=P(I)

```

```

480  P(I)=(W(I)+Z(I))/2.
  FZ=FP
  GO TO 470
C  P IS IN A VALLEY DIFFERENT FROM THE ONE CONTAINING Y.
C  CHECK WHETHER P AND W FORM A BRACKET
490  DO 500 I=1,N
500  R(I)=(P(I)+W(I))/2.
  CALL FUN(R,FR)
  IF(FR.GE.FW) GO TO 560
  IF(FR.GE.FP) GO TO 520
C  P AND W FORM A BRACKET
505  DO 510 I=1,N
510  Z(I)=P(I)
  FZ=FP
  MIN=1
  GO TO 140
C  FR.LT.FW, FR.GE.FP.  HENCE LOOK FOR A BRACKETING VALUE BETWEEN P AND Z.
520  DO 530 I=1,N
  R(I)=P(I)
530  P(I)=(P(I)+Z(I))/2.
  FR=FP
540  CALL FUN(P,FP)
  IF(FP.LE.FZ) GO TO 490
C  FP.GT.FZ.  APPARENTLY Z IS IN THE SAME VALLEY WITH Y.
  DO 550 I=1,N
  Z(I)=P(I)
550  P(I)=(R(I)+Z(I))/2.
  FZ=FP
  GO TO 540
C  FR.GE.FW
560  IF(IFR.LT.FP) GO TO 505
C  FR.GE.FP, R AND W FORM A BRACKET
  DO 570 I=1,N
570  Z(I)=R(I)
  FZ=FR
  GO TO 146
END

```

Б.2. ПРОГРАММА ОПТИМИЗАЦИОННОГО МЕТОДА ПАУЭЛЛА

Отдельная подпрограмма MINI была подготовлена для реализации оптимизационного алгоритма Паузелла, предназначенного для решения задач нелинейного программирования при отсутствии ограничений (без использования производных). Следует отметить, что здесь DER не используется, а INDIC присваивается значение 2. Специальные инструкции относительно сходимости содержатся в комментирующих картах в подпрограмме MINI. Следует положить ICONVG = 1, если один проход по алгоритму Паузелла будет достаточным, или ICONVG = 2, если окончательное решение должно быть изменено и найдено новое и если необходимо выполнить экстраполяцию между ними. Подпрограмму MINI должна сопровождать специальная подпрограмма TEST.

Помимо указанных выше отличий подпрограмма MINI для метода Паузелла полностью совместима с программой EXEC и должна использоваться вместе с нею. Программа EXEC была рассмотрена выше.

```

      SUBROUTINE MINI
C POWELL METHOD OF DIRECT SEARCH
C SUBROUTINE TEST MUST BE PROVIDED FOR CONVERGENCE TESTING
      COMMON /ONE/ X(10),Y(10),S(10),FX,FY
      COMMON /TWO/ DIRECT(10,10),DUM(10),BEFORE(10),FIRST(10)
      COMMON /THREE/ N,NFUNCT,NDRV,ITER,INDIC,IPRINT
      DIMENSION W(10),SECND(10)
      EQUIVALENCE (W,SECND)

C *** N = THE NUMBER OF VARIABLES.
C   ICONVG = THE FINAL CONVERGENCE TEST DESIRED.
C           = 1, TERMINATE AS SOON AS TESTING IS SATISFIED.
C           = 2, AS SOON AS THE TESTING CRITERIA ARE SATISFIED INCREASE
C               ALL THE VARIABLES BY 10*ACC AND SOLVE PROBLEM AGAIN.
C   THEN PERFORM A LINE SEARCH BETWEEN THE SOLUTIONS IF DIFFERENT
C   SOLUTIONS ARE DEEMED TO BE FOUND.
C   STEP = THE INITIAL STEP SIZE.
C   ACC = THE REQUIRED ACCURACY IN THE FUNCTION AND VECTOR VALUES.
C   INSERT IPRINT= 1 FOR COMPLETE PRINT OUT OR IPRINT = 2 FINAL
C   ANSWER ONLY
C   ACC=0.00001
C   STEP=1.0
C INDIC MUST BE SET TO 2
      INDIC=2
      ICONVG=1
      ITER=0
      NTRY=1
      N1=N-1
      STEPA=STEP
C *** SET UP THE INITIAL DIRECTION MATRIX (USING UNIT VECTORS).
      DO 2 I=1,N
      DO 1 J=1,N
      1 DIRECT(J,I)=0.
      2 DIRECT(I,I)=1.
C *** EVALUATED THE FUNCTION AT THE INITIAL VARIABLE VALUES.
      100 CALL FUN(X,FX)
      PRINT 2000, ITER,NFUNCT,FX,(X(I),I=1,N)
      2000 FORMAT(1X,2I7,E16.8,(5E16.8))
      GO TO 301
C *** SAVE THE FINAL FUNCTION VALUE (F1) AND THE FINAL VARIABLE VALUES
C (BEFORE) FROM THE PREVIOUS CYCLE.
      3 ITER=ITER+1
      IF(IPRINT,EQ,1) PRINT 2000, ITER,NFUNCT,FX,(X(I),I=1,N)
      301 F1=FX
      DO 4 I=1,N
      4 BEFORE(I)=X(I)
      SUM=0.
C   AT THE END OF THE CYCLE, SUM WILL CONTAIN THE MAXIMUM CHANGE IN
C   THE FUNCTION VALUE FOR ANY SEARCH DIRECTION, AND ISAVE INDICATES
C   THE DIRECTION VECTOR TO WHICH IT CORRESPONDS.
      DO 9 I=1,N
C   S CONTAINS THE INITIAL STEP SIZES IN THE I-TH DIRECTION.
      DO 5 J=1,N
      5 S(J)=DIRECT(J,I)*STEP
      FIND THE MINIMUM IN THE I-TH DIRECTION, AND THE CHANGE IN FUNCTION
      C   VALUE.
      CALL SEARCH
      A=FX-FY
      IF(A-SUM) 7,7,6
      6 ISAVE=I
      SUM=A
C   TRANSFER THE NEW FUNCTION AND VARIABLE VALUES TO FX AND X.
      7 DO 8 J=1,N
      8 X(J)=Y(J)
      9 FX=FY
C *** NOW INVESTIGATE WHETHER A NEW SEARCH DIRECTION SHOULD BE INCORPOR-

```

```

C      ATED INSTEAD OF THE ISAVE DIRECTION.
F2=FX
DO 10 I=1,N
10 W(I)=2.0*X(I)-BEFORE(I)
CALL FUN(W,F3)
A=F3-F1
IF(A) 11,19,19
11 A=2.0*(F1-2.0*F2+F3)*((F1-F2-SUM)/A)**2
IF(A-SUM) 12,19,19
C *** A NEW SEARCH DIRECTION IS REQUIRED. FIRST REMOVE ROW ISAVE.
12 IF(ISAVE=N) I3=15+15
13 DO 14 I=ISAVE,N1
   II=I+1
   DO 14 J=1,N
14 DIRECT(I,J)=DIRECT(J,I))
C      SET THE N-TH DIRECTION VECTOR EQUAL TO THE NORMALISED DIFFERENCE
C      BETWEEN THE INITIAL AND FINAL VARIABLE VALUES FOR LAST CYCLE.
15 A=0.
DO 16 J=1,N
DIRECT(J,N)=X(J)-BEFORE(J)
16 A=DIRECT(J,N)**2+A
A=1.0/SQRT(A)
DO 17 J=1,N
DIRECT(J,N)=DIRECT(J,N)*A
17 S(J)=DIRECT(J,N)*STEP
CALL SEARCH
FX=FY
DO 18 I=1,N
18 X(I)=Y(I)
C *** TEST FOR CONVERGENCE.
19 CALL TEST(F1,FX,BEFORE,X,FLAG,N,ACC)
IF(FLAG) 22,22,20
C *** CONVERGENCE NOT YET ACHEIVED. COMPUTE A NEW STEP SIZE AND
C      GO BACK TO 3.
20 IF(F1-FX)>120,120,120
121 STEP=-0.4*SQRT(ABS(F1-FX))
GO TO 123
120 STEP=0.4*SQRT(F1-FX)
123 IF(STEPA=STEP) 21,3,3
21 STEP=STEPA
GO TO 3
C *** CONVERGENCE ACHEIVED. IF ICONVG=2, INCREASE ALL VARIABLES BY
C      10*ACC AND GO BACK TO 3.
22 GO TO {23,24},ICONVG
23 RETURN
24 GO TO (25,27),NTRY
25 NTRY=2
DO 26 I=1,N
FIRST(I)=X(I)
26 X(I)=X(I)+ACC*10.
FFIRST=FX
GO TO 100
C *** CONVERGENCE ATTAINED USING TWO DIFFERENT STARTING POINTS. CONSTRUC
C      UNIT VECTOR BETWEEN SOLUTIONS AND SEARCH DIRECTION FOR A MINIMUM.
27 FSECND=FX
A=0.
DO 28 I=1,N
SECND(I)=X(I)
S(I)=FIRST(I)-SECND(I)
28 A=A+S(I)**2
IF(A) 23,23,29
29 A=STEP/SQRT(A)
DO 30 I=1,N
30 S(I)=S(I)*A
CALL SEARCH
C *** TEST IF NEW POINT IS SUFFICIENTLY CLOSE TO EITHER OF THE TWO
C      SOLUTIONS. IF SO RETURN.
CALL TEST(IFIRST,FY,FIRST,Y,FLAG,N,ACC)

```

```

1 IF(FLAG) 32,32,31
31 CALL TEST(FSECND,FY,SECND,Y,FLAG,N,ACC)
1F(FLAG) 32,32,34
32 DO 33 I=1,N
33 X(I)=Y(I)
FX=FY
RETURN
C *** FINAL SOLUTION NOT ACCURATE ENOUGH. REPLACE THE FIRST DIRECTION
C VECTOR BY INTER-SOLUTION VECTOR (NORMALISED) AND RECYCLE
34 A=A/STEP
DO 35 I=1,N
DIRECT(I,I)=(FIRST(I)-SECND(I))*A
35 FIRST(I)=SECND(I)
GO TO 3
END

SUBROUTINE TEST(FI,FF,RI,RF,FLAG,N,ACC)
C THIS SUBROUTINE IS PECULIAR TO THE POWELL METHOD OF DIRECT SEARCH
DIMENSION RI(10),RF(10)
FLAG=+2.
IFI(ABS(FI)-ACC) 2,2,1
1 IF(ABS((FI-FF)/FI)-ACC) 3,3,7
2 IF(ABS(FI-FF)-ACC) 3,3,7
3 DO 6 I=1,N
IFI(ABS(RI(I))-ACC) 5,5,4
4 IF(ABS((RI(I)-RF(I))/RI(I))-ACC) 6,6,7
5 IF(ABS(RI(I)-RF(I))-ACC) 6,6,7
6 CONTINUE
FLAG=-2.
7 RETURN
END

```

Б.3. МЕТОД ОПТИМИЗАЦИИ НЕЛДЕРА И МИДА. ИНСТРУКЦИИ ПО ВВОДУ ДАННЫХ В ПРОГРАММУ

Функцией, представленной в колоде перфократ в данной программе, является функция Вуда. Карта ($SUM (IN) = f(x)$) (третья последняя карта в колоде) должна заменяться при каждом новом выборе функции. Максимальная размерность задачи может быть 50, т. е. от $X(1)$ до $X(50)$.

Пользователь должен подготовить следующие перфокарты с данными:

Карта 1. Перфорируется NX — число переменных целевой функции в формате 15 в колонках 1—5, а также STEP — размер шага в формате F10.5 в колонках 6—16. При отсутствии другой информации выбирается $STEP =$

$$= \min \left\{ \frac{0.2}{n} \sum_{i=1}^n d_i, d_1, d_2, \dots, d_n \right\},$$

где n — число независимых переменных, d_i — область возможного поиска по переменной x_i .

Карта 2. Перфорируется исходное предполагаемое значение каждой переменной в формате F10.5. Карты 2 и 3 могут быть повторены с измененными размером шага и исход-

ными переменными в зависимости от задачи, но после последней карты типа 2 должна следовать:

Карта *m*. Пустая карта.

