DOI:10.13374/j.issn1001-053x.1981.04.006

约束最优化程序设计*

计算机应用室　吴继庚

摘　要

本文介绍了约束最优化问题求解的惩罚函数法（SUMT调用Powell法），附有程序框图及用ALGOL语言编制的过程。文中简要介绍了该方法的基本思想，对“惩罚”的实质和近似求解进行了分析，提出了Powell法的两个加速收敛的条件以及每一计算步骤的终止准则。文中还附有使用本过程的简要说明与计算实例。

最优化方法近年得到了迅速的发展，它研究从尽可能多的可行设计方案中，寻求最佳设计方案的方法。在现代控制理论、系统工程和计算机辅助设计等方面已经成为不可缺少的组成部分。这门学科是在实物模型、数学规划论和电算技术等理论基础上，进行综合分析的计算系统。目前，在国内外许多技术领域得到了广泛的应用，并取得了显著的效果。最优设计方法，首先将所研究的实物模型转换为一定格式的数学模型，再送入最优化方法的电算程序，在电子计算机上计算求解，得出最佳设计方案。

本文用ALGOL语言编制的惩罚函数法求解约束最优化问题的过程，在一九七九年八月试算通过后，已陆续在冶金机械、矿山机械、机械设计与机械制造等方面计算出了正确的结果，并写出了专题报告。经过二年来的上机计算实践证明：程序简单、易于掌握、运算迅速、结果正确，适用于中小题目的计算和求解非线性方程组的问题。

一、基本思想

1. 惩罚函数法（SUMT调用Powell法）就是将约束最优化问题转化为一系列无约束最优化问题来求解，即序列无约束极小化方法—SUMT(Seguential Unconstrained Minimization Technigue)法。这个方法的实质是将约束条件，即对等约束函数$H_v(\mathbf{X})$和不等约束函数$Gu(\mathbf{X})$乘一个或几个可变化的数列——加权参数r(惩罚因子），再与原目标函数$F(\mathbf{X})$按照某种假设条件，构造成一系列新的无约束目标函数$\Phi(\mathbf{X}, r)$，称它为惩罚函数（Penalty Function）。用比较成熟的无约束最优化求解的方法求得惩罚函数$\Phi(\mathbf{X}, r)$的最优化解，以此作为原目标函数$F(\mathbf{X})$的约束近似解。

构造惩罚函数$\Phi(\mathbf{X}, r)$的形式分为内点法、外点法和混合点法。内点法是将惩罚函数

* 本文在编写过程中得到了陈立周同志和冶金机械教研室同志的指导和帮助。

定义在可行区内，从一个初始可行点逐步逼近最优点。它的形式为：

$$\Phi(\overline{X},\ r)=F(\overline{X})+r\sum_{u=1}^{m}1/Gu(\overline{X})$$

或

$$\Phi(\overline{X},\ r)=F(\overline{X})+r\sum_{u=1}^{m}\ln(Gu(\overline{X}))$$

其中：　　$Gu(\overline{X})>0$　　$(u=1,2,\cdots\cdots,m)$

式中可变化的惩罚因子r是一个递减数列：$r^{(0)}>r^{(1)}>r^{(2)}>\cdots\cdots>r^{(k)}$。每确定一个r值就有一个$\Phi(\overline{X},\ r)$无约束函数求解的过程。由于r是一组变化的数列，因此就有一系列的无约束函数$\Phi(\overline{X},\ r)$的求解过程。所以惩罚函数法就是将约束最优化问题转化为一系列的无约束最优化问题求解的方法。由内点法的定义可以看出，初始点$\overline{X}^{(0)}$、迭代过程中的各点$\overline{X}^{(i)}$以及最终优化点$\overline{X}^{(*)}$等都在可行区内，因此，它的解是安全、保守、可靠的。但对于设计变量较多，约束条件又复杂的题目，人为寻求一个可行的初始点$\overline{X}^{(0)}$是较困难的。相反，外点法是将惩罚函数定义在非可行区，即从一个非可行的初始点$\overline{X}^{(0)}$，逐步迭代逼近最优点，而最优点$\overline{X}^{(*)}$并非在可行区内，因此它的解是非可靠的、不安全的。但它对初始点$\overline{X}^{(0)}$要求不严格，而且还能处理等约束问题。它的表达形式是：

