

第三章 非线性规划

§1 非线性规划

1.1 非线性规划的实例与定义

如果目标函数或约束条件中包含非线性函数, 就称这种规划问题为非线性规划问题。一般说来, 解非线性规划要比解线性规划问题困难得多。而且, 也不象线性规划有单纯形法这一通用方法, 非线性规划目前还没有适于各种问题的一般算法, 各个方法都有自己特定的适用范围。

下面通过实例归纳出非线性规划数学模型的一般形式, 介绍有关非线性规划的基本概念。

例1 (投资决策问题) 某企业有 n 个项目可供选择投资, 并且至少要对其中一个项目投资。已知该企业拥有总资金 A 元, 投资于第 $i (i = 1, \dots, n)$ 个项目需花资金 a_i 元, 并预计可收益 b_i 元。试选择最佳投资方案。

解 设投资决策变量为

$$x_i = \begin{cases} 1, & \text{决定投资第 } i \text{ 个项目} \\ 0, & \text{决定不投资第 } i \text{ 个项目} \end{cases}, \quad i = 1, \dots, n,$$

则投资总额为 $\sum_{i=1}^n a_i x_i$, 投资总收益为 $\sum_{i=1}^n b_i x_i$ 。因为该公司至少要对一个项目投资, 并且总的投资金额不能超过总资金 A , 故有限制条件

$$0 < \sum_{i=1}^n a_i x_i \leq A$$

另外, 由于 $x_i (i = 1, \dots, n)$ 只取值 0 或 1, 所以还有

$$x_i(1 - x_i) = 0, \quad i = 1, \dots, n.$$

最佳投资方案应是投资额最小而总收益最大的方案, 所以这个最佳投资决策问题归结为总资金以及决策变量 (取 0 或 1) 的限制条件下, 极大化总收益和总投资之比。因此, 其数学模型为:

$$\begin{aligned} \max Q &= \frac{\sum_{i=1}^n b_i x_i}{\sum_{i=1}^n a_i x_i} \\ \text{s. t. } &0 < \sum_{i=1}^n a_i x_i \leq A \\ &x_i(1 - x_i) = 0, \quad i = 1, \dots, n. \end{aligned}$$

上面例题是在一组等式或不等式的约束下, 求一个函数的最大值 (或最小值) 问题, 其中至少有一个非线性函数, 这类问题称之为非线性规划问题。可概括为一般形式

$$\begin{aligned} \min & f(x) \\ \text{s.t. } & h_j(x) \leq 0, \quad j = 1, \dots, q \\ & g_i(x) = 0, \quad i = 1, \dots, p \end{aligned} \quad (\text{NP})$$

其中 $x = [x_1 \cdots x_n]^T$ 称为模型 (NP) 的决策变量, f 称为目标函数, $g_i (i = 1, \cdots, p)$ 和 $h_j (j = 1, \cdots, q)$ 称为约束函数。另外, $g_i(x) = 0 (i = 1, \cdots, p)$ 称为等式约束, $h_j(x) \leq 0 (j = 1, \cdots, q)$ 称为不等式的约束。

对于一个实际问题, 在把它归结成非线性规划问题时, 一般要注意如下几点:

(i) 确定备选方案: 首先要收集同问题有关的资料和数据, 在全面熟悉问题的基础上, 确认什么是问题的可供选择的方案, 并用一组变量来表示它们。

(ii) 提出追求目标: 经过资料分析, 根据实际需要和可能, 提出要追求极小化或极大化的目标。并且, 运用各种科学和技术原理, 把它表示成数学关系式。

(iii) 给出价值标准: 在提出要追求的目标之后, 要确立所考虑目标的“好”或“坏”的价值标准, 并用某种数量形式来描述它。

(iv) 寻求限制条件: 由于所追求的目标一般都要在一定的条件下取得极小化或极大化效果, 因此还需要寻找出问题的所有限制条件, 这些条件通常用变量之间的一些不等式或等式来表示。

1.2 线性规划与非线性规划的区别

如果线性规划的最优解存在, 其最优解只能在其可行域的边界上达到 (特别是可行域的顶点上达到); 而非线性规划的最优解 (如果最优解存在) 则可能在其可行域的任意一点达到。

1.3 非线性规划的 Matlab 解法

Matlab 中非线性规划的数学模型写成以下形式

$$\begin{aligned} & \min f(x) \\ & \begin{cases} Ax \leq B \\ Aeq \cdot x = Beq \\ C(x) \leq 0 \\ Ceq(x) = 0 \end{cases}, \end{aligned}$$

其中 $f(x)$ 是标量函数, A, B, Aeq, Beq 是相应维数的矩阵和向量, $C(x), Ceq(x)$ 是非线性向量函数。

Matlab 中的命令是

$X = \text{FMINCON}(\text{FUN}, X_0, A, B, Aeq, Beq, LB, UB, \text{NONLCON}, \text{OPTIONS})$

它的返回值是向量 x , 其中 FUN 是用 M 文件定义的函数 $f(x)$; X_0 是 x 的初始值; A, B, Aeq, Beq 定义了线性约束 $A * X \leq B, Aeq * X = Beq$, 如果没有线性约束, 则 $A = [], B = [], Aeq = [], Beq = []$; LB 和 UB 是变量 x 的下界和上界, 如果上界和下界没有约束, 则 $LB = [], UB = []$, 如果 x 无下界, 则 LB 的各分量都为 $-\text{inf}$, 如果 x 无上界, 则 UB 的各分量都为 inf ; NONLCON 是用 M 文件定义的非线性向量函数 $C(x), Ceq(x)$; OPTIONS 定义了优化参数, 可以使用 Matlab 缺省的参数设置。

例2 求下列非线性规划

$$\begin{aligned} & \min f(x) = x_1^2 + x_2^2 + x_3^2 + 8 \\ \text{s.t. } & x_1^2 - x_2 + x_3^2 \geq 0 \\ & x_1 + x_2^2 + x_3^3 \leq 20 \\ & -x_1 - x_2^2 + 2 = 0 \end{aligned}$$

$$x_2 + 2x_3^2 = 3$$

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

解 (i) 编写 M 文件 fun1.m 定义目标函数

```
function f=fun1(x);  
f=sum(x.^2)+8;
```

(ii) 编写 M 文件 fun2.m 定义非线性约束条件

```
function [g,h]=fun2(x);  
g=[-x(1)^2+x(2)-x(3)^2  
x(1)+x(2)^2+x(3)^3-20]; %非线性不等式约束  
h=[-x(1)-x(2)^2+2  
x(2)+2*x(3)^2-3]; %非线性等式约束
```

(iii) 编写主程序文件 example2.m 如下:

```
options=optimset('largescale','off');  
[x,y]=fmincon('fun1',rand(3,1),[],[],[],[],zeros(3,1),[], ...  
'fun2', options)
```

就可以求得当 $x_1 = 0.5522, x_2 = 1.2033, x_3 = 0.9478$ 时, 最小值 $y = 10.6511$ 。

1.4 求解非线性规划的基本迭代格式

记 (NP) 的可行域为 K 。

若 $x^* \in K$, 并且

$$f(x^*) \leq f(x), \quad \forall x \in K$$

则称 x^* 是 (NP) 的整体最优解, $f(x^*)$ 是 (NP) 的整体最优值。如果有

$$f(x^*) < f(x), \quad \forall x \in K, x \neq x^*$$

则称 x^* 是 (NP) 的严格整体最优解, $f(x^*)$ 是 (NP) 的严格整体最优值。

若 $x^* \in K$, 并且存在 x^* 的邻域 $N_\delta(x^*)$, 使

$$f(x^*) \leq f(x), \quad \forall x \in N_\delta(x^*) \cap K,$$

则称 x^* 是 (NP) 的局部最优解, $f(x^*)$ 是 (NP) 的局部最优值。如果有

$$f(x^*) < f(x), \quad \forall x \in N_\delta(x^*) \cap K$$

则称 x^* 是 (NP) 的严格局部最优解, $f(x^*)$ 是 (NP) 的严格局部最优值。

由于线性规划的目标函数为线性函数, 可行域为凸集, 因而求出的最优解就是整个可行域上的全局最优解。非线性规划却不然, 有时求出的某个解虽是一部分可行域上的极值点, 但却并不一定是整个可行域上的全局最优解。

对于非线性规划模型 (NP), 可以采用迭代方法求它的最优解。迭代方法的基本思想是: 从一个选定的初始点 $x^0 \in R^n$ 出发, 按照某一特定的迭代规则产生一个点列 $\{x^k\}$, 使得当 $\{x^k\}$ 是有穷点列时, 其最后一个点是 (NP) 的最优解; 当 $\{x^k\}$ 是无穷点列时, 它有极限点, 并且其极限点是 (NP) 的最优解。

设 $x^k \in R^n$ 是某迭代方法的第 k 轮迭代点, $x^{k+1} \in R^n$ 是第 $k+1$ 轮迭代点, 记

$$x^{k+1} = x^k + t_k p^k \quad (1)$$

这里 $t_k \in R^1, p^k \in R^n, \|p^k\| = 1$, 并且 p^k 的方向是从点 x^k 向着点 x^{k+1} 的方向。式 (1) 就是求解非线性规划模型 (NP) 的基本迭代格式。

