**МАТЕМАТИЧЕСКОЕ ПРОГРАММИРОВАНИЕ**

В общем случае задача математического программирования записывается в виде:



Если хотя бы одна функция в модели (8.1) нелинейна, имеем задачу нелинейного программирования (НП), в противном случае имеем задачу линейного программирования.

**ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ**

**4.1. Постановка задачи**

Задача, модель которой содержит только линейные функции искомых переменных, называется ***задачей линейного программирования***(ЛП).

В общем случае модель задачи ЛП имеет вид

 (4.1)

при ограничениях:

 (4.2)

** (4.3)

где *L* – критерий (целевая функция), называемый также ***линейной формой***; *n* - количество переменных;  *Ci* – параметры (коэффициенты) критерия, не все *Ci*=0; (4.2) - функциональные условия (ограничения); **– параметры условий (могут быть любыми действительными числами, но одновременно все не могут равняться нулю при *i=const).* Во многих случаях они имеют смысл удельных величин (расхода или затрат на единицу переменной, содержания в единице переменной и т. п.). *b*i – параметры (свободные члены), отражающие возможности по ресурсам, допустимые или требуемые значения показателей и т.п. (могут быть любыми действительными числами). На часть или все переменные накладывается условие неотрицательности (4.3).

Задача состоит в определении таких значений переменных, удовлетворяющих условиям (4.2) и (4.3), которые доставляют в зависимости от контекста максимум или минимум линейной форме.

**4.3. Формы записи задач линейного программирования и способы приведения к ним**

***4.3.1*. *Каноническая форма задач ЛП***

Задача ЛП представлена в канонической форме, если в ее модели *все* функциональные условия имеют вид равенств и *все* переменные ограничены по знаку. Направление цели не имеет существенного значения, но для однозначности канонического представления будем иметь в виду максимизацию критерия. Тогда модель задачи ЛП в канонической форме записывается следующим образом

**4.11. Двойственность задач ЛП**

Любой задаче ЛП можно поставить в соответствие другую задачу, называемую сопряженной или *двойственной*. При этом исходную задачу называют прямой.

Выделяют общий и симметричный случаи двойственности. Если в прямой задаче все условия представлены в виде неравенств и все переменные ограничены по знаку, то имеет место симметричная пара двойственных задач.

Когда в исходной задаче есть равенства и/или переменные, которые не ограничены по знаку, то говорят об общем случае двойственности (симметрия моделей отсутствует).

***4.11.1.* *Запись двойственной задачи в симметричном случае***

**Рассмотрим пример.** Пусть некоторая фирма способна выпускать 3 вида продукции, используя 4 вида ресурсов. Известны произведенная стоимость единицы продукции *Cj*, норма расхода каждого вида ресурса на единицу продукции *Aij*и количество ресурсов *bi*.

Модель прямой задачи, отражающей стремление произвести максимум продукции в стоимостном выражении, очевидна:

*L=C*1*x*1*+C*2*x*2*+C*3*x*3*→ max;*

*U*1*: A*11*x*1*+A*12*x*2*+A*13*x*3*≤ b*1*;*

*U*2*: A*21*x*1*+A*22*x*2*+A*23*x*3*≤ b*2*;*

*U*3*: A*31*x*1*+A*32*x*2*+A*33*x*3*≤ b*3*;*

*U4: A4*1*x*1*+A4*2*x*2*+A4*3*x*3*≤ b4;*

*∀xj* ≥0*.*

Она отвечает условиям симметрии, и модель ее двойственной задачи запишется в виде

*=b*1*U*1*+b*2*U*2*+b*3*U*3*+b4U4→ min;*

*A*11*U*1*+A*21*U*2*+A*31*U*3*+A4*1*U4≥ C*1*;*

*A*12*U*1*+A*22*U*2*+A*32*U*3*+A4*2*U4≥ C*2*;*

*A*13*U*1*+A*23*U*2*+A*33*U*3*+A4*3*U4≥ C*3*;*

*∀Ui* ≥ 0*.*

Здесь **– критерий двойственной задачи, *Ui* – переменные двойственной задачи или, просто, двойственные переменные.

**Правила, по которым составлена эта модель, включают 5 пунктов**:

1. Если в прямой задаче целевая функция максимизируется, то в двойственной минимизируется, и наоборот.
2. Коэффициенты критерия двойственной задачи образуются из компонентов вектора ограничений прямой задачи.
3. Компоненты вектора ограничений двойственной задачи образуются из коэффициентов линейной формы (критерия) прямой задачи.
4. Матрица условий двойственной задачи образуется транспонированием матрицы условий прямой задачи.
5. Знаки неравенств двойственной задачи обратны знакам неравенств прямой.

Для однозначной записи двойственной модели в прямой задаче на максимум все неравенства следует привести к виду “меньше или равно”, а в задаче на минимум – к виду “больше или равно”.

Первые 4 правила действуют как в симметричном, так и в общем случае, а пятое правило – только в случае симметрии.

Как следует из приведенных правил, число условий двойственной задачи равно числу переменных прямой задачи, а число переменных двойственной задачи равно числу условий прямой. Если для двойственной задачи построить двойственную, то получим прямую.

***4.11.2.* *Интерпретация двойственной задачи***

Что отражает двойственная модель? Оказывается, она дает возможность оценить решение исходной (прямой) задачи. В рассматриваемом примере прямая задача состоит, фактически, в наилучшем использовании *всех* имеющихся ресурсов. Каждому варианту плана поизводства продукции соответствует свое использование ресурсов, а, следовательно, и их полезность или *значимость.* Под последним понимается степень влияния ресурса на результат. Так как каждому условию прямой задачи, отражающему использование ресурса, ставится в соответствие двойственная переменная, то именно она и является мерилом значимости этого ресурса.

Действительно, рассмотрим уравнение размерности условия двойственной задачи

[A][U]=[C].

Пусть, например, ресурс – фонд времени оборудования (сколько часов оборудование может быть загружено в течение планового периода). Тогда размерность двойственной переменной будет

.

Итак, *U* дает стоимость единицы ресурса в единицах критерия, то есть в нашем случае – прирост произведенной стоимости в рублях на каждый дополнительный час работы оборудования. Ниже, в теоремах двойственности, это будет показано строго математически. Поэтому двойственные переменные называют также *теневыми ценами*. Чтобы увидеть отличие теневой цены от рыночной, возьмем конкретные цифры. Пусть рыночная цена некоторого ресурса, полностью используемого в производстве, равна 500 руб/кг и 1 кг достаточно (при наличии других ресурсов) для выпуска дополнительной продукции на сумму 100000 руб. Тогда теневая цена этого ресурса равна 100000 руб. Если поставщик сорвал поставку данного ресурса, то он должен нести ответственность не в размере рыночной цены, а по теневой цене за каждую единицу недопоставленного ресурса. Такое предложение было высказано впервые Л.Канторовичем, который называл двойственные переменные *объективно обусловленными оценками*, сокращенно О.О.О. (объективные цены, складывающиеся в конкретной ситуации производства и потребления).

Таким образом, чем больше абсолютная величина двойственной переменной, тем выше значимость ресурса в полученном решении, и наоборот, более сильному влиянию ресурса на критерий соответствует большее значение двойственной переменной.

Теперь интерпретируем условия двойственной задачи. Если *Ui* – объективная цена за единицу ресурса, то левая часть неравенства двойственной модели представляет собой *полные затраты* на производство единицы продукции, а все неравенство отражает тот факт, что произведенная стоимость *Ci* не может превышать суммарных затрат.

Значимость ресурса эквивалентна его дефицитности. Поэтому критерий двойственной задачи можно интерпретировать как суммарную дефицитность ресурсов, которую следует минимизировать.

Другая трактовка заключается в том, что двойственная задача моделирует взаимодоговоренности Покупателя и Продавца ресурсов. Продавец готов продать свои ресурсы, отказавшись от производства продукции, если цены на них (*Ui*) будут такими, что он получит за ресурсы, расходуемые им на единицу продукции, не меньше *Ci*, то есть не меньше того, что он имел бы от производства этой продукции*.* Эти требования выражаются неравенствами двойственной задачи. С другой стороны, Покупатель стремится к таким ценам, которые минимизируют плату за все ресурсы. Это стремление и выражает критерий двойственной задачи.

***4.11.3.* *Запись двойственной задачи в общем случае***

Дополнительные правила записи двойственной задачи получим, сводя несимметричные условия прямой задачи к симметричным.

1.Среди условий прямой задачи есть равенство. Пусть таким условием является *k*-е, а остальные условия записаны как неравенства. Заменив *k-*е условие-равенство двумя неравенствами

 ⇔ 

приходим к симметричному случаю. Если новым неравенствам сопоставить неотрицательные двойственные переменные  и , то в соответствии с вышеописанными правилами запишем критерий и неравенства двойственной задачи



После вынесения общих множителей за скобки получаем



Так как  и  входят в модель только в виде разности, то можно произвести замену  и, таким образом, иметь одну двойственнную переменную, соответствующую равенству прямой задачи, но при этом она не будет ограничена по знаку.

2.Переменная *xk* в прямой задаче не ограничена по знаку. Заменим эту переменную всюду в модели разностью неотрицательных переменных:



Этим переменным в двойственной задаче будут соответствовать 2 неравенства



которые эквивалентны равенству



Итак, в общем случае 5-е правило записи двойственной задачи включает 4 пункта, представленные в следующей таблице

|  |  |  |
| --- | --- | --- |
| Правило | Прямая задача | Двойственная задача |
| 5.1 | Переменная *xj*≥0 | *j*-е условие ≥ |
| **5.2** | **Переменная *xj* не ограничена по знаку** | ***j*-е условие =** |
| 5.3 | *i-*е условие ≤ | Переменная *Ui*≥0 |
| **5.4** | ***i-*е условие =** | **Переменная *Ui* не ограничена по знаку** |

Эти правила предполагают, что прямая задача записана с критерием на максимум и неравенствами в виде “меньше или равно”. Очевидно, что в симметричном случае из 5-го правила применяются только пункты 5.1.и 5.3.

Пример 4.4. Прямая задача:

*L=*2*x*1*+x*2*− x*4*+*3*x*5*→ max;*

5*x*1 *−* 7*x*2*+*4*x*3*+*2*x*5≤ 8;

3*x*2*+*6*x*3 *−* 2*x*4≥ 10;

*x*1*+*4*x*2*+x*3 *−*3*x4=*5;

9x1 *− x*2*+*5*x*4 *−*4*x*5≥16;

*x*1≥0, *x*3≥0, *x*4≥0.

