**Методы оптимизации**

*Оптимизация* – выбор наилучшего решения.

Сложность или невозможность отыскания аналитического решения привело к тому, что постепенно стало ясно, что любая задача может считаться решенной, если *указан алгоритм, позволяющий численно построить приближенное решение с требуемой точностью*.

*Математическая теория оптимизации* включает в себя фундаментальные результаты и численные методы, позволяющие находить наилучший вариант из множества возможных альтернатив *без их полного перебора* и сравнения.

Итак, есть варианты решения задачи, среди которых надо найти лучший. Как оценить какой метод лучше? Надо определить критерий качества.

*Критерий качества* – функционал, действующий из множества вариантов решения задачи в множество вещественных чисел. Тогда понятие хуже - лучше тождественно больше - меньше. Один вариант лучше другого, если, например, значение функционала меньше, и неопределенность теряется.

У одной и той же задачи часто бывает возможным наличие нескольких функционалов качества. При этом нахождение их экстремума оказывается сложным, и выбраться из этой ситуации можно за счет методов многокритериальной оптимизации.

В этом курсе мы будем заниматься поиском экстремума одного функционала.

Итак, рассмотрим *функционал* *ϕ:* *X*→ *R̅*, где

*X* – множество вариантов или допустимое множество (область определения функционала);

 – расширенная вещественная прямая.

Пусть *c* ⊂ *X* – некоторое подмножество *X***.**

Задача:  называется *экстремальной задачей с ограничением* *c*. (экстремум – максимум или минимум)

**Терминология и классификация**

*C ≠ X*

*C = X*

Минимизация функции (безусловная минимизация)

Математическое программирование (условная минимизация)

*X = Rn*

*Методы оптимизации*

(исследование операций)

[более широкий термин]

Экстремальные задачи

Вариационное исчисление

Оптимальное управление

*X* – множество функций

*X* [более сложная структура]

**Этапы решения оптимизационной задачи**

Процесс принятия решения в исследовании операций представляет собой сложный процесс, который условно можно разбить на 4 этапа:

***1 этап***: Построение качественной модели рассматриваемой проблемы, т.е. выделение факторов, которые представляются наиболее важными, установление закономерностей, которым они подчиняются.

***2 этап***: Построение математической модели, включающей в себя выбор функционала *ϕ* (или целевой функции переменных), *ϕ*(*x*) → min(max), формирование ограничений (условий) в виде равенств или неравенств, например

.

Этот этап требует привлечения математических знаний и характеризуется, как правило, большим количеством переменных (*n* и *m* – велики).

***3 этап***: Решение математической задачи – выбор метода, реализация его и получение результата (применение ЭВМ, разработка программ, применение существующих СП и т.д.).

***4 этап***: Анализ полученного результата. Выясняется степень адекватности модели (результаты вычислений) и моделируемого объекта (имитационные данные).

**Примеры математических моделей**

Вообще, теория математических моделей является предметом специализированного курса и требует знаний в той области, которой принадлежит моделируемый объект. Рассмотрим традиционные примеры, иллюстрирующие применение метода математического моделирования в задачах экономического содержания.

*Задача о рационе*: Пусть имеется *n* – число продуктов питания и *m* – число питательных веществ.

Пусть  – содержание j-го вещества в единице i-го продукта;

 – минимальная (суточная) потребность (человека) в j-ом веществе;

 – стоимость единицы i-го продукта;

 – искомое количество (суточное потребление) i-го продукта.

Тогда  – общее содержание j-го питательного вещества;

 – стоимость (суточного) рациона.

|  |  |
| --- | --- |
| *Задача*:  найти min  (или тот набор, (*x*1,…,*xn*) при котором достигается минимум)  при условии, что | Типичная задача линейного программирования |

*Транспортная задача:* Требуется составить план перевозок однородного груза таким образом, чтобы общая стоимость перевозок была минимальной.

Пусть  – количество единиц груза в i-ом пункте отправления ();

 – потребность в j-ом пункте назначения () в единицах груза;

 – стоимость перевозки единицы груза из i-го пункта в j-ый;

 – планируемое количество единиц груза для перевозки из i-го пункта в j-ый.

Тогда  – общая (суммарная) стоимость перевозок;

 – количество груза, вывозимого из i-го пункта;

 – количество груза, доставляемого в j-ый пункт.

В простейшем случае должны выполняться следующие очевидные условия:

.

Таким образом, математическая формулировка транспортной задачи имеет вид:

Найти:

,

при условиях

.

Задача носит название *замкнутой транспортной модели*, а условие  является естественным условием разрешимости замкнутой транспортной задачи.

**Обозначения и определения**

*Rn* – *n*-мерное евклидово пространство.

 – вектор столбец в *Rn*;

 – вектор-строка в *Rn*;

 – скалярное произведение, *x*, *y*∈*Rn*;

 – евклидова норма вектора в *Rn*;

 – матрица,  – транспортированная матрица;

 – произведение матрицы (*m*×*n*) на вектор (*n*×1);

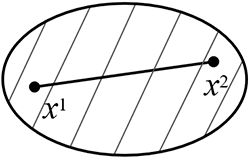
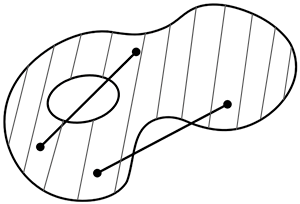
*ϕ*(*x*), *f*(*x*), *g*(*x*),… – как правило, вещественные (скалярные) функции, т.е. *ϕ*: *Rn* → *R*;

– градиент функции *ϕ* в точке *x*0 (*n*-мерный вектор);

|  |  |
| --- | --- |
|  | – матрица Гессе (матрица вторых производных) функции *ϕ* в точке *x*0; |

Т.к. , то *H*(*x*0) – есть вещественная симметричная матрица.

**Определение.** Множество *X*⊂ *Rn* называется *выпуклым*, если для ∀*x*1, *x*2∈*X*, ∀*λ*∈[0,1] . Иными словами, множество *X* выпукло, если оно вместе с любыми своими двумя точками *x*1 и *x*2 содержит соединяющий их отрезок.

* *

Выпуклое множество Невыпуклое множество

**Примеры.**

На числовой прямой *R* выпуклыми множествами являются всевозможные промежутки, т.е.:

* одноточечные множества;
* интервалы;
* полуинтервалы;
* отрезки;
* полупрямые;
* сама прямая.

В пространстве *Rn* примерами выпуклых множеств служат:

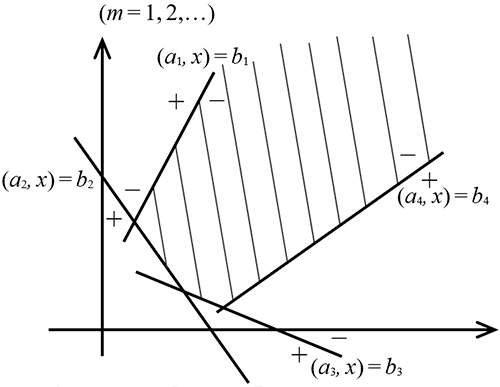
* само подпространство;
* любое его линейное подпространство;
* одноточечное множество;
* шар;
* отрезок,
* а также следующие множества:

 – прямая, проходящая через (⋅) *x*0 в направлении вектора *h*.

 – луч, выходящий из (⋅) *x*0 в направлении *h*.  – гиперпространство с нормалью *p*.

 – порождаемые ею полупространства.

Все перечисленные множества в *Rn*, кроме шара, являются частными случаями *выпуклого множества вида*:

,

где *A* – некоторая матрица размера *m*×*n* со строками ,

*b* ⊂ *Rm* – вектор.

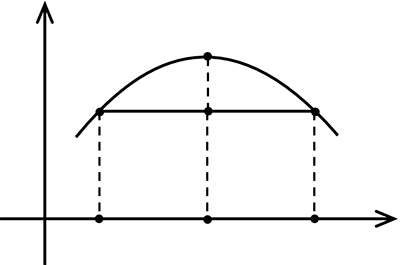
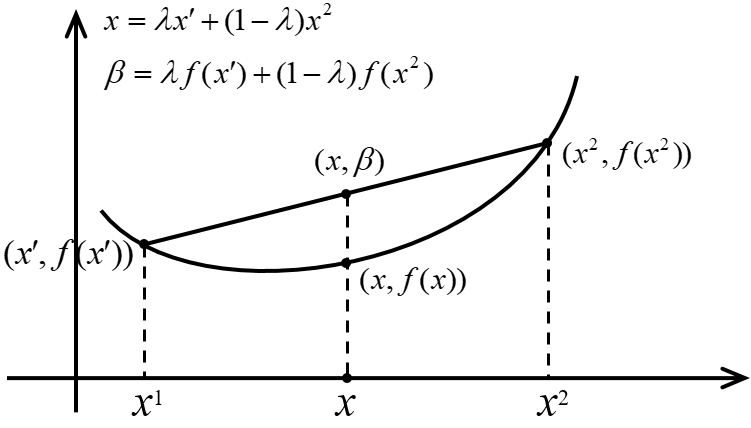
Множества такого вида называют *полиэдральными* или *полиэдрами*. Таким образом, полиэдр – множество решений некоторой системы конечного числа линейных неравенств (пересечение конечного числа полупространств).

**Определение.** Функция *f*, определенная на выпуклом множестве *X*⊂ *Rn*,называется *выпуклой на* *X*, если

,

при ∀*x*1, *x*2∈*X*, *λ*∈[0,1].

Если для любых  неравенство выполняется как строгое, то *f* называется строго выпуклой на *X*. Функция называется (строго) вогнутой, если функция –*f* (строго) выпукла.



Функцию , где *a*∈*Rn*, *b*∈*R* будем называть линейной. Ясно, что для неё исходное неравенство выполняется как равенство. Поэтому она выпукла и вогнута одновременно, но не строго.

#### Минимизация функций

Сама по себе постановка задачи оптимизации проста и естественна: заданы множество *X* и функция *ϕ*(*x*), определенная на *X*, требуется найти точку минимума или максимума функции *ϕ* на *X*.

Условимся записывать задачу на минимум в виде

, где

*ϕ* – *целевая функция*; *X* – *допустимое множество*.

Условимся также, что в дальнейшем будем рассматривать задачу на min, поскольку задача

.

Если допустимое множество *X = Rn*, то задача называется *безусловной* минимизацией, иначе, когда *X*≠ *Rn* – задача *условной* минимизации.

Отметим, что само понятие точки минимума неоднозначно и требует уточнения.

**Определение.** Точка *x*\*∈*X* называется:

1) точкой глобального минимума функции *ϕ* на множестве *X*, или глобальным решением задачи, если ;

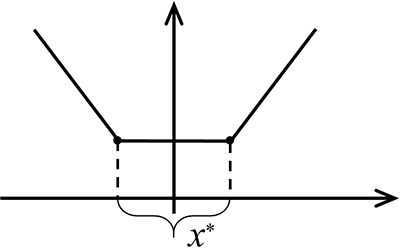
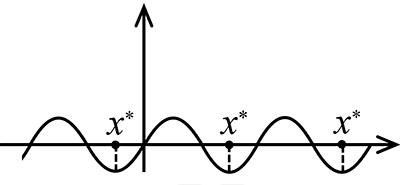
2) точкой локального минимума *ϕ* на *X*, или локальным решением задачи, если существует 

(Если  – *δ*-окрестность (⋅) *x*\* и *x*\* – локальный min, то *x*\* – глобальный min в области *X*∩ Δ).

Если неравенства в 1) и 2) выполняются как строгие, то говорят, что *x*\* – точка строгого min в глобальном или локальном смысле.

Ясно, что глобальное решение является локальным, обратное – не верно.

**Пример.** Глобальных min может быть много:



счетное множество

континуальное множество

Для записи того факта, что *x*\* является точкой глобального min функции *ϕ* на *X* используем запись:



или эквивалентная ей запись:

 – *оптимальная* точка.

Множество всех точек глобального min *ϕ* на *X* обозначим:



Таким образом,  – это произвольная точка из множества .

В дальнейшем мы часто будем прибегать к геометрической интерпретации задач оптимизации, основанной на понятии линий (или поверхностей) уровня функции *ϕ*, т.е. множеств вида:

- такое множество носит название *поверхность уровня α*.

Напомним известный факт из анализа: если функция *ϕ* дифференцируема в точке *x*, то градиент *ϕ*′(*x*) ортогонален к проходящей через *x* линии уровня *α* и направлен (если ) в сторону возрастания функции *ϕ*, т.е. поверхность *Lα* делит *Rn* на два подпространства:

 и .

*Задача* поиска оптимальной точки может быть сформулирована следующим образом: найти *α*\* = min*α* среди тех *α*, для которых *Lα*∩*X*≠ ∅. Тогда любая точка *x*∈*Lα*\* является оптимальной точкой.

Возможны два случая:

* *x\** лежит внутри *X* – рис. 1;
* *x\** лежит на границе *X* – рис. 2.

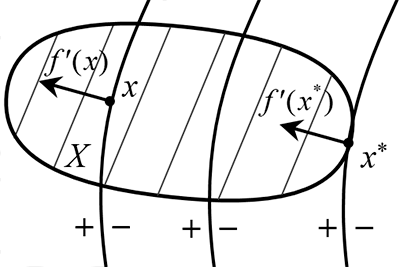
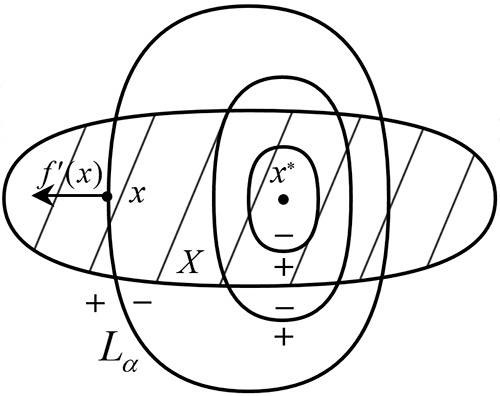


Рис.1

Рис.2

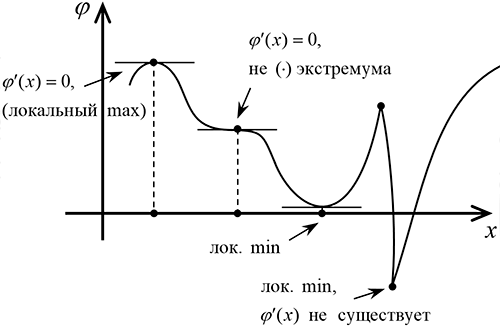
При изучении задач оптимизации в первую очередь возникает вопрос о существовании решения. Напомним в этой связи некоторые результаты из *математического анализа*.

**Теорема 1** (Вейерштрасса).Если *X* – компакт в *Rn* (т.е. замкнутое ограниченное множество), а *ϕ* – непрерывная функция на *X*, то существует *x*\*∈*X*: *x*\* – глобальный минимум *ϕ* на *X*, т.е. *глобальное решение задачи*  *существует*!