通常，我们把基本迭代格式(1)中的 p^k 称为第 k 轮搜索方向， t_k 为沿 p^k 方向的步长，使用迭代方法求解(NP)的关键在于，如何构造每一轮的搜索方向和确定适当的步长。

设 $\bar{x} \in R^n, p \neq 0$ ，若存在 $\delta > 0$ ，使

$$f(\bar{x} + tp) < f(\bar{x}), \forall t \in (0, \delta),$$

称向量 p 是 f 在点 \bar{x} 处的下降方向。

设 $\bar{x} \in R^n, p \neq 0$ ，若存在 $t > 0$ ，使

$$\bar{x} + tp \in K,$$

称向量 p 是点 \bar{x} 处关于 K 的可行方向。

一个向量 p ，若既是函数 f 在点 \bar{x} 处的下降方向，又是该点关于区域 K 的可行方向，称之为函数 f 在点 \bar{x} 处关于 K 的可行下降方向。

现在，我们给出用基本迭代格式(1)求解(NP)的一般步骤如下：

0° 选取初始点 x^0 ，令 $k := 0$ 。

1° 构造搜索方向，依照一定规划，构造 f 在点 x^k 处关于 K 的可行下降方向作为搜索方向 p^k 。

2° 寻求搜索步长。以 x^k 为起点沿搜索方向 p^k 寻求适当的步长 t_k ，使目标函数值有某种意义的下降。

3° 求出下一个迭代点。按迭代格式(1)求出

$$x^{k+1} = x^k + t_k p^k。$$

若 x^{k+1} 已满足某种终止条件，停止迭代。

4° 以 x^{k+1} 代替 x^k ，回到 1°步。

1.5 凸函数、凸规划

设 $f(x)$ 为定义在 n 维欧氏空间 $E^{(n)}$ 中某个凸集 R 上的函数，若对任何实数 $\alpha (0 < \alpha < 1)$ 以及 R 中的任意两点 $x^{(1)}$ 和 $x^{(2)}$ ，恒有

$$f(\alpha x^{(1)} + (1-\alpha)x^{(2)}) \leq \alpha f(x^{(1)}) + (1-\alpha)f(x^{(2)})$$

则称 $f(x)$ 为定义在 R 上的凸函数。

若对每一个 $\alpha (0 < \alpha < 1)$ 和 $x^{(1)} \neq x^{(2)} \in R$ 恒有

$$f(\alpha x^{(1)} + (1-\alpha)x^{(2)}) < \alpha f(x^{(1)}) + (1-\alpha)f(x^{(2)})$$

则称 $f(x)$ 为定义在 R 上的严格凸函数。

考虑非线性规划

$$\begin{cases} \min_{x \in R} f(x) \\ R = \{x \mid g_j(x) \leq 0, j = 1, 2, \dots, l\} \end{cases}$$

假定其中 $f(x)$ 为凸函数， $g_j(x) (j = 1, 2, \dots, l)$ 为凸函数，这样的非线性规划称为凸规划。

可以证明，凸规划的可行域为凸集，其局部最优解即为全局最优解，而且其最优解的集合形成一个凸集。当凸规划的目标函数 $f(x)$ 为严格凸函数时，其最优解必定唯一（假定最优解存在）。由此可见，凸规划是一类比较简单而又具有重要理论意义的非

线性规划。

§2 无约束问题

2.1 一维搜索方法

当用迭代法求函数的极小点时，常常用到一维搜索，即沿某一已知方向求目标函数的极小点。一维搜索的方法很多，常用的有：(1) 试探法（“成功—失败”，斐波那契法，0.618 法等）；(2) 插值法（抛物线插值法，三次插值法等）；(3) 微积分中的求根法（切线法，二分法等）。

考虑一维极小化问题

$$\min_{a \leq t \leq b} f(t) \quad (2)$$

若 $f(t)$ 是 $[a, b]$ 区间上的下单峰函数，我们介绍通过不断地缩短 $[a, b]$ 的长度，来搜索得 (2) 的近似最优解的两个方法。

为了缩短区间 $[a, b]$ ，逐步搜索得 (2) 的最优解 t^* 的近似值，我们可以采用以下途径：在 $[a, b]$ 中任取两个关于 $[a, b]$ 是对称的点 t_1 和 t_2 （不妨设 $t_2 < t_1$ ，并把它俩叫做搜索点），计算 $f(t_1)$ 和 $f(t_2)$ 并比较它们的大小。对于单峰函数，若 $f(t_2) < f(t_1)$ ，则必有 $t^* \in [a, t_1]$ ，因而 $[a, t_1]$ 是缩短了单峰区间；若 $f(t_1) < f(t_2)$ ，则有 $t^* \in [t_2, b]$ ，故 $[t_2, b]$ 是缩短了单峰区间；若 $f(t_2) = f(t_1)$ ，则 $[a, t_1]$ 和 $[t_2, b]$ 都是缩短了单峰。因此通过两个搜索点处目标函数值大小的比较，总可以获得缩短了单峰区间。对于新的单峰区间重复上述做法，显然又可获得更短的单峰区间。如此进行，在单峰区间缩短到充分小时，我们可以取最后的搜索点作为 (2) 最优解的近似值。

应该按照怎样的规则来选取探索点，使给定的单峰区间的长度能尽快地缩短？

2.1.1 Fibonacci 法

如用 F_n 表示计算 n 个函数值能缩短为单位长区间的最大原区间长度，可推出 F_n 满足关系

$$\begin{aligned} F_0 &= F_1 = 1 \\ F_n &= F_{n-2} + F_{n-1}, \quad n = 2, 3, \dots, \end{aligned}$$

数列 $\{F_n\}$ 称为 Fibonacci 数列， F_n 称为第 n 个 Fibonacci 数，相邻两个 Fibonacci 数之比 $\frac{F_{n-1}}{F_n}$ 称为 Fibonacci 分数。

当用斐波那契法以 n 个探索点来缩短某一区间时，区间长度的第一次缩短率为 $\frac{F_{n-1}}{F_n}$ ，其后各次分别为 $\frac{F_{n-2}}{F_{n-1}}, \frac{F_{n-3}}{F_{n-2}}, \dots, \frac{F_1}{F_2}$ 。由此，若 t_1 和 t_2 ($t_2 < t_1$) 是单峰区间 $[a, b]$ 中第 1 个和第 2 个探索点的话，那么应有比例关系

$$\frac{t_1 - a}{b - a} = \frac{F_{n-1}}{F_n}, \quad \frac{t_2 - a}{b - a} = \frac{F_{n-2}}{F_n}$$

从而

$$t_1 = a + \frac{F_{n-1}}{F_n}(b - a), \quad t_2 = a + \frac{F_{n-2}}{F_n}(b - a) \quad (3)$$

它们关于 $[a, b]$ 确是对称的点。

如果要求经过一系列探索点搜索之后，使最后的探索点和最优解之间的距离不超过精度 $\delta > 0$ ，这就要求最后区间的长度不超过 δ ，即

$$\frac{b-a}{F_n} \leq \delta \quad (4)$$

据此，我们应按照预先给定的精度 δ ，确定使 (4) 成立的最小整数 n 作为搜索次数，直到进行到第 n 个探索点时停止。

用上述不断缩短函数 $f(t)$ 的单峰区间 $[a, b]$ 的办法，来求得问题 (2) 的近似解，是 Kiefer(1953 年)提出的，叫做 Finbonacci 法，具体步骤如下：

1° 选取初始数据，确定单峰区间 $[a_0, b_0]$ ，给出搜索精度 $\delta > 0$ ，由 (4) 确定搜索次数 n 。

2° $k=1, a=a_0, b=b_0$ ，计算最初两个搜索点，按 (3) 计算 t_1 和 t_2 。

3° while $k < n-1$

$$f_1 = f(t_1), f_2 = f(t_2)$$

$$\text{if } f_1 < f_2$$

$$a = t_2; t_2 = t_1; t_1 = a + \frac{F(n-1-k)}{F(n-k)}(b-a)$$

else

$$b = t_1; t_1 = t_2; t_2 = b + \frac{F(n-1-k)}{F(n-k)}(a-b)$$

end

$$k = k + 1$$

end

4° 当进行至 $k = n-1$ 时，

$$t_1 = t_2 = \frac{1}{2}(a+b)$$

这就无法借比较函数值 $f(t_1)$ 和 $f(t_2)$ 的大小确定最终区间，为此，取

$$\begin{cases} t_2 = \frac{1}{2}(a+b) \\ t_1 = a + (\frac{1}{2} + \varepsilon)(b-a) \end{cases}$$

其中 ε 为任意小的数。在 t_1 和 t_2 这两点中，以函数值较小者为近似极小点，相应的函数值为近似极小值。并得最终区间 $[a, t_1]$ 或 $[t_2, b]$ 。

