Нелинейное Программирование

- **Типовые примеры применения**
- **Графическое изображение**
- **Виды задач нелинейного программирования**
- **Безусловная оптимизация с одной переменной**
- **Безусловная оптимизация с несколькими переменными**
- **Условие Каруша-Куна-Таккера (ККТ) для оптимизации с ограничениями**
- **Квадратичное программирование**
- **Сепарабельное программирование**
- **Выпуклое программирование**
- **Невыпуклое программирование**

Задачи планирования характеризуются нелинейностью □
необходимо решать такие нелинейные задачи.

Задача нелинейного программирования:

найти

x = (x1, x2, . . . , xn) так, чтобы

Максимизировать ***f(x)***,

Согласно ***gi(x) ≤ bi*** : i = 1, 2,. . . , M,

и x ≥ 0,

f(x), gi(x), заданная функция с ***n*** переменными решения.

Не существует универсального алгоритма для решения каждой конкретной задачи этого формата. Особые случаи решаются путем различных предположений. Рассмотрим типовые приложения и основные идеи решения некоторых важных типов задач нелинейного программирования.

Типовые примеры

- Задача о комбинации продуктов при эластичности цен
- Транспортная задачи с оптовыми скидками на транспортные расходы
- Выбор портфеля активов с

Графическое Изображение

Видов задач нелинейного программирования

- Безусловная оптимизация
- Оптимизация с линейными ограничениями
- Выпуклое программирование
- Сепарабельное программирование
- Невыпуклое программирование
- Геометрическое программирование
- Дробное программированию
- Задача комплиментарности

- Один алгоритм не может решить все эти различные типы задач. Поэтому были разработаны алгоритмы для различных классов (определенных типов) нелинейного программирования.

Безусловная Оптимизация с одной переменной

- Одномерная процедура поиска

Безусловная оптимизация с несколькими переменными

- Процедура градиентного поиска

Условие Каруша-Куна-Таккера (ККТ) для оптимизации с ограничениями

- квадратичное программирование

Квадратичное программирование

- ККТ условия для квадратичного программирования
- Измененный симплекс-метод
- Правило ограниченного доступа

Сепарабельное программирование

- Переформулировка задачи как задача линейного программирования

Выпуклое программирование

- Алгоритм последовательной линейной аппроксимации (Франка-Вульфа)

Невыпуклое програмирование

Техника последовательной безусловной минимизации (SUMT)

Графическое Представление

с 1 или 2 переменными может быть представлено графически. (пример о стекольном заводе для линейного программирования).

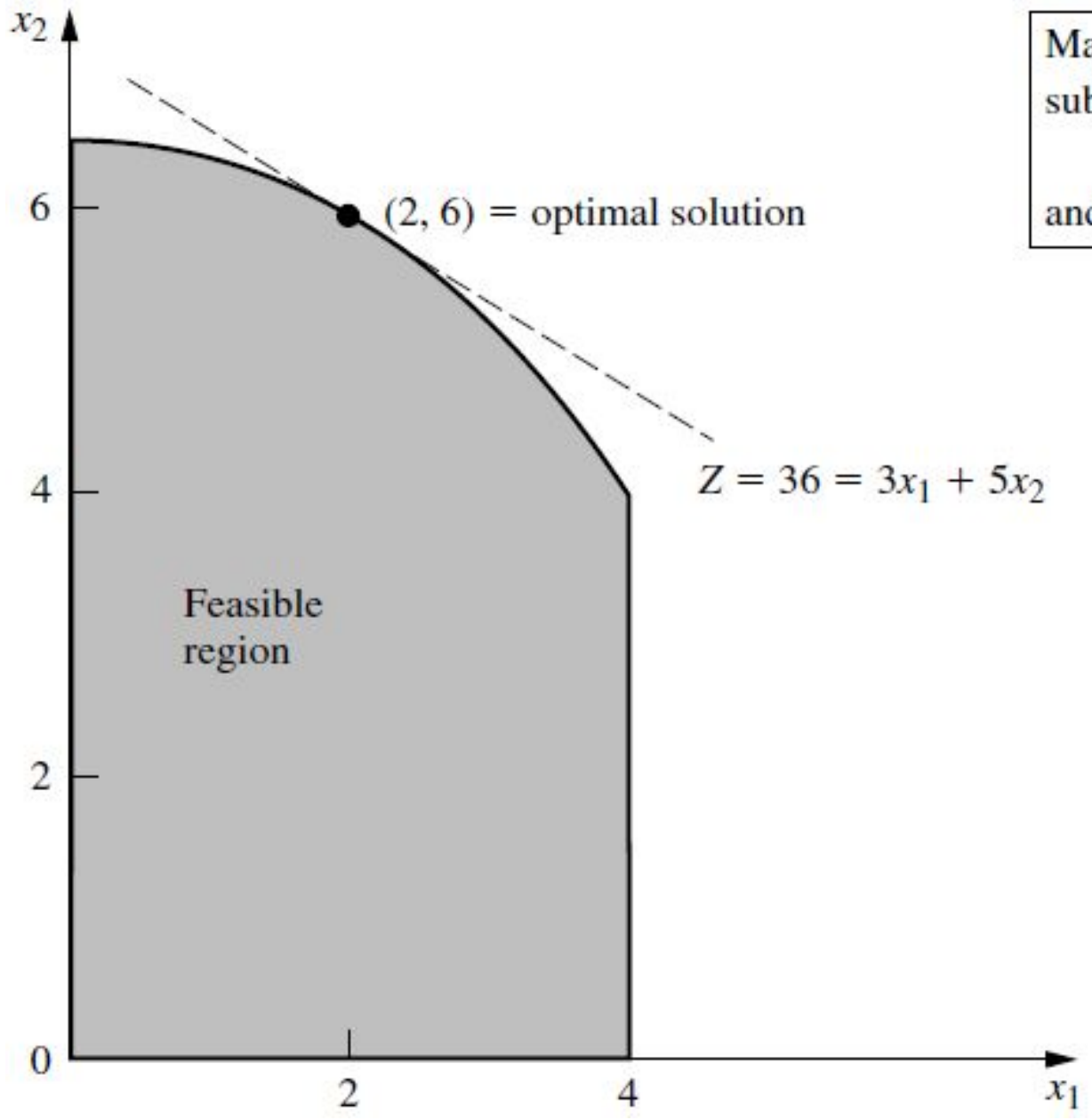
-Если 2-е и 3-е функциональные ограничения заменены одним нелинейным ограничением:

$$9x_1^2 + 5x_2^2 \leq 216$$

- Оптимальное решение все еще остается (x1, x2) = (2, 6) на границы области допустимых значений, но оно не допустимое решение угловой точки(УГД). УГД решение возможно, но с другой целевой функцией: (Z=3x1 + x2).

- Если целевая функция нелинейная:

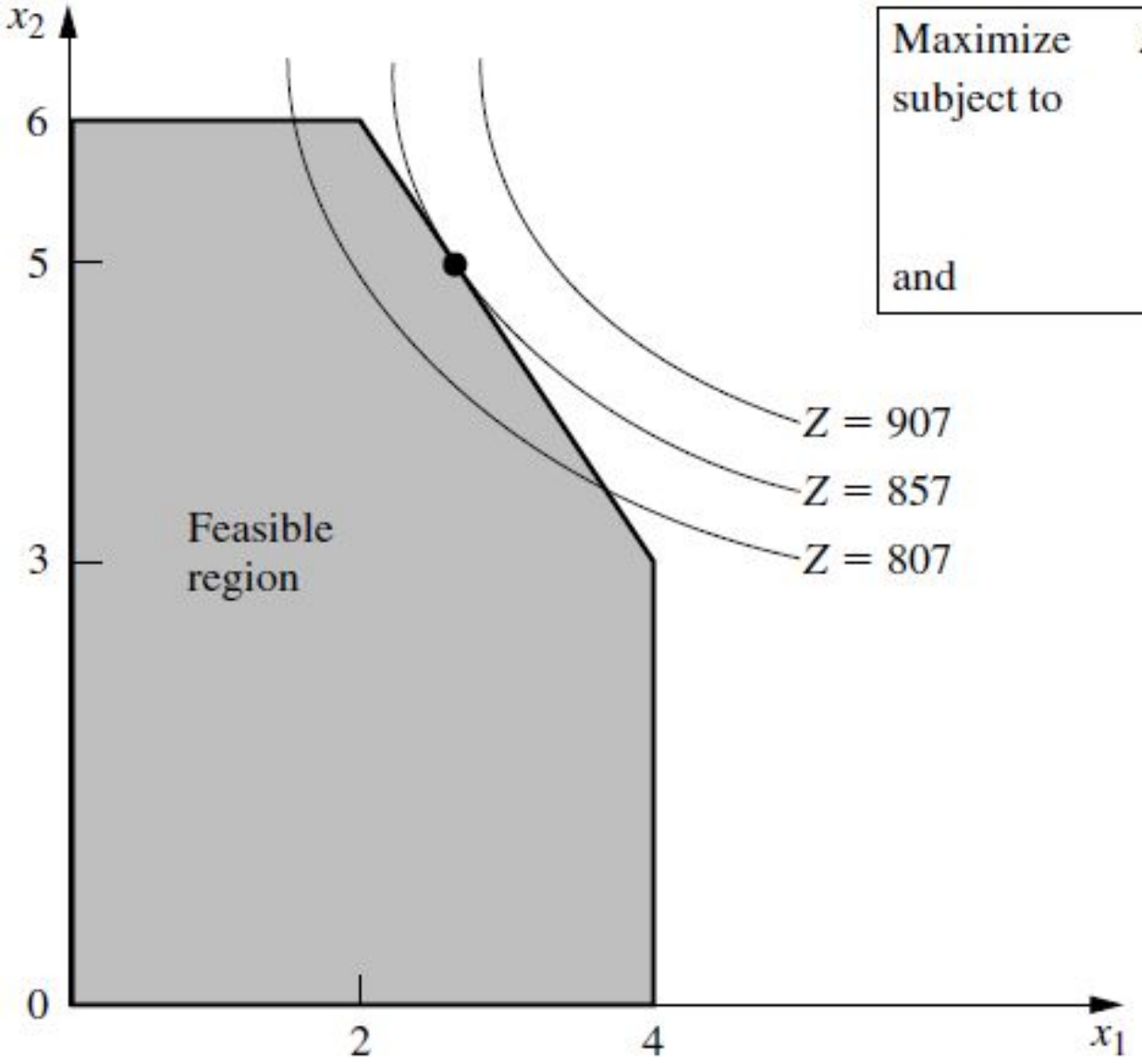
$$Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$$



