DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

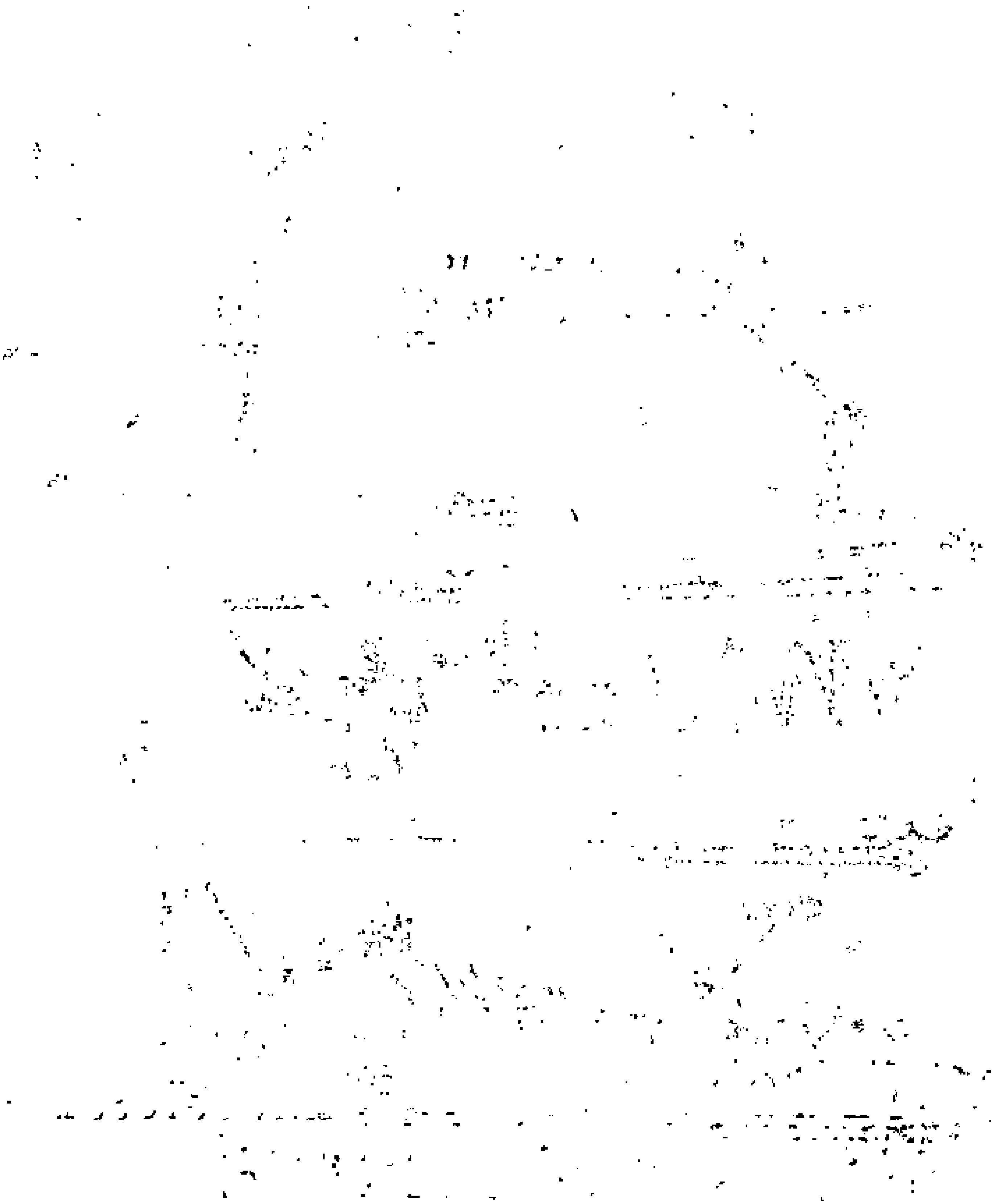
DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

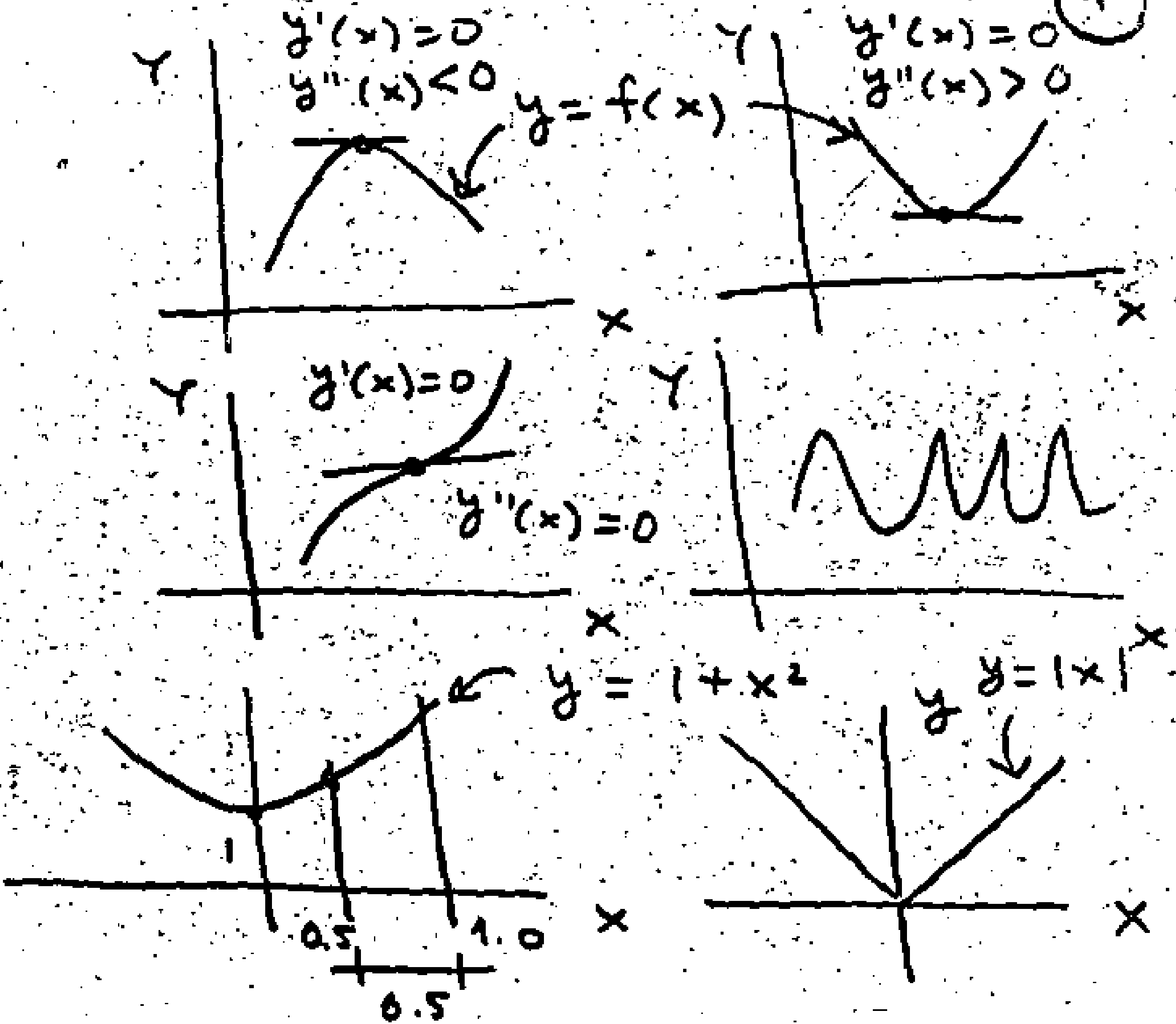
ANTECEDENTES MATEMATICOS Y NUMERICOS DE LAS TECNICAS DE OPTIMIZACION

ANEXO

DR. JORGE ANGELES ALVAREZ

MARZO 1983.





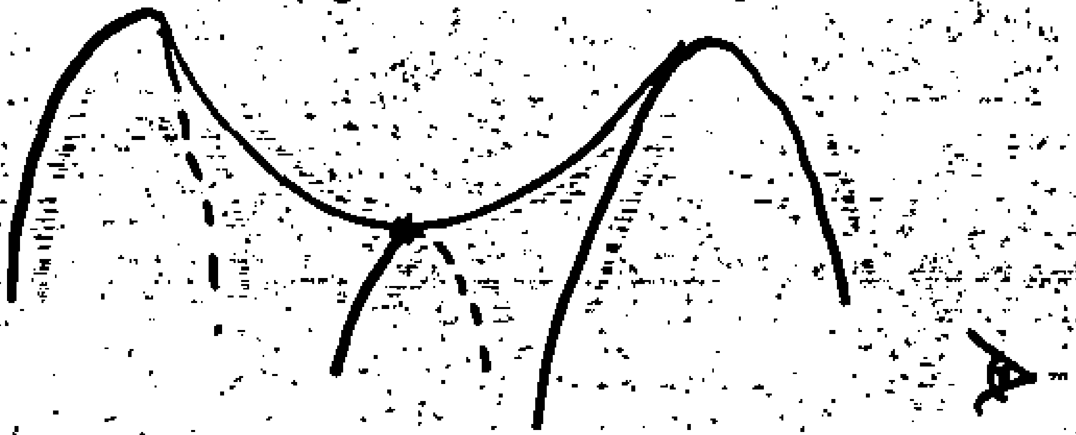
$$y = y(x_1, x_2, \dots, x_n) = y(\underset{\sim}{x})$$

$$\nabla y = \left[\frac{\partial y}{\partial x_1}, \frac{\partial y}{\partial x_2}, \dots, \frac{\partial y}{\partial x_n}\right]^T$$

$$\nabla y = \underset{\sim}{0}$$

$$\nabla\nabla y = \begin{bmatrix} \frac{\partial^2 y}{\partial x_1^2} & \frac{\partial^2 y}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 y}{\partial x_1 \partial x_n} \\ \frac{\partial^2 y}{\partial x_1 \partial x_2} & \frac{\partial^2 y}{\partial x_2^2} & \cdots & \frac{\partial^2 y}{\partial x_2 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 y}{\partial x_1 \partial x_n} & \frac{\partial^2 y}{\partial x_2 \partial x_n} & \cdots & \frac{\partial^2 y}{\partial x_n^2} \end{bmatrix}$$

Punto Silla



Espacio vectorial

Campo: F: conjunto de números,

Si: x_1 y $x_2 \in F$, $x_1 + x_2 \in F$ $(x_1 + x_2 = x_2 + x_1)$

$\exists\, 0 \ni x + 0 = x, \forall x$

$x_1 x_2 \in F$ $(x_1 x_2 = x_2 x_1)$

$\exists\, 1 \ni x \cdot 1 = 1 \cdot x = x, \forall x$

$$x_1(x_2x_3) = (x_1x_2)x_3$$

$$x_1+(x_2+x_3) = (x_1+x_2)+x_3$$

$$x_1(x_2+x_3) = x_1x_2+x_1x_3$$

$$\forall x \quad \exists -x \quad \ni \quad x+(-x)=0$$

$$\forall x \quad \exists x^{-1} \quad \ni \quad xx^{-1}=x^{-1}x=1 \ (x\neq 0)$$

—//—

i) Aditivas

Si $\underset{\sim}{v_1}, \underset{\sim}{v_2}$ y $\underset{\sim}{v_3} \in V$ (esp. vect.)

$\underset{\sim}{v_1}+\underset{\sim}{v_2} = \underset{\sim}{v_2}+\underset{\sim}{v_1} \in V$ (cerradura)

$\exists \ \underset{\sim}{0} \ni \forall \underset{\sim}{v} \in V, \ \underset{\sim}{v}+\underset{\sim}{0} = \underset{\sim}{v}$

ii) Multiplicativas

Si α y $\beta \in F$

$\alpha \underset{\sim}{v} = \underset{\sim}{v}\alpha \in V$

iii) Distributivas

$$(\alpha+\beta)\underset{\sim}{v} = \alpha\underset{\sim}{v}+\beta\underset{\sim}{v}$$

$$(\alpha\beta)\underset{\sim}{v} = \alpha(\beta\underset{\sim}{v})$$

$$\alpha(\underset{\sim}{v_1}+\underset{\sim}{v_2}) = \alpha\underset{\sim}{v_1}+\alpha\underset{\sim}{v_2}$$

$\underset{\sim}{v} = [v_1, v_2, \ldots, v_n]^T, v_i \in \mathbb{R}, \forall i$

$\mathcal{A} = \{\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots, \underset{\sim}{v}_m\}, \underset{\sim}{v}_i \in V$

$\mathcal{B} = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}, \alpha_i \in \mathbb{R}$

$\underset{\sim}{l} = \alpha_1 \underset{\sim}{v}_1 + \alpha_2 \underset{\sim}{v}_2 + \cdots + \alpha_m \underset{\sim}{v}_m$

$\mathcal{A}$ es l.i. si, y sólo si

$\underset{\sim}{l} = \underset{\sim}{0} \iff \alpha_1 = \alpha_2 = \cdots = \alpha_m = 0$

De otra forma, $\mathcal{A}$ es l.d.

Base de un espacio

$\mathcal{B} = \{\underset{\sim}{v}_1, \ldots, \underset{\sim}{v}_n\} \in V$ es una base para V si

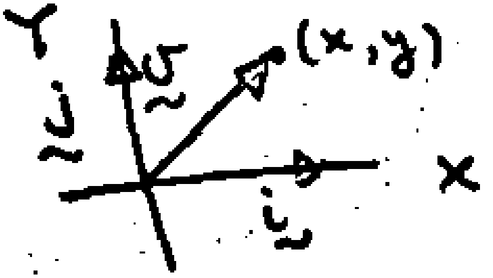
i) $\mathcal{B}$ es l.i.

ii) Cualquier $\underset{\sim}{v} \in V$ se puede expresar como

$$\underset{\sim}{v} = c_1 \underset{\sim}{v}_1 + c_2 \underset{\sim}{v}_2 + \cdots + c_n \underset{\sim}{v}_n$$

$\underset{\sim}{v}_1 = \underset{\sim}{i} + 2\underset{\sim}{j}$, $\underset{\sim}{v}_2 = -\underset{\sim}{i} + \underset{\sim}{j}$

¿ $\{\underset{\sim}{v}_1 , \underset{\sim}{v}_2\}$ es una base para el espacio $E^2(X,Y)$?



$\underset{\sim}{v} = x\,\underset{\sim}{i} + y\,\underset{\sim}{j}$

$$\underset{\sim}{\ell} = \alpha_1 \underset{\sim}{v}_1 + \alpha_2 \underset{\sim}{v}_2 = \alpha_1(\underset{\sim}{i} + 2\underset{\sim}{j}) + \alpha_2(-\underset{\sim}{i} + \underset{\sim}{j})$$

$$= (\alpha_1 - \alpha_2)\underset{\sim}{i} + (2\alpha_1 + \alpha_2)\underset{\sim}{j} = \underset{\sim}{0}$$

$$\begin{cases} \alpha_1 - \alpha_2 = 0 & (1) \\ 2\alpha_1 + \alpha_2 = 0 & (2) \end{cases}$$

$(1) \Rightarrow \alpha_2 = \alpha_1 \quad (3)$

(3) en $(2) \Rightarrow 3\alpha_1 = 0 \Rightarrow \alpha_1 = 0 \Rightarrow \alpha_2 = 0$

$\Rightarrow \{\underset{\sim}{v}_1 , \underset{\sim}{v}_2\}$ es l. i.

¿ Existen números reales c_1 y c_2 $\ni$

$$\underset{\sim}{v} = c_1 \underset{\sim}{v}_1 + c_2 \underset{\sim}{v}_2 \,?$$

$$x\underset{\sim}{i} + y\underset{\sim}{j} = c_1(\underset{\sim}{i} + 2\underset{\sim}{j}) + c_2(-\underset{\sim}{i} + \underset{\sim}{j}) =$$

$$= (c_1 - c_2)\underset{\sim}{i} + (2c_2 + c_1)\underset{\sim}{j}$$

$$c_1 - c_2 = x \qquad (4)$$

$$2c_2 + c_1 = y \qquad (5)$$

(4) ⇒ $c_2 = c_1 - x$ (6)

(6) en (5) ⇒ $2(c_1 - x) + c_1 = y$

⇒ $3c_1 - x = y$ ⇒ $c_1 = \dfrac{x+y}{3}$

⇒ $c_2 = \dfrac{x+y}{3} - x = \dfrac{-2x+y}{3}$

⇒ $\{\underset{\sim}{v}_1$ y $\underset{\sim}{v}_2\}$ <u>sí</u> generan a E^2

⇒ $\{\underset{\sim}{v}_1$ y $\underset{\sim}{v}_2\}$ son <u>una base</u> para E^2

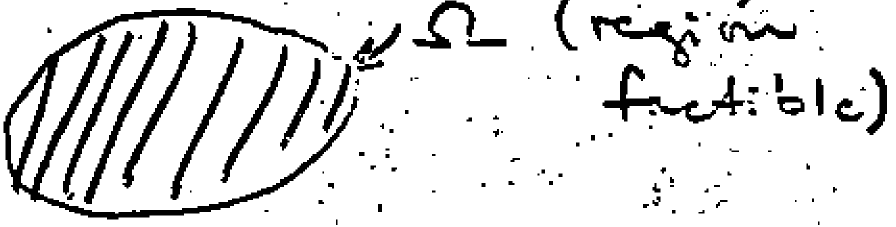
Si $\{\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots, \underset{\sim}{v}_n\}$ es una base para V, $\dim V = n$

En el ejemplo anterior, si $x = 5$, $y = 10$, $c_1 = 5$, $c_2 = 0$

$\underset{\sim}{v} = 5\underset{\sim}{i} + 10\underset{\sim}{j}$ tiene la <u>representación</u> $[5, 10]^T$ en la base $\{\underset{\sim}{i}, \underset{\sim}{j}\}$; pero tiene la representación $[5, 0]^T$ en la base $\{\underset{\sim}{v}_1, \underset{\sim}{v}_2\}$

Un espacio vectorial V

una región Ω de V $(\Omega \subseteq V)$

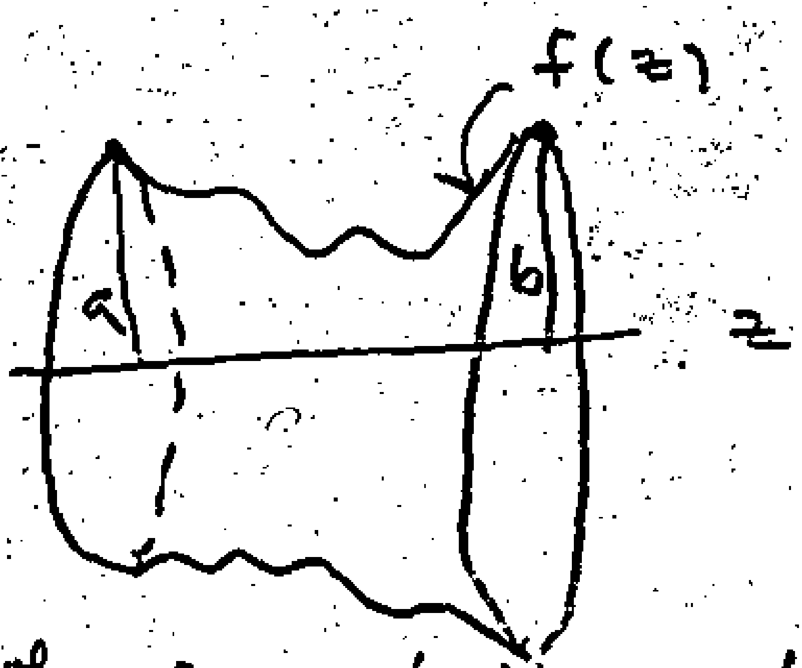


f: un funcional definido sobre Ω

Problema: ~~Hallar~~ el punto ~~[illegible]~~ $\underset{\sim}{x}^* \in \Omega$ donde f alcance un valor óptimo (máximo o mínimo)

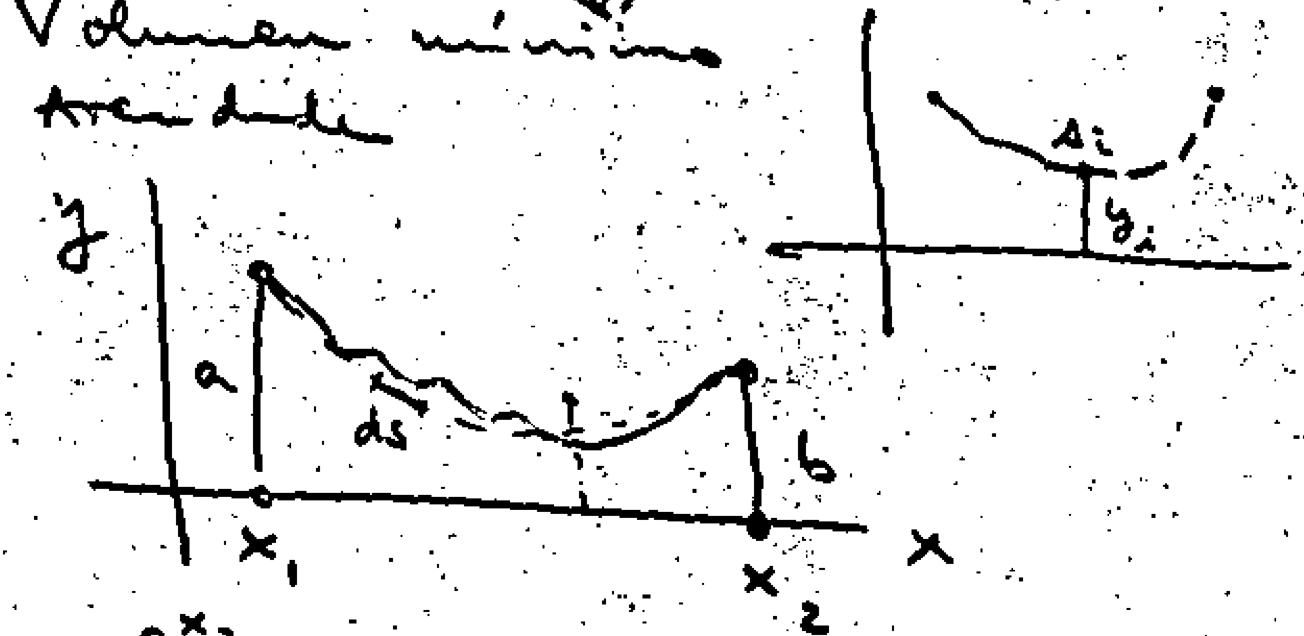
f: función objetivo

V es de dimensión $\begin{cases} \text{finita (algebraica} \\ \text{infinita (cálculo de variaciones)} \end{cases}$



Volumen mínimo

~~Area dada~~



$$V = \int_{x_1}^{x_2} y(x)\, ds \;;\quad ds = \sqrt{dx^2 + dy^2}$$

$$V = \sum_{1}^{n-1} y_i \Delta_i \;,\quad \Delta_i = \sqrt{\Delta x_i^2 + \Delta y_i^2}$$

$$\Delta x_i = x_{i+1} - x_i \,,\; \Delta y_i = y_{i+1} - y_i$$

Transformaciones lineales

U^m, V^n : espacios vectoriales sobre F

$T : U^m \rightarrow V^n$

$\underset{\sim}{u} \in U^m$, $\underset{\sim}{v} \in V^n$, $\underset{\sim}{v} = T(\underset{\sim}{u})$

$\underset{\sim}{v}$ está definido para cada $\underset{\sim}{u}$

T es lineal si

$$T(\alpha_1 \underset{\sim}{u}_1 + \alpha_2 \underset{\sim}{u}_2) = \alpha_1 T(\underset{\sim}{u}_1) + \alpha_2 T(\underset{\sim}{u}_2)$$

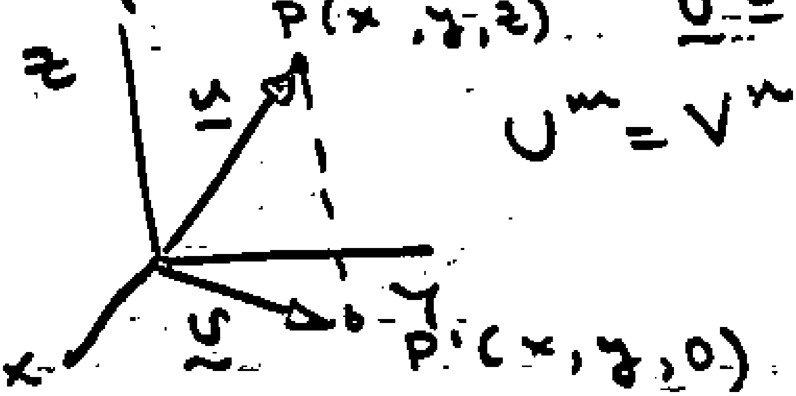
$\alpha_1, \alpha_2 \in F$

Si T es lineal: $\underset{\sim}{v} = \underset{\sim}{T}\, \underset{\sim}{u}$

$\underset{\sim}{T}$ es <u>suprayectiva</u> si para cualquier $\underset{\sim}{v} \in V^n$, existe(n) $\underset{\sim}{u}$(s) $\ni$

$\underset{\sim}{v} = \underset{\sim}{T}\, \underset{\sim}{u}$

<u>Ejemplo</u>:



$\underset{\sim}{v} = T(\underset{\sim}{u})$

$U^m = V^n = E^3$

T es <u>uno a uno</u> si y sólo si

$$\underset{\sim}{u}_1 \neq \underset{\sim}{u}_2 \Rightarrow \underset{\sim}{T}\,\underset{\sim}{u}_1 \neq \underset{\sim}{T}\,\underset{\sim}{u}_2$$

Si $\underset{\sim}{T}$ es suprayectiva y uno a uno, se dice que es invertible:

$$\underset{\sim}{v} = \underset{\sim}{T}\,\underset{\sim}{u} \Rightarrow \underset{\sim}{u} = \underset{\sim}{T}^{-1}\,\underset{\sim}{v}$$

$$B_u = \{\underset{\sim}{u}_1, \underset{\sim}{u}_2, \ldots, \underset{\sim}{u}_m\} \in U^m$$

$$B_v = \{\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots, \underset{\sim}{v}_n\} \in V^n$$

$$\underset{\sim}{T} : U^m \to V^n$$

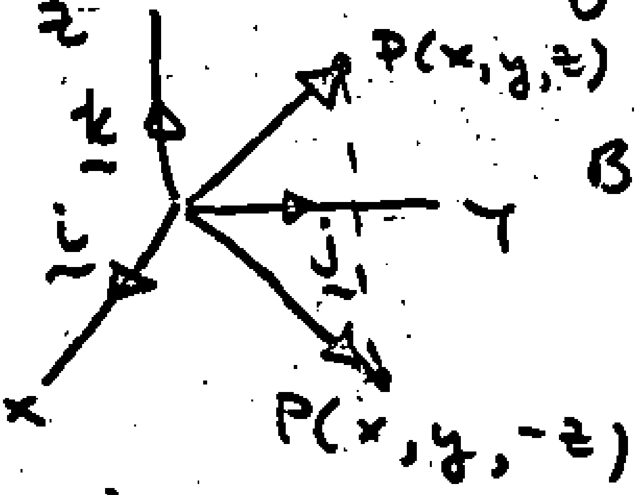
$$\underset{\sim}{T}\,\underset{\sim}{u}_i \in V^n$$

$$\Rightarrow \underset{\sim}{T}\,\underset{\sim}{u}_i = \alpha_{1i}\underset{\sim}{v}_1 + \alpha_{2i}\underset{\sim}{v}_2 + \cdots + \alpha_{ni}\underset{\sim}{v}_n, \quad i = 1, 2, \ldots, m$$

$$\alpha_{1i}, \alpha_{2i}, \ldots, \alpha_{ni} \in F$$

$$[\underset{\sim}{A}] \equiv \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1m} \\ \alpha_{21} & \alpha_{22} & & \alpha_{2m} \\ \vdots & \vdots & & \vdots \\ \alpha_{n1} & \alpha_{n2} & & \alpha_{nm} \end{bmatrix}$$

$U^m = V^n = E^3$



$B_u = B_v = \{\underset{\sim}{i}, \underset{\sim}{j}, \underset{\sim}{k}\}$

$\underset{\sim}{R}\,\underset{\sim}{i} = \underset{\sim}{i}$, $\underset{\sim}{R}\,\underset{\sim}{j} = \underset{\sim}{j}$, $\underset{\sim}{R}\,\underset{\sim}{k} = -\underset{\sim}{k}$

$$[\underset{\sim}{R}] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

Dominio de una T.L.

$\underset{\sim}{T} : U^m \to V^n$; $\underset{\sim}{T}\,\underset{\sim}{u} = \underset{\sim}{v}$

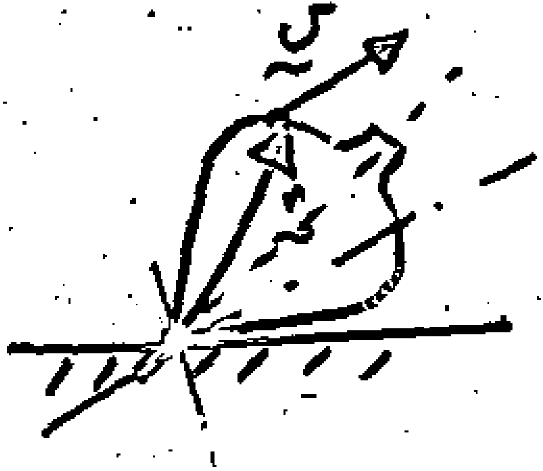
Codominio (range): el conjunto de vectores $\underset{\sim}{v}$ para los cuales existe por lo menos una $\underset{\sim}{u} \ni \underset{\sim}{T}\,\underset{\sim}{u} = \underset{\sim}{v}$

Espacio nulo de $\underset{\sim}{T}$: el conjunto de vectores $\underset{\sim}{u}$ para los cuales

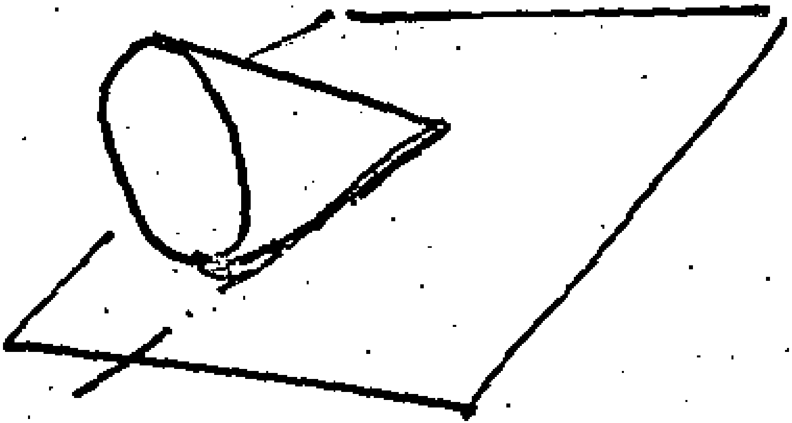
$$\underset{\sim}{T}\,\underset{\sim}{u} = \underset{\sim}{0}$$

Dominio de $\underset{\sim}{T} = U^m$

dim dom = dim cod + dim E.N.



$\underline{v} = \underset{\sim}{T}\,\underline{r}$



Si $\underset{\sim}{T} : U^m \to V^n \ni U^m = V^n$,

$\underset{\sim}{T}$ es un isomorfismo

Dom = Cod

$\underset{\sim}{T}\,\underline{u} = \underline{v} \in U^m$

En general $\underline{u}$ y $\underline{v}$ son l.i.;
pero si sucede que sean l.d.,
esto es, si

$$\underset{\sim}{T}\,\underline{u} = \lambda\,\underline{u}, \quad \lambda \in F \quad (\underline{u} \neq \underline{0})$$

$\underset{\sim}{u}$ es un vector característico de $\underset{\sim}{T}$

λ es un valor característico de $\underset{\sim}{T}$ asociado a $\underset{\sim}{u}$.

$$\underset{\sim}{T}\,\underset{\sim}{u} = \lambda\,\underset{\sim}{u} \Rightarrow (\underset{\sim}{T} - \lambda \underset{\sim}{I})\underset{\sim}{u} = \underset{\sim}{0}$$

$$\Rightarrow \underbrace{\det(\underset{\sim}{T} - \lambda\,\underset{\sim}{I})}_{P_n(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + \dots + a_n\lambda^n} = 0$$

Polinomio característico de $\underset{\sim}{T}$.

Tiene n raíces λ_i, $i = 1, \dots, n$, que son los valores característicos de $\underset{\sim}{T}$.

$\underset{\sim}{H}$ es hermitiana si

$$\underset{\sim}{H} = \underset{\sim}{H}^{+} \text{ (transp. conjugada)}$$

Ejemplo:

$$\underset{\sim}{H} = \begin{bmatrix} 2 & 1+i \\ 1-i & 3 \end{bmatrix} \text{ es hermitiana}$$

Teorema: Los valores car. de una matriz hermitiana son reales y sus vectores característicos son ortogonales.

$$\det(\underset{\sim}{H} - \lambda \underset{\sim}{I}) = \begin{vmatrix} 2-\lambda & 1+i \\ 1-i & 3-\lambda \end{vmatrix} = (2-\lambda)(3-\lambda) - \underbrace{(1-i)(1+i)}_{1-i^2=2} =$$

$$= \lambda^2 - 5\lambda + 6 - 2 = \lambda^2 - 5\lambda + 4 =$$

$$= (\lambda-4)(\lambda-1) \Rightarrow \lambda_1 = 1, \lambda_2 = 4$$

$\lambda_1 = 1$:

$$\begin{bmatrix} 1 & i+1 \\ 1-i & 2 \end{bmatrix} \overset{\underset{\sim}{e}_1}{\begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix}} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$e_{11} + (1+i)e_{21} = 0 \Rightarrow e_{11} = -(1+i)e_{21}$$

$$\|\underset{\sim}{e}_1\|^2 = 1 \Rightarrow \underset{\sim}{e}_1 \underset{\sim}{e}_1^* = 1 = \underset{\sim}{e}_1^* \underset{\sim}{e}_1$$

$$[\tilde{e}_{11}, \tilde{e}_{21}] \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} = |e_{11}|^2 + |e_{21}|^2$$

$$|e_{11}|^2 = [-(1+i)e_{21}][-(1-i)\tilde{e}_{21}] =$$

$$= (2)|e_{21}|^2$$

$$\Rightarrow 2|e_{21}|^2 + |e_{21}|^2 = 3|e_{21}|^2 = 1$$

$$\Rightarrow |e_{21}|^2 = \frac{1}{3} \Rightarrow e_{21} = \frac{1}{\sqrt{3}} = \frac{\sqrt{3}}{3}$$

$$\Rightarrow e_{11} = -(1+i)\frac{\sqrt{3}}{3} \Rightarrow \underset{\sim}{e}_1 = \frac{\sqrt{3}}{3}\begin{bmatrix} -(1+i) \\ 1 \end{bmatrix}$$

$$(\underline{H} - \lambda_i \underline{I})\underset{\sim}{e}_i = \underline{0}$$

$$\begin{bmatrix} -2 & 1+i \\ 1-i & -1 \end{bmatrix} \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-2e_{12} + (1+i)e_{22} = 0 \Rightarrow e_{12} = \frac{1+i}{2} e_{22}$$

$$[\tilde{e}_{12}, \tilde{e}_{22}] \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} = 1$$

$$\Rightarrow |e_{12}|^2 + |e_{22}|^2 = 1 \; ; \; |e_{12}|^2 = \underbrace{\left|\frac{1+i}{2}\right|^2}_{\frac{1}{4}2 = \frac{1}{2}} |e_{22}|^2$$

$$\Rightarrow \frac{1}{2}|e_{22}|^2 + |e_{22}|^2 = 1$$

$$\Rightarrow \frac{3}{2}|e_{22}|^2 = 1 \Rightarrow |e_{22}|^2 = \frac{2}{3} \Rightarrow e_{22} = \sqrt{\frac{2}{3}}$$

$$\Rightarrow \underset{\sim}{e}_2 = \sqrt{\frac{2}{3}} \begin{bmatrix} (1+i)/2 \\ 1 \end{bmatrix}$$

$$\underset{\sim}{e}_1^* \underset{\sim}{e}_2 = \frac{\sqrt{3}}{3} [-1+i, 1] \sqrt{\frac{2}{3}} \begin{bmatrix} (1+i)/2 \\ 1 \end{bmatrix} =$$

$$= \frac{\sqrt{6}}{3\sqrt{3}} \left[\frac{1}{2}(-2) + 1\right] = 0$$

//

$$\underline{L} : U^m \rightarrow V^n = U^m$$

$$B = \{\underset{\sim}{\beta}_1, \underset{\sim}{\beta}_2, \ldots, \underset{\sim}{\beta}_n\} \in U$$

$$\underset{\sim}{v} = \underset{\sim}{L}\,\underset{\sim}{u}$$

$$[\underset{\sim}{L}]_B = \begin{bmatrix} l_{11} & l_{12} & \cdots & l_{1n} \\ l_{21} & l_{22} & & l_{2n} \\ \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & & l_{nn} \end{bmatrix}$$

$$[\underset{\sim}{v}]_B = [v_1, v_2, \ldots, v_n]$$

$$\underset{\sim}{v} = v_1 \underset{\sim}{\beta}_1 + v_2 \underset{\sim}{\beta}_2 + \cdots + v_n \beta_n$$

Dada otra base

$$C = \{\underset{\sim}{\gamma}_1, \underset{\sim}{\gamma}_2, \ldots, \underset{\sim}{\gamma}_n\}$$

$$¿[\underset{\sim}{L}]_C = ?, \quad ¿[\underset{\sim}{v}]_C = ?$$

$$[\underset{\sim}{v}]_C = [v'_1, v'_2, \ldots, v'_n]$$

$$\underset{\sim}{v} = v'_1 \underset{\sim}{\gamma}_1 + v'_2 \underset{\sim}{\gamma}_2 + \cdots + v'_n \underset{\sim}{\gamma}_n$$

$$\underset{\sim}{\gamma}_1 = \alpha_{11} \underset{\sim}{\beta}_1 + \alpha_{21} \underset{\sim}{\beta}_2 + \cdots + \alpha_{n1} \underset{\sim}{\beta}_n$$

$$\underset{\sim}{\gamma}_2 = \alpha_{12} \underset{\sim}{\beta}_1 + \alpha_{22} \underset{\sim}{\beta}_2 + \cdots + \alpha_{n2} \underset{\sim}{\beta}_n$$

$$\vdots$$

$$\underset{\sim}{\gamma}_n = \alpha_{1n} \underset{\sim}{\beta}_1 + \alpha_{2n} \underset{\sim}{\beta}_2 + \cdots + \alpha_{nn} \underset{\sim}{\beta}_n$$

$$[\underline{A}]_B = \begin{bmatrix} \alpha_{11} & \alpha_{12} \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \alpha_{2n} \\ \vdots & \vdots & \vdots \\ \alpha_{n1} & \alpha_{n2} & \alpha_{nn} \end{bmatrix}$$

$$\underset{\sim}{v} = v'_1 \underset{\sim}{\gamma}_1 + v'_2 \underset{\sim}{\gamma}_2 + \cdots + v'_n \underset{\sim}{\gamma}_n =$$

$$= v'_1 \sum_1^n \alpha_{i1} \underset{\sim}{\beta}_i + v'_2 \sum_1^n \alpha_{i2} \underset{\sim}{\beta}_i + \cdots + \underset{\sim}{v}'_n \sum_1^n \alpha_{in} \underset{\sim}{\beta}_i$$

$$= \underbrace{\sum_{j=1}^n v'_j \sum_{i=1}^n \alpha_{ij} \underset{\sim}{\beta}_i}_{\sum_{i=1}^n \sum_{j=1}^n \alpha_{ij} v'_j \underset{\sim}{\beta}_i} = \sum_{i=1}^n v_i \underset{\sim}{\beta}_i$$

$$[\underline{v}]_C = [\underline{A}^{-1}]_B [\underline{v}]_B$$

$$[\underline{v}]_B = [\underline{A}]_B [\underline{v}]_C$$

$$\Uparrow$$

$$v_i = \sum_{j=1}^n \alpha_{ij} v'_j$$

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} & \cdots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \cdots & \alpha_{2n} \\ \vdots & & & \\ \alpha_{n1} & \alpha_{n2} & \cdots & \alpha_{nn} \end{bmatrix} \begin{bmatrix} v'_1 \\ v'_2 \\ \vdots \\ v'_n \end{bmatrix}$$

$$[\underset{\sim}{v}]_B = [\underset{\sim}{A}]_B [\underset{\sim}{v}]_C$$

$$[\underset{\sim}{v}]_C = [\underset{\sim}{A}^{-1}]_B [\underset{\sim}{v}]_B \quad \checkmark$$

$$\underset{\sim}{v} = \underset{\sim}{L}\, \underset{\sim}{u}$$

$$\rightarrow [\underset{\sim}{v}]_B = [\underset{\sim}{L}]_B [\underset{\sim}{u}]_B$$

$$\underbrace{[\underset{\sim}{v}]_C} = [\underset{\sim}{L}]_C \underbrace{[\underset{\sim}{u}]_C}_{} \swarrow [\underset{\sim}{u}]_C$$

$$[\underset{\sim}{A}^{-1}]_B [\underset{\sim}{v}]_B = [\underset{\sim}{L}]_C [\underset{\sim}{A}^{-1}]_B [\underset{\sim}{u}]_B$$

$$[\underset{\sim}{v}]_B = \underbrace{[\underset{\sim}{A}]_B [\underset{\sim}{L}]_C [\underset{\sim}{A}^{-1}]_B}_{[\underset{\sim}{L}]_B} [\underset{\sim}{u}]_B$$

$$[\underset{\sim}{L}]_B = [\underset{\sim}{A}]_B [\underset{\sim}{L}]_C [\underset{\sim}{A}^{-1}]_B$$