由上述分析可知，斐波那契法使用对称搜索的方法，逐步缩短所考察的区间，它能以尽量少的函数求值次数，达到预定的某一缩短率。

例 3 试用斐波那契法求函数 $f(t) = t^2 - t + 2$ 的近似极小点，要求缩短后的区间不大于区间 $[-1, 3]$ 的 0.08 倍。

程序留作习题。

2.1.2 0.618 法

若 $\omega > 0$ ，满足比例关系

$$\frac{\omega}{1} = \frac{1-\omega}{\omega}$$

称之为黄金分割数，其值为 $\omega = \frac{\sqrt{5}-1}{2} = 0.6180339887\cdots$ 。

黄金分割数 ω 和 Fibonacci 分数之间有着重要的关系

$$\omega = \lim_{n \rightarrow \infty} \frac{F_{n-1}}{F_n}。$$

现用不变的区间缩短率 0.618，代替斐波那契法每次不同的缩短率，就得到了黄金分割法（0.618 法）。这个方法可以看成是斐波那契法的近似，实现起来比较容易，效果也相当好，因而易于为人们所接受。

用 0.618 法求解，从第 2 个探索点开始每增加一个探索点作一轮迭代以后，原单峰区间要缩短 0.618 倍。计算 n 个探索点的函数值可以把原区间 $[a_0, b_0]$ 连续缩短 $n-1$ 次，因为每次的缩短率均为 μ ，故最后的区间长度为

$$(b_0 - a_0)\mu^{n-1}$$

这就是说，当已知缩短的相对精度为 δ 时，可用下式计算探索点个数 n ：

$$\mu^{n-1} \leq \delta$$

当然，也可以不预先计算探索点的数目 n ，而在计算过程中逐次加以判断，看是否已满足了提出的精度要求。

0.618 法是一种等速对称进行试探的方法，每次的探索点均取在区间长度的 0.618 倍和 0.382 倍处。

2.2 二次插值法

对极小化问题 (2)，当 $f(t)$ 在 $[a, b]$ 上连续时，可以考虑用多项式插值来进行一维搜索。它的基本思想是：在搜索区间中，不断用低次（通常不超过三次）多项式来近似目标函数，并逐步用插值多项式的极小点来逼近 (2) 的最优解。

2.3 无约束极值问题的解法

无约束极值问题可表述为

$$\min f(x), \quad x \in E^{(n)} \quad (5)$$

求解问题 (5) 的迭代法大体上分为两点：一是用到函数的一阶导数或二阶导数，称为解析法。另一是仅用到函数值，称为直接法。

2.3.1 解析法

2.3.1.1 梯度法（最速下降法）

对基本迭代格式

$$x^{k+1} = x^k + t_k p^k \quad (6)$$

我们总是考虑从点 x^k 出发沿哪一个方向 p^k ，使目标函数 f 下降得最快。微积分的知识告诉我们，点 x^k 的负梯度方向

$$p^k = -\nabla f(x^k),$$

是从点 x^k 出发使 f 下降最快的方向。为此，称负梯度方向 $-\nabla f(x^k)$ 为 f 在点 x^k 处的最速下降方向。

按基本迭代格式 (6)，每一轮从点 x^k 出发沿最速下降方向 $-\nabla f(x^k)$ 作一维搜索，来建立求解无约束极值问题的方法，称之为最速下降法。

这个方法的特点是，每轮的搜索方向都是目标函数在当前点下降最快的方向。同时，用 $\nabla f(x^k) = 0$ 或 $\|\nabla f(x^k)\| \leq \varepsilon$ 作为停止条件。其具体步骤如下：

1° 选取初始数据。选取初始点 x^0 ，给定终止误差，令 $k := 0$ 。

2° 求梯度向量。计算 $\nabla f(x^k)$ ，若 $\|\nabla f(x^k)\| \leq \varepsilon$ ，停止迭代，输出 x^k 。否则，进行 3°。

3° 构造负梯度方向。取

$$p^k = -\nabla f(x^k).$$

4° 进行一维搜索。求 t_k ，使得

$$f(x^k + t_k p^k) = \min_{t \geq 0} f(x^k + t p^k)$$

令 $x^{k+1} = x^k + t_k p^k, k := k+1$, 转 2°。

例 4 用最速下降法求解无约束非线性规划问题

$$\min f(x) = x_1^2 + 25x_2^2$$

其中 $x = (x_1, x_2)^T$ ，要求选取初始点 $x^0 = (2, 2)^T$ 。

解 (i) $\nabla f(x) = (2x_1, 50x_2)^T$

编写 M 文件 detaf.m，定义函数 $f(x)$ 及其梯度列向量如下

```
function [f,df]=detaf(x);
f=x(1)^2+25*x(2)^2;
df=[2*x(1)
    50*x(2)];
```

(ii) 编写主程序文件 zuisu.m 如下：

```
clc
x=[2;2];
[f0,g]=detaf(x);
while norm(g)>0.000001
    p=-g/norm(g);
    t=1.0;f=detaf(x+t*p);
    while f>f0
        t=t/2;
        f=detaf(x+t*p);
    end
    x=x+t*p;
    [f0,g]=detaf(x);
end
x,f0
```

2.3.1.2 Newton 法

考虑目标函数 f 在点 x^k 处的二次逼近式

$$f(x) \approx Q(x) = f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2} (x - x^k)^T \nabla^2 f(x^k) (x - x^k)$$

假定 Hesse 阵

$$\nabla^2 f(x^k) = \begin{bmatrix} \frac{\partial^2 f(x^k)}{\partial x_1^2} & \cdots & \frac{\partial^2 f(x^k)}{\partial x_1 \partial x_n} \\ \vdots & \cdots & \vdots \\ \frac{\partial^2 f(x^k)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x^k)}{\partial x_n^2} \end{bmatrix}$$

正定。

由于 $\nabla^2 f(x^k)$ 正定，函数 Q 的驻点 x^{k+1} 是 $Q(x)$ 的极小点。为求此极小点，令

$$\nabla Q(x^{k+1}) = \nabla f(x^k) + \nabla^2 f(x^k)(x^{k+1} - x^k) = 0,$$

即可解得

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

对照基本迭代格式 (1)，可知从点 x^k 出发沿搜索方向。

$$p^k = -[\nabla^2 f(x^k)]^{-1} \nabla f(x^k)$$

并取步长 $t_k = 1$ 即可得 $Q(x)$ 的最小点 x^{k+1} 。通常，把方向 p^k 叫做从点 x^k 出发的 Newton 方向。从一初始点开始，每一轮从当前迭代点出发，沿 Newton 方向并取步长为 1 的求解方法，称之为 Newton 法。其具体步骤如下：

- 1° 选取初始数据。选取初始点 x^0 ，给定终止误差 $\varepsilon > 0$ ，令 $k := 0$ 。
- 2° 求梯度向量。计算 $\nabla f(x^k)$ ，若 $\|\nabla f(x^k)\| \leq \varepsilon$ ，停止迭代，输出 x^k 。否则，进行 3°。

3° 构造 Newton 方向。计算 $[\nabla^2 f(x^k)]^{-1}$ ，取

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

4° 求下一迭代点。令 $x^{k+1} = x^k + p^k, k := k + 1$ ，转 2°。

例 5 用 Newton 法求解，

$$\min f(x) = x_1^4 + 25x_2^4 + x_1^2 x_2^2$$

选取 $x^0 = (2, 2)^T$ 。

解：(i) $\nabla f(x) = [4x_1^3 + 2x_1x_2^2 \quad 100x_2^3 + 2x_1^2x_2]^T$

$$\nabla^2 f = \begin{bmatrix} 12x_1^2 + 2x_2^2 & 4x_1x_2 \\ 4x_1x_2 & 300x_2^2 + 2x_1^2 \end{bmatrix}$$

(ii) 编写 M 文件 nwfun.m 如下：

```
function [f,df,d2f]=nwfun(x);
f=x(1)^4+25*x(2)^4+x(1)^2*x(2)^2;
df=[4*x(1)^3+2*x(1)*x(2)^2;100*x(2)^3+2*x(1)^2*x(2)];
d2f=[2*x(1)^2+2*x(2)^2,4*x(1)*x(2);
4*x(1)*x(2),300*x(2)^2+2*x(1)^2];
```

(III) 编写主程序文件 example5.m 如下：

```
clc
x=[2;2];
[f0,g1,g2]=nwfun(x);
while norm(g1)>0.00001
    p=-inv(g2)*g1;
    x=x+p;
```

```

    [f0,g1,g2]=nwfun(x);
end
x, f0

```

如果目标函数是非二次函数，一般地说，用 Newton 法通过有限轮迭代并不能保证可求得其最优解。

为了提高计算精度，我们在迭代时可以采用变步长计算上述问题，编写主程序文件 example5_2 如下：

```

clc,clear
x=[2;2];
[f0,g1,g2]=nwfun(x);
while norm(g1)>0.00001
    p=-inv(g2)*g1;p=p/norm(p);
    t=1.0;f=nwfun(x+t*p);
    while f>f0
        t=t/2;f=nwfun(x+t*p);
    end
    x=x+t*p;
    [f0,g1,g2]=nwfun(x);
end
x,f0