Maximize	$Z = 3x_1 + 5x_2$,	
subject to	x_1	≤ 4
	$9x_1^2 + 5x_2^2$	≤ 216
and	$x_1 \geq 0$,	$x_2 \geq 0$

Пример стекольного завода
с нелинейным ограничением

$$9x_1^2 + 5x_2^2 \leq 216$$



Maximize	$Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2,$			
subject to	x_1	$\leq$	4	
	$2x_2$	$\leq$	12	
	$3x_1 + 2x_2$	$\leq$	18	
and	$x_1 \geq 0,$		$x_2 \geq 0$	

Пример о стекольном заводе с исходной областью допустимых значений с нелинейной целевой функцией: $Z = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$

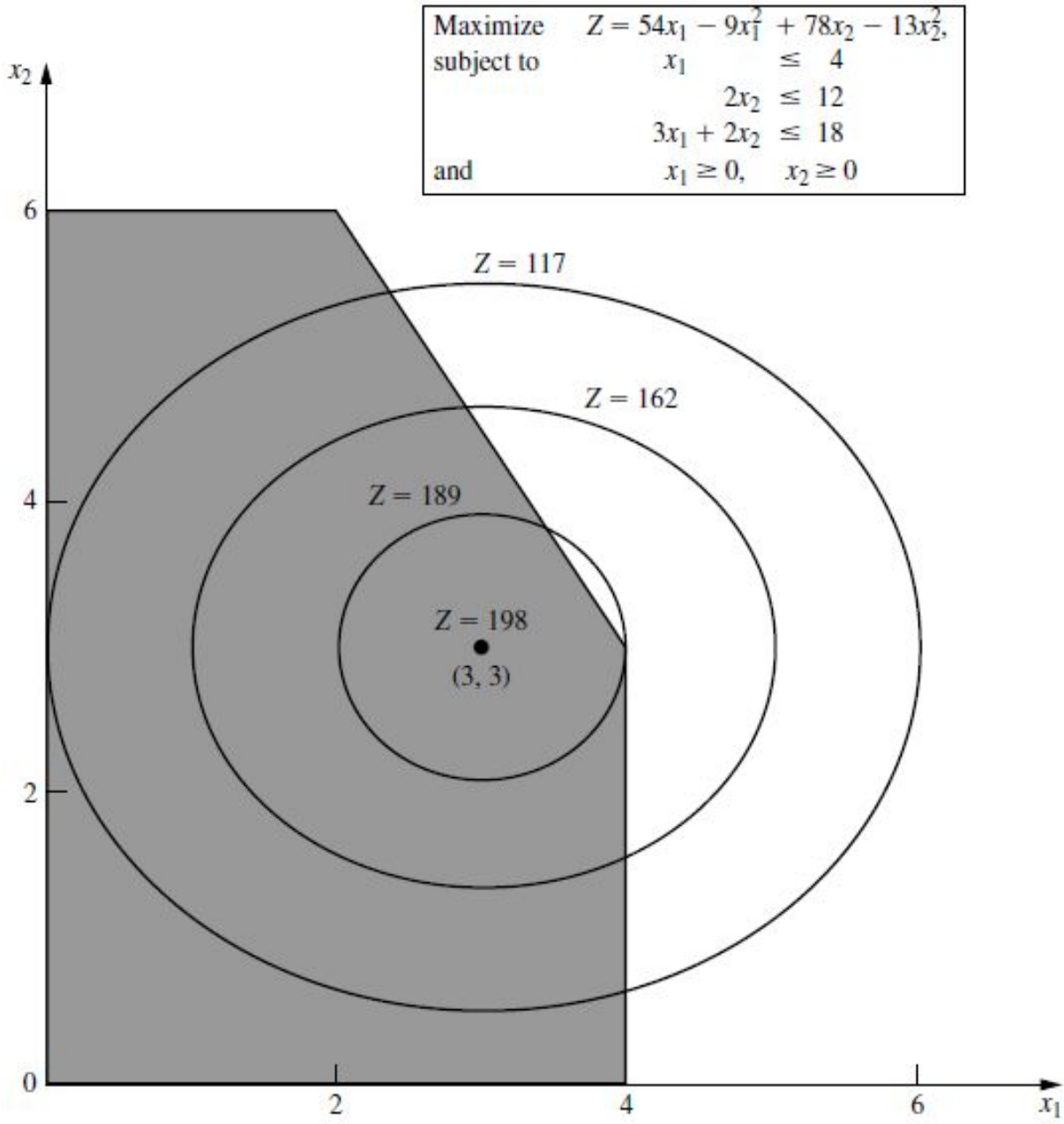
Оптимальным решением является (8/3, 5)оно лежит на границе области допустимых значений (Z = 857), геометрическое область точек с Z = 857 пересекает область допустимых значений именно в этой точке, в то время как геометрическая область точек с любым большим Z вообще не пересекает область допустимых значений).

если $Z = 54x_1 - 9x_1^2 + 78x_2 - 13x_2^2$

Оптимальное решением является (x1, x2) = (3, 3), оно лежит внутри границы области допустимых значений. (Решение остается оптимальным, так как оно является безусловным абсолютным максимумом, и если оно удовлетворяет ограничениям, то оно является оптимальным и для задачи с ограничениями).

Пример о стекольном заводе с исходной областью допустимых значений с другой нелинейной

$$Z = 54x_1 - 9x_1^2 + 78x_2 - 13x_2^2$$



Общий алгоритм должен рассмотреть все решения в области допустимых значений, а не только на ее границе. Локальный максимум не обязательно должен быть глобальным максимумом (общее оптимальное решение). Например, рассмотрим функцию с одной переменной построенную в пределах $0 \leq x \leq 5$ □ функция имеет 3 локальных максимума, *x = 0, 2, 4, но только*
X = 4 - глобальный максимум. (*локальные минимумы в x = 1, 3, 5, но только x = 5 является глобальным минимумом*). Нелинейные алгоритмы программирования не в состоянии различать локальный максимум и глобальный максимум (за исключением нахождения другого *лучшего локального максимума*). Важно знать условия, при которых любой локальный максимум гарантированно будет

Максимизация обычной (дважды дифференцируемой) функции с одной переменной *f(x)*) без каких-либо с$\frac{d^2f}{dx^2} \leq 0$ичений, когда

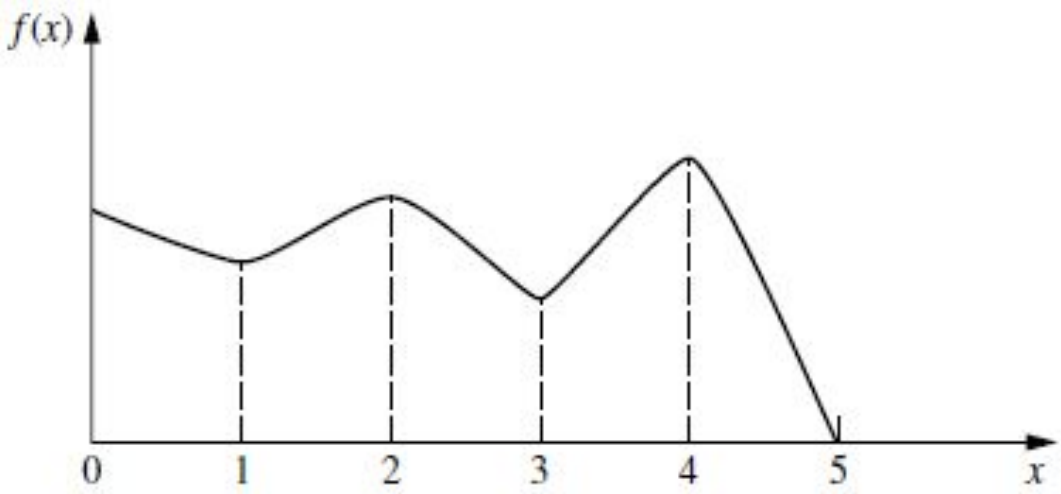
для всех *x* □ функция всегда изогнута вниз: вогнутая функция.

Если ≤ заменены ≥ □ функция всегда изогнута вверх: выпуклая функция.

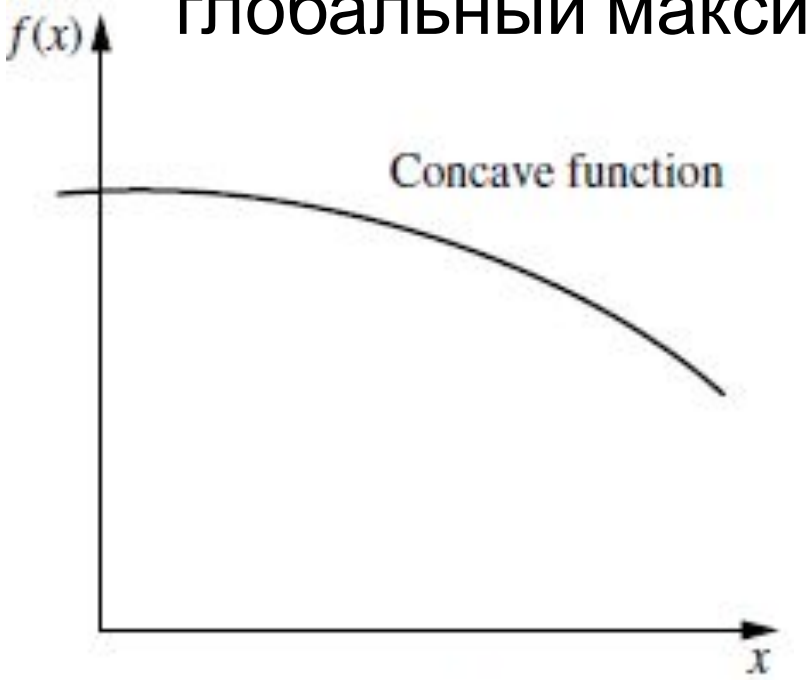
(Линейная *функция* как вогнутые, так и выпуклые).