```

PROGRAM SIMPLEX(INPUT,OUTPUT)
C NX IS THE NUMBER OF INDEPENDENT VARIABLES.
C STEP IS THE INITIAL STEP SIZE.
C X(I) IS THE ARRAY OF INITIAL GUESSES.
C DATA CARDS ARE AS FOLLOWS.
C CARD NO.      PARAMETERS          FFORMAT        COLUMNS
C     1           NX                I5             1 THRU 5
C     1           STEP              F10.5          6 THRU 15
C     2           X(I)              F10.5          1 THRU 10
C CARD 3 IS BLANK.
C TO OPTIMIZE THE OBJECTIVE FUNCTION FOR ANOTHER SET OF PARAMETERS
C REPEAT CARDS 1 AND 2 ONLY.
C FOR PROPER PRINTOUT OF DESIRED X(I) ARRAY, FORMAT STATEMENTS 103
C AND 101 MUST BE REVISED ACCORDINGLY.
C DIMENSION X1(50,50), X(50), SUM(50)
COMMON/1/ X,X1,NX,STEP,K1,SUM,IN
1 FORMAT(15,F10.5)
100 READ1, NX,STEP
   IF(NX) 998,999,998
998 READ 2, (X(I),I=1,NX)
 2 FORMAT(10F10.5)
   ALFA=1.0
   BETA=0.5
   GAMMA=2.0
   DIFER = 0.
   XNX = NX
   IN = 1
   CALL SUMR
   PRINT 102,SUM(1),(X(I),I=1,NX)
   PRINT 1002+STEP
   CALL SECOND(TIME)
   PRINT 105+TIME
105 FORMAT(1/50X,11H TIME IS NOW,F10.3,8H SECONDS/)
   PRINT 103
103 FORMAT(14X,14H FUNCTION VALUE,15X,3HX1=.20X,3HX2=.20X,3HX3=.20X,3HX4
 1=.16X,12H FUNC. CHANGE)
102 FORMAT(1H1+12X,23H FUNCTION STARTING VALUE,F10.5,/,*THE X ARRAY IS*
 1,/,5X,1U(E11.4,2X))
1002 FORMAT(12X*STEP=*,F6.2)
   K1 = NX + 1
   K2 = NX + 2
   K3 = NX + 3
   K4 = NX + 4
   CALL START
25 DO 3 I = 1, K1
   DO 4 J = 1, NX
4 X(J) = X1(I,J)
   IN = 1
   CALL SUMR
3 CONTINUE
C SELECT LARGEST VALUE OF SUM(I) IN SIMPLEX
28 SUMH = SUM(1)
   INDEX = 1
   DO 7 I = 2, K1
   IF(SUM(I)>E.SUMH) GO TO 7
   SUMH = SUM(I)
   INDEX = I
7 CONTINUE

```

```

C SELECT MINIMUM VALUE OF SUM(I) IN SIMPLEX
    SUML = SUM(1)
    KOUNT = 1
    DO 8 I = 2, K1
    IF(SUML.LE.SUM(I)) GO TO 8
    SUML = SUM(I)
    KOUNT = I
8 CONTINUE
C FIND CENTROID OF POINTS WITH I DIFFERENT THAN INDEX
    DO 9 J = 1, NX
    SUM2 = 0.
    DO 10 I = 1, K1
10 SUM2 = SUM2 + X1(I,J)
    X1(K2,J) = 1./NX*(SUM2 - X1(INDEX,J))
C FIND REFLECTION OF HIGH POINT THROUGH CENTROID
    X1(K3,J) = (1. + ALFA)*X1(K2,J) - ALFA*X1(INDEX,J)
9 X(J) = X1(K3,J)
    IN = K3
    CALL SUMR
    IF(SUM(K3).LT.SUML) GO TO 11
C SELECT SECOND LARGEST VALUE IN SIMPLEX
    IF(INDEX.EQ.1) GO TO 38
    SUMS = SUM(1)
    GO TO 39
38 SUMS = SUM(2)
39 DO 12 I = 1, K1
    IF((INDEX - I).EQ.0) GO TO 12
    IF(SUM(I).LE.SUMS) GO TO 12
    SUMS = SUM(I)
12 CONTINUE
    IF(SUM(K3).GT.SUMS) GO TO 13
    GO TO 14
C FORM EXPANSION OF NEW MINIMUM IF REFLECTION HAS PRODUCED ONE MINIMUM
11 DO 15 J = 1, NX
    X1(K4,J) = (1 - GAMA)*X1(K2,J) + GAMA*X1(K3,J)
15 X(J) = X1(K4,J)
    IN = K4
    CALL SUMR
    IF(SUM(K4).LT.SUML) GO TO 16
    GO TO 14
13 IF(SUM(K3).GT.SUMH) GO TO 17
    DO 18 J = 1, NX
18 X1(INDEX,J) = X1(K3,J)
17 DO 19 J = 1, NX
    X1(K4,J) = BETA*X1(INDEX,J) + (1 - BETA)*X1(K2,J)
19 X(J) = X1(K4,J)
    IN = K4
    CALL SUMR
    IF(SUMH.GT.SUM(K4)) GO TO 16
C REDUCE SIMPLEX BY HALF IF REFLECTION HAPPENS TO PRODUCE A LARGER VAL
C LUE THAN THE MAXIMUM
    DO 20 J = 1, NX
    DO 20 I = 1, K1
20 X1(I,J) = 0.5*(X1(I,J) + X1(KOUNT,J))

    DO 29 I = 1, K1
    DO 30 J = 1, NX
30 X(J) = X1(I,J)
    IN = I
    CALL SUMR
29 CONTINUE
    GO TO 26
16 DO 21 J = 1, NX
    X1(INDEX,J) = X1(K4,J)
21 X(J) = X1(INDEX,J)
    IN = INDEX
    CALL SUMR
    GO TO 26
14 DO 22 J = 1, NX

```

```

X1(INDEX,J) = X1(K3,J)
22 X(J) = X1(INDEX,J)
IN = INDEX
CALL SUMR
26 DO 23 J = 1, NX
23 X(J) = X1(K2,J)
IN = K2
CALL SUMR
DIFER = 0.
DO 24 I = 1, K1
24 DIFER = DIFER + (SUM(I) - SUM(K2))**2
DIFER = 1./XNX*SQRT(DIFER)
PRINT 101, SUML, (X1(KOUNT,J), J = 1,NX), DIFER
101 FORMAT(2(2X,E16.6),3(7X,E16.6),12X,E16.6)
IF( DIFER.GE.0.0000001) GO TO 28
CALL SECOND(TIME)
PRINT 105, TIME
GO TO 100
999 CONTINUE
END
SUBROUTINE START
DIMENSION A(50,50), X1(50,50), X(50), SUM(50)
COMMON/1/ X,X1,NX,STEP,K1,SUM,IN
VN = NX
STEP1 = STEP/(VN*SQRT(2.))*SQRT(VN + 1.) + VN - 1.
STEP2= STEP/(VN*SQRT(2.))*(SQRT(VN + 1.) - 1.)
DO 1 J = 1, NX
1 A(1,J) = 0.
DO 2 I = 2, K1
DO 2 J = 1, NX
A(I,J) = STEP2
L = I - 1
A(I,L) = STEP1
2 CONTINUE
DO 3 I = 1, K1
DO 3 J = 1, NX
3 X1(I,J) = X(J) + A(I,J)
RETURN
END
SUBROUTINE SUMR
COMMON/1/ X,X1,NX,STEP,K1,SUM,IN
DIMENSION X1(50,50), X(50), SUM(50)
SUM(IN)=(X(1)+10.*X(2))**2+5.*((X(3)-X(4))**2+(X(2)-2.*X(3))**4+
110.*((X(1)-X(4))**4
RETURN
END

```

4 0.5
3.0 -1.0 0.0 1.0

Б.4. ПРОГРАММА ФЛЕКСИПЛЕКС (МЕТОД СКОЛЬЗЯЩЕГО ДОПУСКА)

1. Назначение.

Программа Флексиплекс решает общую задачу нелинейного программирования:

минимизировать $y = f(x)$, $x \in E^n$,
при ограничениях

$$\begin{aligned} h_i(x) &= 0, \quad i = 1, \dots, m, \\ g_i(x) &\geq 0, \quad i = m + 1, \dots, p, \end{aligned} \tag{Б.4.1}$$

где $f(x)$ является целевой функцией, подлежащей минимизации (или максимизации); $x = (x_1, x_2, \dots, x_n)^T$ — вектор-столбец, элементы которого представляют собой n переменных рассмат-

риваемой задачи в n -мерном пространстве; $h_i(x) = 0$, $i = 1, \dots, m$, — ограничения в виде равенств; $g_i(x) \geq 0$, $i = m + 1, \dots, p$, — ограничения в виде неравенств. Функции $f(x)$, $h_i(x)$, $g_i(x)$ могут быть линейными и (или) нелинейными; m и (или) $p - m$ могут быть равны нулю. Таким образом, для $m = 0$ и $p - m = 0$ оптимизация осуществляется при отсутствии ограничений¹⁾.

2. Ввод задачи в программу Флексиплекс

Целевая функция и ограничения задачи (Б.4.1) вводятся в вычислительную машину посредством подпрограммы SUBROUTINE PROBLEM (INQ). Параметр INQ идентифицирует целевую функцию и служит характеристикой фигурирующей в задаче совокупности ограничений. INQ = 1 соответствует ограничениям в виде равенств; INQ = 2 соответствует ограничениям в виде неравенств. На протяжении всей программы каждое из ограничений в виде равенств и неравенств, а также целевая функция идентифицируются присоединенной переменной

$$R(I), \quad I = 1, \dots, m, \quad m + 1, \dots, p, \quad p + 1.$$

Подпрограмма SOBROUTINE PROBLEM (INQ) организуется следующим образом:

1. После содержащей комментарий карты «Equality Constraints» (ограничения в виде равенств) и оператора 1 вводятся ограничения в виде равенств (если таковые имеются) в следующей форме:

$$R(1) = h_1(x),$$

.

$$R(m) = h_m(x).$$

2. После содержащей комментарий карты «Inequality Constraints» (ограничения в виде неравенств) и оператора 2 необходимо представить ограничения в виде неравенств как

$$R(m + 1) = g_{m+1}(x),$$

.

$$R(p) = g_p(x).$$

3. После содержащей комментарий карты «Objective Function» (целевая функция) и оператора 3 необходимо представить целевую функцию как $R(p + 1) = f(x)$.

Если в формулировке задачи равенства отсутствуют ($m = 0$), то не нужно вводить никаких данных после оператора 1 и $R(m + 1)$ обратится в $R(1)$. Аналогично если имеются только равенства и нет неравенств, то следует опустить все данные после оператора 2 и $R(p + 1)$ обратится в $R(m + 1)$. В случае задачи без ограничений $R(p + 1) = R(1)$.

¹⁾ Более подробно рассматриваемая здесь методика решения задачи (Б.4.1) описана Павиани (Paviani D. A., A New Method for the Solution of the General Non-linear Programming Problem, Ph. D. Dissertation, The Univ. of Texas, Austin, Tex., May 1969).

Например, задача

$$\text{минимизировать } f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2$$

при ограничениях

$$h_1(\mathbf{x}): x_1^2 + x_2^2 + x_3^2 - 25 = 0,$$

$$h_2(\mathbf{x}): 8x_1 + 14x_2 + 7x_3 - 56 = 0,$$

$$g_i(\mathbf{x}): x_i \geq 0, \quad i = 1, \dots, 3,$$

должна быть представлена с использованием подпрограммы PROBLEM (INQ) следующим образом:

C EQUALITY CONSTRAINTS

1 CONTINUE

$$R(1) = X(1)**2 + X(2)**2 + X(3)**2 - 25.$$

$$R(2) = 8.*X(1) + 14.*X(2) + 7.*X(3) - 56.$$

Go to 5

C INEQUALITY CONSTRAINTS

2 CONTINUE

$$R(3) = X(1)$$

$$R(4) = X(2)$$

$$R(5) = X(3)$$

Go to 5

C OBJECTIVE FUNCTION

3 CONTINUE

$$R(6) = 1000. - X(1)**2 - 2.*X(2)**2 - X(3)**2 - X(1)*X(2)$$

5 Return

3. Представление данных

Первая карта с данными

Эта карта является ключевой и должна содержать данные, идентифицирующие решаемую задачу. Любые алфавитно-цифровые операторы могут располагаться в колонках 1—80.

Вторая карта с данными

Эта карта содержит параметры задачи, такие, как NX — общее число переменных (зависимые + независимые) в формате 15 слева направо;

NC — общее число ограничений в виде равенств (m) в формате 15 слева направо;

NIC — общее число ограничений в виде неравенств ($p = m$) в формате 15 слева направо. Следует заметить, что верхний и ниж-

ний пределы для вектора x представляют также ограничения в виде неравенств;

$\text{SIZE} = t$, величина, определяющая размер деформируемого многогранника в исходной фазе поиска (используется формат F10.5). Объяснение того, как следует выбирать значения SIZE , приводится ниже.

$\text{CONVER} = \epsilon$, произвольно выбранное положительное малое число, используемое для окончания поиска; число ϵ рассматривается в качестве индикатора сходимости и обычно выбирается равным 10^{-5} или 10^{-6} .

Рекомендуемыми значениями SIZE являются следующие. Когда верхний и нижний пределы вектора x известны, то выбирается:

1. $\text{SIZE} \approx 20\%$ разности между верхним и нижним пределами x , если ожидаемые интервалы изменения x_i вдоль каждой оси координат приблизительно равны.

2. Если ожидаемые интервалы изменения x вдоль каждой оси координат различны, то SIZE присваивают значение, равное наименьшей разности между соответствующими верхним и нижним пределами любого x_i .

Примеры

(1)

$$\begin{aligned} -11 &\leq x_1 \leq 98,7 \\ -10 &\leq x_2 \leq 100,1 \\ -9,5 &\leq x_3 \leq 101 \\ -10,2 &\leq x_4 \leq 99,5 \\ -9,8 &\leq x_5 \leq 100 \end{aligned}$$

$$\text{SIZE} \approx 0,2 * 110 = 22$$

(2)

$$\begin{aligned} 0 &\leq x_1 \leq 400 \\ 0 &\leq x_2 \leq 1000 \\ 340 &\leq x_3 \leq 420 \\ 340 &\leq x_4 \leq 420 \\ -1000 &\leq x_5 \leq 1000 \\ 0 &\leq x_6 \leq 0,5236 \end{aligned}$$

$$\text{SIZE} \approx 1$$