```

Newton 法的优点是收敛速度快；缺点是有时不好用而需采取改进措施，此外，当维数较高时，计算 $-\left[\nabla^2 f(x^k)\right]^{-1}$ 的工作量很大。

2.3.1.3 变尺度法

变尺度法 (Variable Metric Algorithm) 是近 20 多年来发展起来的，它不仅是求解无约束极值问题非常有效的算法，而且已被推广用来求解约束极值问题。由于它既避免了计算二阶导数矩阵及其求逆过程，又比梯度法的收敛速度快，特别是对高维问题具有显著的优越性，因而使变尺度法获得了很高的声誉。下面我们就来简要地介绍一种变尺度法—DFP 法的基本原理及其计算过程。这一方法首先由 Davidon 在 1959 年提出，后经 Fletcher 和 Powell 加以改进。

我们已经知道，牛顿法的搜索方向是 $-\left[\nabla^2 f(x^k)\right]^{-1}\nabla f(x^k)$ ，为了不计算二阶导数矩阵 $\left[\nabla^2 f(x^k)\right]$ 及其逆阵，我们设法构造另一个矩阵，用它来逼近二阶导数矩阵的逆阵 $\left[\nabla^2 f(x^k)\right]^{-1}$ ，这一类方法也称拟牛顿法 (Quasi-Newton Method)。

下面研究如何构造这样的近似矩阵，并将它记为 $\bar{H}^{(k)}$ 。我们要求：每一步都能以现有的信息来确定下一个搜索方向；每做一次迭代，目标函数值均有所下降；这些近似矩阵最后应收敛于解点处的 Hesse 阵的逆阵。

当 $f(x)$ 是二次函数时，其 Hesse 阵为常数阵 A ，任两点 x^k 和 x^{k+1} 处的梯度之差为

$$\nabla f(x^{k+1}) - \nabla f(x^k) = A(x^{k+1} - x^k)$$

或

$$x^{k+1} - x^k = A^{-1}[\nabla f(x^{k+1}) - \nabla f(x^k)]$$

对于非二次函数，仿照二次函数的情形，要求其 Hesse 阵的逆阵的第 $k+1$ 次近似矩阵 $\bar{H}^{(k+1)}$ 满足关系式

$$x^{k+1} - x^k = \bar{H}^{(k+1)}[\nabla f(x^{k+1}) - \nabla f(x^k)] \quad (7)$$

这就是常说的拟 Newton 条件。

若令

$$\begin{cases} \Delta G^{(k)} = \nabla f(x^{k+1}) - \nabla f(x^k) \\ \Delta x^k = x^{k+1} - x^k \end{cases} \quad (8)$$

则式 (7) 变为

$$\Delta x^k = \bar{H}^{(k+1)} \Delta G^{(k)}, \quad (9)$$

现假定 $\bar{H}^{(k)}$ 已知, 用下式求 $\bar{H}^{(k+1)}$ (设 $\bar{H}^{(k)}$ 和 $\bar{H}^{(k+1)}$ 均为对称正定阵);

$$\bar{H}^{(k+1)} = \bar{H}^{(k)} + \Delta \bar{H}^{(k)} \quad (10)$$

其中 $\Delta \bar{H}^{(k)}$ 称为第 k 次校正矩阵。显然, $\bar{H}^{(k+1)}$ 应满足拟 Newton 条件 (9), 即要求

$$\Delta x^k = (\bar{H}^{(k)} + \Delta \bar{H}^{(k)}) \Delta G^{(k)}$$

或

$$\Delta \bar{H}^{(k)} \Delta G^{(k)} = \Delta x^k - \bar{H}^{(k)} \Delta G^{(k)} \quad (11)$$

由此可以设想, $\Delta \bar{H}^{(k)}$ 的一种比较简单的形式是

$$\Delta \bar{H}^{(k)} = \Delta x^k (Q^{(k)})^T - \bar{H}^{(k)} \Delta G^{(k)} (W^{(k)})^T \quad (12)$$

其中 $Q^{(k)}$ 和 $W^{(k)}$ 为两个待定列向量。

将式 (12) 中的 $\Delta \bar{H}^{(k)}$ 代入 (11), 得

$$\Delta x^k (Q^{(k)})^T \Delta G^{(k)} - \bar{H}^{(k)} \Delta G^{(k)} (W^{(k)})^T \Delta G^{(k)} = \Delta x^k - \bar{H}^{(k)} \Delta G^{(k)}$$

这说明, 应使

$$(Q^{(k)})^T \Delta G^{(k)} = (W^{(k)})^T \Delta G^{(k)} = 1 \quad (13)$$

考虑到 $\Delta \bar{H}^{(k)}$ 应为对称阵, 最简单的办法就是取

$$\begin{cases} Q^{(k)} = \eta_k \Delta x^k \\ W^{(k)} = \xi_k \bar{H}^{(k)} \Delta G^{(k)} \end{cases} \quad (14)$$

由式 (13) 得

$$\eta_k (\Delta x^k)^T \Delta G^{(k)} = \xi_k (\Delta G^{(k)})^T \bar{H}^{(k)} \Delta G^{(k)} = 1 \quad (15)$$

若 $(\Delta x^k)^T \Delta G^{(k)}$ 和 $(\Delta G^{(k)})^T \bar{H}^{(k)} \Delta G^{(k)}$ 不等于零, 则有

$$\begin{cases} \eta_k = \frac{1}{(\Delta x^k)^T \Delta G^{(k)}} = \frac{1}{(\Delta G^{(k)})^T \Delta x^k} \\ \xi_k = \frac{1}{(\Delta G^{(k)})^T \bar{H}^{(k)} \Delta G^{(k)}} \end{cases} \quad (16)$$

于是, 得校正矩阵

$$\Delta \bar{H}^{(k)} = \frac{\Delta x^k (\Delta x^k)^T}{(\Delta G^{(k)})^T \Delta x^k} - \frac{\bar{H}^{(k)} \Delta G^{(k)} (\Delta G^{(k)})^T \Delta \bar{H}^{(k)}}{(\Delta G^{(k)})^T \bar{H}^{(k)} \Delta G^{(k)}} \quad (17)$$

从而得到

$$\bar{H}^{(k+1)} = \bar{H}^{(k)} + \frac{\Delta x^k (\Delta x^k)^T}{(\Delta G^{(k)})^T \Delta x^k} - \frac{\bar{H}^{(k)} \Delta G^{(k)} (\Delta G^{(k)})^T \Delta \bar{H}^{(k)}}{(\Delta G^{(k)})^T \bar{H}^{(k)} \Delta G^{(k)}} \quad (18)$$

上述矩阵称为尺度矩阵。通常, 我们取第一个尺度矩阵 $\bar{H}^{(0)}$ 为单位阵, 以后的尺度矩

阵按式 (18) 逐步形成。可以证明：

(i) 当 x^k 不是极小点且 $\bar{H}^{(k)}$ 正定时，式 (17) 右端两项的分母不为零，从而可按式 (18) 产生下一个尺度矩阵 $\bar{H}^{(k+1)}$ ；

(ii) 若 $\bar{H}^{(k)}$ 为对称正定阵，则由式 (18) 产生的 $\bar{H}^{(k+1)}$ 也是对称正定阵；

(iii) 由此推出 DFP 法的搜索方向为下降方向。

现将 DFP 变尺度法的计算步骤总结如下。

1° 给定初始点 x^0 及梯度允许误差 $\varepsilon > 0$ 。

2° 若 $\|\nabla f(x^0)\| \leq \varepsilon$ ，则 x^0 即为近似极小点，停止迭代，否则，转向下一步。

3° 令

$$\bar{H}^{(0)} = I \quad (\text{单位矩阵}),$$

$$p^0 = -\bar{H}^{(0)} \nabla f(x^0)$$

在 p^0 方向进行一维搜索，确定最佳步长 λ_0 ：

$$\min_{\lambda} f(x^0 + \lambda p^0) = f(x^0 + \lambda_0 p^0)$$

如此可得下一个近似点

$$x^1 = x^0 + \lambda_0 p^0$$

4° 一般地，设已得到近似点 x^k ，算出 $\nabla f(x^k)$ ，若

$$\|\nabla f(x^k)\| \leq \varepsilon$$

则 x^k 即为所求的近似解，停止迭代；否则，计算 $\bar{H}^{(k)}$ ：

$$\bar{H}^{(k)} = \bar{H}^{(k-1)} + \frac{\Delta x^{k-1} (\Delta x^{k-1})^T}{(\Delta G^{(k-1)})^T \Delta x^{k-1}} - \frac{\bar{H}^{(k-1)} \Delta G^{(k-1)} (G^{(k-1)})^T \Delta H^{(k-1)}}{(\Delta G^{(k-1)})^T \bar{H}^{(k-1)} \Delta G^{(k-1)}}$$