$$\Phi(\overline{X},\ r)=F(\overline{X})+r\sum_{u=1}^{m}(Gu(\overline{X}))^2$$

其中：　　$Gu(\overline{X})\geq 0$　　$(u=1,2,\cdots\cdots,m)$

式中惩罚因子r是一个递增数列$r^{(0)}<r^{(1)}<r^{(2)}<\cdots\cdots<r^{(k)}$。

对于上述内点法与外点法所存在的问题，目前在实际应用中都已得到解决。如在内点法中加入随机选择初始可行点的功能，在外点法中将可行区的范围缩小等措施，都是行之有效的方法。

混合点法即是将外点法能处理等约束条件的特点，加到安全可靠的内点法中来，形成了现在常用的罚函数法。

已知设计变量$\overline{X}=[x_1,\ x_2,\cdots\cdots,x_n]^T$

求解目标函数$F(\overline{X})$

满足不等约束函数$Gu(\overline{X})\geq 0$ $(u=1,2,\cdots\cdots,m)$ 和等约束函数$H_v(\overline{X})=0$ $(v=1,2,\cdots\cdots,p)$ $(p<n)$ 的约束条件下极小值的问题。

按照某些假设条件，混合点法罚函数的构造形式是：

$$\Phi(\overline{X},r)=F(\overline{X})+r\sum_{u=1}^{m}\frac{1}{Gu(\overline{X})}+\frac{1}{\sqrt{r}}\sum_{v=1}^{p}[H_v(\overline{X})]^2$$

由上式可知惩罚函数$\Phi(\overline{X},\ r)$具有如下四条极限性质：

$$\lim_{K\to\infty}r^{(K)}=0$$

$$\lim_{K\to\infty}r^{(K)}\sum_{u=1}^{m}\frac{1}{Gu(\overline{X})}=0\quad(u=1,2,\cdots\cdots,m)$$

$$\lim_{K\to\infty}\frac{1}{\sqrt{r^{(K)}}}\sum_{v=1}^{p}[H_v(\overline{X})]^2=0\quad(v=1,2,\cdots\cdots,p)$$

$$\lim_{K\to\infty}\Phi(\overline{X},\ r^{(K)})=F(\overline{X})$$

上述性质揭示了罚函数的实质，即当惩罚因子$r^{(K)}$由一个初值逐渐递减变化趋近于零

时，惩罚项$r^{(K)}\sum_{u=1}^{m}1/Gu(\bar{X})$和$(1/\sqrt{r^{(K)}})\sum_{v=1}^{p}[H_v(\bar{X})]^2$也趋近于零，最后使新目标函数$\Phi(\bar{X}, r^{(K)})$的值趋近于原目标函数$F(\bar{X})$的约束解。因此，罚函数的解$\bar{X}^{(*)}$就是原目标函数$F(\bar{X})$的约束近似解。