**Теорема 2** (необходимые условия локального минимума). Если *ϕ* – дифференцируема в точке *x*\*∈*X* и *x\** – локальный минимум, то *ϕ*′(*x*\*) = 0 (градиент равен нулю).

(обратное не верно).

**Определение.** Точка  в , называется *стационарной*

******

**Теорема 3** (достаточное условие локального минимума).

Если *ϕ* дважды дифференцируема в точке *x*\*∈*Rn* и выполняется:

1) *ϕ*(*x*\*) = 0;

2) матрица Гессе *ϕ*″(*x*\*) положительно определена,

то *x\** – (строгий) локальный минимум функции *ϕ*.

**Определение.** Матрица *H* называется *положительно определенной*, если .

*Критерий Сильвестра*: *H* положительно определена ⇔ ее главные миноры положительны.

Приведем несколько теорем для выпуклых задач.

**Определение.** Задача минимизации  называется выпуклой, если *X* – выпуклое множество, *ϕ* – выпуклая функция на *X*.

Справедливы следующие теоремы:

**Теорема 4.** Если задача минимизации  выпукла, то любое её локальное решение является также глобальным.

**Доказательство.** Пусть *x\** – локальное решение задачи, т.е. при некотором *ε*> 0 выполняется условие:

 при ,

где  – шар радиуса *ε* > 0 с центром в *x\**.

Для любых точек *x*∈*X*: *x* ≠ *x\**, положим .

Тогда . Покажем это.

1. Пусть ,

Если  ⇒ точка 

1. Пусть 



⇒ точка 

и, следовательно,

,

т.е. *x\** – глобальное решение задачи, *ч.т.д.*

Таким образом, *для выпуклых задач* понятия *локального и глобального решений не различаются*.

Второе свойство выпуклых задач можно высказать в виде следующего общего принципа: *необходимые условия оптимальности* в том или ином классе задач минимизации при соответствующих предположениях выпуклости *оказываются и достаточным*.

**Теорема 5.** Пусть функция *ϕ* выпукла на *Rn* и дифференцируема в точке *x*\*∈*Rn*. Если *ϕ*′(*x*\*) = 0, то *x*\* – точка минимума функции на *Rn*, т.е. решение задачи минимизации .

**Доказательство.** Для ∀*x*∈*X*, *λ*∈[0,1] имеем

.

Преобразуя эту формулу и, пользуясь дифференцируемостью функции *ϕ* в точке *x\**, получаем:

;

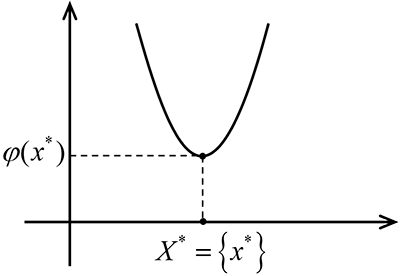
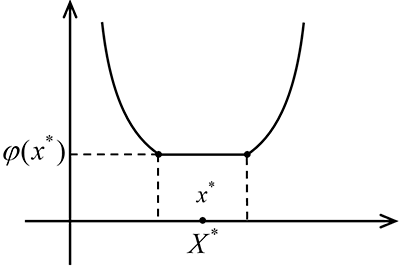
Отсюда предельным переходом при *λ*→ 0 выводим, что , *ч.т.д.* (т.е. для ∀*x*∈*X* ).

Полученные свойства выпуклых задач имеют важное значение не только в теории, но и в численных методах оптимизации. Дело в том, что большинство существующих численных методов позволяет, вообще говоря, находить лишь локальные решения, а точнее – стационарные точки задачи. Теоремы 4 и 5 говорят о том, что для выпуклой задачи *отыскание стационарной точки* автоматически означает *отыскание решения, причем глобального*.

Укажем ещё одно полезное свойство выпуклых задач.

**Теорема 6.** Пусть задача минимизации  выпукла и имеет решение.

Тогда множество её решений  выпукло. Если при этом *ϕ*(*x*) строго выпукла на *X*, то решение единственно, т.е. *X \**состоит из одной точки.



**Доказательство:**

1. Пусть 

При этом

 (\*)

По определению *X\** неравенство может выполняться только как равенство, поскольку 

, т.е. *X\** – выпукло.

1. Пусть *ϕ* – строго выпукла. Если предположить, что в *X\** существуют две различные точки *x*1 и *x*2, то при *λ*∈[0,1] неравенство (\*) должно быть строгим, что невозможно, т.к. *ϕ*\*– min и получается < min.

*Трудности*:

1. В случаях, когда функция *ϕ* достаточно проста, теоремы 1-3 помогают решить задачу минимизации даже в явном виде. Однако зачастую задача поиска стационарных точек является нетривиальной. А затем – перебор стационарных точек в поисках точки локального минимума, затем – перебор локальных экстремумов в поисках глобального экстремума.
2. Для задач условной минимизации теоремы 1-3 применимы в случае, когда локальное решение *x\** – внутренняя точка допустимого множества *X*. Если же экстремум достигается в угловых точках границы множества условий, то нарушается дифференцируемость ⇒ неприменимость методов классического анализа.

Т.о., в большинстве случаев задачу min*ϕ*(*x*) приходится *решать численно с применением ЭВМ и специальных методов минимизации*.

**Безусловная минимизация функции**

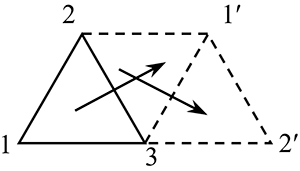
Методы оптимизации функций в *Rn* делятся на:

* локальные методы (поиск локального min, т.е. такой точки *x\**, что существует *δ* > 0, );
* нелокальные (или прямые) методы (поиск глобального min для ограничений снизу функции *ϕ*(*x*), т.е. если *α\** – нижняя грань, то поиск такой точки *x*\*: *ϕ*(*x*\*) = *α*\*). Для этих методов не требуется аналитического задания функции, надо только уметь вычислять ее значение в любой точке. Обычно – для функций сложной структуры.

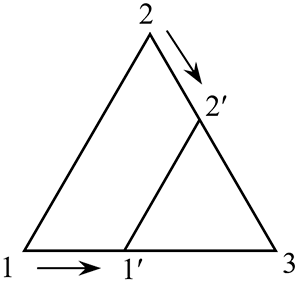
Нелокальные методы сводятся к уменьшению области, внутри которой находится оптимальная точка. Пример нелокального метода – *симплексный метод*.

**Определение.** *Симплекс* – выпуклое тело в *Rn*, состоящее из (*n*+ 1) равноудаленных точек – вершин симплекса, отрезок их соединяющий – ребро симплекса, в *R*2 – треугольник, в *R*3 – тетраэдр.

Неформальное описание симплексного метода: состоит из двух процедур – отражение и сжатие.

– *отражение*: симметричное отражение вершины с наибольшим значением *ϕ*(*x*) относительно противоположной грани ["перекатывание симплекса"]. Если , то выбирается другая (*i +*1)-я вершина.

Когда зацикливание (все (*n*+ 1)-вершины перебрали), то

– *сжатие*: уменьшение размеров симплекса при сохранении вершины с наименьшим значением *ϕ*(*x*), затем переход к отражению, и так далее, пока ребро симплекса не станет меньше некоторого числа: .

*Достоинства*: с большой вероятностью метод не распознает локальный минимум ("не остановится").

*Локальные* методы основаны на построении *релаксационной* последовательности {*xi*} такой, что  и .

Поэтому *релаксационные* методы называют также *методами спуска*.

**Классификация релаксационных методов**

С одной стороны,

* *одношаговые* методы:  – каждый шаг (*i*+ 1) зависит только от предыдущей точки *xi* и значения функции *ϕ*(*xi*);
* *двухшаговые* методы:  – зависимость от двух предыдущих точек;
* *и т.д.;*

С другой стороны,

* *методы нулевого порядка:* если используются только значения минимизируемой функции *ϕ*(*x*);
* *методы первого порядка*: если используются только значение *ϕ*(*x*) и *ϕ*′(*x*);
* *методы второго порядка*: если используются значения *ϕ*(*x*), *ϕ*′(*x*) и *ϕ*″(*x*);
* *etc;*

**Градиентные методы (методы первого порядка)**

Итак, будем рассматривать задачу:

 (безусловная минимизация),

предполагая, что функция *ϕ*(*x*) непрерывно дифференцируема на *Rn*, т.е. *ϕ*(*x*)∈*C*1(*Rn*).

По определению дифференцируемой функции

, (1)

где .

Если , то при достаточно малых  главная часть приращения для *ϕ* будет определяться дифференциалом функции . Оценим величину Справедливо неравенство Коши-Буняковского:

,

причем, если *ϕ*′(*x*) ≠ 0, то правое неравенство превращается в равенство, только при *h*=−*αϕ*′(*x*), а левое только при *h*=*αϕ*′(*x*), где *α*= *const*≥ 0.

Отсюда ясно, что при *ϕ*′(*x*) ≠ 0 *направление наибыстрейшего возрастания* функции *ϕ*(*x*) в точке *x* совпадает с *направлением градиента* *ϕ*(*x*), а направление наибыстрейшего убывания – с направлением антиградиента –*ϕ*′(*x*).

Это свойство градиента лежит в основе ряда итерационных методов минимизации функций. Один из таких – *градиентный*. Он предполагает, как, впрочем, и все остальные итерационные методы, наличие априорной точки начального приближения.

Предположим, что начальная точка *x*0 уже выбрана, тогда градиентный метод заключается в построении последовательности {*xk*} по правилу:

 (2)

*αk* – величина шага, *xk* – направление спуска.

Если , то шаг  можно выбрать так, чтобы получить релаксационную последовательность: . Действительно, подставляя (2) в (1), имеем:

,

при всех достаточно малых *αk*> 0.

Если , то *xk* – стационарная точка. В этом случае процесс (2) прекращается и проводятся дополнительные исследования поведения функции в окрестности точки *xk* для выяснения того, достигается ли в точке *xk* минимум функции *ϕ*(*x*) или не достигается.

Существуют различные *способы выбора величины шага* *α­­k* в методе (2). В зависимости от способа выбора *α­­k* можно получить различные варианты градиентного метода.

**Метод наискорейшего спуска**

На луче , направленном по антиградиенту, введем функцию одной переменной



и определим *α­­k* из условий

.

Другими словами *α­­k* выбирается так, чтобы *ϕ*(*xk*+1) в заданном направлении была наименьшей для чего на любом шаге необходимо решать задачу одномерной минимизации функции *ψ* (*α*), например, с помощью .

**Пример.** Рассмотрим задачу



с начальной точкой .

Из общих соображений ясно, что *ϕ*min = 0 при 

*1-й шаг*:



Ищем

.

Функция *ψ*(*α*) имеет следующий вид:

.

Решаем уравнение , т.е.

;

.

*2-й шаг*:

;

.

Решаем уравнение  ⇒

; ⇒

.

*3-й шаг*:





Решаем уравнение  ⇒

;

, и.т.д.

*Представим решение задачи графически:*

Из графического представления можно сделать вывод, что имеет место:

|  |  |
| --- | --- |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\15.png | ⇒ а) сходимость к истинной точке минимума  б) взаимная перпендикулярность градиентов |

**Свойства метода наискорейшего спуска**

1. На любом шаге направление спуска меняется на ортогональное. Действительно, *αk* ищется из условия  ⇒



1. Точка *xk*+1 лежит на луче, исходящем из точки *xk* и касательным к поверхности уровня *Lϕ*(*xk*+1). Действительно, с одной стороны, несомненно, что . С другой стороны, градиент *ϕ*′(*xk*+1) ортогонален касательной к поверхности уровня *Lϕ*(*xk*+1), поэтому по свойству 1 направление спуска касательно к поверхности *Lϕ*(*xk*+1).

*Иначе.* *ϕ*′(*xk*+1) ортогонален направлению спуска ⇒ луч, проходящий из точки *xk* – касательной к поверхности .

*Проблемы* (общие для релаксационных методов).

1. Имеет ли последовательность {*xk*} предел в смысле сходимости по норме: существует ?
2. Является ли этот предел аргументом, составляющим минимум функции *ϕ*?
3. Какова скорость сходимости  или *ϕ*(*xk*) – *ϕ*(*x*\*)?
4. Каковы вычислительные затраты.

**Исследование метода наискорейшего спуска для квадратичной функции**

Рассмотрим квадратичную функцию

,

где *A* – симметричная, положительно определенная матрица.

Можно показать, что *A* – симметричная положительно определенная матрица ⇔ *ϕ* – строго выпукла.

, т.е.  – стационарная точка.

Попробуем записать метод наискорейшего спуска для квадратичной функции.

Итак,



 (w)

,

т.к. *A* – положительно определена, и значит для нее справедливо: (*Ah*, *h*) > 0 ∀*h*∈*Rn*≠ 0.

Для определения скорости сходимости оценим отношение



Имеем:



С другой стороны,



Для простоты дальнейших изложений предположим, что матрица *A* приведена к диагональному виду (т.е. выполнено преобразование координат) так, что , где *λi* – собственные числа матрицы *A*.

* Собственные числа симметричной положительно определенной матрицы всегда положительны.
* Для симметричной матрицы существует ортогональная матрица (*TT* = *T*-1) *T* такая, что *TTAT* – диагональная матрица .

Если , то

,

Тогда

.

Если ввести обозначение , то



Это называется *геометрической скоростью сходимости* (сходимость геометрической прогрессии).

Рассмотрим величину

.

Верхний предел  называется *порядком сходимости метода*.

В нашем случае квадратичной функции

.

Поэтому



⇒ 

⇒ получили сходимость с порядком 1 или *линейную сходимость*. Бывает порядок больше 1 – *сверхлинейная сходимость*.

При исследовании метода наискорейшего спуска для квадратичной функции получили, в частности, следующие результаты:

1. 
2. 

**Определение.**

Пусть *ϕ*(*xk*)→*ϕ*(*x*\*) при *k* → ∞.

Последовательность *ϕ*(*xk*) сходится к *ϕ*(*x\**) *линейно* (с линейной скоростью, со скоростью геометрической прогрессии), если существуют такие константы *q*∈(0,1) и *k*0, что

, при *k*≥ *k*0.

Последовательность *ϕ*(*xk*) сходится к *ϕ*(*x\**) *сверхлинейно*, если

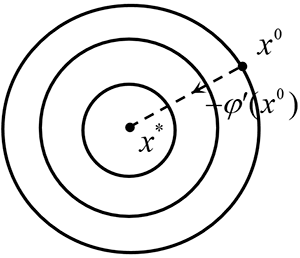
, при *k*→ ∞.

Последовательность *ϕ*(*xk*) сходится к *ϕ*(*x\**) с *квадратичной скоростью*, если существуют такие константы *c*≥ 0 и *k*0, что

, при *k* ≥ *k*0.