并令 $p^k = -\bar{H}^{(k)} \nabla f(x^k)$ ，在 p^k 方向上进行一维搜索，得 λ_k ，从而可得下一个近似点

$$x^{k+1} = x^k + \lambda_k p^k$$

5° 若 x^{k+1} 满足精度要求，则 x^{k+1} 即为所求的近似解，否则，转回 4°，直到求出某点满足精度要求为止。

2.3.2 直接法

在无约束非线性规划方法中，遇到问题的目标函数不可导或导函数的解析式难以表示时，人们一般需要使用直接搜索方法。同时，由于这些方法一般都比较直观和易于理解，因而在实际应用中常为人们所采用。下面我们介绍 Powell 方法。

这个方法主要由所谓基本搜索、加速搜索和调整搜索方向三部分组成，具体步骤如下：

1° 选取初始数据。选取初始点 x^0 ， n 个线性无关初始方向，组成初搜索方向组 $\{p^0, p^1, \dots, p^{n-1}\}$ 。给定终止误差 $\varepsilon > 0$ ，令 $k := 0$ 。

2° 进行基本搜索。令 $y^0 := x^k$ ，依次沿 $\{p^0, p^1, \dots, p^{n-1}\}$ 中的方向进行一维搜索。对应地得到辅助迭代点 y^1, y^2, \dots, y^n ，即

$$y^j = y^{j-1} + t_{j-1} p^{j-1}$$

$$f(y^{j-1} + t_{j-1} p^{j-1}) = \min_{t \geq 0} f(y^{j-1} + t p^{j-1}), \quad j = 1, \dots, n$$

3° 构造加速方向。令 $p^n = y^n - y^0$ ，若 $\|p^n\| \leq \varepsilon$ ，停止迭代，输出 $x^{k+1} = y^n$ 。
 否则进行 4°。

4° 确定调整方向。按下式

$$f(y^{m-1}) - f(y^m) = \max\{f(y^{j-1}) - f(y^j) | 1 \leq j \leq n\}$$

找出 m 。若

$$f(y^0) - 2f(y^n) + f(2y^n - y^0) < 2[f(y^{m-1}) - f(y^m)]$$

成立，进行 5°。否则，进行 6°。

5° 调整搜索方向组。令

$$x^{k+1} = y^n + t_n p^n : f(y^n + t_n p^n) = \min_{t \geq 0} f(y^n + t p^n).$$

同时，令

$$\{p^0, p^1, \dots, p^{n+1}\}_{k+1} := \{p^0, \dots, p^{m-1}, p^{m+1}, \dots, p^{n-1}, p^n\},$$

$k := k + 1$ ，转 2°。

6° 不调整搜索方向组。令 $x^{k+1} := y^n, k := k + 1$ ，转 2°。

2.4 Matlab 求无约束极值问题

在 Matlab 工具箱中，用于求解无约束极值问题的函数有 `fminunc` 和 `fminsearch`，用法介绍如下。

求函数的极小值

$$\min_x f(x),$$

其中 x 可以为标量或向量。

Matlab 中 `fminunc` 的基本命令是

$$[X, FVAL] = \text{FMINUNC}(\text{FUN}, X0, \text{OPTIONS}, P1, P2, \dots)$$

其中的返回值 X 是所求得的极小点， $FVAL$ 是函数的极小值，其它返回值的含义参见相关的帮助。`FUN` 是一个 M 文件，当 `FUN` 只有一个返回值时，它的返回值是函数 $f(x)$ ；当 `FUN` 有两个返回值时，它的第二个返回值是 $f(x)$ 的梯度向量；当 `FUN` 有三个返回值时，它的第三个返回值是 $f(x)$ 的二阶导数阵（Hessian 阵）。 $X0$ 是向量 x 的初始值，`OPTIONS` 是优化参数，可以使用缺省参数。`P1`，`P2` 是可以传递给 `FUN` 的一些参数。

例 6 求函数 $f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$ 的最小值。

解：编写 M 文件 `fun2.m` 如下：

```
function [f,g]=fun2(x);
f=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
g=[-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1));200*(x(2)-x(1)^2)];
```

编写主函数文件 `example6.m` 如下：

```
options = optimset('GradObj','on');
[x,y]=fminunc('fun2',rand(1,2),options)
```

即可求得函数的极小值。

在求极值时，也可以利用二阶导数，编写 M 文件 `fun3.m` 如下：

```
function [f,df,d2f]=fun3(x);
f=100*(x(2)-x(1)^2)^2+(1-x(1))^2;
df=[-400*x(1)*(x(2)-x(1)^2)-2*(1-x(1));200*(x(2)-x(1)^2)];
d2f=[-400*x(2)+1200*x(1)^2+2,-400*x(1)
      -400*x(1),200];
```

编写主函数文件example62.m如下:

```
options = optimset('GradObj','on','Hessian','on');  
[x,y]=fminunc('fun3',rand(1,2),options)
```

即可求得函数的极小值。

求多元函数的极值也可以使用 Matlab 的 fminsearch 命令, 其使用格式为:

```
[X,FVAL,EXITFLAG,OUTPUT]=FMINSEARCH(FUN,X0,OPTIONS,P1,P2,...)
```

例 7 求函数 $f(x) = \sin(x) + 3$ 取最小值时的 x 值。

解 编写 $f(x)$ 的 M 文件 fun4.m 如下:

```
function f=fun4(x);
```

```
f=sin(x)+3;
```

编写主函数文件example7.m如下:

```
x0=2;
```

```
[x,y]=fminsearch(@fun4,x0)
```

即求得在初值 2 附近的极小点及极小值。

§3 约束极值问题

带有约束条件的极值问题称为约束极值问题, 也叫规划问题。

求解约束极值问题要比求解无约束极值问题困难得多。为了简化其优化工作, 可采用以下方法: 将约束问题化为无约束问题; 将非线性规划问题化为线性规划问题, 以及能将复杂问题变换为较简单问题的其它方法。

库恩-塔克条件是非线性规划领域中最重要理论成果之一, 是确定某点为最优点的必要条件, 但一般说它并不是充分条件 (对于凸规划, 它既是最优点存在的必要条件, 同时也是充分条件)。

3.1 二次规划

若某非线性规划的目标函数为自变量 x 的二次函数, 约束条件又全是线性的, 就称这种规划为二次规划。

Matlab 中二次规划的数学模型可表述如下:

$$\begin{aligned} \min & \quad \frac{1}{2} x^T H x + f^T x, \\ \text{s.t.} & \quad \begin{cases} A x \leq b \\ A_{eq} \cdot x = b_{eq} \end{cases} \end{aligned}$$

这里 H 是实对称矩阵, f, b 是列向量, A 是相应维数的矩阵。

Matlab 中求解二次规划的命令是

```
[X,FVAL]=QUADPROG(H,f,A,b,Aeq,beq,LB,UB,X0,OPTIONS)
```

返回值 X 是决策向量 x 的值, 返回值 $FVAL$ 是目标函数在 x 处的值。(具体细节可以参看 Matlab 指令中运行 help quadprog 后的帮助)。

例 8 求解二次规划

$$\begin{cases} \min f(x) = 2x_1^2 - 4x_1x_2 + 4x_2^2 - 6x_1 - 3x_2 \\ x_1 + x_2 \leq 3 \\ 4x_1 + x_2 \leq 9 \\ x_1, x_2 \geq 0 \end{cases}$$

解 编写如下程序:

```

h=[4,-4;-4,8];
f=[-6;-3];
a=[1,1;4,1];
b=[3;9];
[x,value]=quadprog(h,f,a,b,[],[],zeros(2,1))

```

求得

$$x = \begin{bmatrix} 1.9500 \\ 1.0500 \end{bmatrix}, \text{Min } f(x) = -11.0250。$$

3.2 罚函数法

利用罚函数法，可将非线性规划问题的求解，转化为求解一系列无约束极值问题，因而也称这种方法为序列无约束最小化技术，简记为 SUMT (Sequential Unconstrained Minimization Technique)。

罚函数法求解非线性规划问题的思想是，利用问题中的约束函数作出适当的罚函数，由此构造出带参数的增广目标函数，把问题转化为无约束非线性规划问题。主要有两种形式，一种叫外罚函数法，另一种叫内罚函数法，下面介绍外罚函数法。

考虑问题：

$$\begin{aligned} & \min f(x) \\ & \text{s.t.} \begin{cases} g_i(x) \leq 0, i = 1, \dots, r, \\ h_j(x) \geq 0, j = 1, \dots, s, \\ k_m(x) = 0, m = 1, \dots, t \end{cases} \end{aligned}$$

取一个充分大的数 $M > 0$ ，构造函数

$$P(x, M) = f(x) + M \sum_{i=1}^r \max(g_i(x), 0) - M \sum_{i=1}^s \min(h_i(x), 0) + M \sum_{i=1}^t |k_i(x)|$$

$$(\text{或 } P(x, M) = f(x) + M \sum \left(\max \begin{pmatrix} G(x) \\ 0 \end{pmatrix} \right) - M \sum \left(\min \begin{pmatrix} H(x) \\ 0 \end{pmatrix} \right) + M \|K(x)\|$$

这里 $G(x) = [g_1(x) \ \dots \ g_r(x)]$, $H(x) = [h_1(x) \ \dots \ h_s(x)]$,

$K(x) = [k_1(x) \ \dots \ k_t(x)]$, Matlab 中可以直接利用 \max 、 \min 和 sum 函数。) 则以增广目标函数 $P(x, M)$ 为目标函数的无约束极值问题

$$\min P(x, M)$$

的最优解 x 也是原问题的最优解。

例 9 求下列非线性规划

$$\begin{cases} \min f(x) = x_1^2 + x_2^2 + 8 \\ x_1^2 - x_2 \geq 0 \\ -x_1 - x_2^2 + 2 = 0 \\ x_1, x_2 \geq 0. \end{cases}$$

解 (i) 编写 M 文件 test.m

```

function g=test(x);
M=50000;
f=x(1)^2+x(2)^2+8;
g=f-M*min(x(1),0)-M*min(x(2),0)-M*min(x(1)^2-x(2),0)+...