$[\underset{\sim}{L}]_C = [\underset{\sim}{A}^{-1}]_B [\underset{\sim}{L}]_B [\underset{\sim}{A}]_B$: Transformación similar

//

$$\underbrace{\underset{\sim}{A}}_{n \times n} \underset{\sim}{e}_i = \lambda_i \underset{\sim}{e}_i \;,\; i = 1, 2, \ldots, n$$

$\underset{\sim}{A}$ es hermitiana

⇒ Tiene n vectores característicos, mutuamente ortogonales.

$$e_i^T e_j = \delta_{ij} \begin{cases} = 1, & i = j \\ = 0, & i \neq j \end{cases}$$

$$Q = [\, e_1 \vdots e_2 \vdots e_3 \vdots \cdots \vdots e_n \,] =$$

$$= \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & & e_{2n} \\ \vdots & \vdots & & \vdots \\ e_{n1} & e_{n2} & & e_{nn} \end{bmatrix}$$

$$A e_1 = \lambda_1 e_1 , \; A e_2 = \lambda_2 e_2 , \; \ldots , \; A e_n = \lambda_n e_n$$

$$A \underbrace{[\, e_1 \vdots e_2 \vdots \cdots \vdots e_n \,]}_{Q} = [\, \lambda_1 e_1 \vdots \lambda_2 e_2 \vdots \cdots \vdots \lambda_n e_n \,]$$

$$= Q \Lambda$$

$$\Lambda = diag\,(\lambda_1, \lambda_2, \ldots, \lambda_n)$$

$$\Rightarrow A Q = Q \Lambda \Rightarrow \Lambda = \underset{\underset{Q^T}{\shortparallel}}{Q^{-1}} A Q$$

$$\Rightarrow \Lambda = Q^T A Q$$

Si $\underline{A}$ y $\underline{B}$ son $n\times n$ y están relacionados por

$$\underline{B} = \underline{P}^{-1}\,\underline{A}\,\underline{P} \quad (\text{T. S.})$$

$$\Rightarrow \quad \underline{B}^{k} = \underline{P}^{-1}\,\underline{A}^{k}\,\underset{\sim}{P}$$

$$\Rightarrow \quad \underline{Q}^{T}\,\underline{A}^{k}\,\underset{\sim}{Q} = \underset{\sim}{\Lambda}^{k}$$

$$\det(\underline{A} - \lambda\underline{I}) = \begin{vmatrix} 1-\lambda & 2 \\ 2 & 4-\lambda \end{vmatrix} = (1-\lambda)(4-\lambda) - 4$$

$$= \lambda^{2} - 5\lambda + 4 - 4 = 0 \Rightarrow$$

$$\lambda^{2} - 5\lambda = 0 \Rightarrow \lambda_1 = 0, \quad \lambda_2 = 5$$

$$\underset{\sim}{e}_1 = \begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix}, \quad \underline{e}_2 = \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix}$$

$$\underline{A}\,\underline{e}_1 = \lambda_1 \underline{e}_1 \Rightarrow (\underline{A} - \lambda_1 \underline{I})\,\underline{e}_1 = \underset{\sim}{0}$$

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}\begin{bmatrix} e_{11} \\ e_{21} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \Rightarrow \begin{array}{l} e_{11} + 2e_{21} = 0 \\ \Rightarrow e_{11} = -2e_{21} \end{array}$$

$$\|\underline{e}_1\|^{2} = 1 \Rightarrow e_{11}^{2} + e_{21}^{2} = 1 \Rightarrow 4e_{21}^{2} + e_{21}^{2} = 1$$

$$\Rightarrow e_{21} = \frac{\sqrt{5}}{5}, \quad e_{11} = -\frac{2}{5}\sqrt{5}$$

$$\begin{bmatrix} -4 & 2 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} e_{12} \\ e_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$-4e_{12} + 2e_{22} = 0 \Rightarrow e_{22} = 2e_{12}$$

$$e_{12}^2 + e_{22}^2 = 1 \Rightarrow 4e_{12}^2 + e_{12}^2 = 1$$

$$\Rightarrow e_{12} = \frac{\sqrt{5}}{5} \Rightarrow e_{22} = \frac{2}{5}\sqrt{5}$$

$$\underset{\sim}{Q} = [\underline{e}_1 \vdots \underline{e}_2] = \frac{\sqrt{5}}{5} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\underline{Q}\,\underline{Q}^T = \frac{1}{5} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix} = \frac{1}{5} \begin{bmatrix} 5 & 0 \\ 0 & 5 \end{bmatrix} = \underline{I}$$

$$\underset{\sim}{\Lambda} = \begin{bmatrix} 0 & 0 \\ 0 & 5 \end{bmatrix} \Rightarrow \underset{\sim}{\Lambda}^{100} = \begin{bmatrix} 0 & 0 \\ 0 & 5^{100} \end{bmatrix}$$

$$\underline{A}^{100} = \underset{\sim}{Q}\,\underset{\sim}{\Lambda}^{100}\,\underline{Q}^T = \frac{1}{5} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 5^{100} \end{bmatrix} \begin{bmatrix} -2 & 1 \\ 1 & 2 \end{bmatrix}$$

Una matiz cuadrada de $n \times n$, $\underline{A}$ es positiva definida (semidefinida) si

$$f = \underline{x}^T \underline{A}\, \underline{x} > 0 \ (\geq 0) \ , \forall \underset{\sim}{x} \neq \underset{\sim}{0}$$

Si $\underline{A} = \underline{A}^* \Rightarrow$ f es real

$$\underline{A} = \begin{bmatrix} \lambda_1 & & & 0 \\ & \lambda_2 & & \\ & & \ddots & \\ 0 & & & \lambda_n \end{bmatrix}$$

$$\underline{x}^T \underline{A}\, \underline{x} = \sum_{1}^{n} \lambda_i x_i^2 > 0 \underset{}{\ (\geq 0)} \iff \lambda_i > 0 \ (\geq) \ , i=1,\dots,n$$

Una matiz es p.d. (p.s.d.) si todos sus valores característicos son positivos (no negativos)

$$f = f(\underline{x}) \in \mathbb{R}$$

Serie de Taylor alrededor de $\underline{x}_0$

$$f(\underset{\sim}{x}) = f(\underset{\sim}{x}_0) + f'(\underset{\sim}{x}_0)^T (\underline{x} - \underset{\sim}{x}_0) +$$
$$+ \frac{1}{2}(\underline{x} - \underset{\sim}{x}_0)^T f''(\underset{\sim}{x}_0)(\underline{x} - \underset{\sim}{x}) + R(\|\underline{x} - \underset{\sim}{x}_0\|^3)$$

Si $\|\underset{\sim}{x}-\underset{\sim}{x}_0\| \to 0$, el término dominante es el lineal. Un punto estacionario de f es aquel en el cual se anula el término lineal $\forall\ \underset{\sim}{x}-\underset{\sim}{x}_0$

$$\underset{\sim}{f}'(\underset{\sim}{x}_0) = \nabla f \Big|_{\underset{\sim}{x}_0}$$

Si es así, se tiene

$$f(\underset{\sim}{x}) = f(\underset{\sim}{x}_0) + \frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{x}_0)^T f''(\underset{\sim}{x}_0)(\underset{\sim}{x}-\underset{\sim}{x}_0) + R \to 0$$

$\underset{\sim}{x}_0$ en el cual $\underset{\sim}{f}'$ se anula es un máximo si $\Delta f = f(\underset{\sim}{x}) - f(\underset{\sim}{x}_0) \leq 0$

$$f(\underset{\sim}{x}) - f(\underset{\sim}{x}_0) \approx \frac{1}{2}(\underset{\sim}{x}-\underset{\sim}{x}_0)^T \underbrace{f''(\underset{\sim}{x}_0)}_{\text{Hessiana}}(\underset{\sim}{x}-\underset{\sim}{x}_0)$$

Para esto, $f''(\underset{\sim}{x}_0) \leq 0$

Si $f'(\underset{\sim}{x}_0)$ se anula y si $f''(\underset{\sim}{x}_0) \geq 0$, $\underset{\sim}{x}_0$ es un mínimo

Ejemplo: $f = \sum_{1}^{n} x_i^4$

$\nabla f = [4x_1^3, 4x_2^3, \ldots, 4x_n^3]^T$

$$\nabla\nabla f = \begin{bmatrix} 12x_1^2 & 0 & \ldots & 0 \\ 0 & 12x_2^2 & \ldots & 0 \\ \vdots & & & \\ 0 & 0 & \ldots & 12x_n^2 \end{bmatrix}$$

$\nabla f = \underset{\sim}{0} \Rightarrow$ SNL: $\underline{f}(\underset{\sim}{x}) = \underset{\sim}{0}$

Ejemplo: $f(\underset{\sim}{x}) = e^{x_1} \operatorname{sen} x_2 \cosh x_3$

$\nabla f = [e^{x_1} \operatorname{sen} x_2 \cosh x_3, e^{x_1} \cos x_2 \cosh x_3, e^{x_1} \operatorname{sen} x_2 \operatorname{senh} x_3]^T$

$\nabla f = 0 \Rightarrow$

$e^{x_1} \operatorname{sen} x_2 \cosh x_3 = 0$

$e^{x_1} \cos x_2 \cosh x_3 = 0$

$e^{x_1} \operatorname{sen} x_2 \operatorname{senh} x_3 = 0$

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

ANEXO

Dr. Jorge Angeles Alvarez

MARZO, 1982

AN OPTIMIZATION MODEL FOR THE PLANNING OF ELECTRICAL DISTRIBUTION NETWORKS

J.M.Cobián M. Ruiz C.
Instituto de Investigaciones Eléctricas
Cuernavaca, Mor.

INTRODUCTION

The problem of structuring a distribution network in space and time is formulated in this paper as a mixed-integer programming optimization model. The approach also allows for the solution of the problem of the optimal location of distribution substations and their capacity expansion, as well as the related configuration of the corresponding primary-feeder network. This same model can be used also to solve the problem of the optimal design in space and time of secondary networks with their corresponding distribution transformers. The optimization consists of minimizing the present worth of the costs of capital investment and energy losses in the network for the planning horizon to be considered.

Among the advantages of the approach taken here to formulate this problem, as compared to other approaches in the literature, are, that first of all, it considers basically the actual network and not an idealization of it. It also includes the dynamic aspects of the capacity expansion of the network in time, and is able to handle large-scale networks with a relatively small number of integer (binary) variables and automatically solves the important problem of obtaining the optimal distribution of power flows for a given configuration of a network. The application of this model can also help electrical utilities to evaluate the impact of changes in their cost components and in the existing normalization of their equipment in the network.(1,4,6)

The solution algorithm to the problem formulated above is a specialized branch and bound search that takes advantage of the particular structure of this problem to find the optimal solution more efficiently. The corresponding linear programming sub-problems are also of a particular type known as "transshipment" or "distribution" problems which can be solved very rapidly, making the overall algorithm very effective.

FORMULATION OF THE PROBLEM

NOMENCLATURE

T number of stages of the planning horizon.

NP number of proposed expansion projects.

D_t peak demand of the entire system at time t.

N {q/q node index of the corresponding circulatory network}.

RN {(i,j,m,p) / (i,j,m,p) directed arc from node i to j in the circulatory network corresponding to a segment m of the linearized power loss function, with an index p indicating whether or not the arc is been modified by the acceptance of project p}.

PN {(i,j,m,p) / (i,j,m,p) directed arc from node i to j in the circulatory network corresponding to a segment m of the linearized power loss function belonging to project p}.

OUT_q {(i,j,m,p) / (i,j,m,p) ϵRN and i=q}.

$POUT_q$ {(i,j,m,p) / (i,j,m,p) ϵPN and i=q}.

IN_q $\{(i,j,m,p) / (i,j,m,p)$ εRN and $j=q\}$.

PIN_q — $\{(i,j,m,p) / (i,j,m,p)$ εPN and $j=q\}$.

x_{ijmpt} continuous decision variable representing the power flow of arc (i,j,m,p) εRN at time stage t.

w_{ijmpt} continuous decision variable representing the power flow of arc (i,j,m,p) εPN at time stage t.

y_{pt} binary decision variable indicating whether project p is in operation at time stage t.

A_{ijmpt} unit cost at present value associated to arc (i,j,m,p) εRN.

C_{pt} capital cost at present value of the required investment to start project p at time t.

K_{ijmt} capacity of the directed arc (i,j,m,p) εRN (If $j=n+1$ the capacity is equal to the peak demand at load node i at stage t. If $j=0$ the capacity is equal to the total demand D_t of the network at time t).

KK_{ijmp} capacity of the arc (i,j,m,p) εPN.

NE number of restrictions of logical and/or technical dependence between the different projects proposed.

$f_\ell(\cdot)$ functional relationship between the binary variables in the logical and/or technical dependence restrictions.

B_{ijmpt} unit cost at present value associated to arc (i,j,m,p) εPN.

MATHEMATICAL MODEL

It is assumed that: $\ell = 1,\ldots,NE$; $p=1,\ldots,NP$; $t=1,\ldots T$.

$$\text{Minimize } z = \sum_{t=1}^{T} \{ \sum_{(i,j,m,p)\ \varepsilon RN} A_{ijmpt}\, x_{ijmpt} + \sum_{(i,j,m,p)\ \varepsilon PN} B_{ijmpt}\, w_{ijmpt} + \sum_{p=1}^{NP} C_{pt}\,(y_{pt}-y_{pt-1})\} \quad (1)$$

$$\text{subject to} \quad \sum_{(i,j,m,p)\varepsilon OUT_q} x_{ijmpt} + \sum_{(i,j,m,p)\ \varepsilon POUT_q} w_{ijmpt} - \sum_{(i,j,m,p)\ \varepsilon IN_q} x_{ijmpt} - \sum_{(i,j,m,p)\ \varepsilon PIN_q} w_{ijmpt} = 0; q\varepsilon N \quad (2)$$

$$D_t \leqq x_{n+1,0,0,0,t} \leqq K_{n+1,0,0,0,t} \quad (3)$$

$$0 \leqq x_{ijm0t} \leqq K_{ijmt} \ ; \ (i,j,m,0)\ \varepsilon RN \quad (4)$$

$$0 \leqq x_{ijmpt} \leqq K_{ijmt}\ (1-y_{pt}) \ ; \ (i,j,m,p)\ \varepsilon RN \quad (5)$$

$$0 \leqq w_{ijmpt} \leqq KK_{ijmp}\, y_{pt} \quad ; \ (i,j,m,p)\ \varepsilon PN \quad (6)$$

$$y_{p0} = 0 \quad (7)$$

$$y_{pt} - y_{pt-1} \leqq 0 \quad (8)$$

$$y_{pT+1} = 1 \quad (9)$$

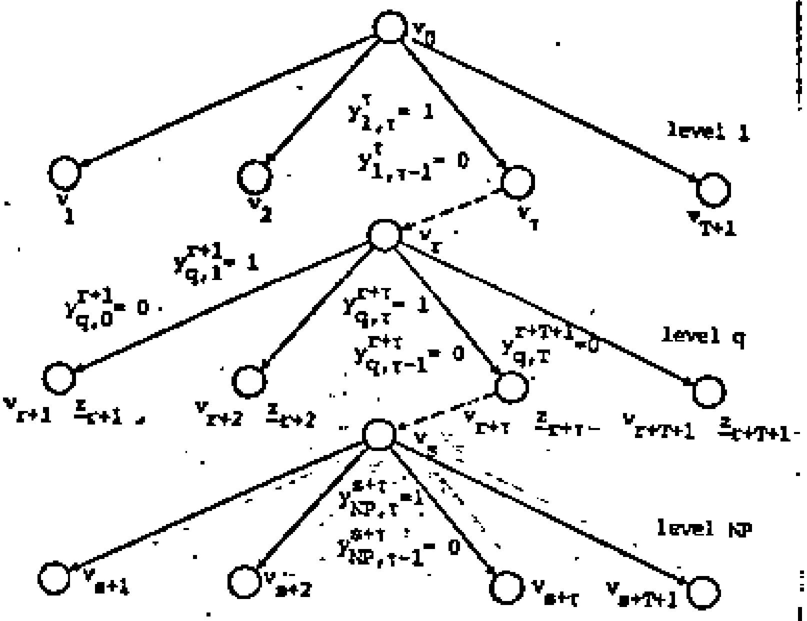
$$f_\ell \ (y_{1,1},\ldots,y_{NP,T}) = 0 \quad (10)$$

$$y_{pt} = 0 \text{ or } 1 \quad (11)$$

Satzspiegelbegrenzung nicht überschreiten.

SOLUTION ALGORITHM

A branch and bound search is conducted where to each level of the search tree is associated a project q. Any vertex $v_{r+\tau}$ at level q in the tree fixes all binary variables corresponding to projects with indices less than or equal to q. Each of the T+1 successor vertices of vertex v_r considers that project q becomes operational at stage τ for τ=1,..., T+1 (stage T+1 is an artificial stage to indicate that project q never becomes operational) (3). At each vertex a lower bound is obtained by solving a relaxation of the original restricted problem which consists of setting to zero the capital costs of the free projects and fixing their corresponding binary variables to one or zero, or both simultaneously, to permit a maximum relaxation of the power flow restrictions. The structure of the resulting mathematical problem is the one known as the transshipment or distribution problem (2,5). The objective function values of the relaxation for different values of τ are as follows:



$$\underline{z}_{r+\tau} = \sum_{t=1}^{\tau-1} \min \zeta^{a}_{rt} + \sum_{t=\tau}^{T} \min \zeta^{b}_{rt} + \sum_{t=1}^{T} \sum_{p=1}^{q} C_{pt} \left(y^{r+\tau}_{pt} - y^{r+\tau}_{pt-1}\right) \qquad (12)$$

for t=1,...,T+1, considering that $\sum_{t=1}^{0} \min \zeta^{a}_{rt}=0$ and $\sum_{t=T+1}^{T} \min \zeta^{b}_{rt}= 0$

where min ζ^{a}_{rt} and min ζ^{b}_{rt} are defined as follows:

$$\text{Minimize} \quad \zeta^{a}_{rt} = \sum_{(i,j,m,p)\,\epsilon RN} A_{ijmpt}\, x_{ijmpt} + \sum_{(i,j,m,p)\,\epsilon PN} B_{ijmpt}\, w_{ijmpt} \qquad (13)$$

$$\text{subject to} \quad \sum_{(i,j,m,p)\,\epsilon OUT_q} x_{ijmpt} + \sum_{(i,j,m,p)\epsilon POUT_q} w_{ijmpt} - \sum_{(i,j,m,p)\,\epsilon IN_q} x_{ijmpt} - \sum_{(i,j,m,p)\epsilon PIN_q} w_{ijmpt} = 0 \; ; q\epsilon N \qquad (14)$$

$$\underline{D}_t \le x_{n+1,0,0,0,t} \le K_{n+1,0,0,0,t} \qquad (15)$$

5. (Upper bound updating). If $q_s \neq NP$ then set $q=q_s+1$, $r=s$, and go to 2. Otherwise, set $\bar{z} = \underline{z}_s$; store y^s_{pt} ∀ p,t as the best current assignment.

6. (Fathoming by bounds). Any live vertices v_u for which $\underline{z}_u \geq \bar{z}$ are fathomed.

7. (Backtracking). If there are no live vertices go to 8. Otherwise, choose vertex v_r where r is such that:

$\underline{z}_r = \min \{ \underline{z}_u ; u/v_u$ is a live vertex $\}$; set $q=q_r+1$; go to 2.

8. (Termination). If $z=\infty$ the problem has no feasible solution. If $\bar{z} < \infty$ the best current assignment is the optimal assignment.

COMPUTATIONAL EXPERIENCE

An application problem of the above formulation and solution procedure was considered and solved in a VAX 11/780 computer in an approximate total time of two minutes. The problem included the consideration of approximately 50 load nodes and 100 existing arcs, with 5 proposed projects in two-time stages which incorporated a total of 20 different arcs. The corresponding mathematical problem had approximately 900 continuous variables and 10 binary variables.

REFERENCES

1. Adams, R.N. and M.A. Laughton (1974),"Optimal Planning of Power Networks Using Mixed-Integer Programming - Part 1 - Static and Time - Phased Network Synthesis", Proc. IEE, Vol. 121, No. 2, 139-147.

2. Bradley G.H., G.G. Brown and G.W. Graves (1974), "Design and Implementation of Large Scale Primal Transshipment Algorithms", Management Science, Vol. 20, 1-34.

3. Garfinkel, R.S. and G.L. Nemhauser (1972), Integer Programming, Wiley Interscience.

4. Masud, E. (1974), "An Interactive Procedure for Sizing and Timing Distribution Substations Using Optimization Techniques", IEEE Trans.PAS, Vol. PAS-93, No. 5, 1281-1286.

5. Simonnard, M. (1966), Linear Programming, Prentice-Hall.

6. Thompson, G.L. and D.L. Wall (1980), "A Branch and Bound Model for Choosing Optimal Substation Locations", Presented at the IEEE-PES Winter Meeting, New York, NY, February 3-8, 1980.

$0 \leq x_{ijm0t} \leq K_{ijmt}$; (i,j,m,0) εRN. (16)

$0 \leq x_{ijmpt} \leq K_{ijmt}\,(1-y^{r}_{pt})$; (i,j,m,p) εRN ; p<q . (17)

$0 \leq x_{ijmqt} \leq K_{ijmt}$; (i,j,m,q) εRN. (18)

$0 \leq x_{ijmpt} \leq K_{ijmt}$; (i,j,m,p) εRN ; p>q. (19)

$0 \leq w_{ijmpt} \leq KK_{ijmp}\, y^{r}_{pt}$; (i,j,m,p) εPN ; p<q. (20)

$w_{ijmqt} = 0$; (i,j,m,q) εPN. (21)

$0 \leq w_{ijmpt} \leq KK_{ijmp}$; (i,j,m,p) εPN ; p>q. (22)

The problem ζ^{b}_{rt} is equivalent to the ζ^{a}_{rt} with the exceptions that set (18) changes to: $x_{ijmqt}=0$; (i,j,m,q) εRN. (23)

and the set (21) changes to: $0\leq w_{ijmqt} \leq KK_{ijmq}$; (i,j,m,q) εPN. (24)

At the last level of the search the relaxation is non-existing, so if it is possible to reach this level without violating restrictions $f_\ell(\cdot)$ and if $\underline{z}_{s+\tau}$ exists (see figure), a feasible solution is obtained to the original problem, which provides an upper bound for the search procedure. The solution algorithm is thus as follows:

1. (Initialization). Set $\bar{z} = \infty$, $q=1$ and $r=0$.

2. (Feasibility phase). For $\tau=1,\dots,T+1$ test the feasibility of $f_\ell(\cdot)=0$ ($\ell=1,\dots,NE$), assuming that: $y^{r+\tau}_{qt}=0$ for $0\leq t\leq \tau-1$, $y^{r+\tau}_{qt}=1$ for $\tau\leq t\leq T+1$, and $y_{pt} = y^{r+\tau}_{pt}$ ∀t and $p\leq q$. If for some τ there is no feasibility, then $\overline{z}_{r+\tau} = \infty$.

3. (Calculation of lower bounds). (a) For $t=1,\dots,T$ calculate min ζ^{a}_{rt} and min ζ^{b}_{rt}. If min ζ^{a}_{rt} does not exist , then let min $\zeta^{a}_{rt}=\infty$. If min ζ^{b}_{rt} does not exist , then let min $\zeta^{b}_{rt} = \infty$.

(b) For $\tau=1,\dots,T+1$ calculate $\underline{z}_{r+\tau}$, except when $\underline{z}_{r+\tau}=\infty$. If $\underline{z}_{r+\tau} < \bar{z}$ then the live vertex $v_{r+\tau}$ is generated; set: $y^{r+\tau}_{qt} = 0$ for $0\leq t\leq \tau-1$, $y^{r+\tau}_{qt}=1$ for $\tau\leq t\leq T+1$, $y^{r+\tau}_{pt} = y^{r}_{pt}$ ∀t and p<q and $q_{r+\tau} = q$; store : $q_{r+\tau}$, $\underline{z}_{r+\tau}$ and $y^{r+\tau}_{pt}$ ∀t and $p\leq q$. Otherwise, continue.

4. (Branching) If no live vertex was generated in step 3, go to 7. Otherwise, choose vertex v_s, where s is such that $\underline{z}_s = \min_{\tau/\tau=1,\dots,T+1} \{\underline{z}_{r+\tau}\}$

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

METODOS DE OPTIMACION CON RESTRICCIONES

Dra Susana Gómez Gómez

Marzo, 1982

AERO-ASTRONAUTICS REPORT NO. 55

GRADIENT METHODS IN MATHEMATICAL PROGRAMMING
PART 1 - REVIEW OF PREVIOUS TECHNIQUES

by

A. MIELE, H.Y. HUANG, and J.W. CANTRELL

RICE UNIVERSITY

1969

Gradient Methods in Mathematical Programming

Part 1 - Review of Previous Techniques[1]

by

A. MIELE[2], H.Y. HUANG[3], AND J.W. CANTRELL[4],

Abstract. This report is the first of a series on gradient methods in mathematical programming.[5] It considers the problem of minimizing a function f(x), where f is a scalar function and x is an n-vector whose components are unconstrained. For this problem, three previous methods are reviewed, namely, the ordinary gradient method, the conjugate-gradient method, and the variable-metric method. A new intuitive derivation of the last two algorithms is presented.

[1] This research was supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67.

[2] Professor of Astronautics and Director of the Aero-Astronautics Group, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[3] Assistant Professor of Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

1. Definitions

The following definitions are used throughout the paper:

(a) The symbol x denotes the position vector

$$x = \begin{bmatrix} x^1 \\ x^2 \\ \cdot \\ \cdot \\ \cdot \\ x^n \end{bmatrix} \qquad (1)$$

whose scalar components are $x^1, x^2, \ldots, x^n$.

(b) The symbol f denotes a scalar function of the vector x, that is

$$f = f(x) \qquad (2)$$

(c) The symbol g denotes the column vector

$$g(x) = \begin{bmatrix} \partial f/\partial x^1 \\ \partial f/\partial x^2 \\ \cdot \\ \cdot \\ \cdot \\ \partial f/\partial x^n \end{bmatrix} \qquad (3)$$

whose components are the first partial derivatives of f with respect to the scalar variables $x^1, x^2, \ldots, x^n$. This is the gradient of the function f.

(d) The symbol H denotes the square matrix

$$H(x) = \begin{bmatrix} \partial^2 f/\partial x^1 \partial x^1 & \partial^2 f/\partial x^1 \partial x^2 & \cdots\cdots & \partial^2 f/\partial x^1 \partial x^n \\ \partial^2 f/\partial x^2 \partial x^1 & \partial^2 f/\partial x^2 \partial x^2 & \cdots\cdots & \partial^2 f/\partial x^2 \partial x^n \\ \cdots & \cdots & \cdots\cdots & \cdots \\ \partial^2 f/\partial x^n \partial x^1 & \partial^2 f/\partial x^n \partial x^2 & \cdots\cdots & \partial^2 f/\partial x^n \partial x^n \end{bmatrix} \qquad (4)$$

whose components are the second partial derivatives of the function f with respect to the scalar variables $x^1, x^2, \ldots, x^n$.

(e) The symbol x denotes the nominal point. The symbol $\tilde{x}$ denotes the point following x. The symbol $\hat{x}$ denotes the point preceding x.

(f) The symbol δ(...) denotes the displacement leading from a point to the next point. Therefore, the following relations hold:

$$\begin{aligned} \tilde{x} &= x + \delta x \\ x &= \hat{x} + \delta\hat{x} \end{aligned} \qquad (5)$$

(g) The superscript T denotes the transpose of a matrix.

2. Introduction

A basic problem of mathematical programming is that of finding the minimum of a function

$$f = f(x) \tag{6}$$

where f is a scalar function and x is an n-vector. If the n components of the vector x are unconstrained, the extremum of (6) occurs when the following necessary condition is satisfied:

$$g(x) = 0 \tag{7}$$

where g is the gradient of the function f with respect to the vector x. For a minimum, the matrix of the second derivatives (4) must be positive definite at the point x defined by (7).

If the function (6) is quadratic, the gradient g(x) is linear with respect to x. Hence, Eq. (7) can be solved analytically. On the other hand, if (6) is nonquadratic, the gradient g(x) is nonlinear. This being the case, approximate methods must be employed to solve Eq. (7). One possible method consists of quasilinearizing (7) about a nominal point. Another method, the descent method, consists of constructing corrections Δx leading from a nominal point x to a varied point $\tilde{x}$ such that

$$f(\tilde{x}) < f(x) \tag{8}$$

Thus, by an iterative procedure (that is, through successive decreases in the value of the function f), it is hoped that the minimum of f is approached to any desired degree of accuracy.

This report is the first of a series on gradient methods in mathematical programming. It reviews three of the existing techniques, namely, the ordinary gradient method, the conjugate-gradient method (Refs. 1-3), and the variable-metric method (Refs. 4-5). For the last two methods, a new intuitive derivation is presented.

3. Ordinary Gradient Method

To first-order terms, the values of the function (6) at the varied point and the nominal point are related by

$$I(\tilde{x}) \cong I(x) + \delta I(x) \tag{9}$$

where the first variation $\delta I(x)$ is given by

$$\delta I(x) = g^T(x)\delta x \tag{10}$$

with

$$\delta x = \tilde{x} - x \tag{11}$$

Also to first-order terms, the greatest decrease in the value of the function is achieved if the first variation (10) is minimized. Here, we limit our analysis to those variations δx which satisfy the constraint

$$K = \delta x^T \delta x \tag{12}$$

where K is a prescribed quantity.

3.1. Derivation of the Algorithm. Standard methods of the theory of maxima and minima show that the fundamental function of this problem is the scalar function

$$F = g^T(x)\delta x + (1/2\alpha)\delta x^T \delta x \tag{13}$$

where $1/2\alpha$ is a constant Lagrange multiplier. The optimum system of variations must be such that

$$G(\delta x) = 0 \tag{14}$$

where G is the gradient of the function F with respect to the scalar variables $\delta x^1, \delta x^2, \ldots, \delta x^n$. In the light of (13), the explicit form of (14) is the following:

$$\delta x = -\alpha g(x) \tag{15}$$

and shows that the optimum correction δx has the gradient direction. This is why the method is called the ordinary gradient method. Upon substituting (15) into (12), we see that

$$K = \alpha^2 g^T(x)g(x) \tag{16}$$

Therefore, a one-to-one correspondence exists between the value of the constant K and the value of α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

3.2. Descent Property. Upon combining Eqs. (10) and (15), we see that the first variation becomes

$$\delta I(x) = -\alpha g^T(x)g(x) \tag{17}$$

and is negative for $\alpha > 0$. Therefore, if α is sufficiently small, the function I decreases. This guaranteed decrease of I at every step is the most important property of the gradient method.

4. Modifications of the Ordinary Gradient Method

The ordinary gradient method is conceptually simple and stable in that the function $f(x)$ is reduced at every iteration; however, it has the drawback of slow convergence. For this reason, methods have been developed to reduce the number of iterations required for convergence. In this connection, let the displacement vector Δx be written in the form

$$\Delta x = -\alpha p \tag{23}$$

where p is the search direction. The following are particular forms of the vector p:

$$p = g(x) + q \tag{24}$$

and

$$p = Ag(x) \tag{25}$$

where q is an n-vector and A is an n x n symmetric matrix. In Section 5, the conjugate-gradient algorithm is derived by reasoning on (24); in Section 6, the variable-metric algorithm is derived by reasoning on (25).

5. Conjugate-Gradient Method

In this section, we consider the algorithm

$$\tilde{x} = x + \Delta x, \qquad \Delta x = -\alpha p, \qquad p = g(x) + q \tag{26}$$

where q is an n-vector to be specified. The first variation of the function (6) is given by Eq. (10) which, in the light of (26), becomes

$$\delta f(x) = -\alpha[g^T(x)g(x) + g^T(x)q] \tag{27}$$

We note that $g^T(x)g(x) > 0$. Therefore, for $\alpha > 0$, the descent property of this algorithm is ensured if one chooses q so that

$$g^T(x)q = 0 \tag{28}$$

If Eqs. (26-1) and (26-2) are combined, the position vector at the end of any iteration becomes

$$\tilde{x} = x - \alpha p \tag{29}$$

For a given point x and a given vector p, Eq. (29) defines a one-parameter family of points $\tilde{x}$ for which the function f takes the form

$$f(\tilde{x}) = f(x - \alpha p) = F(\alpha) \tag{30}$$

The greatest decrease in the function $F(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$F_\alpha = 0 \tag{31}$$

On account [illegible]), the following relation holds:

$$F_\alpha = -g^T(\tilde{x})p \tag{32}$$

Therefore, Eq. (31) becomes

$$g^T(\tilde{x})p = 0 \tag{33}$$

and shows that the gradient $g(\tilde{x})$ is orthogonal to the search direction p.

Next, we apply Eq. (33) to the previous iteration and obtain

$$g^T(x)\hat{p} = 0 \tag{34}$$

By comparing (28) and (34), we conclude that one possible choice of the vector q is the following:

$$q = \gamma\hat{p} \tag{35}$$

where γ is a constant. As a consequence, the algorithm (28) can be rewritten as

$$\tilde{x} = x + \Delta x\ , \quad \Delta x = -\alpha p\ , \quad p = g(x) + \gamma\hat{p} \tag{36}$$

The next step is to determine the constant γ. If Eqs. (36) are combined, the position vector at the end of any iteration becomes

$$\tilde{x} = x - \alpha g(x) - \alpha\gamma\hat{p} \tag{37}$$

For a given point x and a given vector $\hat{p}$, Eq. (37) defines a two-parameter family of points $\tilde{x}$ for which the function f takes the form

$$f(\tilde{x}) = f(x - \alpha g(x) - \alpha\gamma\hat{p}) = F(\alpha, \gamma) \tag{38}$$

The greatest decrease in the function $F(\alpha, \gamma)$ occurs if the parameters [illegible] satisfy the following necessary conditions:

$$F_\alpha = 0\ , \quad F_\gamma = 0$$

On account of (36-3) and (38), the following relations hold:

$$F_\alpha = -g^T(\tilde{x})p\ , \quad F_\gamma = -\alpha g^T(\tilde{x})\hat{p}$$

Therefore, Eqs. (39) become

$$g^T(\tilde{x})p = 0\ , \quad g^T(\tilde{x})\hat{p} = 0 \quad\quad 1)$$

and show that the gradient $g(\tilde{x})$ is orthogonal to the search directions p and $\hat{p}$. A mathematical consequence of Eqs. (36-3) and (41) is that

$$g^T(\tilde{x})g(x) = 0$$

showing that the gradients $g(\tilde{x})$ and $g(x)$ are orthogonal.

5.1. Quadratic Function. Now, consider the particular case of a quadratic function, that is, a function of the form

$$f(x) = a + b^Tx + \frac{1}{2}x^THx \tag{43}$$

where a is a constant scalar, b is a constant n-vector, and H is a constant, symmetric $n \times n$ matrix. For this function, the gradient is a linear function of x, that is,

$$g(x) = b + Hx \tag{44}$$

Since

$$g(\tilde{x}) = b + H\tilde{x} \tag{45}$$

relations (44) and (45) imply that

$$g(\tilde{x}) = g(x) + H\Delta x = g(x) - \alpha Hp \tag{46}$$

Next, we introduce Eqs. (46) into (41) and, after laborious manipulations, obtain the solutions (Ref. 1)

$$\alpha = \frac{g^T(x)g(x)}{p^T Hp} \quad , \qquad \gamma = \frac{g^T(x)g(x)}{g^T(\hat{x})g(\hat{x})} \tag{47}$$

where p is given by (36-3).

For a quadratic function, Hestenes and Stiefel (Ref. 1) proved that, if the first step of the descent process is a gradient step, the following relations hold:

$$g^T(x)g(x_*) = 0 \quad , \quad g^T(x)p_* = 0 \quad , \quad p^T Hp_* = 0 \tag{48}$$

where x_* denotes any state preceding x. Equation (48-1) states that the gradient at each iteration is orthogonal to the gradient at every previous iteration. Equation (48-2) states that the gradient at each iteration is orthogonal to the search direction at every previous iteration. Finally, Eq. (48-3) states that the search direction at each iteration and the search direction at every previous iteration are conjugate with respect to the constant matrix H; this is why the algorithm is called the conjugate-gradient method. The algorithm (36) with α and γ defined by (47) reduces the gradient to zero in no more than n steps; therefore, the minimum of f(x) is reached in no more than n steps.

5.2. Nonquadratic Function. For a nonquadratic function, solving Eqs. (41) for α and γ requires a two-dimensional search (Ref. 6). The difficulty of this process can be avoided if one optimizes α exactly and uses an approximate value for γ, namely, that given by Eq. (47-2). This leads to the algorithm (Ref. 3)

$$\tilde{x} = x + \Delta x, \quad \Delta x = -\alpha p \quad , \quad p = g(x) + \frac{g^T(x)g(x)}{g^T(\hat{x})g(\hat{x})}\hat{p} \tag{49}$$

in which α is optimized by searching for the minimum of f along the direction defined by (49). Theoretically, therefore, the optimization of α requires that the relation (44-1) be satisfied.

For any iteration except the first, the complete algorithm can be stated as follows: (a) for a given nominal point x, the gradient g(x) is known; since the gradient $g(\hat{x})$ and the search direction $\hat{p}$ are known from the previous iteration, the search direction p can be determined with Eq. (49-3); (b) the optimum stepsize α must be determined by minimizing the function f along the search direction p, as in Section 7; (c) the correction Δx to the position vector x is determined using Eq. (49-2); and (d) the new position vector $\tilde{x}$ is computed through Eq. (49-1). Next, the position vector $\tilde{x}$ becomes the nominal point for the subsequent iteration, and the procedure is repeated until a predetermined stopping condition is satisfied (see Section 8). To start the algorithm, one bypasses (49-3) and sets p = g(x), equivalent to stating that the first step is a gradient step.

In closing, the following comments are pertinent: (a) in the conjugate-gradient method it is important that the stepsize α be determined accurately, while this is not the case with the ordinary gradient method; (b) theoretical considerations and numerical experience show the desirability of restarting the process every n or n + 1 iterations, that is, resetting p = g(x) every n or n + 1 iterations (Ref. 3).

6. Variable-Metric Algorithm

In this section, we consider the algorithm

$$\hat{x} = x + \Delta x, \qquad \Delta x = -\alpha p, \qquad p = Ag(x) \tag{50}$$

where A is a symmetric n x n matrix to be specified. The first variation of the function (6) is given by Eq. (10) which, in the light of (50), becomes

$$\delta f(x) = -\alpha g^T(x)Ag(x) \tag{51}$$

We note that, if the matrix A is positive definite, $g^T(x)Ag(x) > 0$. Therefore, for $\alpha > 0$, the descent property of this algorithm is ensured.

Now, consider the points $\hat{x}$ and x. At point $\hat{x}$, the gradient $g(\hat{x})$ and the matrix $\hat{A}$ are known; at point x, the gradient $g(x)$ is known. Therefore, the differences

$$\Delta x = x - \hat{x}, \qquad \Delta g = g(x) - g(\hat{x}) \tag{52}$$

are available. We wish to determine the matrix A so that the relation

$$A\Delta g = \Delta x \tag{53}$$

is satisfied. After defining the matrix difference

$$\Delta\hat{A} = A - \hat{A} \tag{54}$$

we combine (53)-(54) to obtain

$$\Delta\hat{A}\Delta g = \Delta x - \hat{A}\Delta g \tag{55}$$

in which the only unknown is $\Delta\hat{A}$. Equation (55) admits the solution

$$\Delta\hat{A} = \frac{\Delta x\, y^T}{y^T\Delta g} - \frac{\hat{A}\Delta g\, z^T}{z^T\Delta g} \tag{56}$$

where y and z denote arbitrary n-vectors. Therefore, the matrix A must be updated according to the relation

$$A = \hat{A} + \frac{\Delta x\, y^T}{y^T\Delta g} - \frac{\hat{A}\Delta g\, z^T}{z^T\Delta g} \tag{57}$$

In particular, if one chooses

$$y = \Delta x, \qquad z = \hat{A}\Delta g$$

Eq. (57) becomes

$$A = \hat{A} + \frac{\Delta x\Delta x^T}{\Delta x^T\Delta g} - \frac{\hat{A}\Delta g\Delta g^T\hat{A}}{\Delta g^T\hat{A}\Delta g} \tag{58}$$

Note that the second and third matrices on the right-hand side of (58) are symmetric; therefore, if $\hat{A}$ is symmetric, A is also symmetric.

6.1. Quadratic Function. Now, consider the particular case of a function having the form (43). For this quadratic function, the following properties can be shown to hold (Refs. 4-5 and 7-8):

(a) If the initial matrix A is chosen to be the inverse of the second-derivative matrix H, that is, if

$$A = H^{-1} \tag{59}$$

at the initial point, the variable-metric algorithm exhibits one-step convergence. This is due to the fact that the variable-metric algorithm becomes identical with quasilinearization.