```
PROGRAM FLEXI { INPUT, OUTPUT, TAPE10 = INPUT }
```

		60
*** * * * * PROGRAM FLEXIPLEX * * * * *		70
NX	TOTAL NUMBER OF INDEPENDENT VARIABLES	0110
NC	TOTAL NUMBER OF EQUALITY CONSTRAINTS	0120
NIC	TOTAL NUMBER OF INEQUALITY CONSTRAINTS	0130
SIZE	EDGE LENGTH OF THE INITIAL POLYHEDRON	0140
CONVER	CONVERGENCE CRITERION FOR TERMINATION OF THE SEARCH	0150
ALFA	THE REFLECTION COEFFICIENT	0160
BETA	THE CONTRACTION COEFFICIENT	0170
GAMA	THE EXPANSION COEFFICIENT	0180
X(I)	THE ASSUMED VECTOR TO INITIATE THE SEARCH	0190
FDFILER	THE TOLERANCE CRITERION FOR CONSTRAINT VIOLATION	0200
ICONT	A COUNTER TO RECORD STAGE COMPUTATIONS	0210
NCONT	A COUNTER TO PRINT INFORMATION EVERY (NX+1) STAGE	0220
LOW	AN INDEX TO IDENTIFY INFORMATION RELATED TO THE LOWEST VALUE OF OBJ. FUNCTION IN MOST RECENT POLYHEDRON	0230
LHIGH	AN INDEX TO IDENTIFY INFORMATION RELATED TO LARGEST VALUE OF OBJ. FUNCTION IN MOST RECENT POLYHEDRON ..	0240
LSEC	AN INDEX TO IDENTIFY INFORMATION RELATED TO THE SECOND LARGEST VALUE OF OBJ. FUNCTION IN MOST RECENT POLYHEDRON	0250
		0260
		0270
		0280
		290

```

*****  

DIMENSION X(50),X1(50,50),X2(50,50),R(100),SUM(50),F(50),SR(50),
1 ROLD(100), H(50)
COMMUN/1/NX,NC,NIC,STEP,ALFA,BETA,GAMA,IN,INF,FDIFER,SEQL,K1,K2,
1K3,K4,K5,K6,K7,K8,K9,X,X1,X2,R,SUM,F,SR,ROLD,SCALE,FOLD
COMMON/2/LFEAS,L5,L6,L7,L8,L9,R1A,R2A,R3A
PROBLEM IDENTIFICATION HEADER IS READ IN AFTER THIS CARD
READ 759
PARAMETERS FOR THE PROBLEM ARE READ IN AFTER THIS CARD
READ 1, NX,NC,NIC,SIZE,CONVER
ALFA = 1.
BETA = 0.5
GAMA = 2.
PERMANENT DATA FOR THE PROBLEM SHOULD BE READ IN AFTER THIS CARD
10 CALL SECONDIT(TIME)
TEMPORARY DATA FOR THE PROBLEM, SUCH AS VARIABLE COEFFICIENTS OR
NEW PARAMETERS SHOULD BE READ IN AFTER THIS CARD
STEP = SIZE
THE ASSUMED INITIAL VECTOR IS READ IN AFTER THIS CARD
READ 2, (X(I), I = 1, NX)
IF(EOP,10)9999,11
11 PRINT 106
PRINT 759
PRINT 756, NX,NC,NIC,SIZE,CONVER,TIME
K1 = NX + 1
K2 = NX + 2
K3 = NX + 3
K4 = NX + 4
K5 = NX + 5
K6 = NC + NIC
K7 = NC + 1
K8 = NC + NIC
K9 = K8 + 1
N = NX - NC
N1 = N + 1
IF(N1.GE.3) GO TO 50
N1 = 3
N = 2
50 N2 = N + 2
N3 = N + 3
N4 = N + 4
N5 = N + 5
N6 = N + 6
N7 = N + 7
N8 = N + 8
XN = N
XNX = NX
XN1 = N1
RIA = 0.5*(SORT(5.)* 1.
R2A = R1A*RIA
R3A = R2A*RIA
L5 = NX + 5
L6 = NX + 6
L7 = NX + 7
L8 = NX + 8
L9 = NX + 9
ICONT = 1
NCONT = 1
PRINT 115
PRINI 116, (X(J), J = 1, NX)
FDIFER = 2.*INC + 1)*STEP
FOLD = FDIFER
IN = N1
CALL SUMR
SR(N1) = SORT(SEQL)
PRINT 763, FDIFER, SR(N1)

```

0300
310
0320
0330
0340
0350
0360
0370
400
0410
0420
0430
440
0450
0460
0510
0520
0530
540
0550
0560
0570
0580
0590
0600
0610
0620
0630
0640
0650
0660
0670
0680
0690
0700
0710
0720
0730
0740
0750
0760
0770
0780
0800
.0810
0820
0830
0840
0850
0860
0870
0880
0890
0900
0910
0920
0930
0940
0950
0960
0970
0980
0990
1000
1010
1020
1030

```

IF(SR(N1).LT.FDIFER) GO TO 341          1040
CALL WRITEX                               1041
PRINT 757                                 1050
INF = N1                                  1060
STEP = 0.05*FDIFER                         1061
CALL FEASBL                                1070
PRINT 764                                 1080
PRINT 116, (X2(INF,J),J = 1, NX)           1090
PRINT 765, SR(INF)                          1100
IF(FOLD.LT.1.0E-09) GO TO 80               1110
341 PRINT 35                                1120
PRINT 758, ICONT, FDIFER                  1130
CALL WRITEX                               1140
FTER = R(K9)                             1150
C   COMPUTE CENTROID OF ALL VERTICES OF INITIAL POLYHEDRON    1160
237 STEP1 = STEP*(SQRT(XNX + 1.) + XNX - 1.)/(XNX*SQRT(2.))  1170
STEP2 = STEP*(SQRT(XNX + 1.) - 1.)/(XNX*SQRT(2.))           1180
ETA = (STEP1 + (XNX - 1.)*STEP2)/(XNX + 1.)                 1190
DO 4 J = 1, NX                           1200
X(J) = X(J) - ETA                        1210
4 CONTINUE                                1220
CALL START.                               1230
DO 9 I = 1, N1                           1240
DO 9 J = 1, NX                           1250
X2(I,J) = X1(I,J)                        1260
9 CONTINUE                                1270
DO 5 I = 1, N1                           1280
IN = I                                     1290
DO 6 J = 1, NX                           1300
X(J) = X2(I,J)                           1310
CALL SUMR                                1320
SR(I) = SQRT(SEQL)                      1330
IF(SR(I).LT.FDIFER) GO TO 8              1340
CALL FEASBL                                1350
IF(FOLD.LT.1.0E-09) GO TO 80             1360
8 CALL PROBLEM(3)                         1370
F(I) = R(K9)                            1380
5 CONTINUE                                1390
1000 STEP = 0.05*FDIFER                  1400
ICONT = ICONT + 1                         1410
C   SELECT LARGEST VALUE OF OBJECTIVE FUNCTION FROM POLYHEDRON VERTICES 1420
FH = F(1)                                 1430
LHIGH = 1                                 1440
DO 16 I = 2, N1                           1450
IF(F(I).LT.FH) GO TO 16                  1460
FH = F(I)                                1470
LHIGH = I                                1480
16 CONTINUE                                1490
C   SELECT MINIMUM VALUE OF OBJECTIVE FUMCTION FROM POLYHEDRON VERTICES 1500
41 FL = F(1)                                1510
LOW = 1                                    1520
DO 17 I = 2, N1                           1530
IF(FL.LT.F(I)) GO TO 17                  1540
FL = F(I)                                1550
LOW = I                                    1560
17 CONTINUE                                1570
DO 86 J = 1, NX                           1580
86 X(J) = X2(LOW,J)                       1590
IN = LOW                                 1600
CALL SUMR                                1610
SR(LOW) = SQRT(SEQL)                     1620
IF(SR(LOW).LT.FDIFER) GO TO 87            1630
INF = LOW                                1640
CALL FEASBL                                1650
IF(FOLD.LT.1.0E-09) GO TO 80             1660
CALL PROBLEM(3)                         1670
F(LOW) = R(K9)                           1680
GO TO 41                                1690

```

```

C 87 CONTINUE          1700
  FIND CENTROID OF POINTS WITH I DIFFERENT THAN LHIGH
  DO 19 J = 1, NX      1710
    SUM2 = 0.            1730
    DO 20 I = 1, N1      1740
      SUM2 = SUM2 + X2(I,J)
  19 X2(N2,J) = 1./XN*(SUM2-X2(LHIGH,J))           1750
    SUM2 = 0.
    DO 36 I = 1, N1      1760
      DO 36 J = 1, NX      1770
        SUM2 = SUM2 + (X2(I,J) - X2(N2,J))*2
  36 CONTINUE          1780
  FDIFER = (INC + 1)/XN1*SQRT(SUM2)                 1790
  IF(FDIFER.LT.FOLD) GO TO 98
  FDIFER = FOLD
  GO TO 198
  98 FOLD = FDIFER
  198 CONTINUE          1800
    FTER = F(LOW)
  137 NCNT = NCNT + 1          1810
    IF(NCNT.LT.4*N1) GO TO 37
    IF(NCNT.LT.1500) GO TO 337
    FOLD = .05*FOLD          1820
  337 NCNT = 0          1830
    PRINT 35          1840
    PRINT 758, NCNT, FDIFER
    CALL WRITEX          1850
  37 IF(FDIFER.LT.CONVER) GO TO 81          1860
C SELECT SECOND LARGEST VALUE OF OBJECTIVE FUNCTION
  IF(LHIGH.EQ.1) GO TO 43          1870
  FS = F(1)          1880
  LSEC = 1          1890
  GO TO 44          1900
  43 FS = F(2)          1901
  LSEC = 2          1902
  44 DO 18 I = 1, N1          1910
    IF(LHIGH.EQ.I) GO TO 18
    IF(F(I).LT.FS) GO TO 18
    FS = F(1)
    LSEC = 1
  18 CONTINUE          1920
C REFLECT HIGH POINT THROUGH CENTROID          1930
  DO 61 J = 1, NX          1940
    X2(N3,J) = X2(N2,J) + ALFA*(X2(N2,J) - X2(LHIGH,J))
  61 X(J) = X2(N3,J)          1950
    IN = N3          1960
    CALL SUMR          1970
    SR(N3) = SQRT(SEQL)
  89 IF(SR(N3).LT.FDIFER) GO TO 82          1980
    INF = N3          1990
    CALL FEASBL          2000
    IF(FOLD.LT.1.0E-09) GO TO 80          2010
  82 CALL PROBLEM(3)          2020
    F(N3) = R(K9)
    IF(F(N3).LT.F(LOW)) GO TO 84
    IF(F(N3).LT.F(LSEC)) GO TO 92
    GO TO 60          2030
  92 DO 93 J = 1, NX          2040
  93 X2(LHIGH,J) = X2(N3,J)
    SR(LHIGH) = SR(N3)
    F(LHIGH) = F(N3)
    GO TO 1000          2050
C EXPAND VECTOR OF SEARCH ALONG DIRECTION THROUGH CENTROID AND
C REFLECTED VECTOR          2060
  84 DO 23 J = 1, NX          2070
    X2(N4,J) = X2(N3,J) + GAMMA*(X2(N3,J) - X2(N2,J))
  23 X(J) = X2(N4,J)          2080
    IN = N4          2090

```