```

```
M*abs(-x(1)-x(2)^2+2);
```

或者是利用Matlab的求矩阵的极小值和极大值函数编写test.m如下：

```
function g=test(x);
```

```
M=50000;
```

```
f=x(1)^2+x(2)^2+8;
```

```
g=f-M*sum(min([x';zeros(1,2)]))-M*min(x(1)^2-x(2),0)+...
```

```
M*abs(-x(1)-x(2)^2+2);
```

我们也可以修改罚函数的定义，编写test.m如下：

```
function g=test(x);
```

```
M=50000;
```

```
f=x(1)^2+x(2)^2+8;
```

```
g=f-M*min(min(x),0)-M*min(x(1)^2-x(2),0)+M*(-x(1)-x(2)^2+2)^2;
```

(ii)在 Matlab 命令窗口输入

```
[x,y]=fminunc('test',rand(2,1))
```

即可求得问题的解。

3.3 Matlab 求约束极值问题

在 Matlab 优化工具箱中，用于求解约束最优化问题的函数有：fminbnd、fmincon、quadprog、fseminf、fminimax，上面我们已经介绍了函数 fmincon 和 quadprog。

3.3.1 fminbnd 函数

求单变量非线性函数在区间上的极小值

$$\min_x f(x), x \in [x_1, x_2]$$

Matlab 的命令为

```
[X,FVAL] = FMINBND(FUN,x1,x2,OPTIONS),
```

它的返回值是极小点 x 和函数的极小值。这里 fun 是用 M 文件定义的函数或 Matlab 中的单变量数学函数。

例 10 求函数 $f(x) = (x-3)^2 - 1, x \in [0,5]$ 的最小值。

解 编写 M 文件 fun5.m

```
function f=fun5(x);
```

```
f=(x-3)^2-1;
```

在 Matlab 的命令窗口输入

```
[x,y]=fminbnd('fun5',0,5)
```

即可求得极小点和极小值。

3.3.2 fseminf 函数

求

$$\min_x \{F(x) \mid C(x) \leq 0, Ceq(x) = 0, PHI(x, w) \leq 0\}$$

$$\text{s.t.} \begin{cases} A * x \leq B \\ Aeq * x = Beq \end{cases}$$

其中 $C(x), Ceq(x), PHI(x, w)$ 都是向量函数； w 是附加的向量变量， w 的每个分量都限定在某个区间内。

上述问题的 Matlab 命令格式为

```
X=FSEMINF(FUN,X0,NTHETA,SEMINFCON,A,B,Aeq,Beq)
```

其中 FUN 用于定义目标函数 $F(x)$ ；X0 为 x 的初始值；NTHETA 是半无穷约束 $PHI(x, w)$ 的个数；函数 SEMINFCON 用于定义非线性不等式约束 $C(x)$ ，非线性等

式约束 $Ceq(x)$ 和半无穷约束 $PHI(x, w)$ 的每一个分量函数, 函数 SEMINFCON 有两个输入参量 X 和 S , S 是推荐的取样步长, 也许不被使用。

例 11 求函数 $f(x) = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 + (x_3 - 0.5)^2$ 取最小值时的 x 值, 约束为:

$$K_1(x, w_1) = \sin(w_1 x_1) \cos(w_1 x_2) - \frac{1}{1000} (w_1 - 50)^2 - \sin(w_1 x_3) - x_3 \leq 1$$

$$K_2(x, w_2) = \sin(w_2 x_2) \cos(w_2 x_1) - \frac{1}{1000} (w_2 - 50)^2 - \sin(w_2 x_3) - x_3 \leq 1$$

$$1 \leq w_1 \leq 100, \quad 1 \leq w_2 \leq 100$$

解 (1) 编写 M 文件 fun6.m 定义目标函数如下:

```
function f=fun6(x,s);
```

```
f=sum((x-0.5).^2);
```

(2) 编写 M 文件 fun7.m 定义约束条件如下:

```
function [c,ceq,k1,k2,s]=fun7(x,s);
```

```
c=[];ceq=[];
```

```
if isnan(s(1,1))
```

```
    s=[0.2,0;0.2 0];
```

```
end
```

```
%取样值
```

```
w1=1:s(1,1):100;
```

```
w2=1:s(2,1):100;
```

```
%半无穷约束
```

```
k1=sin(w1*x(1)).*cos(w1*x(2))-1/1000*(w1-50).^2-sin(w1*x(3))-x(3)-1;
```

```
k2=sin(w2*x(2)).*cos(w2*x(1))-1/1000*(w2-50).^2-sin(w2*x(3))-x(3)-1;
```

```
%画出半无穷约束的图形
```

```
plot(w1,k1,'-',w2,k2,'+');
```

(3) 调用函数 fseminf

在 Matlab 的命令窗口输入

```
[x,y]=fseminf(@fun6,rand(3,1),2,@fun7)
```

即可。

3.3.3 fminimax 函数

求解 $\min_x \left\{ \max_{F_i} F(x) \right\}$

$$\text{s.t.} \begin{cases} A * x \leq b \\ Aeq * x = Beq \\ C(x) \leq 0 \\ Ceq(x) = 0 \\ LB \leq x \leq UB \end{cases}$$

其中 $F(x) = \{F_1(x), \dots, F_m(x)\}$ 。

上述问题的 Matlab 命令为

```
X=FMINIMAX(FUN,X0,A,B,Aeq,Beq,LB,UB,NONLCON)
```

例 12 求函数族 $\{f_1(x), f_2(x), f_3(x), f_4(x), f_5(x)\}$ 取极大极小值时的 x 值。其中：

$$\begin{cases} f_1(x) = 2x_1^2 + x_2^2 - 48x_1 - 40x_2 + 304 \\ f_2(x) = -x_1^2 - 3x_2^2 \\ f_3(x) = x_1 + 3x_2 - 18 \\ f_4(x) = -x_1 - x_2 \\ f_5(x) = x_1 + x_2 - 8 \end{cases}$$

解 (1) 编写 M 文件 fun8.m 定义向量函数如下：

```
function f=fun8(x);
f=[2*x(1)^2+x(2)^2-48*x(1)-40*x(2)+304
   -x(1)^2-3*x(2)^2
   x(1)+3*x(2)-18
   -x(1)-x(2)
   x(1)+x(2)-8];
```

(2) 调用函数 fminimax

```
[x,y]=fminimax(@fun8,rand(2,1))
```

3.3.4 利用梯度求解约束优化问题

例 13 已知函数 $f(x) = e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 2x_2 + 1)$ ，且满足非线性约束：

$$\begin{cases} x_1x_2 - x_1 - x_2 \leq -1.5 \\ x_1x_2 \geq -10 \end{cases}$$

求 $\min_x f(x)$

分析：当使用梯度求解上述问题时，效率更高并且结果更准确。

题目中目标函数的梯度为：

$$\begin{bmatrix} e^{x_1}(4x_1^2 + 2x_2^2 + 4x_1x_2 + 8x_1 + 6x_2 + 1) \\ e^{x_1}(4x_1 + 4x_2 + 2) \end{bmatrix}$$

解 (1) 编写 M 文件 fun9.m 定义目标函数及梯度函数：

```
function [f,df]=fun9(x);
f=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
df=[exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+8*x(1)+6*x(2)+1);exp(x(1))*(4*x(2)
+4*x(1)+2)];
```

(2) 编写 M 文件 fun10.m 定义约束条件及约束条件的梯度函数：

```
function [c,ceq,dc,dceq]=fun10(x);
c=[x(1)*x(2)-x(1)-x(2)+1.5;-x(1)*x(2)-10];
dc=[x(2)-1,-x(2);x(1)-1,-x(1)];
ceq=[];dceq=[];
```

(3) 调用函数 fmincon，编写主函数文件 example13.m 如下：

```
%采用标准算法
options=optimset('largescale','off');
%采用梯度
options=optimset(options,'GradObj','on','GradConstr','on');
[x,y]=fmincon(@fun9,rand(2,1),[],[],[],[],[],@fun10,options)
```

3.4 Matlab 优化工具箱的用户图形界面解法

Matlab 优化工具箱中的 `optimtool` 命令提供了优化问题的用户图形界面解法。`optimtool` 可应用到所有优化问题的求解，计算结果可以输出到 Matlab 工作空间中。