Функция не является ни вогнутой, ни выпуклой, если изгибы вверх и вниз чередуется между собой.

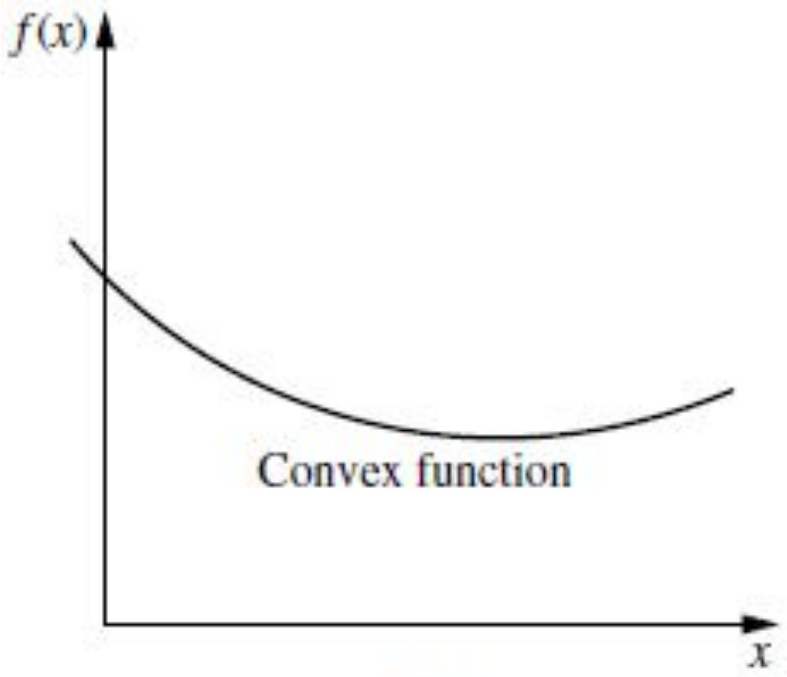
Функции со множеством переменных характеризуются как вогнутые или выпуклые, если всегда изогнуты вниз или вверх. Проверка: для функции с более двух переменных (функция состоит из *суммы более мелких функций с одной или двумя переменными каждая*).



Функция с несколькими локальными максимумами (x = 0, 2, 4), только x = 4 глобальный максимум



(a)



(b)

Если каждая меньшая функция = вогнутая

= общая функция вогнута.

Если каждая меньшая функция = выпуклая

= общая функция выпуклая.

$$f(x_1, x_2, x_3) = 4x_1 - x_1^2 - (x_2 - x_3)^2$$
$$= [4x_1 - x_1^2] + [-(x_2 - x_3)^2]$$

Сумма 2 небольших функций (в квадратных скобках).

$4x_1 - x_1^2$ функция x1 вогнутая, потому что ее вторая производная отрицательна.

$-(x_2 - x_3)^2$ функции *x2 и x3* вогнутые, потому что обе меньших функции вогнуты.

Задача нелинейного программирования не имеет ограничений: Целевая функция = *вогнутая гарантирует, что локальный максимум = глобальный максимум.*

Целевая функция = выпуклая гарантирует, что локальный минимум = глобальный минимум.

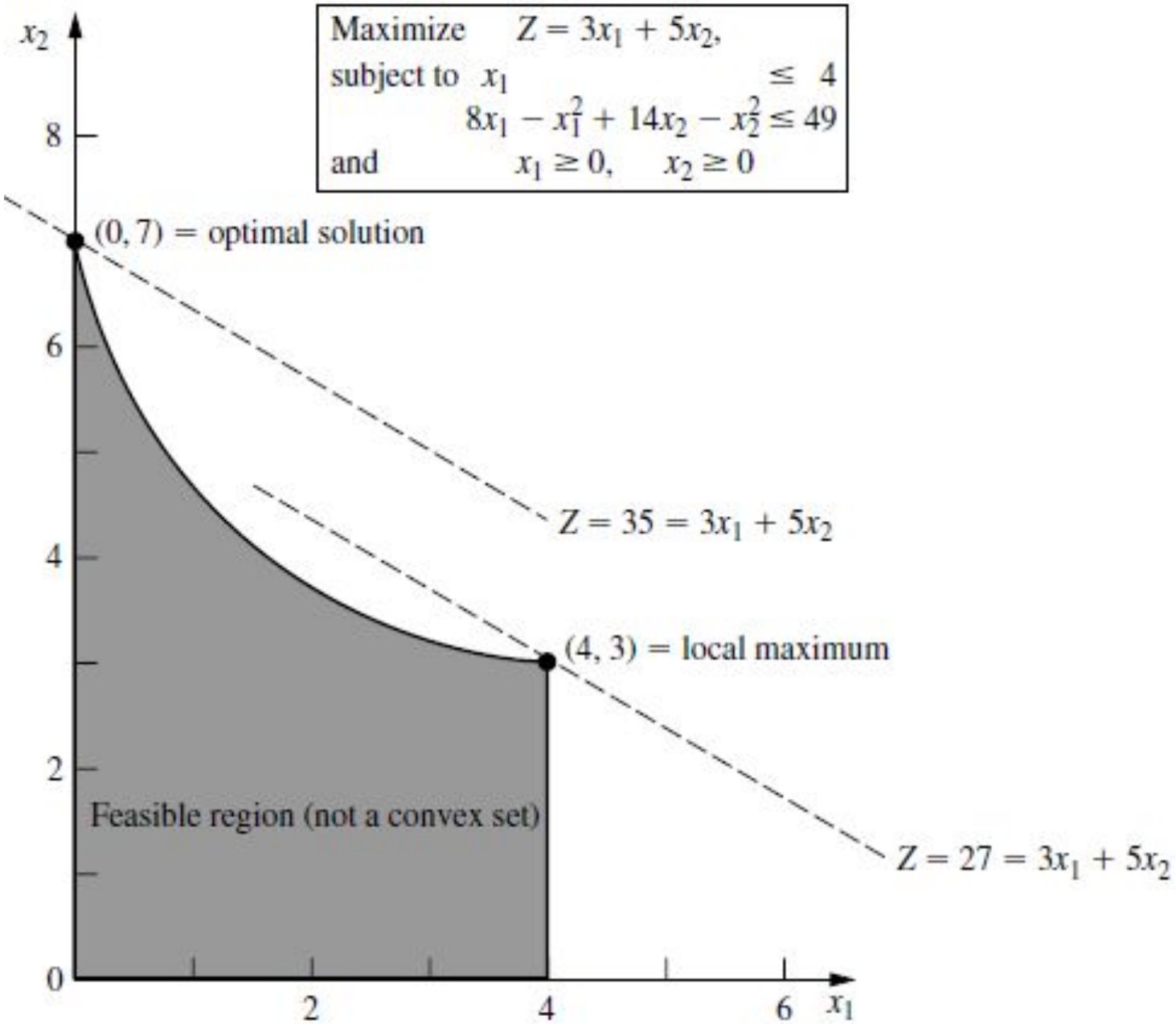
Если есть ограничения еще одно условие обеспечивает эту гарантию.

Область допустимых значений = выпуклое множество (множество точек, где для каждой пары точек в области, весь отрезок линии, соединяющей их, также лежит в области) область допустимых значений для задачи о стекольном заводе = выпуклое множество.

Область допустимых значений для любой задачи *линейного программирования* = выпуклое множество. Область допустимых значений на рис. является выпуклым множеством.

Область допустимых значений для задач нелинейного программирования = выпуклое множество, где все *gi*(*x*) = выпуклые функции [для ограничения *gi*(*x*) ≤ bi]. *g1*(*x*) = *x1* (линейная функция автоматически и вогнутая и выпуклая) и $g_2(\mathbf{x}) = 9x_1^2 + 5x_2^2$

($9x_1^2$ и $5x_2^2$ = выпуклые функции □ их сумма = выпуклая функция). Эти две выпуклые *gi*(*x*) приводят к области допустимых значений, которая явлвется выпуклым множеством. Когда только одна из этих *gi*(*x*) = вогнутая функция. Нел$8x_1 - x_1^2 + 14x_2 - x_2^2 \leq 49$я заменяются на



Пример о стекольном заводе с другими, нелинейными огр$8x_1 - x_1^2 + 14x_2 - x_2^2 \le 49$

$g_2(\mathbf{x}) = 8x_1 - x_1^2 + 14x_2 - x_2^2$ вогнутая функция, так как $8x_1 - x_1^2$ $14x_2 - x_2^2$ и = вогнутые функции. Новая область допустимых значений ≠ выпуклое множество (оно содержит пары точек (0, 7) и (4, 3), и часть отрезка, соединяющего эти две точки не находится в области допустимых значений). Нет гарантии, что локальный максимум = глобальному максимуму. Два локальных максимума, (0, 7) и (4, 3), но только (0, 7) = глобальный максимум □ чтобы гарантировать, что локальный максимум = глобальному максимуму для задачи нелинейного программирования с ограничениями $g_i(x) \le b_i$ (i = 1, 2, ..., т) и $x \ge 0$, целевая функция $f(x)$ должна быть вогнутой и каждая $g_i(x)$ должна быть выпуклой функцией (так называемое выпуклое программирование).

Виды задач нелинейного программирования

- Один алгоритм не может решить все эти различные типы задач. Поэтому были разработаны алгоритмы для отдельных классов (определенных типов) нелинейного программирования.

Безусловная оптимизация:

Нет ограничений

Цель: Максимизировать *F(x)*: *x* = (*x1*, *x2*, ..., *xn*).

Решение: *x* = *x** является оптимальным, когда *F(x)* дифференцируема функци$\frac{\partial f}{\partial x_j} = 0$:*x* = *x**, *j* = *1*, *2*,…, *n*.

Когда *f(x)* вогнутая функция (достаточное условие).

Решение для x* сводится к решению системы *n* уравнений полученных заданием *n* частных производных = *0*.

Для нелинейной функции *f(x)* эти уравнения также нелинейны трудно решить аналитически.