Перепишем эту модель, изменив знаки 2-го и 4-го неравенств и сопоставив условиям двойственные переменные:

*L=*2*x*1*+ x*2*− x*4*+*3*x*5 *→ max;*

*U*1*:* 5*x*1 *−* 7*x*2*+*4*x*3 *+*2*x*5≤ 8;

*U*2*: −*3*x*2*−* 6*x*3 *+* 2*x*4≤ *−*10;

*U*3*:  x*1*+* 4*x*2*+ x*3 *−* 3*x*4 *=*5;

*U*4*: −*9x1 *+ x*2 *−* 5*x*4 *+* 4*x*5≤*−*16;

*x*1≥0, *x*3≥0, *x*4≥0.

В соответствии с правилами для общего случая записываем модель двойственной задачи

*=*8*U*1*−*10*U*2*+*5*U*3*−*16*U*4*→* min;

5*U*1 *+U*3*−* 9*U*4*≥* 2;

*−*7*U*1*−*3*U*2*+*4*U*3*+ U*4*=*1;

4*U*1*−*6*U*2*+ U*3*≥* 0;

2*U*2 *−* 3*U*3*−*5*U*4≥ − 1;

2*U*1*+* 4*U*4*=*3;

*U*1≥ 0, *U*2≥ 0, *U*4≥ 0*.*

***4.11.4.* *Тео******ремы двойственности***

Между решениями прямой и двойственной задач существует тесная взаимосвязь, которая устанавливается теоремами двойственности. Эта связь позволяет по решению одной задачи двойственной пары получать решение другой. Основными являются две теоремы, первая из которых определяет связи критериев, а вторая *−* условий и переменных.

***Теорема* 1**. Если в оптимальном решении прямой задачи условие выполняется как строгое неравенство

, (4.32)

то соответствующая двойственная переменная равна нулю, т. е.

Обоснование следует из смысла двойственных переменных. Неравенство (4.32) означает, что *i-*й ресурс используется не полностью, следовательно, малое изменение этого ресурса не повлияет на результат деятельности (критерий) и поэтому значение двойственной переменной равно нулю.



*Следствие.* Если дополнительная переменная в *i-*м условии прямой задачи больше нуля, то соответствующая двойственная переменная равна нулю.

Действительно, в этом случае *i-*е условие без дополнительной переменной будет заведомо строгим неравенством, что и оговорено в теореме.

*Теорема* 2. Если в ***единственном*** оптимальном решении прямой задачи условие выполняется как равенство, то есть

 (4.33)

то соответствующая двойственная переменная будет не равна нулю.

Равенство (4.33) означает, что *i-*й ресурс полностью исчерпан, следовательно, малые изменения этого ресурса обязательно приведут к изменению критерия и поэтому его значимость не равна нулю.

*Следствие.* Если дополнительная переменная в *i-*м условии равна нулю, то двойственная переменная этого условия не равна нулю.

Теоремы 1 и 2 легко трансформируются для двойственной задачи.

*Теорема *. Если в оптимальном решении двойственной задачи условие выполняется как строгое неравенство

 (4.34)

то соответствующая переменная прямой задачи равна нулю: 

Интерпретация: если затраты превышают производимую стоимость, то производить такую продукцию невыгодно.

*Теорема* Если в ***единственном*** оптимальном решении двойственной задачи условие выполняется как равенство, то соответствующая переменная прямой задачи строго больше нуля:

. (4.35)

Так как производимая стоимость равна затратам, то производство такой продукции окупается.

Обобщением рассмотренных теорем является ***вторая основная теорема двойственности:***

Для того чтобы векторы **X***\** и **U\*** являлись оптимальными решениями прямой и двойственной задач соответственно, необходимо и достаточно выполнение следующих условий:

 (4.36)

Эта теорема учитывает и случай множественности оптимальных решений, когда равенству в одной задаче может сответствовать нулевая переменная в другой.

Теперь покажем на конкретном примере, как приведенные теоремы позволяют находить решение одной из задач двойственной пары по известному решению другой.

Пример 4.5. Рассмотрим задачу, которая решалась ранее графически и симплекс-методом.

|  |  |
| --- | --- |
| Прямая задача (ПЗ)  *L*=7*x*1+5*x*2→max,  2*x*1+3*x*2≤19,  2*x*1+*x*2≤13,  3*x*2≤12,  3*x*1≤17,  *x*1≥0, *x*2≥0 | Каноническая форма ПЗ  *L*=7*x*1+5*x*2→max,  2*x*1+3*x*2+*x*3=19,  2*x*1+*x*2+ *x*4=13,  3*x*2+*x*5=12,  3*x*1+*x*6=17,  ∀*xj*≥0. |

Оптимальное решение этой задачи:

Запишем модель двойственной задачи (ДЗ):

=19*U*1+13*U*2+12*U*3+17*U*4→miin;

2*U*1+2*U*2+3*U*4≥7;

3*U*1+*U*2+3*U*3≥5;

∀ *Ui* ≥0.

Получим ее решение на основе решения ПЗ и теорем двойственности.

Так как дополнительные переменные *х5* и *х6*, входящие в третье и четвертое условия ПЗ, в оптимальном решении не равны нулю, то согласно следствию теоремы 1

.

Из первой группы условий (4.36) следует, что если исходная переменная ПЗ не равна нулю, то ограничение ДЗ будет выпополняться как равенство. Поэтому в нашем примере имеем:



Получили систему 2-х уравнений с двумя неизвестными. Ее решение:



.

Таким образом, мы нашли решение ДЗ без применения симплекс-метода. Как увидим ниже, равенство оптимальных значений критериев ПЗ и ДЗ не случайно. Пример только демонстрирует связь решений двойственной пары задач, а значения двойственных переменных легко получить из оптимальной симплекс-таблицы ПЗ. Они расположены в вспомогательной строке *Z* в столбцах начального базиса. Обратившись к симплекс-таблице 3 в разд.4.9.7, легко убедиться в справедливости этого способа нахождения двойственных переменных. Следующая группа теорем определяет связь между критерими двойственной пары задач.

*Теорема* 3*.* Если **X** и **U** – допустимые решения прямой и двойственной задач соответственно, то

*L(***X**) ≤ (**U**). (4.37)

Доказательство. Так как допустимость решений означает выполнение неравенств  в ПЗ и в ДЗ, то очевидна цепочка соотношений



из которой следует справедливость теоремы.

Таким образом, для ***любых допустимых*** решений значение критерия прямой задачи не может превышать значение критерия двойственной.

*Теорема* 4*.* Если **X**\* и **U**\* - допустимые решения прямой и двойственной задач и *L*(**X**\*)=(**U**\*), то они являются оптимальными решениями двойственной пары задач.

Доказательство. Согласно теореме 3 для любого допустимого **X** справедливо неравенство

*L*(**X**)≤ (**U**\*).

И так как *L*(**X**\*)=(**U**\*) по условию теоремы, то *L*(**X**)≤L(**X**\*). Следовательно, **X**\*- оптимальное решение прямой задачи по определению. Аналогично доказывается оптимальность **U**\* для двойственной задачи.

*Теорема* 5*.* Для любых оптимальных **X**\* и **U**\* линейные формы прямой и двойственной задач равны:

*L*(**X**\*)=(**U**\*). (4.38)

Доказательство. В оптимальных решениях выполняются равенства (4.36). Суммируя первую группу по *j* , а вторую по *i*  и сделав простые преобразования, получаем



Из равенства левых частей следует равенство правых и, значит, справедливость теоремы. Теперь ясно, что совпадение значений критериев в приведенном примере не является случайным.

Теорема позволяет строго объяснить смысл двойственных переменных. Действительно, правомерна запись *L\**=\*=. Отсюда имеем

 (4.39)

Таким образом, в оптимальном решении двойственная переменная является производной оптимального значения критерия по правой части ограничения. Значит, как уже говорилось, оптимальная двойственная переменная показывает, как изменится оптимальное значение критерия при изменении ресурса на единицу (она равна этому изменению критерия).

*Теорема 6.* Если линейная форма одной из задач двойственной пары не ограничена, то условия другой противоречивы. (Обратное не всегда верно, возможна противоречивость в обеих задачах).

Доказательство проведем от противного. Допустим, что при неограниченности *L*(**x**) сверху в прямой задаче условия двойственной задачи непротиворечивы. Тогда существует допустимое решение ДЗ, на котором. значение ее критерия конечно. Но согдасно теореме 3 для допустимых решений должно выполняться неравенство *L*(**x**)≤(**U**), что при принятом допущении невозможно (*L* бесконечно, а  конечно). Следовательно, ДЗ не может иметь допустимых решений, то есть ее условия противоречивы.

Аналогично доказывается 2-я часть теоремы для случая неограниченности снизу .

Обобщением теорем 3-5 является ***первая основная теорема двойственности:***

*Если одна из задач двойственной пары разрешима, то и другая задача разрешима, при этом оптимальные значения критериев равны; при неразрешимости одной из задач другая тоже неразрешима*.▲

Что дает двойственность для решения задач ЛП помимо анализа? Во-первых, вместо решения исходной задачи можно решать двойственную. Это выгодно, если в ПЗ число условий существенно больше числа переменных (тогда в ДЗ будет меньше ограничений и потребуется меньше итераций). Кроме того, переход к ДЗ может уменьшить число искусственных переменных или исключить их совсем.

Во-вторых, теория двойственности породила такие методы как двойственный симплекс-метод и метод сокращения невязок или венгерский метод. В последнем используются неотрицательные оптимальные решения, при которых не выпоняются некоторые ограничения-равенства (имеются невязки), но от итерации к итерации невязки уменьшаются. Нулевые невязки являются признаком достижения допустимого оптимального решения. Метод применяется в основном для решения транспортных задач.

**ТРАНСПОРТНЫЕ ЗАДАЧИ**

Критерием задачи являются суммарные затраты на перевозку. Безотносительно к значениям *ai* и *bj* модель записывается в виде



Однако такая запись модели корректна только тогда, когда  Напомним, что задача, в которой суммарные потребности равны суммарной возможности, то есть

 (5.1)

называется сбалансированной или закрытой. Как будет показано в этой главе, любая несбалансированная задача легко приводится к закрытой.  ∀*Xij*≥ 0.