对于罚函数$\Phi(\bar{X}, r^{(K)})$在无约束最优化求解过程中，约束条件是如何起到限制设计变量不会超界呢？试看，约束函数$Gu(\bar{X})\geq 0$和$H_v(\bar{X})=0$经过数学变换，在罚函数$\Phi(\bar{X}, r^{(K)})$中构成了两个惩罚项$r^{(K)}\sum_{u=1}^{m}1/Gu(\bar{X})$与$(1/\sqrt{r^{(K)}})\sum_{v=1}^{p}[H_v(\bar{X})]^2$，当$r^{(K)}$为某一个定值时，如果某个设计变量$\bar{X}^{(i)}$在变化过程中趋近于边界约束条件或性能约束条件，即将要破坏某些不等约束条件时，使得$G_j(\bar{X})\approx 0$，则$1/G_j(\bar{X})\to\infty$，相应$r^{(K)}\sum_{u=1}^{m}1/G_u(\bar{X})$项变得很大，促使函数$\Phi(\bar{X}, r^{(K)})$值猛然增加，而且$G_j(\bar{X})$值越小，则$\Phi(\bar{X}, r^{(K)})$值就越大。这个变化相当于在约束边界上筑起了一道“高墙”，而且这道“高墙”是越来越高。因此，迫使设计变量在一维搜索变化时，不会再沿着使罚函数$\Phi(\bar{X}, r^{(K)})$值增加的方向发展，也就不会使设计变量再超越约束条件的限制。至此，惩罚项$r^{(K)}\sum_{u=1}^{m}1/G_u(\bar{X})$将起到了不等约束的作用。同理，另一惩罚项$(1/\sqrt{r^{(K)}})\sum_{v=1}^{p}[H_v(\bar{X})]^2$的函数值，是随着设计变量$\bar{X}^{(i)}$越远离最优化点时而增大，而且距离越远此项值越大，也会促使罚函数$\Phi(\bar{X}, r^{(K)})$值猛然增加，同样也筑起了一道“高墙”，必然使设计变量$\bar{X}^{(i)}$在一维搜索变化时，向着使罚函数$\Phi(\bar{X}, r^{(K)})$值减小的方向变化，至此，也起到了等约束的作用。从而可以看到罚函数法近似求解的过程以及约束条件是如何起作用的。这两个特点就是罚函数法的“惩罚实质”。

目前，将约束最优化问题转化为一系列的无约束最优化问题求解的方法很多，惩罚函数$\Phi(\bar{X}, r)$的构型方案也很多。这里采用的公式是：

$$\Phi(\bar{X}, r)=F(\bar{X})+r\sum_{u=1}^{m}\frac{1}{G_u(\bar{X})}+\frac{1}{\sqrt{r}}\sum_{v=1}^{p}[H_v(\bar{X})]^2$$

此公式简单、易懂，在实际应用中是成功的。

关于如何选取惩罚因子r的初始值$r^{(0)}$，有关资料介绍的理论和公式是较多的。但这里人为的给定一个初始值$r^{(0)}\geq 1$的数，在实际应用中再进行调整变化。经验证明：$r^{(0)}$取小值时，使得罚函数构型次数减少，节省计算时间，但容易使其计算结果精度偏低或造成计算失败。相反，$r^{(0)}$取值过大，虽然罚函数构型次数增加，多耗机时，但它对罚函数的每次构型是有利的。对于惩罚因子$r^{(K)}$值数列递减变化的规律，可用一个固定的小于1的系数C逐次去乘$r^{(K)}$，即得到迭代公式$r^{(K+1)}=r^{(K)}*C$。经验证明：C的取值范围可在0.1～0.5之间，或者取值更小C=0.01～0.1。

2. 在无约束最优化方法中，虽然种类繁多，但实质上多数是利用一维搜索（将多元函数看作是一元函数，求解最佳值）迭代求解的方法，其迭代公式是：

$$\bar{X}^{(K+1)}=\bar{X}^{(K)}+\alpha^{(K)}\bar{S}^{(K)}$$

其中：$\bar{X}^{(K+1)}$——迭代后所求新点

$\bar{X}^{(K)}$——迭代前原始初点

$\alpha^{(K)}$——迭代中优化步长

$\bar{S}^{(K)}$——迭代中优化方向

在每一次迭代过程中，都是以原始初点$\bar{X}^{(K)}$出发，沿着本次迭代优化方向和优化步长，得到一个新点$\bar{X}^{(K+1)}$，这样反复迭代求新点，最后即可得到满足一定精度要求的无约束最优化解。上述迭代过程的关键是：每迭代一次都要寻求本次迭代过程中的最优化方向$\bar{S}^{(K)}$和最优化步长$\alpha^{(K)}$。关于求解$\bar{S}^{(K)}$与$\alpha^{(K)}$的方法很多，各有其特点。本文选择最优化方向的方法是改进的共轭方向法——Powell法，选择最优化步长的方法是二次插值法。