```

CALL SUMR          2360
SR(N4) = SORT(SEQ1)
IF(SR(N4).LT.FDIFER) GO TO 25 2370
INF = N4          2380
CALL FEASBL       2390
IF(FOLD.LT.1.0E-09) GO TO 80 2400
25 CALL PROBLEM(3) 2410
F(N4) = R(K9)      2420
IF(IF(LLOW).LT.F(N4)) GO TO 92 2430
DO 26 J = 1, NX    2440
26 X2(LHIGH,J) = X2(N4,J) 2450
F(LHIGH) = F(N4)      2460
SR(LHIGH) = SR(N4)    2470
GO TO 1000        2480
60 IF(F(N3).GT.F(LHIGH)) GO TO 64 2490
DO 65 J = 1, NX    2500
65 X2(LHIGH,J) = X2(N3,J) 2510
66 DO 66 J = 1, NX    2520
66 X2(N4,J) = BETA*X2(LHIGH,J) + (1. - BETA)*X2(N2,J) 2530
66 X(J) = X2(N4,J) 2540
IN = N4          2550
CALL SUMR          2560
SR(N4) = SQRT(SEQ1) 2570
IF(SR(N4).LT.FDIFER) GO TO 67 2580
INF = N4          2590
CALL FEASBL       2600
IF(FOLD.LT.1.0E-09) GO TO 80 2610
67 CALL PROBLEM(3) 2620
F(N4) = R(K9)      2630
IF(F(LHIGH).GT.F(N4)) GO TO 68 2640
DO 69 J = 1, NX    2650
DO 69 I = 1, N1    2660
69 X2(I,J) = 0.5*(X2(I,J) + X2(LOW,J)) 2670
DO 70 I = 1, N1    2680
DO 71 J = 1, NX    2690
71 X(J) = X2(I,J) 2700
IN = I            2710
CALL SUMR          2720
SR(I) = SQRT(SEQ1) 2730
IF(SR(I).LT.FDIFER) GO TO 72 2740
INF = 1            2750
CALL FEASBL       2760
IF(FOLD.LT.1.0E-09) GO TO 80 2770
72 CALL PROBLEM(3) 2780
70 F(I) = R(K9)      2790
GO TO 1000        2800
68 DO 73 J = 1, NX    2820
73 X2(LHIGH,J) = X2(N4,J) 2830
SR(LHIGH) = SR(N4)    2840
F(LHIGH) = F(N4)      2850
GO TO 1000        2860
81 PRINT 760,ICONT, FDIFER 2870
CALL WRITEX       2880
CALL SECOND(TIME) 2890
PRINT 755, TIME    2900
PRINT 761          2910
GO TO 10          2920
80 PRINT 760, ICONT, FDIFER 2930
CALL WRITEX       2931
PRINT 762          2940
GO TO 10          2941
1 FORMAT(3I5,F10.5,E10.3) 2950
2 FORMAT(8F10.5)      2960
35 FORMAT(//,40X,48H * * * * * * * * * * * * * * * * * * * * * * * * * * * * * ) 2970
106 FORMAT(1H1,//)     2980
115 FORMAT(//, 41H THE STARTING VECTOR SELECTED BY USER IS ) 2990
116 FORMAT(BE16.6)     3000
755 FORMAT(//, 35H THE COMPUTATION TIME IN SECONDS = E12.5) 3010

```

```

756 FORMAT(//,10X,40H NUMBER OF INDEPENDENT VARIABLES      15.,/,10X
1,40H NUMBER OF EQUALITY CONSTRAINTS      15.,/,10X,40H NUMBER O 3020
2F INEQUALITY CONSTRAINTS      15.,/,10X,40H SIZE OF INITIAL POLY 3030
3HEDRUN      E12.5.,/,10X,40H THE DESIRED CONVERGENCE IS 3040
4          E12.5.,/,10X,40H THE COMPUTATION TIME IN SECONDS 3050
5E12.5)      3060
757 FORMAT(//,71H THE INITIAL X VECTOR DOES NOT SATISFY THE INITIAL TO 3070
TOLERANCE CRITERION ) 3080
758 FORMAT( /,10X,27H STAGE CALCULATION NUMBER = 15, 20X, 27H THE TOLERANCE 3090
CRITERION = E14.6) 3100
759 FORMAT(80H,
1          )
760 FORMAT(//, 39H TOTAL NUMBER OF STAGES CALCULATIONS = 15, 10X, 25H 3110
1THE CONVERGENCE LIMIT = E14.6) 3120
761 FORMAT(//,50X,25H THESE ARE FINAL ANSWERS ) 3130
762 FORMAT(//,50X,29H THESE ARE NOT FINAL ANSWERS ) 3140
763 FORMAT(//,10X,40H THE INITIAL TOLERANCE CRITERION IS      E12.5.,/
110X,40H THE SUM OF VIOLATED CONSTRAINTS IS      E12.5) 3150
764 FORMAT(//,70H THE VECTOR FOUND BY PROGRAM WHICH SATISFIES THE INIT 3160
IAL TOLERANCE IS ) 3170
765 FORMAT(//, 31H SUM OF VIOLATED CONSTRAINTS = E17.7) 3180
9999 STOP 3190
END 3200
SUBROUTINE FEASBL 3210
C
C*****SUBROUTINE FEASBL MINIMIZES THE SUM OF THE SQUARE VALUES OF THE 3220
VIOLATED CONSTRAINTS. IT IS CALLED EVERY TIME THE COMBINED VALUE 3230
OF THE VIOLATED CONSTRAINTS EXCEEDS THE THE VALUE OF THE TOLERANCE 3240
CRITERION FOR THE CURRENT STAGE 3250
C
C      DIMENSION X(50),X1(50,50),X2(50,50),R(100),SUM(50),F(50),SR(50),
IROLD(100), R1(100), R2(100),R3(100), FLG(10), H(50) 3260
100 FORMAT(8E16.6) 3270
COMMON/1/NX,NC,NIC,STEP,DUM1,DUM2,DUM3,IN,INF,FDIFER,SEQL,K$,K2,
1K3,K4,K5,K6,K7,K8,K9,X,X1,X2,R,RSUM,F,SR,ROLD,SCALE,FOLD 3280
COMMON/2/LFEAS,L5,L6,L7,L8,L9,R1A,R2A,R3A 3290
ALFA = 1. 3300
BETA = 0.5 3310
GAMA = 2. 3320
NXN = NX 3330
ICUNT = 0 3340
LCHEK = 0 3350
ICHEK = 0 3360
25 CALL START 3370
DO 3 I = 1, K1 3380
DO 4 J = 1, NX 3390
4 X(J) = X1(I,J) 3400
IN = I 3410
CALL SUMR 3420
3 CONTINUE 3430
C SELECT LARGEST VALUE OF SUM(I) IN SIMPLEX 3440
28 SUMH = SUM(1) 3450
INDEX = 1 3460
DO 7 I = 2, K1 3470
IF(SUM(I).LE.SUMH) GO TO 7 3480
SUMH = SUM(I) 3490
INDEX = I. 3500
7 CONTINUE 3510
C SELECT MINIMUM VALUE OF SUM(I) IN SIMPLEX 3520
SUML = SUM(1) 3530
KOUNT = 1 3540
DO 8 I = 2, K1 3550
IF(SUML.LE.SUM(I)) GO TO 8 3560
SUML = SUM(I) 3570
KOUNT = I 3580
8 CONTINUE 3590
C FIND CENTROID OF POINTS WITH I DIFFERENT THAN INDEX 3600
DO 9 J = 1, NX 3610

```

```

SUM2 = 0
DO 10 I = 1, K1
 10 SUM2 = SUM2 + X1(I,J)
X1(K2,J) = 1.0/XNX*(SUM2 - X1(INDEX,J))
C FIND REFLECTION OF HIGH POINT THROUGH CENTROID
X1(K3,J) = 2.*X1(K2,J) - X1(INDEX,J)
 9 X(J) = X1(K3,J)
  IN = K3
  CALL SUMR
  IF(SUM(K3).LT.SUML) GO TO 11
C SELECT SECOND LARGEST VALUE IN SIMPLEX
  IF(INDEX.EQ.1) GO TO 38
  SUMS = SUM(1)
  GO TO 39
38 SUMS = SUM(2)
39 DO 12 I = 1, K1
  IF((INDEX - 1).EQ.0) GO TO 12
  IF(SUM(I).LE.SUMS) GO TO 12
  SUMS = SUM(I)
12 CONTINUE
  IF(SUM(K3).GT.SUMS) GO TO 13
  GO TO 14
C FORM EXPANSION OF NEW MINIMUM IF REFLECTION HAS PRODUCED ONE MINIMUM
11 DO 15 J = 1, NX
  X1(K4,J) = X1(K2,J) + 2.0*(X1(K3,J) - X1(K2,J))
15 X(J) = X1(K4,J)
  IN = K4
  CALL SUMR
  IF(SUM(K4).LT.SUML) GO TO 16
  GO TO 14
13 IF(SUM(K3).GT.SUMH) GO TO 17
  DO 18 J = 1, NX
18 X1(INDEX,J) = X1(K3,J)
17 DO 19 J = 1, NX
  X1(K4,J) = 0.5*X1(INDEX,J) + 0.5*X1(K2,J)
19 X(J) = X1(K4,J)
  IN = K4
  CALL SUMR
  IF(SUMH.GT.SUM(K4)) GO TO 6
C REDUCE SIMPLEX BY HALF IF REFLECTION HAPPENS TO PRODUCE A LARGER VAL
C LUE THAN THE MAXIMUM
  DO 20 J = 1, NX
  DO 20 I = 1, K1
20 X1(I,J) = 0.5*(X1(I,J) + X1(KOUNT,J))
  DO 29 I = 1, K1
  DO 30 J = 1, NX
30 X(J) = X1(I,J)
  IN = 1
  CALL SUMR
29 CONTINUE
  5 SUML = SUM(1)
  KOUNT = 1
  DO 23 I = 2, K1
  IF(SUML.LT.SUM(1)) GO TO 23
  SUML = SUM(1)
  KOUNT = I
23 CONTINUE
  SR(INF) = SQRT(SUM(KOUNT))
  DO 27 J = 1, NX
27 X(J) = X1(KOUNT,J)
  GO TO 26
  6 DO 31 J = 1, NX
31 X1(INDEX,J) = X1(K4,J)
  SUM(INDEX) = SUM(K4)
  GO TO 5
16 DO 21 J = 1, NX
  X1(INDEX,J) = X1(K4,J)
21 X(J) = X1(INDEX,J)

```

```

SUM(INDEX) = SUM(K4) 4370
SR(INF) = SQRT(SUM(K4)) 4380
GO TO 26 4390
14 DO 22 J = 1, NX 4400
X1(INDEX,J) = X1(K3,J) 4410
22 X(J) = X1(INDEX,J) 4420
SUM(INDEX) = SUM(K3) 4430
SR(INF) = SQRT(SUM(K3)) 4440
26 ICONT = ICONT + 1 4450
DO 36 J = 1,NX 4460
36 X2(INF,J) = X(J) 4470
IF(ICONT.LT.2*K1) GO TO 50 4480
ICONT = 0 4490
DO 24 J = 1, NX 4500
24 X(J) = X1(K2,J) 4510
IN = K2 4520
CALL SUMR 4530
DIFER = 0. 4540
DO 57 I = 1, K1 4550
57 DIFER = DIFER + (SUM(I) - SUM(K2))**2 4560
DIFER = 1./(K7*XNX)*SQRT(DIFER) 4570
IF(DIFER.GT.1.0E-14) GO TO 50 4580
C IF FLEXIBLE SIMPLEX METHOD FAILED TO SATISFY THE CONSTRAINTS WITHIN 4590
C THE TOLERANCE CRITERION FOR THE CURRENT STAGE, THE SEARCH IS 4600
C PERTURBED FROM THE POSITION WHERE THE X VECTOR IS STUCK AND THEN 4610
C FEASBL IS REPEATED ONCE MORE FROM THE BEGINNING 4620
51 IN = K1 4630
STEP = 20.*FDIFER 4640
CALL SUMR 4650
SR(INF) = SQRT(SEQL) 4660
DO 52 J = 1, NX 4670
52 X1(K1,J) = X(J) 4680
DO 53 J = 1, NX 4690
FACTOR = 1. 4700
X(J) = X1(K1,J) + FACTOR*STEP 4710
X1(L9,J) = X(J) 4720
IN = L9 4730
CALL SUMR 4740
X(J) = X1(K1,J) - FACTOR*STEP 4750
X1(L5,J) = X(J) 4760
IN = L5 4770
CALL SUMR 4780
56 IF(SUM(L9).LT.SUM(K1)) GU TO 54 4790
IF(SUM(L5).LT.SUM(K1)) GU TO 55 4800
GO TO 97 4810
54 X1(L5,J) = X1(K1,J) 4820
SUM(L5) = SUM(K1) 4830
X1(K1,J) = X1(L9,J) 4840
SUM(K1) = SUM(L9) 4850
FACTOR = FACTOR + 1. 4860
X(J) = X1(K1,J) + FACTOR*STEP 4870
IN = L9 4880
CALL SUMR 4890
GO TO 56 4900
55 X1(L9,J) = X1(K1,J) 4910
SUM(L9) = SUM(K1) 4920
X1(K1,J) = X1(L5,J) 4930
SUM(K1) = SUM(L5) 4940
FACTOR = FACTOR + 1. 4950
X(J) = X1(K1,J) - FACTOR*STEP 4960
IN = L5 4970
CALL SUMR 4980
GO TO 56 4990
C ONE DIMENSIONAL SEARCH BY GOLDEN SECTION ALONG EACH COORDINATE 5000
97 H(J) = X1(L9,J) - X1(L5,J) 5010
X1(L6,J) = X1(L5,J) + H(J)*R1A 5020
X(J) = X1(L6,J) 5030
IN = L6 5040

```