(b) If the initial matrix A is chosen to be positive definite, any subsequent matrix A is also positive definite. With this understanding, the following relations hold:

$$g^T(x)p_* = 0, \qquad p^T H p_* = 0 \tag{61}$$

Equation (61-1) states that the gradient at each iteration is orthogonal to the search direction at every previous iteration. Equation (61-2) states that the search direction at each iteration and the search direction at every previous iteration are conjugate with respect to the constant matrix H. As the algorithm progresses, the matrix A tends to the inverse of the second derivative matrix H, and relation (60) becomes satisfied exactly when convergence is achieved. The algorithm (53), with A updated according to (59), reduces the gradient to zero in no more than n steps; therefore, the minimum of f(x) is reached in no more than n steps.

(c) As a particular case of (b), the initial matrix can be chosen to be

$$A = I \tag{62}$$

where I is the n x n identity matrix. Under these conditions, the variable-metric algorithm becomes identical with the conjugate-gradient algorithm of Section 5.

6.2. Nonquadratic Function. For a nonquadratic function, the variable-metric algorithm is represented by

$$\tilde{x} = x + \Delta x, \quad \Delta x = -\alpha p, \quad p = Ag(x) \tag{63}$$

with

$$A = \hat{A} + \frac{\Delta\hat{x}\Delta\hat{x}^T}{\Delta\hat{x}^T\Delta\hat{g}} - \frac{\hat{A}\Delta\hat{g}\Delta\hat{g}^T\hat{A}}{\Delta\hat{g}^T\hat{A}\Delta\hat{g}} \tag{64}$$

The stepsize α is to be optimized by searching for the minimum of f along the direction defined by (63).

For any iteration except the first, the complete algorithm can be stated as follows: (a) for a given nominal point x, the gradient g(x) is known; since g(x̂), Δx̂, Â are known from the previous iteration, the matrix A can be computed with (64) and the search direction p with (63-3); (b) the optimum stepsize α must be determined by minimizing the function f along the search direction p, as in Section 7; (c) the correction Δx to the position vector x is determined using Eq. (63-2); and (d) the new position vector x̃ is computed through Eq. (63-1). Next, the new position vector x̃ becomes the nominal point for the subsequent iteration and the procedure is repeated until a predetermined stopping condition is satisfied (see Section 8). To start the algorithm, one bypasses (64) and sets A equal to any symmetric, positive-definite matrix (for instance, the identity matrix).

In closing, the following comments are pertinent: (a) in the variable-metric method, it is important that the stepsize α be determined accurately; (b) restarting the algorithm every n or n + 1 iterations is not necessary; however, restarting is indispensable whenever the positive-definiteness of the matrix A is violated, for example, if the stepsize α becomes negative.

7. Search Techniques

In each of the previous methods, the stepsize α must be optimized. In this section, we present techniques to solve the equation

$$F_\alpha(\alpha) = 0 \tag{65}$$

that is, to find the minimum of the function F(α) given by Eq. (19) for the ordinary gradient algorithm or Eq. (30) for the conjugate-gradient and variable-metric algorithms. Since the techniques in question involve the consideration of the first derivative F_α and perhaps the second derivative $F_{\alpha\alpha}$, we summarize these derivatives below.

For all of the previous methods, we have

$$F_\alpha(\alpha) = -g^T(\tilde{x})p\,, \quad F_{\alpha\alpha}(\alpha) = p^T H(\tilde{x})p \tag{66}$$

where

$$\tilde{x} = x - \alpha p \tag{67}$$

The search direction is given by p = g(x) for the ordinary gradient method, Eq. (49-3) for the conjugate-gradient method, and Eq. (63-3) for the variable-metric method. Of course, Eq. (66-2) requires that the second-derivative matrix H(x) be explicitly available. If this is not the case, one can use the difference scheme

$$\begin{aligned} F_{\alpha\alpha}(\alpha) &= (1/2\theta)[F_\alpha(\alpha + \theta) - F_\alpha(\alpha - \theta)] \\ &= (1/2\theta)[g(\tilde{x} + \theta p) - g(\tilde{x} - \theta p)]^T p \end{aligned} \tag{68}$$

In practice, one may choose

$$\theta = \epsilon_1/|p| \tag{69}$$

where ϵ_1 is a small number.

7.1. Cubic Interpolation. Let the values of the function F(α) and its derivative $F_\alpha(\alpha)$ be computed for two different values of α, namely, α_1 and α_2, with $\alpha_2 > \alpha_1 > 0$. If α_1 and α_2 are such that

$$F_\alpha(\alpha_1) < 0\,, \quad F_\alpha(\alpha_2) > 0$$

then the minimum of the function F(α) occurs for some value α in the range

$$\alpha_1 < \alpha < \alpha_2 \tag{71}$$

In this range, we represent the function F(α) with the cubic

$$F(\alpha) = A + B(\alpha - \alpha_1) + C(\alpha - \alpha_1)^2 + D(\alpha - \alpha_1)^3$$

whose first and second derivatives are given by

$$F_\alpha(\alpha) = B + 2C(\alpha - \alpha_1) + 3D(\alpha - \alpha_1)^2\,, \quad F_{\alpha\alpha}(\alpha) = 2C + 6D(\alpha - \alpha_1) \tag{73}$$

The scalar coefficients A, B, C, D are determined by requiring (72) to match the ordinate and the slope of the curve F(α) at α_1 and α_2. Therefore, one has to solve the linear equations

$$\begin{aligned} &F(\alpha_1) = A\,, && F(\alpha_2) = A + B(\alpha_2 - \alpha_1) + C(\alpha_2 - \alpha_1)^2 + D(\alpha_2 - \alpha_1)^3\,, \\ &F_\alpha(\alpha_1) = B, && F_\alpha(\alpha_2) = B + 2C(\alpha_2 - \alpha_1) + 3D(\alpha_2 - \alpha_1)^2 \end{aligned} \tag{74}$$

Once the coefficients of the cubic (72) are known, the optimum value of α is determined by the condition (65). Therefore, in the light of (73), one arrives at the solution

$$\alpha = \alpha_1 + (1/3D)[-C + \sqrt{(C^2 - 3BD)}] \tag{75}$$

At this point, one recomputes the function $F(\alpha)$ and the derivative $F_\alpha(\alpha)$. Then, the process is iterated until a predetermined stopping condition is satisfied. For instance, one may require that

$$|F_\alpha(\alpha)| \le \varepsilon_2 \tag{76}$$

or that

$$|F_\alpha(\alpha)| \le \varepsilon_3 |F_\alpha(0)| \tag{77}$$

where ε_2 and ε_3 are prescribed small numbers.

7.2. Quasilinearization. An alternate technique for computing the optimum stepsize, that of quasilinearization with built-in safeguards to ensure that the function decreases at every step of the iterative search, is now presented. Let

$$\delta\alpha = \alpha - \alpha_0 \tag{78}$$

denote the correction to α starting from an arbitrary nominal value α_0. If quasilinearization is applied to Eq. (65), one obtains the linear algebraic equation

$$F_{\alpha\alpha}(\alpha_0)\delta\alpha + F_\alpha(\alpha_0) = 0 \tag{79}$$

Next, we imbed Eq. (79) in the more general equation

$$F_{\alpha\alpha}(\alpha_0)\delta\alpha + \mu\rho F_\alpha(\alpha_0) = 0 \tag{80}$$

where μ denotes a scaling factor and ρ a direction factor such that

$$0 < \mu \le 1 \ , \quad \rho = \pm 1 \tag{81}$$

Equation (80) admits the solution

$$\delta\alpha = -\mu\rho F_\alpha(\alpha_0)/F_{\alpha\alpha}(\alpha_0)$$

The direction factor ρ is determined in such a way that the first variation

$$\delta F(\alpha_0) = F_\alpha(\alpha_0)\delta\alpha \tag{83}$$

is negative. From (82)-(83), we obtain

$$\delta F(\alpha_0) = -\mu\rho F_\alpha^2(\alpha_0)/F_{\alpha\alpha}(\alpha_0) \tag{84}$$

Therefore, $\delta F(\alpha_0)$ is negative if the direction factor ρ is chosen as follows:

$$\rho = \text{sign}\ F_{\alpha\alpha}(\alpha_0)$$

Because of this choice, the correction (82) becomes

$$\delta\alpha = -\mu F_\alpha(\alpha_0)/|F_{\alpha\alpha}(\alpha_0)|$$

To perform the search, a nominal value must be given to α_0. Then, one sets $\mu = 1$, computes $\delta\alpha$ from Eq. (86) and α from Eq. (78). If $F(\alpha) < F(\alpha_0)$, the scaling factor $\mu = 1$ is acceptable. If $F(\alpha) > F(\alpha_0)$, the previous value of μ must be replaced by some smaller value in the range $0 \le \mu \le 1$ until the condition $F(\alpha) < F(\alpha_0)$ is met; this can be obtained through bisection, that is, by successively dividing the value of μ by 2. At this point, the search step is completed. The value obtained for α becomes the nominal value α_0 for the next search step, and the procedure is repeated until a desired degree of accuracy is obtained, that is, until Ineq. (76) or (77) is satisfied. In the absence of better information, the first step of the search procedure can be made with $\alpha_0 = 0$.

8. Termination of the Algorithm

One way to terminate the gradient algorithm is to impose a condition on the modulus of the gradient, for example,

$$g^T(x)g(x) \le \epsilon_4 \tag{87}$$

where ϵ_4 is a prescribed small number. If the function f(x) is rather flat in the neighborhood of the minimum, then Eq. (87) may not yield precise coordinates. In this case, the following additional condition is suggested:

$$\Delta x^T \Delta x \le \epsilon_5 \tag{88}$$

where ϵ_5 is a prescribed small number.

References

1. HESTENES, M.R., and STIEFEL, E., Methods of Conjugate Gradients for Solving Linear Systems, Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, 1952.

2. BECKMAN, F.S., The Solution of Linear Equations by the Conjugate Gradient Method, Mathematical Methods for Digital Computers, Vol. 1, Edited by A. Ralston and H.S. Wilf, John Wiley and Sons, New York, 1960.

3. FLETCHER, R., and REEVES, C.M., Function Minimization by Conjugate Gradients, Computer Journal, Vol. 7, No. 2, 1964.

4. DAVIDON, W.C., Variable-Metric Method for Minimization, Argonne National Laboratory, Report No. ANL-5990, 1959.

5. FLETCHER, R., and POWELL, M.J.D., A Rapidly Convergent Descent Method for Minimization, Computer Journal, Vol. 6, No. 2, 1963.

6. MIELE, A., and CANTRELL, J.W., Gradient Methods in Mathematical Programming, Part 2, Memory Gradient Method, Rice University, Aero-Astronautics Report No. 56, 1969.

7. MYERS, G.E., Properties of the Conjugate Gradient and Davidon Methods, Journal of Optimization Theory and Applications, Vol. 2, No. 4, 1968.

8. PEARSON, J.D., On Variable Metric Methods of Minimization, Research Analysis Corporation, Technical Paper No. RAC-TP-302, 1968.

RICERCHE DI AUTOMATICA
Vol. 2, No. 2, August 1971
Printed in Italy

UPDATING RULES FOR THE PENALTY CONSTANT USED IN THE PENALTY FUNCTION METHOD FOR MATHEMATICAL PROGRAMMING PROBLEMS

A. MIELE, G. M. COGGINS, and A. V. LEVY

Abstract. The problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$ is considered, where f is a scalar, x an n-vector, and φ a q-vector, with $q < n$. The penalty function method is investigated; that is, a sequence of unconstrained minimization problems is solved, each of which involves the penalty function $U(x, k) = f(x) + kP(x)$. Here, the penalty constant k is a positive, scalar quantity, and $P(x) = \varphi^T(x)\varphi(x)$ is the norm squared of the constraint error.

Crucial to the penalty function method is the prediction of the rate $\alpha = k_2/k_1$ at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. Here, k_1 denotes the penalty constant of the present cycle, and k_2 denotes the penalty constant of the next cycle.

In this paper, two variable-rate updating rules are developed: (i) the updating rule $\alpha = \beta\sqrt{(P_1/P_*)}$ and (ii) the updating rule $\alpha = \beta(P_1/P_*)$, where P_1 is the constraint error at the end of the present cycle, P_* is the constraint error allowed for convergence, and $\beta > 1$. Updating rule (i) tends to produce at the end of the next cycle a constraint error P_2 below the geometric mean of the constraint error at the end of the present cycle and that allowed for convergence; updating rule (ii) tends to produce at the end of the next cycle a constraint error P_2 below that allowed for convergence.

In order to evaluate these updating rules, six numerical examples are investigated. The first example deals with a quadratic function subject to linear constraints; the remaining examples deal with nonquadratic functions subject to nonlinear constraints. Each example is solved with the conjugate gradient algorithm for five starting values of the

Manuscript received May 15, 1970 and in revised form November 17, 1970.

A. Miele is Professor of Astronautics and Mathematical Sciences, Rice University, Houston, Texas 77001. G. M. Coggins is Instructor, Department of Mechanics, US Military Academy, West Point, New York 10996. A. V. Levy is Research Associate, University of Mexico, Mexico City 20, Mexico.

This research, supported by the National Science Foundation, Grant No. GP-32453, is a condensation of the investigation described in Ref. 1.

penalty constant k_0, ranging between 10^{-4} and 10^{4}. For each example, the variable-rate updating rules (i) and (ii), with $\beta = 10$, are compared with the constant-rate updating rules $\alpha = 5$ and $\alpha = 10$. From the numerical experiments, it is concluded that the variable-rate updating rules are superior to the constant-rate updating rules, in that they generally lead to convergence in a smaller number of iterations.

1. Introduction

Over the past several years, considerable work has been done on the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$ using numerical methods. Here, f is a scalar, x an n-vector, and φ a q-vector, with $q < n$.

The approaches employed are generally based on one of two basic ideas. One is to develop algorithms such that the constraints are satisfied, at least to first order, at the end of each iteration (Refs. 2-4). Another is to develop algorithms involving cycles in which the vector x is viewed as unconstrained and a new function related to $f(x)$ and $\varphi(x)$ is minimized. The penalty function method (Refs. 5-8) is an approach of the latter type.

Crucial to the penalty function method is the prediction of the rate at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. This key question is considered in this paper, whose objective is the following: to improve the convergence characteristics of the penalty function method by automatically adjusting the penalty constant used in the method.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0, \tag{2}$$

where f is a scalar, x an n-vector, and φ a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions $f(x)$ and $\varphi(x)$ exist and are continuous and that the constrained minimum exists.

2.1. *First-Order Conditions.* From theory of maxima and minima, it is known that the above problem is equivalent to that of minimizing the augmented function

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x) \tag{3}$$

subject to the constraint (2). Here, the q-vector λ is the Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\lambda \tag{4}$$

denotes the gradient of the augmented function, the optimum solution for x and λ must satisfy the relations

$$\varphi(x) = 0, \quad F_x(x, \lambda) = 0, \tag{5}$$

which are a system of $n + q$ equations in the $n + q$ components of x λ. In Eqs. (4)-(5), the gradients f_x and F_x denote n-vectors and the matrix φ_x is $n \times q$.

2.2. *Approximate Solutions.* Since the system (5) is generally nonlinear, approximate methods must be employed. In this connection, we introduce here the scalar performance indexes

$$P(x) = \varphi^T(x)\,\varphi(x), \qquad Q(x, \lambda) = F_x^T(x, \lambda)\,F_x(x, \lambda), \tag{6}$$

which measure the errors in the constraint and the optimum condit respectively. Then, we observe that $P = 0$ and $Q = 0$ for the optimum solution, while $P > 0$ and/or $Q > 0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to val of x and λ such that

$$P(x) \leq P_0, \quad Q(x, \lambda) \leq Q_0. \tag{7}$$

Alternatively, (7) can be replaced by

$$R(x, \lambda) \leq R_0, \tag{8}$$

where

$$R(x, \lambda) = P(x) + Q(x, \lambda)$$

denotes the cumulative error in the constraint and the optimum condition. In Eqs. (7)-(8), P_0, Q_0, R_0 are small, preselected numbers. Note that, if one chooses $P_0 = Q_0 = R_0$, satisfaction of Ineq. (8) implies satisfaction of Ineqs. (7).

3. Penalty Function Method

The penalty function method is based on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. Specifically, the penalty function is defined by

$$U(x, k) = f(x) + kP(x) = f(x) + k\varphi^T(x)\varphi(x) \tag{10}$$

and is obtained by adding to the function $f(x)$ a term quadratic in the constraint $\varphi(x)$, where $k > 0$ is the penalty constant.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the penalty function (10) with respect to x for given k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of a cycle:

$$U_x(x, k) = f_x(x) + kP_x(x) = f_x(x) + 2k\,\varphi_x(x)\,\varphi(x) = 0. \tag{11}$$

If one defines the Lagrange multiplier to be

$$\lambda = 2k\,\varphi(x), \tag{12}$$

Eq. (11) can be rewritten as

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\,\lambda = 0, \tag{13}$$

meaning that the combination of x and λ obtained at the end of a cycle satisfies exactly the optimum condition (5-2). However, if the penalty constant k is arbitrary, the vector x which satisfies Eq. (11) generally violates the constraint condition (5-1).

In order to obtain constraint satisfaction, increasingly larger values of the penalty constant must be employed in successive cycles of the penalty function method. In this connection, let k_1 denote the penalty constant of the present cycle and k_2 denote the penalty constant of the next cycle, with $k_2 > k_1$. Because of the jump in k, the penalty function increases by the amount

$$U(x, k_2) - U(x, k_1) = (k_2 - k_1)\,P(x), \tag{14}$$

and the norm squared of the gradient of the penalty function takes on the

value[1]

$$U_x^T(x, k_2)\,U_x(x, k_2) = (k_2 - k_1)^2\,P_x^T(x)\,P_x(x) = (\alpha - 1)^2 f_x^T(x) f_x(x), \quad (1$$

where

$$P(x) = \varphi^T(x)\,\varphi(x), \qquad P_x(x) = 2\varphi_x(x)\,\varphi(x). \tag{16}$$

The positiveness of the right-hand side of Eq. (14) is the key to the mechanism on which the penalty function method is based.

After a sufficient number of cycles, the constraint error can be made as small as desired providing the penalty constant has become sufficiently large. Theoretically speaking, the condition $\varphi(x) = 0$ is desired at convergence; consequently, the multiplier λ defined by Eq. (12) can be identical with the multiplier satisfying Eqs. (5), which is generally nonzero, only if $k \to \infty$. In a practical digital computer, this means that large values of k are needed at convergence.

3.1. *Numerical Implementation.* From the above considerations, the following outline of the penalty function method emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the penalty function

$$U(x, k) = f(x) + k\varphi^T(x)\,\varphi(x) \tag{17}$$

is minimized with respect to x for given k. The minimum of $U(x, k)$ is achieved when the following stopping condition is satisfied:

$$U_x^T(x, k)\,U_x(x, k) \leq Q_x, \tag{18}$$

where Q_x is a small, preselected number.

(c) Upon termination of a cycle, one checks the following inequality:

$$\varphi^T(x)\,\varphi(x) + U_x^T(x, k)\,U_x(x, k) \leq R_x, \tag{19}$$

where R_x is a small, preselected number. If Ineq. (19) is satisfied, the algorithm is terminated. If Ineq. (19) violated, the algorithm is continued by choosing the solution point of the present cycle as the starting point of the next cycle.

[1] Note that $U_x^T(x, k_1)\,U_x(x, k_1) = 0$.

(d) For the next cycle, a higher value of the penalty constant is selected, specifically,

$$k_2 = \kappa k_1, \tag{20}$$

where $\kappa > 1$ is the penalty constant ratio. After updating the penalty constant, one returns to (b) and continues iteratively.

Remark 3.1. At convergence of a cycle, the stopping condition (18) can be written as

$$Q(x, \lambda) \leq Q_*, \tag{21}$$

where λ is given by Eq. (12). Analogously, at convergence of the algorithm, the stopping condition (19) becomes

$$R(x, \lambda) = P(x) + Q(x, \lambda) \leq R_*, \tag{22}$$

where λ is given by Eq. (12).

4. Updating Rules

Crucial to the penalty function method is the prediction of the rate

$$\kappa = k_2/k_1 \tag{23}$$

at which the penalty constant must be increased when shifting from one cycle of the algorithm to the next. Here, k_1 denotes the penalty constant of the present cycle, and k_2 denotes the penalty constant of the next cycle.

4.1. *Standard Technique.* The standard method (see, for instance, Ref. 7) consists of employing a constant value of κ throughout the algorithm, for example,

$$\kappa = 5 \quad \text{or} \quad \kappa = 10. \tag{24}$$

4.2. *Proposed Technique.* An alternate method consists of employing a variable value of κ throughout the algorithm. For instance, one might select κ so as to achieve a predetermined constraint error at the end of the next cycle of the algorithm.

Let the following definition be introduced:

$$Z = \lambda^T \lambda, \tag{25}$$

with λ given by Eq. (12). Let Eqs. (12) and (25) be applied twice, once at the end of the present cycle (subscript 1) and once at the end of the next cycle (subscript 2). One obtains the relations

$$Z_1 = 4k_1^2 P_1, \quad Z_2 = 4k_2^2 P_2, \tag{26}$$

which imply that

$$\kappa = \sqrt{(Z_2 P_1/Z_1 P_2)}. \tag{27}$$

Now, we introduce the basic assumption [1]

$$Z_1 = Z_2, \tag{28}$$

that is, we neglect the change in the norm squared of the multiplier between the end of the present cycle and the end of the next cycle. With this understanding, Eq. (27) simplifies to

$$\kappa = \sqrt{(P_1/P_2)}.$$

This relation enables one to predict the penalty constant needed for the next cycle in order to achieve a predetermined constraint error P_2.

As an example, assume that the expected constraint error at the end of the next cycle P_2 is below the geometric mean of the constraint error at the end of the present cycle P_1 and that allowed for convergence P_*; that is, assume that

$$P_2 = \sqrt{(P_1 P_*)}/\beta^2, \tag{30}$$

where $\beta > 1$. Then, Eq. (29) simplifies to

$$\kappa = \beta \sqrt[4]{(P_1/P_*)}. \tag{31}$$

As another example, assume that the expected constraint error at the end of the next cycle P_2 is below that allowed for convergence P_*; that is, assume that

$$P_2 = P_*/\beta^2, \tag{32}$$

where $\beta > 1$. Then, Eq. (29) simplifies to

$$\kappa = \beta \sqrt{(P_1/P_*)}. \tag{33}$$

[1] Hypothesis (28) is key to this paper and is supported by the data exhibited in Tables 1-4.

5. Experimental Conditions

In order to evaluate the theory, six numerical examples were solved using a Burroughs B-5500 computer, double-precision arithmetic, and FORTRAN-IV program. Within each cycle of the penalty function method, the conjugate-gradient algorithm was employed (Ref. 1).

Starting Point. In all the examples, the nominal point chosen to start the penalty function method was defined by

$$x_1 = x_2 = \dots = x_n = 3, \tag{34}$$

where $x_1, x_2, \dots, x_n$ denote the components of the vector x.

Starting Penalty Constant. For the first cycle of the penalty function method, five values of the penalty constant were employed, namely,

$$k_1 = 10^{-2}, \quad k_2 = 10^{-1}, \quad k_3 = 10^{0}, \quad k_4 = 10^{1}, \quad k_5 = 10^{2}. \tag{35}$$

Updating Rules. For subsequent cycles, the penalty constant was determined in accordance with one of the following penalty constant ratios:

$$\pi = 5, \quad \pi = 10, \tag{36}$$

and

$$\pi = \beta \sqrt[4]{(P_1/P_*)}, \quad \pi = \beta \sqrt{(P_1/P_*)}, \tag{37}$$

with

$$\beta = 10, \quad P_* = 10^{-16}. \tag{38}$$

The penalty constant ratios (36) are representative of the standard method (see, for instance, Ref. 7) and the penalty constant ratios (37) are representative of the new method proposed here.

Definition of Convergence. Convergence of a cycle was defined as

$$Q(x, \lambda) = U_x^T(x, k)\, U_x(x, k) \le 10^{-16}, \tag{39}$$

and convergence of the complete algorithm was defined as

$$R(x, \lambda) = P(x) + Q(x, \lambda) = \varphi^T(x)\, \varphi(x) + U_x^T(x, k)\, U_x(x, k) \le 10^{-16}, \tag{40}$$

where the multiplier λ is given by Eq. (12).

Search Technique. Within each iteration of the conjugate gradient algorithm, let x denote the nominal point, $\tilde{x}$ the varied point, and p the search direction. Let $\tilde{U}(\alpha)$ denote the function representing the behavior of the penalty function along the search direction, that is,

$$\tilde{U} = U(\tilde{x}, k) = U(x + \Delta x, k) = U(x - \alpha p, k) = \tilde{U}(\alpha). \tag{41}$$

This function was employed in order to determine the optimum stepsize at each iteration. Specifically, a precise one-dimensional search on the function $\tilde{U}(\alpha)$ was conducted such that, in any given step, the inequality [2]

$$\tilde{U}(\alpha) < \tilde{U}(\alpha_0) \tag{42}$$

was satisfied, where α_0 is the nominal stepsize and α is the varied stepsize. The search was terminated when the stopping condition

$$\tilde{U}_\alpha^2(\alpha) \le \tilde{U}_\alpha^2(0) \times 10^{-8} \tag{43}$$

was satisfied.

6. Numerical Examples

In this section, six numerical examples are described. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

EXAMPLE 6.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \tag{44}$$

subject to the constraints

$$x_1 + 3x_2 = 0, \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0. \tag{45}$$

[2] Due to the analytical nature of the examples considered here, multistep, vectorial quasilinearization was employed in order to ensure satisfaction of Ineq. (42). However, in a more realistic situation, one would use quadratic interpolation or cubic interpolation. For the details of the conjugate-gradient algorithm and the search technique employed, the reader should consult Ref. 1.

This function admits the relative minimum $f = 4.0930$ at the point defined by

$$x_1 = -0.7674,\ x_2 = 0.2558,\ x_3 = 0.6279,\ x_4 = -0.1162,\ x_5 = 0.2558 \quad (46)$$

and

$$\lambda_1 = 2.0465,\quad \lambda_2 = 2.2325,\quad \lambda_3 = -5.0534. \quad (47)$$

EXAMPLE 6.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \quad (48)$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0. \quad (49)$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1048,\quad x_2 = 1.1966,\quad x_3 = 1.5352 \quad (50)$$

and

$$\lambda_1 = -0.1072 \times 10^{-1}. \quad (51)$$

EXAMPLE 6.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \quad (52)$$

subject to the constraints

$$x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0,\quad x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0,$$

$$x_1 x_5 - 2 = 0. \quad (53)$$

The function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1911,\quad x_2 = 1.3626,\quad x_3 = 1.4728,\quad x_4 = 1.6350,\quad x_5 = 1.6790 \quad (54)$$

and

$$\lambda_1 = -0.3482 \times 10^{-1},\ \lambda_2 = -0.1072 \times 10^{-1},\ \lambda_3 = -0.2879 \times 10^{-1}. \quad (55)$$

EXAMPLE 6.4. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \quad (56)$$

subject to the inequality constraint

$$x_1 \leq -1. \quad (57)$$

Introduce the auxiliary variable x_3 defined by

$$x_1 + x_3^2 + 1 = 0. \quad (58)$$

Then, the previous problem can be recast as that of minimizing the function (56) subject to the equality constraint (58). The function (56) admits the relative minimum $f = 0.04$ at the point defined by

$$x_1 = -1,\quad x_2 = 1,\quad x_3 = 0$$

and

$$\lambda_1 = 0.04.$$

EXAMPLE 6.5. Consider the problem of minimizing the function

$$f = -x_1 \quad (61)$$

subject to the inequality constraints

$$x_2 \geq x_1^3,\quad x_2 \leq x_1^2. \quad (62)$$

Introduce the auxiliary variables x_3 and x_4 defined by

$$x_2 - x_1^3 - x_3^2 = 0,\quad x_1^2 - x_2 - x_4^2 = 0. \quad (63)$$

Then, the previous problem can be recast as that of minimizing the funct (61) subject to the equality constraints (63). The function (61) admits t relative minimum $f = -1$ at the point defined by

$$x_1 = 1,\quad x_2 = 1,\quad x_3 = 0,\quad x_4 = 0 \quad (64)$$

and

$$\lambda_1 = -1,\quad \lambda_2 = -1. \quad (65)$$

EXAMPLE 6.6. Consider the problem of minimizing the function

$$f = \log x_1 - x_2 \quad (66)$$

subject to the equality constraint

$$x_1^2 + x_2^2 - 4 = 0 \quad (67)$$

and the inequality constraint

$$x_3 \geq 1. \tag{68}$$

Introduce the auxiliary variable x_4 defined by

$$x_3 = 1 + x_4^2. \tag{69}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log (1 + x_1^2) - x_2 \tag{70}$$

subject to the equality constraint

$$(1 + x_1^2)^2 + x_2^2 - 4 = 0. \tag{71}$$

so that x_3 has been eliminated from the problem and can be computed *a posteriori* with (69). The function (70) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0, \quad x_2 = \sqrt{3}, \quad x_3 = 1 \tag{72}$$

and

$$\lambda_1 = 1/2\sqrt{3}. \tag{73}$$

7. Results and Conclusions

The examples described in Section 6 were solved with the penalty function method in conjunction with the conjugate gradient algorithm according to the experimental conditions outlined in Section 5. The numerical results are presented in Tables 1-16. Tables 1-4 supply a justification of the basic assumption (28), and Tables 5-16 supply comparative data for the standard updating rules and the new updating rules.

Basic Hypothesis. Tables 1-4 refer to a particular example, namely, Example 6.5. They show the behavior of the penalty function method for each of the penalty constant ratios (36) and (37) assuming that the starting penalty constant is $k_0 = 1$. In the tables, the quantities P, Q, R, Z are shown versus the iteration number N and the cycle number N_c. To shorten the tables, these quantities are given only at the initial point and the final point of each cycle.

As the tables indicate, the stopping condition (39) is met at convergence of a cycle and the stopping condition (40) is met at convergence of

the algorithm. Of particular interest is the behavior of the quantity Z, defined by $Z = \lambda^T\lambda = 4k^2P$. If the end conditions of successive cycles are compared, it is clear that Z approaches an asymptotic value as the cycle number N_c increases and the penalty constant k becomes sufficiently large. This asymptotic value is independent of the particular penalty constant ratio employed and is the same in each of Tables 1-4, namely $Z = 2$. This is in agreement with the theoretical values of the multipliers at convergence, which are supplied by Eqs. (68). This result confirms the basic assumption (28) and justifies the reasoning leading to the general penalty constant ratio (20).

Comparative Data. Tables 5-16 refer to Examples 6.1 through 6.6. They show the number of iterations at convergence N_* and the number of cycles at convergence N_{c*} versus the starting penalty constant k_0 for each of the penalty constant ratios (36)-(37). Since each iteration requires the same amount of computational work, it is clear that the number of iterations is representative of the computer time required for convergence.

For all of the examples and each starting penalty constant k_0, the variable-rate updating rules are superior to the constant-rate updating rules in that they require a smaller number of iterations for convergence. This smaller number of iterations is achieved by employing a smaller number of cycles and by increasing the penalty constant at a higher rate than that given by Eqs. (36).

Table 1. Example 6.5, $k_s = 1$, $n = 5$.

N_s	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 01	0.57 E + 05	0.57 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.85 E − 01	0.12 E − 15	0.85 E − 01	0.30 E + 00
2	17	0.50 E + 01	0.85 E − 01	0.16 E + 03	0.16 E + 03	0.85 E + 01
2	27	0.50 E + 01	0.92 E − 02	0.12 E − 13	0.92 E − 02	0.93 E + 00
3	27	0.25 E + 02	0.92 E − 02	0.16 E + 03	0.16 E + 03	0.23 E + 03
3	31	0.25 E + 02	0.62 E − 03	0.16 E − 14	0.62 E − 03	0.16 E + 01
4	31	0.13 E + 03	0.62 E − 03	0.16 E + 03	0.16 E + 03	0.39 E + 03
4	47	0.13 E + 03	0.30 E − 04	0.60 E − 14	0.30 E − 04	0.19 E + 01
5	47	0.63 E + 03	0.30 E − 04	0.16 E + 03	0.16 E + 03	0.47 E + 03
5	53	0.63 E + 03	0.13 E − 05	0.13 E − 15	0.13 E − 05	0.20 E + 01
6	53	0.31 E + 04	0.13 E − 05	0.16 E + 03	0.16 E + 03	0.49 E + 03
6	59	0.31 E + 04	0.51 E − 07	0.04 E − 17	0.51 E − 07	0.20 E + 01
7	59	0.16 E + 05	0.51 E − 07	0.16 E + 03	0.16 E + 03	0.50 E + 03
7	65	0.16 E + 05	0.20 E − 08	0.14 E − 16	0.20 E − 08	0.20 E + 01
8	65	0.78 E + 05	0.20 E − 08	0.16 E + 03	0.16 E + 03	0.50 E + 03
8	70	0.78 E + 05	0.82 E − 10	0.25 E − 13	0.82 E − 10	0.20 E + 01
9	70	0.39 E + 06	0.82 E − 10	0.16 E + 03	0.16 E + 03	0.50 E + 03
9	75	0.39 E + 06	0.33 E − 11	0.88 E − 17	0.33 E − 11	0.20 E + 01
10	75	0.20 E + 07	0.33 E − 11	0.16 E + 03	0.16 E + 03	0.50 E + 03
10	79	0.20 E + 07	0.13 E − 12	0.17 E − 13	0.13 E − 12	0.20 E + 01

Table 2. Example 6.5, $k_s = 1$, $n = 10$.

N_s	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 01	0.57 E + 05	0.57 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.85 E − 01	0.12 E − 15	0.85 E − 01	0.30 E + 00
2	17	0.10 E + 02	0.85 E − 01	0.81 E + 03	0.81 E + 03	0.38 E + 03
2	30	0.10 E + 02	0.30 E − 02	0.13 E − 15	0.30 E − 02	0.19 E + 01
3	30	0.10 E + 03	0.30 E − 02	0.81 E + 03	0.81 E + 03	0.12 E + 03
3	40	0.10 E + 03	0.48 E − 04	0.73 E − 14	0.48 E − 04	0.19 E + 01
4	40	0.10 E + 04	0.48 E − 04	0.81 E + 03	0.81 E + 03	0.19 E + 03
4	47	0.10 E + 04	0.50 E − 06	0.50 E − 14	0.50 E − 06	0.20 E + 01
5	47	0.10 E + 05	0.50 E − 06	0.81 E + 03	0.81 E + 03	0.20 E + 03
5	53	0.10 E + 05	0.50 E − 08	0.13 E − 17	0.50 E − 08	0.20 E + 01
6	53	0.10 E + 06	0.50 E − 08	0.81 E + 03	0.81 E + 03	0.20 E + 03
6	59	0.10 E + 06	0.50 E − 10	0.27 E − 20	0.50 E − 10	0.20 E + 01
7	59	0.10 E + 07	0.50 E − 10	0.81 E + 03	0.81 E + 03	0.20 E + 03
7	64	0.10 E + 07	0.50 E − 12	0.19 E − 16	0.50 E − 12	0.20 E + 01

Comparison of Multiplier and Quasilinearization Methods

Alejandro V. Levy*

CIMAS, Universidad Nacional Autonoma de Mexico, Mexico City, Mexico and Department of Mechanical Engineering, Rice University, Houston, Texas

Antonio Montalvo

Department of Mathematics, Universidad Iberoamericana, Mexico City, Mexico

Miele et al. (1972) developed the method of multipliers for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$, where f is a scalar, x is an n vector, and φ is a q vector, with $q < n$. In this paper, a comparison of the standard quasilinearization method and the method of multipliers is presented. The comparison is made by considering the relative stability and the relative speed of convergence of the two methods. Numerical results are presented for nine examples, each of which is solved for 100 different starting points. The examples show that, in general, the method of multipliers is a more economic and robust algorithm than the standard quasilinearization method. In particular, this advantage increases in proportion to the following characteristics of the problem being solved: (a) the ratio q/n and the size of the problem and (b) the nonlinearity of the problem and the number of relative minima and maxima that may exist.

1. Introduction

In recent years, considerable attention has been given to the quasilinearization methods for minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$.

In the *standard quasilinearization algorithm* (SQL), the augmented function $F(x,\lambda) = f(x) + \lambda^T\varphi(x)$ is utilized, where λ, a q-vector, is an unknown Lagrange multiplier. Starting from nominal values of x and λ, the method obtains corrections Δx and $\Delta\lambda$ by solving a linear system of equations. In this way, updated values $\tilde{x} = x + \Delta x$ and $\tilde{\lambda} = \lambda + \Delta\lambda$ are obtained. Then, the process is repeated iteratively until convergence occurs. For the particular case of a quadratic function subject to a linear constraint, the method converges to the solution in one iteration. However, for a nonlinear function subject to nonlinear constraints, the method is rather unstable. Depending on the nominal values of x and λ, it can lead to a relative minimum, a relative maximum, or an inflection point; occasionally, the method produces displacements which are so purposeless that overflow can occur in a given computer.

In the *modified quasilinearization algorithm* (MQL), Miele et al. (1973), the same basic approach is taken as in SQL, with one important modification: the stepsize α, $0 < \alpha \leq 1$, is introduced in order to control the magnitude of the displacements. The result is an improved quasilinearization algorithm that (i) retains quadratic convergence for a linear-quadratic problem and (ii) improves the stability of SQL for nonlinear functions subject to nonlinear constraints.

Both the above mentioned methods require the solution of a linear system of equations of order $n + q$. Thus, if Gaussian elimination is employed, the computational effort is of the order of $(n + q)^3$ multiplications.

For large systems, one way to reduce the computational effort per iteration is to resort to penalty function methods, either in the standard format or the multiplier format. [illegible] the augmented penalty function $W(x,\lambda,k) = F(x,\lambda) + k\varphi^T(x)\varphi(x)$, where the scalar k is the penalty constant.

With reference to the method of multipliers, an important modification was presented by Miele et al. (1972). In this modification, called the *modified method of multipliers* (MMM), the augmented penalty function is employed in connection with several minimization algorithms, namely, the ordinary gradient algorithm, the conjugate gradient algorithm, and the modified quasilinearization algorithm. Improved updating rules for the Lagrange multiplier and the penalty constant were developed in connection with each minimization algorithm.

When the modified method of multipliers is employed in conjunction with the modified quasilinearization algorithm, a linear system of equations of order n must be solved at every iteration. Therefore, if Gaussian elimination is employed, the computational effort is of the order of n^3 multiplications.

Besides reducing the computational effort per iteration, MMM is more stable than SQL. This is because MMM has a descent property on the augmented penalty function $W(x,\lambda,k)$. While SQL might converge to a relative minimum, a relative maximum, or an inflection point, MMM generally leads to a relative minimum.

The standard quasilinearization algorithm was used by Luus and Jaakola (1973) with some success to solve several problems, using random numbers as starting values for x and λ. The purpose of this report is to show that even greater success and computer time savings can be obtained if the modified method of multipliers is utilized instead of the standard quasilinearization algorithm.

In section 2, the statement of the problem is given together with the first-order optimality conditions. In sections 3 and 4, the standard quasilinearization algorithm and the modified method of multipliers are reviewed briefly. In section 5, a comparison of the computational effort per iteration is given. In sections 6 and 7, the experimental conditions and nine numerical examples are presented. In section 8, numerical results are discussed, and the conclusions are stated.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 \tag{2}$$

Table 3. Example 6.6, $k_0 = 1$, $\alpha = 10\sqrt[4]{(P_1/P_0)}$.

N_c	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 03	0.57 E + 06	0.61 E + 05	0.43 E + 03
1	17	0.10 E + 01	0.05 E − 01	0.13 E − 16	0.95 E − 01	0.20 E + 00
2	17	0.56 E + 04	0.55 E − 01	0.31 E + 09	0.61 E + 08	0.13 E + 08
2	12	0.56 E + 04	0.16 E − 07	0.83 E − 13	0.18 E − 07	0.20 E + 01
3	32	0.63 E + 06	0.16 E − 07	0.13 E + 06	0.13 E + 05	0.25 E + 05
3	48	0.63 E + 06	0.11 E − 11	0.33 E − 15	0.13 E − 11	0.20 E + 01
4	36	0.67 E + 07	0.13 E − 11	0.53 E + 07	0.56 E + 03	0.23 E + 03
4	42	0.67 E + 07	0.11 E − 20	0.34 E − 17	0.35 E − 13	0.20 E + 01

Table 4. Example 6.5, $k_0 = 1$, $\alpha = 10\sqrt{(P_1/P_0)}$.

N_c	N	k	P	Q	R	S
1	0	0.10 E + 01	0.10 E + 03	0.57 E + 05	0.67 E + 05	0.42 E + 03
1	17	0.10 E + 01	0.06 E − 01	0.13 E − 13	0.95 E − 01	0.89 E + 00
2	17	0.31 E + 07	0.55 E − 01	0.06 E + 13	0.96 E + 13	0.80 E + 13
2	64	0.31 E + 07	0.63 E − 10	0.64 E − 14	0.63 E − 18	0.20 E + 01

Table 5. Example 6.1, number of iterations N_0.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	70	63	34	23
10^{-1}	60	47	21	17
10^{0}	56	43	21	19
10^{1}	48	37	23	29
10^{2}	40	33	26	26

Table 6. Example 6.1, number of cycles N_c.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	14	10	6	5
10^{-1}	12	8	4	5
10^{0}	11	8	4	3
10^{1}	10	7	4	3
10^{2}	8	6	4	3

TABLE 7. Example 6.2, number of iterations N_i.

k_1	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	60	62	56	41
10^{-1}	64	60	60	43
10^{0}	59	60	58	45
10^{1}	68	67	64	61
10^{2}	80	80	77	76

TABLE 8. Example 6.2, number of cycles N_c.

k_1	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	10	7	4	3
10^{-1}	8	6	3	2
10^{0}	7	5	3	2
10^{1}	6	4	4	3
10^{2}	4	3	6	2

TABLE 9. Example 6.3, number of iterations N_i.

k_1	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	(a)	(a)	134	176
10^{-1}	164	168	120	98
10^{0}	166	162	90	100
10^{1}	138	138	82	79
10^{2}	188	131	107	102

TABLE 10. Example 6.3, number of cycles N_c.

k_1	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_1/P_0)}$	$\alpha = 10\sqrt{(P_1/P_0)}$
10^{-2}	(a)	(a)	4	3
10^{-1}	9	7	4	3
10^{0}	8	8	3	3
10^{1}	6	6	5	3
10^{2}	5	4	5	3

(a) Number of iterations exceeded 200.

