最优化方法及其 Matlab 程序设计

第十二章 序列二次规划法

本章考虑求解一般非线性优化问题

$$\begin{cases} \min\ f(x), \\ \text{s.t.}\ \ h_i(x) = 0,\ i \in E = \{1, \cdots, l\}, \\ \qquad g_i(x) \geq 0,\ i \in I = \{1, \cdots, m\} \end{cases} \tag{12.1}$$

的序列二次规划法 (SQP, Sequential Quadratic Programming). SQP 方法是求解约束优化问题最有效的算法之一, 其基本思想是: 在每一迭代步通过求解一个二次规划子问题来确立一个下降方向, 以减少价值函数来取得步长, 重复这些步骤直到求得原问题的解. 这也是之所以称为序列二次规划法的由来. 本章主要介绍 SQP 方法的基本思想、迭代步骤和收敛性分析. 首先介绍求解等式约束优化问题的牛顿-拉格朗日法.

§12.1 牛顿-拉格朗日法

§12.1.1 牛顿-拉格朗日法的基本理论

考虑纯等式约束的优化问题

$$\begin{cases} \min\ f(x), \\ \text{s.t.}\ \ h_i(x)=0,\ i\in E=\{1,\cdots,l\}, \end{cases} \tag{12.2}$$

其中 $f:\mathbb{R}^n\to\mathbb{R}$, $h_i:\mathbb{R}^n\to\mathbb{R}\,(i\in E)$ 都是二阶连续可微的实函数. 记 $h(x)=(h_1(x),\cdots,h_l(x))^T$, 则不难写出该问题的拉格朗日函数为

$$L(x,\mu)=f(x)-\sum_{i=1}^{l}\mu_i h_i(x)=f(x)-\mu^T h(x),$$

其中 $\mu=(\mu_1,\cdots,\mu_l)^T$ 为拉格朗日乘子向量. 约束函数 $h(x)$ 的梯度矩阵为

$$\nabla h(x)=\big[\nabla h_1(x),\cdots,\nabla h_l(x)\big],$$

则 $h(x)$ 的 Jacobi 矩阵为 $A(x) = \nabla h(x)^T$. 根据问题 (12.2) 的 KT 条件, 可以得到如下的方程组

$$\nabla L(x,\mu) = \begin{bmatrix} \nabla_x L(x,\mu) \\ \nabla_\mu L(x,\mu) \end{bmatrix} = \begin{bmatrix} \nabla f(x) - A(x)^T \mu \\ -h(x) \end{bmatrix} = 0. \qquad (12.3)$$

现在考虑用牛顿法求解上述的非线性方程组 (12.3). 记函数 $\nabla L(x,\mu)$ 的 Jacobi 矩阵为

$$N(x,\mu) = \begin{bmatrix} W(x,\mu) & -A(x)^T \\ -A(x) & 0 \end{bmatrix}, \qquad (12.4)$$

其中

$$W(x,\mu) = \nabla^2_{xx} L(x,\mu) = \nabla^2 f(x) - \sum_{i=1}^{l} \mu_i \nabla^2 h_i(x)$$

是拉格朗日函数 $L(x,\mu)$ 关于 x 的 Hesse 阵. (12.4) 所定义的矩阵 $N(x,\mu)$ 亦称之为 KT 矩阵. 对于给定的点 $z_k = (x_k, \mu_k)$, 牛顿法的迭

代格式为

$$z_{k+1} = z_k + p_k, \tag{12.5}$$

其中 $p_k = (d_k, \nu_k)$ 满足下面的线性方程组:

$$N(x_k, \mu_k)p_k = -\nabla L(x_k, \mu_k),$$

即

$$\begin{bmatrix} W(x_k, \mu_k) & -A(x_k)^T \\ -A(x_k) & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \nu_k \end{bmatrix} = \begin{bmatrix} -\nabla f(x_k) + A(x_k)^T \mu_k \\ h(x_k) \end{bmatrix}. \tag{12.6}$$

不难发现, 只要矩阵 $A(x_k)$ 行满秩且 $W(x_k, \mu_k)$ 是正定的, 那么方程组 (12.6) 的系数矩阵是非奇异的, 且该方程有唯一解. 由于 KT 条件 (12.3) 是拉格朗日函数稳定点的条件, 所以人们通常把基于求解方程 (12.3) 的优化方法称为拉格朗日方法. 特别地, 如果用牛顿法求解

该方程组, 则称之为牛顿-拉格朗日方法. 因此, 根据牛顿法的性质, 该方法具有局部二次收敛性质.

下面写出牛顿-拉格朗日方法的详细算法步骤.

算法 12.1 (牛顿-拉格朗日方法)

步 0 选取 $x_0 \in \mathbb{R}^n$, $\mu_0 \in \mathbb{R}^l$, $\rho, \gamma \in (0,1)$, $0 \leq \varepsilon \ll 1$. 令 $k := 0$.

步 1 计算 $\|\nabla L(x_k, \mu_k)\|$ 的值. 若 $\|\nabla L(x_k, \mu_k)\| \leq \varepsilon$, 停算. 否则, 转步 2.

步 2 解方程组 (12.6) 得 $p_k = (d_k, \nu_k)$.

步 3 若

$$\|\nabla L(x_k + d_k, \mu_k + \nu_k)\|^2 \leq (1-\gamma)\|\nabla L(x_k, \mu_k)\|^2, \tag{12.7}$$

则置 $\alpha_k := 1$, 转步 5; 否则, 转步 4.

步 4 令 m_k 是使下面的不等式成立的最小非负整数 m:

$$\|\nabla L(x_k+\rho^m d_k, \mu_k+\rho^m \nu_k)\|^2 \le (1-\gamma\rho^m)\|\nabla L(x_k,\mu_k)\|^2, \qquad (12.8)$$

置 $\alpha_k = \rho^{m_k}$.

步 5 令 $x_{k+1} = x_k + \alpha_k d_k$, $\mu_{k+1} = \mu_k + \alpha_k \nu_k$. 置 $k := k+1$, 转步 1.

§12.1.2 牛顿-拉格朗日法的 Matlab 程序

本小节通过一个具体的例子来介绍算法 12.1 (牛顿-拉格朗日方法) 的 Matlab 实现.

例 12.1 用算法 12.1 编程计算下列最优化问题的极小点

$$\begin{cases} \min \ f(x) = \mathrm{e}^{x_1x_2x_3x_4x_5} - \dfrac{1}{2}(x_1^3 + x_2^3 + 1)^2, \\ \text{s.t.} \ \ x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 = 0, \\ \qquad x_2x_3 - 5x_4x_5 = 0, \\ \qquad x_1^3 + x_2^3 + 1 = 0. \end{cases}$$

该问题有最优解 $x^* = (-1.71, 1.59, 1.82, -0.763, -0.763)^T$, 最优值 $f(x^*) = 0.0539$.

解 编制 Matlab 程序如下

```
function [x,mu,val,mh,k]=newtlagr(x0,mu0)
% 功能:用牛顿-拉格朗日法求解约束优化问题:
%    min f(x), s.t. h_i(x)=0, i=1,..., l.
%输入:x0是初始点, mu0是乘子向量的初始值
%输出: x,mu分别是近似最优点及相应的乘子,
%val是最优值,mh是约束函数的模,k是迭代次数.
maxk=500; %最大迭代次数
```

```
n=length(x0); l=length(mu0);
rho=0.5; gamma=0.4;
x=x0; mu=mu0;
k=0;  epsilon=1e-8;
while(k<maxk)
    dl=dla(x,mu);  %计算乘子函数的梯度
    if(norm(dl)<epsilon), break; end  %检验终止准则
    N=N1(x,mu);   % 计算拉格朗日矩阵
    dz=-N\dl;    %解方程组得搜索方向
    dx=dz(1:n); du=dz(n+1:n+l);
    m=0; mk=0;
    while(m<20)   % Armijo搜索
        if(norm(dla(x+rho^m*dx,mu+rho^m*du))^2<=(1-gamma*rho^m)*norm(dl)^2)
            mk=m; break;
        end
        m=m+1;
    end
    x=x+rho^mk*dx; mu=mu+rho^mk*du;
    k=k+1;
end
```

```
val=f1(x);
mh=norm(h1(x),inf);
%%%%%%%%% 拉格朗日函数 L(x,mu) %%%%%%%%%%%%
function l=la(x,mu)
f=f1(x);   %调用目标函数文件
h=h1(x);  %调用约束函数文件
l=f-mu'*h; % 计算乘子函数
%%%%%%%%% 拉格朗日函数的梯度 %%%%%%%%%%%%
function dl=dla(x,mu)
df=df1(x); %调用目标函数梯度文件
h=h1(x);  %调用约束函数文件
dh=dh1(x);  %调用约束函数Jacobi矩阵文件
dl=[df-dh'*mu; -h];  %计算乘子函数梯度文件
%%%%%%%%% 拉格朗日函数的Hesse阵 %%%%%%%%%%%
function d2l=d2la(x,mu)
d2f=d2f1(x);   %调用目标函数Hesse阵文件
[d2h1,d2h2,d2h3]=d2h(x); %调用约束函数二阶导数文件
d2l=d2f-mu(1)*d2h1-mu(2)*d2h2-mu(3)*d2h3; %计算乘子函数的Hesse阵
%%%%%%%%% 系数矩阵N(x,mu) %%%%%%%%%%
function N=N1(x,mu)  %计算拉格朗日矩阵
```

```
l=length(mu);
d2l=d2la(x,mu);   dh=dh1(x);
N=[d2l, -dh'; -dh, zeros(l,l)];
%%%%%%%%% 目标函数 f(x) %%%%%%%%%%%%
function f=f1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
f=exp(s)-0.5*(x(1)^3+x(2)^3+1)^2;
%%%%%%%%% 约束函数 h(x) %%%%%%%%%%%%
function h=h1(x)
h=[x(1)^2+x(2)^2+x(3)^2+x(4)^2+x(5)^2-10;x(2)*x(3)-5*x(4)*x(5);x(1)^3+x(2)^3+1];
%%%%%%%%% 目标函数 f(x) 的梯度%%%%%%%%%%%%
function df=df1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
df(1)=s/(x(1))*exp(s)-3*(x(1)^3+x(2)^3+1)*x(1)^2;
df(2)=s/(x(2))*exp(s)-3*(x(1)^3+x(2)^3+1)*x(2)^2;
df(3)=s/(x(3))*exp(s); df1(4)=s/(x(4))*exp(s);
df(5)=s/(x(5))*exp(s);
df=df(:);
%%%%%%%%% 约束函数 h(x) 的Jacobi矩阵A(x)%%%%%%%%%%%%
function dh=dh1(x)
```

```
dh=[2*x(1),2*x(2),2*x(3),2*x(4),2*x(5);0,x(3),x(2),-5*x(5),-5*x(4);...
    3*x(1)^2,3*x(2)^2,0,0,0];
%%%%%%%% 目标函数 f(x) 的Hesse阵%%%%%%%%%%%%
function d2f=d2f1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
d2f=[(s/(x(1)))^2*exp(s)-6*x(1)*(x(1)^3+x(2)^3+1)-9*x(1)^4,...
   (1+s)*x(3)*x(4)*x(5)*exp(s)-9*x(1)^2*x(2)^2,(1+s)*x(2)*x(4)*x(5)*exp(s),...
   (1+s)*x(2)*x(3)*x(5)*exp(s), (1+s)*x(2)*x(3)*x(4)*exp(s); ...
   (1+s)*x(3)*x(4)*x(5)*exp(s)-9*x(1)^2*x(2)^2, ...
   (s/(x(2)))^2*exp(s)-6*x(2)*(x(1)^3+x(2)^3+1)-9*x(2)^4,...
   (1+s)*x(1)*x(4)*x(5)*exp(s),(1+s)*x(1)*x(3)*x(5)*exp(s),...
   (1+s)*x(1)*x(3)*x(4)*exp(s);...
   (1+s)*x(2)*x(4)*x(5)*exp(s),(1+s)*x(1)*x(4)*x(5)*exp(s),s^2/(x(3))*exp(s),...
   (1+s)*x(1)*x(2)*x(5)*exp(s),(1+s)*x(1)*x(2)*x(4)*exp(s); ...
   (1+s)*x(2)*x(3)*x(5)*exp(s),(1+s)*x(1)*x(3)*x(5)*exp(s), ...
   (1+s)*x(1)*x(2)*x(5)*exp(s),s^2/(x(4))*exp(s),(1+s)*x(1)*x(2)*x(3)*exp(s);...
   (1+s)*x(2)*x(3)*x(4)*exp(s),(1+s)*x(1)*x(3)*x(4)*exp(s), ...
   (1+s)*x(1)*x(2)*x(4)*exp(s),(1+s)*x(1)*x(2)*x(3)*exp(s),s^2/(x(5))*exp(s)]';
%%%%%%%% 约束函数 h(x) 的Hesse阵%%%%%%%%%%%%
function [d2h1,d2h2,d2h3]=d2h(x)
```

```
d2h1=[2 0 0 0 0; 0 2 0 0 0; 0 0 2 0 0; 0 0 0 2 0; 0 0 0 0 2]';
d2h2=[0 0 0 0 0; 0 0 1 0 0; 0 1 0 0 0; 0 0 0 0 -5; 0 0 0 -5 0]';
d2h3=[6*x(1) 0 0 0 0;0 6*x(2) 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0]';
```