```

CALL SUMR
X1(L7,J) = X1(L5,J) + H(J)*R2A
X(J) = X1(L7,J)
IN = L7
CALL SUMR
IF(SUM(L6).GT.SUM(L7)) GO TO 68
X1(L8,J) = X1(L5,J) + (1. - R3A)*H(J)
X1(L5,J) = X1(L7,J)
X(J) = X1(L8,J)
IN = L8
CALL SUMR
IF(SUM(L8).GT.SUM(L6)) GO TO 76
X1(L5,J) = X1(L6,J)
SUM(L5) = SUM(L6)
GO TO 75
76 X1(L9,J) = X1(L8,J)
SUM(L9) = SUM(L8)
GO TO 75
68 X1(L9,J) = X1(L6,J)
X1(L8,J) = X1(L5,J) + R3A*H(J)
X(J) = X1(L8,J)
IN = L8
CALL SUMR
STEP = SIZE
SUM(L9) = SUM(L6)
IF(SUM(L7).GT.SUM(L8)) GO TO 71
X1(L5,J) = X1(L8,J)
SUM(L5) = SUM(L8)
GO TO 75
71 X1(L4,J) = X1(L7,J)
SUM(L4) = SUM(L7)
75 IF(ABS(X1(L9,J) - X1(L8,J)).GT.0.01*FDIFER) GO TO 97
X1(K1,J) = X1(L7,J)
X(J) = X1(L7,J)
SUM(K1) = SUM(L5)
SR(INF) = SQRT(SUM(K1))
IF(SR(INF).LT.FDIFER) GO TO 760
53 CONTINUE
ICHEK = ICHEK + 1
STEP = FDIFER
IF(ICHEK.LE.2) GO TO 25
FOLD = 1.0E-12
PRINT 853
PRINT 850
PRINT 851, (X(J),J=1,NX)
PRINT 852, FDIFER, SR(INF)
GO TO 46
760 DO 761 J = 1, NX
X2(INF,J) = X1(K1,J)
761 X(J) = X1(K1,J)
50 IF(SR(INF).GT.FDIFER) GO TO 28
C MODIFIED LAGRANGE INTERPOLATION FOR TIGHT INEQUALITIES
IF(SR(INF).GT.0.) GO TO 35
CALL PROBLEM(3)
FINT = R(K9)
DO 139 J = 1, NX
139 X(J) = X2(INF,J)
CALL PROBLEM(2)
DO 40 J = K7,K8
40 R1(J) = R(J)
DO 41 J = 1, NX
41 X(J) = X1(KOUNT,J)
CALL PROBLEM(2)
DO 42 J = K7,K8
42 R3(J) = R(J)
DO 43 J = 1, NX
43 H(J) = X1(KOUNT,J) - X2(INF,J)
43 X(J) = X2(INF,J) + 0.5*H(J)
5050
5060
5070
5090
5100
5110
5120
5130
5140
5150
5160
5170
5180
5190
5200
5210
5220
5230
5240
5250
5260
5270
540
5280
5290
5300
5310
5320
5330
5340
5350
5360
5370
5380
5390
5400
5410
5420
5430
5440
5450
5460
5470
5480
5490
5500
5510
5520
5530
5540
5550
5560
5570
5580
5590
5600
5610
5620
5630
5640
5650
5660
5670
5680
5690
5700
5710

```