Вообще, порядок сходимости, равный 1, означает, что значение величины Δ*k* убывает, в основном, по закону геометрической прогрессии. Порядок сходимости, равный 2 (квадратичная сходимость) означает, что при достаточно больших *k* Δ*k*+1 ~ Δ*k*2. В этом случае, если к тому же Δ*k* – малая величина, например,  при , то Δ*k*+1 равно , т.е. фактически удваивается число нулей после запятой.

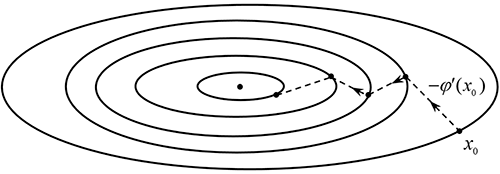
*Частные случаи*:

1. Пусть *l*= *L*, т.е. матрица *A*= ­*LI*= *lI* – пропорциональна единичной окружности (линии уровня – окружности).

Тогда:



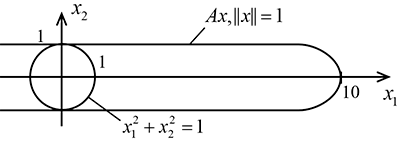
⇒ φ(*xk*+1) = φ(*x*\*) метод сходится за один шаг.

1. *l*≤ *L*: сходимость может быть еле заметной (*q*~ 1), а графически это означает, что линии уровня функции сильно вытянуты и функция имеет так называемый "овражный" характер. Это означает, что небольшое изменение некоторых переменных приводит к резкому изменению значений функции – эта группа переменных характеризует "склон оврага", а по остальным переменным, задающим направление "дна оврага", функция меняется незначительно.

Число  называется числом обусловленности матрицы ⇒ *cond* ≥ 1.

Матрица называется *хорошо обусловленной*, если *cond* ~ 1 и наоборот.

Вообще, число обусловленности геометрически можно трактовать как меру искажения отображения матрицей *A* единичной сферы. Действительно, *cond*(*A*) есть отношение наибольшего к наименьшим расстояниям между точками на единичной сфере после её отображения матрицей *A*. Чем больше *cond*(*A*), тем больше искажение единичной сферы при её преобразовании в эллиптическую форму – пусть *A*= *diag*(10,1).

*Вывод*: Метод наискорейшего спуска быстро сходится для хорошо обусловленных матриц и наоборот.

*Почему так много внимания уделяли квадратичной функции?*

В окрестности locmin любую функцию можно приблизить квадратичной, и всё сказанное выше про матрицу *A* будет справедливым для матрицы Гесса *H*(*x*\*), которая заменяет *A* в рассмотренном выше примере.

*Геометрически*: Линии уровня становятся замкнутыми и по мере приближения к *x*\* всё более напоминают эллипс.

*Общий случай.*

**Определение 1.** Функция *ϕ* на множестве *X*∈*Rn* удовлетворяет *условию Липшица*, если существует . Если градиент функции *ϕ* существует, непрерывен и удовлетворяет условию Липшица, то обозначается *ϕ*∈*C*1,1.

**Определение 2.** Функция *ϕ* называется *сильно выпуклой с параметром* , если.

**Теорема** (о сходимости метода наискорейшего спуска). Рассмотрим задачу . Пусть *ϕ*∈*С*1,1(*Rn*) и *ϕ* – сильно выпуклая c параметром æ. Тогда при любом начальном приближении для последовательности {*xk*}, построенной по методу наискорейшего спуска, справедливы соотношения:

1) 

2) 

**Замечания.**

1. Для *квадратичной функции* :

1. постоянная Липшица *L* есть наибольшее собственное число матрицы *A*:

;

1. она сильно выпукла с параметром . Действительно,





2. Эквивалентные *ограничения сильной выпуклости*:

1. *ϕ* – сильно выпукла ⇔  – выпукла (это означает, что *ϕ* имеет "квадратичный запас" выпуклости);
2. пусть *ϕ*∈*C*1, *ϕ* – сильно выпукла ⇔ ;
3. пусть *ϕ*∈*С*2, *ϕ* – сильно выпукла ⇔ , ∀*x*, т.е.  [в смысле положительной определенности разности матриц]. С другой стороны, из условия Липшица , поэтому для сильно выпуклой *ϕ*∈*С*2 существует двойная оценка матрицы Гессе: .

Покажем, что . С одной стороны, из б) имеем



С другой стороны,



⇒ , *ч.т.д.*

Выпуклость:

* .

Строгая выпуклость:

* .

Сильная выпуклость:

*  для ∀*u*,*υ*∈*Rn*

*Графическое представление дифференциальных критериев выпуклости.*

|  |  |
| --- | --- |
| 21 | График *выпуклой* функции расположен не ниже касательной плоскости , проходящей через произв. точку поверхности . |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\20.png | График *строго выпуклой* функции имеет единственную общую точку с этой плоскостью. |
| C:\Users\user\AppData\Local\Microsoft\Windows\INetCache\Content.Word\22.png | Пусть *υ*\* – (⋅)min ⇒ . Поверхность  – это параболоид вращения с вершиной в точке .  ⇒ График *сильно выпуклой* функции расположен внутри некоторого параболоида вращения. |

**Другие градиентные методы**

Напомним, градиентные методы заключаются в построении релаксационной последовательности:



Градиентные методы различаются между собой способом выбора *αk*.

**1.** **Метод наискорейшего спуска**, который был рассмотрен выше, заключается в выборе

.

Такой способ выбора *αk* является в некотором смысле наилучшим, т.к. он обеспечивает *достижение наименьшего значения функции* вдоль заданного направления. Однако он требует решения на любом шаге *одномерной задачи минимизации*. Эти задачи решаются, как правило, приближенно с помощью численных методов, что приводит к *значительному объему вычислений*. Кроме того, метод может привести к *плохой сходимости* (овраги!).

Другим подходом для построения релаксационной последовательности является попытка определить *αk* до начала вычислений. Какие есть для этого основания?

Допустим, что можно построить оценку для *αk* такую, что для *ε*∈(0,1) выполняется неравенство

 (1)

Тогда очевидно, что  и соответствующий метод минимизации будет методом спуска.

Справедливы следующие утверждения:

##### **Лемма 1.** Пусть функция ϕ∈C1,1(Rn) и



Тогда для ∀*xk*∈*Rn*, *ε*∈(0,1) условие (1) выполнено при



**Лемма 2.** Пусть *ϕ* дважды дифференцируема и матрица Гессе удовлетворяет условию Липшица и



Тогда для ∀*xk*∈*Rn*, *ε*∈(0,1) условие (1) выполняется при



**2. Градиентный метод с постоянным шагом.**

В этом методе полагается *αk*≡ *const*. При этом иногда удается добиться выполнения условия (1). Но для этого необходимо знать константы *M* и *D*, что далеко не всегда удается вычислить.

Т.о., метод прост в реализации, но есть проблемы со сходимостью.

**Пример.** Пусть *ϕ*(*x*) = *αx*2

*ϕ*min = 0; *x\** = 0;

Тогда

 ⇔ метод сходится.

Сходимость *медленная*!

**3. Градиентный метод с убыванием длины шага.**

В ряде методов достаточно потребовать выполнения условий:

 (например, )

На интуитивном уровне объяснение следующее:

* условие сходимости ряда  накладывают, чтобы добиться достаточно быстрой сходимости последовательности *αk* к нулю с целью обеспечения сходимости метода в окрестности точки экстремума *x*\*.
* условие расходимости ряда  призвано обеспечить достижение точки экстремума *x*\* даже при неудачном выборе начального приближения *x*0, т.е. при больших расстояниях от *x*0 до *x*\*.

Сходимость *медленная*!

**4. Градиентный метод с дроблением шага.**

Ещё один адаптивный способ выбора коэффициентов *αk*. Выбираются некоторые *constβ*>0 и 0< *λ*< 1 (обычно *λ*= ½). Для коэффициента *α* = *β* проверяется выполнение условия . Если оно выполняется, то полагают *αk*= *α*. Если нет, то производится дробление шага, т.е. принимается *α* = *λβ*, и т.д. до тех пор, пока не выполнится требуемое неравенство.

Процесс дробления не может продолжаться бесконечно, поскольку −*ϕ*′(*x*) – направление убывания функции. Первое *α*, при котором условие выполнено и принимается за *αk*.

Как показывает следующая лемма, с помощью описанного процесса дробления шага можно добиться выполнения неравенства. (1)

**Лемма 3.** Пусть функция *ϕ* дифференцируема на *Rn*. Тогда для  найдется такое *α*0 > 0, что при ∀*α*∈(0, *α*0] выполнено условие

.

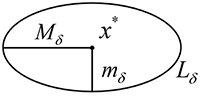
Если необходимое неравенство оказывается выполненным при начальном значении *α*= *β*, то иногда полезно увеличить шаг, взяв *α* = *μβ*, где *μ*> 1. Так можно продолжать до тех пор, пока значения функции не перестанут уменьшаться. Последнее *α*, при котором произошло уменьшение, и берется в этом случае за *αk*.

**5. Овражный метод** – эвристический двухшаговый метод минимизации овражных функций.

*Характеристика степени овражности*:

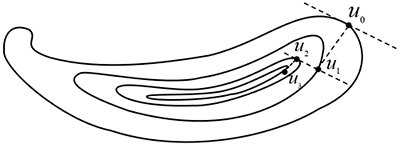
Пусть *x*\* – точка минимума, *δ*> 0.

Рассмотрим поверхность уровня .

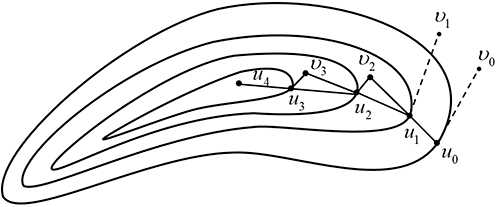
Введем 



**Определение.** Тогда  называется *числом обусловленности точки locmin* .

****Рассмотрим "овражную" функцию (вытянута вдоль некоторых направлений). Если точка лежит на склоне оврага, то направление спуска из этой точки будет почти перпендикулярно к направлению "дна оврага", и в результате приближения {*uk*}, получаемые градиентным методом, будут *поочередно находиться то на одном, то на другом "склоне оврага"*. Если "склоны оврага" достаточно круты, то такие скачки "со склона на склон" точек {*uk*} могут сильно замедлить сходимость градиентного метода.

Для ускорения сходимости можно предложить следующий эвристический прием, называемый *овражным методом*:

Пусть *υ*0, *υ*1 – две произвольные близкие точки. Совершаем из них по одному шагу методом наискорейшего спуска (или ∀ вариант градиентного метода).

Попадаем в окрестность "дна оврага". Соединяя их прямой, делаем большой шаг в полученном направлении, перемещаясь вдоль "дна оврага". Из полученной точки *υ*2, которая находится на "склоне оврага", производят спуск с помощью градиентного метода и определяют следующую точку *u*2 на "дне оврага" и т.д.

Формула метода выглядит следующим образом.



Здесь:

 определяет знак - чтобы спускаться, а не подниматься;

 - определяет направление спуска по дну оврага;

*h -* овражный шаг, выбирается эмпирически и от него многое зависит.

Если *h* – большое, то на крутых склонах точки *υk* могут *слишком далеко удаляться от "дна оврага"* ⇒ большие объемы вычислений для градиентного метода спуска в очередную точку на "дне оврага", кроме этого может произойти выброс точки *υk* из "оврага" и *правильное направление поиска будет потеряно*.

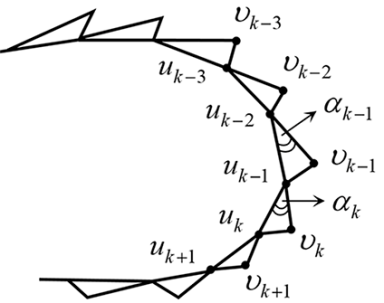
Если *h* – малое, то эффект от применения овражного метода может быть незначительным.

*Эффективность* применения овражного метода может *резко возрасти*, если величину *h* выбирать переменной, реагирующей на "повороты" оврага, с тем, чтобы:

1. быстрее проходить прямолинейные участки на "дне оврага" за счет увеличения овражного шага;
2. на крутых поворотах "оврага" избежать выброса из "оврага" за счет уменьшения овражного шага;
3. добиться min отклонения точки *υk* от дна оврага с целью уменьшения объема вычислений для градиентного метода.

Для правильной реакции на "повороты" оврага надо учитывать "кривизну" оврага.

Причем информацию о кривизне желательно получить по результатам предыдущих шагов.

Один из способов выбора шага:

,

где *αk* – угол между векторами , определяемый

условием

,

*c* − *const* > 1 – параметр алгоритма.

 возрастает при возрастании кривизны ⇒ при переходе от участка с меньшей кривизной на участок с большей кривизной имеем

 и наоборот.

На участках с постоянной кривизной



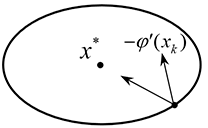
⇒ шаг остается постоянным, который был сформирован при выходе на рассматриваемый участок.

Параметр *с* регулирует чувствительность "метода к изменению кривизны (повороты)" и во многом определяет скорость движения "по оврагу".

# Методы II порядка минимизации функции

(использование вторых производных)

*Общая идея*:



направление спуска по градиентному методу

желаемое направление

(надо "довернуть" направление спуска)

Последовательность {*xk*} будем строить по формулам:

,

где *γk* – длина шага, *Hk* – матрица поворота (*n*×*m*).

Как выбрать матрицу *Hk*?

Если взять квадратичную функцию

,

то хочется сразу попасть в экстремальную точку:



⇒ в качестве "матрицы доворота" надо брать .

*В общем случае*: пусть *ϕ* – дважды дифференцируема в *Rn*, разложим *ϕ*(*x*) в ряд Тейлора в точке

Иначе формулу можно представить в виде:

, где  – квадратичная функция.

Пренебрегаем  и ищем .

;

Пусть *ϕ*″(*xk*) – положительно определена для ∀*xk*∈*Rn* ⇒ существует [*ϕ*″(*xk*)-1].

Решая это уравнение, получим:

 – это и есть метод Ньютона.

Квадратичная функция с положительно определенной *ϕ*″ сильно выпукла, тогда уравнение определяет единственную точку глобального минимума .

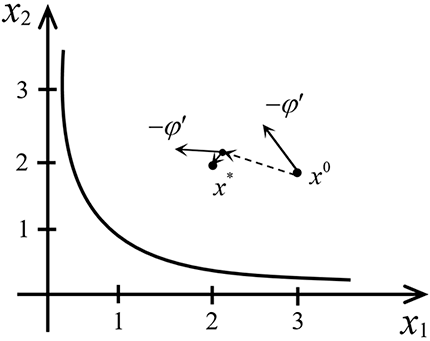
Далее рассмотрим пример использования метода Ньютона для решения задачи минимизации функции.

*Пример.*

Область существования (*ϕ*″(*x*))-1 совпадает с областью положительной определенности матрицы *ϕ*″(*x*), которую мы будем искать с помощью критерия Сильвестра.