TABLE 11. Example 6.4, number of iterations N_i.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	65	48	31	30
10^{-1}	60	47	33	31
10^{0}	60	48	39	35
10^{1}	81	76	50	60
10^{2}	175	170	167	104

TABLE 12. Example 6.4, number of cycles N_c.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	11	8	4	3
10^{-1}	9	7	5	3
10^{0}	8	6	3	3
10^{1}	6	5	3	3
10^{2}	6	6	4	3

TABLE 13. Example 6.5, number of iterations N_i.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	116	93	63	44
10^{-1}	93	70	47	37
10^{0}	70	64	43	34
10^{1}	67	51	35	30
10^{2}	58	40	38	31

TABLE 14. Example 6.5, number of cycles N_c.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	18	9	4	3
10^{-1}	11	8	4	2
10^{0}	10	7	4	3
10^{1}	8	8	3	3
10^{2}	7	5	3	3

TABLE 15. Example 6.6, number of iterations N_*.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	27	23	13	10
10^{-1}	23	18	13	8
10^{0}	23	19	13	13
10^{1}	20	16	13	11
10^{2}	26	21	20	10

TABLE 16. Example 6.6, number of cycles N_c.

k_0	$\alpha = 5$	$\alpha = 10$	$\alpha = 10\sqrt[4]{(P_0/P_*)}$	$\alpha = 10\sqrt{(P_0/P_*)}$
10^{-2}	12	9	4	3
10^{-1}	10	8	4	2
10^{0}	8	7	3	3
10^{1}	7	6	3	3
10^{2}	8	6	3	3

REFERENCES

[1] MIELE, A., COGGINS, G. M., and LEVY, A. V., « *Updating Rules for the Penalty Constant Used in the Penalty Function Method for Mathematical Programming Problems* », Rice University, Aero-Astronautics Report No. 90, 1972.

[2] MIELE, A., HUANG, H. Y., and HEIDEMAN, J. C., « *Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions* », Journal of Optimization Theory and Applications, Vol. 4, No. 4, 1969.

[3] MIELE, A., LEVY, A. V., and CRAGG, E. E., « *Modifications and Extensions of the Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems* », Journal of Optimization Theory and Applications, Vol. 7, No. 6, 1971.

[4] MIELE, A., HEIDEMAN, J. C., and LEVY, A. V., « *Combined Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems* », Ricerche di Automatica, Vol. 2, No. 2, 1971.

[5] KELLEY, H. J., « *Methods of Gradients* », Optimization Techniques, Edited by G. Leitmann, Academic Press, New York, 1962.

[6] BRYSON, A. E., Jr., and HO, Y. C., « *Applied Optimal Control* », Blaisdell Publishing Company, Waltham, Massachusetts, 1969.

[7] FIACCO, A. V., and MCCORMICK, G. P., « *Nonlinear Programming: Sequential Unconstrained Minimization Techniques* », John Wiley and Sons, New York, 1968.

[8] DAVIES, D., and SWANN, W. H., « *Review of Constrained Optimization* », Optimization, Edited by R. Fletcher, Academic Press, New York, 1969.

Paper No.
76-DE-E

an ASME publication

$3.00 PER COPY $1.50 TO ASME MEMBERS

The Society shall not be responsible for statements or opinions advanced in papers or in discussion at meetings of the Society or of its Divisions or Sections, or printed in its publications. Discussion is printed only if the paper is published in an ASME journal or Proceedings. Released for general publication upon presentation. Full credit should be given to ASME, the Technical Division, and the author(s).

G. A. Gabriele
Graduate Assistant.

K. M. Ragsdell
Associate Professor,
Mem. ASME

School of Mechanical Engineering,
Purdue University,
West Lafayette, Ind.

The Generalized Reduced Gradient Method: A Reliable Tool for Optimal Design

This paper is a presentation of a method, called the Generalized Reduced Gradient Method, which has not received wide attention in the engineering design literature. Included is a theoretical development of the method, a description of the basic algorithm, and additional recommendations to produce an efficient code. A Fortran code employing this theory was written and tested on the Eason and Fenton [1][1] test problems, illustrating the method to be efficient and reliable.

Introduction

Engineering design is a multiphase process requiring constant decision making by the designer. Basing decisions on his knowledge and experience, he must arrive at a combination of design variables that best satisfies his objectives. As engineering design has matured so have the guidelines and methods that the designer has at his disposal to aid him in his choices. Although his experience may provide some answers, many of his choices are based on analytical methods. Designers have generally been receptive to new methods, when they simplify and/or otherwise enhance the overall synthesis process. The availability of high-speed digital computing power has encouraged the application of modern optimization methods (mathematical programming methods) to engineering design. This marriage of effort is generating developments in both fields, and has produced an important area of research known as optimal design.

Drawing on his experience the engineer is able to define variables, a design objective and a set of constraints that must be met in order that the design be a workable solution. By developing corresponding equations the design problem can be formulated into a standard form acceptable to mathematical programming techniques. This standard form is defined here

Minimize

$$f(x); \quad x = [x_1, x_2, x_3, \ldots, x_N]^T, \qquad x \in R^N \tag{1}$$

subject to

$$\phi_k(x) \ge 0 \qquad k = 1, 2, 3, \ldots, K \tag{2}$$

$$\psi_l(x) = 0 \qquad l = 1, 2, 3, \ldots, L \tag{3}$$

where

x = a column vector of design variables
N = total number of design variables
$f(x)$ = design criteria or objective function
$\phi_k(x)$ = K inequality constraint functions; these functions delimit regions in the design space
$\psi_l(x)$ = L equality constraint functions; these functions vastly reduce the number of candidate designs, because they require specific combinations of the design variables

When equations (1)–(3) are linear functions in the design variables the problem falls in the general class of linear programming (LP) problems. Much of the research in the past 20 years in the field of mathematical programming has been with linear programming. The simplex algorithm has become highly developed and most linear programs can be solved using this technique.

A more general occurrence in engineering design arises when expressions (1)–(3) are nonlinear. This is known as the nonlinear programming (NLP) problem. No general method has been developed to solve nonlinear problems in the sense that the simplex algorithm exists to solve the linear problem. Although many strategies have been suggested, comparative studies [1, 2] have shown that no method has been successfully applied to all problems.

One procedure for solving the NLP that has seen widespread usage is the penalty function approach. Consider the following function

$$P(x) = f(x) + W(x, R, \phi, \psi) \tag{4}$$

which we call the generalized weighted penalty function. $W(x, R, \phi, \psi)$ is the penalty term, and is constructed in such a manner that feasible points (points satisfying (2) and (3)) are given favored treatment. The basic concept of the penalty function method is that successive minimizations of the unconstrained function $P(x)$ will converge to the constrained minimum of the original NLP. A survey of penalty function techniques is given by Fiacco and McCormick [3].

Contributed by the Design Engineering Division for publication in the JOURNAL OF ENGINEERING FOR INDUSTRY. Manuscript received at ASME Headquarters June 1, 1976. Paper No. 76-DE-E.

[1] Numbers in brackets designate References at end of paper.

An advantage of penalty function methods is their ease of implementation. Any unconstrained searching technique can be used to perform the successive minimizations of $P(x)$. However, there are several difficulties with penalty function methods that do not make them a generally reliable technique. First, if the successive minimizations of $P(x)$ are not trivial, then the time to solution may become impractical. Penalty function methods are also very sensitive to parameter adjustment and constraint scaling, and may require repetitive adjustment to obtain convergence to an optimum. Finally, penalty function methods generally do not possess the ability to move along equality constraints. Rather they tend to get caught on a point that satisfies the constraints, but is not the constrained optimum.

Because the state of the art for linear programming is so well developed, another approach to solving the constrained NLP is linearization of the NLP to tailor it to the LP methods. Basically this can be accomplished by replacing the nonlinear functions of (1)-(3) by first-order Taylor series approximations and solving the resulting LP. The Griffith and Stewart method [4] is based on this approach.

The advantage of this approach lies in being able to use existing linear programming theory. The method performs well on large-scale problems when the objective function and constraints are nearly linear. Difficulties arise when these functions are very nonlinear. Hence in order that the linear approximations hold, only small changes in the design variables are allowed, which may cause progress to the optimum to be slow. Indeed in the presence of highly nonlinear functions, the LP solution may give rise to a poor search direction, therefore hindering the optimization process.

Other methods of handling the constrained NLP are described in the literature [5, 6]; however, many of these methods do not solve the complete NLP described in (1)-(3). The two approaches described here do handle the complete problem and have seen wide acceptance in optimal engineering design today. In this paper another approach known as the Reduced Gradient Method will be described. The Reduced Gradient Method avoids many of the problems associated with penalty function and LP-like methods, producing one of the most powerful methods currently known for handling the constrained nonlinear programming problem.

Reduced Gradient—Theory

The Reduced Gradient Method was originally given by Wolfe for a nonlinear objective function with linear constraints [7, 8]. A generalization of Wolfe's method to accommodate nonlinearities in both the objective function and constraints was first accomplished by Abadie [9]. Concurrently to both Wolfe and Abadie, Wilde and Beightler developed their differential algorithm based on the constrained derivative [10]. The constrained derivative and the reduced gradient employ much the same theoretical basis, but for purposes of this discussion the method shall be known as the Reduced Gradient Method.

The general constrained nonlinear programming problem of (1)-(3) can be restated in the following form:

Minimize

$$f(x); \qquad x = [x_1, x_2, \ldots, x_N]^T, \qquad x \in R^N \tag{5}$$

subject to

$$\psi_m(x) = 0 \qquad m = 1, 2, \ldots, M \tag{6}$$

$$A \le x \le B \tag{7}$$

The $N \times 1$ vectors A and B represent upper and lower bounds on the design vector x. The inequality constraints have been included as equality constraints by using the following transformation:

$$\psi_k(x) = \phi_k(x) - S_k = 0$$

$$0 \le S_k \le \infty \qquad k = 1, 2, \ldots, K \tag{8}$$

The variables S_k are nonnegative slack variables which will be included in the original set of design variables. Hence the parameter N represents the total number of design variables plus the number of slack variables used in the transformation of (8). The parameter M represents the total number of constraints.

$$M = L + K \tag{9}$$

It should be stressed that the constraints of (6) are nontrivial constraints; that is, functional constraints. Variable bounds are defined in (7) and will require separate handling.

Consider the following strategy. Divide the design vector of equation (5) into two classes which shall be known as the decision and state variables.

$$x = [z, y]^T \tag{10}$$

$$z = [z_1, z_2, z_3, \ldots, z_Q]^T; \qquad \text{decision variables} \tag{11}$$

$$y = [y_1, y_2, y_3, \ldots, y_M]^T; \qquad \text{state variables} \tag{12}$$

$$Q = N - M \tag{13}$$

The decision variables are completely independent, and the state variables are slaves to the decision variables used to satisfy the constraints $\psi_m(x)$.

We can see a motivation for this division of the x vector into decision and state variables based on the elimination technique for handling equality constraints. Given the equality constrained NLP,

Minimize

$$f(x); \qquad x = [x_1, x_2, x_3, \ldots, x_N]^T \tag{14}$$

subject to

$$\psi_l(x) = 0 \qquad l = 1, 2, 3, \ldots, L \tag{15}$$

Reformulate each constraint in (15) to explicitly define a different design variable.

$$x_l = h_l(x) \qquad l = 1, 2, 3, \ldots, L \tag{16}$$

Substitution into the objective function (14) would yield

$$f(h_1, h_2, \ldots, h_L, x_{L+1}, \ldots, x_N) \tag{17}$$

This new objective function becomes an unconstrained function in $N - L$ design variables. Minimization of this new objective function would yield the constrained optimal values of the $N - L$ design variables describing (17) with the remaining design variables calculated by backsubstituting into (16).

In the foregoing technique the decision variables become the $N - L$ design variables describing (17) with the state variables described by (16). We can see here a basic approach for constrained optimization

Nomenclature

$f(x), f$ = objective function

$x = x_1, x_2, \ldots, x_N$; set of design variables

R^N = N-dimensional space of reals

$\phi_k(x), \phi$ = inequality constraint functions

$\psi_l(x), \psi$ = equality constraint functions

N = number of design variables

K = number of inequality constraints

L = number of equality constraints

$P(x)$ = penalty function

$W(x, R, \phi, \psi)$ = [illegible]lty term

S_k = slack variables

M = total number of constraints

Q = number of decision variables

z = Q-dimensional column vector of decision variables

y = M-dimensional column vector of state variables

$L(x, \lambda, \theta)$ = Lagrangian function

B, B_Q = transformation matrix, $M \times Q$

$\nabla f(z)$ = reduced gradient, $Q \times 1$

$P(z)$ = search direction of decision variables, $Q \times 1$

$P(y)$ = search direction providing first approximation of state variables, $M \times 1$

α = step length parameter

y^0 = first approximation of state variables

y^* = adjusted values of state variables

N_f = number of objective function evaluations

N_c = number of constraint set evaluations using the decision and state dichotomy. We can perturb the decision variables in the newly formed unconstrained space to seek the minimum of the objective function while adjusting the state variables to maintain feasibility. However, use of the technique just described presupposes that the state variables can be defined explicitly in terms of the constraint functions. In engineering design problems this is generally the exception and not the rule. Some other method must be devised to utilize the decision and state variable dichotomy when nonlinearity is a factor.

The following notation will be useful in the discussion to follow:

$$g(z) = \left[\frac{\partial f(x)}{\partial z_1}, \frac{\partial f(x)}{\partial z_2}, \ldots, \frac{\partial f(x)}{\partial z_Q}\right]^T$$

$$g(y) = \left[\frac{\partial f(x)}{\partial y_1}, \frac{\partial f(x)}{\partial y_2}, \ldots, \frac{\partial f(x)}{\partial y_M}\right]^T$$

$$\frac{\partial\psi}{\partial z} = \begin{bmatrix} \frac{\partial\psi_1}{\partial z_1} & \frac{\partial\psi_1}{\partial z_2} & \cdots & \frac{\partial\psi_1}{\partial z_Q} \\ \vdots & \vdots & & \vdots \\ \frac{\partial\psi_M}{\partial z_1} & \frac{\partial\psi_M}{\partial z_2} & \cdots & \frac{\partial\psi_M}{\partial z_Q} \end{bmatrix} \quad (M \times Q)$$

$$\frac{\partial\psi}{\partial y} = \begin{bmatrix} \frac{\partial\psi_1}{\partial y_1} & \frac{\partial\psi_1}{\partial y_2} & \cdots & \frac{\partial\psi_1}{\partial y_M} \\ \vdots & \vdots & & \vdots \\ \frac{\partial\psi_M}{\partial y_1} & \frac{\partial\psi_M}{\partial y_2} & \cdots & \frac{\partial\psi_M}{\partial y_M} \end{bmatrix} \quad (M \times M)$$

Let us examine the first variation of $f(x)$ and $\psi(x)$,

$$df = g(z)^T dz + g(y)^T dy \tag{18}$$

$$d\psi = \frac{\partial\psi}{\partial z} dz + \frac{\partial\psi}{\partial y} dy = 0 \tag{19}$$

where

dz = $Q \times 1$ vector of differential displacements of z

dy = $M \times 1$ vector of differential displacements of y

Solving (19) for dy yields,

$$dy = -\frac{\partial\psi}{\partial y}^{-1} \frac{\partial\psi}{\partial z} dz \tag{20}$$

Substituting (20) into (8) and rearranging will yield the following linear approximation to the *reduced gradient:*

$$g_r(z)^T = g(z)^T - g(y)^T \frac{\partial\psi}{\partial y}^{-1} \frac{\partial\psi}{\partial z} \tag{21}$$

The reduced gradient defines the rate of change of the objective function with respect to the decision variables with the state variables adjusted to maintain feasibility. Expression (20) gives the changes necessary in the states for a given change in the decisions for linear constraints. Geometrically the reduced gradient can be described as a projection of the original N-dimensional gradient onto the $(N - M)$-dimensional feasible region described by the decision variables.

A necessary condition for the existence of a minimum of an unconstrained nonlinear function is that the elements of the gradient vanish. Similarly, a minimum of the constrained nonlinear function occurs when the appropriate elements of the reduced gradient vanish. This conclusion can be verified by a comparison with the Kuhn-Tucker [11] conditions for the existence of a constrained relative minimum.

By first transforming the variable bounds into inequality constraints,

$$\begin{rmatrix}\end{rmatrix}$$

$$\left.\begin{aligned} \phi_i(x) &= x_i - a_i \ge 0 \\ \phi_{N+i}(x) &= b_i - x_i \ge 0 \end{aligned}\right| \quad i = 1, 2, 3, \ldots, N \tag{22}$$

we can form the following Lagrangian function,

$$L(x, u, w) = f(x) + \sum_{m=1}^{M} w_m \psi_m(x) - \sum_{j=1}^{2N} u_j \phi_j(x) \tag{24}$$

The following Kuhn-Tucker necessary conditions hold for a point to be a relative minimum x^*,

$$g(x^*)^T + \sum_{m=1}^{M} w_m^* \frac{\partial\psi_m}{\partial x} - \sum_{j=1}^{J} u_j^* \frac{\partial\phi_j}{\partial x} = 0 \tag{25}$$

and

$$\psi_m(x^*) = 0 \qquad m = 1, 2, \ldots, M \tag{26}$$

$$\phi_j(x^*) \ge 0 \qquad j = 1, 2, \ldots, J = 2N \tag{27}$$

$$u_j^* \phi_j(x^*) = 0 \qquad j = 1, 2, \ldots, J = 2N \tag{28}$$

$$u_j^* \ge 0 \qquad j = 1, 2, \ldots, J = 2N \tag{29}$$

$$w_m^* \ne 0 \qquad m = 1, 2, \ldots, M \tag{30}$$

Introducing decision and state variables into (25) and decomposing we can obtain the following form:

$$g(z^*)^T + w^{*T} \frac{\partial\psi}{\partial z^*} - u^{*T} \frac{\partial\phi}{\partial z^*} = 0 \tag{31}$$

$$g(y^*)^T + w^{*T} \frac{\partial\psi}{\partial y^*} - u^{*T} \frac{\partial\phi}{\partial y^*} = 0 \tag{32}$$

For reasons to be explained in the next section, a state variable is not allowed to be equal or sufficiently close to either of its bounds. From expression (28) the elements of u^* corresponding to the state variable bounds must be zero. Also those elements of $\partial\phi/\partial y$ corresponding to the decision variable bounds will be zero eliminating the last term of (32). Solving (32) for w^* and substituting into (31) will produce the following expression:

$$g(z^*)^T - g(y^*)^T \frac{\partial\psi}{\partial y^*}^{-1} \frac{\partial\psi}{\partial z^*} - u^{*T} \frac{\partial\phi}{\partial z^*} = 0 \tag{33}$$

Rearranging (33) we obtain

$$u^{*T} \frac{\partial\phi}{\partial z^*} = g(z^*)^T - g(y^*)^T \frac{\partial\psi}{\partial y}^{-1} \frac{\partial\psi}{\partial z} \tag{34}$$

We recognize the right-hand side of (34) to be $g_r(z)$. By examining the possible values of the left-hand side of (34), a candidate point z will be z^* if

$$g_r(z)_i \begin{cases} > 0 \text{ if } z_i = a_i \\ < 0 \text{ if } z_i = b_i \\ = 0 \text{ if } a_i \le z_i \le b_i \end{cases}$$

$$i = 1, 2, 3, \ldots, Q \tag{35}$$

Reduced Gradient—Algorithm

A flow chart of the Reduced Gradient Method to be described here is shown in Fig. 1. The basic steps involved are similar to those described by Abadie. The major differences described here occur in the calculation of the reduced gradient and the line search. Also it should be noted here that the present implementation of the method uses numerical techniques to obtain the required derivatives. In the results reported by Abadie, analytical derivatives were employed. However, obtaining analytical derivatives in many engineering applications can prove to be a tedious task and is often prone to error. In the following sections, implementation and computational considerations for each step in Fig. 1 will be discussed.

1 Specification of State and Decision Variables. In most

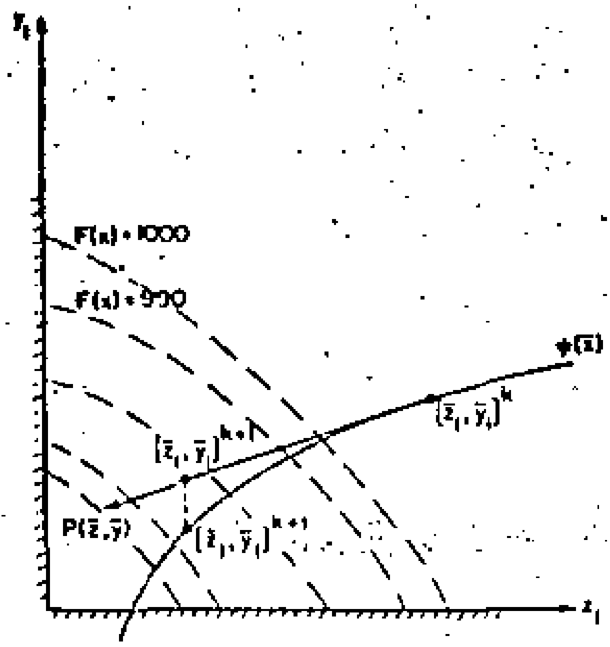

y_1
F(x) = 1000
F(x) = 900
$\psi(\bar{x})$
$[\bar{z}_1, \bar{y}_1]^k$
$[\bar{z}_1, \bar{y}_1]^{k+1}$
$P(\bar{z}, \bar{y})$
$[\bar{z}_1, \bar{y}_1]^{k+1}$
z_1

Fig. 2 Adjustment of state variable to obtain a feasible point during the linear search

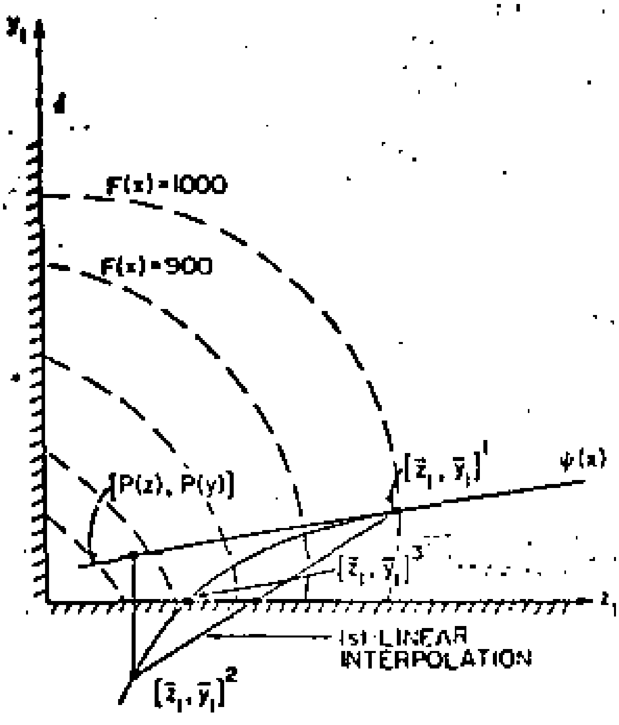

y_1
F(x) = 1000
F(x) = 900
$[P(z), P(y)]$
$[\bar{z}_1, \bar{y}_1]^1$
$\psi(x)$
$[\bar{z}_1, \bar{y}_1]^3$
z_1
1st LINEAR INTERPOLATION
$[\bar{z}_1, \bar{y}_1]^2$

Fig. 3 Adjustment of state variable to locate a feasible point

and

$$z_m^{k+1} = z_m^k + \alpha P(z_m) \qquad m = 1, 2, 3, \ldots, M \qquad (13)$$

where α = step length parameter.

Because of nonlinearities arising in the constraint functions, the point $[z, y]^{k+1}$ is likely to be infeasible. Holding the decision variables z^{k+1} constant, the state variables y^{k+1} are adjusted to obtain a feasible point, $[z, y]^{k+1}$. This situation is shown in Fig. 2 for the case $M = 1$, $Q = 1$. This step is equivalent to the solution of M nonlinear equations $\psi(x) = 0$ in M unknowns (y). A number of numerical techniques exist to perform this task. Newton's method [15] has proven to be an efficient technique as well as convenient since the necessary partial derivatives have already been calculated.

At the completion of the adjustment procedure a new point $[z, y]^{k+1}$ has been determined and the following possible results must be considered:

(a) If all elements of y^{k+1} are within their specified bounds then $f(x)$ is evaluated at $[z, y]^{k+1}$ and the procedure for determining the minimum continues in the normal manner.

(b) If any element of y^{k+1} is not within its specified bounds, then $[z, y]^{k+1}$ is infeasible. Successive linear interpolation is performed between the last feasible point $[z, y]^k$ and the point $[z, y]^{k+1}$ to determine the step length at which the nearest bound becomes active. Hence this step should conclude with a single state variable equal to one of its bounds and all other state variables within their specified bounds. Fig. 3 shows $[z_1, y_1]^2$ being out of bounds ($y_1 < 0$). Using successive false position, the bound point $[z_1, y_1]^3$ can be located. Supplementary tests are then performed to determine whether the local minimum lies at the bound or at some point before it. If the minimum lies at the bound, the line search is terminated. If it lies before the bound, then the minimum has been bracketed and refinement can be tested to locate the minimum.

(c) If the procedure fails to converge in a reasonable amount of time, the step length is reduced to $\alpha = \alpha/2$ and a new trial point generated.

6 Respecify State and Decision Variables. We see in the previous step that it was possible to end a line search with a state variable equal to one of its specified bounds (second result listed previously). To start another iteration the guidelines initially set forth for state variable selection must be satisfied. These state variables that are bounded must be exchanged with decision variables that are unbounded and will not cause $\partial\psi/\partial y$ to become singular.

Assume that y_s is a bound state variable. Abadie [16] suggests the following expression be *maximized* to determine the appropriate decision variable to exchange with y_s.

$$\text{Min}\,(|D_{si}|(b_i - z_i),\ |D_{si}|(z_i - a_i)) \qquad i = 1, 2, \ldots, Q \qquad (14)$$

where

$$D_{si} = \text{element of } \frac{\partial \psi}{\partial y}^{-1} \frac{\partial \psi}{\partial z} \qquad (15)$$

Further Considerations and Recommendations

In the previous section the basic steps of a reduced gradient algorithm are described. In this section we would like to recommend some improvements and point out further modifications that can be made to the basic algorithm.

As with most nonlinear programming techniques, a large amount of effort is expended during the line search. Because of the reduced gradient's use of the state and decision variable dichotomy, much time is spent during the line search performing iterations of Newton's method to adjust the state variables. This time can be reduced through a modification of Newton's method given here:

$$y^{t+1} = y^t - \frac{\partial \psi}{\partial y}^{-1} \psi(z^k, y^t) \qquad (16)$$

where

$\dfrac{\partial \psi}{\partial y}^{-1}$ = the initial inverse used at the start to calculate the reduced gradient.

t = tth iteration of Newton's method

Although this modification does not possess the convergence rate of the more classical method, it does offer a considerable savings in time by avoiding successive reformulations of the Jacobian inverse.

The most stringent requirement of Newton's method is the need for a good initial approximation to the solution. Without it there is no guarantee of convergence. Abadie has suggested linearization of the constraints at the start to provide this initial approximation. This was described in the previous section. The following procedure offers a slight improvement:

1 From the initial starting point of the line search $[z, \bar{y}]^0$, a step α_1 is taken along the search directions with the approximation of the new state variables taken as before

$$\bar{y}^1 = \bar{y}^0 + \alpha_1 l^0(\bar{y})$$

2 Locate the feasible point $\bar{y}^1$ by Newton's method.

3 All other state variable approximations are generated by the following pattern step whenever Newton's method converges for the previous point

$$\bar{y}^k = \bar{y}^0 + \frac{\alpha_k}{\alpha_{k-1}}(\bar{y}^{k-1} - \bar{y}^0) \qquad (47)$$

where

$\bar{y}^{k-1}$ = the feasible point obtained at the previous step

α_{k-1} = the step length used for $[z, \bar{y}]^{k-1}$.

This improvement in providing a starting point for Newton's method decreases the number of Newton iterations required during the bounding phase, but is more readily exhibited during an iteration to a bound or a local minimum. Both approaches are pictured in Fig. 4.

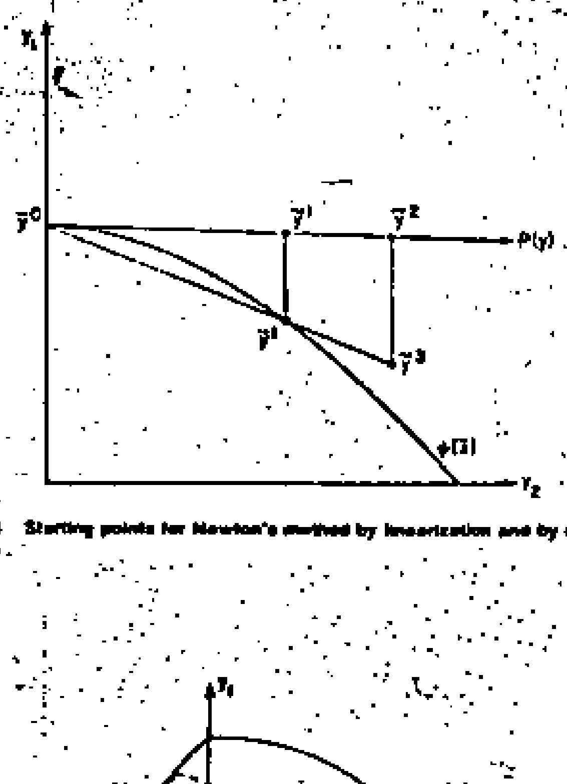


Fig. 4 Starting points for Newton's method by linearization and by direct move

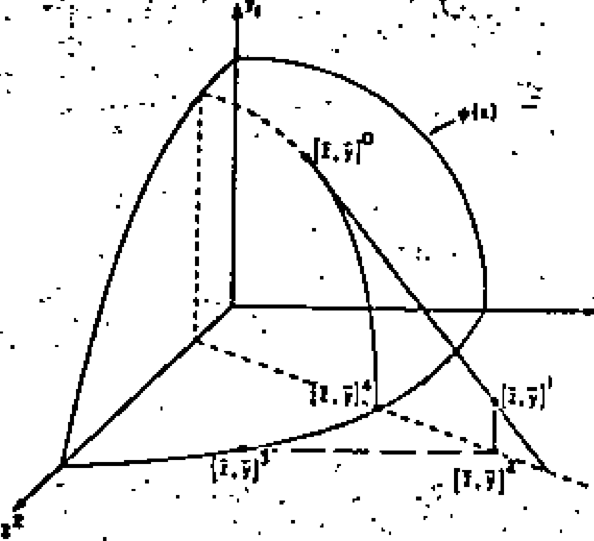


Fig. 5 State variable adjustment strategies

The procedure outlined in (6) in the previous section on line searching differs from an approach that has been suggested by Abadie. In Fig. 5, we move from the point $[z, \bar{y}]^1$ to the point $[z, \bar{y}]^2$ as y_1 is adjusted. Since y_1 is nearly violating its lower bound, Abadie suggests the following scheme. Set y_1 to its lower bound and respecify it as a decision variable, choosing an appropriate decision variable to take its place. Now adjust the new state variable to maintain feasibility. This would result in the point $[z, y]^3$ shown in Fig. 5 with y_1 exchanged with z_1. Following the scheme given previously would result in the point $[z, y]^4$.

Both approaches seem to have advantages and it is unclear which approach is best. The main advantage of Abadie's line search is that it may more quickly locate those points where constraints intersect. However, if the modified Newton's method described previously is used, then a reformulation of the Jacobian inverse would be required every time a state variable was respecified. The approach described here leads to a less complicated algorithm and avoids frequent reformulation of the Jacobian inverse.

Results of Numerical Experiments With OPT

OPT is the name of our Reduced Gradient Fortran code, which implements the theory discussed in the previous sections. We report here testing of the program, which involved application of the code to each of the Eason and Fenton test problems. Eason and Fenton gave results for ten problems, three of which were originally given by Colville. Four of these ten problems are essentially unconstrained, in that the constraints are neither active at the solution nor encountered during the search. If constraints had been encountered during the search, the numerical results would have been useful. Since they were not, the results indicate the performance of the unconstrained searching scheme only, which was not the subject of this study. Accordingly, these problems were not considered further. OPT has no difficulty in identifying interior optima, as is demonstrated in our solution of Eason and Fenton problem number 6. It is certainly worthy of note that the basic concept of the reduced gradient, the state-decision dichotomy, can be employed with any unconstrained searching algorithm. But this will be the subject of later results.

Table 1 gives the results obtained with OPT on Eason and Fenton problems number 1-3, 6, 7, and 9 [1]. In each case OPT terminated at a point which matched or exceeded the convergence criteria specified by Eason and Fenton, using standard program parameters. Listed is the number of objective and constraint function evaluations, and standard time needed to achieve solution.

OPT will terminate whenever the L_2 norm of the projected reduced gradient is less than 10^{-4}, or when the L_2 norm of the change in the design variables is less than 10^{-5}. The gradient condition caused termination in each case except problem 9. The line searches in OPT are conducted to an accuracy of 10^{-6}, and the tight constraints are satisfied to within 10^{-6} at every step of the search. Note that OPT found solutions for all of the problems. Table 2 gives the number of

Table 1

Problem Number	N_f	N_g	Standard Time
1	54	121	0.0157
2	24	65	0.0065
3	16	114	0.0416
6	47	641	0.0012
7	77	147	0.0210
9	44	444	[illegible]

Table 2 Reduced gradient stages required for solution

Problem Number	Stages
1	6
2	2
3	3
6	6
7	3
9	10

Table 3 Relative ranking of optimization codes

N_s		I_1		I_2		T_s	
8	OPT	1.	OPT	.013	OPT	2.31	OPT
5.5	ADRANS	18.11	APPROX	.30	APPROX	6.9	APPROX
	PATSH	99.15	SEEK1	.47	PATSH	10.2	SEEK1
	SIMPLX	194.10	PATSH	.60	SEEK1	11.9	PATSH
4	SEEK1	139.14	MEMGRD	.71	MEMGRD	12.6	SIMPLX
3.5	APPROX	191.44	DAVID	.83	DAVID	14.6	ADRANS
3.5	MEMGRD	300.44	SIMPLX	1.10	SIMPLX	15.9	DAVID
3	DAVID	479.14	ADRANS	1.94	ADRANS	16.8	MEMGRD

reduced gradient stages required to find each solution. Table 3 contains the rating schemes proposed by Eason and Fenton. N_s is the number of problems solved. The Eason and Fenton results were recomputed for the six problems considered here, and codes with an N_s value greater than or equal to three are included in Table 3. I_1 and I_2 are indicators of code efficiency defined by Eason and Fenton. T_s is the sum of the normalized times for each code on all problems. Our CDC 6500 system is rated at 51 s using Colville's standard timing program. Problems 6 and 7 have infeasible starting points. OPT handles this difficulty with the artificial variable approach suggested by Abadie. The artificial variable is used only to find an initial feasible starting point and plays no part in the remainder of the search.

The excellent performance demonstrated by OPT is the result of the ease with which the generalized reduced gradient algorithm identifies and follows active constraints. Additionally, the performance has been enhanced by careful attention to detail in the program implementation. Our experience indicates that even the most outstanding algorithm can perform poorly if the programming stage is improperly attended to.

Additional numerical results are reported in the thesis by Gabriele [17].

Conclusions

The following conclusions can be drawn from this work:

1 The state and decision variable dichotomy helps to reduce the constrained nonlinear programming problem to a simpler, more readily solved problem. By using the dichotomy the problem becomes, in simple terms, a search in the reduced unconstrained feasible region of the decision variables while maintaining feasibility through state variable adjustment.

2 The reduced gradient is a reliable and efficient tool for the solution of a wide variety of nonlinear programming problems, since OPT receives the top rating in each of the categories listed.

Acknowledgments

The authors are indebted to Honeywell Corporation for partial financial support of this project, and to the Purdue University Computing Center for their supply of computing power.

References

1 Eason, E. D., and Fenton, R. G., "A Comparison of Numerical Optimization Methods for Engineering Design," JOURNAL OF ENGINEERING FOR INDUSTRY, TRANS. ASME, Series B, Vol. 96, No. 1, Feb. 1974, pp. 196–200.

2 Colville, A. R., "A Comparative Study of Nonlinear Programming Codes," *Proceedings of the Princeton Symposium on Mathematical Programming*, Kuhn, H. W., ed., Princeton, N. J., 1970, pp. 487–501.

3 Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming*, Wiley, New York, 1968.

4 Griffith, R. E., and Stewart, R. A., "A Nonlinear Programming Technique for the Optimization of Continuous Processing Systems," *Management Science*, Vol. 7, 1961, pp. 379–392.

5 Fox, R. L., *Optimization Methods for Engineering Design*, Addison-Wesley, 1971.

6 Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw-Hill, New York, 1972.

7 Wolfe, P., "Methods for Linear Constraints," *Nonlinear Programming*, Abadie, J., ed., North Holland, Amsterdam, 1967, pp. 99–131.

8 Wolfe, P., "Methods of Nonlinear Programming," *Recent Advances in Mathematical Programming*, Graves, R. L., and Wolfe, P., eds., McGraw-Hill, New York, 1963, pp. 76–77.

9 Abadie, J., and Carpenter, J., "Generalization of the Wolfe Reduced Gradient Method to the Case of Nonlinear Constraints," *Optimization*, Fletcher, R., ed., Academic Press, 1969, p. 37.

10 Wilde, D. J., and Beightler, C., *Foundations of Optimization*, Prentice-Hall, Englewood Cliffs, N. J., 1967, Chapters 2,3.

11 Kuhn, H. W., and Tucker, A. W., "Nonlinear Programming," *Proceedings, 2nd Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, Berkeley, 1951, pp. 481–492.

12 Fletcher, R., and Reeves, C. M., "Function Minimization by Conjugate Gradients," *Computer Journal*, Vol. 6, No. 2, 1964, pp. 163–168.

13 Davidon, W. C., "Variable Metric Method for Minimization," Argonne National Laboratory, ANL-5990 Rev., University of Chicago, 1959.

14 Ragsdell, K. M., "A Survey of Some Useful Optimization Methods," *Proceedings of the Design Engineering Technical Conference*, New York, 1972, pp. 129–135.

15 Ragsdell, K. M., Root, R. R., and Gabriele, G. A., "Newton's Method: A Classical Technique of Contemporary Value," *Proceedings of the Design Engineering Technical Conference*, New York, 1974, pp. 137–145.

16 Abadie, J., "Application of the GRG Algorithm to Optimal Control Problems," *Integer and Nonlinear Programming*, Abadie, J., ed., North Holland, Amsterdam, 1970, p. 191.

17 Gabriele, G. A., "Application of the Reduced Gradient Method to Optimal Engineering Design," Master's thesis, School of Mechanical Engineering, Purdue University, Dec. 1975.

Printed in U.S.A.

Reprinted from Journal of Optimization Theory and Applications

Vol. 4, No. 4, October 1969
Printed in Belgium

Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions—Ordinary and Conjugate Gradient Versions[1]

A. Miele,[2] H. Y. Huang,[3] and J. C. Heideman[4]

Abstract. The problem of minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$ is considered. Here, f is a scalar, x an n-vector, and φ a q-vector. A *sequential algorithm* is presented, composed of the alternate succession of gradient phases and restoration phases.

In the *gradient phase*, a nominal point x satisfying the constraint is assumed; a displacement Δx leading from point x to a varied point y is determined such that the value of the function is reduced. The determination of the displacement Δx incorporates information at only point x for the *ordinary gradient* version of the method (Part 1) and information at both points x and $\hat{x}$ for the *conjugate gradient* version of the method (Part 2).

In the *restoration phase*, a nominal point y not satisfying the constraint is assumed; a displacement Δy leading from point y to a varied point $\tilde{x}$ is determined such that the constraint is restored to a prescribed degree of accuracy. The restoration is done by requiring the least-square change of the coordinates.

If the stepsize α of the gradient phase is of $O(\varepsilon)$, then $\Delta x = O(\varepsilon)$ and $\Delta y = O(\varepsilon^2)$. For ε sufficiently small, the restoration phase preserves the descent property of the gradient phase: the function f decreases between any two successive restoration phases.

[1] Paper received February 3, 1969. This research, supported by the NASA Manned Spacecraft Center, Grant No. NGR-44-006-089, and by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No. AF-AFOSR-828-67, is a continuation of the investigations reported in Refs. 1 and 2.

[2] Professor of Astronautics and Director of the Aero-Astronautics Group, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[3] Assistant Professor of Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Department of Mechanical and Aerospace Engineering and Materials Science, Rice University, Houston, Texas.