```

3 X1(I,J) = X(J) + A(I,J)          6400
  RETURN                                6410
  END                                    6420
  SUBROUTINE WRITEX                     6430
  DIMENSION X(50),X1(50,50),X2(50,50),R(100),SUM(50),F(50),SR(50),
1 ROLD(100)                                6440
  COMMON/1/NX,NC,NIC,STEP,ALFA,BETA,GAMA,IN,INF,FDIFER,SSEQ,L1,K2,
1K3,K4,K5,K6,K7,K8,K9,X,X1,X2,R,SUM,F,SR,ROLD,SCALE,FQLD      6450
  COMMON/2/LFEAS,L5,L6,L7,L8,L9,R1A,R2A,R3A                      6460
  CALL PROBLEM(3)                            6470
  CALL PROBLEM(3)                            6480
  CALL PROBLEM(3)                            6490
  PRINT 1, R(K9)                           6500
1 FORMAT(1, 28H OBJECTIVE FUNCTION VALUE = E17.7)    6510
  PRINT 2, (X(J), J = 1, NX)                6520
2 FORMAT(1, 29H THE INDEPENDENT VECTORS ARE /(6E17.7) 6530
  IF(NIC.EQ.0) GO TO 6                   6540
  CALL PROBLEM(1)                          6550
  PRINT 3, (R(J), J = 1, NC)               6560
3 FORMAT(1, 36H THE EQUALITY CONSTRAINT VALUES ARE /(6E17.7) 6570
  6 IF(NIC.EQ.0) GO TO 5                 6580
  CALL PROBLEM(2)                          6590
  PRINT 4, (R(J), J = K7,K6)              6600
4 FORMAT(1, 34H THE INEQUALITY CONSTRAINT VALUES /(6E17.7) 6610
  5 RETURN                                6620
  END                                    6630
  SUBROUTINE SUMR                         6640
C
C*****THIS SUBROUTINE COMPUTES THE SUM OF THE SQUARE VALUES OF THE   6650
C VIOLATED CONSTRAINTS IN ORDER TO BE COMPARED WITH THE TOLERANCE     6660
C CRITERION                           6670
C                                         6680
C                                         6690
C
  DIMENSION X(50),X1(50,50),X2(50,50),R(100),SUM(50),F(50),SR(50),
1 ROLD(100)                                6700
  COMMON/1/NX,NC,NIC,STEP,ALFA,BETA,GAMA,IN,INF,FDIFER,SSEQ,L1,K2,
1K3,K4,K5,K6,K7,K8,K9,X,X1,X2,R,SUM,F,SR,ROLD,SCALE,FQLD      6710
  COMMON/2/LFEAS,L5,L6,L7,L8,L9,R1A,R2A,R3A                      6720
  SUM(IN) = 0.                                6730
  CALL PROBLEM(2)                          6740
  SSEQ = 0.                                 6750
  IF(NIC.EQ.0) GO TO 4                   6760
  DO 1 J = K7, K8                         6770
  1 IF(R(J).GE.0.) GO TO 1                 6780
  SSEQ = SSEQ + R(J)*R(J)                  6790
  1 CONTINUE                               6800
  4 IF(NIC.EQ.0) GO TO 3                 6810
  CALL PROBLEM(1)                          6820
  DO 2 J = 1, NC                         6830
  2 SSEQ = SSEQ + R(J)*R(J)                  6840
  3 SUM(IN) = SSEQ                         6850
  5 RETURN                                6860
  END                                     6870
                                         6880
                                         6890

```

Третья и последующие карты

Если исходные данные задачи, такие, как константы, коэффициенты, значения $f(x)$ и т. п. должны быть введены в программу, их следует перфорировать на третьей и последующих картах. При этом можно использовать любое необходимое количество перфокарт в любом удобном формате. Эти данныечитываются в основную программу Флекси и переносятся в подпрограмму, которая вычисляет функции и сопутствующие выражения задачи {SUBROUTINE PROBLEM (INQ)} с помощью соответствующих общих операторов.

Операторы FORMAT, COMMON и READ для специфических данных должны обеспечиваться пользователем.

Оператор READ необходимо использовать для данных (констант, коэффициентов и т. п.), которые сохраняются для всех последующих вычислений. Этот оператор располагается в начале основной программы после содержащей комментарий карты «Permanent Data for the Problem...» (неизменяющиеся данные задачи...). Если таких данных нет, то пользователь должен изъять все операторы считывания и распечатки, следующие после комментирующей карты до оператора 10. Тогда с карт, следующих сразу после второй карты данных, будут считываться исходные предполагаемые значения переменных задачи, т. е. $x_i^{(0)}$, $i = 1, \dots, NX$. Исходные значения для $x^{(0)}$ располагаются после последней карты данных, описанной выше, т. е. после второй карты, если нет коэффициентов или констант, которые должны считываться. Исходные предполагаемые значения $x^{(0)}$ должны перфорироваться в формате 8F10.5.

Рассматриваемая программа может управлять более чем одним набором исходных предполагаемых значений переменных. Это означает, что после отыскания решения для первых исходных предполагаемых значений переменных основная программа осуществляет считывание дополнительного набора исходных предполагаемых значений переменных и задача решается еще раз. После завершения решения управление в программе передается оператору 10 для считывания следующего $x^{(0)}$ и соответствующего решения задачи. Если нужно изменить данные в задаче (новые коэффициенты или новые константы), то операторы считывания с соответствующими операторами по размещению и формату общих данных должны быть расположены после оператора 10. Программа заканчивается картой «End of File» (конец файла).

Приложение В

МАТРИЦЫ

•

Алгебра матриц широко используется всякий раз, когда приходится иметь дело с большим числом переменных, связанных линейными соотношениями. Знание некоторых условных обозначений и технических приемов матричного исчисления, а также представление о прикладных возможностях и границах применимости теории матриц необходимы как для освоения методики решения важного класса линейных задач, так и для понимания способов упрощения громоздких математических записей. Следует также отметить, что операции над матрицами могут с большой скоростью выполняться на любой цифровой вычислительной машине. Главное преимущество использования аппарата теории матриц заключается в том, что при этом удается избежать большого числа утомительных, хотя и имеющих стереотипный характер, элементов вычислительных процедур. Ниже приводятся некоторые наиболее существенные свойства матриц и рассматриваются основные операции над матрицами.

B.1. ОПРЕДЕЛЕНИЯ И ОБОЗНАЧЕНИЯ

Таблицу

$$\mathbf{a} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix},$$

содержащую определенным образом упорядоченные элементы a_{ik} , называют *матрицей*. Первый из индексов, которыми снабжен каждый элемент матрицы, обозначает номер строки, второй — номер столбца.

Квадратной называется матрица, в которой число строк равняется числу столбцов. Так, например,

$$\mathbf{a} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix}$$

представляет собой частный случай матрицы размерности $n \times n$ при $n = 3$.

Матрица, имеющая m строк и n столбцов, называется *прямоугольной*. Матрица размерности 1×1 есть скаляр.

Две матрицы считаются равными друг другу, когда каждый элемент первой матрицы равен занимающему соответствующее положение (которое определяется номером строки и номером столбца) элементу второй матрицы.

Матрица, состоящая из единственного столбца, называется *вектор-столбцом* (или просто *столбцом*), а матрица, состоящая из единственной строки, называется *вектор-строкой* (или просто *строкой*). В качестве иллюстрации приведем

$$\text{вектор-столбец} \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} \text{ и вектор-строку } [1 \ 2 \ 3 \ 4].$$

Диагональной называется квадратная матрица, в которой отличными от нуля могут быть только элементы, расположенные на главной диагонали.

Под *единичной матрицей* (обозначаемой, как правило, через I) понимается диагональная матрица с диагональными элементами, равными 1 (остальные элементы равны 0). Так, например,

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

есть единичная матрица размерности 3×3 .

Матрица, все элементы которой равны нулю, называется *нулевой матрицей*, обозначаемой через 0 .

Определитель (детерминант) матрицы a будем обозначать через $\det(a)$. Если $\det(a) \neq 0$, матрица a называется *невырожденной*; если $\det(a) = 0$, то говорят, что матрица a является *вырожденной*.

Если в матрице a поменять местами строки и столбцы, то получится матрица, *транспонированная* по отношению к a :

$$a^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{n1} \\ a_{12} & a_{22} & \dots & a_{n2} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{bmatrix}.$$

Так, например, если

$$a = \begin{bmatrix} 2 & 0 & -1 \\ 1 & 1 & 4 \end{bmatrix},$$

то

$$\mathbf{a}^T = \begin{bmatrix} 2 & 1 \\ 0 & 1 \\ -1 & 4 \end{bmatrix}.$$

Симметрической матрицей называется матрица, удовлетворяющая условию $\mathbf{a} = \mathbf{a}^T$. Например для матрицы

$$\mathbf{a} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}$$

$$\mathbf{a}^T = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix},$$

т. е. $\mathbf{a} = \mathbf{a}^T$, и, следовательно, мы имеем дело с симметрической матрицей.

B.2. ОПЕРАЦИИ НАД МАТРИЦАМИ

Сложение матриц. Сумма двух матриц $\mathbf{a} + \mathbf{b} = \mathbf{c}$, где \mathbf{a} , \mathbf{b} и \mathbf{c} — матрицы размерности $m \times n$, находится путем попарного сложения всех одинаково расположенных элементов \mathbf{a} и \mathbf{b} . Отметим, что сумма матриц разных размерностей не имеет смысла. Проиллюстрируем приведенное выше определение на конкретном примере:

$$\mathbf{c} = \mathbf{a} + \mathbf{b} = \begin{bmatrix} 2 & 0 \\ 6 & 3 \end{bmatrix} + \begin{bmatrix} 1 & -2 \\ 3 & 2 \end{bmatrix} = \begin{bmatrix} 3 & -2 \\ 9 & 5 \end{bmatrix}.$$

Умножение матрицы на скаляр. При умножении матрицы на скаляр получается матрица, в которой каждый элемент равен соответствующему элементу исходной матрицы, умноженному на данный скаляр. Пусть, например, $\alpha = 3$, а

$$\mathbf{a} = \begin{bmatrix} 3 & 4 & 1 \\ 2 & 6 & 2 \\ 1 & 0 & 1 \end{bmatrix}.$$

Тогда

$$\alpha \mathbf{a} = \begin{bmatrix} 3 \times 3 & 3 \times 4 & 3 \times 1 \\ 3 \times 2 & 3 \times 6 & 3 \times 2 \\ 3 \times 1 & 3 \times 0 & 3 \times 1 \end{bmatrix} = \begin{bmatrix} 9 & 12 & 3 \\ 6 & 18 & 6 \\ 3 & 0 & 3 \end{bmatrix}.$$

Из принятых выше определений следует, что

$$\mathbf{a} + (\mathbf{b} + \mathbf{c}) = (\mathbf{a} + \mathbf{b}) + \mathbf{c},$$

$$\mathbf{a} + \mathbf{b} = \mathbf{b} + \mathbf{a},$$

$$\mathbf{a} + \mathbf{0} = \mathbf{a}, \\ \alpha(\mathbf{a} + \mathbf{b}) = \alpha\mathbf{a} + \alpha\mathbf{b}.$$

Произведение двух матриц. Умножение (слева) матрицы \mathbf{a} на матрицу \mathbf{b} возможно только в том случае, когда эти матрицы удовлетворяют следующему требованию: число столбцов матрицы \mathbf{a} должно равняться числу строк матрицы \mathbf{b} (\mathbf{a} называется *левым множителем*, а \mathbf{b} — *правым множителем*). Заметим, что \mathbf{ab} , вообще говоря, не равняется \mathbf{ba} , хотя в некоторых частных случаях равенство $\mathbf{ab} = \mathbf{ba}$ может иметь место. Для того чтобы умножить матрицу \mathbf{a} на матрицу \mathbf{b} , необходимо проделать следующее: взять первый слева элемент первой строки матрицы \mathbf{a} и умножить его на первый сверху элемент первого столбца матрицы \mathbf{b} , затем взять второй слева элемент первой строки матрицы \mathbf{a} и умножить его на второй сверху элемент первого столбца матрицы \mathbf{b} и т. д., пока не будут попарно перемножены все элементы первой строки матрицы \mathbf{a} на соответствующие элементы первого столбца матрицы \mathbf{b} ; после этого все найденные произведения суммируются, в результате чего определяется элемент c_{11} матрицы $\mathbf{c} = \mathbf{ab}$. Таким образом,

$$c_{11} = \sum_{j=1}^n a_{1j} b_{j1}.$$

Аналогично определяются остальные элементы матрицы \mathbf{c} , т. е. элемент, расположенный в i -й строке и k -м столбце матрицы $\mathbf{c} = \mathbf{ab}$, определяется следующим соотношением:

$$c_{ik} = \sum_{j=1}^n a_{ij} b_{jk}.$$

Проиллюстрируем сформулированное выше правило на примере умножения матрицы

$$\mathbf{a} = \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

на матрицу

$$\mathbf{b} = \begin{bmatrix} 0 & 1 & 3 \\ 2 & 1 & 0 \\ 3 & 2 & 1 \end{bmatrix}.$$

Произведение равно

$$\mathbf{ab} = \begin{bmatrix} (0+0+6) & (1+0+4) & (3+0+2) \\ (0+2+3) & (2+1+2) & (6+0+1) \\ (0+2+6) & (0+1+4) & (0+0+2) \end{bmatrix} = \begin{bmatrix} 6 & 5 & 5 \\ 5 & 5 & 7 \\ 8 & 5 & 2 \end{bmatrix}.$$

Определив произведение двух матриц, можно рассмотреть ряд дополнительных свойств, которыми обладают матрицы.

Матрица, транспонированная по отношению к произведению двух матриц. Матрица, транспонированная по отношению к произведению двух матриц, равняется произведению матриц, транспонированных по отношению к исходным матрицам, взятым в обратном порядке, т. е.

$$(ab)^T = b^T a^T.$$

Пусть, например,

$$\mathbf{a} = \begin{bmatrix} 3 & 4 \\ 1 & 5 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 0 & 2 \\ 4 & 3 \end{bmatrix}.$$

Тогда

$$\mathbf{ab} = \begin{bmatrix} 16 & 18 \\ 20 & 17 \end{bmatrix} \text{ и } (ab)^T = \begin{bmatrix} 16 & 20 \\ 18 & 17 \end{bmatrix} = \begin{bmatrix} 0 & 4 \\ 2 & 3 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 4 & 5 \end{bmatrix} = \begin{bmatrix} 16 & 20 \\ 18 & 17 \end{bmatrix}.$$

Обратная матрица. По аналогии с обратной величиной по отношению к некоторому скаляру для невырожденной квадратной матрицы \mathbf{a} существует единственная матрица \mathbf{a}^{-1} , обладающая тем свойством, что

$$\mathbf{aa}^{-1} = \mathbf{a}^{-1}\mathbf{a} = \mathbf{I}.$$

Существуют различные способы, позволяющие выразить элементы матрицы \mathbf{a}^{-1} через элементы матрицы \mathbf{a} ; желающие познакомиться с этими способами могут обратиться к соответствующим учебным пособиям¹⁾.

Если $\mathbf{a}^T = \mathbf{a}^{-1}$, то матрица \mathbf{a} называется *ортогональной*.

Матрица, получающаяся в результате умножения \mathbf{a} на \mathbf{a}^T , является симметрической. Действительно, пусть

$$\mathbf{b} = \mathbf{aa}^T.$$

Тогда

$$\mathbf{b}^T = (\mathbf{aa}^T)^T = (\mathbf{a}^T)^T \mathbf{a}^T = \mathbf{aa}^T.$$

Для матрицы, транспонированной по отношению к невырожденной квадратной матрице \mathbf{a} , обратная матрица равняется матрице, транспонированной по отношению к \mathbf{a}^{-1} , т. е.

$$(\mathbf{a}^T)^{-1} = (\mathbf{a}^{-1})^T.$$

¹⁾ Читателю, знакомому с теорией определителей, напомним, что для нахождения обратной матрицы можно воспользоваться следующей (универсальной) формулой:

$$\mathbf{a}^{-1} \equiv [a_{ik}]^{-1} \equiv \left[\frac{\mathbf{A}_{ki}}{\det(\mathbf{a})} \right],$$

где \mathbf{A}_{ki} — алгебраическое дополнение элемента a_{ik} в определителе $\det(\mathbf{a})$. Ясно, что условие невырожденности \mathbf{a} является существенным: при $\det(\mathbf{a}) = 0$ приведенная выше формула теряет смысл.— *Прим. перев.*

Нормировка. Длина действительного вектора, т. е. вектора, все составляющие которого представляют собой действительные числа, определяется следующим образом:

$$\text{Длина вектора } \mathbf{x} = (\mathbf{x}^T \mathbf{x})^{1/2} = \left(\sum_{i=1}^n x_i^2 \right)^{1/2}.$$

Процедура нормировки вектора \mathbf{x} сводится к делению каждой составляющей \mathbf{x} на длину этого вектора (с тем, чтобы получить в результате единичный вектор). Пусть, например,

$$\mathbf{x} = [1, 2, -3, 0].$$

Тогда

$$\text{Длина } \mathbf{x} = \sqrt{1^2 + 2^2 + (-3)^2 + 0^2} = \sqrt{14}$$

и

$$\hat{\mathbf{x}} = \left[\frac{1}{\sqrt{14}}, \frac{2}{\sqrt{14}}, \frac{-3}{\sqrt{14}}, 0 \right]^T.$$

B.3. ПОЛОЖИТЕЛЬНО ОПРЕДЕЛЕННЫЕ МАТРИЦЫ

Каждой квадратичной форме

$$f(\mathbf{x}) = \sum_{\substack{i=j \\ j=i}}^n a_{ij} x_i x_j$$

соответствует действительная¹⁾ симметрическая квадратная матрица \mathbf{a} , или, используя матричные обозначения, можно написать

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{a} \mathbf{x}.$$

Матрицу \mathbf{a} называют положительно полуопределенной (или неотрицательно определенной), если $f(\mathbf{x}) \geq 0$, и положительно определенной, если $f(\mathbf{x}) > 0$ для любого $\mathbf{x} \neq 0$. Квадратичной форме $f(\mathbf{x})$ соответствует также бесконечное число других квадратных матриц \mathbf{b}_n , не являющихся симметрическими:

$$f(\mathbf{x}) = \mathbf{x}^T \mathbf{a} \mathbf{x} = \mathbf{x}^T \mathbf{b}_n \mathbf{x}.$$

Хорошо известно, что необходимые и достаточные условия, при выполнении которых имеет место положительная определенность, сводятся к следующему: все главные миноры и определитель матрицы \mathbf{a} должны быть положительными (или, что эквивалентно сформулированному выше требованию, должны быть положительными все собственные значения данной матрицы). Эти утверждения справедливы лишь в том случае, если иметь в виду

¹⁾ Матрица называется действительной, если все ее элементы представляют собой действительные числа. — Прим. перев.

именно матрицу \mathbf{a} , а не какую-либо из матриц \mathbf{b}_n . Действительная несимметрическая матрица может удовлетворять приведенным выше требованиям и все же не быть положительно определенной.

Рассмотрим, например,

$$f(\mathbf{x}) = x^2 + 3x + 2 = [1 \quad x] \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = \mathbf{x}^T \mathbf{b} \mathbf{x},$$

где

$$\mathbf{x} = \begin{bmatrix} 1 \\ x \end{bmatrix} \text{ и } \mathbf{b} = \begin{bmatrix} 2 & 3 \\ 0 & 1 \end{bmatrix}.$$

Собственными значениями \mathbf{b} являются 2 и 1, $\det(\mathbf{b}) = 2$ и главный минор \mathbf{b} равен 2; тем не менее $f(\mathbf{x})$ принимает отрицательные значения в интервале $-2 < x < -1$. С другой стороны, $f(\mathbf{x})$ может быть представлена в виде

$$f(\mathbf{x}) = x^2 + 3x + 2 = [1 \quad x] \begin{bmatrix} 2 & \frac{3}{2} \\ \frac{3}{2} & 1 \end{bmatrix} \begin{bmatrix} 1 \\ x \end{bmatrix} = \mathbf{x}^T \mathbf{a} \mathbf{x},$$

где \mathbf{a} — симметрическая матрица. Собственными значениями \mathbf{a} являются $\lambda_1 = \frac{3}{2} + \sqrt{10/2}$ и $\lambda_2 = \frac{3}{2} - \sqrt{10/2}$ (отрицательное число) и $\det(\mathbf{a}) < 0$, что полностью согласуется с приведенными выше критериями положительной определенности при записи $f(\mathbf{x})$ с помощью симметрической матрицы,

Приложение Г

СТАНДАРТНАЯ ТАЙМЕР-ПРОГРАММА