*Критерий Сильвестра.*

Симметрическая матрица *A* положительно определена ⇔ если все её главные миноры положительны. При этом главным минором матрицы *A* называется определитель матрицы, построенной из элементов матрицы *A*, стоящих на пересечении строк и столбцов с одинаковыми номерами.

.

Возьмем 









 , и.т.д.

Можно показать, что сходимость при  будет хуже.

*Достоинства метода Ньютона:*

1. Для квадратичной функции сходится за один шаг (метод Ньютона можно рассмотреть, как градиентный метод с преобразованием координат [умножение на *H*-1] таким, что исчезает "овраг", т.е. линии уровня становятся окружностями).
2. Высокая скорость сходимости. Можно показать, что

.

Порядок сходимости:



⇒ важен выбор *q* в алгоритме. Если *q*=10-1, то за один шаг точность результата увеличивается на 2 разряда, а при линейной сходимости – на один разряд.

*Недостатки метода Ньютона:*

1. Локальная сходимость (матрица Гессе должна быть невырожденной). Начальное приближение надо выбирать в окрестности точки локального минимума.
2. Большие вычислительные затраты:

* вычисление матрицы *ϕ*″;
* необходимость обращать её.

*Общие рекомендации*: сначала применять градиентный метод, затем – метод Ньютона.

Например, существует так называемый метод Марквардта-Левенберга:

.

*При больших* *αk* (вдали от *x*\*)матрица и это фактически градиентный метод.

*При малых* *αk* (вблизи от *x*\*) это метод Ньютона.

Имеет место:

**Теорема** (о сходимости метода Ньютона).

Пусть

1) *ϕ* – сильно выпукла на *Rn* с параметром æ.

2) *ϕ*∈*c*2,2, т.е. *ϕ*″ – дважды дифференцируема и *ϕ*″ удовлетворяет условию Липшица:

;

3) Начальное приближение *x*0 удовлетворяет условию

,

т.е. , где 0 < *q* < 1.

Тогда последовательность  сходится к точке минимума *x*\* с квадратичной скоростью:

 (квадратичная сходимость)

1. Несколько слов о норме матрицы:

**Определение.** Норму (*n*×*n*)-матрицы определим следующим образом:

,

Тогда

.

Поскольку *ATA* есть симметричная (*n*×*n*)-матрица, то существует ортогональная матрица  такая, что  – диагональная матрица  ⇒

, где  – наибольшее собственное значение матрицы *ATA*.

 – наименьшее собственное значение матрицы *ATA*.

Для такой нормы выполняются все три условия

1. , если *A* – ненулевая (покомпонентно);
2. ;
3. , где *α* – скаляр.

Кроме того, из определения нормы матрицы следует, что

.

Имеем также

.

2. Отметим ещё раз, что для сходимости метода Ньютона начальная точка *x*0 должна выбираться достаточной близкой к искомой точке *x\**. Это требование в теореме выражено условием 3). Действительно, сильная выпуклость *ϕ* означает:

;

⇒ чем меньше *q*, тем ближе надо выбирать точку *x*0 к *x\** и тем быстрее сходимость.

***Достоинства и недостатки градиентных методов и метода Ньютона***

|  |  |  |
| --- | --- | --- |
| **Метод** | **Достоинства** | **Недостатки** |
| Градиентный метод | 1. Глобальная сходимость, т.е. слабые требования на начальные приближения точки х0 и к *f*(*x*);  2. Относительная простота вычислений | 1. Медленная сходимость (геометрическая скорость сходимости, порядок сходимости *d*=1);  2. Необходимость вычисления длины шага. |
| Метод Ньютона | 1. Быстрая сходимость. | 1. Локальная сходимость (начальная точка должна быть близка к *х\**); 2. Большой объем вычислений (на любом шаге требуется вычислять матрицу вторых производных и обращать её). 3. Жесткие требования на саму функцию (непрерывная вторая производная). |

*Порядок применения методов*

1. На 1-м этапе – методы 1-ого порядка, т.к. они обеспечивают глобальную сходимость.
2. На 2-м этапе (когда приращения невелики) – выгодно применять методы 2-ого порядка.

Перечисленные методы (градиентные и Ньютона) являются классическими.

Можно предложить методы более высокого порядка, тогда естественно ожидать, что порядок сходимости будет равен *p*.

Как уже отмечалось, к недостаткам метода Ньютона относится:

* локальная сходимость;
* большой объем вычислений;
* жесткие требования на гладкость функции.

В силу названных причин применение классического метода Ньютона не всегда приводит к успеху.

Многочисленные модификации направлены на то, чтобы, *сохраняя* основное достоинство метода Ньютона – *быструю сходимость*, *уменьшить трудоемкость и ослабить* требования на выбор *начального приближения*.

**Метод Ньютона с регулировкой шага**

Рассмотрим метод

,

это метод Ньютона с регулировкой шага.

При *αk* ≡ 1 мы получили классический метод Ньютона.

Выбор *αk* обычно – из условия min функции, вдоль заданного направления, или методом дробления шага, обеспечивающего выполнение условия *ϕ*(*xk*+1) < *ϕ­*(*xk*).

Можно показать, что подобные методы регулировки шага *сходятся* при *любой начальной* точке *x*0∈*Rn*, причем скорость сходимости будет либо *сверхлинейна*, либо *квадратичная* в зависимости от требований, которым удовлетворяет функция *ϕ*.

Таким образом, с помощью регулировки длины шага преодолевается недостаток метода, связанный с необходимостью отыскания хорошего начального приближения.

Однако, трудоемкость вычислений при этом не исчезает.

Более перспективным в этом плане оказывается другой подход, при котором строится аппроксимация матрицы (*ϕ*″(*xk*))-1 на основе информации о значениях градиентов *ϕ*′(*xk*), *ϕ*′(*xk*+1),…

**Квазиньютоновы методы**

Пусть функция *ϕ* дважды дифференцируема. Рассмотрим метод

 (1)

*αk* – шаг, *Hk* – матрица.

1. Если *Hk* = единичная, имеем градиентный метод.
2. Если , то это метод Ньютона (с точностью до шага).
3. Если , то имеем метод, который объединяет достоинства обоих методов.

Заметим, что

.

Полагая невырожденной матрицу *ϕ*″(*xk*+1), отсюда с точностью до членов более высокого порядка малости по сравнению с  имеем:

.

Рассмотрим квадратичную функцию . Для нее

,

и приближенное равенство обращается в точное:

.

Поэтому естественно потребовать, чтобы для матрицы *Hk*+1, приближающей (*ϕ*″(*xk*+1))-1, выполнялось условие:

 (\*)

Это условие называется *квазиньютоновским*. Оно лежит в основе целого ряда методов аппроксимации (*ϕ*″)-1. Соответствующие методы минимизации, для которых на любом шаге выполняется квазиньютоновское условие, также называются квазиньютоновскими.

Пусть приближения к (*ϕ*″)-1 пересчитываются шаг от шага по формуле .

Различные квазиньютоновские методы различаются способом вычисления "добавки" Δ*Hk* таким образом, чтобы удовлетворялось соотношение (\*).

**1. Метод Дэвидона-Флетчера-Пауэлла**

Обозначим:



Метод заключается в построении релаксационной последовательности по следующему правилу:

 (1.1)

Длина шага *αk* в квазиньютоновых методах выбирается так же, как в методе наискорейшего спуска:



Как правило, начальное значение *H*0 = *I*. Вообще, если *H*0 ­– симметричная матрица, то *Hk* ­– симметричная матрица для любого *k*.

**2. Метод Бройдена-Флетчера-Шанно**

Имеем.

Если поставить задачу уточнять обратную матрицу, т.е.  тогда:

 (1.2)

(этот метод более устойчив к ошибкам округления)

Можно доказать, что для квадратичной функции , где *A* – симметричная, положительно определенная матрица, оба метода (1.1) и (1.2) при любом начальном приближении *x*0∈*Rn* генерируют одну и ту же последовательность точек , причем

,

т.е. *квазиньютоновские методы* позволяют найти min *квадратичной* функции *за n-шагов*.

Для неквадратичной функции, это не так. Однако можно показать, что при соответствующих предположениях , причем скорость сходимости сверхлинейна.

Так, например, пусть *ϕ* – дважды непрерывно дифференцируемая функция, сильно выпукла на *Rn*.

Тогда при любом начальном приближении *x*0∈*Rn* последовательность точек {*xk*}, определяемая формулами (1.1.) и (1.2), сходится к *x*\*.

Если, при этом, для всех *x*:, справедливы неравенства

,

то *xk* сходится к *x*\* сверхлинейно

.

*Замечания о квазиньютоновских методах*:

1. Это двухшаговые методы.
2. Для квадратичных функций сходятся за *n*-шагов.
3. Обладают следующими преимуществами:

* небольшая вычислительная сложность;
* более глобальная сходимость, чем в методе Ньютона;
* сверхлинейная скорость сходимости.

#### Математическое программирование

(постановка задачи и основные определения)

*Основная* задача математического программирования состоит в минимизации вещественной функции на множестве, определенном системой ограничений типа равенства и/или неравенства.

Записывать задачу будем в следующем виде:

, где

**Определение.** *ϕ* называется *целевой функцией*.

*Ограничения или условия* записываются в виде:

, где

**Определение.** *X* называется *допустимым множеством*.

*Размерность задачи*: *n* – число переменных; *m+l* – число ограничений.

Запись min*ϕ*(*x*) означает:

1. либо найти оптимальную точку

.

**Определение.** Всякая допустимая *точка* *x*∈*X* называется *планом*; *x\** – *оптимальный план*.

1. если *x\** не существует, то найти , например, .
2. либо показать, что *X* = ∅ (допустимое множество – пусто).

**Классификация задач математического программирования**

1. Если целевая функция линейна, т.е. , где *c*∈*Rn* и ограничения линейны, т.е. имеют вид:

*Ax* ≤ *b*, где *A* – *m*×*n* – матрица, *b*∈*Rm*

*Gx*=*h*, где *G* – *l*×*n* – матрица, *h*∈*Rl*,

то это *задача линейного программирования* (иначе, нелинейного, например, квадратичного).

1. Если целевая функция *ϕ* – выпукла и допустимое множество *X* – выпуклое (*fi*, *gk* – выпуклые функции), то это *задача выпуклого программирования*.
2. Если по условию переменные – целые числа, т.е. , то это – *задача целочисленного программирования* (в данном курсе не рассматривается).

**Спецификация задач математического программирования**

* как правило, методы классического анализа для отыскания условных экстремумов неприменимы (экстремум достигается в угловых точках допустимого множества).
* большое количество переменных и ограничений в практических задачах, так что задача перебора точек, подозреваемых в экстремальности, может оказаться нетривиальной.

⇒ целью математического программирования является *создание*, где это возможно, *аналитических методов* определения решения, а при отсутствии таких методов – *создание эффективных вычислительных способов* получения *приближенного решения*.

Наименование предмета – *математическое программирование* – связано с тем, что целью решения задач является *выбор программы действий*.

**Cведения о выпуклых множествах**

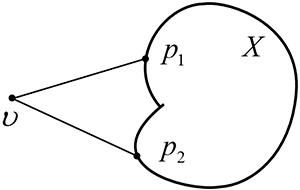
**Определение 1.**Множество  называется *выпуклым*, если   (выпуклое множество содержит отрезок, соединяющий две любые его точки).

**Определение 2.**Точка *x*∈*X* называется *внутренней*, если существует , где  – *ε*-окрестность точки *x*, т.е. точка *x*∈*X* называется *внутренней*, если существует такая её окрестность, все точки которой принадлежат *X*. И наоборот, если найдется такая *ε*-окрестность точки *υ*, которая не содержит ни одной точки множества *X* – такая точка называется *внешней* по отношению к множеству *X*.

**Определение 3.** Точка *x*∈*X* называется *граничной*, если ∀*ε* >0 существует и существует , т.е. в любой окрестности точки *x* содержатся как точки, принадлежащие множеству *X*, так и точки, не предлежащие этому множеству.

**Определение 4.** *Проекцией* точки *υ* на множество *X* называют такую точку *p*∈*X*, что

.

При этом,  называют "расстояние" от точки *υ* до множества *X*. Ясно, что если *υ*∈*X*, то *p*= *υ*. Если же *υ*∉*X*, и множество *X* – открыто, то проекция *p* не существует. Если множество *X* – не выпукло, то проекция может быть не единственной.

Верно следующее утверждение:

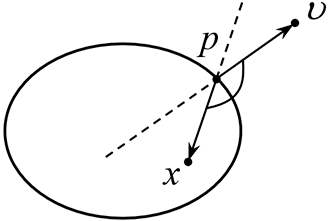
**Лемма 1.** Пусть X – выпуклое замкнутое множество из Rn, , тогда:

1) Любая точка *υ*∈*Rn* имеет и притом единственную проекцию на это множество;

2) Для того чтобы точка *p*∈*X* была проекцией точки *υ* на множество *X*, необходимо и достаточно выполнения неравенства  для ∀*x*∈*X*.

**Доказательство.**

тупой угол (*x*–*p*,*υ*–*p*) ≤ 0)



*Докажем первое утверждение леммы.*

Рассмотрим функцию *g*(*x*) вида

.

Поскольку *g*(*x*) сильно выпукла, то по следствию из теоремы об ограниченности множеств Лебега для сильно выпуклой функции можно утверждать, что *g*(*x*) достигает своей нижней грани на Х в единственной точке *p*′∈*X*.

Это означает, что

,

Причем равенство здесь возможно, только когда *x*=*p*′ (т.к. *p*′ единственная точка), а тогда *p*=*p*′, что и требовалось доказать.

*Докажем второе утверждение леммы.*

*Необходимость*. Пусть *p* – проекция точки *υ* на *X*.

Возьмем произвольную точку *x*∈*X,* отличную от *р,* рассмотрим точку . Ввиду выпуклости множества Х для ∀*α*∈[0,1] точка *z*∈*X.*

Так как , и из определения проекции следует, что , то .

Поскольку это неравенство справедливо для ∀*α*∈[0,1], то .

Переходя к пределу при *a->0,* имеем , что и требовалось доказать.

**Достаточность.** Пусть верно  ∀*x*∈*X*, тогда ∀*x*∈*X* верно

,

т.е. точка *p* является проекцией точки *υ* на *X*.

**Определение 5.** Гиперплоскостью в *Rn* называется множество вида ,

где  – вектор ∈ *Rn,* .

**Свойства гиперплоскостей**

1. Это множество всегда *не пусто*: если, например *ci*≠ 0, то точка *x*0 с координатами ,  удовлетворяет равенству , т.е. .

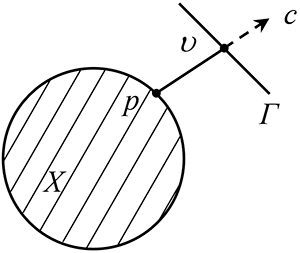
2. Пусть *x*0 – ∀(⋅) из *Гc,λ* , т.е. , тогда .