The ordinary gradient version of the algorithm exhibits asymptotic convergence, but not quadratic convergence. On the other hand, the conjugate gradient version exhibits quadratic convergence in the neighborhood of the minimum point. In particular, for a quadratic function subject to a linear constraint, the minimum point is obtained in no more than $n - q$ iterations.

1. Introduction

In recent years, considerable attention has been given to the iterative algorithms for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$, where f is a scalar, x an n-vector, and φ a q-vector. With reference to the gradient method, one possible approach is that of the penalty functions. The advantage of this approach is that the constrained minimal problem is replaced by a mathematically simpler, unconstrained minimal problem. The disadvantages are these: no clear-cut method exists for choosing the penalty constants; the algorithm must be repeated several times for increasing values of the penalty constants; the values of the function between iterations are not comparable, since the constraints are not satisfied; and, even when the algorithm is terminated, the constraints are generally not satisfied.

In this paper, we present a *sequential algorithm* constructed in such a way that the values of the function $f(x)$ between iterations are comparable. The algorithm is composed of the alternate succession of gradient phases and restoration phases (Fig. 1). In the *gradient phase*, a nominal point x satisfying the constraint is assumed; then, a displacement Δx leading from point x to a varied point y is determined by minimizing the first variation $\delta f(x)$ subject to the linearized constraint $\delta\varphi(x) = 0$ and a quadratic constraint on Δx. Due to the fact that the constraint is accounted for only to first order, the varied point y may be such that $\varphi(y) \neq 0$. This being the case, a restoration phase is needed prior to starting the next gradient phase. In this *restoration phase*, a nominal point y not satisfying the constraint is assumed; then, a displacement Δy leading from point y to a varied point $\tilde{x}$ is determined by minimizing the distance $\Delta y^T \Delta y$ subject to the linearized constraint $k\varphi(y) + \delta\varphi(y) = 0$, where $0 < k \leq 1$. If a single restoration cycle fails to produce the required degree of accuracy in the constraint, several cycles must be employed, as shown in Fig. 2. After the restoration phase is finished, the iteration is completed, and the next iteration is started using $\tilde{x}$ as the nominal point x.

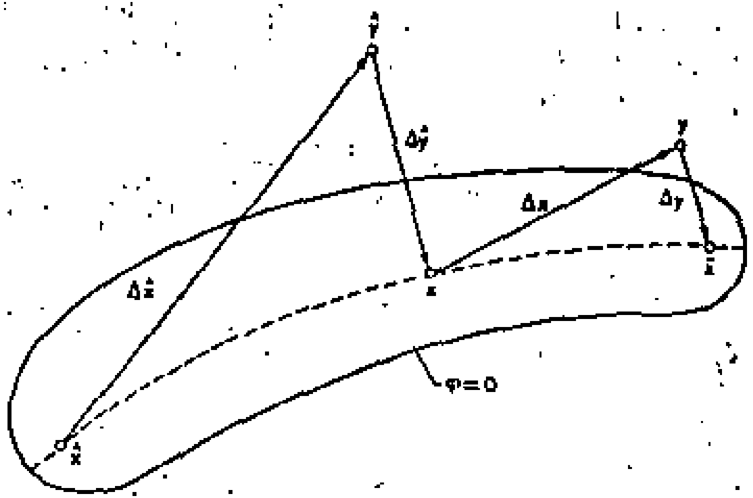


Fig. 1

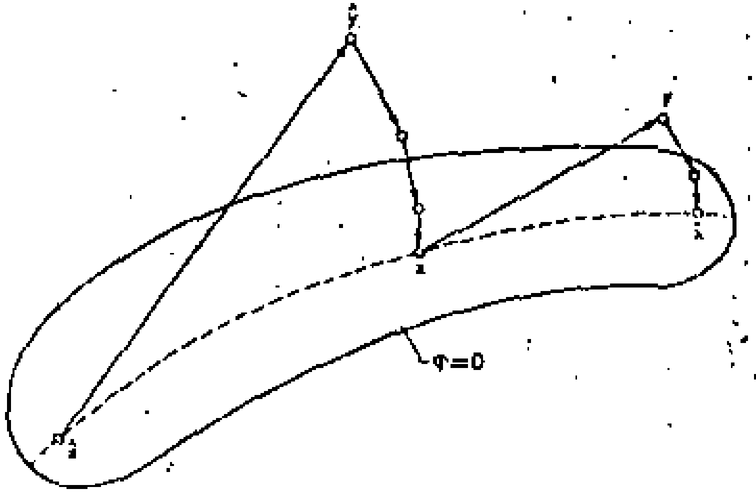


Fig. 2

Two versions of the method are presented: (a) the *sequential ordinary gradient-restoration algorithm*, in which the gradient phase uses information at point x only, and (b) the *sequential conjugate gradient-restoration algorithm*, in which the gradient phase uses information at both points x and $\hat{x}$, where $\hat{x}$ is the point preceding x. Method (a) is given in Part 1; Method (b) is given in Part 2; then, in Part 3, several numerical examples are presented.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 \tag{2}$$

In the above equation, f is a scalar, x an n-vector, and φ a q-vector,[4] where $q < n$. It is assumed that the first and second partial derivatives of the function f and the constraint φ with respect to x exist and are continuous; it is also assumed that the constrained minimum exists.

PART 1: SEQUENTIAL ORDINARY GRADIENT-RESTORATION ALGORITHM

In this part, we present the ordinary gradient version of the sequential gradient-restoration algorithm. The gradient phase is treated in Section 3 and the restoration phase in Section 4. In Section 5, an order-of-magnitude analysis is presented. Finally, in Section 6, the sequential ordinary gradient-restoration algorithm is summarized.

3. Gradient Phase

Consider a displacement Δx leading from a nominal point x to a varied point y such that

$$y = x + \Delta x \tag{3}$$

[4] All the vectors in this paper are column vectors.

Assume that the nominal point x satisfies (2) exactly and that the varied point y satisfies (2) to first order. The first-order change of the function (1) is given by

$$\delta f(x) = f_x^T(x)\, \Delta x \tag{4}$$

where $f_x(x)$ is the gradient of the scalar function f with respect to the vector x and the symbol T denotes the transpose of a matrix. In turn, the first-order change of the constraint (2) is represented as

$$\delta\varphi(x) = \varphi_x^T(x)\, \Delta x = 0 \tag{5}$$

where $\varphi_x(x)$, an $n \times q$ matrix, denotes the gradient of the vectorial function φ with respect to the vector x.

Next, consider the following quadratic constraint on the displacement Δx:

$$K = \Delta x^T \Delta x \tag{6}$$

where K is a constant. With this understanding, we formulate the following problem: Find the variation Δx which minimizes (4) subject to (5) (6).

3.1. Displacement Δx.

Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is the scalar function

$$\Omega = f_x^T(x)\, \Delta x + \lambda^T \varphi_x^T(x)\, \Delta x + (1/2\alpha)\, \Delta x^T \Delta x \tag{7}$$

where $1/2\alpha$ is a scalar Lagrange multiplier and λ a q-vector Lagrange multiplier. If one introduces the *augmented function*

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x) \tag{8}$$

and observes that

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\, \lambda \tag{9}$$

the fundamental function (7) becomes

$$\Omega = F_x^T(x, \lambda)\, \Delta x + (1/2\alpha)\, \Delta x^T \Delta x \tag{10}$$

In Eqs. (9)–(10), the symbol $F_x(x, \lambda)$ denotes the gradient of the augmented function F with respect to the vector x. The optimal displacement Δx satisfies the relation

$$\Omega_{\Delta x} = 0 \tag{11}$$

where $\Omega_{\Delta x}$ denotes the gradient of the fundamental function Ω with respect to the vector Δx. The explicit form of (11) is the following:

$$\Delta x = -\alpha F_x(x, \lambda) \tag{12}$$

and shows that the displacement vector Δx has the same direction as the gradient of the augmented function F. Note that (12) determines Δx providing λ, α are specified.

3.2. Relation Between K and α. As Eq. (12) shows, the displacement Δx is proportional to α, the *stepsize* of the gradient phase. Upon substituting (12) into (6), we see that

$$K = \alpha^2 F_x^T(x, \lambda) F_x(x, \lambda) \tag{13}$$

Therefore, a correspondence exists between the values of the constant K and the values of the stepsize α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

3.3. Determination of λ. If Eqs. (5), (9), (12) are combined, we obtain the relation

$$\varphi_x^T(x) f_x(x) + \varphi_x^T(x) \varphi_x(x) \lambda = 0 \tag{14}$$

Since the vector $f_x(x)$ and the matrix $\varphi_x(x)$ are known at the nominal point x, Eq. (14) supplies the multiplier λ; this is precisely the value which guarantees satisfaction of the constraint (2) to first order.

3.4. Descent Property of the Gradient Phase. The first variation of the augmented function is given by

$$\delta F(x, \lambda) = F_x^T(x, \lambda) \Delta x \tag{15}$$

which, in the light of (12), can be written as

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{16}$$

Equation (16) shows that, for $\alpha > 0$, the first variation of the augmented function is negative. Therefore, if α is sufficiently small, the augmented function F decreases during the gradient phase.

Alternatively, the first variation of $F(x, \lambda)$ can be written as

$$\delta F(x, \lambda) = \delta f(x) + \lambda^T \delta\varphi(x) \tag{17}$$

Because of (5), Eq. (17) reduces to the form

$$\delta f(x) = \delta F(x, \lambda) \tag{18}$$

which states that the functions $f(x)$ and $F(x, \lambda)$ behave identically, to first order. Therefore, the descent property also holds for the function f.

3.5. Stepsize. The next step is to assign a value to the stepsize α. If Eqs. (3) and (12) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha F_x(x, \lambda) \tag{19}$$

Since λ is known through Eq. (14), Eq. (19) defines a one-parameter family of points y for which the augmented function F takes the form

$$F(y, \lambda) = F(x - \alpha F_x(x, \lambda), \lambda) = \Psi(\alpha) \tag{20}$$

The greatest decrease of the function $\Psi(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$\Psi_\alpha(\alpha) = 0 \tag{21}$$

After observing that

$$\Psi_\alpha(\alpha) = -F_x^T(y, \lambda) F_x(x, \lambda) \tag{22}$$

we see that Eq. (21) can be written as

$$F_x^T(y, \lambda) F_x(x, \lambda) = 0 \tag{23}$$

showing that the gradient of the augmented function at point y is orthogonal to the gradient at point x.

To obtain satisfaction of (21) or (23), some one-dimensional search method must be employed. In particular, *cubic interpolation* (it employs first derivatives only) and *quasilinearization* (it employs both first and second derivatives) are powerful methods (Refs. 1 and 3). These methods are to be employed iteratively until Eq. (21) is satisfied to a desired degree of accuracy, that is, until

$$\Psi_\alpha^2(\alpha) \leq \theta_1 \tag{24}$$

where θ_1 is a small number. In practice, θ_1 can be fixed or one may choose

$$\theta_1 = \varepsilon_1 \Psi_\alpha^2(0) \tag{25}$$

where ε_1 is a small number.

4. Restoration Phase

At the end of the gradient phase, the point y is known. If the constraint is linear, the relation $\varphi(y) = 0$ holds. On the other hand, if the constraint is nonlinear, $\varphi(y) \neq 0$, which means that some degree of dissatisfaction exists. Therefore, a restoration phase is needed prior to starting the next gradient phase. Specifically, one has to apply a small variation Δy to y to generate a new position vector

$$\tilde{x} = y + \Delta y \tag{26}$$

such that $\varphi(\tilde{x}) = 0$. While there are infinite ways to perform the restoration, the most logical is that developed in Ref. 4: the constraint is restored to a prescribed degree of accuracy with the least-square change of the position vector.

If quasilinearization is employed, Eq. (2) is approximated by

$$\varphi(y) + \varphi_x^T(y)\,\Delta y = 0 \tag{27}$$

In order to prevent the variation Δy from becoming too large, we imbed Eq. (27) into the one-parameter family of equations

$$k\varphi(y) + \varphi_x^T(y)\,\Delta y = 0 \tag{28}$$

where

$$0 \leqslant k \leqslant 1 \tag{29}$$

denotes a scaling factor.

In the light of previous discussion, we seek the minimum of the function

$$J = \tfrac{1}{2}\,\Delta y^T\,\Delta y \tag{30}$$

subject to the linearized constraint (28). Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is given by

$$\omega = \tfrac{1}{2}\,\Delta y^T\,\Delta y + \sigma^T[k\varphi(y) + \varphi_x^T(y)\,\Delta y] \tag{31}$$

where σ, a q-vector, denotes an undetermined, constant Lagrange multiplier. The optimal change Δy satisfies the relation

$$\omega_{\Delta y} = 0 \tag{32}$$

where $\omega_{\Delta y}$ denotes the gradient of the fundamental function ω with respect to the vector Δy. The explicit form of (32) is the following:

$$\Delta y = -\varphi_x(y)\,\sigma \tag{33}$$

The Lagrange multiplier σ is obtained by combining (28) and (33) to eliminate Δy. This yields the relation

$$k\varphi(y) - \varphi_x^T(y)\,\varphi_x(y)\,\sigma = 0 \tag{34}$$

For any given k in the range (29), Eq. (34) supplies the Lagrange multiplier vector σ. Once σ is known, the correction Δy is given by (33) and the corrected position vector $\tilde{x}$ is given by (26). Of course, the restoration phase must be performed iteratively until a desired degree of accuracy is obtained, that is, until the inequality

$$P(\tilde{x}) \leqslant \theta_2 \tag{35}$$

is satisfied, where θ_2 is a small number and the performance index

$$P(x) = \varphi^T(x)\,\varphi(x) \tag{36}$$

measures the error in the constraint.

4.1. Descent Property of the Performance Index. The first variation of the performance index $P(y)$ is given by

$$\delta P(y) = 2\varphi^T(y)\,\varphi_x^T(y)\,\Delta y \tag{37}$$

and, because of Eq. (28), reduces to

$$\delta P(y) = -2k\varphi^T(y)\,\varphi(y) \tag{38}$$

which, in turn, can be written as

$$\delta P(y) = -2kP(y) \tag{39}$$

Since $P(y) > 0$, Eq. (39) shows that the first variation of the performance index is negative for $k > 0$. Therefore, if k is sufficiently small, the decrease of the performance index is guaranteed. In practice, one can use $k = 1$. If this value of k does not result in a decrease in P, then k is successively bisected until P is decreased.

5. Order-of-Magnitude Analysis

The position vector $\tilde{x}$ at the end of the restoration phase and the position vector x at the beginning of the gradient phase are related by

$$\tilde{x} = x + \Delta x + \Delta y \tag{40}$$

where Δx is the gradient displacement and Δy is the restoration displacement. From Eq. (12), we see that, if the stepsize α is of $O(\epsilon)$, the gradient displacement has the order

$$\Delta x = O(\epsilon) \tag{41}$$

We observe that, to second order,

$$\varphi(y) = \varphi(x) + \varphi_x^T(x)\,\Delta x + \tfrac{1}{2}\,\Delta x^T \varphi_{xx}(x)\,\Delta x \tag{42}$$

where $\varphi_{xx}(x)$ denotes the array of the second partial derivatives of φ. Since the nominal point x satisfies (2) exactly and the variation Δx satisfies (2) to first order, the first two terms on the right-hand side of (42) vanish. Therefore, in the light of (41)-(42), we conclude that

$$\varphi(y) = O(\epsilon^2) \tag{43}$$

Next, we turn our attention to Eqs. (33)-(34) and observe that, because of (43),

$$\Delta y = O(\epsilon^2) \tag{44}$$

In conclusion, if the gradient displacement is of $O(\epsilon)$, the restoration displacement is of $O(\epsilon^2)$. This guarantees that, for sufficiently small ϵ,

$$|\Delta y| \ll |\Delta x| \tag{45}$$

5.1. Descent Property of the Algorithm. Finally, we consider the points x and $\tilde{x}$, both satisfying the constraint (2). To first order, the difference of the values of the function f at these points is given by

$$f(\tilde{x}) - f(x) = f_x^T(x)(\Delta x + \Delta y) \tag{46}$$

On account of (45), the second term on the right-hand side of (46) can be neglected with respect to the first. Hence, Eq. (46) becomes

$$f(\tilde{x}) - f(x) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{47}$$

Therefore, for ϵ sufficiently small, the restoration algorithm preserves the descent property of the gradient algorithm: the function f decreases between any two successive restoration phases.

6. Summary of the Algorithm

The algorithm presented in Part 1 consists of the alternate succession of gradient phases and restoration phases. A summary of the algorithm is given below.

6.1. Gradient Phase. For this phase, the algorithm is represented by

$$\begin{aligned} \lambda &= -[\varphi_x^T(x)\,\varphi_x(x)]^{-1}\,\varphi_x^T(x)\,f_x(x) \\ F_x(x, \lambda) &= f_x(x) + \varphi_x(x)\,\lambda \\ \Delta x &= -\alpha F_x(x, \lambda) \\ y &= x + \Delta x \end{aligned} \tag{48}$$

The sequence of operations is as follows: (a) select a nominal point x such that $\varphi(x) = 0$; (b) at this point, determine the vector $f_x(x)$ and the matrix $\varphi_x(x)$; (c) compute the multiplier λ with (48-1) and the vector $F_x(x, \lambda)$ with (48-2); (d) determine the optimal stepsize α by a one-dimensional search in which either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y, \lambda)$ is minimized along the search direction $F_x(x, \lambda)$; the search is terminated when Ineq. (24) is satisfied; (e) compute the displacement Δx with (48-3) and the varied point y with (48-4).

6.2. Restoration Phase. For this phase, the algorithm is represented by

$$\begin{aligned} \sigma &= k[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\,\varphi(y) \\ \Delta y &= -\varphi_x(y)\,\sigma \\ \tilde{x} &= y + \Delta y \end{aligned} \tag{49}$$

The sequence of operations is as follows: (a) at point y, compute the vector $\varphi(y)$ and the matrix $\varphi_x(y)$; (b) assuming $k = 1$, determine the multiplier σ with (49-1), the displacement Δy with (49-2), and the varied point $\tilde{x}$ with (49-3); (c) if $P(\tilde{x}) < P(y)$, the scaling factor $k = 1$ is acceptable; if $P(\tilde{x}) > P(y)$, the previous value of k must be replaced by some smaller value in the range (29) until the condition $P(\tilde{x}) < P(y)$ is met; this can be achieved through successive bisections of k; (d) return to step (a) and repeat the restoration algorithm using $\tilde{x}$ as the starting point y for the subsequent iteration; (e) terminate the restoration algorithm when the stopping condition (35) is satisfied; (f) once the restoration algorithm is completed, verify the inequality

$$f(\tilde{x}) < f(x) \tag{50}$$

If Ineq. (50) is satisfied, start the next gradient phase. If Ineq. (50) is violated, return to the previous gradient phase and reduce the stepsize α until, after restoration, Ineq. (50) is satisfied.

6.3. Stopping Condition.

The algorithm is terminated when

$$Q(\tilde{x}, \lambda) \leq \theta_2 \tag{51}$$

where θ_2 is a small number and

$$Q(x, \lambda) = F_x^T(x, \lambda) F_x(x, \lambda) \tag{52}$$

measures the error in the optimal conditions.

PART 2: SEQUENTIAL CONJUGATE GRADIENT-RESTORATION ALGORITHM

In this part, we present the conjugate gradient version of the sequential gradient-restoration algorithm. The gradient phase is discussed in Section 7 in general; the case of a quadratic function subject to a linear constraint is given in Section 8; then, the practical method to be used for a nonquadratic function subject to a nonlinear constraint is given in Section 9. The treatment of the restoration phase is omitted, since it is covered in Section 4. In Section 10, an order-of-magnitude analysis is presented. Finally, in Section 11, the sequential conjugate gradient-restoration algorithm is summarized.

7. Gradient Phase: General Discussion

Consider a displacement Δx leading from a nominal point x to a varied point y given by Eq. (3). Assume that the nominal point x satisfies (2) exactly and that the varied point y satisfies (2) to first order. The first-order change of the function (1) is given by Eq. (4). In turn, the first-order change of the constraint (2) is represented by Eq. (5). Next, consider the following quadratic constraint on the displacement Δx:

$$K = (\Delta x - \beta \Delta \hat{x})^T (\Delta x - \beta \Delta \hat{x}) \tag{53}$$

where K and β are constants and $\Delta \hat{x}$ is the displacement of the previous gradient phase, that is, the displacement leading from the nominal point $\hat{x}$ to the varied point y (see Figs. 1 and 2). With this understanding, we formulate the following problem: Find the displacement Δx which minimizes (4) subject to (5) and (53).

7.1. Displacement Δx.

Standard methods of the theory of maxima and minima show that the *fundamental function* of this problem is the scalar function

$$\Omega = f_x^T(x) \Delta x + \lambda^T \varphi_x^T(x) \Delta x + (1/2\alpha)(\Delta x - \beta \Delta \hat{x})^T (\Delta x - \beta \Delta \hat{x}) \tag{54}$$

where $1/2\alpha$ is a scalar Lagrange multiplier and λ a q-vector Lagrange multiplier. If one introduces the *augmented function* (8) and its gradient (9), the fundamental function (54) becomes

$$\Omega = F_x^T(x, \lambda) \Delta x + (1/2\alpha)(\Delta x - \beta \Delta \hat{x})^T (\Delta x - \beta \Delta \hat{x}) \tag{55}$$

The optimal displacement Δx satisfies the relation (11), whose explicit form is the following:

$$\Delta x = -\alpha F_x(x, \lambda) + \beta \Delta \hat{x} \tag{56}$$

If one defines the search direction p from

$$\Delta x = -\alpha p \tag{57}$$

and observes that, for the previous iteration,

$$\Delta \hat{x} = -\hat{\alpha} \hat{p} \tag{58}$$

the following relation ensues from (56)-(58):

$$p = F_x(x, \lambda) + \gamma \hat{p} \tag{59}$$

where

$$\gamma = \beta \hat{\alpha} / \alpha \tag{60}$$

In conclusion, the displacement Δx during the gradient phase is given by (57), with p governed by Eq. (59), where $\hat{p}$ is known from the previous iteration. Note that (57) and (59) determine Δx providing λ, α, γ are specified.

7.2. Relation Between K and α.

As Eq. (57) shows, the displacement Δx is proportional to α, the *stepsize* of the gradient phase. Upon substituting (56) into (53), we obtain Eq. (13). Therefore, a correspondence exists between the values of the constant K and the values of the stepsize α. This being the case, one can bypass prescribing K and reason directly on α, as in the considerations which follow.

7.3. Determination of λ, α, γ. If Eqs. (5), (9), (57), (59) are combined, we obtain the relation

$$\varphi_x^T(x) f_x(x) + \varphi_x^T(x)\varphi_x(x)\lambda + \gamma\varphi_x^T(x)\hat{p} = 0 \qquad (61)$$

which ensures satisfaction of the constraint (2) to first order. Equation (61) is a linear relation between λ and γ and admits the solution

$$\lambda = -[\varphi_x^T(x)\varphi_x(x)]^{-1}[\varphi_x^T(x) f_x(x) + \gamma\varphi_x^T(x)\hat{p}] \qquad (62)$$

Therefore, the rate of change of λ with respect to γ is given by the vector equation

$$\lambda_\gamma = -[\varphi_x^T(x)\varphi_x(x)]^{-1}\varphi_x^T(x)\hat{p} \qquad (63)$$

The next step is to assign values to α and γ. If Eqs. (3), (57), (59) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha F_x(x,\lambda) - \alpha\gamma\hat{p} \qquad (64)$$

Since λ depends on γ through Eq. (62), Eq. (64) defines a two-parameter family of points y for which the augmented function F takes the form

$$F(y,\lambda) = F(x - \alpha F_x(x,\lambda) - \alpha\gamma\hat{p}, \lambda) = \Psi(\alpha,\gamma). \qquad (65)$$

The greatest decrease of the function $\Psi(\alpha,\gamma)$ occurs if the parameters α, γ satisfy the following necessary conditions:

$$\Psi_\alpha(\alpha,\gamma) = 0, \qquad \Psi_\gamma(\alpha,\gamma) = 0 \qquad (66)$$

After observing that

$$\begin{aligned} \Psi_\alpha(\alpha,\gamma) &= -F_x^T(y,\lambda)p \\ \Psi_\gamma(\alpha,\gamma) &= -\alpha F_x^T(y,\lambda)\hat{p} + [\alpha F_x^T(y,\lambda)\varphi_x(x) - \varphi^T(y)]\lambda_\gamma \end{aligned} \qquad (67)$$

we see that Eqs. (66) can be written as

$$F_x^T(y,\lambda)p = 0, \qquad \alpha F_x^T(y,\lambda)\hat{p} + [\alpha F_x^T(y,\lambda)\varphi_x(x) - \varphi^T(y)]\lambda_\gamma = 0 \qquad (68)$$

In the light of (59) and (62)–(64), Eqs. (68) constitute a system of two scalar equations in the unknowns α and γ. Once the stepsize α and the coefficient γ are known from (68), the multiplier λ follows from (62), the search direction p from (59), the displacement Δx from (57), and the position vector y from (3). Thus, the problem of determining λ, α, γ is solved in principle; computationally, however, further simplifications are needed to make the algorithm practical.

8. Gradient Phase: Quadratic Function, Linear Constraint

Now, consider the particular case of a quadratic function and a linear constraint given in the form

$$f(x) = a + b^T x + \tfrac{1}{2}x^T c x, \qquad \varphi(x) = d + e^T x \qquad (69)$$

where a is a scalar, b is an n-vector, c an $n \times n$ symmetric matrix, d a q-vector, and e an $n \times q$ matrix. Here, all the coefficients are constant. The gradients of the functions f and φ become

$$f_x(x) = b + cx, \qquad \varphi_x(x) = e \qquad (70)$$

Because of the linearity, the constraint is never violated during the gradient phase and, consequently,

$$\Delta y = 0, \qquad \hat{x} = x + \Delta x, \qquad \varphi_x^T(x)\hat{p} = 0 \qquad (71)$$

This being the case, Eq. (62) supplies the following expression for the multiplier:

$$\lambda = -[\varphi_x^T(x)\varphi_x(x)]^{-1}\varphi_x^T(x) f_x(x), \qquad (72)$$

which is now independent of γ, that is,

$$\lambda_\gamma = 0 \qquad (73)$$

The relations (68) optimizing α and γ become

$$F_x^T(y,\lambda)p = 0, \qquad F_x^T(y,\lambda)\hat{p} = 0 \qquad (74)$$

showing that the gradient of the augmented function at point y is orthogonal to both the present and previous search directions. A mathematical consequence of (59) and (74) is that

$$F_x^T(y,\lambda)F_x(x,\lambda) = 0 \qquad (75)$$

showing that the gradients at point y and point x are orthogonal. Furthermore, after laborious manipulations, Eqs. (74)–(75) lead to

$$F_x^T(\hat{x},\hat{\lambda})p = 0, \qquad F_x^T(\hat{x},\hat{\lambda})\hat{p} = 0, \qquad F_x^T(\hat{x},\hat{\lambda})F_x(x,\lambda) = 0 \qquad (76)$$

For a quadratic function subject to a linear constraint, the following relationship can be shown to hold:

$$F_x(y,\lambda) = F_x(x,\lambda) - \alpha cp \qquad (77)$$

Invoking (69)–(77), we see that (74-1) yields the following solution for the optimal stepsize:

$$\alpha = F_x^T(x, \lambda)\, F_x(x, \lambda)/p^T c p \tag{78}$$

Furthermore, (74-2) leads to

$$p^T c \hat{p} = 0 \tag{79}$$

which states that the search directions p and $\hat{p}$ are conjugate with respect to the matrix c. In turn, (79) yields the following explicit solution for γ:

$$\gamma = F_x^T(x, \lambda) F_x(x, \lambda)/F_x^T(\hat{x}, \hat{\lambda}) F_x(\hat{x}, \hat{\lambda}) \tag{80}$$

In conclusion, for a given nominal point x, the multiplier λ is supplied by (72), the coefficient γ by (80), the search direction p by (59), the optimum stepsize α by (78), the displacement Δx by (57), and the position vector y by (3).

8.1. Convergence Properties. For a quadratic function subject to a linear constraint, the following relations can be shown to hold *providing the first step of the algorithm is a gradient step*:

$$F_x^T(x, \lambda) F_x(x_*, \lambda_*) = 0, \qquad F_x^T(x, \lambda)\, p_* = 0, \qquad p^T c p_* = 0 \tag{81}$$

where x_* denotes any state preceding x. Equations (81) can be derived from (76) and (79) through mathematical induction. The first of Eqs. (81) states that the gradient at each point is orthogonal to the gradient at every previous point. The second of Eqs. (81) states that the gradient at each point is orthogonal to the search direction at every previous point. Finally, the third of Eqs. (81) states that the search direction at each point and the search direction at every previous point are conjugate with respect to the constant matrix c; this is why the algorithm is called the *conjugate-gradient algorithm*.

Laborious manipulations, omitted for the sake of brevity, show that the algorithm defined by (3), (57), (59) with λ, α, γ defined by (72), (78), (80) reduces the gradient $F_x(x, \lambda)$ to zero in no more than $n - q$ steps; therefore, the minimum of the function $f(x)$ subject to the constraint $\varphi(x) = 0$ is reached in no more than $n - q$ steps. If the function is unconstrained, that is, if $q = 0$, then the minimum of $f(x)$ is reached in no more than n steps (see Refs. 5 and 6).

9. Gradient Phase: Nonquadratic Function and/or Nonlinear Constraint

If the function $f(x)$ is nonquadratic and/or the constraint $\varphi(x)$ is nonlinear, the relations (72), (78), (80) defining λ, α, γ are not simultaneously valid. We observe that (72) and (80) involve first derivatives only, while (78) involves the second-derivative matrix c. This being the case, we choose to discard (78) and retain (72) and (80). Thus, we employ the algorithm

$$\begin{aligned} \lambda &= -[\varphi_x^T(x)\, \varphi_x(x)]^{-1}\, \varphi_x^T(x)\, f_x(x) \\ F_x(x, \lambda) &= f_x(x) + \varphi_x(x)\, \lambda \\ \gamma &= F_x^T(x, \lambda)\, F_x(x, \lambda)/F_x^T(\hat{x}, \hat{\lambda})\, F_x(\hat{x}, \hat{\lambda}) \\ p &= F_x(x, \lambda) + \gamma \hat{p} \\ \Delta x &= -\alpha p \\ y &= x + \Delta x \end{aligned} \tag{82}$$

which, for the unconstrained case, reduces to the well-known Fletcher–Reeves algorithm (see Refs. 3 and 7). For the constrained case, the justification of the expressions for the multiplier λ and the coefficient γ is presented in Section 10. Once the nominal point x is given, the multiplier λ is determined with (82-1), the gradient $F_x(x, \lambda)$ with (82-2), the coefficient γ with (82-3), and the search direction p with (82-4); for a given stepsize α, the displacement Δx is computed with (82-5) and the varied point y with (82-6). The determination of α is discussed in the following section.

9.1. Stepsize. If Eqs. (82-5) and (82-6) are combined, the position vector at the end of the gradient phase becomes

$$y = x - \alpha p \tag{83}$$

For a given nominal point x, the vector p is known through Eqs. (82-1)–(82-4); therefore, Eq. (83) defines a one-parameter family of points y for which the augmented function $F(y, \lambda)$ takes the form

$$F(y, \lambda) = F(x - \alpha p, \lambda) = \Psi(\alpha) \tag{84}$$

The greatest decrease in the function $\Psi(\alpha)$ occurs if the parameter α satisfies the following necessary condition:

$$\Psi_\alpha(\alpha) = 0 \tag{85}$$

After observing that

$$\Psi_\alpha(\alpha) = -F_x^T(y,\lambda)\,p \tag{86}$$

we see that Eq. (85) can be written as

$$F_x^T(y,\lambda)\,p = 0 \tag{87}$$

showing that the gradient of the augmented function at point y is orthogonal to the search direction p.

To obtain satisfaction of (85) or (87), some one-dimensional search method must be employed. In particular, *cubic interpolation* (it employs first derivatives only) and *quasilinearization* (it employs both first and second derivatives) are powerful methods (Refs. 1-3). These methods are to be employed iteratively until Eq. (85) is satisfied to a desired degree of accuracy, that is, until Ineq. (24) is satisfied.

9.2. Starting the Algorithm. The algorithm (82) requires that the search direction $\hat{p}$ be known from the previous iteration. Since this is not the case for the first iteration, some assumption concerning $\gamma\hat{p}$ is needed in order to start the algorithm. To retain quadratic convergence, one must choose $\hat{p} = 0$ or $\gamma = 0$. Consequently, for the first step, the search direction (82-4) becomes

$$p = F_x(x,\lambda) \tag{88}$$

meaning that the first step is a pure gradient step.

9.3. Restarting the Algorithm. For a quadratic function subject to a linear constraint, the present algorithm converges to the exact minimum in no more than $n - q$ steps. For the general case, this suggests the idea of restarting the algorithm every $\Delta N = n - q$ or $\Delta N = n - q + 1$ steps.

10. Order-of-Magnitude Analysis

The sequential conjugate gradient-restoration algorithm is represented by Eqs. (62) and (49). While Eqs. (49) are exact, Eqs. (82) are an approximation to the true optimal conditions. In this section, an estimate of the error involved in the computation of the Lagrange multiplier λ is given; furthermore, a verification of the descent properties of the algorithm is presented.

10.1. Lagrange Multiplier. Here, we assume that, if ϵ is a small quantity,

$$\alpha = O(\epsilon) \tag{89}$$

for every gradient phase. Concerning the multiplier λ, we note that the exact equation (62) has been replaced by the approximate equation (82-1); therefore, the q-vector

$$R_1 = \gamma\varphi_x^T(x)\,\hat{p} \tag{90}$$

is representative of the order of magnitude of the error in λ. This is due to the fact that the terms $\varphi_x^T(x)\,f_x(x)$ and $\varphi_x^T(x)\,\varphi_x(x)$ in Eq. (62) can be regarded to be of $O(1)$.

If λ is computed with the approximate equation (82-1), the first-order change of the constraint (2) is not exactly zero and, because of Eqs. (82), is given by

$$\delta\varphi(x) = -\alpha\gamma\varphi_x^T(x)\,\hat{p} \tag{91}$$

If one observes that $\varphi(x) = 0$ and that, to first order,

$$\delta\varphi(x) = \varphi(y) - \varphi(x) = \varphi(y) \tag{92}$$

it follows that

$$\varphi(y) = -\alpha\gamma\varphi_x^T(x)\,\hat{p} \tag{93}$$

If Eqs. (49) and (93) are combined, the displacement Δy associated with the restoration phase can be written as

$$\Delta y = \alpha\gamma k\varphi_x(y)[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\,\varphi_x^T(x)\,\hat{p} \tag{94}$$

To first order, the expansion of the gradient $\varphi_x(\tilde{x})$ is given by

$$\varphi_x(\tilde{x}) = \varphi_x(x) + \varphi_{xx}(x)(\Delta x + \Delta y) \tag{95}$$

where $\varphi_{xx}(x)$ denotes the array of the second partial derivatives of φ. If the transposes of both sides of (95) are postmultiplied by p and if Eqs. (82-5) and (94) are accounted for, one deduces that

$$\varphi_x^T(\tilde{x})\,p = \varphi_x^T(x)\,p - \alpha p^T\varphi_{xx}(x)\,p + \alpha\gamma k p^T\varphi_{xx}(x)\,\varphi_x(y)[\varphi_x^T(y)\,\varphi_x(y)]^{-1}\,\varphi_x^T(x)\,\hat{p} \tag{96}$$

Because of (82-1) and (82-4), the first term on the right-hand side of (96) can be written as

$$\varphi_x^T(x)\,p = \gamma\varphi_x^T(x)\,\hat{p} \tag{97}$$

As a consequence, the third term on the right-hand side of (96) is negligible with respect to the first and Eq. (96) can be approximated by

$$\varphi_x^T(\hat{x})\, p = \gamma \varphi_x^T(x)\, \hat{p} - \alpha p^T \varphi_{xx}(x)\, p \tag{98}$$

This is the key recurrence formula necessary to estimate the order of magnitude of the terms of the form $\varphi_x^T(x)\hat{p}$ or $\varphi_x^T(\hat{x})p$. Since α is of $O(\epsilon)$, we see that, for the first step of the algorithm ($\gamma = 0$ or $\hat{p} = 0$),

$$\varphi_x^T(\hat{x})\, p = O(\epsilon) \tag{99}$$

Therefore, for any subsequent step, an analogous relation holds. Thus, at any point x, we conclude that

$$\varphi_x^T(x)\, \hat{p} = O(\epsilon) \tag{100}$$

Since R_1 given by (90) is proportional to the left-hand side of (100), we see that

$$R_1 = O(\epsilon) \tag{101}$$

Therefore, the Lagrange multiplier λ computed with (82-1) is precise to $O(\epsilon)$.

10.2. Remark. Because of (89), (91), (93), (100), we conclude that

$$\delta q(x) = O(\epsilon^2), \qquad \psi(y) = O(\epsilon^2) \tag{102}$$

Furthermore, from (82-5), (89), (94), (100), we see that

$$\Delta x = O(\epsilon), \qquad \Delta y = O(\epsilon^2) \tag{103}$$

10.3. Descent Property of the Gradient Phase. The first-order change of the function $F(x, \lambda)$ between points x and y is given by Eq. (15): in the light of (62), this can be written as

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) - R_2 \tag{104}$$

where R_2 is a scalar given by

$$R_2 = \alpha\gamma F_x^T(x, \lambda)\, \hat{p} \tag{105}$$

On account of (9), Eq. (105) can be rewritten as

$$R_2 = R_3 + R_4 \tag{106}$$

where

$$R_3 = \alpha\gamma[f_x(x) + \varphi_x(x)\hat{\lambda}]^T \hat{p}, \qquad R_4 = \alpha\gamma(\lambda - \hat{\lambda})^T \varphi_x^T(x)\, \hat{p} \tag{107}$$

We observe that, to first order,

$$f_x(x) = f_x(\hat{y}) + f_{xx}(\hat{y})\, \Delta\hat{y}, \qquad \varphi_x(x) = \varphi_x(\hat{y}) + \varphi_{xx}(\hat{y})\, \Delta\hat{y} \tag{108}$$

and that

$$\lambda - \hat{\lambda} = \Phi_x^T(\hat{x})(\Delta\hat{x} + \Delta\hat{y}) \tag{109}$$

where $\Phi(x)$ denotes the right-hand side of (82-1). In the light of Eq. (87) applied to the previous iteration, Eqs. (107) become

$$R_3 = \alpha\gamma\, \Delta\hat{y}^T F_{xx}(\hat{y}, \hat{\lambda})\, \hat{p}, \qquad R_4 = \alpha\gamma(\Delta\hat{x} + \Delta\hat{y})^T \Phi_x(\hat{x})\, \varphi_x^T(x)\, \hat{p} \tag{110}$$

If (89), (100), (103) are recalled, we see that

$$R_3 = O(\epsilon^3), \qquad R_4 = O(\epsilon^3) \tag{111}$$

so that

$$R_2 = O(\epsilon^3) \tag{112}$$

Since the first term on the right-hand side of Eq. (104) is of $O(\epsilon)$, R_2 can be neglected and Eq. (104) can be approximated by

$$\delta F(x, \lambda) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{113}$$

This relationships shows that, if $\alpha > 0$, $\delta F(x, \lambda) < 0$. This is the descent property of the fundamental function during the gradient phase.

Because of definition (8), the relationship (17) can be established. Since $\delta F(x, \lambda)$ is of $O(\epsilon)$ and $\delta q(x)$ is of $O(\epsilon^2)$, Eq. (17) can be approximated by

$$\delta f(x) = \delta F(x, \lambda) \tag{114}$$

which states that the functions $f(x)$ and $F(x, \lambda)$ behave identically, to first order. This and (113) establish the descent property of the function $f(x)$ during the gradient phase.

10.4. Descent Property of the Algorithm. Finally, we consider points x and $\tilde{x}$, both satisfying the constraint (2). To first order, the difference of the values of the function $f(x)$ at these points is given by

$$f(\tilde{x}) - f(x) = f_x^T(x)(\Delta x + \Delta y) \tag{115}$$

On account of (103), the second term on the right-hand side of (115) can be neglected with respect to the first. Hence, Eq. (115) becomes

$$f(\tilde{x}) - f(x) = -\alpha F_x^T(x, \lambda) F_x(x, \lambda) \tag{116}$$

Therefore, for ϵ sufficiently small, the restoration algorithm preserves the descent property of the gradient algorithm: the function f decreases between any two successive restoration phases.

10.5. Coefficient γ. In the present algorithm, expression (82-3) has been used for the coefficient γ. The main justification for (82-3) is that it produces quadratic convergence in the terminal stage of the algorithm. The additional justification is that the terms (90) and (105) containing γ are of $O(\epsilon)$ and $O(\epsilon^2)$, respectively. As a consequence, they do not seriously affect the computation of the multiplier λ and the descent property of the algorithm.

11. Summary of the Algorithm

The algorithm presented in Part 2 consists of the alternate succession of gradient phases and restoration phases. A summary of the algorithm is given below.