Powell法是在共轭方向法的基础上，为了加快迭代收敛速度，提出了如下的分析：

(1) 经过一轮一维搜索迭代求解后，所求新的方向$\bar{S}^{(n+1)}$在共轭方向法中就直接选用了，而Powell法要经过分析判断，然后再决定是否取舍方向$\bar{S}^{(n+1)}$，其判断条件是：

$$f_3 < f_1 \quad 与$$

$$(f_1-2f_2+f_3)(f_1-f_2-\Delta_m)<0.5\Delta_m(f_1-f_3)^2$$

式中：f_1，f_2，f_3分别为一轮搜索过程中初始点的函数值，终止点的函数值和反射点的函数值。Δ_m为相邻两维函数值之差的最大值。

上述两式同时成立时，则选取方向$\bar{S}^{(n+1)}$为共轭方向，否则仍用原来的方向矩阵进行下一轮的一维搜索。

(2) 在选用共轭方向$\bar{S}^{(n+1)}$后，共轭方向法则用$\bar{S}^{(n+1)}$取代最前面的方向$\bar{S}^{(1)}$。而Powell法则采用淘汰相邻两维函数值之差最大值Δ_m所对应的那个方向$\bar{S}^{(m)}$，即是用$\bar{S}^{(n+1)}$排列在原方向矩阵的最后一列，再删掉原方向矩阵中$\bar{S}^{(m)}$列向量，形成新的方向矩阵：

$$\bar{S}^{(1)},\ \bar{S}^{(2)},\cdots\cdots,\bar{S}^{(m-1)},\bar{S}^{(m+1)},\cdots\cdots,\bar{S}^{(n)},\bar{S}^{(n+1)}$$

其中：$\Delta_m=\max|f_{i-1}-f_i| \quad i=1,2,\cdots\cdots,n$

由此可见，Powell法中采用上述两个分析后，即可比共轭方向法具有更快的收敛速度。

二次插值方法是求解最优化步长的常用方法，其所用的公式为：

$$\alpha^*=0.5(\alpha_1+\alpha_3-c_1/c_2)$$

$$c_1=(f_3-f_1)/(\alpha_3-\alpha_1)$$

$$c_2=((f_2-f_1)/(\alpha_2-\alpha_1)-c_1)/(\alpha_2-\alpha_3)$$

式中：α_1，α_2，α_3为一维搜索中二次插值各点值，f_1，f_2，f_3为各插值点所对应的插值函数值，α^*为二次插值的极值点，即为所求的最优化步长。

3. 求解终止准则：

关于二次插值求解最优化步长α^*时的终止准则是：

$$\left|\frac{f_2-f_4}{f_2}\right|<\varepsilon_1 \qquad |f_2-f_4|<\varepsilon_1$$