Отметим особенности рассматриваемой задачи:

* Модель содержит две группы условий, размерность которых равна соответствующему числу ПО и ПН; число переменных равно произведению *m*×*n*;
* Все коэффициенты при переменных в условиях (5.3), (5.4) равны единице;
* Каждая переменная входит в условия ровно 2 раза, один и только один раз в группу (5.3) и также один раз в группу (5.4);
* Задача имеет простые условия разрешимости, которые определяются следующей теоремой.

*Теорема.* Для разрешимости Т-задачи *необходимо и достаточно*, чтобы она была *сбалансированной.*

***5.1.5. Транспортные задачи по критерию времени***

При осуществлении перевозок определяющим показателем могут быть не затраты, а время доставки. Тогда вместо матрицы транспортных затрат дается матрица времени [tij], а критерий выражает время завершения всех перевозок:



где максимум берется по коммуникациям, на которых перевозки больше нуля. Предполагается, что перевозки между всеми пунктами начинаются одновременно и ведутся параллельно. Условия задачи записываются как и в случаях с критерием-затратами.

**Глава 7. ЦЕЛОЧИСЛЕННОЕ ПРОГРАММИРОВАНИЕ**

Задачи дискретного программирования отличаются тем, что на переменные накладывается требование дискретности, в частном случае – целочисленности.

**7.1. Проблема целочисленности**

На первый взгляд может показаться, что целочисленная задача решается проще непрерывной. При малой размерности и узких диапазонах переменных задачу можно решить простым перебором. В других случаях необходимы соответствующие методы.

Несмотря на линейность модели допустимое множество целочисленной задачи не является выпуклым. Так, в полностью целочисленной задаче оно представляет собой множество отдельных точек, имеющих целочисленные координаты. Методы линейного программирования базируются на выпуклости допустимого множества и поэтому непосредственно не могут быть применимы к целочисленным задачам.

Можно, конечно, пренебречь требованием целочисленности и использовать один из методов ЛП, но тогда, за редким исключением, результат не будет целочисленным. Округление дробных значений переменных проблематично. Действительно, так как оптимальное решение непрерывной задачи лежит в вершине допустимого множества, округление может привести к недопустимости. При двух дробных переменных имеется 4, а при *n* переменных *–* 2*n* вариантов округления! Какие из них дают допустимые решения, можно определить только после проверки всех ограничений. При этом следует иметь в виду, что, во-первых, целочисленная задача может оказаться неразрешимой несмотря на разрешимость непрерывной задачи; во-вторых, допустимость округленного решения еще не означает его оптимальность.

Для решения таких задач применяются различные подходы. Из точных методов можно назвать следующие:

1. методы отсечений;
2. метод ветвей и границ;
3. метод построения последовательности планов
4. модификации динамического программирования;
5. методы последовательного анализа вариантов.

Кроме точных методов имеется также большое число приближенных методов.

**7.3. Метод ветвей и границ**

Это, скорее, даже не метод, а концепция или процедурная оболочка, на основе которой стали разрабатывать алгоритмы решения целочисленных задач различной природы. Ценность предложенной идеи стала особенно заметна после появления первого точного алгоритма решения задачи коммивояжера, построенного по схеме ветвей и границ (Литтл с соавторами, 1963). Метод можно применять как к полностью, так и частично целочисленным задачам.

Метод заключается в построении дерева задач, корнем которого является исходная задача, возможно без условия целочисленности (НЗ). Нижележащие задачи порождаются вышележащими так, что их допустимые множества (ДМ) являются непересекающимися подмножествами ДМ вышележащей задачи. Рост дерева происходит за счет перспективных ветвей. Перспективность определяется по *оценке критерия* терминальной задачи ветви *V* и *рекорду* *Z.* Оценка *V*– это значение критерия, заведомо не хуже оптимального, а *Z* – достигнутое в процессе решения значение критерия исходной задачи (в качестве начального может приниматься значение, заведомо хуже оптимального). Значит, задача будет порождающей только при условии, что ее оценка лучше рекорда. При этом уровень, на котором находится задача, не имеет значения.

Рассмотрим метод применительно к линейной целочисленной задаче.

Для алгоритмической реализации схемы ветвей и границ необходимо решить два основополагающих вопроса:

1.Каким образом разбивать перспективное множество на подмножества;

2.Как определять верхнюю оценку критерия на рассматриваемом множестве.

Базовый алгоритм, реализующий метод ветвей и границ, включает следующие шаги.

1. Задается начальное значение рекорда и в список задач помещается исходная задача без требования целочисленности переменных.
2. Анализируется список задач: если он пуст, то переход на шаг 6. Иначе выбирается одна из задач с удалением ее из списка.
3. Выбранная задача решается одним из методов линейного программирования. Если задача неразрешима или оптимальное значение критерия *L\** ≤ *Z*, ветвь обрывается (задача прозондирована). Переход на шаг 2.
4. Полученное решение анализируется на целочисленность. Если решение целочисленное, оно фиксируется, рекорду присваивается оптимальное значение критерия решенной непрерывной задачи (*Z*:=*L\**), ветвь обрывается и осуществляется переход на шаг 2.
5. Выбирается одна из переменных, имеющих нецелочисленные значения. По ней производится ветвление: порождаются 2 задачи, одна образуется присоединением к решенной (родительской) задаче условия *хj* ≤, другая – добавлением к родительской ограничения *хj≥* +1. Эти задачи заносятся в список задач. Переход на шаг2.
6. Вывод результатов (если значение рекорда больше начального, получено оптимальное решение исходной задачи, иначе задача неразрешима).

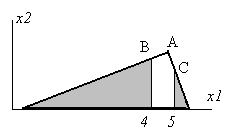
Приведенный алгоритм является базовым, так как не включает однозначных правил выбора задачи из списка и ветвящей переменной. Для частично целочисленных задач при выборе переменной для ветвления исключаются непрерывные переменные.

*Пример 7.3*. Применим алгоритм ветвей и границ к задаче

*L=*9*x*1*+*5*x*2 *→* max; 3*x*1*-* 6*x*2≥1; 5*x*1+2*x*2≤ 28; ∀ *xj* ≥ *0 ,*целые. Отбрасывая условие цедочисленности, получаем непрерывную задачу, которую помещаем в список задач. Так как коэффициенты критерия положительны, начальное значение рекорда принимаем равным нулю. Берем из списка единственную задачу и решаем ее. Получаем оптимальное решение в вершине А (рис. 7.7): *x*1\*=4,72; *x*2\*=2,19 . Ветвление производим по переменной *x*1. Добавляя к решенной задаче ограничение *x*1≤4, образуем задачу 2, а добавление *x*1≥5 дает задачу 3. Допустимые множества новых задач покзаны на рис. 7.7. Эти задачи помещаем в список задач. Решение задачи 2 достигается в точке В, а задачи 3 – в С. Весь ход решения исходной задачи представлен в виде дерева решений на рис. 7.10. Порядок решения задач из списка отражает счетчик итераций *k*. На 3-й 41 итерации (задача 4)

) получено целочисленное решение со значением критерия (точка D на рис. 7.8). Поэтому изменяется рекорд: *Z*=41. Задача 6 имеет нецелочисленное решение (вершина Е на рис. 7.9), задача 8 – целочисленное решение в точке F. В результате после 7-й итерации рекорд становится равным 50. Остальные задачи не имеют допустимых решений, то есть список задач исчерпывается и, таким образом, констатируем получение оптимального решения исходной задачи, равное решению непрерывной задачи 8.

Рис. 7.7



|  |  |
| --- | --- |
| **Задача 4** | |
| *k* =3 | *Z* =0 |
| *L*\*=41  *x*1\*=4  *x*2\*=1 | |

*х*1≥ 6

*х*1≤ 5

*х*2≥ 2

*х*2≤ 1

*х*2≤ 1

*х*2≥ 2

|  |  |
| --- | --- |
| **Задача 1** | |
| *k* =1 | *Z* =0 |
| *L*\*=53,47  *x*1\*=4,72  *x*2\*=2,19 | |

|  |  |
| --- | --- |
| **Задача 2** | |
| *k* =2 | *Z* =0 |
| *L*\*=45,17  *x*1\*=4  *x*2\*=1,83 | |

|  |  |
| --- | --- |
| **Задача 3** | |
| *k* =5 | *Z* =41 |
| *L*\*=52,5  *x*1\*=5  *x*2\*=1,5 | |

*х*1≥ 5

*х*1≤ 4

|  |  |
| --- | --- |
| **Задача 5** | |
| *k* =4 | *Z* =41 |
| Недопустимо | |

|  |  |
| --- | --- |
| **Задача 6** | |
| *k* =6 | *Z* =41 |
| *L*\*=51,8  *x*1\*=5,2  *x*2\*=1 | |

|  |  |
| --- | --- |
| **Задача 7** | |
| *k* =9 | *Z* =50 |
| Недопустимо | |

|  |  |
| --- | --- |
| **Задача 8** | |
| *k* =7 | *Z* =41 |
| *L*\*=50  *x*1\*=5  *x*2\*=1 | |

|  |  |
| --- | --- |
| **Задача 9** | |
| *k* =8 | *Z* =50 |
| Недопустимо | |

Рис. 7.10

Рис.7.8

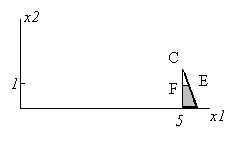
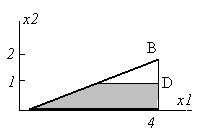


Рис. 7.9

Из приведенного дерева решений видно, что число задач в списке могло быть меньше при другом порядке решения задач. Действительно, если бы сначала были решены задачи правой ветви с рекордом *Z=*50, то после решения задачи 2 не произошло бы ветвления, так как верхняя оценка оказалась бы ниже рекорда (*V=L\**=45,17<50).

Из алгоритма и приведенного примера следует, что ветвь обрывается по одной из трех причин: неразрешимость задачи; задача имеет целочисленное решение; верхняя оценка не больше рекорда.

Теперь сделаем ряд замечаний относительно метода ветвей и границ. Как уже отмечалось, в базовом алгоритме не оговариваются правила выбора задачи и переменной. В большинстве программных реализаций метода используются правила, основанные на эвристических оценках перспективности задач и переменных. В некоторых пакетах, например, "ЛП в АСУ" предлагается несколько вариантов управления процессом решения: от автоматического до ручного, в котором пользователь может сам делать выбор как задачи, так и переменной. Кроме того, алгоритмы, основанные на методе ветвей и границ, могут существенно отличаться в связи с учетом особенностей класса задач. Например, для задачи коммивояжера, определение оценки значительно упрощено (не требуется решать непрерывную линейную задачу).

