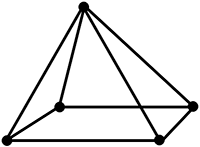
**Замечания.**

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.

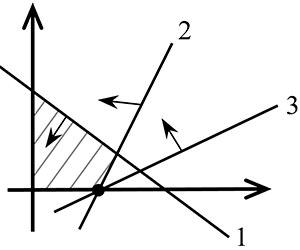
**Определение.**Крайняя точка называется *невырожденной*, если она лежит на пересечении ровно  гиперплоскостей. В противном случае она называется *вырожденной*. Задача линейного программирования, у которой существует вырожденная точка, также называется вырожденной.

*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.

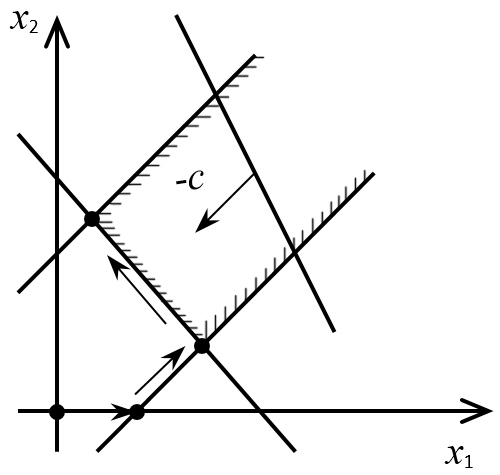
Следовательно, естественным образом подходим к основной идее *симплекс-метода* – полный перебор следует заменить упорядоченным, разумным.

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: .

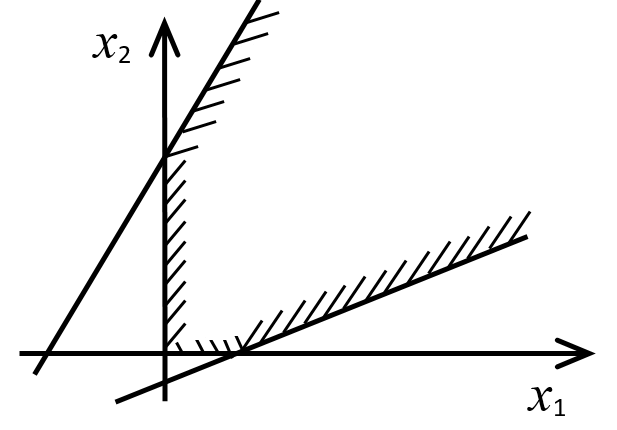
Итак, допустимое множество *X* ограниченно (*m* + *n*) – гиперплоскостями: .

1. **Алгоритм поиска крайней точки**

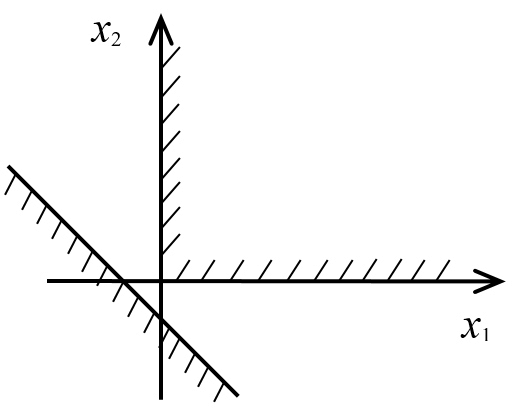
Имеем систему:



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 

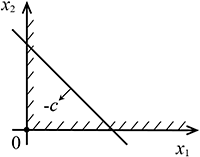
, *ч.т.д.*

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.

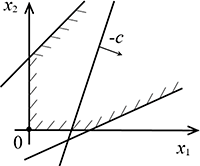
1. **Алгоритм поиска оптимальной точки**

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:



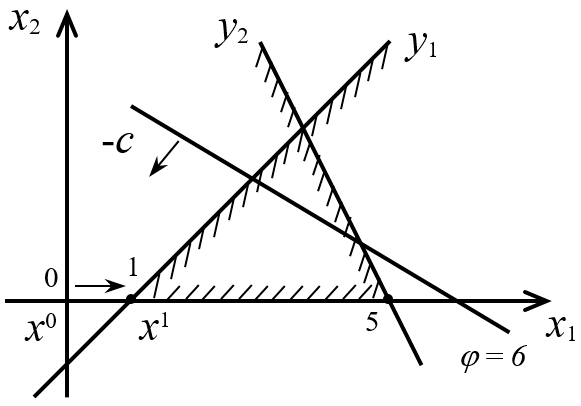
**Примеры.**

**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:



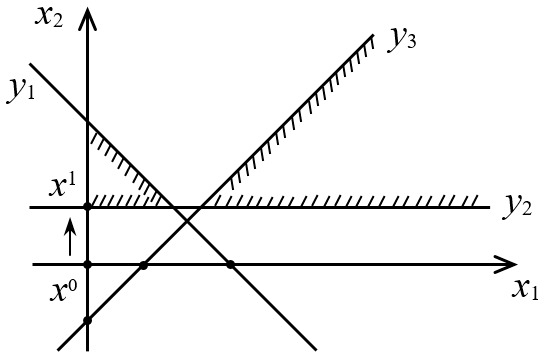


Рис.2

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:







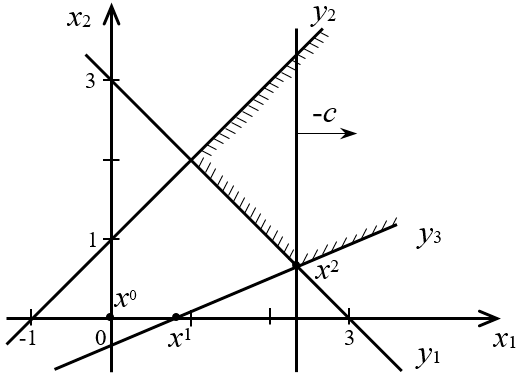
⇒ приходим к таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | *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* |  |  |  |