11.1. Gradient Phase. For this phase, the algorithm is represented by

$$\begin{aligned}
\lambda &= -[\varphi_x^T(x)\,\varphi_x(x)]^{-1}\,\varphi_x^T(x)\,f_x(x)\\
F_x(x,\lambda) &= f_x(x) + \varphi_x(x)\,\lambda\\
\gamma &= F_x^T(x,\lambda)\,F_x(x,\lambda)/F_x^T(\hat{x},\hat{\lambda})\,F_x(\hat{x},\hat{\lambda})\\
p &= F_x(x,\lambda) + \gamma\hat{p}\\
\Delta x &= -\alpha p\\
y &= x + \Delta x
\end{aligned} \tag{117}$$

The sequence of operations is as follows: (a) select a nominal point x such that $\varphi(x) = 0$; (b) at this point, determine the vector $f_x(x)$ and the matrix $\varphi_x(x)$; (c) compute the multiplier λ with (117-1), the vector $F_x(x,\lambda)$ with (117-2), the coefficient γ with (117-3), and the search direction p with (117-4), where $\hat{p}$ is known from the previous iteration; (d) determine the optimal stepsize α by a one-dimensional search in which either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y,\lambda)$ is minimized along the search direction p; the search is terminated when Ineq. (24) is satisfied; (e) compute the displacement Δx with (117-5) and the varied point y with (117-6).

11.2. Restoration Phase. For this phase, the algorithm is represented by

$$\begin{aligned}
\sigma &= k[\varphi_y^T(y)\,\varphi_y(y)]^{-1}\,\varphi(y)\\
\Delta y &= -\varphi_y(y)\,\sigma\\
\tilde{y} &= y + \Delta y
\end{aligned} \tag{118}$$

The sequence of operations is as follows: (a) at point y, compute the vector $\varphi(y)$ and the matrix $\varphi_y(y)$; (b) assuming $k = 1$, determine the multiplier σ with (118-1), the displacement Δy with (118-2), and the varied point $\tilde{y}$ with (118-3); (c) if $P(\tilde{y}) < P(y)$, the scaling factor $k = 1$ is acceptable; if $P(\tilde{y}) > P(y)$, the previous value of k must be replaced by some smaller value in the range (29) until the condition $P(\tilde{y}) < P(y)$ is met; this can be achieved through successive bisections of k; (d) return to step (a) and repeat the restoration algorithm using $\tilde{y}$ as the starting point y for the subsequent iteration; (e) terminate the restoration algorithm when the stopping condition (35) is satisfied; (f) once the restoration algorithm is completed, verify the inequality

$$f(\tilde{x}) < f(x) \tag{119}$$

If Ineq. (119) is satisfied, start the next gradient phase. If Ineq. (119) is violated, return to the previous gradient phase and reduce the stepsize α until, after restoration, Ineq. (119) is satisfied.

11.3. Starting Condition. At the start of the algorithm, no information pertaining to the previous iteration is available; hence, we set $\hat{p} = 0$ or $\gamma = 0$. This means that the first step is a pure gradient step.

11.4. Restarting Condition. The algorithm must be restarted (a) when the optimal stepsize α cannot be employed due to violation of Ineq. (119) and (b) at the end of every $\Delta N = n - q$ or $\Delta N = n - q + 1$ iterations. The restarting is performed by setting $\hat{p} = 0$ or $\gamma = 0$.

11.5. Stopping Condition. The algorithm is terminated when Ineq. (51) is satisfied.

PART 3: NUMERICAL EXAMPLES

In order to illustrate the theory, several numerical examples are developed using a Burroughs B-5500 computer and double-precision arithmetic. Concerning the *gradient phase*, the one-dimensional search is performed so as to minimize either $\Psi(\alpha) = f(y)$ or $\Psi(\alpha) = F(y, \lambda)$ with respect to α along the search direction. Quasilinearization is used; the following stopping condition is employed:

$$\Psi_\alpha'(\alpha)/\Psi_\alpha'(0) \leqslant 10^{-4} \tag{120}$$

corresponding to $\epsilon_1 = 10^{-4}$ in (25). Concerning the *restoration phase*, the restoration algorithm is employed iteratively until the error in the constraint satisfies the inequality

$$P(\tilde{x}) \leqslant 10^{-20} \tag{121}$$

corresponding to $\theta_2 = 10^{-20}$ in (35). Finally, the *sequential gradient-restoration algorithm* is terminated when the error in the optimal conditions satisfies the inequality

$$Q(x, \lambda) \leqslant 10^{-15} \tag{122}$$

corresponding to $\theta_4 = 10^{-15}$ in (51).

In the following sections, two groups of numerical examples are presented: (a) examples pertaining to quadratic function and linear constraint and (b) examples pertaining to nonquadratic function and/or nonlinear constraint. Both the ordinary gradient and the conjugate gradient versions of the algorithm are used. The following terminology is adopted: N is the iteration number (each iteration includes a gradient phase and a restoration phase), N_R is the number of restoration cycles per iteration, N_* is the number of iterations for convergence, and ΔN is the number of iterations between successive restarting points. For simplicity, the symbols employed throughout Part 3 are scalar.

12. Examples: Quadratic Function, Linear Constraint

The first group of examples deals with functions of the form (69-1) subject to a constraint of the form (69-2). For these functions, the following properties hold: (a) no restoration is needed, that is, $N_R = 0$; (b) the search performed by quasilinearization yields the optimal value of α in one step; (c) the value of α computed by minimizing $\Psi = f$ is the same as that computed by minimizing $\Psi = F$; and (d) convergence to the minimum occurs in no more than $n - q$ iterations in the conjugate gradient version of the algorithm.

Example 12.1. We consider the problem of minimizing the function

$$f = (x + y)^2 + (y + z)^2 \tag{123}$$

subject to the constraint

$$x + 2y + 3z - 1 = 0 \tag{124}$$

Note that $n = 3$, $q = 1$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = \tfrac{1}{2}, \quad y = -\tfrac{1}{2}, \quad z = \tfrac{1}{2} \tag{125}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = -4, \quad y = 1, \quad z = 1 \tag{126}$$

consistent with (124). Convergence is achieved in $N_* = 19$ iterations with the ordinary gradient version of the algorithm and $N_* = 2$ iterations with the conjugate gradient version. For the latter, Table 1 shows the detailed results.

Table 1 ($n - q = 2$)

N	N_R	x	y	z	f	Q
0	—	−4.0000	1.0000	1.0000	0.13×10^{2}	0.33×10^{3}
1	0	−1.6439	1.8759	−0.3686	0.23×10^{1}	0.50×10^{1}
2	0	0.5000	−0.5000	0.5000	0.93×10^{-44}	0.50×10^{-14}

Example 12.2. We consider the problem of minimizing the function

$$f = (x - 1)^2 + (y - z)^2 + (u - w)^2 \tag{127}$$

subject to the constraints

$$x + y + z + u + w - 5 = 0, \qquad x - 2(u + w) + 3 = 0 \tag{128}$$

Note that $n = 5$, $q = 2$, so that $n - q = 3$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{129}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 3, \quad y = 5, \quad z = -3, \quad u = 2, \quad w = -2 \tag{130}$$

consistent with (128). Convergence is achieved in $N_* = 17$ iterations with the ordinary gradient version of the algorithm and $N_* = 3$ iterations with the conjugate gradient version. For the latter, Table 2 shows the detailed results.

Table 2 ($n - q = 3$)

N	N_s	x	y	z	u	w	f	Q
0	—	3.0000	5.0000	−5.0000	2.0000	−2.0000	0.84×10^{2}	0.62×10^{1}
1	0	1.4422	0.6651	0.6363	0.7209	1.0373	0.78×10^{0}	0.26×10^{-4}
2	0	1.1304	0.8169	0.9417	1.1059	0.8549	0.10×10^{0}	0.63×10^{-4}
3	0	1.0000	1.0000	1.0000	1.0000	1.0000	0.12×10^{-10}	0.69×10^{-10}

Example 12.3. We consider the problem of minimizing the function

$$f = (x - y)^2 + (y + z - 2)^2 + (u - 1)^2 + (w - 1)^2 \tag{131}$$

subject to the constraints

$$x + 3y - 4 = 0, \quad z + u - 2w = 0, \quad y - w = 0 \tag{132}$$

Note that $n = 5$, $q = 3$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{133}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = \tfrac{5}{2}, \quad y = \tfrac{1}{2}, \quad z = 2, \quad u = -1, \quad w = \tfrac{1}{2} \tag{134}$$

consistent with (132). Convergence is achieved in $N_* = 13$ iterations with the ordinary gradient version of the algorithm and $N_* = 2$ iterations with the conjugate gradient version. For the latter, Table 3 shows the detailed results.

Table 3 ($n - q = 2$)

N	N_s	x	y	z	u	w	f	Q
0	—	2.5000	0.5000	2.0000	−1.0000	0.5000	0.85×10^{1}	0.40×10^{1}
1	0	0.8086	1.0638	1.6032	0.5244	1.0638	0.73×10^{0}	0.37×10^{1}
2	0	1.0000	1.0000	1.0000	1.0000	1.0000	0.70×10^{-11}	0.30×10^{-12}

13. Examples: Nonquadratic Function and/or Nonlinear Constraint

The second group of examples deals with general functions and general constraints. Concerning the search, the value of α computed by minimizing $\Psi = f$ is different from that computed by minimizing $\Psi = F$. For the conjugate gradient version, the algorithm is restarted every ΔN iterations. Note that $\Delta N = 1$ corresponds to the ordinary gradient version.

Example 13.1. We consider the problem of minimizing the function

$$f = (x - y)^2 + (y - z)^4 \tag{135}$$

subject to the constraint

$$x(1 + y^2) + z^4 - 3 = 0 \tag{136}$$

Note that $n = 3$, $q = 1$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1 \tag{137}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = -13/5, \quad y = 2, \quad z = 2 \tag{138}$$

consistent with (136). Table 4 shows N_* for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 2$ and $\Psi = F$ are presented in Table 5.

Table 4 ($n - q = 2$)

ΔN	N_* ($\Psi = f$)	N_* ($\Psi = F$)
1	3821	4360
2	8	13
3	12	13
4	7	10

Table 5 ($n - q = 2$, $\Delta N = 2$, $\Psi = F$)

N	N_s	x	y	z	f	Q
0	—	−2.0000	2.0000	2.0000	0.21×10^{1}	0.15×10^{2}
1	5	−0.3763	0.0112	1.3556	0.33×10^{1}	0.82×10^{1}
2	5	1.3359	0.9312	0.4357	0.43×10^{0}	0.21×10^{2}
3	3	1.2026	1.1752	0.6076	0.10×10^{0}	0.86×10^{0}
4	6	0.7711	0.9447	1.1140	0.30×10^{-1}	0.22×10^{0}
5	2	0.8515	0.8659	1.1078	0.35×10^{-1}	0.34×10^{-1}
6	4	1.0764	1.0316	0.9214	0.72×10^{-2}	0.25×10^{-1}
7	1	1.0658	1.0641	0.9234	0.39×10^{-2}	0.19×10^{-2}
8	3	1.0211	1.0225	0.9769	0.62×10^{-4}	0.17×10^{-4}
9	1	1.0218	1.0217	0.9770	0.39×10^{-4}	0.17×10^{-4}
10	2	1.0055	1.0036	0.9943	0.26×10^{-5}	0.85×10^{-5}
11	1	1.0055	1.0035	0.9943	0.15×10^{-5}	0.43×10^{-10}
12	2	1.0008	1.0008	0.9991	0.12×10^{-10}	0.24×10^{-10}
13	0	1.0008	1.0008	0.9991	0.97×10^{-10}	0.65×10^{-14}

Example 13.2. We consider the problem of minimizing the function

$$f = (x - y)^2 + (z - 1)^2 + (u - 1)^4 + (w - 1)^6 \tag{139}$$

subject to the constraints

$$xz^2 + \sin(u - w) - 1 = 0, \qquad y + z^4u^2 - 2 = 0 \tag{140}$$

Note that $n = 5$, $q = 2$, so that $n - q = 3$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{141}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 1/\sqrt{2}, \quad y = 7/4, \quad z = \tfrac{1}{2}, \quad u = 2, \quad w = 2 \tag{142}$$

Table 6 ($n - q = 3$)

ΔN	N_* ($\Psi = f$)	N_* ($\Psi = F$)
1	30	112
2	10	10
3	8	11
4	10	12
5	17	12

Table 7 ($n - q = 3$, $\Delta N = 3$, $\Psi = F$)

N	N_s	x	y	z	u	w	f	Q
0	—	0.7071	1.7500	0.5000	2.0000	2.0000	0.33×10^{1}	0.58×10^{2}
1	5	0.9072	1.3220	1.0192	0.7926	0.4377	0.20×10^{0}	0.11×10^{1}
2	3	1.0700	1.0988	1.1376	0.7734	0.5725	0.30×10^{-1}	0.38×10^{-1}
3	3	1.0302	1.0681	1.0338	0.9012	0.8619	0.26×10^{-2}	0.44×10^{-2}
4	2	1.0187	1.0485	1.0105	0.9608	0.8728	0.18×10^{-2}	0.38×10^{-2}
5	1	1.0073	1.0074	1.0047	0.9868	0.9862	0.22×10^{-3}	0.13×10^{-4}
6	2	1.0063	1.0082	1.0038	0.9562	0.9569	0.10×10^{-3}	0.67×10^{-4}
7	1	1.0066	1.0069	1.0040	0.9885	0.9902	0.16×10^{-3}	0.13×10^{-4}
8	2	1.0030	1.0030	1.0000	0.9983	1.0028	0.31×10^{-5}	0.19×10^{-5}
9	1	1.0030	1.0030	1.0000	0.9983	1.0028	0.26×10^{-5}	0.78×10^{-6}
10	1	1.0030	1.0036	1.0000	0.9983	1.0028	0.22×10^{-6}	0.13×10^{-6}
11	1	1.0029	1.0029	1.0000	0.9985	1.0029	0.50×10^{-11}	0.61×10^{-10}

consistent with (140). Table 6 shows N_* for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 3$ and $\Psi = F$ are presented in Table 7.

Example 13.3. We consider the problem of minimizing the function

$$f = (x - y)^2 + (y - z)^2 + (z - u)^4 + (u - w)^4 \tag{143}$$

subject to the constraints

$$x + y^2 + z^3 - 3 = 0, \qquad y - z^2 + u - 1 = 0, \qquad xw - 1 = 0 \tag{144}$$

Note that $n = 5$, $q = 3$, so that $n - q = 2$. This function admits the relative minimum $f = 0$ at the point defined by

$$x = 1, \quad y = 1, \quad z = 1, \quad u = 1, \quad w = 1 \tag{145}$$

The nominal point chosen for starting the algorithm is the point of coordinates

$$x = 2, \quad y = \sqrt{2}, \quad z = -1, \quad u = 2 - \sqrt{2}, \quad w = \tfrac{1}{2} \tag{146}$$

consistent with (144). Table 8 shows N_* for several values of ΔN and both $\Psi = f$ and $\Psi = F$. The detailed results pertaining to $\Delta N = 2$ and $\Psi = F$ are presented in Table 9.

Table 8 ($n - q = 2$)

ΔN	N_* ($\Psi = f$)	N_* ($\Psi = F$)
1	29	41
2	10	11
3	11	14
4	10	12

Table 9 ($n - q = 2, \Delta N = 2, \Psi = F$)

N	N_ρ	x	y	z	u	v	f	Q
0	—	2.0000	1.4142	−1.0000	0.5557	0.5000	0.11×10^{2}	0.35×10^{2}
1	4	1.6940	1.0517	−0.0536	−0.0488	0.5279	0.20×10^{1}	0.40×10^{2}
2	4	1.6611	1.1390	0.3402	−0.0233	0.6012	0.10×10^{2}	0.37×10^{2}
3	6	1.0921	0.7644	1.0979	1.4409	0.9156	0.30×10^{0}	0.27×10^{1}
4	4	0.9059	0.8443	1.1111	1.3877	1.1037	0.63×10^{-1}	0.30×10^{0}
5	3	0.8561	0.9538	1.0719	1.1542	1.1677	0.21×10^{-1}	0.22×10^{-1}
6	3	1.0033	1.0147	0.9758	0.9376	0.9606	0.71×10^{-4}	0.26×10^{-3}
7	2	1.0017	1.0099	0.9927	0.9755	0.9982	0.36×10^{-5}	0.10×10^{-5}
8	2	1.0017	0.9979	0.9994	0.9965	0.9982	0.32×10^{-5}	0.89×10^{-5}
9	1	1.0003	1.0004	0.9995	0.9986	0.9994	0.10×10^{-6}	0.98×10^{-8}
10	1	0.9999	1.0000	0.9999	0.9999	1.0000	0.70×10^{-11}	0.29×10^{-14}
11	0	0.9999	0.9999	1.0000	1.0000	1.0000	0.20×10^{-14}	0.21×10^{-16}

14. Discussion and Conclusions

In the previous sections, a sequential algorithm is developed for minimizing a function $f(x)$ subject to the constraint $\varphi(x) = 0$. The algorithm is composed of the alternate succession of gradient phases and restoration phases. For the gradient phase, two versions are presented, one in which information at only point x is used (ordinary gradient version) and one in which information at both points x and $\hat{x}$ is used (conjugate gradient version). For the restoration phase, the criterion employed is that of the least-square change of the coordinates.

While the ordinary gradient version of the algorithm exhibits only asymptotic convergence, the conjugate gradient version exhibits quadratic convergence; this means that, for a quadratic function subject to a linear constraint, the minimum point is obtained in no more than $n - q$ iterations.

In order to illustrate the theory, several numerical examples are presented. The first three examples are concerned with a quadratic function subject to a linear constraint: the computer results confirm the quadratic convergence properties of the conjugate gradient version of the algorithm. The next three examples are concerned with general functions subject to general constraints; the computer results show the rapidly convergent properties of the conjugate gradient version of the algorithm and its superiority with respect to the ordinary gradient version.

In the theory as well as in the examples, the function $\Psi(\alpha) = f(y)$ or the function $\Psi(\alpha) = F(y, \lambda)$ is minimized along the search direction. However, the occurrence of cases where $\Psi(\alpha)$ does not possess a relative minimum is conceivable. For these cases, an upper limit must be imposed on the stepsize α or the performance index $P(y)$.

In the numerical examples, the starting point was chosen so that $\varphi(x) = 0$. However, the present algorithms can be started even if $\varphi(x) \neq 0$. In this case, the first phase is a restoration phase rather than a gradient phase.

References

1. Miele, A., and Heideman, J. C., *Mathematical Programming for Constrained Minimal Problems, Part 1, Sequential Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No. 59, 1969.
2. Miele, A., Huang, H. Y., and Heideman, J. C., *Mathematical Programming for Constrained Minimal Problems, Part 2, Sequential Conjugate Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No. 61, 1969.
3. Miele, A., Huang, H. Y., and Cantrell, J. W., *Gradient Methods in Mathematical Programming, Part 1, Review of Previous Techniques*, Rice University, Aero-Astronautics Report No. 55, 1969.
4. Miele, A., Heideman, J. C., and Damoulakis, J. N., *The Restoration of Constraints in Holonomic and Nonholonomic Problems*, Journal of Optimization Theory and Applications, Vol. 3, No. 5, 1969.
5. Hestenes, M. R., and Stiefel, E., *Methods of Conjugate Gradients for Solving Linear Systems*, Journal of Research of the National Bureau of Standards, Vol. 49, No. 6, 1952.
6. Beckman, F. S., *The Solution of Linear Equations by the Conjugate Gradient Method*, Mathematical Methods for Digital Computers, Edited by A. Ralston and H. S. Wilf, John Wiley and Sons, New York, 1960.
7. Fletcher, R., and Reeves, C. M., *Function Minimization by Conjugate Gradients*, Computer Journal, Vol. 7, No. 2, 1964.

Additional Bibliography

Rosen, J. B., *The Gradient Projection Method for Nonlinear Programming, Part 2, Nonlinear Constraints*, SIAM Journal on Applied Mathematics, Vol. 9, No. 4, 1961.
Bryson, A. E., Jr., and Ho, Y. C., *Applied Optimal Control*, Chapter 1, Blaisdell Publishing Company, New York, 1969.

PRINTED IN BELGIUM BY THE ST. CATHERINE PRESS, LTD.

On the Method of Multipliers for Mathematical Programming Problems[1]

A. Miele,[2] P. E. Moseley,[3] A. V. Levy,[4] and G. M. Coggins[5]

Abstract. In this paper, the numerical solution of the basic problem of mathematical programming is considered. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q \le n$.

The approach employed is based on the introduction of the augmented penalty function $W(x, \lambda, k) = f(x) + \lambda^T\varphi(x) + k\varphi^T(x)\varphi(x)$. Here, the q-vector λ is an approximation to the Lagrange multiplier, and the scalar $k > 0$ is the penalty constant.

Previously, the augmented penalty function $W(x, \lambda, k)$ was used by Hestenes in his method of multipliers. In Hestenes' version, the method of multipliers involves cycles, in each of which the multiplier and the penalty constant are held constant. After the minimum of the augmented penalty function is achieved in any given cycle, the multiplier λ is updated, while the penalty constant k is held unchanged.

In this paper, two modifications of the method of multipliers are presented in order to improve its convergence characteristics. The improved convergence is achieved by (i) increasing the updating frequency so that the number of iterations in a cycle is shortened to $\Delta N = 1$ for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and $\Delta N = n$ for the conjugate-gradient algorithm, (ii) imbedding Hestenes' updating rule for the multiplier λ

[1] Paper received February 18, 1971; in revised form, April 28, 1971. This research was supported by the National Science Foundation, Grant No. GP-32453. The authors are indebted to Messieurs E. E. Cragg and A. Esterle for computational assistance.

[2] Professor of Astronautics and Mathematical Sciences, Rice University, Houston, Texas.

[3] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Senior Research Engineer, Esso Production Research Company, Houston, Texas.

[4] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Research Associate, University of Mexico, Mexico City, Mexico.

[5] Graduate Student in Aero-Astronautics, Rice University, Houston, Texas. Presently, Instructor, US Military Academy, West Point, New York.

into a one-parameter family and determining the scalar parameter β so that the error in the optimum condition is minimized, and (iii) updating the penalty constant k so as to cause some desirable effect in the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. For the sake of identification, Hestenes' method of multipliers is called Method MM-1, the modification including (i) and (ii) is called Method MM-2, and the modification including (i), (ii), (iii) is called Method MM-3.

Evaluation of the theory is accomplished with seven numerical examples. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. Each example is solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which are employed in conjunction with Methods MM-1, MM-2, and MM-3.

The numerical results show that (a) for given penalty constant k, Method MM-2 generally exhibits faster convergence than Method MM-1, (b) in both Methods MM-1 and MM-2, the number of iterations for convergence has a minimum with respect to k, and (c) the number of iterations for convergence of Method MM-3 is close to the minimum with respect to k of the number of iterations for convergence of Method MM-2. In this light, Method MM-3 has very desirable characteristics.

1. Introduction

Over the past several years, considerable work has been done on the numerical solution of the constrained minimization problem. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$.

The methods employed are generally based on one of two basic ideas. One approach ensures constraint satisfaction, at least to first order, at the end of each iteration (see, for example, Refs. 1–3). The other approach depends on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. With regard to the latter approach, the standard penalty function method (see, for example, Refs. 4–6) and Hestenes' method of multipliers (Ref. 7) must be mentioned.

As the numerical experiments of Refs. 8–9 indicate, the method of multipliers generally exhibits faster convergence than the standard penalty function method. Crucial to the method of multipliers is the manner in which the multiplier λ is estimated and the penalty constant k is updated at the beginning of each cycle. These key questions are considered in this paper, whose objective is to enlarge the investigation of Ref. 10 and to develop techniques for improving the convergence characteristics of the method of multipliers. The resulting algorithm is called *modified method of multipliers.*

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) \tag{1}$$

subject to the constraint

$$\varphi(x) = 0. \tag{2}$$

In the above equations, f is a scalar, x is an n-vector, and φ is a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions f and φ exist and are continuous and that the constrained minimum exists.

2.1. First-Order Conditions. The previous problem can be recast as that of minimizing the augmented function

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x) , \tag{3}$$

subject to the constraint (2). The q-vector λ is the Lagrange multiplier, and the superscript T denotes the transpose of a matrix.

From theory of maxima and minima, it is known that the optimal solution must satisfy the relations

$$\varphi(x) = 0, \qquad F_x(x, \lambda) = f_x(x) + \varphi_x(x) \lambda = 0, \tag{4}$$

which are a system of $n + q$ equations in x and λ. The subscript x denotes the gradient of a function: in this case, f_x and F_x are n-vectors, and φ_x is an $n \times q$ matrix, defined in such a way that its ith column is the gradient of the ith scalar component of φ with respect to x.

2.2. Approximate Solutions. In general, the system (4) is nonlinear; consequently, approximate methods must be employed. Here, we introduce the scalar performance indexes

$$P(x) = \varphi^T(x)\varphi(x), \qquad Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda), \tag{5}$$

which measure the errors in the constraint and the optimum condition, respectively. At the solution point, $P = 0$ and $Q = 0$, while $P > 0$ and/or $Q > 0$ for any approximation of the solution. When approximate methods are employed, they must ultimately lead to values of x, λ such that

$$P(x) \leq \epsilon_1, \qquad Q(x,\lambda) \leq \epsilon_2. \tag{6}$$

Alternatively, (6) can be replaced by

$$R(x,\lambda) \leq \epsilon_3, \tag{7}$$

where

$$R(x,\lambda) = P(x) + Q(x,\lambda) \tag{8}$$

denotes the cumulative error in the constraint and the optimum condition. Here, $\epsilon_1, \epsilon_2, \epsilon_3$ are small, preselected numbers. Note that satisfaction of Ineq. (7) implies satisfaction of Ineqs. (6) if one chooses $\epsilon_1 = \epsilon_2 = \epsilon_3$.

3. Review of Penalty Function Methods

The penalty function method is based on the construction of a sequence of special functions having, in the limit, an unconstrained minimum point coincident with the solution of the original constrained minimization problem. In this section, two versions of the penalty function method are reviewed: (i) the standard penalty function method and (ii) the method of multipliers. Method (i) is based on the standard penalty function (9) and Method (ii) is based on the augmented penalty function (22).

3.1. Standard Penalty Function Method. This method is based on the consideration of the standard penalty function

$$U(x,k) = f(x) + k\varphi^T(x)\varphi(x). \tag{9}$$

This is obtained by adding to the function $f(x)$ a term quadratic in the constraint $\varphi(x)$, $k > 0$ being the penalty constant.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (9) with respect to x for given k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

$$U_x(x,k) = f_x(x) + 2k\varphi_x(x)\varphi(x) = 0. \tag{10}$$

If the penalty constant k is arbitrary, the vector x which satisfies Eq. (10) is such that $\varphi(x) \neq 0$. However, if one defines the Lagrange multiplier to be

$$\lambda = 2k\varphi(x), \tag{11}$$

Eq. (10) reduces to

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda = 0, \tag{12}$$

meaning that the combination of x and λ thus obtained satisfies exactly the optimum condition.

In order to obtain constraint satisfaction, increasingly larger values of the penalty constant must be employed in successive cycles of the standard penalty function method. In this connection, let k_1 denote the penalty constant of the present cycle and k_2 denote the penalty constant of the next cycle, with $k_2 > k_1$. Because of the jump in k, the standard penalty function increases by the amount

$$U(x,k_2) - U(x,k_1) = (k_2 - k_1)P(x), \tag{13}$$

and the norm of the gradient of the standard penalty function takes the value[4]

$$U_x^T(x,k_2)U_x(x,k_2) = (k_2 - k_1)^2 P_x^T(x)P_x(x), \tag{14}$$

where

$$P(x) = \varphi^T(x)\varphi(x), \qquad P_x(x) = 2\varphi_x(x)\varphi(x). \tag{15}$$

The positiveness of the right-hand side of Eq. (13) is the key to the mechanism on which the standard penalty function method is based.

After a sufficient number of cycles, the constraint error can be made as small as desired providing the penalty constant has become sufficiently large. Theoretically speaking, the condition $\varphi(x) = 0$ is desired at convergence; consequently, the multiplier λ defined by Eq. (11) can be identical with the multiplier satisfying Eqs. (4), which is

[4] Note that $U_x(x,k_1) = 0$.

generally nonzero, only if $k \to \infty$. In a practical digital computer, this means that very large values of k are needed at convergence.

Numerical Implementation. From the above considerations, the following outline of the standard penalty function method emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the standard penalty function

$$U(x, k) = f(x) + k\varphi^T(x)\,\varphi(x) \tag{16}$$

is minimized with respect to x for given k. The minimum of $U(x, k)$ is achieved when the following stopping condition is satisfied:

$$U_x^T(x, k)\, U_x(x, k) \leq \epsilon_1 , \tag{17}$$

where ϵ_1 is a small, preselected number.

(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the standard penalty function method.

(d) For the next cycle, a higher value of the penalty constant is selected, one choice being

$$k_2 = \alpha k_1 , \tag{18}$$

where $\alpha > 1$ is the penalty constant ratio.

(e) After updating the penalty constant, one returns to (b) and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

$$\varphi^T(x)\,\varphi(x) + U_x^T(x, k)\, U_x(x, k) \leq \epsilon_2 , \tag{19}$$

where ϵ_2 is a small, preselected number.

Remark. At convergence of a cycle, the stopping condition (17) can be written as

$$Q(x, \lambda) \leq \epsilon_1 , \tag{20}$$

where λ is given by Eq. (11). Analogously, at convergence of the algorithm, the stopping condition (19) becomes

$$R(x, \lambda) = P(x) + Q(x, \lambda) \leq \epsilon_2 , \tag{21}$$

where λ is given by Eq. (11).

3.2. Method of Multipliers. This method is based on the consideration of the augmented penalty function

$$W(x, \lambda, k) = f(x) + \lambda^T\varphi(x) + k\varphi^T(x)\,\varphi(x). \tag{22}$$

This is obtained by adding to the penalty function $U(x, k)$ a term linear in the constraint $\varphi(x)$, the q-vector λ being an approximation to the Lagrange multiplier. The use of this function was suggested by Hestenes (Ref. 7) in order to circumvent the numerical difficulties associated with the extremely large values of the penalty constant required by the standard penalty function method.

The problem of minimizing the function (1) subject to the constraint (2) is replaced by a sequence of unconstrained minimization problems. In each element of the sequence or cycle, one minimizes the function (22) with respect to x for given λ and k. Therefore, theoretically speaking, the following necessary condition must be satisfied at the end of each cycle:

$$W_x(x, \lambda, k) = f_x(x) + \varphi_x(x)\,\lambda + 2k\varphi_x(x)\,\varphi(x) = 0 \tag{23}$$

and is equivalent to

$$W_x(x, \lambda, k) = f_x(x) + \varphi_x(x)\,[\lambda + 2k\varphi(x)] = 0. \tag{24}$$

If the penalty constant k and the multiplier λ are arbitrary, the vector x which satisfies Eq. (24) is such that $\varphi(x) \neq 0$. However, by means of a proper updating rule, a new Lagrange multiplier can be found such that the optimum condition is satisfied exactly. In this connection, let λ_1 denote the Lagrange multiplier of the present cycle and λ_2 denote the Lagrange multiplier of the next cycle. If λ_2 is chosen to be

$$\lambda_2 = \lambda_1 + 2k\varphi(x), \tag{25}$$

Eq. (24) reduces to

$$F_x(x, \lambda_2) = f_x(x) + \varphi_x(x)\,\lambda_2 = 0, \tag{26}$$

meaning that the combination of x and λ_2 thus obtained satisfies exactly the optimum condition.

At the end of any given cycle, whenever the value of the Lagrange multiplier is changed from λ_1 to λ_2, the augmented penalty function increases by the amount

$$W(x, \lambda_2, k) - W(x, \lambda_1, k) = 2kP(x), \tag{27}$$

and the norm of the gradient of the augmented penalty function takes the value[7]

$$W_x^T(x, \lambda_2, k)\, W_x(x, \lambda_2, k) = k^2 P_x^T(x)\, P_x(x), \tag{28}$$

where $P(x)$ and $P_x(x)$ are given by Eqs. (15). The positiveness of the right-hand side of Eq. (27) is the key to the mechanism on which the method of multipliers is based.

The attention of the reader is called on the essential similarity between Eqs. (13) and (27). In the standard penalty function method, the drive toward constraint satisfaction is supplied by increasing the penalty constant from cycle to cycle. In the method of multipliers, the drive toward constraint satisfaction is supplied by changing the multiplier from cycle to cycle in accordance with Eq. (25).

While the standard penalty function method requires extremely large values of the penalty constant at convergence, this is not the case with the method of multipliers. In the latter method, convergence can be achieved even with moderate values of the penalty constant.

Numerical Implementation. From the above considerations, the following outline of the method of multipliers (Method MM-1) emerges.

(a) The original constrained minimization problem is replaced by a sequence of unconstrained minimization problems.

(b) In each element of the sequence or cycle, the augmented penalty function

$$W(x, \lambda, k) = f(x) + \lambda^T \varphi(x) + k\varphi^T(x)\, \varphi(x) \tag{29}$$

is minimized with respect to x for given λ and k. The minimum of $W(x, \lambda, k)$ is achieved when the following stopping condition is satisfied:

$$W_x^T(x, \lambda, k)\, W_x(x, \lambda, k) \leqslant \epsilon_1, \tag{30}$$

where ϵ_1 is a small, preselected number.

(c) The solution point of any given cycle is chosen as the starting point of the next cycle of the method of multipliers.

(d) For the next cycle, the multiplier is updated according to the simple rule

$$\lambda_2 = \lambda_1 + 2k\varphi(x). \tag{31}$$

[7] Note that $W_x(x, \lambda_1, k) = 0$.

(e) After updating the multiplier, one returns to (b) and continues iteratively.

(f) The algorithm is terminated when the following stopping condition is satisfied:

$$\varphi^T(x)\, \varphi(x) + W_x^T(x, \lambda, k)\, W_x(x, \lambda, k) \leqslant \epsilon_2, \tag{32}$$

where ϵ_2 is a small, preselected number.

(g) To start the algorithm some assumption concerning the multiplier is necessary. The simplest assumption is

$$\lambda = 0 \tag{33}$$

and is equivalent to stating that the augmented penalty function (29) and the standard penalty function (16) are identical for the first cycle of the algorithm.

Remark. At convergence of a cycle, the stopping condition (30) can be written as

$$Q(x, \lambda_2) \leqslant \epsilon_1, \tag{34}$$

where λ_2 is given by Eq. (31). Analogously, at convergence of the algorithm, the stopping condition (32) becomes

$$R(x, \lambda_2) = P(x) + Q(x, \lambda_2) \leqslant \epsilon_2, \tag{35}$$

where λ_2 is given by Eq. (31).

4. Modifications of the Method of Multipliers

The method of multipliers described in Section 3 has one drawback: a sequence of unconstrained minimization problems must be solved, each possibly requiring a large number of iterations ΔN. Consequently, the total number of iterations for convergence $N_* = \Sigma(\Delta N)$ may become excessive for practical applications.

In order to accelerate convergence, we explore here several modifications of the method of multipliers. These modifications are obtained by (i) shortening the length of a cycle, (ii) improving the estimate of the multiplier, and (iii) selecting the penalty constant in an appropriate fashion.

4.1. Updating Frequency. Let a cycle be defined as any sequence of iterations in which the multiplier λ and the penalty constant

k are held unchanged, while the vector x is viewed as unconstrained. Let ΔN denote the number of iterations in a cycle, regardless of whether complete convergence or incomplete convergence is achieved.

To shorten the cycle, we assign *a priori* the value of ΔN. Precisely, we choose ΔN as the *smallest* number of iterations compatible with the characteristics of the particular algorithm being considered. Therefore,

$$\Delta N = 1 \tag{36}$$

for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and

$$\Delta N = n \tag{37}$$

for the conjugate-gradient algorithm. Therefore, the stopping condition (30) for a cycle is bypassed and is replaced by (36) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by (37) for the conjugate-gradient algorithm.

4.2. Multiplier Estimate. Now, we assume that the cycle length ΔN is defined by Eq. (36) for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by Eq. (37) for the conjugate-gradient algorithm. Then, we inquire about ways in which the multiplier λ can be estimated.

First Estimate. We assume that Hestenes' updating rule (31) is employed at the end of any cycle of ΔN iterations. We note that the updated error in the optimum condition can be larger or smaller than the error prior to updating. Since an increase in the error $Q(x, \lambda)$ is not desirable, the multiplier λ might be chosen as follows:

$$\begin{aligned} \lambda_2 &= \lambda_1 \qquad \text{if} \qquad Q(x, \lambda_u) \geq Q(x, \lambda_1), \\ \lambda_2 &= \lambda_u \qquad \text{if} \qquad Q(x, \lambda_u) < Q(x, \lambda_1), \end{aligned} \tag{38}$$

where λ_u is given by

$$\lambda_u = \lambda_1 + 2k\varphi(x). \tag{39}$$

Second Estimate. The previous estimate can be improved if Hestenes' updating rule (31) is renounced and is replaced with the more general updating rule

$$\lambda_2 = \lambda_1 + 2\beta\varphi(x), \tag{40}$$

where β is a scalar parameter. This parameter must be determined so as to produce some optimum effect.

For given values of x and λ_1, a change in β causes a change in the updated multiplier λ_2. Consequently, the updated error in the optimum condition

$$Q(x, \lambda_2) = F_x^T(x, \lambda_2)\, F_x(x, \lambda_2) \tag{41}$$

changes. The optimum value of β is that which gives $Q(x, \lambda_2)$ the smallest value for given x and λ_1. After combining (40)-(41), we obtain the relations

$$\begin{aligned} Q(x, \lambda_1, \beta) &= [F_x(x, \lambda_1) + \beta P_x(x)]^T [F_x(x, \lambda_1) + \beta P_x(x)], \\ Q_\beta(x, \lambda_1, \beta) &= 2P_x^T(x)\,[F_x(x, \lambda_1) + \beta P_x(x)], \\ Q_{\beta\beta}(x, \lambda_1, \beta) &= 2P_x^T(x)\, P_x(x), \end{aligned} \tag{42}$$

the second of which vanishes for

$$\beta = -P_x^T(x)\, F_x(x, \lambda_1)/P_x^T(x)\, P_x(x). \tag{43}$$

This value of β minimizes $Q(x, \lambda_1, \beta)$, since $Q_{\beta\beta}(x, \lambda_1, \beta) > 0$ provided $P_x(x)$ does not vanish.

The method of multipliers with cycle stopping condition (30) replaced by (36) or (37) and with multiplier updating rule (31) replaced by (40) and (43) is called modified method of multipliers or Method MM-2. For this method, the following comments are pertinent.

(i) Equations (40) and (43) imply that

$$P_x^T(x)\, F_x(x, \lambda_2) = 0. \tag{44}$$

Therefore, the optimum value of the parameter β is such that the gradient of the constraint error and the gradient of the updated augmented function are orthogonal.

(ii) The relation between the present updating rule and Hestenes' updating rule can be obtained as follows. Let the gradient of the augmented penalty function prior to updating be rewritten as

$$W_x(x, \lambda_1, k) = F_x(x, \lambda_1) + kP_x(x). \tag{45}$$

Then, combining (43) and (45) yields the relation

$$\beta = k - P_x^T(x)\, W_x(x, \lambda_1, k)/P_x^T(x)\, P_x(x), \tag{46}$$

which shows that

$$\beta \cong k, \tag{47}$$

providing

$$W_x(x, \lambda_1, k) = 0 \qquad \text{or} \qquad P_x^T(x)\, W_x(x, \lambda_1, k) = 0. \tag{48}$$

Clearly, the present updating rule and Hestenes' updating rule are identical if applied at complete convergence, that is, at a point where Ineq. (30) is satisfied.

(iii) Equations (40) and (43) are to be employed at the beginning of each cycle of ΔN iterations, including the first cycle. However, for the first cycle, λ_1 is not defined. This being the case, an assumption is necessary, and the simplest assumption is

$$\lambda_1 = 0. \tag{49}$$

(iv) Whenever (40) and (43) are employed, the algorithm should be started at a point where $\varphi(x) \neq 0$.

Third Estimate. In the previous section, the Lagrange multiplier was estimated within the frame of the one-parameter family (40). An even better estimate can be obtained by removing this limitation and minimizing the performance index $Q(x, \lambda)$ with respect to the q-vector λ for given x. After observing that

$$\begin{aligned} Q(x, \lambda) &= [f_x(x) + \varphi_x(x)\lambda]^T [f_x(x) + \varphi_x(x)\lambda], \\ Q_\lambda(x, \lambda) &= 2\varphi_x^T(x)[f_x(x) + \varphi_x(x)\lambda], \\ Q_{\lambda\lambda}(x, \lambda) &= 2\varphi_x^T(x)\varphi_x(x), \end{aligned} \tag{50}$$

we see that the optimal multiplier is defined by the relation

$$\varphi_x^T(x)\varphi_x(x)\lambda + \varphi_x^T(x) f_x(x) = 0, \tag{51}$$

which is a system of q scalar equations in the q components of the multiplier λ. The solution of (51) minimizes Q, since the matrix (50–3) is positive definite. The following comments are pertinent.

(i) On premultiplication by $\varphi^T(x)$, Eq. (51) leads to

$$P_x^T(x) F_x(x, \lambda) = 0. \tag{52}$$

Therefore, the optimum value of the multiplier λ is such that the gradient of the constraint error and the gradient of the updated augmented function are orthogonal.

(ii) Obviously, Eq. (51) supplies the best estimate of the Lagrange multiplier λ compatible with any given position vector x (Ref. 3). However, its use requires the solution of a system of q linear equations in q unknowns. This being the case, the decision on whether to use (40) and (43) or (51) should be made on the basis of the particular algorithm employed.

(iii) For the ordinary-gradient algorithm and the conjugate-gradient algorithm, the displacement vector Δx is computed without solving a system of linear equations. Hence, the estimation of λ should be made with (40) and (43) rather than (51).