利用上面的程序, 取乘子向量的初值为 $\mu_0 = (0, 0, 0)$, 终止准则值取为 $\|\nabla L(x_k, \mu_k)\|^2 \leq 10^{-12}$, 对于不同的初始点得到计算结果如下表所示.

初始点 (x_0)	迭代次数 (k)	$f(x_k)$ 的值	$\|h(x_k)\|$ 的值
$(-1.71.51.8 - 0.6 - 0.6)^T$	11	0.0539	3.2894e-011
$(-1.7, 1.6, 1.8, -0.7, -0.7)^T$	11	0.0539	5.9272e-012
$(-1.8, 1.7, 1.9, -0.8, -0.8)^T$	9	0.0539	3.0606e-011
$(-2, 1.5, 2, -1, -1)^T$	14	0.0539	1.4766e-011
$(-3, 2, 3, -2, -2)^T$	16	0.0539	3.6545e-011

说明 Matlab 调用方式为: 在命令窗口依次输入如下命令并回车即得计算结果.

```
x0=[-1.7,1.6,1.8,-0.7,-0.7]';
```

```
mu0=[0.1 0.1 0.1]';
[x,mu,val,mh,k]=newtlagr(x0,mu0)
```

§12.2 SQP 方法的算法模型

§12.2.1 基于拉格朗日函数 Hesse 阵的 SQP 方法

前一节介绍的牛顿-拉格朗日法, 由于每一迭代步求解方程组 (12.6) 数值上不是很稳定, 因此这一方法并不实用. 但它有一个重要的贡献, 就是以它为基础发展了序列二次规划方法 (SQP 方法). 鉴于方程组 (12.6) 的求解数值不稳定, 故考虑将它转化为一个严格凸二次规划问题. 转化的条件是 (12.2) 的解点 x^* 处最优性二阶充分条件成立, 即对满足 $A(x^*)^T d = 0$ 的任一向量 $d \neq 0$, 成立

$$d^T W(x^*, \mu^*) d > 0.$$

这时, 由引理 9.3 知, 当 $\tau > 0$ 充分小时, 有

$$W(x^*, \mu^*) + \frac{1}{2\tau} A(x^*)^T A(x^*)$$

正定. 考虑 (12.6) 中的 $W(x_k, \mu_k)$ 用一个正定矩阵来代替, 记

$$B(x_k, \mu_k) = W(x_k, \mu_k) + \frac{1}{2\tau} A(x_k)^T A(x_k),$$

则当 $(x_k, \mu_k) \to (x^*, \mu^*)$ 时, 矩阵 $B(x^*, \mu^*)$ 正定. 注意到 (12.6) 的第一个展开式为

$$W(x_k, \mu_k) d_k - A(x_k)^T \nu_k = -\nabla f(x_k) + A(x_k)^T \mu_k,$$

将上式变形为

$$\left[W(x_k, \mu_k) + \frac{1}{2\tau} A(x_k)^T A(x_k)\right] d_k - A(x_k)^T \left[\mu_k + \nu_k + \frac{1}{2\tau} A(x_k) d_k\right] = -\nabla f(x_k).$$

令

$$\bar{\mu}_k := \mu_k + \nu_k + \frac{1}{2\tau} A(x_k) d_k,$$

即得

$$B(x_k,\mu_k)d_k - A(x_k)^T\bar{\mu}_k = -\nabla f(x_k).$$

因此 (12.6) 等价于

$$\begin{bmatrix} B(x_k,\mu_k) & -A(x_k)^T \\ A(x_k) & 0 \end{bmatrix} \begin{bmatrix} d_k \\ \bar{\mu}_k \end{bmatrix} = - \begin{bmatrix} \nabla f(x_k) \\ h(x_k) \end{bmatrix}. \tag{12.9}$$

进一步, 可以把方程 (12.9) 转化为严格凸二次规划. 我们有下面的定理:

定理 12.1 设 $B(x_k,\mu_k)$ 是 $n\times n$ 正定矩阵, $A(x_k)$ 是 $l\times n$ 行满秩矩阵. 则 d_k 满足 (12.9) 的充要条件是 d_k 是下列严格凸二次规划

$$\begin{cases} \min \quad q_k(d) = \dfrac{1}{2}d^T B(x_k,\mu_k)d + \nabla f(x_k)^T d \\ \text{s.t.} \quad h(x_k) + A(x_k)d = 0 \end{cases} \tag{12.10}$$

的全局极小点.

证 设 d_k 是 (12.10) 的全局极小点, 注意到 $A(x_k)$ 行满秩, 故由 KT 条件, 存在乘子向量 $\bar{\mu}_k$, 使得

$$\nabla f(x_k) + B(x_k, \mu_k)d_k - A(x_k)^T\bar{\mu}_k = 0.$$

再由 (12.10) 的约束条件知, $(d_k, \bar{\mu}_k)$ 是方程组 (12.9) 的解.

反之, 设 $(d_k, \bar{\mu}_k)$ 是方程组 (12.9) 的解. 由于 $B(x_k, \mu_k)$ 正定, $A(x_k)$ 行满秩, 故方程组 (12.9) 的系数矩阵是非奇异的, 从而这个解是唯一的. 由定理 11.3 知, $(d_k, \bar{\mu}_k)$ 是 (12.10) 的 KT 对, 从而 d_k 是 (12.10) 的全局极小点. □

为了方便, 定义罚函数

$$P(x, \mu) = \|\nabla L(x, \mu)\|^2 = \|\nabla f(x) - A(x)^T\mu\|^2 + \|h(x)\|^2. \quad (12.11)$$

不难证明, 由 (12.6) 确定的 p_k 满足 (参见文献 [7])

$$\nabla P(x_k, \mu_k)^T p_k = -2P(x_k, \mu_k) \le 0. \quad (12.12)$$

我们有下面的算法:

算法 12.2 (纯等式约束优化问题的 **SQP** 方法)

步 0 选取 $x_0 \in \mathbb{R}^n$, $\mu_0 \in \mathbb{R}^l$, $\rho, \gamma \in (0,1)$, $0 \leq \varepsilon \ll 1$. 令 $k := 0$.

步 1 计算 $P(x_k, \mu_k)$ 的值. 若 $P(x_k, \mu_k) \leq \varepsilon$, 停算. 否则, 转步 2.

步 2 求解二次规划子问题 (12.10) 得 d_k 和 $\bar{\mu}_k$, 并置

$$\nu_k = \bar{\mu}_k - \mu_k - \frac{1}{2\tau} A(x_k) d_k.$$

步 3 若

$$P(x_k + d_k, \mu_k + \nu_k) \leq (1-\gamma) P(x_k, \mu_k), \tag{12.13}$$

则置 $\alpha_k := 1$, 转步 5; 否则, 转步 4.

步 4 令 m_k 是使下面的不等式成立的最小非负整数 m:

$$P(x_k + \rho^m d_k, \mu_k + \rho^m \nu_k) \leq (1 - \gamma\rho^m) P(x_k, \mu_k), \tag{12.14}$$

置 $\alpha_k = \rho^{m_k}$.

步 5 令 $x_{k+1} = x_k + \alpha_k d_k$, $\mu_{k+1} = \mu_k + \alpha_k \nu_k$. 置 $k := k+1$, 转步 1.

不难发现, 在上面的算法中, 若 $\alpha_k < 1$, 则必有

$$P(x_k + \rho^{m_k-1} d_k, \mu_k + \rho^{m_k-1}\nu_k) > (1 - \gamma\rho^{m_k-1}) P(x_k, \mu_k). \qquad (12.15)$$

下面给出算法 12.2 的全局收敛性定理.

定理 12.2 对于等式约束问题 (12.2), 若 SQP 算法 12.2 生成的迭代序列 $\{(x_k, \mu_k)\}$ 使得 KT 矩阵的逆矩阵 $N(x_k, \mu_k)^{-1}$ 一致有界, 则 $\{(x_k, \mu_k)\}$ 的任何聚点 (x^*, μ^*) 都满足 $P(x^*, \mu^*) = 0$. 特别地, $\{x_k\}$ 的任一聚点都是问题 (12.2) 的 KT 点.

证 用反证法. 不失一般性, 假定 $\{(x_k, \mu_k)\} \to (x^*, \mu^*)$. 若 $P(x^*, \mu^*) >$

0. 由步 3 和步 4 可知

$$P(x_{k+1},\mu_{k+1}) \le (1-\gamma\alpha_k)P(x_k,\mu_k) < P(x_k,\mu_k).$$

由上式及 $P(x_k,\mu_k) \to P(x^*,\mu^*) > 0$ 可推得

$$\lim_{k\to\infty} \alpha_k = 0.$$

另外, 由 (12.6) 可得

$$p_k = \begin{bmatrix} d_k \\ \nu_k \end{bmatrix} = -N(x_k,\mu_k)^{-1}\nabla L(x_k,\mu_k).$$

注意到矩阵 $N(x_k,\mu_k)^{-1}$ 的一致有界性, $p_k = (d_k,\nu_k)$ 也是一致有界的, 且 $p_k \to p^* = (d^*,\nu^*)$, 其中 p^* 满足牛顿方程 $N(x^*,\mu^*)p^* = -\nabla L(x^*,\mu^*)$. 由于 $\alpha_k' = \alpha_k/\rho = \rho^{m_k-1} \to 0$. 故由 (12.15) 有

$$\frac{P(x_k+\alpha_k' d_k, \mu_k+\alpha_k'\nu_k) - P(x_k,\mu_k)}{\alpha_k'} > -\gamma P(x_k,\mu_k).$$

对上式两边取极限得

$$-\gamma P(x^*, \mu^*) \leq \nabla P(x^*, \mu^*)^T p^* = -2P(x^*, \mu^*),$$

即

$$(2-\gamma)P(x^*, \mu^*) \leq 0.$$

注意到 $0 < \gamma < 1$, 故得 $P(x^*, \mu^*) \leq 0$, 这与假设 $P(x^*, \mu^*) > 0$ 矛盾. 因此必有 $P(x^*, \mu^*) = 0$.