$$|\alpha_2-\alpha_1|<\varepsilon_1 \qquad |\alpha_2-\alpha_3|<\varepsilon_1$$

其中任意一个条件成立时，即可终止求解。

关于Powell法求解惩罚函数$\Phi(\bar{X},\ r^{(K)})$的无约束最优化问题极小化解的终止准则是：

$$\sum_{i=1}^{n}(x_i^{(0)}-x_i^{(n)})^2<\varepsilon_1$$

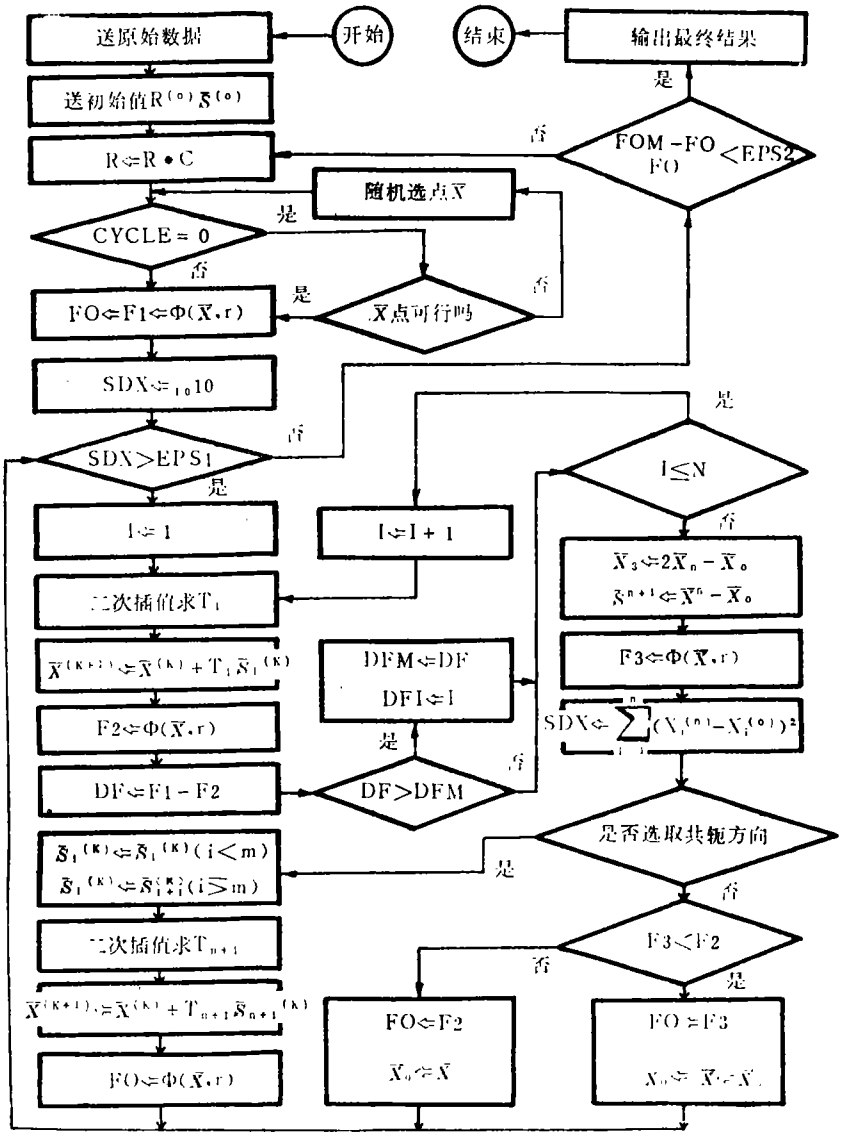
关于惩罚函数$\Phi(\bar{X},\ r^{(K)})$的求解逐步趋近于原目标函数$F(\bar{X})$的解，是随着惩罚因子$r^{(K)}$值数列的递减变化而不断逼近。其求解终止准则是：

$$\left|\frac{\Phi(\bar{X}^{(K)},r^{(K)})-\Phi(\bar{X}^{(K+1)},r^{(K+1)})}{\Phi(\bar{X}^{(K+1)},r^{(K+1)})}\right|<\varepsilon_2$$