```

PROGRAM STDTR (OUTPUT)
DIMENSION A(40,40),C(40),NA(41),NR(41)
1 NDIM = 40
N = 40
NN = 10
T = 0.000001
DO 50 L = 1, NN
DO 25 I = 1, N
DO 25 J = 1, N
A(I,J) = 1.
IF(I-J) 25,20,25
20 A(I,J) = I + J
25 CONTINUE
CALL MATINV(A,NA,NP,N,NDIM,T)
50 CONTINUE
PRINT 100, ((A(I,J)+J = 1, N)*I = 1, N)
100 FORMAT(1, (10F13.5))
END
SUBROUTINE MATINV(A,IROW,ICOL,N,NDIM,SMLST)
DIMENSION A(1),IPRW(1),ICOL(1)
NPI = N + 1
DO 5 I = 1, N
ICOL(I) = I
5 IPW(1) = I
DO 75 ITER = 1, N
MAXR = ITER
15 CONTINUE
IF(TEMP-SMLST) 20,20,25
20 IROW(NPI) = ITER
RETURN
25 IF(MAXR-ITER) 30,40,30
30 DO 35 J = 1, N
MAXPJ = (J-1)*NDIM + MAXR
ITJ = (J-1)*NDIM + ITER
TFMP = A(MAXPJ)
A(MAXPJ) = A(ITJ)
45 ITJ = TEMP
ITEMP = IROW(MAXR)
IPRW(MAXP) = IPW(ITER)
IPW(ITER) = ITEMP
40 IF(IMAXO = 1) 45,55,45
45 DO 50 I = 1, N
IMAXO = (MAXO-1)*NDIM + I
TEMP = A(I)
A(I) = A(IMAXO)
50 A(IMAXO) = TEMP
55 ITEMP = ICOL(MAXO)
ICOL(MAXO) = ICOL(1)
ICOL(1) = ITEMP
60 TEMP = A(ITER)
ITEMP = ICOL(1)
DO 60 J = 2, N
ITJM1 = (J-2)*NDIM + ITER
ITJ = (J-1)*NDIM + ITER
A(ITJM1) = A(ITJ)/TEMP
60 ICOL(J-1) = ICOL(J)
ITN = (N-1)*NDIM + ITER
A(ITN) = 1.0/TEMP
ICOL(N) = ITEMP
DO 75 I = 1, N
IF(I-ITER) 65,75,65
65 TEMP = A(I)
DO 70 J = 2, N
IJM1 = (J-2)*NDIM + 1
IJ = (J-1)*NDIM+1
ITJM1 = (J-2)*NDIM+1TER
A(IJM1) = A(IJ) - A(ITJM1)*TE
70 CONTINUE
IN = (N-1)*NDIM + 1
ITN = (N-1)*NDIM + ITER
A(IN) = -(TEMP*A(ITN))
75 CONTINUE
DO 100 I = 1, N
DO 80 J = 1, N
IF(IROW(J) = 1) 80,89,80

```