现在来证明定理的第二部分. 设 x^* 是序列 $\{x_k\}$ 的任一聚点, 不失一般性, 假设 $\{x_k\} \to x^*$. 注意到序列 $\{P(x_k, \mu_k)\}$ 是单调不增的且矩阵序列 $\{N(x_k, \mu_k)^{-1}\}$ 是一致有界的, 由此可推得 $\alpha_k \to 0\,(k \to \infty)$, 且 $\{p_k = (d_k, \nu_k)\}$ 是有界的, 故 $\{d_k\}$ 和 $\{\nu_k\}$ 都是有界的. 又由步 2, 严格凸二次规划子问题的解 $\{(d_k, \bar{\mu}_k)\}$ 也是有界的, 故 $\{\bar{\mu}_k\}$ 是有界的, 即 $\{\bar{\mu}_k = \mu_k + \nu_k\}$ 是有界的. 特别地, 对于充分大的 k, $\{\mu_{k+1} := \mu_k + \alpha_k \nu_k\}$

是有界的, 于是存在收敛的子序列. 假设 μ^* 是 $\{\mu_{k+1}\}$ 的某个聚点, 则 (x^*, μ^*) 是迭代序列 $\{(x_k, \mu_k)\}$ 的一个聚点. 由第一部分的证明可知 $P(x^*, \mu^*) = 0$, 因此, x^* 是问题 (12.2) 的 KT 点. □

关于算法 12.2 的收敛速度, 我们有下面的定理, 其证明过程可参见文献 [5], 兹略去不证.

定理 12.3 设算法 12.2 产生的迭代序列 $\{x_k\}$ 收敛到一个局部极小点 x^*. 若函数 $f, h = (h_1, \cdots, h_l)^T$ 在 x^* 附近三阶连续可微, Jacobi 矩阵 $A(x^*) = \nabla h(x^*)^T$ 行满秩, 且二阶最优性充分条件成立. 则有

(1) 必有 $\{\mu_k\} \to \mu^*$, 其中 μ^* 是等式约束优化问题 (12.6) 的拉格朗日乘子, 且整个迭代序列 $\{(x_k, \mu_k)\}$ 是二阶收敛的, 即

$$\|(x_{k+1} - x^*, \mu_{k+1} - \mu^*)\| = O(\|(x_k - x^*, \mu_k - \mu^*)\|^2).$$

(2) 序列 $\{x_k\}$ 超线性收敛到 x^*, 且

$$\|x_{k+1}-x^*\| = o\Big(\|x_k-x^*\|\prod_{i=1}^{t}\|x_{k-i}-x^*\|\Big),$$

其中 t 是任意给定的正整数.

例 12.2 编制 Matlab 程序, 用算法 12.2 计算下列最优化问题的极小点

$$\begin{cases} \min & f(x)=\mathrm{e}^{x_1x_2x_3x_4x_5}-\dfrac{1}{2}(x_1^3+x_2^3+1)^2, \\ \text{s.t.} & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2-10=0, \\ & x_2x_3-5x_4x_5=0, \\ & x_1^3+x_2^3+1=0. \end{cases}$$

该问题有最优解 $x^*=(-1.71,1.59,1.82,-0.763,-0.763)^T$, 最优值 $f(x^*)=0.0539$.