Процедуры поиска значения *x,* описанные для *n =1* и *n > 1*, играют важную роль в решении многих типов

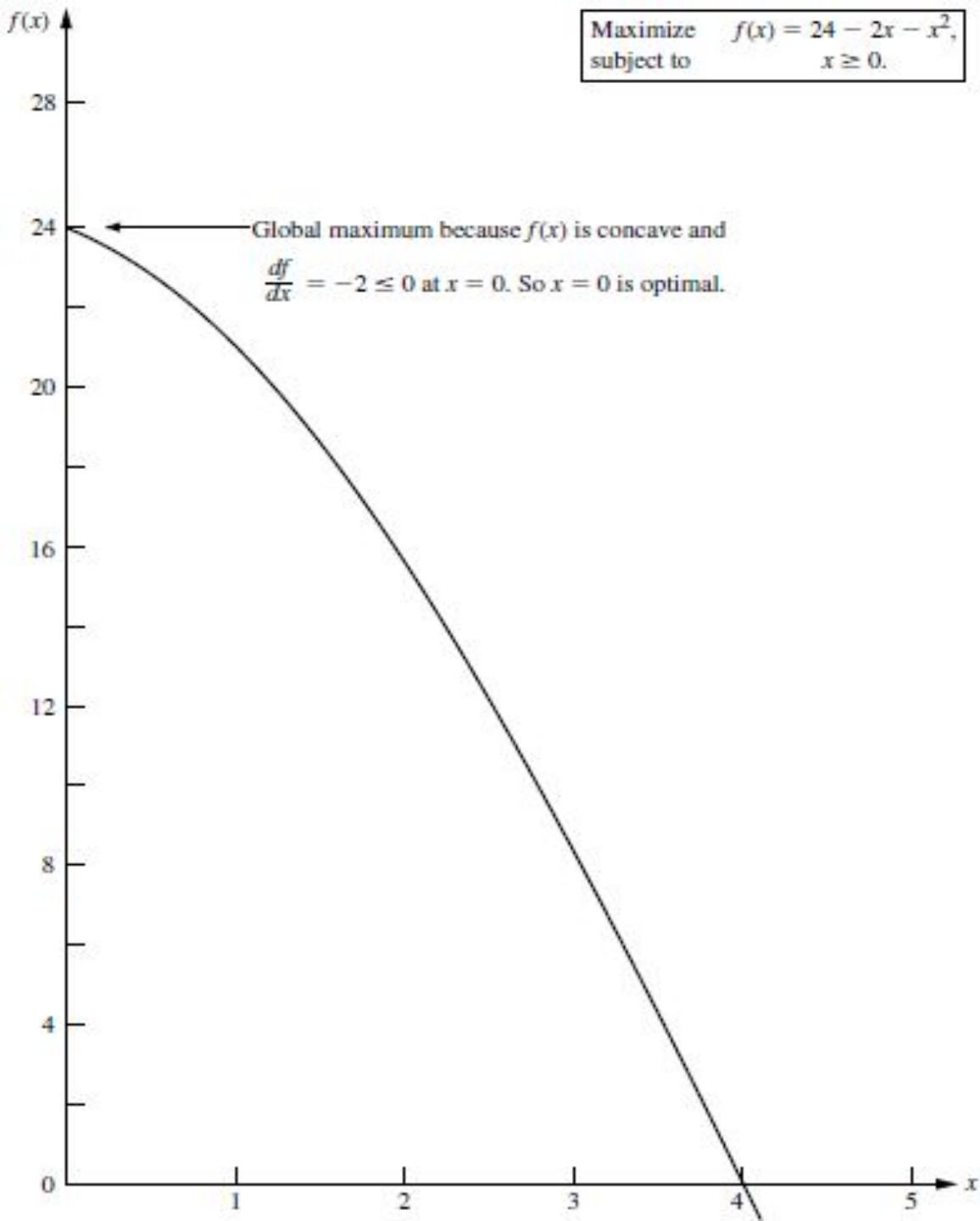
Алгоритмы задач с ограничениями сосредоточены на варианте неограниченный задачи на каждой итерации. Когда *xj* имеет ограничение неотрицательности
xj ≥ *0*, необходим $\frac{\partial f}{\partial x_j} \begin{cases} \leq 0 & \text{at } \mathbf{x} = \mathbf{x}^*, \quad \text{if } x_j^* = 0 \\ = 0 & \text{at } \mathbf{x} = \mathbf{x}^*, \quad \text{if } x_j^* > 0 \end{cases}$ а:

для каждого такого *j*, показанного на рис., где оптимальное решение задачи с одной переменной при *x* = *0*, и даже производной отрицательно, а не равно *0*. Вогнутая функция, которая следует максимизировать согласно условию неотрицательности, если имеет производные ≤ *0* при *x* = *0* это является необходимым и достаточным условием для
x = *0* оптимальным. Задача с ограничениями

Линейно ограниченная оптимизация:
gi(x) линейная функции с ограничениями, *f(x)* целевая функция нелинейна. Задача упрощается если имеем одну нелинейную функцию, с областью допустимых значений линейного программирования. Разработаны специальные алгоритмы, основанные на расширенном симплекс-методе для учета нелинейной целевой функции (особый случай: квадратичное программирование).

Квадратичное программирование:
линейные ограничения, *f(x)* квадратичная целевая функция. Отличие от линейного программирования: некоторые слагаемые в целевой функции включают квадрат переменной или произведение двух переменных.

Оптимальное решение в точке, где производная отрицательна, а не равна 0, по границе ограничений неотрицательности.



Расширенный симплекс-метод используется там, где *F(x)* вогнутая функция.

Квадратичное программирование очень важно: его формулировка естественно возникает во многих приложениях (Выбор портфеля активов с рисковыми ценными бумагами).

Общим подходом к решению общих линейно ограниченных задач оптимизации является решение последовательности аппроксимаций

Выпуклое программирование:
Широкий класс задач, который включает все предыдущие типы (как частный случай), где *f(x)* вогнутая функция. Предположения:
1. *f(x)* является вогнутой функцией.
2. Каждый *gi(x)* является выпуклой функцией.
Этого достаточно, чтобы обеспечить то, что локальный максимум = глобальному максимуму. Необходимыми и достаточными условиями для такого оптимального решения являются естественное обобщение условий для безусловной оптимизации и ее расширения, чтобы включить ограничения

Сепарабельное программирование:
Особый случай выпуклого программирования с дополнительным предположением:
3. Все *f(x)* и *gi(x)* – сепарабельные функции.
Сепарабельные функции = каждое слагаемое включает только одну переменную, поэтому функцию можно разделить на сумму функций с отдельными переменными: $f(\mathbf{x}) = \sum_{j=1}^{n} f_j(x_j)$

каждая *fj(xj)* функция включает только слагаемые, содержащие только *xj*. В линейном программировании сепарабельное программирование удовлетворяет предположению аддитивности, но не предположению пропорциональности (для нелинейных функций).

Целевая функция:

$$f(x_1, x_2) = 126x_1 - 9x_1^2 + 182x_2 - 13x_2^2$$

- сепарабельная функция, поскольку она может быть выражена как:

$$f(x_1, x_2) = f_1(x_1) + f_2(x_2)$$

где $f_1(x_1) = 126x_1 - 9x_1^2$ and $f_2(x_2) = 182x_2 - 13x_2^2$

каждая функция одной переменной, *x1* и *x2*.

Целевая функция: тоже сепарабельна. Задачи сепарабельного программирования отличаются от других задач выпуклого программирования тем, что они приводятся к задачам линейного программирования, так что может быть использован симплекс-метод.

Невыпуклое программирование :
Охватывает все задачи нелинейного программирования, которые не удовлетворяют предположениям выпуклого программирования. Даже если успешно найден локальный максимум, нет никаких гарантий, что это будет глобальный максимум никакой алгоритм не гарантирует нахождение оптимального решения для всех таких задач. Некоторые алгоритмы относительно хорошо подходят для поиска локальных максимумов, особенно когда формы нелинейных функций не слишком сильно отличаются от предложенных для выпуклого программирования. Тем не менее, некоторые специфические виды задач невыпуклого программирования можно решить без особого труда с помощью специальных методов.

Геометрическое программирование:
В применении нелинейного программирования для инженерных задач проектирования, целевая функция и функции ограничений примут вид:

$$g(\mathbf{x}) = \sum_{i=1}^{N} c_i P_i(\mathbf{x})$$

Где $P_i(\mathbf{x}) = x_1^{a_{i1}} x_2^{a_{i2}} \cdots x_n^{a_{in}}$ для *i = 1, 2,. . . , n*.
ci и *aij* представляют физические константы, *xj* переменные проектирования. Функции не является ни <u>выпуклой, ни вогнутой</u> □ методы выпуклого программирования не могут быть применены к этим задачам геометрического программирования.

Важный случай может быть преобразован в эквивалентную задачу выпуклого программирования (где коэффициенты *ci* в каждой функции строго положительны) функции здесь - обобщенные положительные многочлены (posynomials), целевая функция будет минимизироваться. Эквивалентная задача выпуклого программирования с переменными решения *y1*,*y2*,... ,*yn*, полученная задание $x_j = e^{y_j}$ для j = 1, 2, ... , n. по всей исходной модели, так что могут быть применены алгоритмы выпуклого программирования. Разработаны альтернативные процедуры решения для решения этих полиноминальных проблем программирования, а также для других типов задач геометрического программирования.

Дробное программирования:

Целевая функция в форме дроби максимизировать

$$f(\mathbf{x}) = \frac{f_1(\mathbf{x})}{f_2(\mathbf{x})}$$

Такие задачи возникают при максимизации отношения произведенного товара на количество затраченных человеко-часов (производительность) или прибыли на израсходованный капитал (доходность), или ожидаемого значения к стандартному отклонению меры производительности для инвестиционного портфеля (доходность/риск). Разработаны некоторые специальные процедуры решения для определенных форм *f1(x)* и *f2(x)*.