**Глава 8. НЕЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ**

 (8.1)

Если хотя бы одна функция в модели нелинейна, имеем задачу нелинейного программирования (НП).

Размерность задачи характеризуется размерностью вектора переменных *n* и числом условий *m*1+*m*2. Однако сложность задачи определяется не столько размерностью, сколько свойствами функций цели и ограничений. Универсальных методов решения таких задач не существует. Имеется весьма ограниченное число точных методов и намного больше приближенных.

Наиболее развиты методы решения **задач *выпуклого программирования***. К этому классу относятся задачи НП с выпуклым допустимым множеством и выпуклой целевой функцией при минимизации или вогнутой при максимизации. Допустимое множество выпуклое, если все функции  линейные и  выпуклы при неравенстве ≤ или вогнуты при ≥. Например, условие *x*12+*x*22 ≤ *r*2 порождает выпуклое множество, пересечение которого с прямой *x*1+*x*2=0 дает тоже выпуклое множество. Очевидно, что задачи ЛП относятся к этому классу. Главная особенность задач выпуклого программирования в том, что они *унимодальны*, то есть любой их локальный оптимум является глобальным. Для ряда задач выпуклого программирования с дифференцируемыми функциями разработаны точные методы. Наибольшие сложности возникают при решении многоэкстремальных задач, которые по определению не относятся к классу выпуклых.

Важным классом НП являются задачи ***квадратичного программирования*.** В них целевая функция представляет собой сумму линейной и квадратичной форм, а все условия линейные. При выпуклости (вогнутости) квадратичной формы они являются частным случаем задач выпуклого программирования.

В нелинейном программировании выделяют также задачи *сепарабельного программирования*. Это задачи, в которых все функции сепарабельны. **Функция сепарабельна, если она представляется в виде сумы функций отдельных переменных**. Линейная функция – частный случай сепарабельной. Сепарабельная задача может быть одновременно и задачей выпуклого программирования.

**Задачи *геометрического программирования*** составляют отдельный класс НП. Все функции в таких задачах являются *позиномами*. В общем виде позиномом называется функция

,

в которой *αkj* – любые действительные числа.

Задачи геометрического программирования ставятся только на минимум:



Такие задачи чаще возникают в конструкторских разработках. Для них разработаны специальные методы.

Пример 8.1. Спроектировать открытый контейнер с прямоугольными стенками и днищем для перевозки из карьера гравия объемом V, если стоимость одной перевозки *С* не зависит от объема контейнера, а стоимость 1м2 днища равна *a*, передней и задней стенок – *b*, боковых стенок – *d*.

Пусть *x* – ширина, y – глубина и z – высота контейнера. Тогда целевая функция, суммарные затраты, запишется в виде позинома

.▲

*Кусочно-линейное программирование* включает специальные методы решения задач с кусочно-линейными функциями. В частности, такими являются функции

Рис. 8.2



 и 

если все *fi*(**X**) – линейные функции. Первая из них – выпуклая (рис. 8.2), вторая – вогнутая. Задачи с такими функциями могут входить в класс задач выпуклого программирования. Их решение строится на преобразовании модели к линейному виду с последующим применением методов ЛП.

К линейным сводятся также задачи *дробно-линейного программирования*. Они отличаются от линейных только дробной целевой функцией, числитель и знаменатель которой – линейные функции.

**8.2. Условия оптимальности**

Рассмотрим общую задачу нелинейного программирования



Примечание. В случае задачи классического типа имеют место ограничения в виде равенства.

**8.2.1. Простейшая задача НП.** Это задача классической оптимизации , которая дополнена неравенствами 

Пусть в соответствующей КлЗ на минимум найдено решение , которая может быть либо внутренней, либо граничной точкой. Тогда:

Если , то отклонения в обе стороны возможны, т.к. в оптимальной точке  Если же эта точка на границе (), отклонение возможно лишь в стороны увеличения функции Лагранжа, т.е. 

Обобщая, находим необходимые условия решения классической задачи

 (\*)

**8.2.2. Общий случай задачи НП.** В случае наличия в задаче оптимизации ограничений неравенств () введем дополнительную переменную такую,  так, чтобы , причем 

Мы получили задачу оптимизации в форме функции Лагранжа с переменными



Условия (\*) применимы и к переменным , а именно



(понятно, что для неравенства выполняется условие ).

Поскольку  имеем:





Эти условия называются **условиями Куна-Таккера – необходимые условия решения задачи нелинейной оптимизации**.

Примечание. В задачах выпуклого программирования условия Куна-Таккера являются не только необходимыми, но и достаточными.

Теорема. В задаче выпуклого программирования локальный оптимум является глобальным.

Функция выпукла на выпуклом множестве (если ), если . Напомним, ч то функция выпукла, если ее матрица Гессе положительно определенная и вогнута, если нечетные миноры этой матрицы отрицательны, а четные положительны. Например, функция  выпукла.

**Доказательство**. Пусть  выпукла на выпуклом множестве и пусть .- локальный максимум. Тогда для точки глобального максимума имеем 

Рассмотрим отрезок соединяющий точки локального и глобально максимума и выберем на этом отрезке произвольную точку  В силу вогнутости функции (решается задача максимум) имеем 

Усилим это неравенство, подставив справа функцию в точке локального максимума.



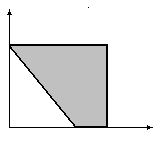
Что и требовалось доказать

Из полученного следует, **что  достигает минимума по Х и минимума по** , т.е. 

Эта точка называется **седловой.** Следовательно, решение задачи нелинейной оптимизации можно рассматривать как нахождение седловой точки соответствующей функции Лагранжа.

##### Рис. 8.6

*x*1



*x*2

# D

A

B

*D*

**8.6. Методы "спуска"**

Так называются **численные итерационные методы оптимизации, ориентированные на поиск минимума.**

Общая схема решения заключается в генерации последовательности прибли­жений, асимптотически сходящейся к минимуму. Как правило, это локальный минимум, но в случае задач выпуклого программирования он является и глобальным. За конечное число итераций методы позволяют получить приближенное решение с заданной наперед точностью. Сходимость метода, как и скорость сходимости, зависят от свойств задачи и начального приближения. Скорость сходимости во многом определяет эффективность метода. Основными показателями эффективности (скорости сходимости) являются число итераций и число вычислений функции при прочих равных условиях. Количество итераций для одной и той же задачи сильно зависит о начального приближения (начальной точки) и требуемой точности. При этом количество необходимых итераций растет гораздо быстрее, чем точность.

При выборе метода следует учитывать свойства целевой функции: унимодальность или многоэкстремальнсть, дифференцируемость, выпуклость-вогнутость или их отсутствие и т. д. Кроме того, функции могут обладать “неприятными” для метода особенностями, такими как седловые точки и овражность. “Овраг” (при максимизации “гребень”) проявляются в том, что вдоль него функция изменяется намного слабее, чем в поперечном направлении. На карте линий уровня он виден по сильной вытянутости линий вдоль “дна” оврага с одновременной “сплющенностью” в поперечнике. Самым простым примером такой функции является эллипсоид

.

При *a*>>*b* функция имеет овраг вдоль оси *x*1, которая является его “дном” (рис.8.7).

*x*2

*x*1

Рис. 8.7

Чем сильнее овражность и нелинейность “дна”, тем в большей степени замедляется скорость сходимости метода или возможен останов при недостижении окрестности минимума. Поэтому учет таких свойств особенно важен при выборе метода оптимизации.

Различают *методы безусловной оптимизации, применяемые для нахождения минимума без ограничений на переменные, и условной оптимизации, когда поиск производится при наличии ограничений*.

По информации, используемой для определения направления поиска, выделяют методы

* нулевого порядка или прямые, если вычисляется только значение целевой функции;
* первого порядка или градиентные, использующие первые производные (градиент);
* второго порядка, требующие вычисления также вторых производных;
* случайного поиска, применяющие механизм случайного выбора направления;
* генетические, сочетающие элементы детерминизма и случайности выбора;
* комбинированные.

При оптимизации *n*-мерных функций часто используются методы одномерной минимизации. Поэтому сначала рассмотрим эти методы, затем многомерные методы безусловной оптимизации и, наконец, методы поиска условного минимума. Все приводимые ниже методы предназначены для минимизации *унимодальных* функций.

**8.7. Методы одномерной минимизации**

**Методы нулевого порядка или прямые, если вычисляется только значение целевой функции**

**8.7.1.** ***Метод деления шага пополам***

Начальная точка и начальный шаг задаются на основе предварительных знаний о функции. Движение с неизменным шагом в одном направлении продолжается пока значение функции уменьшается. При увеличении значения направление меняется на противоположное и шаг уменьшается в два раза. Поиск заканчивается, когда при очередном шаге значение ухудшилось, а длина шага меньше заданной точности (рис. 8.8). Очевидно, что после прохождения минимума происходят колебания вокруг него с уменьшающейся амплитудой.

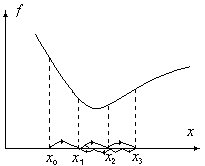


Рис. .8.8

*Алгоритм.*

1. Задать начальную точку *x*0, начальный шаг *h*0 и точность *ε* ; *k*=0.
2. Вычислить *f*(*xk*).
3. *xk*+1 = *xk*+*hk*.
4. Вычислить *f*(*xk*+1).
5. Если *f*(*xk*+1)< *f*(*xk*), то *hk*+1= *hk* и идти на 9.
6. Если *hk*<*ε*, перейти на 10. Конец.
7. Если *k*=0, то *hk*+1= - *hk* и идти на 9.
8. *hk*+1= - *hk*/2;
9. *k*= *k*+1и перейти на 3.

Проверка в п.7 необходима для того, чтобы не уменьшать шаг до достижения окрестности минимума. После окончания алгоритма в качестве оптимального *x\** может быть взято любое значение между *xk* и *xk*+1. При сильной изменчивости функции в окрестности минимума условие п.6 можно заменить на (*hk*<*ε*) & (*f*(*xk*+1) – *f*(*xk*)< *εf*), где*εf* – точность по функции.