Известно, что два вектора *a*,*b*∈*Rn* – ортогональны, если (*a*,*b*)=0 ⇒ гиперплоскость *Гc,λ* состоит из тех и только тех точек *x*, для которых вектор *x*–*x*0 ортогонален вектору *с*. Вектор *с* называют *нормальным* вектором гиперплоскости *Гc,λ* .

3. В пространстве *Rn* гиперплоскость определяет два полупространства:

 и .

**Лемма 2** (Теорема отделимости).Для любого выпуклого и замкнутого множества *X* и любой точки *υ*, не принадлежащей множеству *X*, существует такая гиперплоскость *Г*, что  и  для ∀*x*∈*X*.

Очевиден геометрический смысл теоремы: существует проходящая через точку *υ* гиперплоскость *Г* такая, что *X* лежит в одном из полупространств, определенных *Г*.

*Доказательство.* Пусть *p* – проекция *υ* на *X*.

Определим , и рассмотрим

гиперплоскость ,

для которой выполняется первое утверждение леммы 2.

По лемме 1, если *p* – проекция, то ∀*x*∈*X* справедливо .

Поскольку точка *υ*∉*X*, то расстояние .

Итак, имеем для ∀*x*∈*X* : 

, *ч.т.д.*

**Теорема** (об опорной гиперплоскости).

В любой граничной точке *x*0 выпуклого множества существует *опорная гиперплоскость*, т.е. существует *c*≠0 и λ :

, и для всех *x*∈*X* .

|  |  |
| --- | --- |
| 33  Опорная гиперплоскость единственна (если существует касательная гиперплоскость, то она совпадает с опорной). И в этом случае опорная гиперплоскость единственная. | 34  Понятие опорной гиперплоскости – шире касательной. В точке *x*0 не существует касательной, но существуют опорные гиперплоскости, причем в качестве вектора *c* можно выбрать любой, лежащий между *c*1,*c*2. |

**Доказательство.**Рассмотрим последовательность {*υk*} – внешних точек относительно  (т.е. по определению сходимости ).

По лемме 2 (теорема отделимости) существует последовательность гиперплоскостей

, где  и .

Т.к. длину *ck* можно выбирать произвольно, то, не умаляя общности, можно считать, что . Не меняя обозначений, считаем, что .

Далее воспользуемся леммой Больцано-Вейерштрасса.

**Лемма**(Больцано-Вейерштрасса). Из любой ограниченной последовательности всегда можно извлечь такую частичную последовательность, которая сходилась бы к конечному пределу.

Рассмотрим .

Переходя к пределу в соотношении, определяющем *Гk*, получим гиперплоскость:

, где *λ*=(*с*, *x*0).

А, переходя к пределу в соотношении  ∀*x*∈*X*, получим:

 для ∀*x*∈*X* , *ч.т.д.* (равенство возникает, поскольку *x*0∈*X*, а (*c*, *x*0) = *λ*) .

**Теорема** (о разделяющей гиперплоскости).

Пусть *X*0 – множество внутренних точек выпуклого множества *X*; *Y* – выпуклое множество.

Если  (множество *Х*0 – не пусто) и  (не пересекается с другим множеством), то для множеств *X* и *Y* существует *разделяющая гиперплоскость*, т.е.

существует *c* ≠ 0 : ∀*x*∈*X*, ∀*y*∈*Y* справедливо соотношение (*c*,*y*) ≤ (*c*,*x*).

**Доказательство.**Рассмотрим множество .Это множество выпукло. Действительно,

.

Точка *z*=0 не является внутренней точкой множества *Z* (т.к. ).

Поэтому существует *с* ≠ 0: . Это неравенство справедливо:

- по теореме об опорной гиперплоскости, если точка *z*= 0 – граничная для множества *Z;*

- или по теореме о разделяющей гиперплоскости, если точка *z*= 0 – внешняя (тогда неравенство строгое).

⇒ .

Последнее неравенство остается справедливым и для ∀*y*∈*Y* и ∀*x*∈*X*, поскольку предельный переход не нарушает нестрогих неравенств, *ч.т.д.*

Введем два определения:

**Определение 1.** Точка *x* множества *X* (*x*∈*X*) называется *угловой* (*или крайней*) точкой, если в *X* не существует таких точек *x*′ и *x*″, *x*′ ≠ *x*″, что , при некотором *α*∈(0,1).

*Геометрически*: точка *x* – крайняя в *X*, если её нельзя поместить внутрь отрезка, концы которого лежат в *X*.

Например,

* у треугольника крайние точки – вершины;
* у луча – начало;
* у круга – все точки окружности;
* прямая, гиперплоскость – крайних точек не имеют.

**Определение 2.** Точка *x*0∈*X* называется *выпуклой комбинацией* точек *x*1,…, *xn*∈*X*, если существуют , существуют *xi*∈*X, i=1,…N,* такие, что:



**Теорема** Крейна–Мильмана (о представлении).

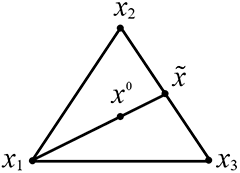
Пусть *X* – выпуклое, замкнутое, ограниченное множество, тогда ∀*x*0∈*X* может быть представлена в виде выпуклой комбинации конечного числа угловых точек множества, т.е. ∀*x*0∈*X* существуют *αi* ≥ 0, существуют *xi*∈*X* – угловые точки:



**Доказательство.** Индукция по размерности пространства *n*.

При *n*= 1 *X* – отрезок ⇒ утверждение теоремы очевидно.





Предположим, что для *n*= *k* – 1 теорема справедлива.

Пусть *X*∈*Rk*. Возможны два случая:

1) *x*0 – граничная точка *X*.

Построим в этой точке гиперплоскость, опорную к *X* (существует по теореме об опорной гиперплоскости):

, где *λ*= (*c*, *x*0) – опорная гиперплоскость.

Рассмотрим множество *X*0 = *X*∩ *Гc*,*λ..* Оно, как пересечение выпуклого, замкнутого, ограниченного множества *X* с выпуклым замкнутым множеством *Гc*,*λ*, само выпукло, замкнуто и ограничено.

Кроме этого, .

По индукционному предположению существуют  – угловые точки *X*0:



Покажем, что  являются угловыми точками и для множества *X*.

Предположим противное, т.е. что некоторая точка *xi* не является угловой для множества *X*. Это означает, что существует  и .

Т.к. , то



и т.к. *Гc,λ* – опорная к *X*, то

 (\*)

Поскольку 0 < *α*< 1, можно записать:

⇒.

Из последнего соотношения следует, что (*c*,*x*′) ≥ (*c*,*x*0), но

,

Поскольку .

Аналогично можно показать, что *x*″∈*X*0 ⇒ противоречие с тем, что *xi* – угловая точка *X*0  ⇒ *xi* – угловая точка *X*, *ч.т.д*

2) Пусть теперь *x*0 – внутренняя точка множества *X*. Проведем через *x*0 прямую *l*. Пересечение *l*∩ *X* является отрезком с концами *x͂* и *x͌*, принадлежащими границе множества *X*, и, поскольку *x*0 – внутренняя точка *X* ⇒ существует *α*∈(0,1):

.

Поскольку для граничных точек *x͂* и *x͌* теорема верна, то верна она и для *x*0. Действительно, для граничных точек имеют место соотношения:



,

где все *yi* и *zi* – угловые точки множества *X*.

А тогда , *ч.т.д.*

*Замечание*: Можно доказать, что в указанном представлении число угловых точек не превосходит величины *n*–размерности пространства.

**Линейное программирование**

Задача линейного программирования:

(1)

Здесь *cj*, *aij*, *bi* – заданные числа, причем не все *cj* и *aij*= 0.

Это – задача линейного программирования (ЗЛП) со смешанными ограничениями. К задачам такого вида (1) сводятся многие прикладные задачи технико-экономического содержания. Из общей задачи линейного программирования обычно выделяют и исследуют два её подкласса – *основную* задачу и *каноническую*.

Если *k*= *m* (только ограничение неравенства) и *s*= *n* (прямые ограничения) накладываются на все элементы вектора, то это *основная (стандартная) форма ЗЛП*.

(2)

*A*– *m*× *n* – матрица условий; *b*– *m* ­– вектор ограничений.

Если *k*= 0 (только ограничения равенства) и *s*= *n*, то это – *каноническая форма ЗЛП*.

(3)

Задачи в формах (1), (2), (3) могут быть сведены друг к другу, т.е. приведены к эквивалентной задаче (с тем же множеством решений).

*Сведение* (1) *к* (2):

Обозначим  – множество всех ограничений;

 – множество ограничений неравенств;

 – множество ограничений равенств.

Аналогично:

Обозначим ;

 – множество прямых ограничений;



Заметим, что задача (1) приводится к виду:

Здесь:





*Идея*: Нужно (*m* – *k*)–равенств заменить неравенствами:



Ввести (*n*– *s*)–прямых ограничений:



Т.о., делаем замену переменных , где элементы вектора



Тогда задача (1) приводится к виду:

(2)

Это задача в основной форме (2) с размерностью  и числом ограничений

.

*Сведение* (2) *к* (3)*:*

Введем *m*–дополнительных переменных и рассмотрим задачу в пространстве *Rn*+*m*:

;

Тогда (2) можно записать в виде:

(3′)

Эта задача в канонической форме (3) с размерностью  и *m*-ограничениями.

Нетрудно убедиться в том, что множество решений, рассмотренных выше задач совпадают, либо пусты одновременно.

**Геометрическая интерпретация основной задачи линейного программирования**

Рассмотрим задачу ЛП в форме (2), т.е. основную ЗЛП:

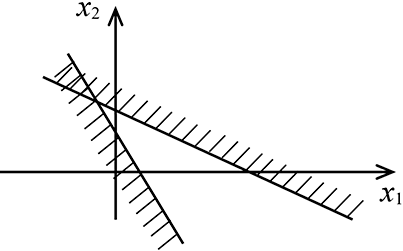
Пусть ⇒ задача сводится к виду:



Введем множества:

 – положительные квадрант плоскости.

 – полуплоскость, образованная прямой .

Ясно, что множество *X* является пересечением множеств  и возможны следующие случаи:

1) Может случиться, что это пересечение пусто, тогда задача теряет смысл (а).

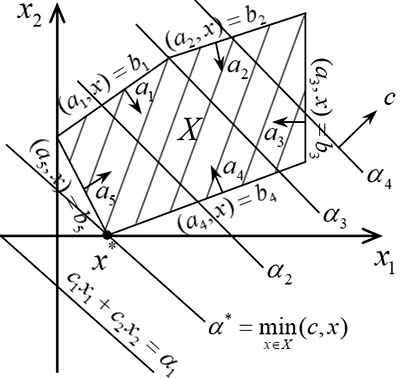
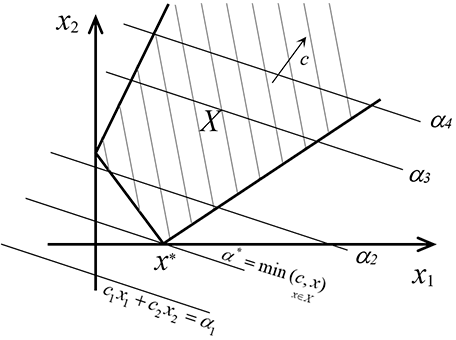
(a)

2) Если множество *X* не пусто, то оно образовано пересечением конечного числа полуплоскостей ⇒ множество *X* представляет собой *выпуклое многоугольное множество*, границей которого является ломаная, составленная *из отрезков каких-либо координатных осей* и *прямых* . Это многоугольное множество может быть ограниченным (выпуклый многоугольник) (б) и неограниченным (в).

Рассмотрим уровни минимизируемой функции, т.е. .

При изменении *α* от –∞ до +∞ прямая, перемещаясь параллельно самой себе, "зачертит" всю плоскость. При этом направление вектора *с* задает движение линии уровня по направлению возрастания функции (*c, x*).

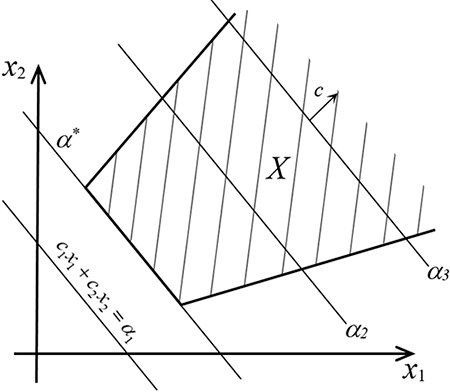
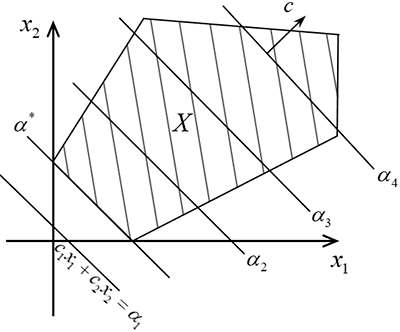
Если *X* – многоугольник (б), то при изменении *α* от –∞ до +∞ прямая, соответствующая линии уровня, при некотором значении *α*\* впервые коснется *X* (выпуклого многоугольника) и будет иметь с этим множеством *X* общую точку *x*\*, т.е. *x*\* – решение задачи.



(в)

(б)

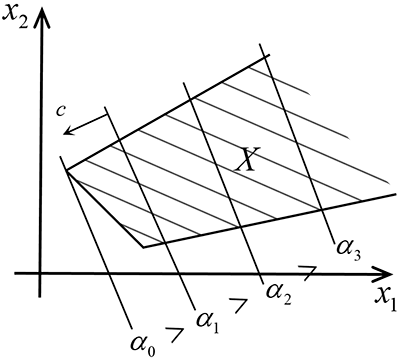
Возможен случай, когда при первом касании линией уровня множества *X*, общей окажется целая сторона многоугольника, тогда решением будет целая прямая. Это может случиться, когда множество *X* имеет сторону, перпендикулярную вектору *c* ((г) и (д)).



(г)

(д)

Если многоугольное множество *X* не ограничено, то возможна ситуация, когда прямая линия уровня при всех  имеет общую точку с множеством *X* (е), тогда . В этом случае первого касания с прямой нет – задача не имеет решения.



(е)

Из рассмотренных случаев ясно, что ЗЛП может не иметь ни одного решения ((а) и (е)), может иметь единственное решение ((б) и (в)) и, наконец, может иметь бесконечное множество решений (линия уровня параллельна одной из граней допустимого множества).

На примере рассмотренной выше ЗЛП нетрудно увидеть, что, *если задача имеет решение*, то среди решений найдется хотя бы одна угловая точка многоугольного множества . Ниже мы увидим, что это не случайно ­– и в более общей ЗЛП, оказывается, нижняя грань минимизируемой функции достигается на *X* в угловой точке множества.