```
80 CONTINUE
85 IF(I-J)90,100,90
90 DO 95 L = 1, N
      LI = (I-1)*NDIM + L
      LJ = (J-1)*NDIM + L
      TEMP = A(LI)
      A(LI) = A(LJ)
      A(LJ) = TEMP
      IROW(J) = IROW(I)
100 CONTINUE
      DO 125 I = 1, N
      DO 105 J = 1, N
      IF(ICOL(J)-I) 105,110,105
110 IF(I-J)115,125,115
115 DO 120 L = 1, N
      IL = (L - 1)*NDIM + I
      JL = (L - 1)*NDIM + J
      TEMP = A(IL)
      A(IL) = A(JL)
      A(JL) = TEMP
      ICOL(J) = ICOL(I)
      GO TO 125
105 CONTINUE
125 CONTINUE
      IROW(NP1) = 0
      RETURN
END
```

Приложение Д

ОБОЗНАЧЕНИЯ



- a — константы или постоянные коэффициенты (различаются индексами);
 a_{ij} — элементы матрицы \mathbf{a} (или \mathbf{A});
 \mathbf{a} — матрица коэффициентов;
 a_j — вектор-столбец матрицы;
 $\mathcal{A}^{(s)}$ — числовое значение, определяемое по формуле (8.3.7);
 \mathbf{A} — матрица, состоящая из постоянных элементов a_{ij} ;
 \mathbf{A}_l — векторы, ассоциированные с методом Розенброка;
 $\mathbf{A}^{(k)}$ — матрица, ассоциированная с соотношением (3.4.5);
 \mathbf{A}_l — матрица, состоящая из коэффициентов, фигурирующих в совокупности l ограничений;
 b — константы или постоянные коэффициенты (различаются индексами);
 \tilde{b} — собственное значение матрицы (каждому собственному значению присваивается свой индекс);
 \mathbf{b} — вектор-столбец, составленный из коэффициентов при соответствующих переменных;
 b_j — j -я строка матрицы \mathbf{B}_m ;
 \mathbf{B}_l — матрицы, определение которых дано в связи с рассмотрением соотношения (4.3.2);
 $\mathbf{B}^{(k)}$ — матрица, ассоциированная с соотношением (3.4.5);
 \mathbf{B}_m — матрица, состоящая из коэффициентов, фигурирующих в системе ограничений, имеющих вид неравенств;
 c — константы или постоянные коэффициенты (различаются индексами);
 c_i — константа;
 c_{ii} — диагональный элемент матрицы \mathbf{C} ;
 \mathbf{c} — вектор-столбец матрицы \mathbf{C} ;
 \mathbf{C} — диагональная матрица, определение которой дано в связи с рассмотрением соотношения (3.2.8);

- d — символ полной производной; элемент матрицы \mathbf{D} ;
- d_i — случайная величина, характеризующаяся нормальным распределением;
- d_k — величина, определение которой дано в связи с рассмотрением соотношения (6.5.10);
- D_i — направляющий косинус;
- \mathbf{D} — матрица, с помощью которой строится правильный многогранник в E^n ;
- \mathbf{D}_i — вектор-столбец матрицы \mathbf{D} ;
- \mathbf{e}_i — собственный вектор, соответствующий a_i ;
- E — эффективность при одномерном поиске;
- $E(k)$ — относительное улучшение целевой функции на k -м этапе оптимизационного поиска;
- E^n — n -мерное евклидово пространство;
- $E_{i,i}$ — погрешность за счет линеаризации в методе ПОП;
- $E(\mathbf{x}, \mathbf{u}, \mathbf{w})$ — функция, двойственная по отношению к P -функции;
- $f(\mathbf{x})$ — целевая функция; значение целевой функции в точке \mathbf{x} ;
- $f^*(\mathbf{x}^*)$ — оптимальное значение $f(\mathbf{x})$;
- $F(\mathbf{x})$ — аргумент $f(\mathbf{x})$;
- F_n — число Фибоначчи;
- $F(\mathbf{x}^{(k)}, d)$ — параметр в алгоритме Голдштейна — Прайса;
- $g(\mathbf{x})$ — функция, задающая ограничение в виде неравенства (каждой функции присваивается свой индекс);
- $\Delta g^{(i)} = [\nabla f(\mathbf{x}^{(i+1)}) - \nabla f(\mathbf{x}^{(i)})];$
- $G(g_i(\mathbf{x}))$ — функционал над множеством ограничений в виде неравенств;
- $\mathbf{G}^{(i)} = [\Delta g^{(0)} \Delta g^{(1)} \dots \Delta g^{(i-1)}];$
- $h_{ii} (\tilde{h}_{ii})$ — диагональный элемент матрицы $\mathbf{H}(\mathbf{x}) (\tilde{\mathbf{H}}(\mathbf{x}))$;
- \mathbf{h} — вектор, составляющими которого являются функции, задающие ограничения в виде равенств;
- $h_i(\mathbf{x})$ — функция, задающая i -е ограничение в виде равенства;
- $\mathbf{H}, \mathbf{H}(\mathbf{x})$ — матрица Гессе целевой функции;
- $\tilde{\mathbf{H}}, \tilde{\mathbf{H}}(\mathbf{x}), \tilde{\mathbf{H}}^*(\mathbf{x})$ — приближенная (или усредненная) матрица Гессе;
- $H(h_i(\mathbf{x}))$ — функционал над множеством ограничений в виде равенств;
- \mathbf{I} — единичная матрица;
- \mathbf{I}_j — j -й столбец матрицы \mathbf{I} ;
- \mathbf{J} — матрица Якоби;

- K — большое (фиксированное) число;
 K' — совокупность индексов, дополняющих множество индексов, ассоциированных с базисом;
 L — функция Лагранжа;
 L_i — нижняя граница для x_i ;
 m — число ограничений в виде равенств;
 $m_j^{(k)}$ — величины, определение которых дано в связи с рассмотрением (6.1.3);
 $\mathbf{M}_l^{(k)}$ — оценка $(\mathbf{A}_l \boldsymbol{\eta}^{(k)} \mathbf{A}_l^T)^{-1}$ на k -м этапе вычислительного процесса;
 M — линейное многообразие;
 n — суммарное число переменных;
 N — норма;
 p — полное число ограничений (как в виде равенств, так и в виде неравенств);
 \hat{p} — число ограничений в виде неравенств, оказавшихся нарушенными во внешней точке;
 p^* — полное число ограничений в виде неравенств;
 $p_j^{(k)}$ — величины, определение которых дано в связи с рассмотрением (6.3.1);
 $P(\mathbf{x}^{(k)}, \rho^{(k)})$ — штрафная функция (обобщенная присоединенная функция) с весом $\rho^{(k)}$ на k -м шаге;
 $P(\mathbf{x}^{(k)}, r^{(k)})$ — штрафная функция (обобщенная присоединенная функция) с весом $r^{(k)}$ на k -м шаге;
 $P(\mathbf{x})$ — обобщенная присоединенная функция; штрафная функция;
 $\mathbf{P}_l^{(k)}$ — проектирующая матрица на k -м шаге (индекс l обозначает число связанных с нею ограничений);
 $\hat{\mathbf{P}}_l^{(k)}$ — проектирующая матрица (индекс l обозначает число связанных с нею ограничений);
 q — константа;
 $q_i^{(k)}$ — величины, определение которых дано в связи с рассмотрением (6.3.1);
 $q(\mathbf{x})$ — квадратичная целевая функция или ее приближение;
 \mathbf{Q} — положительно определенная квадратная матрица;
 $r = (n - m)$ — число степеней свободы; весовой коэффициент в методе штрафных функций;
 $\mathbf{r}(\mathbf{x}^{(k)})$ — вектор возврата, определяемый соотношением (6.3.17);
 \mathbf{r}_i — диагональная матрица, элементами которой являются случайные числа;
 R — множество допустимых точек в E^n ;

- $$R(x) = \sum_{i=1}^p \frac{1}{g_i(x)};$$
- R** — симметрическая матрица, используемая в алгоритме Пирсона;
- s** — расстояние;
- s** — вектор-столбец, задающий направление оптимизационного поиска;
- \hat{s} — единичный вектор-столбец, указывающий направление оптимизационного поиска;
- S^* — множество допустимых точек;
- $S^{(k)}$ — вектор, составляющими которого являются масштабные коэффициенты;
- t** — расстояние между двумя вершинами; параметр размера исходного многогранника в методе скользящего допуска;
- T(x)** — функционал над множеством всех ограничений, т. е. над множеством функций $h_i(x)$ и $g_i(x)$;
- u_i — ослабляющая переменная для i -го неравенства;
- u_j — множитель Лагранжа, ассоциированный с j -м ограничением в виде неравенства;
- u_j^* — множитель Лагранжа в экстремальной точке;
- u** — вектор, составляющими которого являются множители Лагранжа;
- $u_l^{(k)}$ — полученный на k -м шаге вектор, составляющими которого являются множители Лагранжа в l активных ограничениях;
- U** — унитарная матрица;
- U_i — оператор Хевисайда (принимает значения 0 или 1);
- U_i — верхняя граница для x_i ;
- v_j — ослабляющая переменная для j -го неравенства;
- v** — любой вектор-столбец общего вида;
- w_i — искусственная переменная для i -го равенства;
- w_i — множитель Лагранжа, ассоциированный с ограничением в виде равенства;
- w_j^* — множитель Лагранжа в экстремальной точке;
- W** — положительно определенная симметрическая матрица;
- x** — произвольная независимая переменная;
- x_i — составляющие вектора x ;
- x_i^* — составляющие вектора x^* ;
- x_{ij} — j -я составляющая i -го вектора в E^n ;
- \tilde{x}_i — аппроксимация x_i ;

- \mathbf{x} — вектор-столбец, составляющими которого являются независимые переменные;
- \mathbf{x}^* — значение \mathbf{x} , соответствующее оптимальному решению задачи нелинейного программирования;
- x_h — вершина, соответствующая наибольшему значению целевой функции;
- \mathbf{x}_i — i -й вектор (или i -я вершина), ассоциированный (или ассоциированная) с минимизацией $f(\mathbf{x})$;
- \mathbf{x}_l — вершина, соответствующая наименьшему значению целевой функции;
- $\hat{\mathbf{x}}_l$ — i -й вектор, ассоциированный с минимизацией $T(\mathbf{x})$;
- $\tilde{\mathbf{x}}^*$ — приближенное значение \mathbf{x} , при котором достигается минимум $f(\mathbf{x})$;
- X_t — произвольный скаляр; произвольный полином; произвольный вектор;
- $\mathbf{X}^{(t)} = [\Delta\mathbf{x}^{(0)} \Delta\mathbf{x}^{(1)} \dots \Delta\mathbf{x}^{(t-1)}]^T$;
- y — переменная величина; в ряде случаев — целевая функция;
- $y_i^{(k)}$ — значение независимой переменной, полученное в результате одномерного оптимизационного поиска;
- \mathbf{y} — произвольный вектор-столбец;
- \mathbf{z}_j — составляющая приведенного градиента;
- \mathbf{z} — вектор-столбец произвольного вида; приведенный градиент; вектор ретроспективных данных, используемый в (4.5.1);
- $Z(\mathbf{x}) = \sum_{i=1}^p g_i^2(\mathbf{x})$ — сумма квадратов функций, ассоциированных с теми ограничениями, которые оказываются нарушенными (для внешней точки).

ГРЕЧЕСКИЕ БУКВЫ И СИМВОЛЫ

- α — коэффициент растяжения в методе Розенброка; обычная константа; коэффициент отражения в методе Нелдера и Мида;
- α_i — собственное значение матрицы;
- α_k — коэффициент;
- β — коэффициент сжатия в методе Нелдера и Мида;

- константа в соотношении (3.2.11); коэффициент редукции в методе Розенброка;
- γ — коэффициент растяжения в методе Нелдера и Мида; малая постоянная величина; обычная константа;
- γ — величина, определяемая с помощью соотношения (6.3.8);
- Γ — величина, определение которой дано в подразд. 6.5.4;
- δ — параметр в алгоритме Голдштейна — Прайса;
- δ_i — единичный вектор вдоль i -й оси координат;
- $\delta_j^{(k)}$ — константа; параметр;
- $\delta(\mathbf{x})$ — аргумент, для которого функция $f(\mathbf{x})$ при некоторых \mathbf{x} в E^n не определена;
- ∂ — символ частной производной;
- Δ — символ приращения переменной величины или вектора (например, Δx);
- $\Delta^{(k)}$ — ограничивающий интервал при одномерном поиске;
- $\nabla_{\mathbf{x}}$ — градиент, включающий частные производные лишь по составляющим вектора \mathbf{x} ;
- $\nabla \phi(\mathbf{x})$ — вектор-градиент функции $\phi(\mathbf{x})$;
- $\nabla^2 \phi(\mathbf{x})$ — матрица Гессе для $\phi(\mathbf{x})$;
- ε — произвольно малое положительное число;
- η — весовой коэффициент при лагранжиане;
- $\eta^{(k)}, \eta(\mathbf{x}^{(k)})$ — матрица, задающая направление поиска на k -м этапе вычислительного процесса;
- $\eta_l^{(k)}$ — матрица, задающая направление поиска на k -м этапе вычислительного процесса при наличии l ограничивающих условий;
- θ — константа, значения которой лежат в интервале от 0 до 1; в ряде случаев обозначает также угол;
- $\theta^{(k)}$ — параметр в алгоритме Голдштейна — Прайса; см. соотношение (8.1.2);
- λ, λ^* — параметр, задающий длину шага в направлении оптимизационного поиска;
- $\lambda^{(k)}, \lambda^{*(k)}$ — длина шага, обеспечивающего минимизирующую поправку для $f(\mathbf{x})$ при перемещении в направлении поиска (в ряде случаев имеет нижний индекс);
- λ_j — содержимое счетчика в методе НЛП;
- Λ_i — направление поиска, определение которого дано в связи с рассмотрением (4.3.1);
- μ_i — масштабный множитель;
- ν^0 — коэффициент в (3.3.4);

- v_0 — двойственное верхнее предельное значение в МПБМ;
 ξ — выбираемая константа, используемая в ПОП, а также малая величина;
 π_k — плоскость, касательная к линии уровня целевой функции в точке $x^{(k)}$;
 Π — масштабированная приближенная матрица Гессе;
 $\tilde{\Pi}$ — аппроксимация Π , но с положительными собственными значениями;
 ρ — весовой множитель в структуре штрафной функции;
 ρ_j — параметр, учитывающий число осцилляций в процессе применения метода НЛП;
 τ — единичный координатный вектор по отношению к базису A ;
 τ_j — составляющие вектора τ ;
 $\gamma^{(k)}$ — аппроксимация матрицы Гессе целевой функции;
 ϕ — случайный угол (при равномерном распределении);
 $\phi(x)$ — выпуклая функция; иногда функция произвольного вида;
 ϕ — вектор-столбец, введенный в (6.3.17);
 $\Phi^{(k)}$ — критерий допуска для нарушений ограничений в методе скользящего допуска;
 $\varphi(x^{(k)})$ — направление в алгоритме Голдштейна — Прайса;
 $\varphi(x)$ — функция вектора x ;
 ω_i — весовой коэффициент;
 Ω , — r -я строка матрицы $(\partial h / \partial x_i)^{-1}$

ВЕРХНИЕ ИНДЕКСЫ

- j, c, m — обозначают точки при одномерном поиске;
 k — номер этапа (или шага) в процессе минимизации $f(x)$;
 s — номер этапа (шага) в процессе минимизации $T(x)$;
 $T.$ — обозначает операцию транспонирования.

НИЖНИЕ ИНДЕКСЫ

- h — i -я вершина в E^n , соответствующая наибольшему значению $f(x)$;
 l — множество ограничений рассматриваемой задачи;
 M — проекция вектора на многообразие M .

ЗНАЧКИ НАД БУКВАМИ И СИМВОЛАМИ

- \sim — знак приближения или преобразования;
- \wedge — знак, применяемый для обозначения единичного вектора;
- $'$ — используется для того, чтобы отличить одну константу (или переменную величину) от другой константы (или переменной величины).

ДОПОЛНИТЕЛЬНЫЕ ОБОЗНАЧЕНИЯ

- $\| \cdot \|$ — показывает, что имеется в виду длина вектора; норма, т. е. квадратный корень из суммы квадратов составляющих вектора;
- $| \cdot |$ — символ абсолютного значения скаляра;
- $\{ \cdot \}$ — символ множества;
- $| \dots |$ — означает «при условии, если»;
- \in — знак принадлежности (элемента множеству);
- \prod — оператор умножения.

СОДЕРЖАНИЕ¹⁾

Предисловие	5
Часть I. Предварительные сведения	
Глава 1. Введение	8
Глава 2. Задача нелинейного программирования и ее оптимальное решение	14
2.1. Задача линейного программирования	14
2.2. Общая задача нелинейного программирования	18
2.3. Связь задачи нелинейного программирования с реальным процессом	22
2.4. Обозначения и терминология	30
2.5. Необходимые и достаточные условия оптимальности решения	39
2.6. Эффективные методы одномерного поиска	50
2.7. Классификация методов нелинейного программирования	58
Литература	69
Часть II. Методы нелинейного программирования без ограничений	
Глава 3. Методы минимизации без ограничений, использующие производные	72
3.1. Градиентные методы	72
3.2. Метод вторых производных (метод Ньютона) и связанные с ним алгоритмы	83
3.3. Сопряженность и сопряженные направления	98
3.4. Методы переменной метрики	117
3.5. Краткий обзор алгоритмов программирования без ограничений	142
Литература	153
Глава 4. Методы минимизации без ограничений, не использующие производные (методы поиска)	156
4.1. Прямой поиск	157
4.2. Поиск по деформируемому многограннику	163
4.3. Методы Розенброка и Дэвиса, Свенна, Кемпи	173
4.4. Метод Паузэлла	184
4.5. Методы случайного поиска	193
Литература	207
Глава 5. Сравнение алгоритмов нелинейного программирования при отсутствии ограничений	210

¹⁾ Перевод предисловия и глав 1—5 выполнен И. М. Быховской, главы 6—9, приложения А—Д переведены Б. Т. Вавиловым.

5.1. Критерии оценки	210
5.2. Тестовые задачи	214
5.3. Оценивание алгоритмов нелинейного программирования при отсутствии ограничений	220
Литература	237
Часть III. Методы нелинейного программирования при наличии ограничений	
Глава 6. Процедуры минимизации при наличии ограничений: методы линейной аппроксимации	242
6.1. Аппроксимирующее линейное программирование	246
6.2. Алгоритм нелинейного программирования	266
6.3. Проективные методы	269
6.4. Метод допустимых направлений (метод Заутендейка)	300
6.5. Метод обобщенного приведенного градиента (МОПГ)	303
Литература	329
Глава 7. Процедуры минимизации при наличии ограничений: методы штрафных функций	383
7.1. Методы штрафных функций специальной структуры	337
7.2. Метод последовательной безусловной минимизации (комбинированный метод штрафных функций)	345
Литература	378
Глава 8. Процедуры минимизации при наличии ограничений: метод скользящего допуска	381
8.1. Определение Φ , $T(x)$ и почти допустимых точек	382
8.2. Стратегия алгоритма скользящего допуска	385
8.3. Процедура отыскания допустимых и почти допустимых точек	392
8.4. Начало и окончание поиска	400
8.5. Методы решения задач нелинейного программирования с зональной неопределенностью	407
Литература	410
Глава 9. Оценка эффективности методов нелинейного программирования при наличии ограничений	411
9.1. Критерии, используемые при оценке эффективности алгоритмов нелинейного программирования	411
9.2. Сравнение некоторых алгоритмов нелинейного программирования при наличии ограничений: двумерные задачи	416
9.3. Сравнение некоторых алгоритмов оптимизации при наличии ограничений в случае более сложных задач	424
Литература	443
Приложение А. Задачи нелинейного программирования и их решения	444
Приложение Б. Программы на языке ФОРТРАН, непоставляемые коммерчески	480
Приложение В. Матрицы	516
Приложение Г. Стандартная таймер-программа	523
Приложение Д. Обозначения	525

УВАЖАЕМЫЙ ЧИТАТЕЛЬ!

Ваши замечания о содержании книги, ее оформлении, качестве перевода и другие просим присыпать по адресу: 129820, Москва, И-110, ГСП, 1-й Рижский пер., 2, издательство «Мир».

Д. ХИММЕЛЬБАУ
ПРИКЛАДНОЕ НЕЛИНЕЙНОЕ
ПРОГРАММИРОВАНИЕ

Редактор Л. П. Якименко

Художник С. А. Бычков

Художественный редактор В. К. Бисенгалиев

Технический редактор Т. А. Максимова

Сдано в набор 22/1 1975 г. Подписано к печати 29/VIII 1975 г.
Бумага № 2 60×90^{1/16} = 16,75 бум. л. 33,50 печ. л. Уч.-изд. л. 31,78.
Изд. № 20/7553. Цена 2 р. 35 к. Зак. 844.

ИЗДАТЕЛЬСТВО «МИР»
Москва, 1-й Рижский пер., 2

Отпечатано в ордена Трудового Красного Знамени
Ленинградской типографии № 2 имени Евгении Соколовой
Союзполиграфпрома при Государственном комитете
Совета Министров СССР
по делам издательств, полиграфии и книжной торговли.
198052, Ленинград Л-52, Измайловский проспект, 29, с матриц
Головного предприятия республиканского производственного
объединения «Полиграфкнига» Госкомиздата УССР, г. Киев,
ул. Довженко, 3.