解 编制 Matlab 程序如下

```
function [x,mu,val,k]=lagsqp(x0,mu0)
%功能: 用基于拉格朗日函数Hesse阵的SQP方法求解约束优化问题:
%       min f(x)      s.t. h_i(x)=0, i=1,..., l.
%输入:  x0是初始点, mu0是乘子向量的初始值
%输出:  x, mu 分别是近似最优点及相应的乘子,
%  val是最优值, mh是约束函数的模, k是迭代次数.
maxk=50;   %最大迭代次数
n=length(x0); l=length(mu0);
rho=0.5; gamma=0.2;
x=x0; mu=mu0;  tau=0.1;
k=0;  epsilon=1e-8;
while(k<maxk)
    P1=P(x,mu)  %计算罚函数的值
    if(P1<epsilon), break; end  %检验终止准则
    H=B(x,mu,tau);     % 计算KT矩阵
    c=df1(x);     %计算目标函数梯度
    be=-h1(x); %计算约束函数
    Ae=dh1(x); %计算约束函数的Jacobi矩阵
    [dx,lam]=qsubp(H,c,Ae,be);
    du=lam-mu-1.0/(2*tau)*dh1(x)*dx;
```

```
    m=0; mk=0;
    while(m<20)   % Armijo搜索
        if(P(x+rho^m*dx,mu+rho^m*du)<=(1-gamma*rho^m)*P1)
            mk=m; break;
        end
        m=m+1;
    end
    x=x+rho^mk*dx; mu=mu+rho^mk*du;
    k=k+1;
end
val=f1(x);
%%%%%%% 求解子问题 %%%%%%%%%%%%%
function [x,mu1]=qsubp(H,c,Ae,be)
ginvH=pinv(H);
[m,n]=size(Ae);
if(m>0)
    rb=Ae*ginvH*c + be;
    mu1=pinv(Ae*ginvH*Ae')*rb;
    x=ginvH*(Ae'*mu1-c);
else
```

```
    x=-ginvH*c;
    mu1=zeros(m,1);
end
%%%%%%%%% 拉格朗日函数 L(x,mu) %%%%%%%%%%%%
function l=la(x,mu)
f=f1(x);   %调用目标函数文件
h=h1(x);  %调用约束函数文件
l=f-mu'*h; % 计算乘子函数
%%%%%%%%% 拉格朗日函数的梯度 %%%%%%%%%%%%
function dl=dla(x,mu)
df=df1(x); %调用目标函数梯度文件
h=h1(x);  %调用约束函数文件
dh=dh1(x);  %调用约束函数Jacobi矩阵文件
dl=[df-dh'*mu; -h];  %计算乘子函数梯度文件
%%%%%%%%% 罚函数P(x,mu) %%%%%%%%%%%%%%%
function s=P(x,mu)
dl=dla(x,mu);
s=norm(dl)^2;
%%%%%%%%% 拉格朗日函数的Hesse阵 %%%%%%%%%%
function d2l=d2la(x,mu)
```

```
d2f=d2f1(x);   %调用目标函数Hesse阵文件
[d2h1,d2h2,d2h3]=d2h(x); %调用约束函数二阶导数文件
d2l=d2f-mu(1)*d2h1-mu(2)*d2h2-mu(3)*d2h3;
%%%%%%%%% KT矩阵B(x,mu) %%%%%%%%%%
function H=B(x,mu,tau)  %计算KT矩阵
d2l=d2la(x,mu);   %计算Hesse阵
dh=dh1(x);  %约束函数的Jacobi矩阵
H=d2l+1.0/(2*tau)*dh'*dh;
%%%%%%%%% 目标函数 f(x) %%%%%%%%%%%%
function f=f1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
f=exp(s)-0.5*(x(1)^3+x(2)^3+1)^2;
%%%%%%%%% 约束函数 h(x) %%%%%%%%%%%%
function h=h1(x)
h=[x(1)^2+x(2)^2+x(3)^2+x(4)^2+x(5)^2-10; ...
   x(2)*x(3)-5*x(4)*x(5);x(1)^3+x(2)^3+1];
%%%%%%%%% 目标函数 f(x) 的梯度%%%%%%%%%%%%
function df=df1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
df(1)=s/(x(1))*exp(s)-3*(x(1)^3+x(2)^3+1)*x(1)^2;
```

```
df(2)=s/(x(2))*exp(s)-3*(x(1)^3+x(2)^3+1)*x(2)^2;
df(3)=s/(x(3))*exp(s);
df(4)=s/(x(4))*exp(s);
df(5)=s/(x(5))*exp(s);
df=df(:);
%%%%%%%% 约束函数 h(x) 的Jacobi矩阵A(x)%%%%%%%%%%%%
function dh=dh1(x)
dh=[2*x(1),2*x(2),2*x(3),2*x(4),2*x(5);0,x(3),x(2),-5*x(5),-5*x(4); ...
    3*x(1)^2,3*x(2)^2,0,0,0];
%%%%%%%% 目标函数 f(x) 的Hesse阵%%%%%%%%%%%%
function d2f=d2f1(x)
s=x(1)*x(2)*x(3)*x(4)*x(5);
d2f=[(s/(x(1)))^2*exp(s)-6*x(1)*(x(1)^3+x(2)^3+1)-9*x(1)^4,...
    (1+s)*x(3)*x(4)*x(5)*exp(s)-9*x(1)^2*x(2)^2, ...
    (1+s)*x(2)*x(4)*x(5)*exp(s),...
    (1+s)*x(2)*x(3)*x(5)*exp(s),(1+s)*x(2)*x(3)*x(4)*exp(s); ...
    (1+s)*x(3)*x(4)*x(5)*exp(s)-9*x(1)^2*x(2)^2,  ...
    (s/(x(2)))^2*exp(s)-6*x(2)*(x(1)^3+x(2)^3+1)-9*x(2)^4, ...
    (1+s)*x(1)*x(4)*x(5)*exp(s),(1+s)*x(1)*x(3)*x(5)*exp(s), ...
    (1+s)*x(1)*x(3)*x(4)*exp(s); ...
```

```
    (1+s)*x(2)*x(4)*x(5)*exp(s),(1+s)*x(1)*x(4)*x(5)*exp(s), ...
     s^2/(x(3))*exp(s),(1+s)*x(1)*x(2)*x(5)*exp(s), ...
    (1+s)*x(1)*x(2)*x(4)*exp(s); ...
    (1+s)*x(2)*x(3)*x(5)*exp(s),(1+s)*x(1)*x(3)*x(5)*exp(s), ...
    (1+s)*x(1)*x(2)*x(5)*exp(s),s^2/(x(4))*exp(s), ...
    (1+s)*x(1)*x(2)*x(3)*exp(s);...
    (1+s)*x(2)*x(3)*x(4)*exp(s),(1+s)*x(1)*x(3)*x(4)*exp(s), ...
    (1+s)*x(1)*x(2)*x(4)*exp(s),(1+s)*x(1)*x(2)*x(3)*exp(s), ...
     s^2/(x(5))*exp(s)]';
%%%%%%%%%%%%%%%约束函数h(x)的Hesse阵%%%%%%%%%%%%%%%%%%%%
function [d2h1,d2h2,d2h3]=d2h(x)
d2h1=[2 0 0 0 0; 0 2 0 0 0; 0 0 2 0 0; 0 0 0 2 0; 0 0 0 0 2]';
d2h2=[0 0 0 0 0; 0 0 1 0 0; 0 1 0 0 0; 0 0 0 0 -5; 0 0 0 -5 0]';
d2h3=[6*x(1) 0 0 0 0;0 6*x(2) 0 0 0;0 0 0 0 0;0 0 0 0 0;0 0 0 0 0]';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

利用上面的程序, 取乘子向量的初值为 $\mu_0 = (0, 0, 0)$, 终止准则值取为 $P(x_k, \mu_k) \leq 10^{-12}$, 对于不同的初始点得到计算结果如下表所示.

初始点 (x_0)	迭代次数 (k)	$f(x_k)$ 的值	$P(x_k,\mu_k)$ 的值
$(-1.71.51.8-0.6-0.6)^T$	11	0.0539	4.7981e-013
$(-1.7,1.6,1.8,-0.7,-0.7)^T$	11	0.0539	8.6424e-014
$(-1.8,1.7,1.9,-0.8,-0.8)^T$	9	0.0539	4.4646e-013
$(-2,1.5,2,-1,-1)^T$	14	0.0539	2.1538e-013
$(-3,2,3,-2,-2)^T$	16	0.0539	5.3309e-013

§12.2.2 基于修正 Hesse 阵的 SQP 方法

首先, 考虑将上一节中关于构造二次规划子问题求解等式约束优化问题的思想推广到一般形式的约束优化问题 (12.1). 在给定点 (x_k,μ_k,λ_k) 之后, 将约束函数线性化, 并且对拉格朗日函数进行二次多项式近似, 得到下列形式的二次规划子问题:

$$\begin{aligned} \min \quad & \frac{1}{2}d^TW_kd+\nabla f(x_k)^Td \\ \text{s.t} \quad & h_i(x_k)+\nabla h_i(x_k)^Td=0,\ i\in E, \\ & g_i(x_k)+\nabla g_i(x_k)^Td\geq 0,\ i\in I, \end{aligned} \tag{12.16}$$

Back
Close

其中 $E=\{1,\cdots,l\}$, $I=\{1,\cdots,m\}$, $W_k=\nabla^2_{xx}L(x_k,\mu_k,\lambda_k)$, 而拉格朗日函数为

$$L(x,\mu,\lambda)=f(x)-\sum_{i\in E}\mu_i h_i(x)-\sum_{i\in I}\lambda_i g_i(x).$$

于是, 迭代点 x_k 的校正步 d_k 以及新的拉格朗日乘子估计量 μ_{k+1}, λ_{k+1} 可以分别定义为问题 (12.16) 的最优解 d^* 和相应的拉格朗日乘子 μ^*, λ^*.

上述的二次规划子问题 (12.16) 可能不存在可行点, 为了克服这

一困难, Powell 引进了一个辅助变量 ξ, 然后求解下面的线性规划

$$\begin{aligned} \min \quad & -\xi \\ \text{s.t.} \quad & -\xi h_i(x_k) + \nabla h_i(x_k)^T d = 0, \ i \in E, \\ & -\xi g_i(x_k) + \nabla g_i(x_k)^T d \geq 0, \ i \in U_k, \\ & g_i(x_k) + \nabla g_i(x_k)^T d \geq 0, \ i \in V_k, \\ & -1 \leq \xi \leq 0, \end{aligned} \tag{12.17}$$

其中 $U_k = \{i | g_i(x_k) < 0, \ i \in I\}$, $V_k = \{i | g_i(x_k) \geq 0, \ i \in I\}$. 显然 $\xi = 0, d = 0$ 上述线性规划的一个可行点, 并且该线性规划的极小点 $\bar{\xi} = -1$ 当且仅当二次规划子问题 (12.16) 是相容的, 即子问题的可行域非空.

当 $\bar{\xi} = -1$ 时, 可以用线性规划问题的最优解 $\bar{d}$ 作为初始点, 求出二次规划子问题的最优解 d_k. 而当 $\bar{\xi} = 0$ 或接近于 0 时, 二次规划子问题无可行点, 此时需要重新选择迭代初始点 x_k, 然后启动 SQP 算

法. 当 $\bar{\xi} \neq -1$ 但比较接近 -1 时, 可以用对应 $\bar{\xi}$ 的约束条件代替原来的约束条件, 再求解修正后的二次规划子问题.

不失一般性, 以后为了讨论的方便, 我们假设二次规划子问题是相容的. 下面的定理描述了迭代点对应的有效约束指标集与最优有效约束指标集之间的关系, 其证明可参见文献[15].

定理 12.4 给定约束优化问题 (12.1) 的一个 KT 点 x^* 和相应的拉格朗日乘子向量 $\mu^*, \lambda^* \geq 0$. 假定在 x^* 处, 下面的条件成立:

(1) 有效约束的 Jacobi 矩阵 $J_{S(x^*)}$ 行满秩, 此处 $S(x^*) = E \cup I(x^*)$.

(2) 严格互补松弛条件成立, 即 $g_i(x^*) \geq 0$, $\lambda_i^* \geq 0$, $\lambda_i^* g_i(x^*) = 0$, $\lambda_i^* + g_i(x^*) > 0$.

(3) 二阶最优性充分条件成立, 即对任意满足 $A(x^*)d = 0$ 的向

量 $d \neq 0$, 成立

$$d^T W(x^*, \mu^*, \mu^*) d > 0.$$

那么, 若 (x_k, μ_k, λ_k) 充分靠近 (x^*, μ^*, λ^*), 则二次规划子问题 (12.16) 存在一个局部极小点 d^*, 使得其对应的有效约束指标集 $S(d^*)$ 与原问题在 x^* 处的有效约束指标集 $S(x^*)$ 是相同的.

注意到在构造二次规划子问题 (12.16) 时, 需要计算拉格朗日函数在迭代点 x_k 处的 Hesse 阵 $W_k = W(x_k, \mu_k, \lambda_k)$, 其计算量是巨大的. 为了克服这一缺陷, 1976 年华裔数学家韩世平 (S.P. Han) 基于牛顿-拉格朗日方法提出了一种利用对称正定矩阵 B_k 替代拉格朗日矩阵 W_k 的序列二次规划法. 另外由于 R.B. Wilson 在 1963 年较早地考虑了牛顿-拉格朗日方法, 加之 M.J.D. Powell 于 1977 年修正了 Han 的方法, 所以人们也将这种序列二次规划法称之为 WHP 方法 (Wilson-Han-Powell 方法).

对于一般约束的优化问题 (12.1), 在迭代点 (x_k, μ_k, λ_k) 处, WHP 方法需要构造一个下列形式的二次规划子问题:

$$\begin{aligned} \min \quad & \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d \\ \text{s.t.} \quad & h_i(x_k) + \nabla h_i(x_k)^T d = 0, \ i \in E, \\ & g_i(x_k) + \nabla g_i(x_k)^T d \geq 0, \ i \in I, \end{aligned} \tag{12.18}$$

并且用该二次规划子问题的解 d_k 作为原问题的变量 x 在第 k 次迭代过程中的搜索方向. 顺便提一下, 这个搜索方向 d_k 具有一个比较好的性质: 即它是许多罚函数 (价值函数) 的下降方向, 比如 ℓ_1 罚函数 (价值函数)

$$P_\sigma(x) = f(x) \ + \ \frac{1}{\sigma} \left[\sum_{i \in E} |h_i(x)| + \sum_{i \in I} \left| [g_i(x)]_- \right| \right], \tag{12.19}$$

其中罚参数 $\sigma > 0$, $[g_i(x)]_- = \max\{0, -g_i(x)\}$.

现在我们写出 WHP 方法的算法步骤如下.

算法 12.3 (WHP 方法)

步 0 给定初始点 $x_0 \in \mathbb{R}^n$, 初始对称矩阵 $B_0 \in \mathbb{R}^{n\times n}$, 容许误差限 $0 \leq \varepsilon \ll 1$ 和满足 $\sum_{k=0}^{\infty} \eta_k < +\infty$ 的非负数列 $\{\eta_k\}$. 取参数 $\sigma > 0$ 和 $\delta > 0$. 置 $k := 0$.

步 1 求解子问题 (12.18), 得最优解 d_k.

步 2 若 $\|d_k\| \leq \varepsilon$, 停算, 输出 x_k 作为近似极小点.

步 3 利用 ℓ_1 罚函数 $P_\sigma(x)$, 按照某种线搜索规则确定步长 $\alpha_k \in (0, \delta]$, 使得

$$P_\sigma(x_k + \alpha_k d_k) \leq \min_{\alpha\in(0,\delta]} P_\sigma(x_k + \alpha d_k) + \eta_k.$$

步 4 置 $x_{k+1} := x_k + \alpha_k d_k$, 更新 B_k 为 B_{k+1}.

步 5 令 $k := k + 1$, 转步 1.

在一定的条件下, 可以证明算法 12.3 的全局收敛性. 下面, 我们

不加证明地给出算法12.3 的全局收敛性定理, 其证明过程可参见文献[1].

定理 12.5 对于约束优化问题 (12.1), 假设 $f, h_i\,(i \in E)$ 和 $g_i\,(i \in I)$ 都是连续可微的, 且存在常数 $0 < m \leq M$, 使得算法12.3 中的对称正定矩阵 B_k 满足

$$m\|d\|^2 \leq d^T B_k d \leq M\|d\|^2, \quad (\forall\, d \in \mathbb{R}^n,\ k = 1, 2, \cdots).$$

若罚参数 $\sigma > 0$ 和二次规划子问题 (12.18) 的拉格朗日乘子向量 μ_{k+1}, $\lambda_{k+1} \geq 0$ 满足

$$\sigma \max\{\|\lambda_{k+1}\|_\infty, \|\mu_{k+1}\|_\infty\} \leq 1, \quad (\forall\, k = 1, 2, \cdots).$$

则算法12.3 生成的序列 $\{x_k\}$ 的任何聚点都是问题 (12.1) 的 KT 点.

§12.3 SQP 方法的相关问题

§12.3.1 二次规划子问题的 Hesse 阵

下面以纯等式的约束优化问题为例, 说明如何选择二次规划子问题的 Hesse 阵, 其中二次规划子问题为

$$\begin{aligned} \min \quad & \frac{1}{2} d^T W_k d + \nabla f(x_k)^T d \\ \text{s.t} \quad & h_i(x_k) + \nabla h_i(x_k)^T d = 0, \ i = \{1, \cdots, l\}. \end{aligned} \tag{12.20}$$

.

1. 基于拟牛顿校正公式的选择方法

由于拟牛顿法是处理无约束优化问题的有效算法, 故可利用拟牛顿法的基本原理, 对拉格朗日函数 $L(x_k, \mu_k)$ 的 Hesse 阵 $\nabla^2_{xx} L(x_k, \mu_k)$ 的近似矩阵 B_k 进行修正. 令

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla_x L(x_{k+1}, \mu_{k+1}) - \nabla_x L(x_k, \mu_{k+1}). \tag{12.21}$$

因 BFGS 校正公式要求向量 s_k 和 y_k 满足曲率条件, 即 $s_k^T y_k > 0$, 但由 (12.21) 确定的向量 s_k, y_k 可能不满足这一条件. 为此, 有必要对向量 y_k 进行修正. 下面的修正策略是由 Powell 于 1978 年给出的: 令

$$z_k = \theta_k y_k + (1 - \theta_k) B_k s_k, \tag{12.22}$$

其中参数 θ_k 定义为

$$\theta_k = \begin{cases} 1, & \text{若 } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ \dfrac{0.8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{若 } s_k^T y_k < 0.2 s_k^T B_k s_k. \end{cases} \tag{12.23}$$

于是, 矩阵 B_k 的约束 BFGS 校正公式为

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{z_k z_k^T}{s_k^T z_k}. \tag{12.24}$$

注意到参数 θ_k 的定义 (12.23), 不难验证: 当 $\theta_k = 1$时,

$$s_k^T z_k = s_k^T y_k \geq 0.2 s_k^T B_k s_k > 0,$$

而当 $\theta_k \neq 1$ 时,

$$s_k^T z_k = \theta_k s_k^T y_k + (1-\theta_k) s_k^T B_k s_k = 0.2 s_k^T B_k s_k > 0.$$

因此, 约束 BFGS 校正公式 (12.24) 可以保持正定性.

2. 基于增广拉格朗日函数的选择方法

下面考虑增广拉格朗日函数

$$L_A(x,\mu,\sigma) = f(x) - \mu^T h(x) + \frac{1}{2\sigma}\|h(x)\|^2, \tag{12.25}$$

其中罚参数 $\sigma > 0$, $h(x) = (h_1(x), \cdots, h_l(x))^T$. 在 KT 点 (x^*, μ^*) 处, 根据 $h(x^*) = 0$, 可知增广拉格朗日函数的 Hesse 阵为

$$\nabla_{xx}^2 L_A(x^*,\mu^*,\sigma) = \nabla_{xx}^2 L(x^*,\mu^*) + \frac{1}{\sigma} A(x^*)^T A(x^*). \tag{12.26}$$

对于纯等式约束的优化问题, 若约束函数在 x^* 处的 Jacobi 矩阵 $A(x^*)$ 行满秩, 并且二阶最优性充分条件成立, 则存在某个阈值 $\bar{\sigma} > 0$, 使得

$\forall\,\sigma\in(0,\bar{\sigma}]$, $\nabla^2_{xx}L_A(x_k,\mu_k,\sigma)$ 是正定的. 于是二次规划子问题中的 W_k 可取为 $\nabla^2_{xx}L_A(x_k,\mu_k,\sigma)$, 或者取对 $\nabla^2_{xx}L_A(x_k,\mu_k,\sigma)$ 进行拟牛顿近似的校正矩阵 B_k.

§12.3.2 价值函数与搜索方向的下降性

为了保证序列二次规划法 (SQP 方法) 的全局收敛性, 通常借助于某价值函数来确定搜索步长. 例如, 目标函数、罚函数以及增广拉格朗日函数等都可以作为价值函数, 用来衡量一维搜索的好坏. 下面介绍两类著名的价值函数.

1. ℓ_1 价值函数

首先, 以纯等式约束的优化问题为例, 考虑下列 ℓ_1 价值函数

$$\phi_1(x)=f(x)+\frac{1}{\sigma}\|h(x)\|_1, \tag{12.27}$$

Back
Close

其中 $\sigma > 0$ 为罚参数.

命题 12.1 设 d_k 和 μ_{k+1} 分别是子问题 (12.20) 的最优解和拉格朗日乘子, 则 ϕ_1 沿方向 d_k 的方向导数满足

$$\phi_1'(x_k, \sigma; d_k) \leq -d_k^T W_k d_k - (\sigma^{-1} - \|\mu_{k+1}\|_\infty)\|h(x_k)\|_1. \qquad (12.28)$$

证 由泰勒公式可得

$$\begin{aligned} f(x_k + \alpha d_k) &= f(x_k) + \alpha \nabla f(x_k)^T d_k + \frac{1}{2}\alpha^2 d_k^T \nabla^2 f(\xi_k) d_k, \\ h(x_k + \alpha d_k) &= h(x_k) + \alpha A(x_k) d_k + \frac{1}{2}\alpha^2 d_k^T \nabla^2 h(\eta_k) d_k \\ &= (1-\alpha) h(x_k) + \frac{1}{2}\alpha^2 d_k^T \nabla^2 h(\eta_k) d_k, \end{aligned}$$

其中

$$d_k^T \nabla^2 h(\eta_k) d_k = \left[d_k^T \nabla^2 h_1(\eta_k) d_k,\ d_k^T \nabla^2 h_2(\eta_k) d_k,\ \cdots,\ d_k^T \nabla^2 h_l(\eta_k) d_k\right]^T.$$