Итак, рассмотрим основную задачу линейного программирования, т.е. задачу в форме (2):



**Теорема.**Допустимое множество в задаче (2) – выпукло и замкнуто.

**Доказательство.**

*Замкнутость.* Рассмотрим последовательность

, т.е.  для ∀*k*.

Пусть (непрерывность линейной функции).

Переходя к пределу в неравенстве: , получим , и соответственно,

,

т.е. множество содержит свои предельные точки, следовательно, оно замкнуто.

*Выпуклость.* Пусть . Это означает, что



Рассмотрим  и покажем, что  , т.е. множество Х - выпукло.

Действительно:

.

Аналогично:

, *ч.т.д.*

*Теорема доказана*.

**Теоремы об оптимальных точках основной задачи линейного программирования**

**Теорема 1*.*** Если допустимое множество задачи (2) не пусто (*X* ≠ ∅) и целевая функция ограничена снизу на *X*, то существует *x*\*– оптимальная точка, причем *x*\* лежит на границе множества *X*.

**Доказательство.** (индукция по *n*)

а) Пусть *n*= 1. В этом случае утверждение теоремы очевидно:

допустимое множество – отрезок  или луч, а целевая функция - *ϕ*(*x*)­ = *cx*.

Если *X* – отрезок, то min достигается и находится на границе.

Если *X* – луч, то *c*> 0 (в силу ограниченности *ϕ*(*x*) снизу на *X)* ⇒ min достигается на границе луча.

б) Пусть теорема верна для *n* – 1, и докажем её для *n*.

Рассмотрим грани допустимого множества *X*:

 – ограничения - неравенства;

и "координатные" грани:

– прямые ограничения.

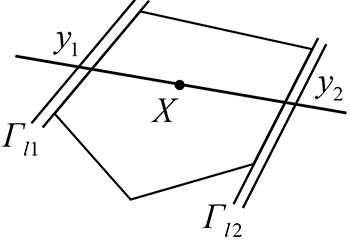
Т.к. , *Гl* – линейное пространство размерности (*n*– 1), выпуклое замкнутое множество, то по индукционному предположению имеем (*m*+ *n*) - точек (на ∀ грани – по одной), где *ϕ*(*x*) достигает min.

Пусть *x*\* – одна из них, где функция минимальна. Докажем, что *x*\* – оптимальная точка на множестве *X*.

Предположим *противное*, пусть существует .

Можно утверждать, что точка *x* – внутренняя точка множества *X*. Действительно, если бы она была граничной, то для нее не могло бы выполняться .

А тогда через точку *x* можно провести прямую, пересекающую две граничные гиперплоскости ⇒ существуют  такие, что:



*y*1,*y*2 – граничные и 

⇒ существует и существует  ⇒** – по выбору точки *x*\*.

Получаем:

 – противоречие!

*Теорема доказана*.

**Теорема 2.**Если допустимое множество задачи (2) не пусто и целевая функция задачи (2) ограничена снизу на допустимом множестве *X*, то среди оптимальных точек задачи линейного программирования есть крайняя (угловая).

**Доказательство.**

Пусть *x*\* – оптимальная точка ЗЛП (она существует по теореме 1).

1) Предположим, что *X* – ограниченное множество. Но *X* – выпукло и замкнуто (по доказанной выше теореме). Тогда по теореме Крейна-Мильмана (о представлении), точка *x*\* может быть представлена в виде выпуклой комбинации конечного числа угловых точек множества *X*, т.е. существуют *y*1,…, *yk* – крайние точки в *X* и существуют *λ*1,…, *λk≥0*, такие, что:



Поскольку *ϕ*  - линейная функция, можно записать:

.

Докажем, что в этом случае , т.е. тем самым мы докажем, что среди оптимальный точек *x*\* есть крайние - 

Предположим, что  (\*)

Тогда: 

⇒ с учетом (\*) имеем .

Далее будет доказывать по индукции.

Пусть для некоторого , справедливо: .

Докажем, что .

Действительно,

Min Min среди суммируемых по (\*)

Тогда, перенеся в левую часть неравенства, получим

,

а тогда, с учетом (\*), имеем

, *ч.т.д.*

2) Предположим, что *X* – неограниченное множество, а *x*\* – оптимальная точка ЗЛП.

Рассмотрим вектор

.

Пусть . Введем новое ограничение .

Тогда точка *x*\* лежит в новом допустимом множестве, при этом, т.к. , то новое допустимое множество ограничено. ⇒ По доказанному выше существует :

.

Осталось доказать, что существует , т.е. крайняя оптимальная точка не лежит на новой гиперплоскости, т.е. *zj* – крайняя точка множества *X* и оптимальна.

Имеем:

, *ч.т.д.*

(Условие того, что точка *x*\* лежит на новой гиперплоскости (*d*,*x\**) = *μ*+ 1). *Теорема доказана*.

Итак, *для решения задачи линейного программирования надо искать крайние точки*. Ранее было введено геометрическое определение крайней точки. Для того чтобы уметь находить её, следует ввести её *алгебраическое её определение*.

**Характеристика крайних точек**

Рассмотрим основную форму ЗЛП, когда допустимое множество *X* имеет вид:

 – всего (*m* + *n*) ограничений.

**Определение.**Если в точке *x* для некоторых ограничений выполняются равенства {(*Ai*, *x*) = *bi* или *xj* = 0 }, то они называются *активными*, остальные ограничения называются *пассивными*.

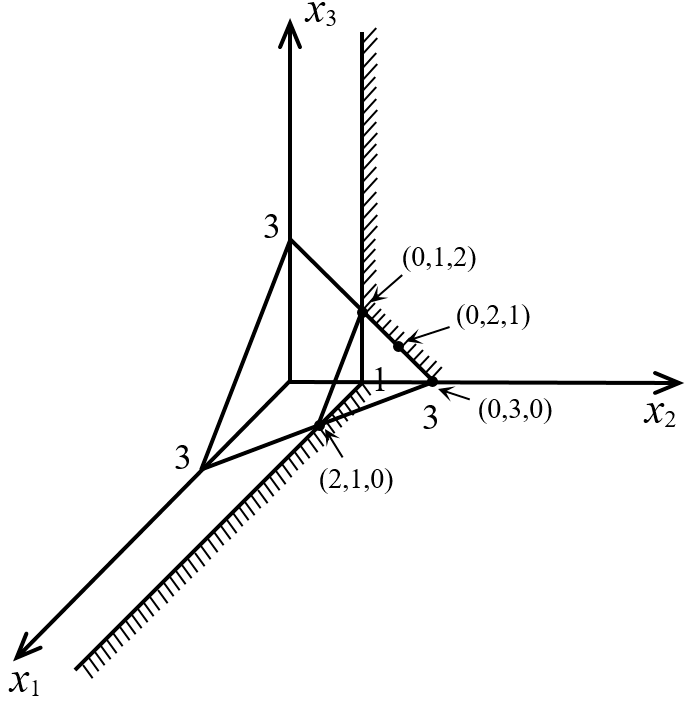
Рассмотрим матрицу ограничений: 

Для ∀*x*∈*X* определим множество индексов активных ограничений:



Обозначим через *Bx* матрицу, составленную из векторов-строк матрицы *B*, соответствующую активным ограничениям точки *x*∈*X*.

**Пример.** Пусть , где *m* = *2*, *n* = *3*;



Матрица *B* имеет вид: 

Единичная матрица, соответствующая прямым ограничениям

(0, 1, 2) – крайняя точка, для неё:



(2, 1, 0) – крайняя точка:



(0, 3, 0) – крайняя точка:



(0, 2, 1) – не крайняя точка:



Для внутренних точек *x* матрицы *Bx* будет состоять из пустых строк.

**Теорема*.*** Для того чтобы допустимая точка *x* задачи (2) была крайней, необходимо и достаточно, чтобы ранг матрицы *Bx* был максимальным, т.е. равен *n*, т.е. *x*∈*X* – крайняя точка ⇔ *rang Bx* = *n*.

**Доказательство.**

*Достаточность.* Пусть . Это означает, что из соотношения , следует, что *z*= 0. Докажем, что тогда точка *x*∈*X* – крайняя.

Предположим обратное, т.е. пусть существует , где *λ*∈(0,1)

Обозначим  (т.е. вектор *bx* состоит из компонент вектора *b* и нулей).

Т.к. *x*1,*x*2 – допустимые точки, то  и сложим



По определению *bx* в этом соотношении должно быть равенство

⇒ 

Аналогично можно показать, что  ⇒ *x*=*x*2.

Значит наше предположение не верно, точка *x* крайняя, *ч.т.д.*

*Необходимость.* Пусть точка *x* – крайняя. Предположим, что , т.е. существует .

Зафиксируем такую точку *z*∈*X* и при малом *ε* > 0 рассмотрим точки: 

|  |  |
| --- | --- |
| Для *x*1 и *x*2 выполняются активные ограничения | По определению точки *z* имеем:    ( т.к. для крайней точки справедливо  и по предположению ) Отсюда следует, что активные ограничения для точки *x*1 и точки *x*2 верны. |

|  |  |
| --- | --- |
| Для *x*1 и *x*2 выполняются пассивные ограничения | *ε* можно выбрать таким образом, чтобы и пассивные ограничения сохранялись: , , значит можно выбрать *ε* такимобразом,  чтобы выполнялось неравенство  Аналогично можно показать, что и прямые пассивные ограничения сохраняются: |
|  |  |

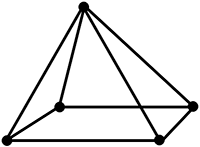
Это означает, что *x*1,*x*2 – допустимые точки. Но  – противоречит тому, что *x* – крайняя точка. Т.е. не существует такого вектора , *ч.т.д*.

**Замечания.**

1. В крайней точке пересекаются не менее чем *n* гиперплоскостей (граней допустимого множества). Для *n* = 2 – две прямых, для *n* = 3– три плоскости и т.д.

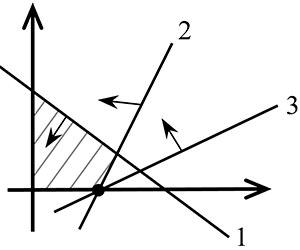
**Определение.**Крайняя точка называется *невырожденной*, если она лежит на пересечении ровно  гиперплоскостей. В противном случае она называется *вырожденной*. Задача линейного программирования, у которой существует вырожденная точка, также называется вырожденной.

*Примеры вырожденных точек*:



вершина 4-х

угольной пирамиды



вырожденная точка

Далее, если не оговорено особо, будем рассматривать невырожденные задачи ЛП.

2. Т.к. ограничений (*m*+ *n*), а в крайних точках сходится *n*-гиперплоскостей, то всего *крайних точек* может быть не более  – *конечное число*. (Число сочетаний из n+m по n).

Доказанные выше теоремы (о существовании решения ЗЛП и алгебраической характеристике её крайних точек) означают, что для поиска решения основной задачи ЛП, *достаточно перебрать лишь крайние точки допустимого множества X*, число которых *конечно*. Крайние точки могут быть найдены с использованием теоремы о характеристике крайних точек за конечное число арифметических операций.

*Выводы из 3-х теорем.*

1. *Справедлива следующая альтернатива:*

*- либо целевая функция на допустимом множестве не ограничена снизу;*

*- либо существует крайняя оптимальная точка.*

*2. Количество крайних точек допустимого множества конечно.*

Таким образом, вышедоказанные теоремы обосновывают принципиальную возможность решения задачи ЛП за конечное число шагов методом полного перебора крайних точек.

Однако "конечное" – не значит "малое". При сколько-нибудь больших *m* и *n* этот простой метод требует огромной вычислительной работы.

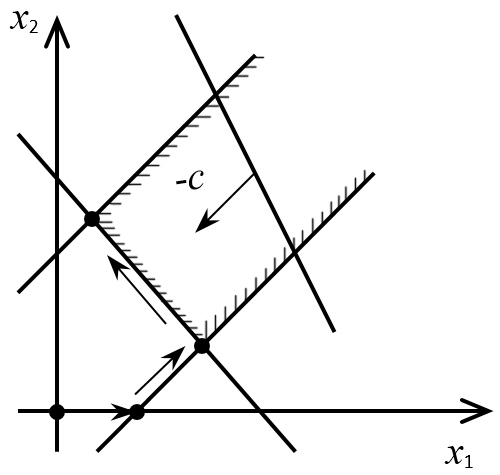
Следовательно, естественным образом подходим к основной идее *симплекс-метода* – полный перебор следует заменить упорядоченным, разумным.

Название метода связано с тем, что он впервые разрабатывался применительно к ЗЛП, в которых множество *X* представляло собой симплекс в . Затем метод был обобщен на случай более общих множеств *X*, но первоначальное название так и сохранилось. В литературе его ещё называют *методом последовательного улучшения плана*.

Итак, *симплекс метод* – это алгоритм преобразования таблицы (состоящей из коэффициентов *aij*, *bj*, *c­j*), основанных на *методе Жордановых исключений* при решении системы линейных уравнений.

Метод, состоит из 2-х этапов:

1. *поиск крайней точки*, в результате которой могут быть три ситуации:



* крайней точки нет (т.е. *Х* = ∅);
* крайняя точка не найдена;
* крайняя точка найдена, в этом случае начинается второй этап:

1. *перебор крайних точек и поиск оптимальной*:

* оптимальной точки нет (т.е. *ϕ*(*x*) не ограничена снизу на *X*);
* оптимальная точка не найдена;
* оптимальная точка найдена, на этом алгоритм симплекс-метода кончается.

*Переход к следующей* крайней точке в поисках оптимальной осуществляется *исходя из предположения, что значение целевой функции уменьшается*.

Поэтому, поскольку:

* число крайних точек конечно и среди них обязательно есть решение задачи (если оно существует),
* возврат к уже просмотренным точкам невозможен, то за *конечное число* итераций эта процедура приведет к решению, либо к выводу о том, что *X* = ∅.

Теоретически не исключается ситуация, когда метод пройдется по всем крайним точкам множества *X* (и такие патологические примеры построены). Однако, как показывает практика, для большинства задач количество итераций симплекс-метода находится в пределах от *m* до *2m*.

**Алгоритм симплекс-метода решения основной ЗЛП**

Рассмотрим основную ЗЛП:

,

*A*–(*m*× *n*) – матрица, *b*–(*m* × 1) – вектор

,

*Ai*–(1 × *n*) – строка, *bi* – число, *i*∈1,…, *m*.

Введем *m*-переменных: .

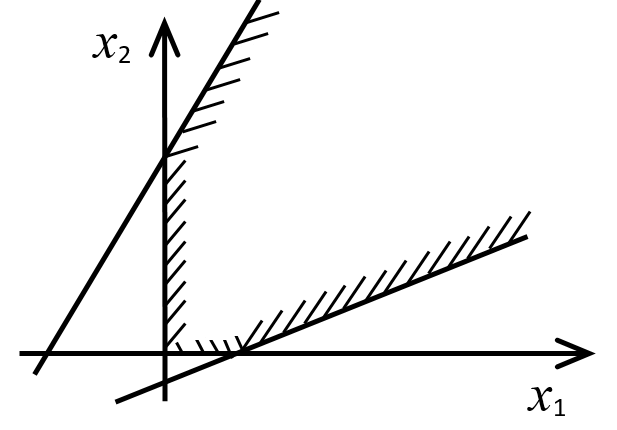
Итак, допустимое множество *X* ограниченно (*m* + *n*) – гиперплоскостями: .