(iv) For the modified-quasilinearization algorithm, the displacement Δx is computed by solving a system of n linear equations in n unknowns. Hence, it is optional to estimate λ with (40) and (43) or (51). In the sequel, the estimate given by (40) and (43) is employed.

4.3. Penalty Constant Estimate.

Now, the question arises as to whether the penalty constant can be selected in such a manner as to improve the convergence characteristics of Method MM-2. In this section, two techniques are presented for updating the penalty constant at the end of a cycle, one suitable for the ordinary-gradient algorithm and one suitable for the conjugate-gradient algorithm and the modified-quasilinearization algorithm. Method MM-2 with Eqs. (40) and (43) completed by a relation updating the penalty constant is called Method MM-3.

Ordinary-Gradient Algorithm. When this algorithm is employed, the multiplier updating rule (40) and (43) induces an interesting characteristic: a descent property in the constraint error $P(x)$. This characteristic is utilized in this section to establish an updating rule for the penalty constant.

While the ordinary-gradient algorithm is described in detail in Section 5, we note here that the displacement Δx leading from the nominal point x to the varied point $\tilde{x}$ produces a change in the constraint error $P(x)$. To first order, this change is given by

$$\delta P(x) = -\alpha k P_x^T(x) P_x(x), \tag{53}$$

where α is the stepsize, and k is the penalty constant. Therefore, $\delta P(x) < 0$, since $\alpha > 0$ and $k > 0$. This result guarantees that

$$P(\tilde{x}) < P(x), \tag{54}$$

providing α is sufficiently small.

Among all the values which can be attributed to the penalty constant, we select k in such a way that, on the average, the constraints are satisfied to first order. Thereby, we determine k from the relation

$$\delta P(x) = -2\alpha P(x). \tag{55}$$

Comparing (53) and (55), we see that the appropriate value of the penalty constant should be

$$k = 2P(x)/P_x^T(x)\, P_x(x). \tag{56}$$

Since both the numerator and the denominator of the right-hand side of Eq. (56) contain equal powers of $\varphi(x)$, the penalty constant k varies slowly along the algorithm and is finite at convergence. For the particular case of a single scalar constraint ($q = 1$), Eq. (56) reduces to

$$k = 1/2\varphi_x^T(x)\, \varphi_x(x). \tag{57}$$

where $\varphi_x(x)$ is now an n-vector.

Conjugate-Gradient Algorithm and Modified-Quasilinearization Algorithm. The penalty constant estimate developed for the ordinary-gradient algorithm is based on the descent property (53) and the descent requirement (55). It produces a slowly varying penalty constant (56), which is finite at convergence.

For the conjugate-gradient algorithm and the modified-quasilinearization algorithm, the penalty constant (56) is not desirable for the reasons indicated below. Consider a quadratic function $f(x)$ and a linear constraint $\varphi(x)$. Regardless of the value of k, the augmented penalty function $W(x, \lambda, k)$ is quadratic in x. Theoretically speaking, the optimality condition (24) is satisfied exactly after n iterations of the conjugate-gradient algorithm and after one iteration of the modified quasilinearization algorithm; hence, the theoretical optimum value of the penalty constant should be $k = \infty$, in that it guarantees simultaneous satisfaction of the constraint equation $\varphi(x) = 0$ at the end of a cycle. In a practical digital computer, this result means that large values of the penalty constant should be employed if fast convergence is desired.

If the function $f(x)$ is nonquadratic and/or the constraint $\varphi(x)$ is nonlinear, the above reasoning is approximately true near the solution of the constrained minimization problem. This leads to the concept of programming k so as to achieve moderate values far away from the solution and large values near the solution, even though these large values are not as large as those needed with the standard penalty function method.

Possible choices of the penalty constant satisfying the above property are the following:

$$k_0 = |\lambda_2^T\varphi(x)|/P(x) \tag{58}$$

or

$$k_0 = \sqrt{\{[\varphi_x(x)\,\lambda_2]^T\,[\varphi_x(x)\,\lambda_2]/P_x^T(x)\,P_x(x)\}}. \tag{59}$$

If Eq. (58) is employed, the order of magnitude of the linear term and the quadratic term appearing in the augmented penalty function is the same. If Eq. (59) is employed, the order of magnitude of the gradient of the linear term and the gradient of the quadratic term appearing in the augmented penalty function is the same.[5]

If the convergence requirements on the constraint error $P(x)$ and the error in the optimum condition $Q(x, \lambda)$ are the same, it might be desirable to reduce $P(x)$ and $Q(x, \lambda)$ at approximately the same rate. With this idea in mind, we propose the following updating rule for the penalty constant:

$$\begin{aligned} k_2 &= \min(k_0, k_1) && \text{if} \quad P(x) \leqslant Q(x, \lambda_2), \\ k_2 &= \max(k_0, \pi k_1) && \text{if} \quad P(x) > Q(x, \lambda_2), \end{aligned} \tag{60}$$

where $\pi \geqslant 1$, k_0 is given by (58) or (59), k_1 is the penalty constant prior to updating, and k_2 is the penalty constant after updating. Obviously, (60-1) prevents an increase of the penalty constant if the constraint error is relatively small. Conversely, (60-2) prevents a decrease of the penalty constant if the constraint error is relatively large.

Equation (60) is to be employed at the beginning of each cycle of ΔN iterations, including the first cycle. However, for the first cycle, k_1 is not defined. This being the case, an assumption is necessary, and the simplest assumption is

$$k_1 = k_0 . \tag{61}$$

4.4. Summary of Methods.

In this section, we summarize the combination of methods arising from the previous discussion, as follows.

(i) Method MM-1 is characterized by updating decision (30), multiplier estimate (31), and penalty constant unchanged throughout a particular algorithm.

(ii) Method MM-2 is characterized by updating decision (36) or

[5] Whenever (58) or (59) is employed, the algorithm should be started at a point where $\varphi(x) \neq 0$.

(37), multiplier estimate (40) and (43), and penalty constant unchanged throughout a particular algorithm.

(iii) Method MM-3 is characterized by updating decision (36) or (37), multiplier estimate (40) and (43), and penalty constant estimate (56) or (58) and (60).

Remark. The updating decision (36) is to be employed with the ordinary-gradient algorithm and the modified-quasilinearization algorithm, and the updating decision (37) is to be employed with the conjugate-gradient algorithm. Also, the penalty constant estimate (56) is to be employed with the ordinary-gradient algorithm, and the penalty constant estimate (58) and (60) is to be employed with the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

5. Unconstrained Minimization Algorithms

In this section, the unconstrained minimization algorithms employed to compute the displacement vector Δx in connection with Methods MM-2 and MM-3 are described. They are the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. All of these algorithms make use of the augmented penalty function

$$W(x, \lambda, k) = F(x, \lambda) + kP(x), \tag{62}$$

where

$$F(x, \lambda) = f(x) + \lambda^T \varphi(x), \qquad P(x) = \varphi^T(x)\, \varphi(x), \tag{63}$$

and are employed with this understanding: in each cycle of ΔN iterations, the multiplier λ and the penalty constant k are held unchanged, and the vector x is viewed as unconstrained.

5.1. Ordinary-Gradient Algorithm. Let x denote the nominal point, $\tilde{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, and α the stepsize. With this understanding, the ordinary-gradient algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\, \lambda, \tag{64-1}$$

$$P_x(x) = 2\varphi_x(x)\, \varphi(x), \tag{64-2}$$

$$W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x), \tag{64-3}$$

$$p = W_x(x, \lambda, k), \tag{64-4}$$

$$\Delta x = -\alpha p, \tag{64-5}$$

$$\tilde{x} = x + \Delta x. \tag{64-6}$$

For given nominal point x, multiplier λ, and penalty constant k, Eqs. (64) constitute a complete iteration leading to the varied point $\tilde{x}$, providing one specifies the stepsize α.

Descent Properties. To first order, the changes in the function $W(x, \lambda, k)$ and $P(x)$ are given by

$$\delta W(x, \lambda, k) = W_x^T(x, \lambda, k)\, \Delta x, \qquad \delta P(x) = P_x^T(x)\, \Delta x, \tag{65}$$

which, in the light of (64), become

$$\delta W(x, \lambda, k) = -\alpha W_x^T(x, \lambda, k)\, W_x(x, \lambda, k),$$
$$\delta P(x) = -\alpha P_x^T(x)[F_x(x, \lambda) + kP_x(x)]. \tag{66}$$

Recalling Eq. (44), we see that the following orthogonality condition holds:

$$P_x^T(x)\, F_x(x, \lambda) = 0, \tag{67}$$

so that

$$\delta W(x, \lambda, k) = -\alpha W_x^T(x, \lambda, k)\, W_x(x, \lambda, k),$$
$$\delta P(x) = -\alpha k P_x^T(x)\, P_x(x). \tag{68}$$

Since the right-hand sides of (68) are negative, the following inequalities can be enforced for α sufficiently small:

$$W(\tilde{x}, \lambda, k) < W(x, \lambda, k), \qquad P(\tilde{x}) < P(x). \tag{69}$$

While enforcement of (69-1) is mandatory, enforcement of (69-2) is optional, but can be used in order to give greater stability to the ordinary-gradient algorithm.

5.2. Conjugate-Gradient Algorithm. Let x denote the nominal point, $\hat{x}$ the previous point, $\tilde{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the present search direction, $\hat{p}$ the previous search direction, γ the directional coefficient, and α the stepsize. With this understanding, the conjugate-gradient algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\, \lambda, \tag{70-1}$$

$$P_x(x) = 2\varphi_x(x)\, \varphi(x), \tag{70-2}$$

$$W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x), \tag{70-3}$$

$$\gamma = W_x^T(x, \lambda, k)\, W_x(x, \lambda, k)/W_x^T(\hat{x}, \lambda, k)\, W_x(\hat{x}, \lambda, k), \tag{70-4}$$

$$p = W_x(x, \lambda, k) + \gamma\hat{p}, \tag{70-5}$$

$$\Delta x = -\alpha p, \tag{70-6}$$

$$\tilde{x} = x + \Delta x. \tag{70-7}$$

For given nominal point x, multiplier λ, directional coefficient γ, and penalty constant k, Eqs. (70) constitute a complete iteration leading to the varied point $\tilde{x}$, providing one specifies the stepsize α. For the first iteration of a cycle, Eq. (70-4) is bypassed and is replaced by $\gamma = 0$.

Descent Properties. To first order, the changes in the function $W(x, \lambda, k)$ and $P(x)$ are given by Eqs. (65) which, in the light of (70), become

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W_x^T(x, \lambda, k)\, [W_x(x, \lambda, k) + \gamma\hat{p}], \\ \delta P(x) &= -\alpha P_x^T(x)[F_x(x, \lambda) + kP_x(x) + \gamma\hat{p}]. \end{aligned} \tag{71}$$

For the first iteration of a cycle, relation (67) must be applied in conjunction with $\gamma = 0$, leading to

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W_x^T(x, \lambda, k)\, W_x(x, \lambda, k), \\ \delta P(x) &= -\alpha k P_x^T(x)\, P_x(x). \end{aligned} \tag{72}$$

For subsequent iterations, relation (67) does not hold and $\gamma \neq 0$. However, since the previous stepsize is optimized, the following orthogonality relation can be invoked:

$$W_x^T(x, \lambda, k)\, \hat{p} = 0, \tag{73}$$

with the consequence that

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha W_x^T(x, \lambda, k)\, W_x(x, \lambda, k), \\ \delta P(x) &= -\alpha P_x^T(x)[F_x(x, \lambda) + kP_x(x) + \gamma\hat{p}]. \end{aligned} \tag{74}$$

Inspection of (72) and (74) shows that the descent property on the augmented penalty function

$$W(\tilde{x}, \lambda, k) < W(x, \lambda, k) \tag{75}$$

can be enforced for all iterations of a cycle regardless of the value of k.

On the other hand, the descent property on the constraint error

$$P(\tilde{x}) < P(x) \tag{76}$$

can be enforced for the first iteration of a cycle regardless of the value of k and for subsequent iterations only if k is sufficiently large.

5.3. Modified-Quasilinearization Algorithm. Let x denote the nominal point, $\tilde{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the search direction, $\rho = \pm 1$ the direction factor, and α the stepsize. With this understanding, the modified-quasilinearization algorithm is represented by

$$F_x(x, \lambda) = f_x(x) + \varphi_x(x)\, \lambda, \tag{77-1}$$

$$P_x(x) = 2\varphi_x(x)\, \varphi(x), \tag{77-2}$$

$$W_x(x, \lambda, k) = F_x(x, \lambda) + kP_x(x), \tag{77-3}$$

$$F_{xx}(x, \lambda) = f_{xx}(x) + \varphi_{xx}(x)\, \lambda, \tag{77-4}$$

$$P_{xx}(x) = 2[\varphi_{xx}(x)\, \varphi(x) + \varphi_x(x)\, \varphi_x^T(x)], \tag{77-5}$$

$$W_{xx}(x, \lambda, k) = F_{xx}(x, \lambda) + kP_{xx}(x), \tag{77-6}$$

$$W_{xx}(x, \lambda, k)\, A + W_x(x, \lambda, k) = 0, \tag{77-7}$$

$$\rho = \operatorname{sign}[W_x^T(x, \lambda, k)\, A], \tag{77-8}$$

$$p = \rho A, \tag{77-9}$$

$$\Delta x = -\alpha p, \tag{77-10}$$

$$\tilde{x} = x + \Delta x. \tag{77-11}$$

For given nominal point x, multiplier λ, and penalty constant k, Eqs. (77) constitute a complete iteration leading to the varied point $\tilde{x}$, providing one specifies the stepsize α.

Descent Properties. To first order, the changes in the functions $W(x, \lambda, k)$ and $P(x)$ are given by Eqs. (65) which, in the light of (77), become

$$\begin{aligned} \delta W(x, \lambda, k) &= -\alpha \operatorname{sign}[W_x^T(x, \lambda, k)\, A]\, W_x^T(x, \lambda, k)\, A, \\ \delta P(x) &= -\alpha \operatorname{sign}[F_x^T(x, \lambda)\, A + kP_x^T(x)\, A]\, P_x^T(x)\, A. \end{aligned} \tag{78}$$

Inspection of (78) shows that the descent property on the augmented penalty function

$$W(\tilde{x}, \lambda, k) < W(x, \lambda, k) \tag{79}$$

can be enforced regardless of the value of k. On the other hand, the descent property on the constraint error

$$P(\tilde{x}) < P(x) \qquad (80)$$

can be enforced for any k if

$$F_x^T(x, \lambda)\, A/P_x^T(x)\, A > 0 \qquad (81)$$

and only for k sufficiently large if

$$F_x^T(x, \lambda)\, A/P_x^T(x)\, A < 0. \qquad (82)$$

6. Stepsize Determination

For all of the previous algorithms, the position vector at the end of any step can be written as

$$\tilde{x} = x - \alpha p, \qquad (83)$$

where p denotes the search direction. This is a one-parameter family of varied points $\tilde{x}$, for which the augmented penalty function (62) takes the form

$$W(\tilde{x}, \lambda, k) = W(x - \alpha p, \lambda, k) = W(\alpha). \qquad (84)$$

A precise search to be employed with the conjugate-gradient algorithm and an approximate search to be employed with the ordinary-gradient algorithm and the modified-quasilinearization algorithm are described below.

Precise Search. We now assume that a minimum of $W(\alpha)$ exists. Then, we employ some one-dimensional search scheme (for instance, quadratic interpolation, cubic interpolation, or quasilinearization) to determine the value of α for which

$$W_\alpha(\alpha) = 0. \qquad (85)$$

Ideally, this procedure should be used iteratively until the modulus of the slope satisfies any of the following inequalities:

$$|W_\alpha(\alpha)| \leqslant \epsilon_1 \qquad \text{or} \qquad |W_\alpha(\alpha)| \leqslant \epsilon_2 |W_\alpha(0)|, \qquad (86)$$

where ϵ_1 and ϵ_2 are small, preselected numbers. Of course, the value of α satisfying Ineq. (86) must be such that

$$W(\alpha) < W(0). \qquad (87)$$

Approximate Search. Since the rigorous determination of α might require excessive computing time, one might renounce solving Eq. (85) with a particular degree of precision and determine the stepsize in a noniterative fashion. For instance, one might employ a bisection procedure on α, starting from a reference value $\alpha = \alpha_*$, until satisfaction of Ineq. (87) occurs. For the ordinary gradient algorithm, the reference stepsize α_* can be chosen to be the first optimum value of α supplied by the search procedure. For the modified-quasilinearization algorithm, the reference stepsize α_* can be chosen to be $\alpha_* = 1$.

Remark. Optionally, Ineq. (87) can be completed by the additional inequality

$$P(\alpha) < P(0), \qquad (88)$$

which is designed to give greater stability to the algorithm. Inequality (88) can be enforced in the ordinary-gradient algorithm for any k. In the conjugate-gradient algorithm and the modified-quasilinearization algorithm, Ineq. (88) can be enforced only for k sufficiently large.

7. Experimental Conditions

In order to evaluate the theory, seven numerical examples were explored. Each example was solved with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm, which were employed in conjunction with Methods MM-1, MM-2, and MM-3. All of the algorithms were programmed in FORTRAN IV, and the numerical results were obtained using a Burroughs B-5500 computer and double-precision arithmetic.

Starting Point of the Algorithm. For all of the examples, the nominal point chosen to start the algorithm was defined by

$$x_1 = x_2 = \cdots = x_n = 2, \qquad (89)$$

where n denotes the dimension of the vector x.

Convergence of the Algorithm. Convergence of an algorithm was defined through the inequality

$$P(x) + Q(x, \lambda) \leqslant 10^{-8} \qquad (90)$$

for the ordinary-gradient algorithm and the inequality

$$P(x) + Q(x, \lambda) \leqslant 10^{-13} \qquad (91)$$

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

Nonconvergence of the Algorithm. Conversely, nonconvergence of an algorithm was defined by means of the inequalities

$$\text{(a)}\quad \begin{cases} N > 1000 & \text{for the ordinary-gradient algorithm,} \\ N > 200 & \text{for the conjugate-gradient algorithm,} \\ N > 100 & \text{for the modified-quasilinearization algorithm,} \end{cases} \tag{92}$$

or

$$\text{(b)}\quad N_s > 20 \tag{93}$$

or

$$\text{(c)}\quad M > 0.4 \times 10^{69}. \tag{94}$$

Here, N is the iteration number, N_s is the number of bisections of the stepsize α required to satisfy Ineq. (87) [and optionally Ineq. (88)], and M is the modulus of any of the quantities employed in the algorithm. Satisfaction of Ineq. (92) indicates divergence or extreme slowness of convergence; satisfaction of Ineq. (93) indicates extreme smallness of the displacement Δx; and satisfaction of Ineq. (94) indicates exponential overflow. Each of these situations is undesirable.

Convergence of a Cycle. When Method MM-1 was employed, convergence of a cycle was defined through the inequality

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-4} \tag{95}$$

for the ordinary-gradient algorithm and the inequality

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-10} \tag{96}$$

for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

When Methods MM-2 and MM-3 were employed, convergence of a cycle was defined by

$$\Delta N = 1 \tag{97}$$

for the ordinary-gradient algorithm and the modified-quasilinearization algorithm and by either

$$\Delta N = n \tag{98}$$

or

$$W_x^T(x, \lambda_1, k)\, W_x(x, \lambda_1, k) \leqslant 10^{-10}, \tag{99}$$

whichever occurred first, for the conjugate-gradient algorithm.

Search Technique. For the ordinary-gradient algorithm, an approximate search was employed. This consisted of one-step, corrected quasilinearization, followed by a bisection process until the inequality

$$W(\alpha) < W(0) \tag{100}$$

was satisfied.

For the conjugate-gradient algorithm, a precise search was employed. This consisted of multistep, corrected quasilinearization such that, in any given step, the inequality

$$W(\alpha) < W(\alpha_0) \tag{101}$$

was satisfied, where α_0 is the nominal stepsize and α is the varied stepsize. The search was started with $\alpha_0 = 0$ and was terminated when the following stopping condition was satisfied:

$$W_\alpha^2(\alpha) \leqslant W_\alpha^2(0) \times 10^{-4}. \tag{102}$$

For the modified-quasilinearization algorithm, an approximate search was employed. This consisted of assigning the value

$$\alpha = 1 \tag{103}$$

to the stepsize, followed by a bisection process until Ineq. (100) was satisfied.

Penalty Constant Estimate. For Method MM-3, the penalty constant k was estimated with Eq. (56) for the ordinary-gradient algorithm and with Eqs. (58) and (60), with $\pi = 1$, for the conjugate-gradient algorithm and the modified-quasilinearization algorithm.

8. Numerical Examples

In this section seven numerical examples are described. The first example pertains to a quadratic function subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

Example 8.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \tag{104}$$

subject to the constraints

$$x_1 + 3x_2 = 0, \qquad x_3 + x_4 - 2x_5 = 0, \qquad x_2 - x_5 = 0. \tag{105}$$

This function admits the relative minimum $f = 4.0930$ at the point defined by

$$x_1 = -0.7674, \quad x_2 = 0.2558, \quad x_3 = 0.6279, \quad x_4 = -0.1162, \quad x_5 = 0.2558 \tag{106}$$

and

$$\lambda_1 = 2.0165, \quad \lambda_2 = 2.2325, \quad \lambda_3 = -5.9534. \tag{107}$$

Example 8.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \tag{108}$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0. \tag{109}$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1048, \quad x_2 = 1.1966, \quad x_3 = 1.5352 \tag{110}$$

and

$$\lambda_1 = -0.1072 \times 10^{-1}. \tag{111}$$

Example 8.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 \tag{112}$$

subject to the constraints

$$x_1^2 x_4 + \sin(x_4 - x_5) - 2\sqrt{2} = 0, \quad x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0. \tag{113}$$

This function admits the relative minimum $f = 0.2415$ at the point defined by

$$x_1 = 1.1661, \quad x_2 = 1.1821, \quad x_3 = 1.3802, \quad x_4 = 1.5060, \quad x_5 = 0.6109 \tag{114}$$

and

$$\lambda_1 = -0.8553 \times 10^{-1}, \quad \lambda_2 = -0.3187 \times 10^{-1}. \tag{115}$$

Example 8.4. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \tag{116}$$

subject to the constraints

$$x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0, \quad x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0, \quad x_1 x_5 - 2 = 0. \tag{117}$$

This function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

$$x_1 = 1.1911, \quad x_2 = 1.3626, \quad x_3 = 1.4728, \quad x_4 = 1.6350, \quad x_5 = 1.6790 \tag{118}$$

and

$$\lambda_1 = -0.3882 \times 10^{-1}, \quad \lambda_2 = -0.1672 \times 10^{-1}, \quad \lambda_3 = -0.2879 \times 10^{-1}. \tag{119}$$

Example 8.5. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \tag{120}$$

subject to the inequality constraint

$$x_1 \leqslant -1. \tag{121}$$

Introduce the auxiliary variable x_3 defined by

$$x_1 + x_3^2 + 1 = 0. \tag{122}$$

Then, the previous problem can be recast as that of minimizing the function (120) subject to the equality constraint (122). The function (120) admits the relative minimum $f = 0.04$ at the point defined by

$$x_1 = -1, \quad x_2 = 1, \quad x_3 = 0 \tag{123}$$

and

$$\lambda_1 = 0.04. \tag{124}$$

Example 8.6. Consider the problem of minimizing the function

$$f = -x_1 \tag{125}$$

subject to the inequality constraints

$$x_2 \geqslant x_1^3, \quad x_2 \leqslant x_1^2. \tag{126}$$

Introduce the auxiliary variables x_3 and x_4 defined by

$$x_2 - x_1^3 - x_3^2 = 0, \quad x_1^2 - x_2 - x_4^2 = 0. \tag{127}$$

Then, the previous problem can be recast as that of minimizing the function (125) subject to the equality constraints (127). The function (125) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1, \quad x_2 = 1, \quad x_3 = 0, \quad x_4 = 0 \tag{128}$$

and

$$\lambda_1 = -1, \quad \lambda_2 = -1. \tag{129}$$

Example 8.7. Consider the problem of minimizing the function

$$f = \log x_3 - x_2 \tag{130}$$

subject to the equality constraint

$$x_2^2 + x_3^2 - 4 = 0 \tag{131}$$

and the inequality constraint

$$x_3 \geqslant 1. \tag{132}$$

Introduce the auxiliary variable x_1 defined by

$$x_3 = 1 + x_1^2. \tag{133}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log(1 + x_1^2) - x_2 \tag{134}$$

subject to the equality constraint

$$(1 + x_1^2)^2 + x_2^2 - 4 = 0. \tag{135}$$

Note that x_3 has been eliminated from the problem and can be computed *a posteriori* with (133). The function (134) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0, \quad x_2 = \sqrt{3}, \quad x_3 = 1 \tag{136}$$

and

$$\lambda_1 = 1/2\sqrt{3}. \tag{137}$$

9. Results and Conclusions

The examples described in Section 8 were solved with Hestenes' method of multipliers (Method MM-1) and the modified method of multipliers (Methods MM-2 and MM-3) according to the experimental

Table 1. Method MM-1, ordinary-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	24	565	580	(a)	(a)	44
10^{-1}	664	73	847	193	516	(a)	17
10^{0}	350	646	(a)	431	58	946	35
10^{1}	863	(a)	(a)	377	761	762	155
10^{2}	(a)	(a)	(a)	(a)	92	746	774

(a) Number of iterations exceeded 1000.

Table 2. Method MM-2, ordinary-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	14	135	102	138	455	54
10^{-1}	394	21	433	94	39	318	31
10^{0}	94	661	(a)	306	33	278	32
10^{1}	266	998	(a)	(a)	161	129	49
10^{2}	(a)	(a)	(a)	(a)	194	(a)	133

(a) Number of iterations exceeded 1000.

Table 3. Method MM-3, ordinary-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eq. (56)	88	13	193	101	20	275	17

Table 4. Method MM-1, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	25	(a)	(a)	93	(a)	58
10^{-1}	(a)	28	142	138	(a)	(a)	16
10^{0}	(a)	37	103	84	40	184	14
10^{1}	49	54	145	81	71	32	11
10^{2}	20	76	176	95	164	33	18

(a) Number of iterations exceeded 200.

Table 5. Method MM-2, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	16	(a)	(a)	(a)	(a)	55
10^{-1}	(a)	16	76	97	63	(a)	13
10^{0}	(a)	33	79	42	49	150	10
10^{1}	47	48	110	50	61	36	12
10^{2}	20	71	172	49	(a)	28	16

(a) Number of iterations exceeded 200.

Table 6. Method MM-3, conjugate-gradient algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eqs. (58), (60)	43	15	75	53	32	50	14

Table 7. Method MM-1, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	14	(a)	(a)	56	(a)	66
10^{-1}	(a)	11	31	34	21	(a)	25
10^{0}	46	14	37	17	14	86	20
10^{1}	10	13	19	13	26	29	23
10^{2}	4	13	25	22	46	20	35

(a) Number of iterations exceeded 100.

Table 8. Method MM-2, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
10^{-2}	(a)	9	90	(a)	(c)	(a)	46
10^{-1}	(a)	8	20	26	(c)	(a)	36
10^{0}	46	11	(d)	12	16	38	14
10^{1}	10	10	17	11	21	19	20
10^{2}	4	12	25	21	45	16	33

(a) Number of iterations exceeded 100.

(c) Exponential overflow.

(d) Algorithm converged to a different relative minimum.

Table 9. Method MM-3, modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
k	8.1	8.2	8.3	8.4	8.5	8.6	8.7
Eqs. (58), (60)	9	9	13	9	33	21	12

conditions outlined. These methods were employed in conjunction with the ordinary-gradient algorithm, the conjugate-gradient algorithm, and the modified-quasilinearization algorithm. For Methods MM-1 and MM-2, several values of the penalty constant, ranging between 10^{-2} and 10^2, were considered.

The numerical results are presented in Tables 1-9, where the number of iterations required for convergence N_* is shown. Tables 1-3 refer to the ordinary-gradient algorithm, Tables 4-6 refer to the conjugate-gradient algorithm, and Tables 7-9 refer to the modified-quasilinearization algorithm.

Comparison of Methods MM-1 and MM-2 shows that, for given k, Method MM-2 generally exhibits faster convergence than Method MM-1. For both Methods MM-1 and MM-2, the number of iterations for convergence has a minimum with respect to k.

Concerning Method MM-3, the number of iterations for convergence is close to the minimum with respect to k of the number of iterations for convergence of Method MM-2. In this light, Method MM-3 has very desirable characteristics.

Remark 9.1. In Section 4.2, several ways to estimate the Lagrange multiplier λ were presented. The estimate represented by Eqs. (40) and (43) was preferred to that represented by Eq. (51) in order to avoid solving a set of q linear equations in q unknowns. The above statement is significant if one employs the ordinary-gradient algorithm and the conjugate-gradient algorithm, since the computation of the displacement vector Δx is made bypassing the solution of sets of linear equations. On the other hand, if one employs the modified-quasilinearization algorithm, a case can be made for using the estimate (51) for the multiplier, since the computation of the displacement vector Δx requires the solution of a set of n linear equations in n unknowns.

For the sake of discussion, let the modified-quasilinearization algorithm be employed and let the following terminology be used: (a) Method MM-3 is the modified method of multipliers with Lagrange multiplier represented by (40) and (43) and penalty constant represented by (58) and (60); and (b) Method MM-4 is the modified method of multipliers with Lagrange multiplier represented by (51) and penalty constant represented by (58) and (60).

As a point of interest, the seven numerical examples of Section 8 were solved employing both Methods MM-3 and MM-4. The comparative results are given in Table 10, where the number of iterations at convergence N_* is shown. As expected, for a single scalar constraint $(q = 1)$, Methods MM-3 and MM-4 converge to the solution in the same number of iterations. This is precisely the case with Examples 8.2, 8.5, and 8.7. On the other hand, if several constraints are present, Method MM-4 generally converges to the solution in a smaller number of iterations than Method MM-3.

Table 10. Modified-quasilinearization algorithm, number of iterations N_*.

	Examples						
Method	8.1	8.2	8.3	8.4	8.5	8.6	8.7
MM-3	9	9	13	9	13	21	12
MM-4	8	9	10	11	13	15	12

Remark 9.2. Both Methods MM-3 and MM-4 employed in conjunction with the modified-quasilinearization algorithm are interesting in that they generally lead to a minimum point (or at most a saddle point), while standard quasilinearization may also lead to a maximum point.[4] To verify this concept, the following example was considered: Extremize the function

$$f = \sin(\pi x_1/12)\cos(\pi x_2/16) \tag{138}$$

subject to the constraint

$$4x_1 - 3x_2 = 0. \tag{139}$$

This function has a relative minimum $f = -0.5$ at

$$x_1 = -3, \quad x_2 = -4, \quad \text{and} \quad \lambda_1 = -\pi/96. \tag{140}$$

and a relative maximum $f = 0.5$ at

$$x_1 = 3, \quad x_2 = 4, \quad \text{and} \quad \lambda_1 = -\pi/96. \tag{141}$$

The following nominal point was considered:

$$x_1 = 2, \quad x_2 = 2. \tag{142}$$

It was found that Methods MM-3 and MM-4 employed in conjunction with the modified-quasilinearization algorithm led to the relative

[4] Standard quasilinearization involves the solution of a system of $n + q$ linear equations in $n + q$ unknowns (see, for example, Ref. 3).

minimum (140) in $N_* = 5$ iterations. On the other hand, standard quasilinearization (Ref. 3) let to the relative maximum (141) in $N_* = 3$ iterations.

Remark 9.3. In Ref. 11, it was suggested that Hestenes' method of multipliers could be accelerated by employing Eq. (31) at the beginning of each iteration, rather than at the beginning of each cycle. With particular reference to the conjugate-gradient algorithm, this technique has the disadvantage of producing discontinuities in the augmented penalty function $W(x, \lambda, k)$ within a cycle of $\Delta N = n$ iterations. To verify this point, numerical experiments were carried out for the seven examples of Section 8 and one value of the penalty constant, namely, $k = 1$. It was found that, in the majority of the examples, the algorithm of Ref. 11 had not converged to the solution within the required accuracy (91) in the imposed limit of 200 iterations.

References

1. Miele, A., Huang, H. Y., and Heideman, J. C., *Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions*, Journal of Optimization Theory and Applications, Vol. 4, No. 4, 1969.
2. Miele, A., Heideman, J. C., and Levy, A. V., *Combined Conjugate Gradient-Restoration Algorithm for Mathematical Programming Problems*, Ricerche di Automatica, Vol. 2, No. 2, 1971.
3. Miele, A., and Levy, A. V., *Modified Quasilinearization and Optimal Initial Choice of the Multipliers, Part 1, Mathematical Programming Problems*, Journal of Optimization Theory and Applications, Vol. 6, No. 5, 1970.
4. Kelley, H. J., *Method of Gradients*, Optimization Techniques, Edited by G. Leitmann, Academic Press, New York, 1962.
5. Bryson, A. E., Jr., and Ho, Y. C., *Applied Optimal Control*, Blaisdell Publishing Company, Waltham, Massachusetts, 1969.
6. Fiacco, A. V., and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, New York, 1968.
7. Hestenes, M. R., *Multiplier and Gradient Methods*, Journal of Optimization Theory and Applications, Vol. 4, No. 5, 1969.
8. Miele, A., Moseley, P. E., and Cragg, E. E., *Numerical Experiments on Hestenes' Method of Multipliers for Mathematical Programming Problems*, Rice University, Aero-Astronautics Report No. 85, 1971.

Penalty Constant Used in the Penalty Function Method for Mathematical Programming Problems, Rice University, Aero-Astronautics Report No. 90, 1972.

10. Miele, A., Moseley, P. E., and Cragg, E. E., *A Modification of the Method of Multipliers for Mathematical Programming Problems*, Techniques of Optimization, Edited by A. V. Balakrishnan, Academic Press, New York, 1972.
11. Tripathi, S. S., and Narendra, K. S., *Constrained Optimization Using Multiplier Methods*, Journal of Optimization Theory and Applications, Vol. 9, No. 1, 1972.

COMPARISON OF SEVERAL GRADIENT ALGORITHMS FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

Angelo MIELE, J. L. TIETZE and A. V. LEVY
Department of Mechanical and Aerospace
Engineering and Material Science
Rice University
Houston, Texas

COMPARISON OF SEVERAL GRADIENT ALGORITHMS FOR MATHEMATICAL PROGRAMMING PROBLEMS

by

ANGELO MIELE, J.L. TIETZE and A.V. LEVY

This work is dedicated by the senior author to Professor Carlo Ferrari with deep friendship and profound admiration for his many achievements in aerospace engineering and applied mathematics throughout the years.

Abstract

In this paper, the numerical solution of the basic problem of mathematical programming is considered. This is the problem of minimizing a function $f(x)$ subject to a constraint $\varphi(x) = 0$. Here, f is a scalar, x an n-vector, and φ a q-vector, with $q<n$.

Six variations of the sequential gradient-restoration algorithm and the combined gradient-restoration algorithm are considered, and their relative efficiency (in terms of number of iterations for convergence) is evaluated. The variations being considered are as follows:

(i) SGRA-CR, sequential gradient-restoration algorithm, complete restoration,

(ii) SGRA-IR, sequential gradient-restoration algorithm, incomplete restoration,

(iii) SGRA-OR, sequential gradient-restoration algorithm, optional restoration,

(iv) CGRA-NR, combined gradient-restoration algorithm, no restoration,

(v) CGRA-AR, combined gradient-restoration algorithm, alternate restoration,

(vi) CGRA-OR, combined gradient-restoration algorithm, optional restoration.

Evaluation of these algorithms is accomplished through eight numerical examples. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. The results indicate that (a) the inclusion of a restoration phase is necessary for

rapid convergence and (b) the algorithms with alternate restoration or optional restoration are the most efficient among those considered here.

1. Introduction

In previous papers (refs.1,2,3), two basic algorithms for the minimization of constrained functions were developed: the sequential gradient-restoration algorithm (SGRA) and the combined gradient-restoration algorithm (CGRA). The former is an iterative algorithm which consists of the alternate succession of gradient phases and restoration phases; the latter is an iterative algorithm in which the gradient phase and the restoration phase are combined in a single phase.

In the gradient phase of SGRA, one generates a displacement Δx lowering the value of the function, while avoiding excessive constraint violation; in the restoration phase of SGRA, one generates a displacement Δx restoring the constraint to a predetermined accuracy, while avoiding excessive change in the value of the function. On the other hand, in the gradient-restoration phase of CGRA, one generates a displacement Δx lowering the value of the augmented function, while simultaneously reducing the constraint violation.

In this paper, six variations of the sequential gradient-restoration algorithm and the combined gradient-restoration algorithm are considered, and their relative efficiency (in terms of number of iterations for convergence) is evaluated through eight numerical examples. The variations being considered are indicated below:

(i) SGRA-CR, sequential gradient-restoration algorithm, complete restoration,

(ii) SGRA-IR, sequential gradient-restoration algorithm, incomplete restoration,

(iii) SGRA-OR, sequential gradient-restoration algorithm, optional restoration,

(iv) CGRA-NR, combined gradient-restoration algorithm, no restoration,

(v) CGRA-AR, combined gradient-restoration algorithm, alternate restoration,

(vi) CGRA-OR, combined gradient-restoration algorithm, optional restoration.

2. Statement of the Problem

We consider the problem of minimizing the function

$$f = f(x) , \tag{1}$$

subject to the constraint

$$\varphi(x) = 0 , \tag{2}$$

where f is a scalar, x an n-vector, and φ a q-vector, with $q < n$. Here, all vectors are column vectors. It is assumed that the first and second partial derivatives of the functions $f(x)$ and $\varphi(x)$ exist and are continuous and that the constrained minimum exists.

2.1. First-Order Conditions. From theory of maxima and minima, it is known that the above problem is equivalent to that of minimizing the augmented function

$$F(x,\lambda) = f(x) + \lambda^T \varphi(x) , \tag{3}$$

subject to the constraint (2). Here, the q-vector λ is the Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda \tag{4}$$

denotes the gradient of the augmented function, the optimum solution for x and λ must satisfy the relations

$$\varphi(x) = 0 , \quad F_x(x,\lambda) = 0 , \tag{5}$$

which are a system of $n + q$ equations in the $n + q$ components of x and λ. In Eqs. (4)-(5), the gradients f_x and F_x denotes n-vectors and the matrix φ_x is $n \times q$.

2.2. Approximate Solutions. Since the system (5) is generally nonlinear, approximate methods must be employed. In this connection, we introduce here the scalar performance indexes

$$P(x) = \varphi^T(x)\,\varphi(x) , \quad Q(x,\lambda) = F_x^T(x,\lambda)\,F_x(x,\lambda) , \tag{6}$$

which measure the errors in the constraint and the optimum condition, respectively. Then, we observe that $P = 0$ and $Q = 0$ for the optimum solution, while $P > 0$ and/or $Q > 0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to values of x and λ such that

$$P(x) \leq \varepsilon_1 \;, \qquad Q(x,\lambda) \leq \varepsilon_2 \;. \tag{7}$$

Alternatively, (7) can be replaced by

$$R(x,\lambda) \leq \varepsilon_3 \;, \tag{8}$$

where

$$R(x,\lambda) = P(x) + Q(x,\lambda) \tag{9}$$

denotes the cumulative error in the constraint and the optimum condition. In (7)-(8), ε_1, ε_2, ε_3 are small, preselected numbers. Note that, if one chooses $\varepsilon_1 = \varepsilon_2 = \varepsilon_3$, satisfaction of Ineq.(8) implies satisfaction of Ineqs.(7).

3. Description of the Algorithms

In this section, the algorithms being investigated are described.

SGRA-CR. *Sequential gradient-restoration algorithm, complete restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started providing

$$P(x) \leq \varepsilon_1 \;. \tag{10}$$

It involves a single iteration, in which the augmented function is reduced subject to an upper limit for the constraint error, that is,

$$F(\tilde{x},\lambda) < F(x,\lambda) \;, \quad P(\tilde{x}) \leq \varepsilon_4 \;. \tag{11}$$

The symbol x denotes the nominal point, $\tilde{x}$ the varied point, and λ the Lagrange multiplier.

The restoration phase is started providing

$$P(x) > \varepsilon_1 \;. \tag{12}$$

It involves several iterations, in each of which the constraint error is reduced, that is,

$$P(\tilde{x}) < P(x) \;, \tag{13}$$

The restoration phases is terminated whenever Ineq.(10) is satisfied.

Remark. The algorithm is started with a gradient phase if Ineq.(10) is satisfied or a restoration phase if Ineq.(10) is violated. Normally, a gradient phase is followed by a restoration phase. Occasionally, the gradient phase is followed by another gradient phase, that is, the restoration phase is bypassed: this is precisely the case whenever Ineq.(10) is satisfied.

SGRA-IR. *Sequential gradient-restoration algorithm, incomplete restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started regardless of whether Ineq.(10) is satisfied. It involves a single iteration, in which the augmented function is reduced subject to an upper limit on the constraint error, that is,

$$F(\tilde{x},\lambda) < F(x,\lambda) \;, \quad P(\tilde{x}) \leq P(x) + \varepsilon_4 \;. \tag{14}$$

The restoration phase is started only if Ineq.(12) is satisfied. It involves a single iteration, in which the constraint error is reduced in accordance with Ineq.(13).

The starting condition and the bypassing condition for SGRA-IR are identical with those of SGRA-CR (see Remark).

SGRA-OR. *Sequential gradient-restoration algorithm, optional restoration.* This algorithm consists of the alternate succession of gradient phases and restoration phases.