Решение проблемы дробного программирования – преобразовать ее в стандартный тип аналогичной задачи, для которых уже существуют эффективные процедуры решения. *f(x)* дробно-линейных форм программирования

$$f(\mathbf{x}) = \frac{\mathbf{cx} + c_0}{\mathbf{dx} + d_0}$$

где c, d - векторы строки, *x* - вектор-столбец,

c0, *d0* скаляры. Функции ограничений *gi(x)* линейные, поэтому ограничения в матричной форме: $Ax \leq B$ и $x \geq 0$.

При дополнительных предположениях, трансформируем к эквивалентной задаче линейного программирован ия и

$$y = \frac{x}{dx + d_0} \qquad t = \frac{1}{dx + d_0}$$

таким образом, что x = y/т □

Максимизировать $Z = cy + c0t$,
Согласно $Ay - bt \leq 0$,
$dy + d0t = 1$,
и $y \geq 0, T \geq 0$,
Что может быть решено симплекс-методом. Такие же преобразования используются для преобразования задач дробного программирования с вогнутыми $f1(x), f2(x), gi(x)$ в эквивалентные задачи выпуклого программирования.

Задача дополнительности

Решение некоторых задач нелинейного программирования может быть сведено к решению задач дополнительности. С учетом переменных *w1*, *w2*,. . . , *wp* и *z1*, *z2*,…, *zp*, задача дополнительности в том, чтобы найти подходящее решение для набора ограничений $w = F(z), w \geq 0, z \geq 0$ Которое удовлетворяет ограничен$\mathbf{w}^T\mathbf{z} = 0$

w и *z* = векторы-столбцы, *F* = заданная вектор-функция, *T* обозначает транспонирование. Задача не имеет целевую функцию, так что это не полноценная задача нелинейного программирования (задача дополнительности) из-за дополнительных отношений, которые либо

wi = 0 или *zi* = 0 (или оба) для каждого *i = 1, 2, ... , p*.

Важным частным случаем является линейная задача дополнительности:

$$F(z) = q + Mz,$$

где *q* = заданный вектор столбец и *M* = *P* x *P* заданная матрица. Эффективные алгоритмы, разработанные для решения этой проблемы при соответствующих предположениях о свойствах матрицы *M*. Один тип включает в себя поворот из одного базисного допустимого решения (BF) к другому, как симплекс-метод для линейного программирования. Задачи взаимодополняемости находят применение в нелинейном программировании, теории игр, экономических задачах равновесия и инженерных

Безусловной Оптимизации

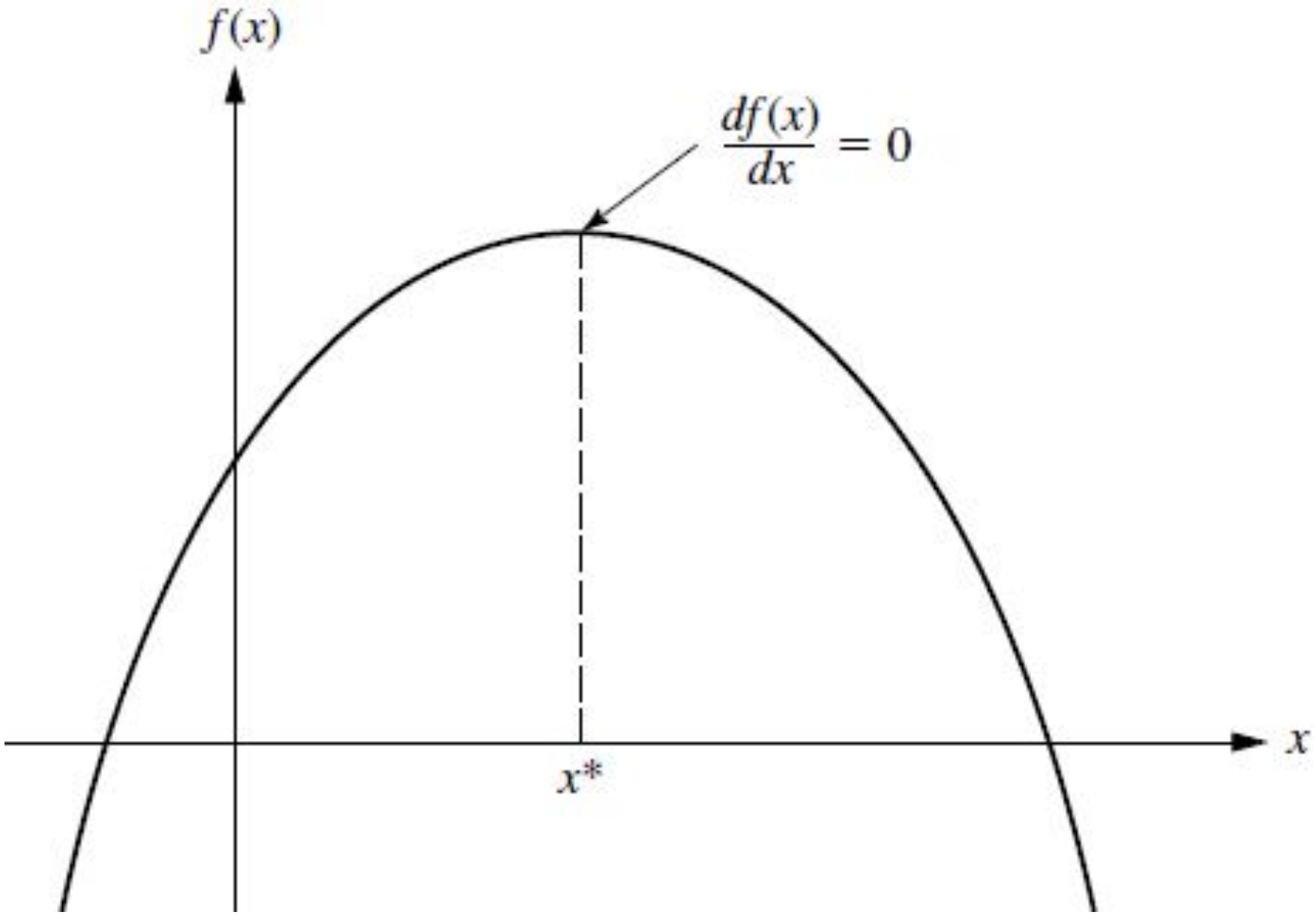
Безусловной оптимизации с одной переменной x (N = 1), где дифференцируемой функции (x) до максимума вогнутой необходимым и достаточным условием для конкретного решения x = x*, чтобы быть оптимальным (глобального максимума) находится на x = x*

$$\frac{df}{dx} = 0$$

Как на рис. Это уравнение может быть решена явно в X *, или, если F(x) не простая функция, поэтому производная не только линейной или квадратичной функцией, уравнение не может быть решена аналитически.Одномерного поиска процедура обеспечивает простой способ решения проблемы численно.

Одномерные Процедура поиска

Одномерного поиска процедура находит последовательность решения суда, который приводит к оптимальному решению. На каждой итерации, вы начинаете с текущим решением суда проводить систематический поиск, который завершается путем



1-переменной задачи безусловной программирования, когда функция вогнута.

Если наклон (производная) положительное или отрицательное решение в суде указывает, является ли улучшение лежит к правой или левой если производная в конкретное значение x положительно x^* должна быть > x (см. рис), так что эта x становится нижней границей решения суда должны быть рассмотрены позже. Если производная отрицательна x^*, должны быть <x, поэтому x станет верхняя граница. Определение обе границы, каждое новое решение суда выбран границы между текущим предусматривает новые жесткие связанный одного типа, сужая тем самым область поиска. Как правило выбора каждого пробного раствора Таким образом, результирующая последовательность проб решения должны сходиться к x^*. На практике, то есть с продолжающимися последовательности, пока расстояние между границ не достаточно мал, что следующее решение суда должно быть в пределах заранее оговоренного ошибке допуска

Для выбора каждого нового решения суда, который используется процедурой является серединой правило (так называемый план поиска Больцано), который выбирает середину между двумя текущие границы.

Резюме: **Одномерные процедуру поиска**. Инициализации: Выбрать є. Найти начальные $\underline{x}$ и $\bar{x}$ и осмотр (или найти любое значение *x*, при котором производная положительна, а затем отрицательный). Выберите исходный раствор суда

$$x' = \frac{\underline{x} + \bar{x}}{2}$$

Итерация:

1. Оценка $\frac{df(x)}{dx}$ при $x = x'$.

2. Если $\frac{df(x)}{dx} \geq 0$, сброс $\underline{x} = x'$.

3. Если $\frac{df(x)}{dx} \leq 0$, сброс $\overline{x} = x'$.

4. Выберите новый.

Правила остановки:

Если $\overline{x} - \underline{x} \leq 2\epsilon$, таким образом, что новый x’ должен быть в є из x*, остановиться.