1. **Алгоритм поиска крайней точки**

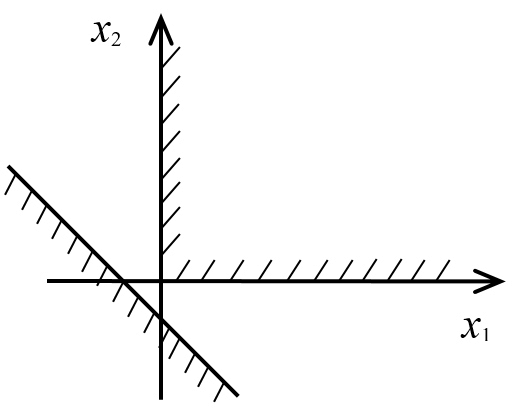
Имеем систему:



Т.к. в крайней точке неравенство заменяется равенством, то используем метод Жордановых исключений для решения системы линейных уравнений.



1. Если ∀*i* –*bi* > 0, то крайняя точка найдена, это точка *х* = 0.
2. Если существуют *s*: –*bs*< 0, торассмотрим коэффициенты *as*1,…, *asn*. Если все они ≤ 0, то





⇒ допустимое множество пусто, крайних точек нет.

1. Иначе: при  существует . Тогда делается один шаг Жордановых преобразований, который состоит в замене координат . Из s-го уравнения:

.

Подставим *xr* в другие уравнения системы:

,

где

 (1)

В новом *s*-м уравнении:

 (2)

В табличной форме:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | *Внебазисные*  *переменные* | | |  |
|  | *x*1 | *xr* | *xn* | –*b* | → |  | *x*1 | *ys* | *xn* | –*b* |
| *Базисные переменные* | *y*1 | *a*11 | *a*1*r* | *a*1*n* | –*b*1 | *y*1 | *ã*11 | *ã*1*r* | *ã*1*n* | –*b̃*1 |
| *ys* | *as*1 | *asr* | *asn* | –*bs* | *xr* | *ãs*1 | *ãsr* | *ãsn* | –*b̃s* |
| *ym* | *am*1 | *amr* | *amn* | –*bm* | *ym* | *ãm*1 | *ãmr* | *ãmn* | –*b̃m* |
|  |  |  |  |  |  |  |  |  |  |  |  |

1) Обратим внимание на то, что теперь

.

**Определение.** Элемент *asr* называется *разрешающим*.

Зависимые координаты  (в левом столбце), называются *базисными*.

Независимые координаты  (в верхней строке), называются *внебазисными*.

Один шаг Жордановых исключений – это замена базиса. *Крайняя точка найдена: все независимые координаты равны* 0, *все* .

2) Для того чтобы последовательно приближаться к крайней точке, необходимо чтобы ∀ шаг увеличивал (не уменьшал!) число положительных компонент вектора *b*.

Для этого, в пункте в) фиксируем столбец *r* (для которого при существует ) и в нём выбирается такая строка *s*, т.е. такой разрешающий элемент *asr*, чтобы

,

или *отрицательное отношение было бы максимальным среди всех отрицательных отношений.*

Покажем, что при таком выборе разрешающего элемента, число положительных компонент вектора *b* не уменьшится.

1. Если компонента корректируется по формуле (2), то, т.к.  имеем .

(показали ранее).

1. Пусть компонента вектора *b* корректируется по формуле (1). Тогда, нас интересует случай, когда . Хочется, чтобы .
   * пусть  и 

, *ч.т.д.*

* + пусть  и 

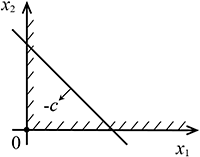
, *ч.т.д.*

Таким образом, алгоритм сходится к крайней точке за конечное число шагов в предположении, что среди крайних точек нет вырожденных. На практике это означает, что ∀*i* *bi*≠ 0.

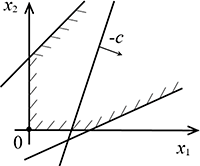
1. **Алгоритм поиска оптимальной точки**

Продолжаем преобразование таблицы. Т.к. от одного базиса всегда можно перейти к другому, то считаем, что в начале этого этапа (в верхней строке) находятся независимые координаты *x*1,…, *xn*, при этом, точка *x* = 0 (т.е. ) – крайняя.

1. Если ∀*j* *cj* ≥ 0, то *оптимальная точка найдена*. Это точка *x*= 0. Действительно  достигает min в нуле.



1. Если существует , то рассмотрим коэффициенты *alr*,…, *amr*. Если все они ≥ 0, то оптимальной точки нет. *Целевая функция не ограничена снизу на* *X*. Действительно, для ∀*i* в неравенстве , *xr* может неограниченно расти. С другой стороны, при росте *xr*, *ϕ*(*x*) – убывает .



1. Иначе: при  существует . Делается один шаг Жордановых исключений, т.е. меняется базис относительно разрешающего элемента . Предположим, что целевая функция имела вид:

,

подставим в нее *xr,* выраженный из *s*-го уровня , где

 (3)

Остальные элементы вычисляются по формулам (1) и (2).

В табличной форме:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *x*1 | *xr* | *xn* | –*b* | → |  | *x*1 | *ys* | *xn* | –*b* |
| *y*1 | *a*11 | *a*1*r* | *a*1*n* | –*b*1 | *y*1 | *ã*11 | *ã*1s | *ã*1*n* | –*b̃*1 |
| *ys* | *as*1 | *asr* | *asn* | –*bs* | *xr* | *ãs*1 | *ãsr* | *ãsn* | –*b̃s* |
| *ym* | *am*1 | *amr* | *amn* | –*bm* | *ym* | *ãm*1 | *ãmr* | *ãmn* | –*b̃m* |
| *ϕ* | *c*1 | *cr* | *cn* | *α* | *ϕ* | *с̃*1 | *с̃r* | *с̃n* | *α̃* |

Необходимо обеспечить, чтобы ∀ следующий шаг преобразований не ухудшал достигнутого, т.е. чтобы правый столбец таблицы оставался положительным (–*bi* > 0) и чтобы функция *ϕ* – уменьшалась (количество положительных компонент вектора "*c*" может меняться).

Для того, чтобы двигаться по крайним точкам к точке min функции *ϕ*, аналогично случаю поиска крайней точки, в пункте в) фиксируют столбец *r* и в нем выбирают такую строку *s*, т.е. такой разрешающий элемент *asr*, чтобы

.

Выше было показано, что при таком выборе *asr* (выше) количество положительных компонент вектора *b* не уменьшается.

При этом, элемент , а значение функции 

, *ч.т.д.*

Таким образом, алгоритм выбора оптимальной точки также *сходится к оптимальной точке за конечное число итераций*, для невырожденной ЗЛП (среди компонент вектора "*b*" не должно быть нулевых).

Координаты оптимальной точки определяются следующим образом:

* если *xj* находится на *i*-ом месте левого столбца, то его значение равно *bi*;
* если *xi* находится на *j*-ом месте верхней строки, то его значение равно 0.

Далее рассмотрим несколько примеров:

Представим формулы для пересчета таблицы в более компактном виде. Итак,

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *x*1 | *xr* | *xn* | –*b* | → |  | *x*1 | *ys* | *xn* | –*b* |
| *y*1 | *a*11 | *a*1*r* | *a*1*n* | –*b*1 | *y*1 | *ã*11 | *ã*1*r* | *ã*1*n* | –*b̃*1 |
| *ys* | *as*1 | *asr* | *asn* | –*bs* | *xr* | *ãs*1 | *ãsr* | *ãsn* | –*b̃s* |
| *ym* | *am*1 | *amr* | *amn* | –*bm* | *ym* | *ãm*1 | *ãmr* | *ãmn* | –*b̃m* |
| *ϕ* | *c*1 | *cr* | *cn* | *α* | *ϕ* | *с̃*1 | *с̃r* | *с̃n* | *α̃* |

Для элементов "разрешающей" строки:



Для элементов "разрешающего" столбца:



Для остальных элементов:



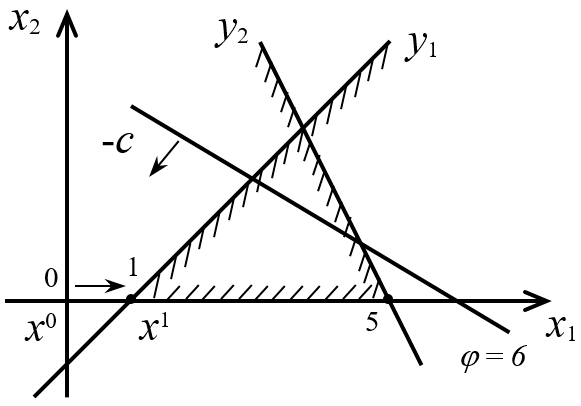
**Примеры.**

**Пример 1**

1. *Целевая функция имеет вид:* 



Приводим к основному виду задачи линейного программирования (ЗЛП) 



⇒

Рис.1

На рис.1 представлено графическое решение задачи с определением допустимого множества и линии уровня целевой функции. Как видно из рис 1, решение задачи достигается в точке (1,0). Решим задачу с помощью алгоритма симплекс-метода.

Cоставляем таблицу: Начинаем решение с точки х1= 0, х2=0.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *x*1 | *x*2 | –*b* | Точка (0,0) – не крайняя ⇒ ищем разрешающий элемент.  Для этой точки существует элемент а11 ⇒ фиксируем первый столбец и рассмотрим отрицательные величины  и выберем среди них максимальное  и рассмотрим  и .  Разрешающий элемент - а11. |
| *y*1 | 1 | –1 | –1 |
| *y*2 | –2 | –1 | 10 |
| *ϕ* | 1 | 2 | 0 |
|  |  |  |  |

⇒ Делаем первый шаг преобразований:

1. Для 1-ой строки имеем:







1. Для 2-ой строки имеем:







1. Для коэффициентов ϕ при целевой функции имеем:







⇒ получаем таблицу вида:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *y*1 | *x*2 | –*b* | Пришли в точку х1=1, х2=0.  ⇒ поскольку ∀*i* *bi* > 0 точка (1,0) – крайняя,  и поскольку ∀*r*  точка (1,0) – оптимальная, при этом  *ϕ*min =1. |
| *x*1 | 1 | 1 | 1 |
| *y*2 | –2 | –3 | 8 |
| *ϕ* | 1 | 3 | 1 |

1. **Пример 2**

*Целевая функция имеет вид:* 



Приводим к основному виду задачи линейного программирования:



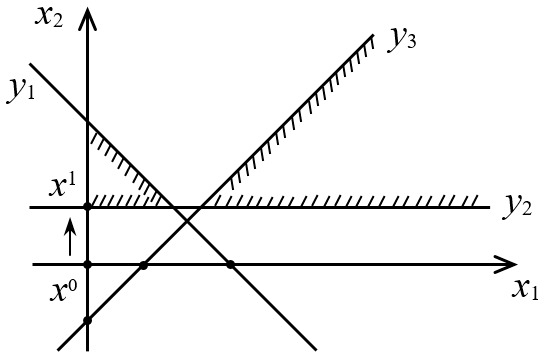


Рис.2

На рис.2 представлено графическое решение задачи с определением допустимого множества и линии уровня целевой функции. Как видно из рис 2, допустимое множество - пусто. Решим задачу с помощью алгоритма симплекс-метода.

Составим таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *x*1 | *x*2 | –*b* | Начинаем решение с точки х1= 0, х2=0.  Ищем крайнюю точку. Начинаем решение с точки х1= 0, х2=0.  Можно продолжать её искать, поскольку для  существует  и для  существует .  Выбираем ∀ из *b*2 или *b*3. |
| *y*1 | –1 | –1 | 2 |
| *y*2 | 0 | 1 | –1 |
| *y*3 | 1 | –1 | –1 |
| *ϕ* | 1 | 0 | 0 |

Допустим, выбрали *b*2, тогда в столбце 2 рассмотрим соотношения:

,  – max среди отрицательных ⇒ разрешающий элемент *a*22.

⇒ Делаем первый шаг преобразований:

1. Для 2-ой строки:







1. Для 1-ой строки:







1. Для 3-ей строки:







1. Для коэффициентов *ϕ* имеем:







⇒ приходим к таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *x*1 | *y*2 | –*b* | Пришли в точку (х1=0,х2=1)  Продолжаем искать крайнюю точку.  Её можно продолжать искать, т.к. существует для .  фиксируем 1-й столбец и рассмотрим в нем отношения:  , ­– min ⇒ *a*11 – разрешающий элемент |
| *y*1 | –1 | –1 | 1 |
| *х*2 | 0 | 1 | 1 |
| *y*3 | 1 | –1 | –2 |
| *ϕ* | 1 | 0 | 0 |

Делаем второй шаг преобразований:

1. Для 1-ой строки имеем:







1. Для 2-ой строки имеем:







1. Для 3-ей строки имеем:







1. Для коэффициентов *ϕ* имеем:







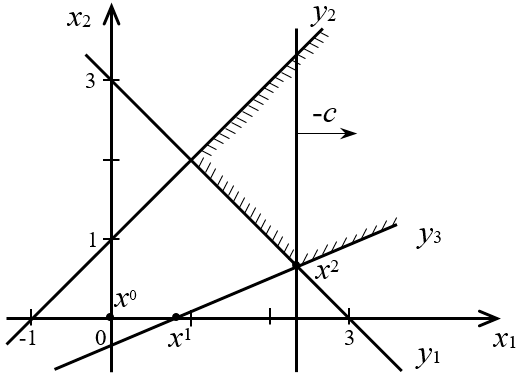
⇒ приходим к таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *y*1 | *y*2 | –*b* | Пришли в точку х1=1, х2=1.  В этой строке все элементы  при *bi*<0  ⇒ допустимое множество пусто. |
| *x*1 | –1 | –1 | 1 |
| *x*2 | 0 | 1 | 1 |
| *y*3 | –1 | –2 | –1 |
| *ϕ* | –1 | –1 | 1 |

1. **Пример 3**

*Целевая функция имеет вид:* 





Приводим к основному виду задачи линейного программирования:

 Рис.3

На рис.3 представлено графическое решение задачи с определением допустимого множества и линии уровня целевой функции. Как видно из рис 3, целевая функция неограничена на допустимом множестве. Решим задачу с помощью алгоритма симплекс-метода.