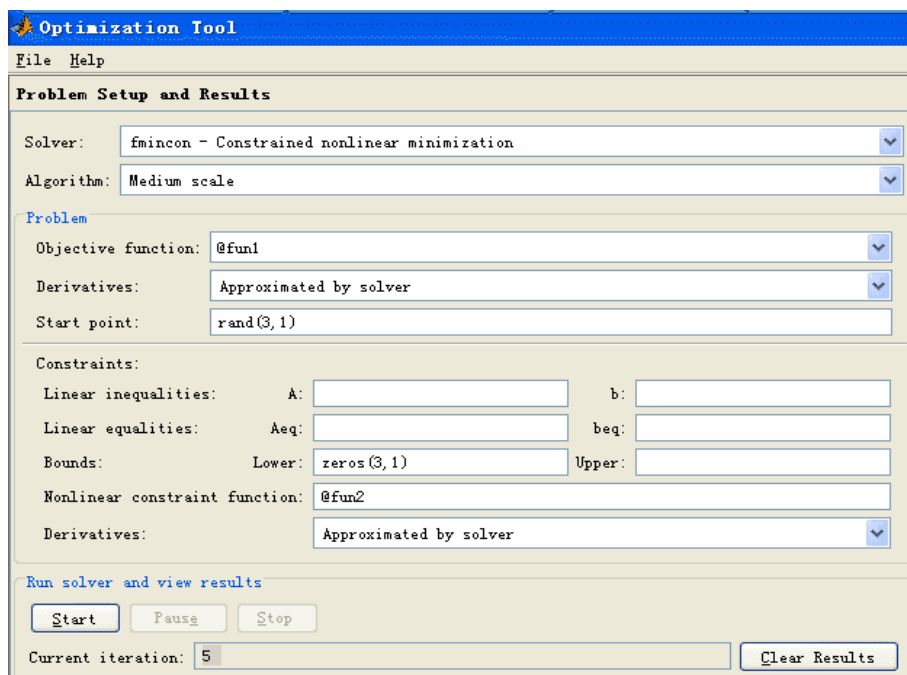


图 1 优化问题用户图形界面解法示意图

例 14 用 `optimtool` 重新求解例 1。

利用例 1 已经定义好的函数 `fun1` 和 `fun2`。在 Matlab 命令窗口运行 `optimtool`，就打开图形界面，如图 1 所示，填入有关的参数，未填入的参数取值为空或者为默认值，然后用鼠标点一下“start”按钮，就得到求解结果，再使用“file”菜单下的“Export to Workspace...”选项，把计算结果输出到 Matlab 工作空间中去。

§ 4 飞行管理问题

在约 10,000m 高空的某边长 160km 的正方形区域内，经常有若干架飞机作水平飞行。区域内每架飞机的位置和速度向量均由计算机记录其数据，以便进行飞行管理。当一架欲进入该区域的飞机到达区域边缘时，记录其数据后，要立即计算并判断是否会与区域内的飞机发生碰撞。如果会碰撞，则应计算如何调整各架（包括新进入的）飞机飞行的方向角，以避免碰撞。现假定条件如下：

- 1) 不碰撞的标准为任意两架飞机的距离大于 8km;
- 2) 飞机飞行方向角调整的幅度不应超过 30 度;
- 3) 所有飞机飞行速度均为每小时 800km;
- 4) 进入该区域的飞机在到达区域边缘时，与区域内飞机的距离应在 60km 以上;
- 5) 最多需考虑 6 架飞机;
- 6) 不必考虑飞机离开此区域后的状况。

请你对这个避免碰撞的飞行管理问题建立数学模型，列出计算步骤，对以下数据进行计算（方向角误差不超过 0.01 度），要求飞机飞行方向角调整的幅度尽量小。

设该区域 4 个顶点的座标为(0,0)，(160,0)，(160,160)，(0,160)。记录数据见表 1。

表 1 飞行记录数据

| 飞机编号 | 横座标 x | 纵座标 y | 方向角 (度) |
|------|---------|---------|---------|
| 1 | 150 | 140 | 243 |
| 2 | 85 | 85 | 236 |
| 3 | 150 | 155 | 220.5 |
| 4 | 145 | 50 | 159 |
| 5 | 130 | 150 | 230 |
| 新进入 | 0 | 0 | 52 |

注：方向角指飞行方向与 x 轴正向的夹角。

为方便以后的讨论，我们引进如下记号：

D 为飞行管理区域的边长；

Ω 为飞行管理区域，取直角坐标系使其为 $[0, D] \times [0, D]$ ；

a 为飞机飞行速度， $a = 800 \text{ km/h}$ ；

(x_i^0, y_i^0) 为第 i 架飞机的初始位置；

$(x_i(t), y_i(t))$ 为第 i 架飞机在 t 时刻的位置；

θ_i^0 为第 i 架飞机的原飞行方向角，即飞行方向与 x 轴夹角， $0 \leq \theta_i^0 < 2\pi$ ；

$\Delta\theta_i$ 为第 i 架飞机的方向角调整， $-\frac{\pi}{6} \leq \Delta\theta_i \leq \frac{\pi}{6}$ ；

$\theta_i = \theta_i^0 + \Delta\theta_i$ 为第 i 架飞机调整后的飞行方向角。

4.1 模型一

根据相对运动的观点在考察两架飞机 i 和 j 的飞行时，可以将飞机 i 视为不动而飞机 j 以相对速度

$$v_{ij} = v_j - v_i = (a \cos \theta_j - a \cos \theta_i, a \sin \theta_j - a \sin \theta_i) \quad (19)$$

相对于飞机 i 运动，对 (19) 式进行适当的化约可得

$$\begin{aligned} v &= 2a \sin \frac{\theta_j - \theta_i}{2} \left(-\sin \frac{\theta_j + \theta_i}{2}, \cos \frac{\theta_j + \theta_i}{2} \right) \\ &= 2a \sin \frac{\theta_j - \theta_i}{2} \left(\cos \left(\frac{\pi}{2} + \frac{\theta_j + \theta_i}{2} \right), \cos \left(\frac{\pi}{2} + \frac{\theta_j + \theta_i}{2} \right) \right) \end{aligned} \quad (20)$$

不妨设 $\theta_j \geq \theta_i$ ，此时相对飞行方向角为 $\beta_{ij} = \frac{\pi}{2} + \frac{\theta_i + \theta_j}{2}$ ，见图 2。

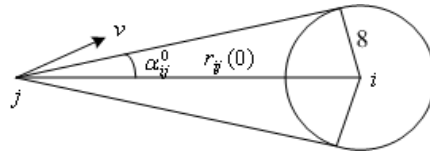


图 2 相对飞行方向角

由于两架飞机的初始距离为

$$r_{ij}(0) = \sqrt{(x_i^0 - x_j^0)^2 + (y_i^0 - y_j^0)^2} \quad (21)$$

$$\alpha_{ij}^0 = \arcsin \frac{8}{r_{ij}(0)} \quad (22)$$

则只要当相对飞行方向角 β_{ij} 满足

$$\alpha_{ij}^0 < \beta_{ij} < 2\pi - \alpha_{ij}^0 \quad (23)$$

时，两架飞机不可能碰撞（见图 2）。

记 β_{ij}^0 为调整前第 j 架飞机相对于第 i 架飞机的相对速度（矢量）与这两架飞机连线（从 j 指向 i 的矢量）的夹角（以连线矢量为基准，逆时针方向为正，顺时针方向为负）。则由式（23）知，两架飞机不碰撞的条件为

$$\left| \beta_{ij}^0 + \frac{1}{2}(\Delta\theta_i + \Delta\theta_j) \right| > \alpha_{ij}^0 \quad (24)$$

其中

$$\begin{aligned} \beta_{mn}^0 &= \text{相对速度 } v_{mn} \text{ 的幅角} - \text{从 } n \text{ 指向 } m \text{ 的连线矢量的幅角} \\ &= \arg \frac{e^{i\theta_n} - e^{i\theta_m}}{(x_m + iy_m) - (x_n + iy_n)} \end{aligned}$$

（注意 β_{mn}^0 表达式中的 i 表示虚数单位）这里我们利用复数的幅角，可以很方便地计算角度 β_{mn}^0 （ $m, n = 1, 2, \dots, 6$ ）。

本问题中的优化目标函数可以有不同的形式：如使所有飞机的最大调整量最小；所有飞机的调整量绝对值之和最小等。这里以所有飞机的调整量绝对值之和最小为目标函数，可以得到如下的数学规划模型：

$$\begin{aligned} \min \quad & \sum_{i=1}^6 |\Delta\theta_i| \\ \text{s.t.} \quad & \left| \beta_{ij}^0 + \frac{1}{2}(\Delta\theta_i + \Delta\theta_j) \right| > \alpha_{ij}^0, \quad i, j = 1, 2, \dots, 6, \quad i \neq j \\ & |\Delta\theta_i| \leq 30^\circ, \quad i = 1, 2, \dots, 6 \end{aligned}$$