上述各式中的ε_1与ε_2皆为给定精度值，本文取值为：$\varepsilon_1=10^{-7}$；$\varepsilon_2=10^{-3}$

二、程序设计

1. 程序框图：



SUMT——PoWell方法程序框图

2. SUMT过程全文：

```
PROCEDURE SUMT(N,FK,GK,HK
    EPS1, EPS2, R, X, BL, BU, FX,
    GX,HX, FGH);
VALUE N , FK , GK , HK, EPS1,
    EPS2, R;
INTEGER N, FK, GK, HK
REAL EPS1, EPS2, R;
ARRAY X, BL, BU, FX, GX, HX;
PROCEDURE FGH;
BEGIN
INTEGER I,K,CYCLE, ITER, FUN;
REAL C,YO,FO,HO,TO,FOM,SDX,
    Q,M,M35,M36,M37,SFO,SF,SG,
    SH;
ARRAY XO , XX [1:N], S[1:N+1,
    1:N];
PROCEDURE RANDOM;
BEGIN
M:=M*5;
IF M≥M37 THEN M:=M-M37;
IF M≥M36 THEN M:=M-M36;
IF M≥M35 THEN M:=M-M35;
Q:=M/M35;
END;
PROCEDURE FUNCT(X, Y);
REAL Y; ARRAY X;
BEGIN
FGH(X);
FUN:=FUN+1;
SF:=SG:=SH:=0;
FOR K:=1 STEP 1 UNTIL FK DO
SF:=SF+FX(K);
FOR K:=1 STEP 1 UNTIL GK DO
SG:=SG+1/GX(K);
FOR K:=1 STEP 1 UNTIL HK DO
SH:=SH+HX(K)*HX(K);
Y:=SF+R*SG+SH/#SQRT(R);
IF FUN=1 THEN SFO:=SF;
IF FUN=1 V #BO(GOOOO3,1)<O
    THEN
#PRINT(O,'10',R,Y,SF,SG,SH,X,
    FX,GX,HX);
END;
PROCEDURE PENALTY(T,Y);
REAL T,Y;
BEGIN
TO:=T-TO;
FOR K:=1 STEP 1 UNTIL N DO
XX(K):=XX(K)+TO*S(I, K)
TO:=T;
FUNCT(XX, Y);
END;
PROCEDURE LINEMIN(HO, T, Y);
VALUE HO; REAL HO, T, Y;
BEGIN
REAL HT, T1, T2,T3,T4,Y1,Y2,Y3,
    Y4,C1,C2,A;
FOR K:=1 STEP 1 UNTIL N DO
XX(K):=X(K);
HT:=T2:=HO; TO:=T1:=O;
    Y1:=YO;
L7: PENALTY(T2, Y2);
FOR K:=1 STEP 1 UNTIL GK DO
IF GX(K)<10-15 THEN BEGIN
T2:=T2*0.5;
GOTO L7; END;
IF Y2<Y1 THEN GOTO L2;
HT:=-HT; T3:=T1; Y3:=Y1;
L1: T1:=T2; Y1:=Y2;
    T2:=T3; Y2:=Y3;
L2: T3:=T2+HT;
PENALTY(T3, Y3);
FOR K:=1 STEP 1 UNTIL GK DO
IF GX(K)<10-15 THEN BEGIN
HT:=HT*0.5;
GOTO L2; END;
IF Y2>Y3 THEN BEGIN
HT:=HT+HT;
```

```
GOTO L1; END;
IF ¬ABS(T2-T1)≤10-10 ∨
¬ABS(T2-T3)≤10-10 THEN GOTO
    L4;
L3: C1:=(Y3-Y1)/(T3-T1);
C2:=((Y2-Y1)/(T2-T1)-C1)/(T2-
    T3);
IF ¬ABS(C2)<10-10 THEN GOTO
    L4;
T4:=0.5*(T1+T3-C1/C2);
IF (T4-T1)*(T3-T4)≤0 THEN
    GOTO L4;
PENALTY(T4, Y4);
IF ¬ABS(Y2)<1 THEN A:=1 ELSE
    A:=Y2;
IF ¬ABS((Y2-Y4)/A)<EPS1 ∨
   ¬ABS(T2-T1)<10-15 ∨
    ¬ABS(T2-T3)<10-15 THEN
    BEGIN
IF Y2>Y4 THEN GOTO L5
ELSE GOTO L4; END ELSE