故对比较小的 $\alpha > 0$, 有

$$\phi_1(x_k + \alpha d_k) - \phi_1(x_k) \leq \alpha(\nabla f(x_k)^T d_k - \sigma^{-1}\|h(x_k)\|_1) + \alpha^2\vartheta\|d_k\|^2,$$

其中 $\vartheta > 0$ 是某个常数. 同理可证

$$\phi_1(x_k + \alpha d_k) - \phi_1(x_k) \geq \alpha(\nabla f(x_k)^T d_k - \sigma^{-1}\|h(x_k)\|_1) - \alpha^2\vartheta\|d_k\|^2.$$

根据多元函数方向导数的定义, 可得

$$\phi_1'(x_k, \sigma; d_k) = \nabla f(x_k)^T d_k - \sigma^{-1}\|h(x_k)\|_1. \qquad (12.29)$$

由 KT 条件, 有

$$\nabla f(x_k) + W_k d_k - A(x_k)^T \mu_{k+1} = 0,$$

于是有

$$\begin{aligned} \nabla f(x_k)^T d_k &= -d_k^T W_k d_k + d_k^T A(x_k)^T \mu_{k+1} \\ &= -d_k^T W_k d_k - h(x_k)^T \mu_{k+1}. \end{aligned}$$

注意到由 Schwartz 不等式, 有

$$-h(x_k)^T\mu_{k+1} \le \|h(x_k)\|_\infty\|\mu_{k+1}\|_\infty \le \|\mu_{k+1}\|_\infty\|h(x_k)\|_1.$$

至此, 命题的结论已得到证明. □

根据命题 12.1 可知, 若矩阵 W_k 正定, 且取罚参数

$$\sigma^{-1} \ge \|\mu_{k+1}\|_\infty + \bar{\delta},$$

其中 $\bar{\delta} > 0$, 则可保证 d_k 是 $\phi_1(x,\sigma)$ 在相应点 x_k 处的下降方向. 此外, 在构造二次规划子问题 (12.20) 时, 如果用对称正定矩阵 B_k 替代 W_k, 那么相应的最优解 d_k 也可以是价值函数 ϕ_1 的下降方向.

对于一般的约束优化问题 (12.1), 可以考虑相应的二次规划子问

题 (12.18), 并且将 (12.19) 定义的罚函数作为价值函数:

$$\begin{aligned} P_\sigma(x) &= f(x) + \sigma^{-1}\left[\|h(x)\|_1 + \|g(x)_-\|_1\right] \\ &= f(x) + \frac{1}{\sigma}\left[\sum_{i\in E}|h_i(x)| + \sum_{i\in I}\left|[g_i(x)]_-\right|\right] \end{aligned}$$

其中 $h(x) = (h_1(x), \cdots, h_l(x))^T$, $g(x) = (g_1(x), \cdots, g_m(x))^T$, $g(x)_- = \max\{0, -g_i(x)\}$.

命题 12.2 设 d_k 和 μ_{k+1}, $\lambda_{k+1} \geq 0$ 分别是子问题 (12.18) 的最优解和拉格朗日乘子, 则 P_σ 沿 d_k 的方向导数满足

$$\begin{aligned} P'_\sigma(x_k; d_k) \leq\ & -d_k^T B_k d_k - (\sigma^{-1} - \|\mu_{k+1}\|_\infty)\|h(x_k)\|_1 \\ & -(\sigma^{-1} - \|\lambda_{k+1}\|_\infty)\|g(x_k)_-\|_1. \end{aligned} \tag{12.30}$$

证 注意到 1-范数 $\|\cdot\|_1$ 是凸函数以及函数 $(\cdot)_-$ 具有下列性质

$$\forall\, t_1 \geq t_2, \quad \text{有} \quad (t_1)_- \leq (t_2)_-.$$

利用与命题 12.1 类似的方法可得

$$P'_\sigma(x_k; d_k) = \nabla f(x_k)^T d_k - \sigma^{-1}\big[\|h(x_k)\|_1 + \|g(x_k)_-\|_1\big]. \tag{12.31}$$

再利用二次规划子问题 (12.18) 的 KT 条件, 有

$$\begin{cases} \nabla f(x_k) + B_k d_k = \nabla h(x_k)^T \mu_{k+1} + \nabla g(x_k)^T \lambda_{k+1}, \\ \lambda_{k+1} \geq 0, \quad g(x_k) + \nabla g(x_k)^T d_k \geq 0, \\ \lambda_{k+1}^T [g(x_k) + \nabla g(x_k)^T d_k] = 0. \end{cases} \tag{12.32}$$

又因 d_k 满足可行性条件: $h(x_k) + \nabla h(x_k)^T d_k = 0$, 由此及 (12.32) 的第一、三两式可得

$$\nabla f(x_k)^T d_k = -d_k^T B_k d_k - h(x_k)^T \mu_{k+1} - g(x_k)^T \lambda_{k+1}. \tag{12.33}$$

由 Schwartz 不等式可知

$$\begin{cases} -h(x_k)^T\mu_{k+1} = \|h(x_k)\|_\infty\|\mu_{k+1}\|_\infty \le \|\mu_{k+1}\|_\infty\|h(x_k)\|_1, \\ -g(x_k)^T\lambda_{k+1} \le \lambda_{k+1}^T[g(x_k)]_- \le \|\lambda_{k+1}\|_\infty\|g(x_k)\|_1. \end{cases} \tag{12.34}$$

最后, 由 (12.31), (12.33) 和 (12.34) 即得命题的结论. □

2. Fletcher 价值函数

Fletcher 价值函数也叫做增广拉格朗日价值函数. SQP 方法问世后, 除了使用 ℓ_1 价值函数以得到算法的全局收敛性外, 优化工作者还引进过其他的价值函数. 例如, Fletcher 曾经针对纯等式约束优化问题引入了下列增广拉格朗日价值函数:

$$\phi_F(x,\sigma) = f(x) - \mu(x)^T h(x) + \frac{1}{2\sigma}\|h(x)\|^2, \tag{12.35}$$

其中 $\mu(x)$ 是乘子向量, $\sigma > 0$ 是罚参数. 若函数 $h(x)$ 的 Jacobi 矩阵

$A(x) = \nabla h(x)^T$ 是行满秩的, 则乘子向量可取为

$$\mu(x) = [A(x)A(x)^T]^{-1}A(x)\nabla f(x), \tag{12.36}$$

即 $\mu(x)$ 是下面的最小二乘问题的解:

$$\min_{\mu\in\mathbb{R}^l} \|\nabla f(x) - A(x)^T\mu\|.$$

另一方面, Fletcher 函数在 x_k 处的梯度为

$$\nabla\phi_F(x_k,\sigma) = \nabla f(x_k) - A_k^T\mu(x_k) - \nabla\mu(x_k)h(x_k) + \sigma^{-1}A_k^T h(x_k), \tag{12.37}$$

其中 $A_k = A(x_k)$ 是 $h(x)$ 在 x_k 处的 Jacobi 矩阵, 且 A_k 行满秩. 假设 d_k 是子问题

$$\begin{aligned} \min\ & \frac{1}{2}d^T B_k d + \nabla f(x_k)^T d \\ \text{s.t.}\ & h(x_k) + A_k d = 0 \end{aligned} \tag{12.38}$$

的最优解, 并记

$$d_k = A_k^T d_k^y + Z_k d_k^z,$$

其中 Z_k 的列向量是零空间 $\mathcal{N}(A_k)$ 的一组基. 根据子问题 (12.38) 的可行性条件, 我们有

$$d_k^y = -(A_k A_k^T)^{-1} h(x_k).$$

然后, 利用最小二乘乘子 $\mu(x_k)$ 的定义 (12.36) 可得

$$\nabla f(x_k)^T A_k^T d_k^y = -\mu(x_k)^T h(x_k).$$

故有

$$\nabla \phi_F(x_k, \sigma)^T d_k = \nabla f(x_k)^T Z_k d_k^z - h(x_k)^T \nabla \mu(x_k)^T d_k - \sigma^{-1} \|h(x_k)\|^2. \tag{12.39}$$

另由 KT 条件有

$$W_k d_k + \nabla f(x_k) - A_k \mu_k = 0,$$

其中 μ_k 是拉格朗日乘子. 那么, 我们有

$$\begin{aligned} Z_k^T W_k d_k &= Z_k^T W_k Z_k d_k^z + Z_k^T W_k A_k^T d_k^y \\ &= Z_k^T [A_k \mu_k - \nabla f(x_k)] \\ &= -Z_k^T \nabla f(x_k). \end{aligned}$$

于是, 有

$$\begin{aligned} \nabla \phi_F(x_k, \sigma)^T d_k &= -(d_k^z)^T [Z_k^T W_k Z_k] d_k^z - (d_k^y)^T A_k W_k Z_k d_k^z \\ &\quad - h(x_k)^T \nabla \mu(x_k)^T d_k - \sigma^{-1} \|h(x_k)\|^2. \end{aligned} \tag{12.40}$$

因此, 当简约 Hesse 阵 $Z_k^T W_k Z_k$ 正定时, 若取罚参数满足

$$\sigma^{-1} \geq \left[\frac{-(d_k^y)^T A_k W_k Z_k d_k^z - h(x_k)^T \nabla \mu(x_k)^T d_k}{\|h(x_k)\|^2} \right] + \bar{\delta}, \tag{12.41}$$

其中 $\bar{\delta} > 0$, 则 d_k 是 Fletcher 价值函数的下降方向.

下面我们给出一般形式约束优化问题的 SQP 方法的算法步骤.

算法 12.4 (一般约束优化问题的 SQP 方法)

步 0　给定初始点 $(x_0, \mu_0, \lambda_0) \in \mathbb{R}^n \times \mathbb{R}^l \times \mathbb{R}^m$, 对称正定矩阵 $B_0 \in \mathbb{R}^{n\times n}$. 计算

$$A_0^E = \nabla h(x_0)^T, \quad A_0^I = \nabla g(x_0)^T, \quad A_0 = \begin{bmatrix} A_0^E \\ A_0^I \end{bmatrix}.$$

选择参数 $\eta \in (0, 1/2)$, $\rho \in (0, 1)$, 容许误差 $0 \leq \varepsilon_1, \varepsilon_2 \ll 1$. 令 $k := 0$.

步 1　求解子问题

$$\begin{aligned} \min \quad & \frac{1}{2} d^T B_k d + \nabla f(x_k)^T d \\ \text{s.t.} \quad & h(x_k) + A_k^E d = 0, \\ & g(x_k) + A_k^I d \geq 0, \end{aligned} \tag{12.42}$$

得最优解 d_k.

步 2　若 $\|d_k\|_1 \leq \varepsilon_1$ 且 $\|h_k\|_1 + \|(g_k)_-\|_1 \leq \varepsilon_2$, 停算, 得到原问题的一个近似 KT 点 (x_k, μ_k, λ_k).

步 3 对于某种价值函数 $\phi(x,\sigma)$, 选择罚参数 σ_k, 使得 d_k 是该函数在 x_k 处的下降方向.

步 4 Armijo 搜索. 令 m_k 是使下列不等式成立的最小非负整数 m :

$$\phi(x_k+\rho^m d_k,\sigma_k)-\phi(x_k,\sigma_k)\le\eta\rho^m\phi'(x_k,\sigma;d_k), \tag{12.43}$$

令 $\alpha_k:=\rho^{m_k}$, $x_{k+1}:=x_k+\alpha_k d_k$.

步 5 计算

$$A^E_{k+1}=\nabla h(x_{k+1})^T,\quad A^I_{k+1}=\nabla g(x_{k+1})^T,\quad A_{k+1}=\begin{bmatrix}A^E_{k+1}\\ A^I_{k+1}\end{bmatrix}.$$

以及最小二乘乘子

$$\begin{bmatrix}\mu_{k+1}\\ \lambda_{k+1}\end{bmatrix}=\left[A_{k+1}A^T_{k+1}\right]^{-1}A_{k+1}\nabla f_{k+1}. \tag{12.44}$$

步 6 校正矩阵 B_k 为 B_{k+1}. 令

$$s_k = \alpha_k d_k, \quad y_k = \nabla_x L(x_{k+1}, \mu_{k+1}, \lambda_{k+1}) - \nabla_x L(x_k, \mu_{k+1}, \lambda_{k+1}),$$

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{z_k z_k^T}{s_k^T z_k}, \tag{12.45}$$

其中

$$z_k = \theta_k y_k + (1 - \theta_k) B_k s_k, \tag{12.46}$$

参数 θ_k 定义为

$$\theta_k = \begin{cases} 1, & \text{若 } s_k^T y_k \geq 0.2 s_k^T B_k s_k, \\ \dfrac{0.8 s_k^T B_k s_k}{s_k^T B_k s_k - s_k^T y_k}, & \text{若 } s_k^T y_k < 0.2 s_k^T B_k s_k. \end{cases} \tag{12.47}$$

步 7 令 $k := k + 1$, 转步 1.

注 (1) 算法12.4 步 5 隐含地假设了 A_k 是行满秩的. 如果这个条件不成立, 则在计算最小二乘乘子时, 需要使用计算广义你的技巧, 即

此时 (12.44) 为

$$\begin{bmatrix} \mu_{k+1} \\ \lambda_{k+1} \end{bmatrix} = A_{k+1}^{+} \nabla f_{k+1}, \tag{12.48}$$

其中 A_{k+1}^{+} 是 A_{k+1} 的 Penrose-Moore 广义逆.

(2) 算法 12.4 步 3 若选择 ℓ_1 价值函数

$$\phi(x, \sigma) = f(x) + \sigma^{-1}\big[\|h(x)\|_1 + \|g(x)_-\|_1\big],$$

可令

$$\tau = \max\{\|\mu_k\|, \|\lambda_k\|\},$$

任意选择一个 $\delta > 0$, 定义罚参数的修正规则为

$$\sigma_k = \begin{cases} \sigma_{k-1}, & \text{若 } \sigma_{k-1}^{-1} \geq \tau + \delta, \\ (\tau + 2\delta)^{-1}, & \text{若 } \sigma_{k-1}^{-1} < \tau + \delta. \end{cases} \tag{12.49}$$

在本节最后, 需要指出的是, 对于无约束优化问题, 所谓的“超线性收敛步”成立, 即如果 x^* 是 $f(x)$ 的稳定点, 且 Hesse 阵 $\nabla^2 f(x^*)$ 正定, 那么只要迭代序列 $\{x_k\} \to x^*$, 且搜索方向满足

$$\lim_{k\to\infty} \frac{\|x_k + d_k - x^*\|}{\|x_k - x^*\|} = 0,$$

对于充分大的 k, 就必然成立

$$f(x_k + d_k) < f(x_k).$$

换言之, 对于无约束优化问题, 超线性收敛步总是可以接受的. 但是对于约束优化问题情况并非如此, 即对于有些约束优化问题, 不管 x_k 如何靠近 x^*, 都不会有

$$\phi(x_k + d_k, \sigma) \le \phi(x_k, \sigma)$$

成立, 且 $x_k + d_k$ 对应的目标函数值和可行度比 x_k 对应的目标函数值和可行度还要差一些. 也就是说, 对于约束优化问题, 在这种情况下,

超线性收敛步是无法接受的, SQP 方法会失去收敛阶高的优点, 人们把这种现象称之为 Maratos 效应.

为了克服 Maratos 效应, 人们已经提出了许多方法, 如放松接受试探步 $x_k + d_k$ 的条件 (Watchdog 技术); 或者引进满足 $\|\tilde{d}_k\| = O(\|d_k\|^2)$ 的二阶校正步 $\tilde{d}_k$, 使得相应的价值函数值 $\phi(x_k + d_k + \tilde{d}_k, \sigma) < \phi(x_k, \sigma)$; 或者使用光滑的精确罚函数作为价值函数, 以提高超线性收敛步接受度, 例如, Schittkowski 提出增广拉格朗日函数

$$\begin{aligned}\phi(x, v, r) \;=\; & f(x) - \sum_{j\in E}\Big(v_j h_j(x) - \frac{1}{2} r_j h_j^2(x)\Big) \\ & - \sum_{j\in J(x,v)}\Big(v_j g_j(x) - \frac{1}{2} r_j g_j^2(x)\Big) - \frac{1}{2}\sum_{j\in K(x,v)} v_j^2/r_j,\end{aligned} \tag{12.50}$$

其中

$$J(x, v) = \{j \in I \,|\, g_j(x) \le v_j/r_j\}, \quad K(x, v) = \{j \in I \,|\, g_j(x) > v_j/r_j\}.$$

一般来说, ℓ_1 精确罚函数形式简单, 但是非光滑, 增广拉格朗日函数是光滑的, 数值计算结果较好.

§12.4 SQP 方法的 Matlab 程序

本节讨论 SQP 方法的 Matlab 实现. 注意到算法 12.4 每一迭代步的主要计算量是求解子问题 (12.42), 因此, 我们先讨论子问题的 Matlab 实现.

§12.4.1 SQP 子问题的 Matlab 实现

利用 KT 条件, 问题 (12.42) 等价于

$$H_1(d,\mu,\lambda) = B_k d - (A_k^E)^T\mu - (A_k^I)^T\lambda + \nabla f(x_k) = 0, \quad (12.51)$$

$$H_2(d,\mu,\lambda) = h(x_k) + A_k^E d = 0, \quad (12.52)$$

$$\lambda \geq 0,\ g(x_k) + A_k^I d \geq 0,\quad \lambda^T[g(x_k) + A_k^I d] = 0. \quad (12.53)$$

注意到 (12.53) 是一个 m 维的线性互补问题, 我们定义光滑 FB-函数

$$\phi(\varepsilon, a, b) = a + b - \sqrt{a^2 + b^2 + 2\varepsilon^2},$$

其中 $\varepsilon > 0$ 是光滑参数. 令

$$\Phi(\varepsilon, d, \lambda) = (\phi_1(\varepsilon, d, \lambda), \phi_2(\varepsilon, d, \lambda), \cdots, \phi_m(\varepsilon, d, \lambda))^T,$$

其中

$$\phi_i(\varepsilon, d, \lambda) = \lambda_i + [g_i(x_k) + (A_k^I)_i d] - \sqrt{\lambda_i^2 + [g_i(x_k) + (A_k^I)_i d]^2 + 2\varepsilon^2},$$

其中 $(A_k^I)_i$ 表示矩阵 A_k^I 的第 i 行. 记 $z = (\varepsilon, d, \mu, \lambda) \in \mathbb{R}_+ \times \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^l$. 那么方程组 (12.51)-(12.52) 等价于

$$H(z) := H(\varepsilon, d, \mu, \lambda) = \begin{bmatrix} \varepsilon \\ H_1(d, \mu, \lambda) \\ H_2(d, \mu, \lambda) \\ \Phi(\varepsilon, d, \lambda) \end{bmatrix} = 0. \qquad (12.54)$$

不难计算出 $H(z)$ 的 Jacobi 矩阵为

$$H'(z)=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & B_k & -(A_k^E)^T & -(A_k^I)^T\\ 0 & A_k^E & 0 & 0\\ v & D_2(z)A_k^I & 0 & D_1(z)\end{bmatrix}, \tag{12.55}$$

其中 $v=\nabla_\varepsilon\Phi(\varepsilon,d,\lambda)=(v_1,\cdots,v_m)^T$, v_i 由下式确定

$$v_i=-\frac{2\varepsilon}{\sqrt{\lambda_i^2+[g_i(x_k)+(A_k^I)_i d]^2+2\varepsilon^2}}, \tag{12.56}$$

而 $D_1(z)=\mathrm{diag}(a_1(z),\cdots,a_m(z))$, $D_2(z)=\mathrm{diag}(b_1(z),\cdots,b_m(z))$, 其

中 $a_i(z), b_i(z)$ 由下式确定

$$a_i(z) = 1 - \frac{\lambda_i}{\sqrt{\lambda_i^2 + [g_i(x_k) + (A_k^I)_i d]^2 + 2\varepsilon^2}}, \tag{12.57}$$

$$b_i(z) = 1 - \frac{g_i(x_k) + (A_k^I)_i d}{\sqrt{\lambda_i^2 + [g_i(x_k) + (A_k^I)_i d]^2 + 2\varepsilon^2}}. \tag{12.58}$$

给定参数 $\gamma \in (0,1)$, 定义非负函数

$$\beta(z) = \gamma \|H(z)\| \min\{1, \|H(z)\|\}. \tag{12.59}$$

算法 12.5 (求解子问题的光滑牛顿法)

步 0　选取 $\rho,\ \eta \in (0,1),\ \varepsilon_0 > 0,\ (d_0, \mu_0, \lambda_0) \in \mathbb{R}^n \times \mathbb{R}^l \times \mathbb{R}^m$. 置 $z_0 = (\varepsilon_0, d_0, \mu_0, \lambda_0)$, $\bar{z} = (\varepsilon_0, 0, 0, 0)$. 选取 $\gamma \in (0,1)$ 使 $\gamma\mu_0 < 1$ 及 $\gamma\|H(z_0)\| < 1$. 令 $j := 0$.

步 1　如果 $\|H(z_j)\| = 0$, 算法终止; 否则, 计算 $\beta_j = \beta(z_j)$.

步 2　求解下列方程组得解 $\Delta z_j = (\Delta\varepsilon_j, \Delta d_j, \Delta\mu_j, \Delta\lambda_j,)$,

$$H(z_j) + H'(z_j)\Delta z_j = \beta_j \bar{z}. \tag{12.60}$$

步 3　设 m_j 为满足下式的最小非负整数:

$$\|H(z_j + \rho^{m_j}\Delta z_j)\| \leq [1 - \sigma(1 - \beta\mu_0)\rho^{m_j}]\|H(z_j)\|. \tag{12.61}$$

令 $\alpha_j := \rho^{m_j}$, $z_{j+1} = z_j + \alpha_j \Delta z_j$.

步 4　令 $j := j + 1$, 转步 1.

下面我们给出算法 12.5 的 Matlab 程序.

程序 12.1 利用光滑牛顿法求解二次规划子问题.