利用如下的程序：

```
clc,clear
x0=[150 85 150 145 130 0];
y0=[140 85 155 50 150 0];
q=[243 236 220.5 159 230 52];
xy0=[x0; y0];
d0=dist(xy0); %求矩阵各个列向量之间的距离
d0(find(d0==0))=inf;
a0=asind(8./d0) %以度为单位的反函数
xy1=x0+i*y0
xy2=exp(i*q*pi/180)
for m=1:6
    for n=1:6
        if n~=m
            b0(m,n)=angle((xy2(n)-xy2(m))/(xy1(m)-xy1(n)));
        end
    end
end
```

```

end
end
b0=b0*180/pi;
dlmwrite('txt1.txt',a0,'delimiter','\t','newline','PC');
fid=fopen('txt1.txt','a');
fwrite(fid,~, 'char');          %往纯文本文件中写 LINGO 数据的分割符
dmlwrite('txt1.txt',b0,'delimiter','\t','newline','PC','-append','roffset', 1)

```

求得 α_{ij}^0 的值如表 2 所示。

表 2 α_{ij}^0 的值

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|----------|----------|----------|----------|----------|----------|
| 1 | 0 | 5.39119 | 32.23095 | 5.091816 | 20.96336 | 2.234507 |
| 2 | 5.39119 | 0 | 4.804024 | 6.61346 | 5.807866 | 3.815925 |
| 3 | 32.23095 | 4.804024 | 0 | 4.364672 | 22.83365 | 2.125539 |
| 4 | 5.091816 | 6.61346 | 4.364672 | 0 | 4.537692 | 2.989819 |
| 5 | 20.96336 | 5.807866 | 22.83365 | 4.537692 | 0 | 2.309841 |
| 6 | 2.234507 | 3.815925 | 2.125539 | 2.989819 | 2.309841 | 0 |

求得 β_{ij}^0 的值如表 3 所示。

表 3 β_{ij}^0 的值

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---------|---------|---------|---------|---------|---------|
| 1 | 0 | 109.26 | -128.25 | 24.18 | 173.07 | 14.475 |
| 2 | 109.26 | 0 | -88.871 | -42.244 | -92.305 | 9 |
| 3 | -128.25 | -88.871 | 0 | 12.476 | -58.786 | 0.31081 |
| 4 | 24.18 | -42.244 | 12.476 | 0 | 5.9692 | -3.5256 |
| 5 | 173.07 | -92.305 | -58.786 | 5.9692 | 0 | 1.9144 |
| 6 | 14.475 | 9 | 0.31081 | -3.5256 | 1.9144 | 0 |

上述飞行管理的数学规划模型可如下输入 LINGO 求解：

```

model:
sets:
plane/1..6/:delta;
link(plane,plane):alpha,beta;
endsets
data:
alpha=@file('txt1.txt'); !需要在alpha的数据后面加上分隔符"~";
beta=@file('txt1.txt');
enddata
min=@sum(plane:@abs(delta));
@for(plane:@bnd(-30,delta,30));
@for(link(i,j)|i#ne#j:@abs(beta(i,j)+0.5*delta(i)+0.5*delta(j))>a
lpha(i,j));
end

```

求得的最优解为 $\Delta\theta_3 = 2.83858^\circ$ ， $\Delta\theta_5 = -21.0138^\circ$ ， $\Delta\theta_6 = 0.7908^\circ$ ，其它调整角度为 0。

4.2 模型二

两架飞机 i, j 不发生碰撞的条件为

$$(x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2 > 64 \quad (25)$$

$$1 \leq i \leq 5, \quad i+1 \leq j \leq 6, \quad 0 \leq t \leq \min\{T_i, T_j\}$$

其中 T_i, T_j 分别表示第 i, j 架飞机飞出正方形区域边界的时刻。这里

$$x_i(t) = x_i^0 + at \cos \theta_i, \quad y_i(t) = y_i^0 + at \sin \theta_i, \quad i = 1, 2, \dots, n;$$

$$\theta_i = \theta_i^0 + \Delta \theta_i, \quad |\Delta \theta_i| \leq \frac{\pi}{6}, \quad i = 1, 2, \dots, n;$$

下面我们把约束条件 (25) 加强为对所有的时间 t 都成立, 记

$$l_{i,j} = (x_i(t) - x_j(t))^2 + (y_i(t) - y_j(t))^2 - 64 = \tilde{a}(i, j)t^2 + \tilde{b}(i, j)t + \tilde{c}(i, j)$$

$$\text{其中 } \tilde{a}(i, j) = 4a^2 \sin^2 \frac{\theta_i - \theta_j}{2},$$

$$\tilde{b}(i, j) = 2a[(x_i(0) - x_j(0))(\cos \theta_i - \cos \theta_j) + (y_i(0) - y_j(0))(\sin \theta_i - \sin \theta_j)]$$

$$\tilde{c}(i, j) = (x_i(0) - x_j(0))^2 + (y_i(0) - y_j(0))^2 - 64$$

则两架 i, j 飞机不碰撞的条件是

$$\Delta(i, j) = \tilde{b}(i, j)^2 - 4\tilde{a}(i, j)\tilde{c}(i, j) < 0 \quad (26)$$

这样我们建立如下的非线性规划模型

$$\begin{aligned} & \sum_{i=1}^6 (\Delta \theta_i)^2 \\ \text{s.t. } & \Delta(i, j) < 0, \quad 1 \leq i \leq 5, \quad i+1 \leq j \leq 6 \\ & |\Delta \theta_i| \leq \frac{\pi}{6}, \quad i = 1, 2, \dots, 6 \end{aligned}$$

习 题 三

1. 用最速下降法 (梯度法) 求函数:

$$f(x) = 4x_1 + 6x_2 - 2x_1^2 - 2x_1x_2 - 2x_2^2$$

的极大点。给定初始点 $x^0 = (1, 1)^T$ 。

2. 试用牛顿法求解:

$$\min f(x) = -\frac{1}{x_1^2 + x_2^2 + 2}$$

取初始点 $x^{(0)} = (4, 0)^T$, 并将采用变步长和采用固定步长 $\lambda = 1.0$ 时的情形做比较。

3. 某工厂向用户提供发动机, 按合同规定, 其交货数量和日期是: 第一季度末交 40 台, 第二季末交 60 台, 第三季末交 80 台。工厂的最大生产能力为每季 100 台, 每季的生产费用是 $f(x) = 50x + 0.2x^2$ (元), 此处 x 为该季生产发动机的台数。若工厂生产的多, 多余的发动机可移到下季向用户交货, 这样, 工厂就需支付存贮费, 每台发动机每季的存贮费为 4 元。问该厂每季应生产多少台发动机, 才能既满足交货合同, 又使工厂所花费的费用最少 (假定第一季度开始时发动机无存货)。

4. 用 Matlab 的非线性规划命令 `fmincon` 求解飞行管理问题的模型二。

5. 用罚函数法求解飞行管理问题的模型二。

6. 求下列问题的解

$$\begin{aligned} \max \quad & f(\mathbf{x}) = 2x_1 + 3x_1^2 + 3x_2 + x_2^2 + x_3 \\ \text{s.t.} \quad & x_1 + 2x_1^2 + x_2 + 2x_2^2 + x_3 \leq 10 \\ & x_1 + x_1^2 + x_2 + x_2^2 - x_3 \leq 50 \\ & 2x_1 + x_1^2 + 2x_2 + x_3 \leq 40 \\ & x_1^2 + x_3 = 2 \\ & x_1 + 2x_2 \geq 1 \\ & x_1 \geq 0, \quad x_2, x_3 \text{ 不约束} \end{aligned}$$

7.

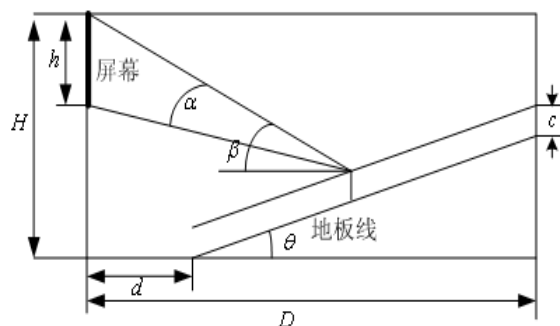


图3 影院剖面示意图

图3为影院的剖面示意图，座位的满意程度主要取决于视角 α 和仰角 β 。视角 α 是观众眼睛到屏幕上、下边缘视线的夹角， α 越大越好；仰角 β 是观众眼睛到屏幕上边缘视线与水平线的夹角， β 太大使人的头部过分上仰，引起不舒服感，一般要求 β 不超过 30° 。

记影院屏幕高 h ，上边缘距地面高 H ，地板线倾角 θ ，第一排和最后一排座位与屏幕水平距离分别为 d 和 D ，观众平均座高为 c （指眼睛到地面的距离）。已知参数 $h=1.8$ ， $H=5$ ， $d=4.5$ ， $D=19$ ， $c=1.1$ （单位：m）。

- (1) 地板线倾角 $\theta=10^\circ$ ，问最佳座位在什么地方？
- (2) 求地板线倾角（一般不超过 20° ），使所有观众的平均满意程度最大。
- (3) 地板线设计成什么形状可以进一步提高观众的满意程度。