**8.7.2.** ***Квадратичная аппроксимация***

Для нахождения приближенного минимума исходная функция аппроксимируется функцией второго порядка

, (8.35)

где *х*0 – точка отсчета реального диапазона значений *x*, в частности *х*0=0. Для определения коэффициентов, входящих в функцию (8.35), выбираются 3 точки и в них вычисляется значение функции *f*. Получается простая система 3-х уравнений с 3-мя неизвестными *a*, *b* и *c*:



Стационарная точка функции (8.35) вычисляется из равенства ее производной нулю:

. (8.36)

Она используется в условии останова и как новая точка для аппроксимации.

*Алгоритм.*

1. Задать первую точку *x*1, шаг Δ*х* и точность по координате *ε*и функции *εf.*
2. Определить 2-ю точку: *x*2= *x*1+Δ*х.*
3. Вычислить *f*(*x*1) и *f*(*x*2).
4. Если *f*(*x*2)<*f*(*x*1), принять *x*3= *x*2+Δ*х*, иначе *x*3= *x*1-Δ*х*.
5. Вычислить *f*(*x*3).
6. Вычислить *fm*=min{*f*(*x*1), *f*(*x*2), *f*(*x*3)} и определить точку *xm*, соответствующую *fm*.
7. По трем точкам и значениям функции в них найти коэффициенты в (8.35).
8. Вычислить *xa* по формуле (8.36).
9. Если (| *fm*-*f*(*xa*)|< *εf*) & (|*xm*-*xa* |< *ε*), окончить поиск.
10. Если *fm*<*f*(*xa*), взять *xm* и две ближайшие к ней точки, иначе взять *xa* и две ближайшие к ней точки. Выбранные точки пронумеровать в порядке возрастания значений, вычислить *f* в новой точке и перейти на 6.

**8.7.3. *Метод деления интервала пополам***

Этот метод, как и два следующих, относится к методам сокращения интервала неопределенности. Предполагается, что точка минимума находится на заданном интервале [*a*,*b*].

Рассматриваемый метод также называют трехточечным. Точки выбираются так, что делят интервал на 4 равные части (рис. 8.9):

Рис. 8.9



*xm*=(*a+b*)/2, *x*1=(*a+ xm*)/2, *x*2=(*xm*+*b*)/2.

Функция вычисляется в этих точках и после сравнения ее значений интервал сокращается в 2 раза. С новым интервалом действия повторяются.

*Алгоритм.*

1. Задать точность по координате *ε*.
2. Вычислить *xm* и *f*(*xm*).
3. Вычислить *x*1 и *x*2.
4. Вычислить *f*(*x*) в этих точках.
5. Если *f*(*x*1) < *f*(*xm*) , то принять *b* = *xm*, *xm*= *x*1. Иначе, если *f*(*x*2) < *f*(*xm*), положить *a= xm*, *xm*= *x*2 ,иначе – *a= x*1, *b* = *x*2.
6. Если *b - a*<*ε,* закончить поиск, иначе перейти на 3.

Нетрудно подсчитать, что после *n* вычислений функции интервал неопределенности уменьшится в  раз. Доказано, что этот метод эффективнее других прямых методов, использующих равноудаленные точки.

**8.7.4. *Метод золотого сечения***

Золотое сечение – это определенное отношение части к целому. Отрезок АВ делится точкой С в отношении золотого сечения (рис. 8.10), если

. (8.37)



Рис. 8.10

Положим АВ = 1, АС = *х*, СВ = 1 – *х*, тогда из (8.37) получаем уравнение

*х*2 + *х* – 1 = 0,

из которого следует

, .

Эти отношения используются для выбора двух точек внутри интервала неопределенности. Они располагаются, как показано на рис. 8.11. Каждая из точек делит интервал [*a*, *b*] в отношении золотого сечения.

В этих точках вычисляется функция. Если *f*(*x*1) > *f*(*x*2), то отбрасывается часть интервала [*a*,*x*1], если *f*(*x*1) < *f*(*x*2), то отсекается часть [*x*2,*b*], а при равенстве значений функции – любая из них. Оставшаяся часть интервала равна  от величины исходного. Очевидно, что после такого сокращения интервала одна из внутренних точек остается с изменением индекса, а вторая берется на основе золотого сечения или, что одно и то же, симметрично оставшейся (рис. 8.12). Сокращение интервала продолжается до достижения заданной точности.

*b*

*a*





*x*

2

*x*

1

0.382

0.618





Рис. 8.11

*Алгоритм*.

1. Задать точность по координате *ε*.
2. Вычислить  
3. Вычислить *f*( *x*1), *f*( *x*2).

*x*2

*x*1

*x*2

*x*2

*x*1

*x*1

##### Рис. 8.12

1. Если *f*(*x*1)>*f*(*x*2), положить *a=x*1, *x*1=*x*2,  или *x*2=*a+b-x*1, иначе – *b=x*2, *x*2= *x*1,  или *x*1= *a+b-x*2.
2. Если (*b-a*)< *ε*, закончить поиск.
3. Вычислить функцию в новой точке и перейти на 4.▲

Итерации алгоритма графически иллюстрируются на рис. 8.12.

Покажем, что сохраняемая точка (*x*1 или *x*2) делит сокращенный интервал также в отношении золотого сечения. Пусть на *k*-й итерации внутренние точки делят интервал [*ak*, *bk*] в отношении золотого сечения. Обозначив = *bk - ak*,  имеем

.

Тогда для нового, сокращенного, интервала находим

,



В результате получаем:

.

Благодаря этому свойству, внутренние точки не сливаются при любом числе итераций.

Согласно алгоритму функция вычисляется 2 раза на начальном интервале и по одному разу на всех последующих. Поэтому после *n* вычислений функции интервал неопределенности составит  от величины первоначального. При заданной точности можно найти необходимое количество вычислений функции *n* из условия



**Методы первого порядка или градиентные, использующие первые производные (градиент)**

**8.7.6. *Метод первого порядка***

Он применим к дифференцируемым функциям в случаях, когда первая производная может вычисляться с высокой точностью. Рассматриваемый метод средней точки алгоритмически является самым простым из методов одномерной оптимизации на заданном интервале. Он заключается в определении средней точки интервала *c=*(*a+b*)*/*2 и вычислении в ней производной.

Если *f '*(*c*) < 0, полагают *а* = *с*, при *f '*(*c*) > 0, принимают *b* = *с*. Условие окончания поиска учитывает точность по координате *ε*  и по производной функции *ε*1:

((*b-a)< ε* )&(| *f '*(*c*) | < *ε*1).

Эффективность метода зависит от трудоемкости вычисления производной. Задаваемая точность *ε*1 не должна быть выше точности вычисления производной. С ростом трудоемкости и снижением точности вычислений производной следует отдавать предпочтение прямым методам.

**8.7.7. *Методы второго порядка***

Методы применимы к дважды дифференцируемым функциям. При этом предъявляются высокие требования к точности вычисления производных. Наиболее удачными считаются задачи, в которых известно аналитическое выражение первой производной, а еще лучше – и второй.

Здесь приводится только один из наиболее часто используемых методов – метод Ньютона – Рафсона.

Он основан на линейной аппроксимации первой производной минимизируемой функции *f* в окрестности текущей точки *xk*:



В стационарной точке аппроксимации, которая принимается за очередное приближение *xk+*1, производная равна нулю:

.

Отсюда следует рекуррентная формула для построения последовательности приближений к искомому минимуму:

 (8.41)

Очевидно, что применение (8.41) возможно только при условии, что  для каждого *k*. Поиск завершается по условию достижения точности, заданной величиной первой производной | *f '*(*xk*) | < *ε*1 или расстоянием между двумя точками |*xk+*1-*xk* | < *ε*. Возможно одновременное использование этих условий.

В общем случае процедура (8.41) не гарантирует сходимость к стационарной точке. Если начальная точка достаточно близка к стационарной, то метод сходится. При сходимости обеспечивается высокая скорость приближения к минимуму. На рис. 8.14 приведен случай сходимости метода. Очередному приближению соответствует точка пересечения оси *x* аппроксимирующей прямой. Как видно, последовательность точек *x*0,*x*1,*x*2,… приближается к минимуму *x\** .

Для некоторых функций результат поиска зависит от выбора начальной точки. Так, например, при начальной точке, взятой правее максимуму производной, так как показано на рис. 8.15, метод расходится.



##### Рис. 8.14

**8.8. Многомерный поиск безусловного минимума**

Здесь рассматриваются методы прямого поиска (разд. 8.8.1, 8.8.2 и 8.8.3), градиентные методы (разд.8.8.4), метод второго порядка, алгоритмы случайного поиска и приводятся краткие сведения о других методах.

**8.8.1. *Метод Гаусса-Зейделя (покоординатного спуска)***

Процедура поиска сводится к решению последовательности задач одномерной минимизации по каждой переменной. Пусть выбрана начальная точка

.

Зафиксируем все переменные, кроме первой, на начальных значениях и решаем задачу



одним из одномерных методов. Фиксируем *х*1 на полученном в решении значении x1' и делаем свободной переменную *х*2.Приходим к очередной одномерной задаче

.

Аналогично строятся и решаются последующие одномерные задачи, последняя из которых имеет вид:

.

Эти *n* задач составляют один цикл. Его результатом является точка X1. Она принимается за начальную точку для следующего аналогичного цикла. Поиск заканчивается, когда расстояние между двумя последовательными точками становится меньше заданной величины:

.

Работу метода иллюстрирует рис. 8.16, на котором показана траектория поиска минимума функции *f=*(2-*x*1)4+2(*x*1-2*x*2)2.

Метод отличается алгоритмической простотой. Однако ему присущ ряд недостатков. Его эффективность существенно зависит от направления осей координат относительно линий уровня. Это хорошо видно на примере квадратичной функции: при совпадении координат с осями эллипсов минимум достигается за один цикл из любой начальной точки (рис. 8.17), а при их повороте число циклов значительно возрастает (рис. 8.18). Из этого примера также следует, что метод неэффективен в условиях оврага. Если функция не дифференцируема в отдельных точках, поиск может остановиться, не достигнув окрестности минимума. Рис. 8.19 демонстрирует такой случай: точка останова А далека от искомого минимума.