```
function  [d,mu,lam,val,k]=qpsubp(dfk,Bk,Ae,hk,Ai,gk)
% 功能: 求解二次规划子问题:  min qk(d)=0.5*d'*Bk*d+dfk'*d+,
%          s.t.   hk+Ae*d=0,  gk+Ai*d>=0.
%输入:  dfk是xk处的梯度, Bk是第k次近似Hesse阵, Ae,hk线性等式约束
%          的有关参数, Ai,gk是线性不等式约束的有关参数
```

```
%输出： d,val分别是是最优解和最优值, mu,lam是乘子向量, k是迭代次数.
n=length(dfk);  l=length(hk); m=length(gk);
gamma=0.05;  epsilon=1.0e-6;  rho=0.5;  sigma=0.2;
ep0=0.05;  mu0=0.05*zeros(l,1); lam0=0.05*zeros(m,1);
d0=ones(n,1);  u0=[ep0;zeros(n+l+m,1)];
z0=[ep0; d0; mu0;lam0,];
k=0;   %k为迭代次数
z=z0; ep=ep0; d=d0; mu=mu0;  lam=lam0;
while (k<=150)
   dh=dah(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk);
   if(norm(dh)<epsilon)
       break;
   end
   A=JacobiH(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk);
   b=beta(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk,gamma)*u0-dh;
   dz=A\b;
   if(l>0&m>0)
       de=dz(1); dd=dz(2:n+1); du=dz(n+2:n+l+1); dl=dz(n+l+2:n+l+m+1);
   end
   if(l==0)
```