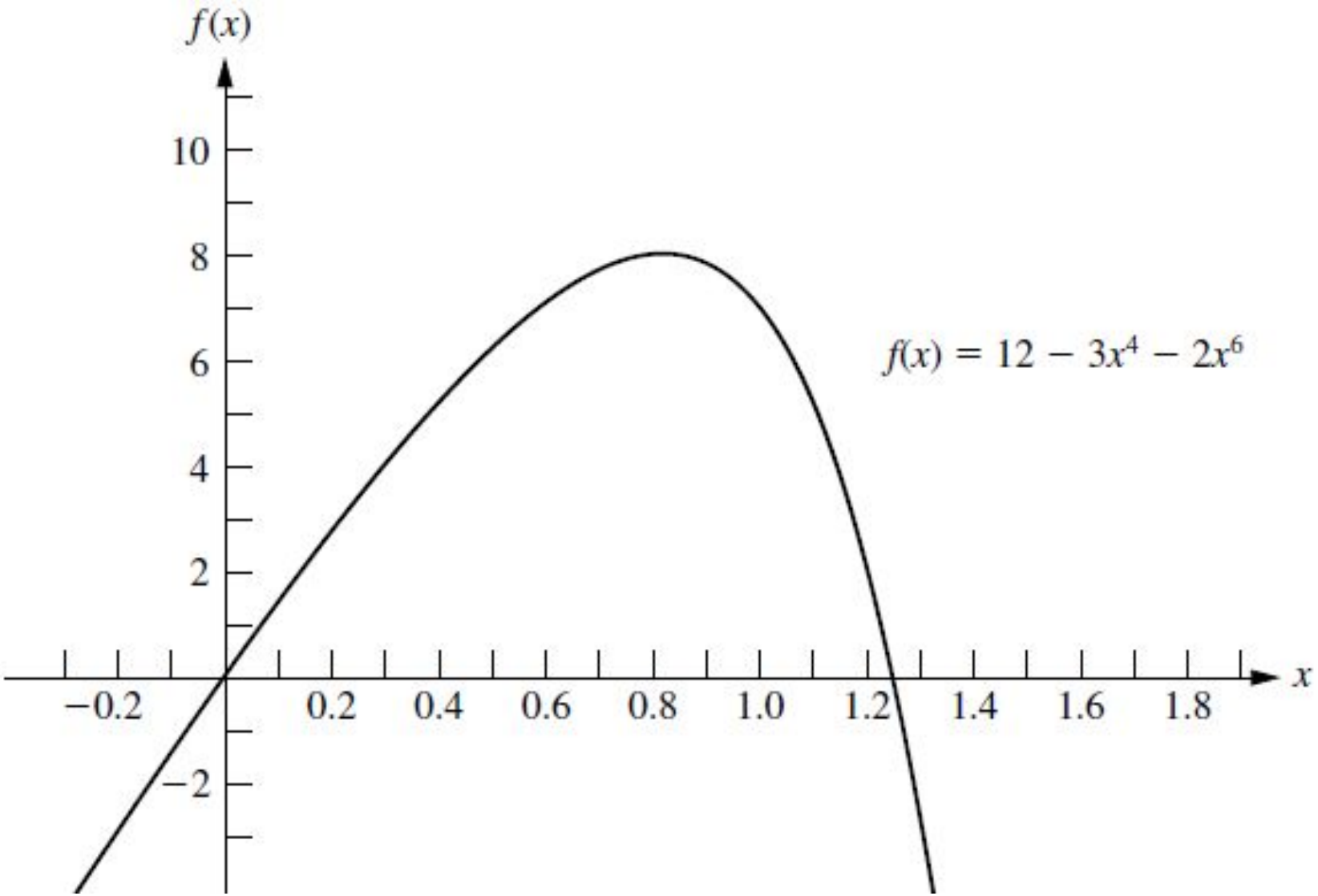
В противном случае выполните другой итерации.

Пример: Предположим, что функция, которая будет максимальным: $f(x) = 12x - 3x^4 - 2x^6$, А на рис.

Первых двух производных: $\frac{df(x)}{dx} = 12(1 - x^3 - x^5),$

$$\frac{d^2f(x)}{dx^2} = -12(3x^2 + 5x^4).$$

Вторая производная отлична от положительной везде, *f(x)* является вогнутой функцией, поэтому одномерного поиска процедура может быть применена безопасно найти свой глобальный максимум (при условии, глобальный максимум существует). Быстрый осмотр этой функции (даже без построения ее графика, как показано на рис.) Показывает, что *f(x)* является положительным для малых положительных значений *x*, но это негативно для



Одномерные Пример процедуры поиска.

Iteration	$\frac{df(x)}{dx}$	$\underline{x}$	$\overline{x}$	New x'	$f(x')$
0		0	2	1	7.0000
1	−12	0	1	0.5	5.7812
2	+10.12	0.5	1	0.75	7.6948
3	+4.09	0.75	1	0.875	7.8439
4	−2.19	0.75	0.875	0.8125	7.8672
5	+1.31	0.8125	0.875	0.84375	7.8829
6	−0.34	0.8125	0.84375	0.828125	7.8815
7	+0.51	0.828125	0.84375	0.8359375	7.8839
Stop					

Одномерного поиска процедура подачи заявок

$\underline{x}$ = 0, $\bar{x}$ = 2, используются в качестве исходных границ, с их средней точке *x = 1*, а начальное решение суда. Пусть *ϵ = 0,01* быть ошибки терпимости x* в правила остановки, так что с$(\bar{x} - \underline{x})$ательное

≤ 0,02 с окончательным x’ в середине. 1D-поисковый процедура дает последовательность результаты представлены в табл. [Таблица включает как функция и производные величины, где производное оценивали при испытании решение сгенерировано предшествующих итерации алгоритма не нужно вычислить *f(x)* вообще и что она нужна только для расчета производной достаточно далеко, чтобы определить его знак] , сделать вывод, что

$$x^* \approx 0{,}836,$$

$0.828125 < x^* < 0.84375$.

Многомерных Безусловной Оптимизации:
Максимизация вогнутой функции *f(x)* несколько переменных x (x1, x2, ..., xn), когда нет никаких ограничений на возможные значения.
Необходимые и достаточные условия оптимальности, дается система уравнений, установив соответствующие частные производные = 0, не может быть решена аналитически, поэтому численная процедура поиска должна быть использована. Как предыдущую процедуру поиска 1D быть продлен до этой многогранной проблемы?
Стоимость обыкновенных производных используется для выбора одного из всего двух возможных направлений (увеличение или уменьшение *x*), в котором, чтобы перейти от текущего решения суда к следующему.

Цель: достичь точки, где эта производная = 0. Есть бесчисленное множество возможных направлений, в которых для перемещения, соответствуют возможно пропорциональное соотношение, при котором соответствующие переменные могут быть изменены. Достигнув точки, где все частные производные = 0 □ расширения 1D процедуру поиска требует использования значений частных производных, чтобы выбрать определенное направление, в котором двигаться (включает в себя использование градиента целевой функции). Потому что цельовой функции *f(x)* предполагается дифференцируемой, он обладает градиент $\nabla f(x)$, в каждой точке x.

Градиент в конкретной точке $x = x'$ является вектором, элементами которого являются частные производные от $\nabla f(\mathbf{x}') = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_n}\right)$ при $x = x'$. Значение градиента в том, что (бесконечно малых) изменение x, который максимизирует скорость, с которой $f(x)$ увеличивает это изменение, которое пропорционально $\nabla f(\mathbf{x})$. Геометрически направлен$\nabla f(\mathbf{x}')$ градиента имеет направленный отрезок (стрелка) от координат ($\partial f/\partial x_1, \partial f/\partial x_2$, до т$\partial f/\partial x_n$и (, ..., $\partial f/\partial x_j$), где оценивается в $xj = xj'$ □ скорость, с которой $f(x)$ возрастает, если максимальна (бесконечно малых) изменений x в нап$\nabla f(\mathbf{x})$влении градиента .

Цель: найти максимально допустимое решение $f(x)$, казалось бы целесообразно пытаться двигаться в направлении градиента как можно больше.

Процедура Градиент Поиск:
Текущая проблема не имеет ограничений, градиент предполагает, что эффективная процедура поиска должна продолжать двигаться в направлении градиента, пока не достигнет оптимального решения x^*, где $f(x^*) = 0$. Это не практично изменить x непрерывно в направлении, потому что это ряд изменений потребует непрерывной переоценке и изменению пути направления □ лучше было бы продолжать двигаться в фиксированном направлении от текущего решения суда, не останавливаясь, пока $f(x)$ не перестает расти.

Остановка точка будет следующее решение суда, поэтому градиент затем будут пересчитаны, чтобы определить новое направление. При таком подходе каждая итерация включает в себя изменение текущего х решение суда $\mathbf{x}' = \mathbf{x}' + t^* \nabla f(\mathbf{x}')$ образом: сброс , где t* является положительным значением т, которая максим$f(\mathbf{x}' + t \nabla f(\mathbf{x}'))$, то есть

$$f(\mathbf{x}' + t^* \nabla f(\mathbf{x}')) = \max_{t \geq 0} f(\mathbf{x}' + t \nabla f(\mathbf{x}'))$$

[*f(x)*= $f(\mathbf{x}' + t \nabla f(\mathbf{x}'))$, где $x_j = x_j' + t \left(\frac{\partial f}{\partial x_j} \right)_{\mathbf{x}=\mathbf{x}'}$, *j = 1, 2,. . . , n*, и что эти выражения для *xj* включает только константы и *t*, поэтому *f(x)* становится функцией только одной переменной t]

Итераций этой процедуры поиска градиента продолжаются $\nabla f(\mathbf{x})$ = 0 в небольших толерантности, т.е $\left|\frac{\partial f}{\partial x_j}\right| \leq \epsilon$: j= 1, 2, . . . , n

Чтобы подняться на вершину холма, близорукие, не может видеть вершину холма, чтобы идти непосредственно в этом направлении. Когда стоит на месте, землю вокруг ног хорошо видно, чтобы определить направление, в котором холме наклонной вверх наиболее резко □ ходить в прямую линию. Во время прогулки, состоянии сказать, когда прекращении прохождения (ноль склона в направлении). Предполагая, что холм вогнута, градиентной процедуры поиска можно использовать для восхождения к началу эффективно. 2-переменной проблема, где (*x1*, *x2*) представляет координаты (без учета высоты) текущего местоположения. Функция *f*(*x1*, *x2*) дает холм высотой в (*x1*, *x2*). Начало каждой итерации в текущем положении (текущее решение суда), определяя направление [в (*x1*, *x2*) система координат], в которой холм наклонной вверх наиболее резко (направление градиента) в этой точке

Тогда начните ходьбу в этом фиксированном направлении и продолжаться, пока продолжают расти. Остановка на новое место суда (решение), когда холм становится уровень в направлении, в котором готовятся сделать еще один итерации в другом направлении. Продолжить этих итераций, после зигзагообразные пути в гору, пока не дойдете до суда месте, где наклон = 0 во всех направлениях. Хилл

[*f(x1, x2)*] вогнута, то вершину холма. Найти *t**, значения *t*, которая максимизирует f в направлении $\nabla f(\mathbf{x})$ градиента, на каждой итерации. x и имеют фиксированные значения для максимизации, и *f(x)* вогнута □ проблема максимизации вогнутой функции одной переменной *t*. Решаемые 1D процедура поиска (где начальная нижняя граница *t* не может быть отрицательным из-за $t \geq 0$ ограничение). Если е простую функцию, то можно получить аналитическое решение, установив

Резюме градиентной процедуры поиска:

Инициализации: выбраϵ и любой начальной x решение суда. К первым правилом остановки. Итерация:

1. Экспресс$f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ в зависимости от т путем устан$x_j = x'_j + t\left(\frac{\partial f}{\partial x_j}\right)_{\mathbf{x}=\mathbf{x}'}$: *j=1, 2, . . . , n,*

а затем Подставляя эти выражения в *f(x)*.