А

Рис. 8.19

Рис. 8.18

Рис. 8.17

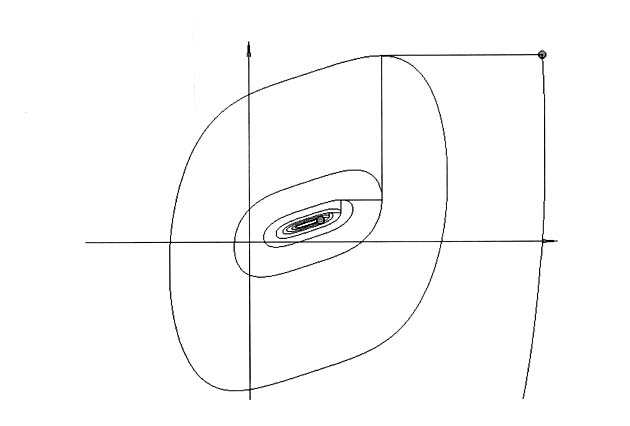


Рис. 8.16

*x2*

Из анализа траекторий поиска в приведенных примерах можно заключить, что эффективность поиска повысится, если к описанным однотипным циклам добавить движение в направлении, проходящем через точки X(*k*) и X(*k*+1). Это движение называют ускоряющим шагом. Он используется в методе, рассматриваемом в следующем разделе.

**8.8.4. *Градиентные методы***

Если функция дифференцируема, вычисление производных дает информацию о поведении функции в исследуемой точке и, следовательно, позволяет находить лучшее направление поиска..

Скорость изменения функции *f*(X) в произвольном направлении *l* определяется производной

. (8.43)

Здесь частные производные  представляют собой направляющие косинусы. Геометрическая иллюстрация для случая функции двух переменных приведена на рис. 8.28. Очевидно, что

.



Рис. 8.28

Поверхность уровня целевой функции, описываемая уравнением *f*(X) = *const*, имеет размерность *n* – 1.

Зададим координаты следующим образом. Проведем через рассматриваемую точку *n*– 1 взаимно ортогональные касательные к поверхности уровня, приняв их за оси координат. В качестве последней оси возьмем нормаль к поверхности уровня (рис. 8.29).

В такой системе координат производные функции по всем *xj* равны нулю. Поэтому из (8.43) имеем



Рис. 8.29

.

Отсюда следует, что максимальная скорость увеличения функции будет в направлении *l*, совпадающем с нормалью. Вектор, имеющий направление нормали к поверхности функции в данной точке и длину , называется *градиентом*.

Обозначается градиент как grad *f*(X) или ∇*f*(X). Он полностью определяется своими проекциями – производными функции по переменным:



В задачах на минимум используется противоположное направление –  *антиградиент*.

Значения производных могут быть найдены по приближенным формулам:

,

.

Более точный результат дает вторая формула, но она требует больше вычислений. Чем точнее необходимо вычислить производную, тем меньше должно быть Δ*х*. Однако недопустимо, чтобы разность значений функции была соизмерима с погрешностью вычисления.

Если переменные имеют разные единицы измерения, можно перейти к относительным переменным *yi*, используя минимально и максимально возможные значения переменных *xi*:

.

Очевидно, что значения *yi* лежат в диапазоне [0, 1].

Знание градиента (антиградиента) позволяет осуществлять перемещение из текущей точки в направлении наибольшего улучшения функции. Для линейных функций оно постоянно и поэтому его требуется определять всего один раз. В нелинейных функциях значение градиента зависит от вектора X, то есть его необходимо пересчитывать при любом изменении X.

В градиентном методе поиск минимума производится согласно рекуррентным формулам:

****

****, (8.44)

где *hk* - шаг. В первой формуле величина изменения переменных ΔX зависит как от шага, так и от величины градиента. Более удобно, когда расстояние между последовательными точками зависит только от величины шага. Поэтому предпочтительнее вторая формула. В ней используется нормированный градиент, длина которого всегда равна единице (за счет деления на длину). Поэтому он задает лишь направление, но не влияет на величину ΔX.

Формула (8.44) в записи по отдельным переменным принимает вид:

. (8.45)

*Алгоритм* градиентного метода.

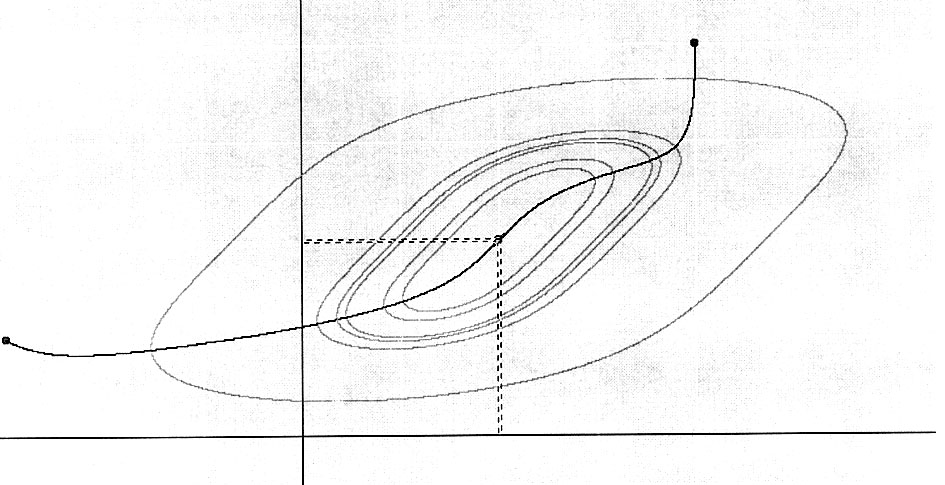
1. Задать начальную точку, начальное значение шага и точность по величине градиента *ε*, вычислить *f*(X0) и положить *k*= 0.
2. В текущей точке X*k* вычислить градиент (производные  и его длину.
3. Проверить: если  закончить поиск.
4. Определить новую точку X*k*+1 согласно (8.45) и вычислить в ней значение функции.
5. Проверить: если *f*(X*k+*1)≤ *f*(X*k*), увеличить *k* на единицу и перейти на шаг 2; если иначе, уменьшить *hk* в два раза и перейти на шаг 4.▲

При гладкой производной траектория поиска в идеале (при непрерывном вычислении градиента) тоже будет гладкой и ортогональной линиям уровня. При конечной величине шага траектория становится кусочно-линейной. Качественный характер поиска показан на рис. 8.30, а на рис. 8.31 приведена реализация алгоритма с двух начальных точек применительно к функции *f*(X)=(*x*1-*x*2)2+(*x*2-2)4. Обе траектории приводят практически в одну точку.



Рис. 8.30

Самой трудоемкой частью алгоритма является вычисление градиента, а он вычисляется после каждого успешного шага. В методе *наискорейшего спуска*, который представляет собой модификацию градиентного метода, градиент определяется реже, особенно в начальный период поиска.



*x*1

*x*2

##### Рис.8.31

Модификация заключается в том, что после вычисления градиента вместо одного дискретного шага ищется минимум на направлении антиградиента, то есть проводится одномерный поиск по *h*:

 (8.46)

В результате решения задачи (8.46) определяется оптимальный шаг *h*\*, по которому находится следующая точка:

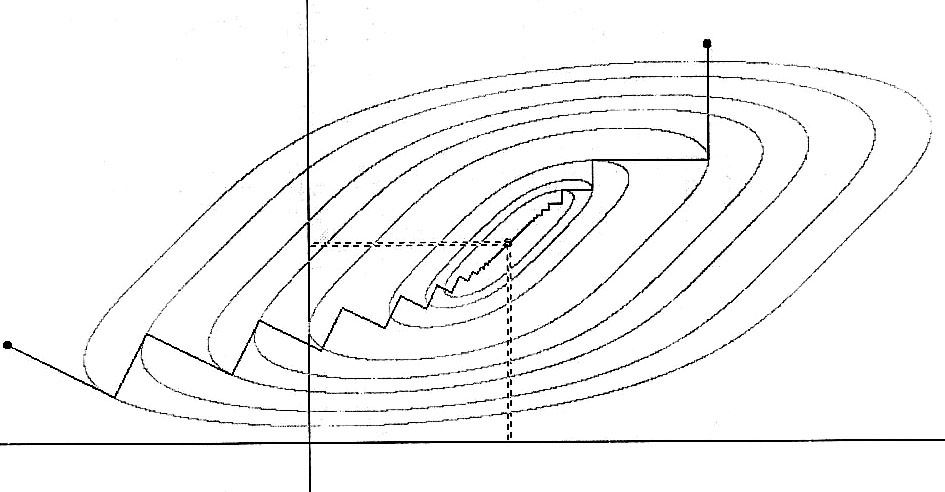
 (8.47)

Очевидно, что при таком определении новой точки, значение функции в ней будет лучше, чем в X*k*.

*Алгоритм* наискорейшего спуска.

1. Задать начальную точку и точность по величине градиента *ε*, положить *k*= 0.
2. В текущей точке X*k* вычислить градиент (производные  и его длину.
3. Проверить: если  закончить поиск.
4. Провести одномерный поиск (8.46).
5. Определить новую точку X*k*+1 согласно (8.47), увеличить *k* на единицу и перейти на шаг 2. ▲

На рис. 8.32 показаны две траектории движения к минимуму функции *f*(X)=(*x*1-*x*2)2+(*x*2-2)4, полученные алгоритмом. Минимум на направлении антиградиента достигается в точке касания с линией уровня, а градиент в этой точке ортоганален ей. Поэтому каждое последующее направление ортогонально непосредственно предшест­вую­щему. Из рис. 8.32 видно, что с приближением к экстремуму частота вычисления градиента увеличивается, и вблизи минимума метод наискорейшего спуска вырождается в градиентный (см. рис. 8.31 и 8.32).

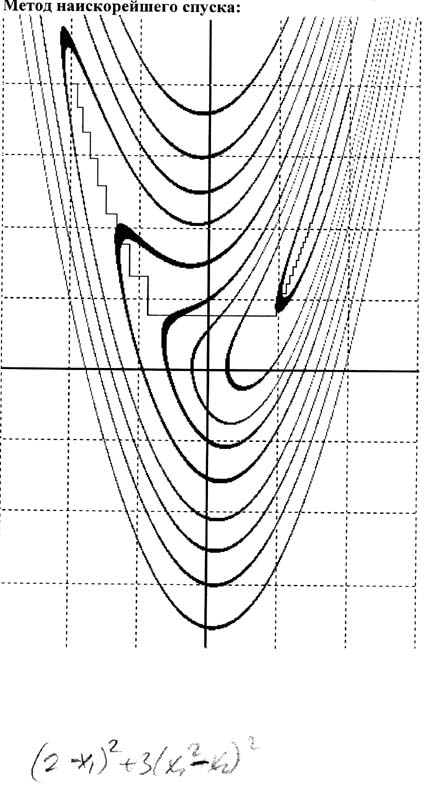


*x*1

*x*2

Рис. 8.32

Градиентные методы плохо работают в условиях оврага: при попадании на дно оврага резко падает скорость движения и возможна остановка поиска до достижения окрестности минимума (рис. 8.33, *f*=(2-*x*1)2+3(*x*12-*x*2 )2).