Составим таблицу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *x*1 | *x*2 | –*b* | Начинаем решение с точки х1= 0, х2=0.  Точка (0,0) – не крайняя и можно продолжать искать крайнюю точку, т.к. для –*b*1 = –3 существует *a*11 и *a*12 > 0 ⇒ выбираем ∀ из них, например 1-й столбец.  Рассмотрим отношения  ⇒ *a*31– разрешающий элемент |
| *y*1 | 1 | 1 | –3 |
| *y*2 | 1 | –1 | 1 |
| *y*3 | –1 | 2 | 1 |
| *ϕ* | –1 | 0 | 0 |

⇒ Делаем первый шаг преобразований:

1. Для 3-ей строки имеем:







1. Для 1-ой строки имеем:







1. Для 2-ой строки имеем:







1. Для коэффициентов *ϕ* имеем:







|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *y*3 | *x*2 | –*b* | Пришли в точку х1=1, х2=0  Точка (1,0) – не крайняя (есть отрицательный элемент).  Крайнюю точку можно искать, т.к. для –*b*1 = –2 существует .  ⇒ *a*12 – разрешающий элемент. |
| *y*1 | –1 | 3 | –2 |
| *y*2 | –1 | 1 | 2 |
| *x*1 | –1 | 2 | 1 |
| *ϕ* | 1 | –2 | –1 |

⇒ Делаем второй шаг преобразований:

1. Для 1-ой строки имеем:







1. Для 2-ой строки имеем:







1. Для 3-ей строки имеем:







1. Для коэффициентов *ϕ* имеем:







 приходим к таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *y*3 | *y*1 | –*b* | Пришли в точку х1=  , х2=  ⇒ Точка  – крайняя, но  в этом столбце при  все *aij* > 0 ⇒ оптимальной точки нет, inf*ϕ* = –∞ (целевая функция не ограничена на допустимом множестве) |
| *x*2 |  |  |  |
| *y*2 |  |  |  |
| *x*1 |  |  |  |
| *ϕ* |  |  |  |

**Выпуклое программирование**

Рассмотрим задачу математического программирования следующего вида:

(\*) 

**Определение.** Если в задаче (\*) целевая функция *ϕ*(*x*) – выпуклая и допустимое множество *X* – выпукло, то задача (\*) называется задачей выпуклого программирования.

Рассмотрим теперь задачу математического программирования следующего вида:

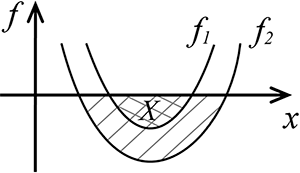
(\*\*)

Покажем, что допустимое множество *X* задачи (\*\*) – выпукло.

Действительно, пусть *x*1, *x*2∈*X*, *α*∈(0,1)

⇒ покажем, что , т.е. ∀*i* *fi*(*x*) ≤ 0.

Имеем, для ∀*i* 

⇒ т.о. ∀*i fi*(*x*) ≤ 0 ⇒ точка *x*∈*X*, т.е. *X* – выпукло.

"Надграфик" выпуклой функции, т.е. множество

 – выпуклое множество.

**Определение.** Задача (\*\*) называется основной задачей выпуклого программирования (ОЗВП).

**Свойства выпуклых функций**

1. **Неравенство Йенсена**

Пусть *f*(*x*) – выпуклая функция на выпуклом множестве *X*. Тогда

 при всех ;

**Доказательство.**Индукция по *m*. Пусть *m* = 1 ⇒ очевидно.

Пусть для *m*= *k* утверждение верно.

Докажем для *m* = *k* + 1.

Пусть .

Если , то .

Если  ⇒ представим *x* в следующем виде:

, где .

Тогда:

, *ч.т.д.*

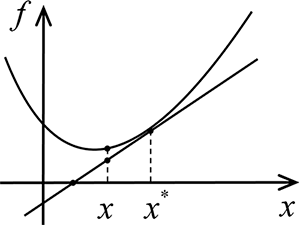
1. Пусть *f*(*x*) – выпуклая на выпуклом множестве ** функция. Тогда любой её локальный минимум на множестве *X* является одновременно и глобальным (доказательство было ранее в лекциях).
2. Пусть *f*(*x*) – выпуклая функция на выпуклом множестве , дифференцируемая в точке *x*\*∈*X*. Тогда .

График *f* лежим не ниже касательной гиперплоскости к графику функции *f* в точке (*x*\*, *f*(*x*\*)).

(*Напоминание: график линейной функции*  *называется касательной гиперплоскостью к графику функции f в точке* (*x*\*, *f*(*x*\*))).

**Доказательство.**По определению выпуклой функции для ∀*x*, *x*\*∈*X*, *λ*∈[0,1] имеем:

.

Преобразуя эту формулу, имеем:



Переходя к пределу при *λ* → 0, имеем искомое соотношение.

1. Пусть *f* – дважды непрерывно дифференцируемая функция на выпуклом множестве Тогда *f* выпукла на *X* ⇔ матрица Гессе *f*″ неотрицательно определена, т.е. ∀*x*\*∈*X*, ∀*h*\*∈*Rn* .

Был ранее без доказательства критерий сильной выпуклой функции *f* с параметром ∅≥0:



**Доказательство.**

*Необходимость.* Пусть *f* – выпукла на *X*.

а) Сначала предположим, что (*x*\* – внутренняя точка множества *X*, т.е. существует *ε*-окрестность точки *x*\*, все точки которой принадлежат *X*). Тогда для ∀*h*\*∈*Rn* имеем  при всех достаточно малых *α* > 0.Поскольку *f* – дважды дифференцируема в *x*\*, то можно записать:

⇒

⇒

.

Переходя к пределу при *α* → 0, имеем требуемое соотношение.

б) Пусть теперь *x*\*∈*X* – произвольная точка ⇒ существует последовательность точек , сходящаяся к *x*\*. По доказанному выше, для ∀*h*∈*Rn* имеем:

,

причем последовательность матриц  сходится к  в силу непрерывности  в *x*\* (*непрерывность всех вторых частных производных*) ⇒ имеем требуемое соотношение.

*Достаточность:* Пусть справедливо . Тогда, рассмотрим произвольные точки  и положим . Используя формулу Тейлора с остаточным членом в форме Лагранжа, имеем для некоторого *α*∈(0,1)

 (по предположению)

Итак, имеем:

 для .

Надо показать, что для  ,

т.е. тем самым показать, что функция f – выпуклая.

Для этого зафиксируем произвольные и рассмотрим точку *х*\*: (*в силу выпуклости X*).

Тогда:



и сложим их: ⇒

⇒,

т.е. *f* – выпуклая, *ч.т.д.*

**Пример.** Пусть  – квадратичная функция, *A* – симметричная матрица. Тогда *f* – выпуклая ⇔ *A* – неотрицательно определена.

Вообще, можно привести *критерии строгой и сильной выпуклости функций* аналогично тем, которые сейчас были доказаны для выпуклой функции (доказательство – аналогичное).

* Итак, для дифференцируемой функции *f*(*x*):
* *строгая выпуклость* эквивалентна неравенству

,

* *сильная выпуклость* эквивалентна неравенству

.

*Графически:*

*выпуклость* – возможно касание касательной плоскости;

*строгая выпуклость* – единственная точка касания с касательной плоскостью;

*сильная выпуклость* – график расположен внутри некоторого параболоида вращения

.

* Для дважды дифференцируемых функций *f*(*x*):
* достаточным условием *строгой выпуклости* *f*(*x*) является *положительная определенность* при ∀*x*∈*X* её матрицы Гессе *f*″(*x*);
* достаточным условием *сильной выпуклости* *f*(*x*) является *положительная определенность* матрицы , где *E* – единичная матрица, а *l* > 0.

Эти критерии в сочетании с критерием Сильвестра дают удобный аппарат для проверки выпуклости функций небольшого числа переменных.

**Функция Лагранжа**

Рассмотрим основную задачу выпуклого программирования:



*ϕ*(*x*), *fi*(*x*) – выпуклые функции.

В основной задаче выпуклого программирования имеем *m* условий, определяющих допустимое множество *X*.

Рассмотрим вектор .

**Определение.** Функция

,

называется *функцией Лагранжа* для основной задачи выпуклого программирования, где .

**Определение.** Пара (*x*\*, *λ*\*) называется *седловой точкой* функции Лагранжа на множестве , если ∀*x*∈*Rn* и ∀*λ*≥ 0

,

т.е.



*Наличие седловой точки означает, что операции минимизации и максимизации можно переставлять местами.*

В задачах *классического* анализа об условном экстремуме (задачи, в которых допустимое множество задается *системой уравнений*) важную роль играет метод множителей Лагранжа: решение исходной задачи ищется среди стационарных точек функции *L*(*x*, *λ*) – точек, удовлетворяющих системе уравнений

.

В задачах выпуклого (и, в частности, линейного) программирования функции Лагранжа также отводится важное место: при весьма общих предположениях задача выпуклого программирования сводится к отысканию седловых точек функции Лагранжа.

**Теорема** *о седловой точке функции Лагранжа* (достаточные условия оптимальности).

Если пара (*x*\*,*λ*\*) является седловой точкой функции Лагранжа  на множестве *x*∈*Rn*, *λ*≥ 0, то *x*\* – оптимальная точка основной ЗВП.

**Доказательство.** По определению седловой точки имеем:

 (\*)

а) Из левого неравенства (\*) убираем *φ*(*x*\*), и получаем:

, для ∀*i*, т.е. *x*\* – допустимая точка.

Действительно, если бы существовал индекс *i*: *fi*(*x*\*) > 0, то слева имели бы неограниченную сумму (т.к. ∀*λi*≥ 0), а справа имеем ограничение ⇒ для ∀*i fi*(*x*\*) ≤ 0.

б) В частности, левое неравенство (\*) верно и для *λ* = 0, тогда имеем, но

 (\*\*)

в) Подставим (\*\*) в правое неравенство (\*):



Поскольку, для ∀*x*∈*X* 

Итак, получили, что для ∀*x*∈*X*  , т.е. *x*\* – оптимальная точка, *ч.т.д.*

Отметим, что при доказательстве теоремы нигде не использовались ни свойства выпуклости функций *ϕ*(*x*), *fi*(*x*), ни свойства выпуклости множества *Rn*, ни какие-либо свойства гладкости.

Т.о., наличие седловой точки (*x*\*, *λ*\*) функции Лагранжа определяет оптимальность точки *x*\* для общей задачи математического программирования. Обратное утверждение, что из оптимальности точки *x*\* следует существование седловой точки (*x*\*, *λ*\*) функции Лагранжа, справедливо лишь для задачи выпуклого программирования при выполнении определенных ограничений относительно допустимого множества *X*.

Сформулируем эти ограничения и саму теорему, известную как *теорема Куна-Таккера*.

**Определение 1.** Рассмотрим допустимое множество

.

Если для всех *i*∈1,…,*m* существует такая точка *xi*∈*X*, что *fi*(*xi*) < 0, то говорят, что допустимое множество *X* удовлетворяет *условию регулярности*.

**Определение 2.** Пусть существует такая точка *x*∈*X*, что для всех *i*∈1,…,*m* выполняется *fi*(*x*) < 0. Тогда говорят, что допустимое множество удовлетворяет *условию регулярности Слейтера*.

Определения (1) и (2) – эквивалентны.

Действительно, из (2) ⇒ (1) – очевидно (*xi* ≡ *x*).

Пусть теперь выполнено (1).

Выберем ,

Неравенство Йенсена для выпуклых функций

тогда для ∀*i*∈1,…,*m* имеем: , *ч.т.д.*

Условие (2) означает, что существует точка внутри допустимого множества.

**Теорема** (Куна-Таккера(необходимые и достаточные условия оптимальности)). Пусть в основной задаче выпуклого программирования допустимое множество *X* обладает свойством регулярности. Тогда необходимым и достаточным условием оптимальности точки *x*\* является существование такого *λ*\*­ ≥ 0, чтобы пара (*x*\*, *λ*\*) была седловой точкой для функции Лагранжа на множестве *x*∈*Rn*, *λ*≥ 0.

**Доказательство.**

*Достаточность* доказана в теореме о седловой точке функции Лагранжа.

*Необходимость*.

Пусть *x*\* – оптимальная точка. Рассмотрим два множества в пространстве *Rm*+1:

– множество ;

– и множество , где ∀*x* .

*Множество P – выпукло.*

Действительно, пусть *z*′,*z*″∈*P* ⇒ рассмотрим *z* = *αz*′ + (1 – *α*)*z*″ ∀*α*∈[0,1] и покажем что z∈*P*.

Положим 



*Множество S – выпукло.*

Действительно, пусть ⇒

Рассмотрим

, ∀*α*∈[0,1]

По определению множества *S*:

⇒ рассмотрим *x*=*αx*′+(1–*α*)*x*″ и покажем, что .

Т.к. *ϕ* – выпуклая функция, то 

Т.к. *fi*(*x*) – выпуклая функция, то  ∀*i*

⇒ ⇒ *S* – выпукло.

Рассмотрим  – множество внутренних точек *P* и покажем, что пересечение *P*0 ∩ *S* = ∅.

* Для ∀*x*∈*X*  (оптимальность), но .
* Для ∀*x*∉*X* , но .

⇒ общих точек в множествах *P*0 и *S* – нет.

Применим к множествам *P* и *S* теорему о разделяющей гиперплоскости.

Существует разделяющая гиперплоскость, т.е. существует

 для  и .

При этом вектор , т.к. компоненты векторов из *P* неограниченны снизу.

Выберем  на границе множества  и .

Тогда получим:

 (\*)

Покажем что *u*0 ≠ 0 (тем самым покажем, что *u*0 > 0, т.к. по условию *u*0 ≥ 0).

Допустим, что *u*0 = 0, тогда (*u*,*f*(*x*)) ≥ 0 ∀*x*∈*Rn*.

При этом, поскольку , то существует индекс *i*: *ui*≠ 0, т.е. *ui*> 0.

С другой стороны, ∀*x*∈*X* *f*(*x*) ≤ 0 ⇒ для *ui*> 0 для ∀*x*∈*X* *fi*(*x*) = 0, что противоречит свойству регулярности.

Итак, *u*0 > 0, и определим .

Для этого вектора соотношение (\*) примет вид:

 (\*\*)

⇒ при *x*=*x*\* (*λ*\*, *f*(*x*\*)) ≥0.

Но т.к. *λ*\* ≥ 0, а .

Далее, для .

Собирая все вместе, получим:

,

или  ∀*λ*≥ 0, ∀*x*∈*Rn*,

т.е. (*x*\*, *λ*\*) – седловая точка функции Лагранжа, *ч.т.д.*

**Замечаниe.**

Теорема Куна-Таккера лежит в *основе теории двойственности* математического программирования. Она также находит *применение в численных методах* решения задач математического программирования. Она позволяет исходную задачу заменить задачей отыскания седловой точки функции Лагранжа, т.е. задачей вида:



"Простые" ограничения этой задачи позволяют применять для её решения методы, во многом схожие с численными методами безусловной оптимизации, достаточно хорошо изученные и апробированные.