2. Используйте одномерных процедура поиска (или исчисление), чтобы $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ которая максимизирует над $\mathbf{x}' = \mathbf{x}' + t^*\,\nabla f(\mathbf{x}')$

: $t \geq 0$.

3. ($\left|\frac{\partial f}{\partial x_j}\right| \leq \epsilon$. Затем перейдите на правило ост

Правила остановки: Оценка при x = x'. Убедитесь в том, для всех j = 1, 2,. . . , n.

Пример: с двумя переменными проблема:

Развернуть $f(\mathbf{x}) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$. Таким образом,

$$\frac{\partial f}{\partial x_1} = 2x_2 - 2x_1,$$

$$\frac{\partial f}{\partial x_2} = 2x_1 + 2 - 4x_2.$$

F (X) вогнута. Градиент процедуру поиска: X = (0, 0) выбран в качестве начального решения суда. Соответствующие частными производными = 0 и 2, градиент в этой точке: □ начать первой итерации, комплект: x1 = 0 + T (0) = 0,

x2 = 0 + T (2) = 2т,

а затем подс $f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) = f(0, 2t)$

$= 2(0)(2t) + 2(2t) - 0^2 - 2(2t)^2$

$= 4t - 8t^2.$

получить

потому что $f(0, 2t^*) = \max_{t \geq 0} f(0, 2t) = \max_{t \geq 0} \{4t - 8t^2\}$

и $\frac{d}{dt}(4t - 8t^2) = 4 - 16t = 0$

 Сброс x '= (0, 0) + 1/4 (0, 2) = 0, 1/2.
Для этого нового решения суда, градиент

$$\nabla f\left(0, \frac{1}{2}\right) = (1, 0)$$

 Для второй итерации, установите
x = (0, 1/2) + т (1, 0) = (т, 1/2), так

$$\begin{aligned} f(\mathbf{x}' + t\,\nabla f(\mathbf{x}')) &= f\left(0 + t, \frac{1}{2} + 0t\right) = f\left(t, \frac{1}{2}\right) \\ &= (2t)\left(\frac{1}{2}\right) + 2\left(\frac{1}{2}\right) - t^2 - 2\left(\frac{1}{2}\right)^2 \\ &= t - t^2 + \frac{1}{2}. \end{aligned}$$

потому что $f\left(t^*, \frac{1}{2}\right) = \max_{t \geq 0} f\left(t, \frac{1}{2}\right) = \max_{t \geq 0} \left\{t - t^2 + \frac{1}{2}\right\}$

и $\frac{d}{dt}\left(t - t^2 + \frac{1}{2}\right) = 1 - 2t = 0$

затем
t = 1/2, так Сброс $\mathbf{x}' = \left(0, \frac{1}{2}\right) + \frac{1}{2}(1, 0) = \left(\frac{1}{2}, \frac{1}{2}\right)$
Организация таблицы, которая суммирует 2 предшествующих итераций. На каждой итерации, 2-й столбец показывает текущее решение суда, и правая колонка показывает возможного нового решения суда, которое затем осуществляется вниз в 2 колонки для следующей итерации. Четвёртое Заголовки столбцов дает выражения для xj в терминах т, которые должны быть заменены в *f(x)* с получением указанного в пятой колонке. Продолжая таким образом, последующие решения процесс будет (1/2, 3/4), (3/4, 3/4), (3/4, 7/8), (7/8, 7/8), ... , Как на фиг. Потому что эти точки сходятся к x* = (1, 1), это решение является оптимальным решением, так как подтверждается тем, что

$$\nabla f(1, 1) = (0, 0)$$

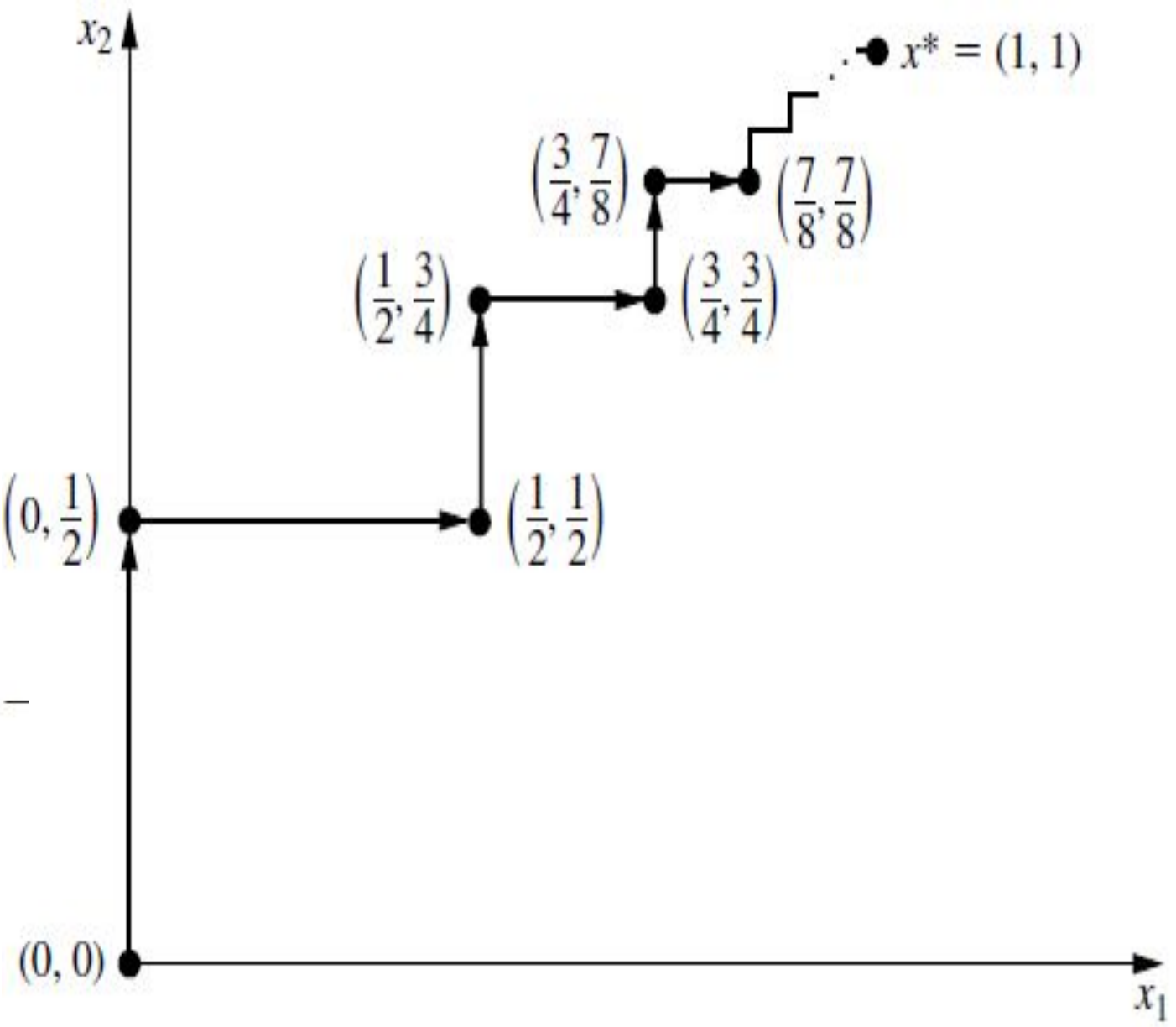
Применение градиентной процедуры поиска

Iteration	$\mathbf{x}'$	$\nabla f(\mathbf{x}')$	$\mathbf{x}' + t\,\nabla f(\mathbf{x}')$	$f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$	t^*	$\mathbf{x}' + t^*\,\nabla f(\mathbf{x}')$
1	(0, 0)	(0, 2)	(0, 2t)	$4t - 8t^2$	$\frac{1}{4}$	$\left(0, \frac{1}{2}\right)$
2	$\left(0, \frac{1}{2}\right)$	(1, 0)	$\left(t, \frac{1}{2}\right)$	$t - t^2 + \frac{1}{2}$	$\frac{1}{2}$	$\left(\frac{1}{2}, \frac{1}{2}\right)$

Поскольку этот сходящейся последовательности проб решения никогда не достигает своего предела, процедура останавливается где-то (в зависимости от) немного ниже (1, 1) в качестве окончательного приближения x*. Как видно из рис. предлагает, градиент зигзаги процедуру поиска оптимального решения, а не двигаться по прямой линии. Некоторые модификации разработанной методики, ускоряющих движение к оптимальному решению, приняв этот зигзаг поведения во внимание. Если *f(x)* не были вогнутая функция, процедура пои$f(\mathbf{x}' + t\,\nabla f(\mathbf{x}'))$ все равно бы сходятся к локальному максимуму . Только смена описания процедуры в этом случае является то, что t* Теперь будет соответствовать первый локальный максимум как t увеличивается от 0. Если целью было свести к минимуму *f(x)* вместо этого, одно изменение в процедуре будет двигаться в противоположном направлени$\mathbf{x}' = \mathbf{x}' - t^*\,\nabla f(\mathbf{x}')$а каждой итерации. Правило для получения следующей точке будет Reset . Толь$f(\mathbf{x}' - t\,\nabla f(\mathbf{x}'))$изменения в том, что t* тепер$f(\mathbf{x}' - t^*\,\nabla f(\mathbf{x}')) = \min_{t \geq 0} f(\mathbf{x}' - t\,\nabla f(\mathbf{x}'))$ значением t, что сводит к минимуму, то есть .

Иллюстрация процедуры поиска, когда

$f(x_1, x_2) = 2x_1x_2 + 2x_2 - x_1^2 - 2x_2^2$.