##### Рис. 8.33

**8.8.5. *Метод Ньютона***

Это классический метод второго порядка. Если направление градиента находится из линейного представления функции в окрестности текущей точке, то в методе Ньютона используется квадратичная аппроксимация функции цели.

Квадратичная аппроксимация дифферен­цируемой функции *f* в точке X*k* записывается в виде

, (8.48)

где Н – матрица вторых производных функции  *f* (матрица Гессе). В стационарной точке градиент функции равен нулю, что применительно к (8.48) дает

. (8.49)

Полагая матрицу H неособенной (существует обратная к ней матрица), из (8.49) получаем рекуррентный способ генерирования последовательности точек

. (8.50)

Исходя из вывода формулы (8.50) ясно, что для квадратичной функции цели X*k*+1 является точкой минимума. Следовательно, минимум такой функции из любой начальной точки достигается за один шаг (8.50). Для нелинейной унимодальной функции, отличной от квадратичной, точки последовательности (8.50) асимтотически приближаются к минимуму. Условие окончания поиска такое же, как и в градиентных методах: . (В случае линейной аппроксимации матрица Н становится единичной и поиск вырождается в градиентный).

Необходимым условием сходимости метода является существование обратной матрицы во всех генерируемых точках. Доказательство сходимости метода получено при условии, что начальная точка достаточно близка к точке минимума. При этом метод имеет высокую скорость сходимости. Если поиск начинается с удаленной точки и функция существенно отличается от квадратичной, метод может не сходиться или давать слабую сходимость. Причина такого поведения в том, что значение функции в точке X*k+*1 не обязательно меньше, чем в точке X*k*. Так на рис. 8.34 траектория поиска имеет зигзагообразный характер из-за того, что после 2-й итерации значение функции ухудшилось.

**8.8.7. *Методы случайного поиска***

Рассматриваемые здесь методы основаны на использовании случайного механизма задания начальной точки и выбора направления движения. Так как в процессе поиска вычисляются значения только целевой функции, эти методы можно отнести к классу прямых.

Случайный механизм выбора направления реализуется с помощью датчика случайных чисел *β*, равномерно распределенных на интервале [-*b*, *b*]. Направление задается случайным вектором

Ξ = (*ξ*1, *ξ*2, *ξ*3, ..., *ξ*n),

компоненты которого вычисляются по формуле

,

где *n*  случайных чисел *βi* генерируются датчиком. Очевидно, что такой случайный вектор имеет единичную длину и определяет только направление. При этом все направления равновероятны.

Приведем несколько простых алгоритмов случайного поиска.

***Алгоритм с возвратом при неудачном шаге***

1. Задать начальный шаг *h*0, число проб *s*≤ *n* и точность *ε.*
2. Сгенерировать или задать начальную точку X0 и вычислить в ней функцию *f*; положить *i=*1 (*i* – счетчик проб).
3. Сгенерировать случайный вектор направления ⋅.
4. На направлении определить точку X*k*+1 = X*k* + *hk*⋅Ξ*k* и вычислить в ней функцию *f*.
5. Проверить: а) если *f*(X*k*+1)<*f*(X*k*), положить *k*=*k+*1, *i=*1 и вернуться на 3; б) если *f*(X*k*+1)≥*f*(X*k*) и *i*<*s*, положить *i*= *i+*1 и вернуться на 3; в) если *f*(X*k*+1)≥*f*(X*k*), *i*=*s* и *hk* >*ε*, положить *hk*= *hk*/2, *i=*1 и вернуться на 3. г) если *f*(X*k*+1)≥*f*(X*k*), *i*=*s* и *hk*≤*ε*, поиск закончить, приняв X*k* за точку минимума.▲

Таким образом, поиск останавливается, если в текущей точке *s* направлений, сгенерированных подряд, оказались неудачными при шаге, меньшем заданной точности. На рис. 8.38 показан характер движения при поиске по данному алгоритму (жирной линией выделены успешные шаги).

**.**

**.**

**X0**

##### Рис. 8.38

***Алгоритм с обратным шагом***

Он построен путем модификации предыдущего алгоритма с целью уменьшить число розыгрышей направлений. Внесено следующее изменение: если сгенерированное направление неудачное, то после возврата в исходную точку X*k* делается шаг в противоположном направлении; если же и оно неудачное, то генерируется новое направление из точки X*k*. Логика действия очевидна: если в данном направлении функция ухудшается, то можно ожидать, что в противоположном направлении она улучшится. Понятно, что эффект модернизации алгоритма тем выше, чем ближе функция в текущей области к линейной.

**8.9. Методы условной оптимизации**

В общем случае задача может содержать ограничения в виде равенств и неравенств, которые и определяют допустимое множество *D*. Наличие ограничений существенно усложняет решение задачи оптимизации. В большинстве случаев условный оптимум достигается на границе допустимого множества, при этом даже унимодальная функция может иметь на границе несколько локальных экстремумов.

Из большого числа методов условной оптимизации можно выделить 3 основные группы:

* методы *возможных направлений*: метод проектирования градиента, методы Зойтендейка, Вулфа и др.;
* методы *штрафных и барьерных функций*;
* модификации методов безусловной оптимизации.

Методы первой группы отличаются способом определения возможных направлений. Направление **d** в точке **x***k*∈*D* называется *возможным направлением*, если существует *λ≠*0, при котором



Эти условия означают, что на направлении **d** найдутся *допустимые* точки, в которых значение функции лучше, чем в точке **x***k*.

Ниже рассматривается один из методов возможных направлений.

**8.9.1. *Метод проектирования градиента***

Градиент дает направление, в котором функция возрастает с наибольшей скоростью. Однако при условной оптимизации оно, как правило, не является возможным направлением. Поэтому используют не сам градиент (антиградиент), а его проекцию на поверхность ограничений, точнее, на плоскость, аппроксимирующую эту поверхность в текущей точке. Очевидно, что проекция градиента определяет направление наискорейшего изменения функции на поверхности ограничений.

Приведем один из вариантов метода проектирования градиента сначала для задачи с ограничениями-равенствами, а затем для общего случая. Метод применим, если целевая функция и все функции ограничений дифференцируемы.

Пусть ограничения заданы в виде

 (8.55)

Найдем возможное направление *l*, на котором скорость изменения целевой функции максимальна:

 (8.56)

В допустимой области *D* функции *ψj* постоянны. Поэтому искомое направление должно удовлетворять системе равенств

 (8.57)

Из связи направления с координатами следует:



что перепишем в виде

 (8.58)

Таким образом, для нахождения наилучшего возможного направления необходимо решить задачу оптимизации (8.56) – (8.58). Так как условия имеют вид равенств, а функции дифференцируемы, для решения этой вспомогательной задачи воспользуемся методом Лагранжа.

Запишем функцию Лагранжа:

. (8.59)

Неизвестными в ней являются векторы  и **Λ**. Взяв производные  и приравняв их нулю, получаем



Отсюда выразим компоненты искомого вектора:

 (8.60)

Подставив (8.60) в (8.58), находим:



С учетом этого выражения формула (8.60) принимает вид

 (8.61)

Для определения неизвестных множителей *λj* воспользуемся системой (8.57). Подставив в нее (8.61), получаем:



После раскрытия скобок окончательно имеем:

 (8.62)

Так как направление ищется в конкретной точке, то все производные в (8.62) – известные константы. Поэтому система уравнений (8.62) является линейной системой относительно *λj*. Ее решение не вызывает затруднений (при условии, что матрица системы не является особенной). Найдя значения *λj* и подставив их в (8.61), получаем компоненты проекции градиента (в знаменателе (8.61) имеем длину проекции градиента). Движение осуществляется в направлении, противоположном проекции.

Аналогично градиентному методу новая точка вычисляется по формуле

. (8.63)

*Алгоритм*.

1. Задать начальную точку, удовлетворяющую системе (8.55), начальное значение *h0* и точность по величине проекции градиента *ε*.
2. В текущей точке вычислить градиенты всех функций (*f* и *ψj*) и решить систему (8.62).
3. Вычислить проекцию градиента по формуле (8.61).
4. Проверить: если  завершить поиск.
5. Вычислить новую точку по формуле (8.63).
6. Проверить: если значение целевой функции улучшилось, перейти на шаг 2, иначе уменьшить *hk* в два раза и перейти на шаг 5.▲

Качественный характер работы алгоритма иллюстрирует рис. 8.39. Здесь функции зависят от 2-х переменных, поэтому в каждой точке на линии ограничения может быть всего 2 направления, лучшее из которых определяет проекция градиента. В многомерной задаче таких направлений бесконечное множество.

При линейных ограничениях могут возникать проблемы поиска лишь при очень малых значениях градиентов функций ограничений и совпадении их направлений, так как это приводит к вырожденности матрицы системы (8.62).

-∇*f*(X0)

X0

X\*

##### Рис. 8.39

*ψ=*0

В случае нелинейных ограничений движение на основе линейной аппроксимации нарушает равенства. Поэтому в алгоритм необходимо внести изменения, обеспечивающие выполнение равенств с необходимой точностью *εψ*. С этой целью алгоритм дополняется проверкой величины невязки, которая выполняется в каждой новой точке. Если , то включается процедура коррекции, заключающаяся в минимизации невязки:

 (8.64)

Для решения задачи (8.64) можно применить любой метод безусловной оптимизации, но в данном контексте целесообразен метод градиентов. При этом значения (*n*-*m*) переменных фиксируются, так как для выполнения равенств достаточно *m* переменных. Понятно, что при частых коррекциях трудоемкость алгоритма значительно возрастает.