IF(T4-T2)*HT>0 THEN BEGIN
IF Y2>Y4 THEN BEGIN
T1:=T2; Y1:=Y2;
T2:=T4;Y2:=Y4; END ELSE BEGIN
T3:=T4; Y3:=Y4; END END ELSE
IF Y2>Y4 THEN BEGIN
T3:=T2; Y3:=Y2;
T2:=T4;Y2:=Y4; END ELSE BEGIN
T1:=T4; Y1:=Y4; END;
GOTO L3;
L4: T:=T2; Y:=Y2; GOTO L6;
L5: T:=T4; Y:=Y4;
L6: END;
PROCEDURE MINIMIZE;
BEGIN
REAL T, F1, F2, F3,DF,DFM,DFI;
ARRAY X3 [1:N];
SDX:=10;
LF: ITER:=ITER+1;
IF SDX>EPS1 THEN BEGIN
YO:=F1:=FO;
DFM:=0; DFI:=1;
FOR I:=1 STEP 1 UNTIL N DO
BEGIN
LINEMIN (HO, T, F2);
FOR K:=1 STEP 1 UNTIL N DO
X(K):=X(K)+T*S(I, K);
FUNCT(X, F2);
DF:=F1-F2; YO:=F1:=F2;
IF DF>DFM THEN BEGIN
DFM:=DF; DFI:=I; END;
END;
FOR K:=1 STEP 1 UNTIL N DO
BEGIN
X3(K):=2*X(K)-XO(K);
S(N+1, K):=X(K)-XO(K);
END;
FUNCT(X3, F3);
SDX:=0;
FOR K:=1 STEP 1 UNTIL N DO
SDX:=SDX+(X(K)-XO(K))*(X(K)
    -XO(K));
IF F3<FO ∧
(FO-2*F2+F3)*(FO-F2-DFM)*(FO
    -F2-DFM)
<0.5*DFM*(FO-F3)*(FO-F3)THEN
BEGIN
FOR I:=DFI STEP 1 UNTIL N DO
FOR K:=1 STEP 1 UNTIL N DO
S(I, K):=S(I+1, K);
YO:=F2;
LINEMIN (HO, T, FO);
FOR K:=1 STEP 1 UNTIL N DO
XO(K):=X(K):=X(K)+T*S(N, K);
FUNCT(X, FO);
END ELSE
IF F3<F2 THEN BEGIN
FO:=F3;
FOR K:=1 STEP 1 UNTIL N DO
```

```
XO(K):=X(K):=X3(K); END
ELSE BEGIN
FOR K:=1 STEP 1 UNTIL N DO
XO(K):=X(K);
FO:=F2; END;
GOTO LF; END;
ITER:=ITER-1;
END;
#PRINT(O,'10',N, FK, GK, HK,
    EPS1, EPS2, C, R);
#PRINT(O, '10', BL, BU, X);
M:=2657863; M35:=2↑35;
M36:=2*M35; M37:=2*M36;
FOM:=10; HO:=0.01; C:=0.2;
R:=R/C;
CYCLE:=ITER:=FUN:=O;
FOR K:=1 STEP 1 UNTIL N DO
XO(K):=X(K);
FOR I:=1 STEP 1 UNTIL N DO
FOR K:=1 STEP 1 UNTIL N DO
S(I,K):=IF I=K THEN 1 ELSE O;
LL: R:=R*C;
CYCLE:=CYCLE+1;
#PRINT(O,'10',CYCLE, R);
LQ: FUNCT(X, FO); C
IF CYCLE=0 THEN BEGIN
FOR I:=1 STEP 1 UNTIL GK DO
IF GX(I)<10-15 THEN BEGIN
FOR K:=1 STEP 1 UNTIL N DO
BEGIN
RANDOM;
X(K):=BL(K)+Q*(BU(K)-BL
    (K))
END;
GOTO LQ; END; END;
MINIMIZE;
#PRINT(O,'10',ITER,FO,SF,X);
IF #ABS((FOM-FO)/FO) < EPS2
    THEN
Q:=#ABS((SFO-SF)/SFO) ELSE
    BEGIN
FOM:=FO; GOTO LL; END;
#PRINT(O,'10',CYCLE,ITER,FUN,
    R, FO, SF, SFO, Q, X, FX,GX,
    HX);
END;
```