Каруша-Куна-Таккера (ККТ) условия Условной оптимизации

Как распознать оптимальное решение для задачи нелинейного программирования (с дифференцируемых функций). Каковы необходимые и достаточные условия, такие решения должны выполняться? Эти условия для безусловной оптимизации, сведенные в 1-ых двух строках таблицы. Эти условия для небольшого расширения оптимизации без ограничений, где только ограничения неотрицательности ограничения, показанный на 3-й строке таблицы. Как указано в последнем ряду, условия общем случае Karush-Куна-Таккера (или ККТ условиях).

Теорема. Предположим, что *f(x), g1(x), g2(x),. . . , gm(x)* дифференцируемы функции, удовлетворяющие некоторым условиям регулярности.

Problem	Necessary Conditions for Optimality	Also Sufficient if:
One-variable unconstrained	$\frac{df}{dx} = 0$	$f(x)$ concave
Multivariable unconstrained	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$	$f(\mathbf{x})$ concave
Constrained, nonnegativity constraints only	$\frac{\partial f}{\partial x_j} = 0 \quad (j = 1, 2, \ldots, n)$ (or ≤ 0 if $x_j = 0$)	$f(\mathbf{x})$ concave
General constrained problem	Karush-Kuhn-Tucker conditions	$f(\mathbf{x})$ concave and $g_i(\mathbf{x})$ convex $(i = 1, 2, \ldots, m)$

Необходимые и достаточные условия оптимальности

Тогда *x* = (x1*, x2*, ..., xn*)*

может быть оптимальным решением для задачи нелинейного программирования, только если существуют т номерами U1, U2,. . . , Гм, что все следующие условия удовлетворены ККТ:

$$\left.\begin{aligned} &1.\ \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \le 0 \\ &2.\ x_j^* \left(\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} \right) = 0 \end{aligned}\right\} \quad \text{at } \mathbf{x} = \mathbf{x}^*, \text{ for } j = 1, 2, \ldots, n.$$

$$\left.\begin{aligned} &3.\ g_i(\mathbf{x}^*) - b_i \le 0 \\ &4.\ u_i[g_i(\mathbf{x}^*) - b_i] = 0 \end{aligned}\right\} \quad \text{for } i = 1, 2, \ldots, m.$$

$$5.\ x_j^* \ge 0, \quad \text{for } j = 1, 2, \ldots, n.$$

$$6.\ u_i \ge 0, \quad \text{for } i = 1, 2, \ldots, m.$$

Оба условия 2 и 4 требует, чтобы произведение двух величин = 0 каждого из этих условий действительно говорит, что по крайней мере один из двух величин должна быть равна нулю. Условие 4 могут быть объединены с условием 3 выразить их в другую эквивалентную форму в качестве

$$g_i(\mathbf{x}^*) - b_i = 0 \qquad i = 1, 2, \ldots, m.$$
$$(\text{or} \le 0 \quad \text{if } u_i = 0),$$

Условие 2 могут быть объединены с условием 1 в качес$\frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i}{\partial x_j} = 0$

При m = 0 (без функциональных ограничений), это суммирование выпадает и сложенном состоянии (1, 2) сводится к состоянии приведены в таблице 3-й ряд. Для m > 0, каждый член в суммирование изменяет m = 0 условие включения эффекта соответствующие функциональные ограничения. В условиях 1, 2, 4 и 6, интерфейса, соответствующих двойственных переменных линейного программирования, и они имеют сопоставимые экономическую интерпретацию. *Ui* возникла в математический вывод как множителей Лагранжа. Условия 3 и 5 Убедитесь, что решение целесообразности. Другие условия устранить большинство возможных решений качестве возможных кандидатов на оптимальное решение. Удовлетворение этих условий не гарантирует, что решение является оптимальным. Правой колонке таблицы некоторых дополнительных предположениях выпуклости Для получения этой гарантии как обобщение теоремы. **Следствие**. *f(x)* является вогнутой функцией и *g1(x), g2(x), …, gm* (x) являются выпуклыми функциями (выпуклого программирования), где все эти функции удовлетворяют условиям регулярности. Тогда *x* = (x1*, x2*,*

Пример: с двумя переменными задачи нелинейного программирования:

Развернуть $f(\mathbf{x}) = \ln(x_1 + 1) + x_2$

подлежат $2x1 + x2 \leq 3$

И $x1 \geq 0, x2 \geq 0,$

где LN является натуральный логарифм. m = 1 (один функциональной связи) и *g1(x) = 2x1 + x2*, поэтому *g1(x)* является выпуклым. Кроме того, он может быть легко проверено, что *f(x)* вогнута. Следствие относится, так что любое решение, удовлетворяющее условиям ККТ будет оптимальным решением. Применение формулы, приведенные в теореме дает следующие условия ККТ для этого примера:

$$\mathbf{1}(j = 1). \quad \frac{1}{x_1 + 1} - 2u_1 \leq 0.$$

$$\mathbf{2}(j = 1). \quad x_1\left(\frac{1}{x_1 + 1} - 2u_1\right) = 0.$$

$$\mathbf{1}(j = 2). \quad 1 - u_1 \leq 0.$$

$$\mathbf{2}(j = 2). \quad x_2(1 - u_1) = 0.$$

$$\mathbf{3}. \quad 2x_1 + x_2 - 3 \leq 0.$$

$$\mathbf{4}. \quad u_1(2x_1 + x_2 - 3) = 0.$$

$$\mathbf{5}. \quad x_1 \geq 0, x_2 \geq 0.$$

$$\mathbf{6}. \quad u_1 \geq 0.$$

Шаги в решении ККТ условия для этого примера:

1. u1 ≥ 1, из условия 1 (j = 2). x1 ≥ 0, из условия 5.

2. Таким образом, $\frac{1}{x_1 + 1} - 2u_1$ <0.

3. Поэтому x1 = 0, из условия 2 (j = 1).

4. u1 ≠ 0 следует, что 2x1 + x2 - 3 = 0, из условия 4.

5. Шаги 3 и 4 следует, что x2 = 3.

6. x2 ≠ 0 следует, что u1 = 1, из условия 2 (j = 2).

7. Условия не нарушены x1 = 0, x2 = 3, u1 = 1.

 такое число u1 = 1 такие, что x1 = 0, x2 = 3, u1 = 1 удовлетворяют всем условиям x* = (0, 3) является оптимальным решением этой проблемы. Прогресс решить ККТ условия отличаются от одной проблемы к другой. Когда логика не очевидна, полезно рассмотреть отдельно различные случаи, когда каждая XJ и пользовательский интерфейс указанные быть либо = 0 или> 0, а затем не пытается каждом случае, пока один приводит к решению. В этом примере имеются восемь таких случаях соответствующий 8 комбинаций x1 = 0 в сравнении с x1> 0, x2 = 0 против x2> 0, u1 = 0, в зависимости от u1> 0. Каждый случай приводит к более простым заявлением и анализ условий.

Следующий случай: x1 = 0, x2 = 0, u1 = 0 ККТ Условия:

$$1(j=1).\ \frac{1}{0+1} \le 0.$$

$$1(j=2).\ 1-0 \le 0.$$

Противоречие.

3. 0 + 0 ≤ 3. (Все другие условия являются избыточными.)

Другие три случая, когда u1 = 0 также дать немедленную противоречий таким же образом, так что ни одно решение не доступно.

Случай x1 = 0, x2> 0, u1 = 0 противоречит условиям 1 (= 1), 1 (y = 2) и 2 (y = 2).

Случай x1> 0, x2 = 0, u1 = 0 противоречит условиям 1 (= 1), 2 (= 1) и 1 (y = 2).

Случай x1> 0, x2> 0, u1 = 0 противоречит условиям 1 (= 1), 2 (= 1), 1 (y = 2) и 2 (y = 2).

Случай x1> 0, x2> 0, u1> 0 позволяет удалять эти ненулевые множители из условий 2 (= 1), 2 (y = 2) и 4, который затем позволяет удалить условия 1 (= 1), 1 (y = 2) и 3 как излишним, как представлено следующее. ККТ условий случая x1> 0, x2> 0, u1> 0

1 (y = 1).

2 (y = 2). 1 - u1 = 0.

$$\frac{1}{x_1+1} - 2u_1$$

4. 2x1 + x2 - 3 = 0. (Все другие условия являются избыточными.)
Таким образом, U1 1, поэтому x1 1 2, что противоречит x1 0. Теперь предположим, что дело x1 0, x2 0, u1 0 пробуется следующий. ККТ условий случая x1 = 0, x2> 0, u1> 0
$\frac{1}{0+1} - 2u_1$
1 (y = 1).
2 (y = 2). 1 - u1 = 0.
5. 0 + x2 - 3 = 0. (Все другие условия являются избыточными.) Поэтому x1 = 0, x2 = 3, u1 = 1. Найдя решение, мы знаем, что никаких дополнительных случаев не нужно считаться. Для более сложных задач, невозможно, для получения оптимального решения непосредственно от ККТ условиях. Тем не менее, эти условия еще дать ценные подсказки относительно личности оптимального решения, и они также позволяют проверить, является ли предлагаемое решение может быть оптимальным. Там также много ценных косвенного применения ККТ условиях.

Одно из этих приложений возникает двойственность в теории, развитой для нелинейного программирования параллельных теории двойственности в линейном программировании. Для любого ограниченного задачи максимизации (прямая задача), ККТ условия могут быть использованы для определения тесно связана двойственная задача, которая задаче условной минимизации. Переменные в двойственной задачи состоят как из интерфейса множителей Лагранжа (I = 1,2, ..., т) и первичных переменных XJ (J = 1,2, ..., N). Частный случай, когда исходная задача является задачей линейного программирования, переменные XJ выпадают из двойственной задачи, и она становится знакомым двойственной задачи линейного программирования (UI переменных здесь соответствуют Yi переменных). Когда прямая задача выпуклого программирования является проблемой, можно установить связь между основной задачи и двойственной задачи аналогичны линейного программирования. Например, сильным свойством двойственности, в котором говорится, что оптимальная значений целевой функции двух задач совпадают, имеет место и здесь. Значения щ переменных в оптимальное решение для двойной проблема может снова быть интерпретированы как тень цены, то есть, они дают скорость, с которой оптимальное значение целевой функции для прямой задачи может быть увеличена (слегка) увеличение правой стороне соответствующего ограничения. Теория двойственности нелинейного программирования является сложная тема.