Теперь рассмотрим случай, когда ограничения заданы в виде неравенств *ϕj*(**x**)≤0. Поиск начинается из точки, в которой удовлетворяются все ограничения. Пока они выполняются как строгие, движение осуществляется по градиентному методу. Когда достигается граница допустимого множества, одно или несколько ограничений обращаются в равенства. Такие ограничения называют активными. В точках с активными ограничениями направление движения определяется по проекции градиента на поверхность этих ограничений, то есть применяется вышеприведенный алгоритм. Пример поиска минимума при двух линейных неравенствах показан на рис. 8.40. Очевидно, что при смешанных и нелинейных ограничениях алгоритм весьма существенно усложняется и требует большого объема вычислений. Поэтому метод проектирования градиентов целесообразно применять при линейных ограничениях.

**8.9.2. *Метод штрафных функций***



##### Рис. 8.40

Совершенно иной подход используется в методах штрафных и барьерных функций. Ограничения задачи специальным образом отражаются в критерии, в результате чего критерий модифицируется, а исходная задача на условный экстремум сводится к задаче на безусловный экстремум.

В методе штрафных функций в критерий вводится штраф при нарушении условий задачи. Пусть в общем случае имеем задачу

*f*(**x**) 🡪 min; (8.65)

*ϕi*(**x**) ≤ 0, ; (8.66)

*ψi*(**x**) = 0 , . (8.67)

Тогда можно построить вспомогательную функцию

*Θ*(**x**) = *f*(**x**) + *α*⋅*H*(**x**), (8.68)

где *H*(**x**)–функция штрафа, *α* - параметр штрафа.

Вспомогательная функция играет роль модифицированного критерия, который при выполнении всех ограничений должен совпадать с исходным. Поэтому необходимо, чтобы в допустимой области *Н*(**х**) равнялась нулю, а вне ее была положительной. Для задачи (8.65)-(8.67) функция штрафа включает две составляющие *Н*(**х**) =*Нϕ*(**x**) + *Нψ*(**x**), учитывающие ограничения-неравенства и ограничения-равенства соответственно и удовлетворяющие условиям

 (8.69)

Возможны разные конструкции функций, обладающих указанными свойствами. Типичные представители составляющих штрафной функции имеют вид



где *р* – натуральное число. Для дифференцируемости функций берут четные значения *р*, обычно *р* = 2.

Чем больше *α*, тем сильнее влияет функция штрафа и, значит, тем точнее выполняются условия задачи.

Пример 8.8. Дана задача

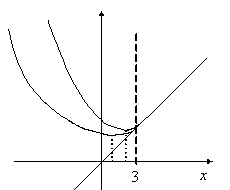
*f*(*x*) = *x* 🡪 min;

*ϕ*(*x*)=3 – *x* ≤ 0.

Ответ очевиден: *x*\*=3. Теперь сведем эту задачу к определению безусловного экстремума вспомогательной функции. Построим штрафную функцию в соответствии с (8.69): *H* = [max (0, 3-x)]2. Тогда приходим к задаче

*Θ*=*x*+*α*[max (0, 3-x)]2→ min.

На рис. 8.41 и 8.42 показаны функции *αH* и *Θ* для двух значений *α*. Видно, что точки минимума вспомогательной функции с увеличением *α* приближаются к точке условного минимума исходной задачи. Такой же вывод следует из аналитического решения. Действительно, при *x*<3 вспомогательная функция имеет вид



*f*

Θ

*α*1

*α*2

*f*,*θ*

##### Рис. 8.42



##### Рис. 8.41

*Θ* = *x* + *α*⋅ (3 - *x*)2.

Находим минимум этой функции:



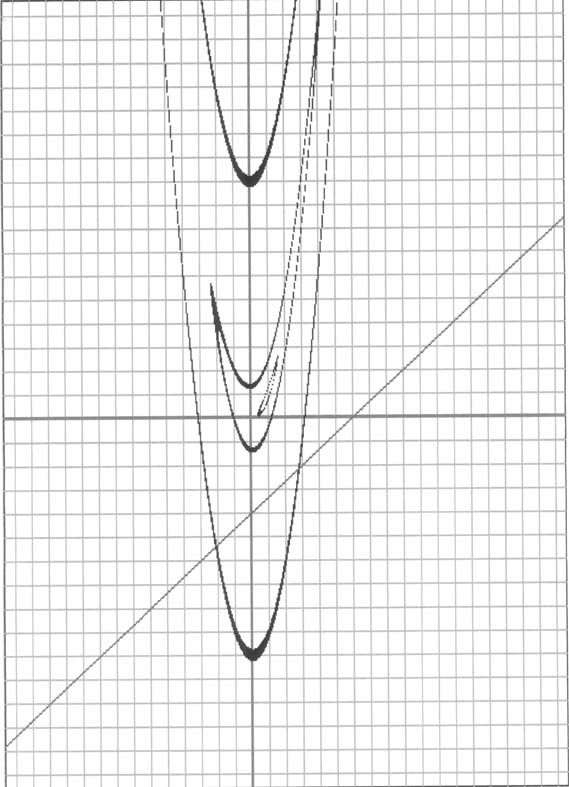
Отсюда получаем 

Пример 8.9. Рассмотрим влияние параметра шага в задаче



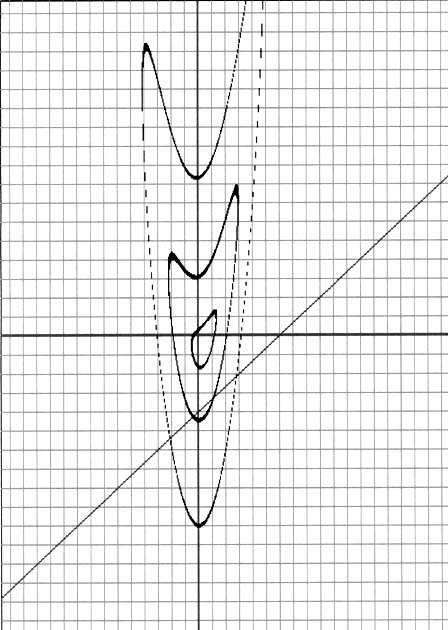
Здесь  и На рис. 8.43 построены линии уровня функции *θ* для разных значений *α* и линия ограничения *ψ*. При *α*=0 имеем *θ*=*f*, и минимум *θ* достигается в точке безусловного минимума *f*: *x*1=*x*2=1. С увеличением *α* меняется форма линий уровня *θ* и положение минимума. При *α*=1 минимум *θ* смещается к линии ограничения, а при *α*=1000 он практически точно совпадает с условным минимумом задачи.

В обоих примерах с увеличением *α* генерируемые точки приближаются к оптимальному решению извне допустимого множества. Поэтому ряд авторов называют рассматриваемый метод методом *внешних штрафов*.

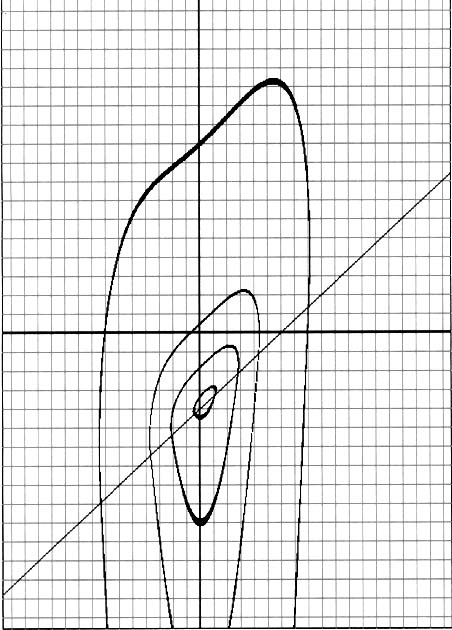


##### α*=*0

##### *ψ*=0



##### α=1



##### *α*=1000

Рис. 8.43

Таким образом, чтобы безусловный минимум вспомогательной функции был близок к условному минимуму решаемой задачи, необходимо брать очень большое значение *α*, теоретически бесконечное. Однако при больших *α* возникают серьезные трудности при поиске минимума вспомогательной функции. Поэтому предлагается решать последовательность задач минимизации *Θ* с возрастающими значениями *α*. При этом в качестве начальной точки следующей задачи берется оптимальная точка предыдущей. Такой прием использован в следующем алгоритме штрафных функций.

*Алгоритм*.

1. Задать: начальную точку **x**0, точность *ε*, начальное значение *α*0 и число *β* > 1.

2. Минимизировать *Θ*(**x**) одним из методов безусловной оптимизации, в результате чего определяется .

3. Проверить: если , то остановиться, приняв  за оптимальное решение задачи.

1. Положить , за начальную точку принять  и вернуться на шаг 2.▲

Рекомендуется выбирать значения параметров алгоритма из диапазонов: *α*0∈(0,1], *β*∈(1,10]. Начальную точку следует задавать в недопустимой области.

Пример 8.10. Алгоритмом штрафных функций решить задачу



Возьмем начальную точку **x**0=(-5;5), *α*0=0,21, *β*=5 и *ε*=0,0001. Применяя для минимизации *Θ* метод Ньютона, получаем результаты, представленные в табл. 8.4.

Таблица 8.4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № итерации | *α* | *x*1 | *x*2 | *f* | *Θ* | *αH* |
| 0 | 0.21 | -5 | 5 | 270 | 283.533 | 13.533 |
| 1 | 1.05 | -0.191 | -0.132 | -0.094 | 0.939 | 1.032 |
| 2 | 5.25 | -0.209 | -0.169 | -0.09 | 5.035 | 5.125 |
| 3 | 26.25 | -0.654553 | -1.059105 | 1.651353 | 13.504372 | 11.853019 |
| 4 | 131.25 | -0.990765 | -1.731532 | 5.068137 | 7.691651 | 2.623514 |
| 5 | 656.25 | -1.046856 | -1.843717 | 5.814225 | 6.343889 | 0.529664 |
| 6 | 3281.25 | -1.057736 | -1.865478 | 5.964774 | 6.070887 | 0.106113 |
| 7 | 16406.25 | -1.059899 | -1.869804 | 5.994933 | 6.016163 | 0.02123 |
| 8 | 82031.25 | -1.060331 | -1.870668 | 6.000967 | 6.005213 | 0.004246 |
| 9 | 410156.25 | -1.060417 | -1.870841 | 6.002174 | 6.003023 | 0.000849 |
| 10 | 2050781.25 | -1.060434 | -1.870876 | 6.002415 | 6.002585 | 0.00017 |
| 11 | >107 | -1.060434 | -1.870884 | 6.002469 | 6.002503 | 3.397E-05 |