```
   de=dz(1);   dd=dz(2:n+1);  dl=dz(n+2:n+m+1);
end
if(m==0)
   de=dz(1);   dd=dz(2:n+1);  du=dz(n+2:n+l+1);
end
i=0; %mk=0;
while (i<=20)
  if(l>0&m>0)
   dh1=dah(ep+rho^i*de,d+rho^i*dd,mu+rho^i*du,lam+rho^i*dl,dfk,Bk,Ae,hk,Ai,gk);
  end
  if(l==0)
    dh1=dah(ep+rho^i*de,d+rho^i*dd,mu,lam+rho^i*dl,dfk,Bk,Ae,hk,Ai,gk);
  end
  if(m==0)
     dh1=dah(ep+rho^i*de,d+rho^i*dd,mu+rho^i*du,lam,dfk,Bk,Ae,hk,Ai,gk);
  end
  if(norm(dh1)<=(1-sigma*(1-gamma*ep0)*rho^i)*norm(dh))
     mk=i; break;
  end
  i=i+1;
```

```
      if(i==20), mk=10; end
   end
   alpha=rho^mk;
   if(l>0&m>0)
      ep=ep+alpha*de;    d=d+alpha*dd;
      mu=mu+alpha*du;   lam=lam+alpha*dl;
   end
   if(l==0)
      ep=ep+alpha*de;    d=d+alpha*dd;
      lam=lam+alpha*dl;
   end
   if(m==0)
      ep=ep+alpha*de;    d=d+alpha*dd;
      mu=mu+alpha*du;
   end
   k=k+1;
end
%%%%%%%%%%%%%%%%%%%%%%%%%
function p=phi(ep,a,b)
p=a+b-sqrt(a^2+b^2+2*ep^2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%
function dh=dah(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk)
n=length(dfk); l=length(hk); m=length(gk);
dh=zeros(n+l+m+1,1);
dh(1)=ep;
if(l>0&m>0)
   dh(2:n+1)=Bk*d-Ae'*mu-Ai'*lam+dfk;
   dh(n+2:n+l+1)=hk+Ae*d;
   for(i=1:m)
      dh(n+l+1+i)=phi(ep,lam(i),gk(i)+Ai(i,:)*d);
   end
end
if(l==0)
   dh(2:n+1)=Bk*d-Ai'*lam+dfk;
   for(i=1:m)
     dh(n+1+i)=phi(ep,lam(i),gk(i)+Ai(i,:)*d);
   end
end
if(m==0)
   dh(2:n+1)=Bk*d-Ae'*mu+dfk;
```

```
   dh(n+2:n+l+1)=hk+Ae*d;
end
dh=dh(:);
%%%%%%%%%%%%%%%%%%%%%%%%%
function bet=beta(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk,gamma)
dh=dah(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk);
bet=gamma*norm(dh)*min(1,norm(dh));
%%%%%%%%%%%%%%%%%%%%%%%%%
function [dd1,dd2,v1]=ddv(ep,d,lam,Ai,gk)
m=length(gk);
dd1=zeros(m,m); dd2=zeros(m,m); v1=zeros(m,1);
for(i=1:m)
    fm=sqrt(lam(i)^2+(gk(i)+Ai(i,:)*d)^2+2*ep^2);
    dd1(i,i)=1-lam(i)/fm;
    dd2(i,i)=1-(gk(i)+Ai(i,:)*d)/fm;
    v1(i)=-2*ep/fm;
end
%%%%%%%%%%%%%%%%%%%%%%%%%
function A=JacobiH(ep,d,mu,lam,dfk,Bk,Ae,hk,Ai,gk)
n=length(dfk); l=length(hk); m=length(gk);
```

```
A=zeros(n+l+m+1,n+l+m+1);
[dd1,dd2,v1]=ddv(ep,d,lam,Ai,gk);
if(l>0&m>0)
    A=[1,            zeros(1,n),  zeros(1,l),  zeros(1,m);
       zeros(n,1),   Bk,          -Ae',        -Ai';
       zeros(l,1),   Ae,          zeros(l,l),  zeros(l,m) ;
       v1,           dd2*Ai,      zeros(m,l),  dd1];
end
if(l==0)
     A=[1,          zeros(1,n),  zeros(1,m);
        zeros(n,1), Bk,          -Ai';
        v1,         dd2*Ai,      dd1];
end
if(m==0)
    A=[1,           zeros(1,n), zeros(1,l);
       zeros(n,1),  Bk,          -Ae';
       zeros(l,1),  Ae,        zeros(l,l)];
end
```

下面我们利用程序 12.1, 求解三个分别是纯等式约束、纯不等式

约束以及混合约束的二次规划问题.

例 12.3 解二次规划问题

$$\begin{aligned} \min \quad & f(x) = x_1^2 + 2x_2^2 + x_3^2 - 2x_1x_2 + x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 - 4 = 0, \\ & 2x_1 - x_2 + x_3 - 2 = 0. \end{aligned}$$

该问题的极小点为 $x^* = \left(\frac{21}{11}, \frac{43}{22}, \frac{3}{22}\right)^T$.

解 在 Matlab 命令窗口依次输入下列命令:

```
dfk=[0 0 1]';
Bk=[2 -2 0; -2 4 0; 0 0 2];
Ae=[1 1 1; 2 -1 1];
hk=[-4 -2]';
Ai=[];
gk=[];
[d,mu,lam,val,k]=qpsubp(dfk,Bk,Ae,hk,Ai,gk)
```

得到计算结果:

```
d =
    1.9091
    1.9545
    0.1364
mu =
    2.6364
   -1.3636
lam =
    Empty matrix: 0-by-1
val =
    3.9773
k =
    2
```

例 12.4 解二次规划问题

$$\begin{aligned}\min\ & f(x)=\frac{1}{2}x_1^2-x_1x_2+x_2^2-6x_1-2x_2\\ \text{s.t.}\ & -2x_1-x_2+3\geq 0,\\ & x_1-x_2+1\geq 0;\\ & -x_1-2x_2+2\geq 0;\\ & x_1,x_2\geq 0.\end{aligned}$$

该问题的极小点为 $x^*=\left(\frac{4}{3},\frac{1}{3}\right)^T$.

解 在 Matlab 命令窗口依次输入下列命令:

```
dfk=[-6 -2]';
Bk=[1 -1; -1 2];
Ae=[ ];
hk=[ ]';
Ai=[-2 -1;1 -1;-1 -2; 1 0; 0 1];
gk=[3 1 2 0 0]';
[d,mu,lam,val,k]=qpsubp(dfk,Bk,Ae,hk,Ai,gk)
```

得到计算结果:

```
d =
  1.3333
  0.3333
mu =
  Empty matrix: 0-by-1
lam =
  2.4444
  0.0000
  0.1111
  0.0000
  0.0000
val =
  -8.1111
k =
  5
```

例 12.5 解二次规划问题

$$\begin{aligned} \min \quad & f(x) = x_1^2 + x_1x_2 + 2x_2^2 - 6x_1 - 2x_2 - 12x_3 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 - 2 = 0, \\ & x_1 - 2x_2 + 3 \geq 0; \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

该问题的极小点为 $x^* = (0, 0, 2)^T$.

解 在 Matlab 命令窗口依次输入下列命令:

```
dfk=[-6 -2 -12]';
Bk=[2 1 0;1 4 0; 0 0 0];
Ae=[1 1 1];
hk=[-2]';
Ai=[1 -2 0;1 0 0; 0 1 0;0 0 1];
gk=[3 0 0 0]';
[d,mu,lam,val,k]=qpsubp(dfk,Bk,Ae,hk,Ai,gk)
```

得到计算结果:

```
d =
    0.0000
    0.0000
    2.0000
mu =
    12.0000
lam =
    0.0000
    6.0000
    10.0000
    0.0000
val =
  -24.0000
k =
  6
```

§12.4.2 SQP 方法的 Matlab 实现

本小节给出 SQP 方法 (算法12.4) 的一个 Matlab 程序, 该程序在

某种意义上是通用的, 不同的问题只需编写目标函数、约束函数以及它们的梯度和 Jacobi 矩阵的 M 文件即可调用该程序.

程序 12.2 一般约束优化问题 SQP 方法的 Matlab 程序, 该程序在每一迭代步调用了程序 12.1 qpsubp.m 求解二次规划子问题.

```
function [x,mu,lam,val,k]=sqpm(x0,mu0,lam0)
%功能: 用基于拉格朗日函数Hesse阵的SQP方法求解约束优化问题:
%      min f(x)     s.t. h_i(x)=0, i=1,..., l.
%输入: x0是初始点, mu0是乘子向量的初始值
%输出: x, mu分别是近似最优点及相应的乘子,
%val是最优值, mh是约束函数的模, k是迭代次数.
maxk=100;   %最大迭代次数
n=length(x0); l=length(mu0); m=length(lam0);
rho=0.5; eta=0.1;  B0=eye(n);
x=x0; mu=mu0;  lam=lam0;
Bk=B0; sigma=0.8;
epsilon1=1e-6; epsilon2=1e-5;
[hk,gk]=cons(x);  dfk=df1(x);
[Ae,Ai]=dcons(x); Ak=[Ae; Ai];
k=0;
```

```
while(k<maxk)
    [dk,mu,lam]=qpsubp(dfk,Bk,Ae,hk,Ai,gk);  %求解子问题
    mp1=norm(hk,1)+norm(max(-gk,0),1);
    if(norm(dk,1)<epsilon1)&(mp1<epsilon2)
        break;
    end  %检验终止准则
    deta=0.05;  %罚参数更新
    tau=max(norm(mu,inf),norm(lam,inf));
    if(sigma*(tau+deta)<1)
        sigma=sigma;
    else
        sigma=1.0/(tau+2*deta);
    end
    i=0;   %Armijo搜索
    while(i<=20)
      if(phi1(x+rho^i*dk,sigma)-phi1(x,sigma)<eta*rho^i*dphi1(x,sigma,dk))
        mk=mm;
        break;
      end
      i=i+1;
```

```
  if(i==20),   mk=10;   end
end
alpha=rho^mk; x1=x+alpha*dk;
[hk,gk]=cons(x1);  dfk=df1(x1);
[Ae,Ai]=dcons(x1);  Ak=[Ae; Ai];
lamu=pinv(Ak)'*dfk;  %计算最小二乘乘子
if(l>0&m>0)
   mu=lamu(1:l); lam=lamu(l+1:l+m);
end
if(l==0), mu=[]; lam=lamu;  end
if(m==0), mu=lamu; lam=[];  end
sk=alpha*dk;  %更新矩阵Bk
yk=dlax(x1,mu,lam)-dlax(x,mu,lam);
if(sk'*yk>0.2*sk'*Bk*sk)
   theta=1;
else
   theta=0.8*sk'*Bk*sk/(sk'*Bk*sk-sk'*yk);
end
zk=theta*yk+(1-theta)*Bk*sk;
Bk=Bk+zk*zk'/(sk'*zk)-(Bk*sk)*(Bk*sk)'/(sk'*Bk*sk);
```

```
   x=x1;  k=k+1;
end
val=f1(x);
%p=phi1(x,sigma)
%dd=norm(dk)
%%%%%%% l1精确价值函数 %%%%%%%
function p=phi1(x,sigma)
f=f1(x); [h,g]=cons(x); gn=max(-g,0);
l0=length(h);  m0=length(g);
if(l0==0), p=f+1.0/sigma*norm(gn,1); end
if(m0==0),  p=f+1.0/sigma*norm(h,1); end
if(l0>0&m0>0)
   p=f+1.0/sigma*(norm(h,1)+norm(gn,1));
end
%%%%% 价值函数的方向导数%%%%%
function dp=dphi1(x,sigma,d)
df=df1(x); [h,g]=cons(x);  gn=max(-g,0);
l0=length(h);  m0=length(g);
if(l0==0),  dp=df'*d-1.0/sigma*norm(gn,1); end
if(m0==0), dp=df'*d-1.0/sigma*norm(h,1); end
```

```
if(l0>0&m0>0)
    dp=df'*d-1.0/sigma*(norm(h,1)+norm(gn,1));
end
%%%%%%%% 拉格朗日函数 L(x,mu) %%%%%%%%%%%%
function l=la(x,mu,lam)
f=f1(x);   %调用目标函数文件
[h,g]=cons(x);  %调用约束函数文件
l0=lemgth(h);  m0=length(g);
if(l0==0), l=f-lam*g;  end
if(m0==0),  l=f-mu'*h;  end
if(l0>0&m0>0)
   l=f-mu'*h-lam'*g;
end
%%%%%%%% 拉格朗日函数的梯度 %%%%%%%%%%%%
function dl=dlax(x,mu,lam)
df=df1(x); %调用目标函数梯度文件
[Ae,Ai]=dcons(x);  %调用约束函数Jacobi矩阵文件
[m1,m2]=size(Ai); [l1,l2]=size(Ae);
if(l1==0), dl=df-Ai'*lam; end
if(m1==0), dl=df-Ae'*mu; end
```

```
if(l1>0&m1>0), dl=df-Ae'*mu-Ai'*lam; end
```

下面我们利用程序 12.2 来计算两个约束优化问题的极小点.

例 12.6 解非线性规划问题

$$\begin{aligned} \min \quad & f(x) = -\pi x_1^2 x_2 \\ \text{s.t.} \quad & \pi x_1 x_2 + \pi x_1^2 - 150 = 0, \\ & x_1 \geq 0, \ x_2 \geq 0. \end{aligned}$$

解 首先编写四个 m 函数:

```
%%%%%%%目标函数f(x)%%%%%%%%%%
function f=f1(x)  %f1.m
f=-pi*x(1)^2*x(2);
%%%%%%%目标函数f(x)的梯度%%%%%%%%
function df=df1(x)  %df1.m
df=[-2*pi*x(1)*x(2),-pi*x(1)^2]';
%%%%%%%%%约束函数 %%%%%%%%%%%%%%
function [h,g]=cons(x)  %cons.m
```

```
h=[pi*x(1)*x(2)+pi*x(1)^2-150];
g=[x(1);x(2)];
%%%%%%%%约束函数Jacobi矩阵%%%%%%%%
function [dh,dg]=dcons(x) %dcons.m
dh=[pi*x(2)+2*pi*x(1), pi*x(1)];
dg=[1 0; 0 1];
```

在 Matlab 命令窗口依次输入下列命令:

```
x0=[3 3]';
mu0=[0]';
lam0=[0 0]';
[x,mu,lam,val,k]=sqpm(x0,mu0,lam0)
```

得到计算结果:

```
x =
   3.9894
   7.9788
mu =
   -3.9894
lam =
   1.0e-007 *
```

```
    0.4248
    0.0039
val =
   -398.9423
k =
    8
```

例 12.7 解非线性规划问题

$$\begin{aligned}\min\ & f(x)=x_1^2+x_2^2-16x_1-10x_2\\ \text{s.t.}\ & -x_1^2+6x_1-4x_2+11\geq 0,\\ & x_1x_2-3x_2-\mathrm{e}^{x_1-3}+1\geq 0,\\ & x_1\geq 0,\ x_2\geq 0.\end{aligned}$$

解 首先编写四个 m 函数:

```
%%%%%%%%目标函数f(x)%%%%%%%%%%
function f=f1(x)  %f1.m
f=x(1)^2+x(2)^2-16*x(1)-10*x(2);
```

```
%%%%%%%目标函数f(x)的梯度%%%%%%%%
function df=df1(x)  %df1.m
df=[2*x(1)-16; 2*x(2)-10];
%%%%%%%%%约束函数 %%%%%%%%%%%%%%
function [h,g]=cons(x)  %cons.m
h=[ ];
g=[-x(1)^2+6*x(1)-4*x(2)+11; ...
   x(1)*x(2)-3*x(2)-exp(x(1)-3)+1;x(1);x(2)];
%%%%%%%%约束函数Jacobi矩阵%%%%%%%%
function [dh,dg]=dcons(x) %dcons.m
dh=[ ];
dg=[-2*x(1)+6,-4;x(2)-exp(x(1)-3),x(1)-3;1,0;0,1];
```

在 Matlab 命令窗口依次输入下列命令:

```
x0=[4 4]';
mu0=[ ]';
lam0=[0 0 0 0]';
[x,mu,lam,val,k]=sqpm(x0,mu0,lam0)
```

得到计算结果:

```
x =
```

```
    5.2396
    3.7460
mu =
    Empty matrix: 0-by-1
lam =
    0.8133
    0.3327
    0.0000
    0.0000
val =
   -79.8078
k =
    5
```