3. 形式参数说明：

N 设计变量个数（维数）

FK 分目标函数个数

GK 不等约束函数个数

HK 等约束函数个数

EPS1 二次插值法的收敛精度值

EPS2 求解罚函数的精度值

R 罚因子值，初值宜用控制变量给出

X(1:N) 设计变量的向量数组

BL(1:N) 设计变量的下界数组

BU(1:N) 设计变量的上界数组

FX(1:FK) 目标函数数组

GX(1:GK) 不等约束函数数组

HX(1:HK) 等约束函数数组

FGH(X) 待算数学模型的过程，其表达形式为：

```
PROCEDURE FGH(X);
ARRAY X;
```

```
BEGIN
    ..........
    ..........   } FGH (X) 过程体
END;
```

4. 过程使用说明：

使用本过程计算实际问题时，只需将待算问题的数学模型分别以目标函数数组、不等约束函数数组和等约束函数数组的形式写在数学模型过程 FGH(X) 过程中，再给出与本过程形参所对应的实在参数（初始数据），即可调用本过程求得最优化解。

初始数据的输入均可采用数据纸带输入方式或赋值语句给出的方式。初始罚因子R 值对计算成败和精度是有影响的，因此，宜用控制台变量给出，以便临机时容易修改。

使用本过程时，力求将数学模型编制无误，即目标函数要准确，边界约束条件要实际，性能约束条件要全面，就能得到比较理想的最优设计方案。

关于本过程中标识符的意义及使用中的其它问题，请参考专题报告。

5. 计算实例：

本程序中采用的数学模型是：

求 $F(\bar{X}) = 4X_1 - X_2^2 - 12$ 最小

满足 $h_1(\bar{X}) = 25 - X_1^2 - X_2^2 = 0$

$g_2(\bar{X}) = 10X_1 - X_1^2 + 10X_2 - X_2^2 - 34 \geq 0$

$g_3(\bar{X}) = X_1 \geq 0$

$g_4(\bar{X}) = X_2 \geq 0$

的最优化解 $\bar{X}^{(*)}$ 和最优化函数值 $F(\bar{X}^{(*)})$

计算方案：分别以靠近最优点、远离最优点、非可行区点为不同的初始点 $\bar{X}^{(0)}$，进行三次计算，其计算结果基本相同。

现将三次计算结果列表如下：

项　目	标识符	第一次	第二次	第三次
初始点特征		靠近最优点	远离最优点	非可行区点
$X_1^{(0)}$	X(1)	1.150	3	10
$X_2^{(0)}$	X(2)	4.918	3	10
构型数	CYCLE	7	7	8
迭代数	ITER	18	28	24
$r^{(0)}$	R	1	1	10
$r^{(K)}$	R	0.000064	0.000064	0.000128
$X_1^{(K)}$	X(1)	1.00243	1.00241	1.00293
$X_2^{(K)}$	X(2)	4.89889	4.89990	4.89897
$F(\bar{X}^{(K)})$	FX(1)	-31.98960	-31.90090	-31.98824
$\Phi(\bar{X}^{(K)}, r^{(K)})$	Y	-31.98038	-31.98039	-31.98937

6. 例题计算程序：

```
BEGIN
INTEGER N, FK, GK, HK;
```

```
#READ (O, 'N', N, FK, GK, HK);
BEGIN
REAL EPS1, EPS2, R;
ARRAY X, BL, BU[1:N], FX[1:FK], GX[1:GK], HX[1:HK];
PROCEDURE FGH(X)
ARRAY X;
BEGIN
FX[1]:=4*X[1]-X[2]*X[2]-12;
GX[1]:=10*X[1]-X[1]*X[1]+10*X[2]-X[2]*X[2]-34;
GX[2]:=X[1];
GX[3]:=X[2];
HX[1]:=25-X[1]*X[1]-X[2]*X[2];
END;
.....
                                    } SUMT 过程 全文
.....
EPS1:=10-7; EPS2:=10-3;
R:=HOOOOO;
#READ (O, 'N', X, BL, BU);
SUMT (N, FK, GK, HK, EPS1, EPS2, R, X, BL, BU, FX, GX, HX,
   FGH);
END
END
```

第一段数据　2，1，3，1；

第二段数据　3，3；

第三段数据　0，0；

第四段数据　10，10；

参 考 文 献

〔1〕 李炳威、结构的最优化设计、科学出版社（1979年）、289页。

〔2〕 南京大学数学系、最优化方法、科学出版社（1978年）、185页。

〔3〕 陈立周、机构最优设计方法、北京钢铁学院（1979年）、126页。

〔4〕 王德人、非线性方程组解法与最优化方法、人民教育出版社(1979年)、272页。

〔5〕 南京大学数学系、光学自动设计程序汇编、国防工业出版社(1978年)、108页。

〔6〕 北京工业大学计算站、怎样使用计算机、科学出版社（1976年）、244页。