The gradient phase is started providing

$$Z(x,\lambda) \leq 1 \;, \tag{15}$$

where the parameter Z is defined by

$$Z = r\,P(x)/Q(x,\lambda) \;, \tag{16}$$

with

$$r = \varepsilon_2/\varepsilon_1 \;. \tag{17}$$

It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

The restoration phase is started providing

$$Z(x,\lambda) > 1 \;, \tag{18}$$

It involves several iterations, in each of which the constraint error is reduced in accordance with Ineq.(13). The restoration phase is terminated whenever Ineq.(15) is satisfied.

The bypassing condition for SGRA-OR is identical with that of SGRA-CR (see Remark)

CGRA-NR. *Combined gradient-restoration algorithm, no restoration.* In this algorithm, the gradient phase and the restoration phase are combined together in a single phase. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

CGRA-AR. *Combined gradient-restoration algorithm, alternate restoration.* This algorithm consists of the alternate succession of combined gradient-restoration phases and restoration phases.

The combined gradient-restoration phase is started regardless of whether Ineq.(10) is satisfied. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

The restoration phase is started only if Ineq.(12) is satisfied. It involves a single iteration, in which the constraint error is reduced in accordance with Ineq.(13).

The starting condition and the bypassing condition for CGRA-AR are identical with those of SGRA-CR (see Remark).

CGRA-OR. *Combined gradient-restoration algorithm, optional restoration.* This algorithm consists of the alternate succession of combined gradient-restoration phases and restoration phases.

The combined gradient-restoration phase is started providing Ineq.(15) is satisfied. It involves a single iteration, in which the augmented function is reduced in accordance with Ineqs.(14).

The restoration phase is started providing Ineq.(18) is satisfied. It involves several iterations, in each of which the constraint error is reduced in accordance with Ineq.(13). The restoration phase is terminated whenever Ineq.(15) is satisfied.

The bypassing condition for CGRA-OR is identical with that of SGRA-CR (see Remark).

Remark. For the algorithms with optional restoration, the multiplier λ appearing in (15)-(18) is computed as follows. For SGRA-OR, Eq.(19-1) must be solved with $C_1 = 1$ and $C_2 = 0$. For CGRA-OR, Eq.(19-1) must be solved with $C_1 = 1$ and $C_2 = 1$.

4. Generalized Algorithm

Let x denote the nominal point, $\tilde{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, and α the stepsize. With this understanding, the previous algorithms can be represented in the following generalized form:

(19-1) $$\varphi_x^T(x)\varphi_x(x)\lambda + C_1\,\varphi_x^T(x)f_x(x) - C_2\varphi(x) = 0\ ,$$

(19-2) $$p = C_1 f_x(x) + \varphi_x(x)\lambda\ ,$$

(19-3) $$\Delta x = -\alpha p\ ,$$

(19-4) $$\tilde{x} = x + \Delta x\ .$$

For given nominal point x and constants C_1 and C_2, Eqs.(19) constitute a complete iteration leading to the varied point $\tilde{x}$, providing one specifies the stepsize α. The constants C_1 and C_2 depend on the particular algorithm and take the values given in Table 1.

The detailed derivation of Eqs.(19) is presented in refs.1,2,3 and, hence, is not repeated here.

Table 1. Characteristic constants.

Algorithm	Phase	C_1	C_2
SGRA	Gradient	1	0
	Restoration	0	1
CGRA	Gradient-restoration	1	1
	Restoration	0	1

5. Stepsize Determination

For all of the previous algorithms, the position vector at the end of any step can be written as

(20) $$\tilde{x} = x - \alpha p\ .$$

where p denotes the search direction, which is given by (19-2). This is a one-parameter family of varied points $\tilde{x}$, for which the augmented function (3), the constraint error (6-1), and the error in the optimum condition (6-2) take the form

$$F(\tilde{x},\lambda) = F(x - \alpha p, \lambda) = \tilde{F}(\alpha) , \tag{21}$$

$$P(\tilde{x}) = P(x - \alpha p) = \tilde{P}(\alpha) , \tag{22}$$

$$Q(\tilde{x},\lambda) = Q(x - \alpha p, \lambda) = \tilde{Q}(\alpha) . \tag{23}$$

For the gradient phase of a SGRA-algorithm or the combined gradient-restoration phase of a CGRA-algorithm, Ineqs. (11) and (14)' can be written in the general form

$$\tilde{F}(\alpha) < \tilde{F}(0) , \quad \tilde{P}(\alpha) \leq \tilde{P}(0) + \varepsilon_4 . \tag{24}$$

Their satisfaction can be ensured by employing a bisection process, starting from a suitably chosen reference stepsize

$$\alpha = \alpha_0 . \tag{25}$$

For the determination of the reference stepsize, see Section 6.

For the restoration phase of a SGRA-algorithm or a CGRA-algorithm, Ineq. (13) can be written as

$$\tilde{P}(\alpha) < \tilde{P}(0) . \tag{26}$$

Its satisfaction can be ensured by employing a bisection process, starting from the reference stepsize

$$\alpha = 1 . \tag{27}$$

This value reduces the constraint error $P(x)$ to zero, if the constraint function $\varphi(x)$ is linear in x.

6. Reference Stepsize

The search technique outlined in Section 5 for the gradient stepsize employs a bisection process, starting from the reference stepsize (25)', until satisfaction of Ineqs. (24) occurs. A procedure useful to determine this reference stepsize is outlined here and is based on a quadratic

representation of the augmented function associated with the one-parameter family of solutions (20).

Let the function $\tilde{F}(\alpha)$ be represented in the quadratic form

$$\tilde{F}(\alpha) = k_0 + k_1\alpha + k_2\alpha^2 , \tag{28}$$

and let the coefficients of the quadratic be determined so as to match the values of the ordinate and the slope at $\alpha = 0$ and the value of the ordinate at $\alpha = 1$. This yields the relations

$$\tilde{F}(0) = k_0 , \quad \tilde{F}_\alpha(0) = k_1 , \quad \tilde{F}(1) = k_0 + k_1 + k_2 , \tag{29}$$

which imply that

$$k_0 = \tilde{F}(0) , \quad k_1 = -\tilde{Q}(0) , \quad k_2 = \tilde{F}(1) - \tilde{F}(0) + \tilde{Q}(0) . \tag{30}$$

With the coefficients known, the following possibilities arise:

$$\text{(i)}\ k_2 > 0 \quad \text{or} \quad \text{(ii)}\ k_2 < 0 . \tag{31}$$

In Case (i), the quadratic function (28) has a minimum for the following value of the gradient stepsize:

$$\alpha = -k_1/2k_2 . \tag{32}$$

In Case (ii), the quadratic function (28) decreases monotonically with α. This suggests the use of the following reference values for the gradient stepsize:

$$\begin{aligned} \alpha_0 &= -k_1/2k_2 \quad \text{if} \quad k_2 > 0 , \\ \alpha_0 &= 1 \quad \text{if} \quad k_2 < 0 . \end{aligned} \tag{33}$$

7. Experimental Conditions

In order to evaluate the previous algorithms, eight numerical examples were considered. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints. Each example was solved with the three versions of SGRA and the three versions of CGRA outlined in Section 3. All of the algorithms were programmed in FORTRAN IV, and the numerical results were obtained using a Burroughs B - 5500 computer and double-precision arithmetic.

Starting Point. For all of the examples, the nominal point chosen to start an algorithm was defined by

(34) $$x_1 = x_2 = \dots = x_n = 2,$$

where n denotes the dimension of the vector x.

Search Technique. The determination of the gradient stepsize and the restoration stepsize was performed in accordance with Sections 5 and 6. For the gradient phase, the stepsize α was subject to the inequalities

(35) $$\tilde{P}(\alpha) < \tilde{P}(0), \quad \tilde{P}(\alpha) \leq \tilde{P}(0) + 1 .$$

For the restoration phase, the stepsize was subject to the inequality

(36) $$\tilde{P}(\alpha) < \tilde{P}(0) .$$

Convergence. Convergence of an algorithm was defined through the inequalities

(37) $$P(x) \leq 10^{-8}, \quad Q(x,\lambda) \leq 10^{-4}.$$

Nonconvergence. Conversely, nonconvergence of an algorithm was defined by means of the inequalities

(38-1) (a) $$N > 100 ,$$

or

(38-2) (b) $$N_s > 20 ,$$

or

(38-3) (c) $$M > 0.4 \times 10^{69} .$$

Here, N is the iteration number, N_s is the number of bisections of the stepsize α required to satisfy Ineq.(35) or (36), and M is the modulus of any of the quantities employed in the algorithm. Satisfaction of Ineq.(38-1) indicates divergence or extreme slowness of convergence; satisfaction of Ineq.(38-2) indicates extreme smallness of the displacement Δx ; and satisfaction of Ineq.(38-3) indicates exponential overflow. Each of these situations is undesirable.

8. Numerical Examples

In this section, eight numerical examples are described. The first two examples pertain to quadratic functions subject to linear constraints. The remaining examples pertain to nonquadratic functions subject to nonlinear constraints.

Example 8.1. Consider the problem of minimizing the function

(39) $$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 ,$$

subject to the constraints

(40) $$x_1 + 3x_2 = 0 , \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 .$$

This function admits the relative minimum $f = 4.0930$ at the point defined by

(41) $$x_1 = -0.7674, \; x_2 = 0.2558, \; x_3 = 0.6279, \; x_4 = -0.1162, \; x_5 = 0.2558$$

and

(42) $$\lambda_1 = 2.0465 , \; \lambda_2 = 2.2325 , \; \lambda_3 = -5.9534 .$$

Example 8.2. Consider the problem of minimizing the function

(43) $$f = (4x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 ,$$

subject to the constraints

(44) $$x_1 + 3x_2 = 0 , \quad x_3 + x_4 - 2x_5 = 0, \quad x_2 - x_5 = 0 .$$

This function admits the relative minimum $f = 5.3266$ at the point defined by

(45) $$x_1 = -0.9455 \times 10^{-1}, \; x_2 = 0.3151 \times 10^{-1}, \; x_3 = 0.5157 , \\ x_4 = -0.4527, \; x_5 = 0.3151 \times 10^{-1}$$

and

(46) $$\lambda_1 = 3.2779, \; \lambda_2 = 2.9054 , \; \lambda_3 = -7.7478 .$$

Example 8.3. Consider the problem of minimizing the function

(47) $$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 ,$$

subject to the constraint

(48) $$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0 .$$

This function admits the relative minimum $f = 0.3256 \times 10^{-1}$ at the point defined by

(49) $$x_1 = 1.1048,\ x_2 = 1.1966,\ x_3 = 1.5352$$

and

(50) $$\lambda_1 = - 0.1072 \times 10^{-1} .$$

Example 8.4. Consider the problem of minimizing the function

(51) $$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 ,$$

subject to the constraints

(52) $$x_1^2 x_4 + \sin (x_4 - x_5) - 2\sqrt{2} = 0,\ x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0.$$

This function admits the relative minimum $f = 0.2415$ at the point defined by

(53) $$x_1 = 1.1661,\ x_2 = 1.1821,\ x_3 = 1.3802,\ x_4 = 1.5060,\ x_5 = 0.6109$$

and

(54) $$\lambda_1 = -0.8553 \times 10^{-1} ,\ \lambda_2 = -0.3187 \times 10^{-1}.$$

Example 8.5. Consider the problem of minimizing the function

(55) $$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^2 + (x_3 - x_4)^4 + (x_4 - x_5)^4,$$

subject to the constraints

(56) $$x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0,\ x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0,\ x_1 x_5 - 2 = 0 .$$

This function admits the relative minimum $f = 0.7877 \times 10^{-1}$ at the point defined by

(57) $$x_1 = 1.1911,\ x_2 = 1.3626,\ x_3 = 1.4728,\ x_4 = 1.6350,\ x_5 = 1.6790$$

and

(58) $$\lambda_1 = -0.3882 \times 10^{-1} ,\ \lambda_2 = -0.1672 \times 10^{-1},\ \lambda_3 = -0.2879 \times 10^{-1}.$$

Example 8.6. Consider the problem of minimizing the function

(59) $$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 ,$$

subject to the inequality constraint

(60) $$x_1 \leqslant - 1 .$$

Introduce the auxiliary variable x_3 defined by

(61) $$x_1 + x_3^2 + 1 = 0 .$$

Then, the previous problem can be recast as that of minimizing the function (59) subject to the equality constraint (61). The function (59) admits the relative minimum $f = 0.04$ at the point defined by

(62) $$x_1 = -1,\ x_2 = 1 ,\ x_3 = 0$$

and

(63) $$\lambda_1 = 0.04 .$$

Example 8.7. Consider the problem of minimizing the function

(64) $$f = -x_1 ,$$

subject to the inequality constraints

(65) $$x_2 \geqslant x_1^3 ,\quad x_2 \leqslant x_1^2 .$$

Introduce the auxiliary variables x_3 and x_4 defined by

(66) $$x_2 - x_1^3 - x_3^2 = 0,\quad x_1^2 - x_2 - x_4^2 = 0 .$$

Then, the previous problem can be recast as that of minimizing the function subject to the equality constraints (66). The function (64) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1\ ,\ x_2 = 1\ ,\ x_3 = 0\ ,\quad x_4 = 0 \tag{67}$$

and

$$\lambda_1 = -1\ ,\ \lambda_2 = -1\ . \tag{68}$$

Example 8.8. Consider the problem of minimizing the function

$$f = \log x_3 - x_2\ , \tag{69}$$

subject to the equality constraint

$$x_2^2 + x_3^2 - 4 = 0 \tag{70}$$

and the inequality constraint

$$x_3 \geq 1\ . \tag{71}$$

Introduce the auxiliary variable x_1 defined by

$$x_3 = 1 + x_1^2\ . \tag{72}$$

Then, the previous problem can be recast as that of minimizing the function

$$f = \log(1 + x_1^2) - x_2\ , \tag{73}$$

subject to the equality constraint

$$(1 + x_1^2)^2 + x_2^2 - 4 = 0\ . \tag{74}$$

Note that x_3 has been eliminated from the problem and can be computed *a posteriori* with (72). The function (73) admits the relative minimum $f = -\sqrt{3}$ at the point defined by

$$x_1 = 0,\quad x_2 = \sqrt{3},\ x_3 = 1 \tag{75}$$

and

$$\lambda_1 = 1/2\sqrt{3}\ . \tag{76}$$

9. Results and Conclusions

The examples described in Section 8 were solved with the three versions of SGRA and the three versions of CGRA described in Section 3. The numerical results are presented in Tables 2-3, where the number of iterations for convergence N_* is shown. For the eight examples considered, Table 4 shows the cumulative number of iterations for convergences ΣN_*. From the tables, the following conclusions arise: (a) a restoration of some form is necessary for rapid convergence; and (b) while SGRA-CR is the most stable among the algorithms considered here, rapidity of convergence can be increased somewhat if one employs algorithms with alternate restoration or optional restoration.

Table 2. Number of iterations for convergence N_*.

Example	SGRA-CR	SGRA-IR	SGRA-OR
8.1	5	5	5
8.2	8	8	8
8.3	18	14	16
8.4	56	51	42
8.5	8	7	7
8.6	15	12	16
8.7	9	15	9
8.8	11	11	10

Table 3. Number of iterations for convergence N_*.

Example	CGRA-NR	CGRA-AR	CGRA-OR
8.1	17	5	5
8.2	65	8	8
8.3	22	16	16
8.4	36	54	43
8.5	7	7	7
8.6	>100	19	13
8.7	13	7	9
8.8	15	8	10

Table 4. Cumulative number of iterations for convergence ΣN_*.

Algorithm	ΣN_*
SGRA-CR	130
SGRA-1R	123
SGRA-OR	113
CGRA-NR	> 275
CGRA-AR	124
CGRA-OR	111

REFERENCES

1. Miele, A., Huang, H.Y., and Heideman, J.C., *Sequential Gradient-Restoration Algorithm for the Minimization of Constrained Functions, Ordinary and Conjugate Gradient Versions*, Journal of Optimization Theory and Applications, Vol.4, No.4, 1969.

2. Miele, A., and Heideman, J.C., *Mathematical Programming for Constrained Minimal Problems, Part 1, Sequential Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No.59,1969.

3. Miele, A., Heideman, J.C., and Levy, A.V., *Mathematical Programming for Constrained Minimal Problems, Part 2, Combined Gradient-Restoration Algorithm*, Rice University, Aero-Astronautics Report No.60,1970.

This research was supported by the Office of Scientific Research, Office of Aerospace Research, United States Air Force, Grant No.AF-AFOSR-72-2185

REPRINTED

PROCESS DESIGN AND DEVELOPMENT

Comparison of Multiplier and Quasilinearization Methods

Alejandro V. Levy
Antonio Montalvo

Reprinted from I&EC Process Design and Development
Volume 11, October 1972, Page [illegible]

Copyright 1972 by the American Chemical Society and reprinted by permission of the copyright owner

In the above equations, f is a scalar, x an n-vector and φ a q-vector (all vectors are column vectors) where $q < n$. It is assumed that the first and second partial derivatives of the functions f and φ with respect to x exist and are continuous; it is also assumed that the constrained minimum exists.

2.1. Exact First-Order Conditions. From theory of maxima and minima, it is known that the previous problem can be recast as that of minimizing the augmented function

$$F(x,\lambda) = f(x) + \lambda^T\varphi(x) \qquad (3)$$

subject to the constraint (2). Here, λ is a q-vector Lagrange multiplier and the superscript T denotes the transpose of a matrix. If

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda \qquad (4)$$

denotes the gradient of the augmented function (In eq 4, the gradients f_x and F_x denote n-vectors and the matrix φ_x is $n \times q$.), the optimum solution x,λ must satisfy the simultaneous equations

$$\varphi(x) = 0;\ F_x(x,\lambda) = 0 \qquad (5)$$

2.2. Approximate Solutions. In general, the system (5) is nonlinear; consequently, approximate methods must be employed. These are of two kinds: first-order methods (see, for instance, Miele et al. (1969)) and second-order methods. Here, we introduce the scalar quantities

$$P(x) = \varphi^T(x)\varphi(x);\ Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda) \qquad (6)$$

which measure the errors in the constraint and the optimum condition, respectively. We observe that $P = 0$ and $Q = 0$ for the optimum solution, while $P > 0$ and/or $Q > 0$ for any approximation to the solution. When approximate methods are used, they must ultimately lead to values of x,λ such that

$$P(x) \le \epsilon_1;\ Q(x,\lambda) \le \epsilon_2 \qquad (7)$$

Alternately, (7) can be replaced by

$$R(x,\lambda) \le \epsilon_3 \qquad (8)$$

where

$$R(x,\lambda) = P(x) + Q(x,\lambda) \qquad (9)$$

denotes the cumulative error in the constraint and the optimum condition. Here, $\epsilon_1,\epsilon_2,\epsilon_3$ are small, preselected numbers. Note that satisfaction of inequality 8 implies satisfaction of inequalities 7, if one chooses $\epsilon_1 = \epsilon_2 = \epsilon_3$.

3. Review of the SQL-Algorithm

Here, a review of SQL is given. Let x,λ denote the nominal values, and let $\Delta x, \Delta\lambda$ denote the displacements leading from the nominal values to the varied values $\tilde{x},\tilde{\lambda}$. With this understanding, SQL can be summarized as follows.

(i) For given x and λ, compute $f(x)$, $\varphi(x)$, $f_x(x)$, $\varphi_x(x)$.

(ii) Compute

$$F_x(x,\lambda) = f_x(x) + \varphi_x(x)\lambda \qquad (10\text{-}1)$$

$$Q(x,\lambda) = F_x^T(x,\lambda)F_x(x,\lambda) \qquad (10\text{-}2)$$

$$P(x) = \varphi^T(x)\varphi(x) \qquad (10\text{-}3)$$

(iii) If $P(x) \le \epsilon_1$ and $Q(x,\lambda) \le \epsilon_2$, convergence is achieved and the algorithm is stopped; otherwise, go to step (iv).

(iv) Compute $f_{xx}(x)$, $\varphi_{xx}(x)$ and obtain

$$F_{xx}(x,\lambda) = f_{xx}(x) + \varphi_{xx}(x)\lambda \qquad (11)$$

(v) Solve the linear system of equations (of order $n + q$)

$$\begin{bmatrix} F_{xx}(x,\lambda) & \varphi_x(x) \\ \varphi_x^T(x) & 0 \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta\lambda \end{bmatrix} + \begin{bmatrix} F_x(x,\lambda) \\ \varphi(x) \end{bmatrix} = 0 \qquad (12)$$

(vi) Obtain the varied point

$$\tilde{x} = x + \Delta x \qquad (13\text{-}1)$$

$$\tilde{\lambda} = \lambda + \Delta\lambda \qquad (13\text{-}2)$$

and go back to step (i).

4. Review of the MMM-Algorithm

Here, a review of MMM is given in conjunction with the modified quasilinearization algorithm. Let x denote the nominal point, $\tilde{x}$ the varied point, Δx the displacement leading from the nominal point to the varied point, p the search direction, $\rho = \pm 1$ the direction factor, and α the step size. With this understanding, the modified method of the multipliers can be summarized as follows.

(i) Choose a nominal value for x not satisfying the constraint. Compute $f(x)$ and $\varphi(x)$.

(ii) Compute $f_x(x)$, $\varphi_x(x)$ and obtain

$$P(x) = \varphi^T(x)\varphi(x) \qquad (14\text{-}1)$$

$$P_x(x) = 2\varphi_x(x)\varphi(x) \qquad (14\text{-}2)$$

$$F_x(x,\lambda_1) = f_x(x) + \varphi_x(x)\lambda_1 \qquad (14\text{-}3)$$

$$\beta = -P_x^T(x)F_x(x,\lambda_1)/P_x^T(x)P_x(x) \qquad (14\text{-}4)$$

where λ_1 is the Lagrange multiplier of the previous iteration.

(iii) Update the Lagrange multiplier to the value λ_2 given by

$$\lambda_2 = \lambda_1 + 2\beta\varphi(x) \qquad (15)$$

(iv) Compute

$$F_x(x,\lambda_2) = f_x(x) + \varphi_x(x)\lambda_2 \qquad (16\text{-}1)$$

$$Q(x,\lambda_2) = F_x^T(x,\lambda_2)F_x(x,\lambda_2) \qquad (16\text{-}2)$$

(v) Obtain a tentative penalty constant

$$k_0 = |\lambda_2^T\varphi(x)|/P(x) \qquad (17\text{-}1)$$

and the updated penalty constant k_2 as follows

$$k_2 = \min(k_0,k_1) \text{ if } P(x) \le Q(x,\lambda_2) \qquad (17\text{-}2)$$

$$k_2 = \max(k_0,k_1) \text{ if } P(x) > Q(x,\lambda_2) \qquad (17\text{-}3)$$

where k_1 is the penalty constant of the previous iteration.

(vi) Compute the augmented penalty function and its gradient

$$W(x,\lambda_2,k_2) = f(x) + \lambda_2^T\varphi(x) + k_2P(x) \qquad (18\text{-}1)$$

$$W_x(x,\lambda_2,k_2) = f_x(x) + \varphi_x(x)\lambda_2 + k_2P_x(x) \qquad (18\text{-}2)$$

(vii) Compute the second derivative matrices $f_{xx}(x)$, $\varphi_{xx}(x)$ and obtain

$$F_{xx}(x,\lambda_2) = f_{xx}(x) + \varphi_{xx}(x)\lambda_2 \qquad (19\text{-}1)$$

$$P_{xx}(x) = 2\,[\varphi_{xx}(x)\varphi(x) + \varphi_x(x)\varphi_x^T(x)] \qquad (19\text{-}2)$$

$$W_{xx}(x,\lambda_2,k_2) = F_{xx}(x,\lambda_2) + k_2P_{xx}(x) \qquad (19\text{-}3)$$

(viii) Solve the linear system of equations (of order n) and obtain [illegible] A [illegible]

$$W_{xx}(x,\lambda_2,k_2)A + W_x(x,\lambda_2,k_2) = 0 \qquad (20)$$

(ix) Set the stepsize at the value $\alpha = 1$ and compute the direction factor ρ and the search direction p from the relations

$$p = \text{sign}\,[W_x^T(x,\lambda_2,k_2)A] \qquad (21\text{-}1)$$

$$p = \rho A \qquad (21\text{-}2)$$

(x) Compute the displacement Δx, the varied point $\tilde{x}$, and the new value of the augmented penalty function with

$$\Delta x = -\alpha p \qquad (22\text{-}1)$$

$$\tilde{x} = x + \Delta x \qquad (22\text{-}2)$$

$$W(\tilde{x},\lambda_2,k_2) = f(\tilde{x}) + \lambda_2^T\varphi(\tilde{x}) + k_2P(\tilde{x}) \qquad (22\text{-}3)$$

(xi) If

$$W(\tilde{x},\lambda_2,k_2) < W(x,\lambda_2,k_2) \qquad (22\text{-}4)$$

accept the present value of the step size α and go back to step (ii); otherwise, go to step (xii).

(xii) If

$$W(\tilde{x},\lambda_2,k_2) > W(x,\lambda_2,k_2) \qquad (22\text{-}5)$$

bisect the step size as many times as needed (The bisections are started from $\alpha = 1$) and go back to step (x) until inequality (22-4) is satisfied.

Remarks. For the first iteration, the previous values of λ_1 and k_1 are not known. This being the case, one sets

$$\lambda_1 = 0, k_1 = k_0 \qquad (23)$$

5. Operational Count

In this section, we give the number of arithmetic operations performed by each algorithm per iteration. This operational count gives a measure of the computer time needed per iteration. The computer time needed to compute the functions $f(x)$ and $\varphi(x)$ and the derivatives $f_x(x)$, $\varphi_x(x)$, $f_{xx}(x)$, $\varphi_{xx}(x)$ is not considered in the comparison. In the absence of step size bisections, SQL and MMM require the same number of function evaluations per iteration. Therefore, the number of function evaluations is not important for the comparison if the basic criterion is the computer time difference between the two algorithms.

We note that the summation and subtraction operations require much less computer time than the multiplication and division operations. Therefore, the operational count is done only on the basis of the number of multiplications and divisions required to complete one iteration.

5.1. Operational Count for the SQL-Algorithm. Let μ_1 denote the number of multiplications and divisions required by the SQL-algorithm per iteration in order to solve the linear system of equations of order $n + q$ (Isaacson and Keller (1966)). Let μ_2 denote the number of remaining operations. These numbers are given by

$$\mu_1 = (n + q)^3/3 + (n + q)^2 - (n + q)/3 \qquad (24\text{-}1)$$

$$\mu_2 = n^2q/2 + 3nq/2 + n + q \qquad (24\text{-}2)$$

As a consequence, their sum

$$\mu = \mu_1 + \mu_2 \qquad (25)$$

becomes

$$\mu = (n + q)^3/3 + (n + q)^2 - (n + q)/3 + n^2q/2 + 3nq/2 + n + q \qquad (26)$$

5.3. Operational Count for the [illegible]-Algorithm. Let [illegible] denote the number of multiplications and divisions required [illegible] the linear system of equations of order [illegible] and Keller (1960)). Let μ_4 denote the number of remaining [illegible]

$$\mu_3 = n^3/3 + n^2 - n/3 \qquad (27)$$

Table 1. Critical Value of the Number of Constraints

n	q_c	n	q_c	n	q_c
2	2	20	5	200	14
3	3	30	6	300	17
4	3	40	7	400	20
5	3	50	7	500	22
6	3	60	8	600	24
7	4	70	9	700	26
8	4	80	9	800	28
9	4	90	10	900	30
10	4	100	10	1000	32

$$\mu_4 = 3n^2q/2 + 9nq/2 + 3(n + q) + n^2 + 6n + 2q + 5 \qquad (28)$$

As a consequence, their sum (25) becomes

$$\mu = n^3/3 + 3n^2q/2 + 2n^2 + 9nq/2 + 26n/3 + 5q + 5 \qquad (29)$$

5.4. Comparison of Operational Counts. Comparison between SQL and MMM shows that the former requires a larger number of operations to solve the linear system and a smaller number of operations to perform the remaining tasks. It is of interest to compute the difference Δ between the overall number of operations, that is

$$\Delta = (\mu)_{SQL} - (\mu)_{MMM} \qquad (30)$$

From eq 26, 29, and 30, we obtain

$$\Delta = q^3/3 + q^2(n + 1) - (n^2 + nq + 8n + 13q/3 + 5) \qquad (31)$$

Now, let $\Delta = 0$, so that eq 31 becomes

$$q^3/3 + q^2(n + 1) - (n^2 + nq + 8n + 13q/3 + 5) = 0 \qquad (32)$$

For given n, let q_* denote the solution of eq 32 and let q_c denote the closest integer (from below) to q_*, subject to the limitation $q_c \le n$. Inspection of eq 31 and 32 shows that, for any given n

$$\Delta \le 0; 0 \le q \le q_c \qquad (33\text{-}1)$$

$$\Delta > 0, q_c + 1 \le q \le n \qquad (33\text{-}2)$$

The values of q_c are given in Table 1 as a function of the number of variables n. As an example, consider the case $n = 10$, for which $q_c = 4$. If $q \le 4$, SQL requires a smaller number of operations than MMM. On the other hand, if $q \ge 5$, the opposite is true. From the table, it appears that MMM might become superior to SQL for large systems (large n), even when these large systems are moderately constrained.

6. Experimental Conditions

In order to illustrate the characteristics of the SQL and MMM algorithms, nine numerical examples were solved using an IBM 370/165 computer and double precision arithmetic. Both algorithms were programmed in Fortran IV.

Nominal Values. For both algorithms, the nominal [illegible] following values of x

[illegible]

For the SQL-alorithm, the nominal values of the Lagrange multipliers were chosen as $\lambda_i = \gamma$, $i = 1, 2, 3, \ldots, q$, with the following values of γ

$$\gamma = \pm 10, \pm 8, \pm 6, \pm 4, \pm 2 \qquad (34\text{-}2)$$

For the MMM-algorithm, one starts with $\lambda = 0$ (section 4). Therefore, 100 runs were made for each problem with the SQL algorithm and 10 runs were made for each problem with the MMM algorithm.

Stopping Conditions. The convergence criterion for both algorithms was chosen to be

$$P(x) + Q(x, \lambda) \leq 10^{-6} \qquad (35)$$

Conversely, the nonconvergence criteria were chosen as follows:

$$\text{(a) } N \geq 100 \qquad (36\text{-}1)$$

$$\text{(b) } N_b \geq 20 \qquad (36\text{-}2)$$

$$\text{(c) } M \geq 10^{75} \qquad (36\text{-}3)$$

Here, N is the iteration number, N_b is the number of bisections of the stepsize α required to satisfy inequality (22-4), and M is the modulus of any of the quantities employed in the algorithm. Condition (a) implies slow convergence; condition (b) denotes extreme smallness of the displacements and, consequently, slow convergence; and, finally, condition (c) denotes an overflow in the number range of the computer.

Percentage of Success. Each problem was solved several times starting with the nominal values given by eq 34. Let N_r denote the total number of runs made for a given problem, using a given method; let N_s denote the number of runs for which the algorithm succeeded in converging to a relative minimum. Then, the percentage of success p is given by

$$p = N_s/N_r \qquad (37)$$

It is noted that $0 \leq p \leq 1$ and that the higher the value of p, the more robust the algorithm is, since it means that it has a larger range of successful convergence.

Actual Computer Time per Run. Let T_i denote the CPU time in seconds employed in the ith run to solve a given problem with a given algorithm. For a given problem and a given algorithm, the average CPU time per successful run is given by

$$T_{av} = (1/N_s) \sum_{i=1}^{N_s} T_i \qquad (38)$$

Effective Computer Time per Run. The main characteristics of the algorithms considered in this report, namely, robustness (high p) and speed (low T_{av}) can be combined in a single parameter by considering the effective CPU time per run, which, for a given problem and a given algorithm, is given by

$$T_{eff} = T_{av}/p \qquad (39)$$

Thus, an algorithm which is fast and has a high percentage of success has a smaller effective computer time than an algorithm which is slow and has a small percentage of success.

Relative Efficiency of Two Algorithms. In order to compare the SQL and MMM algorithms and arrive at a significant parameter which is machine independent, we introduce the relative efficiency

$$E = (T_{eff})_{SQL}/(T_{eff})_{MMM} \qquad (40)$$

Therefore, $E < 1$ implies that SQL is more efficient than MMM, while the opposite is true for $E > 1$.

7. Numerical Examples

In this section nine numerical examples are described. Examples 7.1 through 7.7 were considered by Miele et al. (1972), while example 7.8 was considered by Luus and Jaakola (1973). For simplicity, scalar notation is used.

Example 7.1. Consider the problem of minimizing the function

$$f = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2 \qquad (41)$$

subject to the constraints

$$x_1 + 3x_2 = 0; \; x_3 + x_4 - 2x_5 = 0; \; x_2 - x_5 = 0 \qquad (42)$$

The function (41) admits the relative minimum $f = 0.4093\,E + 01$ at the point defined by

$$x_1 = -0.7674; \; x_2 = 0.2558; \; x_3 = 0.6279; \; x_4 = -0.1162; \; x_5 = 0.2558 \qquad (43)$$

$$\lambda_1 = 0.2046\,E + 01; \; \lambda_2 = 0.2232\,E + 01; \; \lambda_3 = -0.5953\,E + 01 \qquad (44)$$

Example 7.2. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4 \qquad (45)$$

subject to the constraint

$$x_1(1 + x_2^2) + x_3^4 - 4 - 3\sqrt{2} = 0 \qquad (46)$$

(i) At the point defined by

$$x_1 = 0.1104\,E + 01; \; x_2 = 0.1196\,E + 01; \; x_3 = 0.1535\,E + 01 \qquad (47)$$

$$\lambda_1 = -0.1072\,E - 01 \qquad (48)$$

the function (45) has the relative minimum $f = 0.3256\,E - 01$.

(ii) At the point defined by

$$x_1 = 0.9861\,E - 01; \; x_2 = -0.6954; \; x_3 = 0.1685\,E + 01 \qquad (49)$$

$$\lambda_1 = -0.4029 \qquad (50)$$

the function (45) has the relative minimum $f = 0.2159\,E + 01$.

Example 7.3. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_3 - 1)^2 + (x_4 - 1)^4 + (x_5 - 1)^6 \qquad (51)$$

subject to the constraints

$$x_1^2 x_4 + \sin(x_4 - x_5) - 2\sqrt{2} = 0; \; x_2 + x_3^4 x_4^2 - 8 - \sqrt{2} = 0 \qquad (52)$$

(i) At the point defined by

$$x_1 = 0.1166\,E + 01; \; x_2 = 0.1182\,E + 01; \; x_3 = 0.1380\,E + 01; \; x_4 = 0.1506\,E + 01; \; x_5 = 0.6110 \qquad (53)$$

and

$$\lambda_1 = -0.5553\,E - 01; \; \lambda_2 = -0.3187\,E - 01 \qquad (54)$$

the function (51) has the relative minimum $f = 0.2415$.

(ii) At the point defined by

$$x_1 = 0.1069\,E + 01;\ x_2 = 0.1177\,E + 01;$$
$$x_3 = -0.1231\,E + 01;\ x_4 = 0.1747\,E + 01;$$
$$x_5 = 0.5913 \quad (55)$$

and

$$\lambda_1 = -0.1469\,E - 03;\ \lambda_2 = -0.1774 \quad (56)$$

the function (51) has the relative minimum $f = 0.5523\,E + 01$.

(iii) At the point defined by

$$x_1 = -0.1025\,E + 01;\ x_2 = -0.1017\,E + 01;$$
$$x_3 = 0.1354\,E + 01;\ x_4 = 0.1760;\ x_5 = 0.4531 \quad (57)$$

and

$$\lambda_1 = -0.1126\,E + 01;\ \lambda_2 = -0.2301\,E - 01 \quad (58)$$

the function (51) has the relative minimum $f = 0.4602\,E + 01$.

(iv) At the point defined by

$$x_1 = -0.5263;\ x_2 = -0.9142;\ x_3\ -0.1302\,E + 01;$$
$$x_4 = 0.1603\,E + 01;\ x_5 = 0.4975 \quad (59)$$

and

$$\lambda_1 = -0.1102\,E + 01;\ \lambda_2 = -0.1452 \quad (60)$$

the function (51) has the relative minimum $f = 0.9908\,E + 01$.

Example 7.4. Consider the problem of minimizing the function

$$f = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^3 + (x_3 - x_4)^4 + (x_4 - x_5)^4 \quad (61)$$

subject to the constraints

$$x_1 + x_2^2 + x_3^3 - 2 - 3\sqrt{2} = 0;$$
$$x_2 - x_3^2 + x_4 + 2 - 2\sqrt{2} = 0;\ x_1x_5 - 2 = 0 \quad (62)$$

(i) At the point defined by

$$x_1 = 0.1191\,E + 01;\ x_2 = 0.1362\,E + 01;$$
$$x_3 = 0.1472\,E + 01;$$
$$x_4 = 0.1635\,E + 01;\ x_5 = 0.1679\,E + 01 \quad (63)$$

and

$$\lambda_1 = -0.3882\,E - 01;\ \lambda_2 = -0.1672\,E - 01;$$
$$\lambda_3 = -0.2873\,E - 03 \quad (64)$$

the function (61) has the relative minimum $f = 0.7877\,E - 01$.

(ii) At the point defined by

$$x_1 = 0.2717\,E + 01;\ x_2 = 0.2033\,E + 01;\ x_3 = -0.8479;$$
$$x_4 = -0.4539;\ x_5 = 0.7359 \quad (65)$$

and

$$\lambda_1 = -0.2625\,E + 01;\ \lambda_2 = 0.7106\,E + 01;$$
$$\lambda_3 = -0.2654\,E + 01 \quad (66)$$

the function (61) has the relative minimum $f = 0.1396\,E + 02$.

(iii) At the point defined by

$$x_1 = -0.7651;\ x_2 = 0.2666\,E + 01;\ x_3 = -0.4681;$$
$$x_4 = -0.[illegible]\,E + 01;\ x_5 = -0.[illegible]\,E + 01 \quad (67)$$

and

$$\lambda_1 = -0.2375\,E + 01;\ \lambda_2 = 0.2202\,E + 01;$$
$$\lambda_3 = -0.5085\,E + 01 \quad (68)$$

the function (61) has the relative minimum $f = 0.2745\,E + 02$.

(iv) At the point defined by

$$x_1 = -0.1246\,E + 01;\ x_2 = 0.2422\,E + 01;$$
$$x_3 = 0.1174\,E + 01;\ x_4 = -0.2132;$$
$$x_5 = -0.1604\,E + 01 \quad (69)$$

and

$$\lambda_1 = -0.2016\,E + 01;\ \lambda_2 = -0.6231\,E - 01;$$
$$\lambda_3 = -0.5632\,E + 01 \quad (70)$$

the function (61) has the relative minimum $f = 0.2752\,E + 02$.

(v) At the point defined by

$$x_1 = 0.9194;\ x_2 = -0.2296\,E + 01;\ x_3 = 0.5377;$$
$$x_4 = 0.3384\,E + 01;\ x_5 = 0.2106\,E + 01 \quad (71)$$

and

$$\lambda_1 = -0.2464\,E + 02;\ \lambda_2 = -0.1006\,E + 03;$$
$$\lambda_3 = 0.8790\,E + 01 \quad (72)$$

the function (61) has the relative minimum $f = 0.5652\,E + 02$.

(vi) At the point defined by

$$x_1 = -0.2702\,E + 01;\ x_2 = -0.2989\,E + 01;\ x_3 = 0.1719;$$
$$x_4 = 0.3647\,E + 01;\ x_5 = -0.7401 \quad (73)$$

and

$$\lambda_1 = -0.9508\,E + 02;\ \lambda_2 = -0.5850\,E + 03;$$
$$\lambda_3 = -0.1429\,E + 03 \quad (74)$$

the function (61) has the relative minimum $f = 0.6495\,E + 03$.

Example 7.5. Consider the problem of minimizing the function

$$f = 0.01(x_1 - 1)^2 + (x_2 - x_1^2)^2 \quad (75)$$

subject to constraint

$$x_1 + x_3^2 + 1 = 0 \quad (76)$$

At the point defined by

$$x_1 = -1;\ x_2 = 1;\ x_3 = 0 \quad (77)$$

$$\lambda_1 = 0.04 \quad (78)$$

this function (75) has the relative minimum $f = 0.04$.

Example 7.6. Consider the problem of minimizing the function

$$f = -x_1 \quad (79)$$

subject to constraints

$$x_2 - x_1^3 - x_3^2 = 0;\ x_1^2 - x_2 - x_4^2 = 0 \quad (80)$$

The function (79) admits the relative minimum $f = -1$ at the point defined by

$$x_1 = 1;\ x_2 = 1;\ x_3 = 0;\ x_4 = 0 \quad (81)$$

$$\lambda_1 = -1;\ \lambda_2 = -1 \quad (82)$$

Example 7.7. Consider the problem of minimizing the function

$$f = \log\,(1 + x_1^2) - x_2 \quad (83)$$

subject to the constraint

$$(1 + x_1^2)^2 + x_2^2 - 4 = 0 \quad (84)$$

(i) At the point defined by

Table II. Results for the Examples

	SQL-algorithm			MMM-algorithm		
Example	N_t	N_s	$\sum_{i=1}^{N_s} t_i$	N_t	N_s	$\sum_{i=1}^{N_s} t_i$
7.1	100	100	5.60	10	10	1.33
7.2	100	95	12.92	10	9	1.02
7.3	100	91	67.65	10	8	5.27
7.4	100	76	33.32	10	10	3.03
7.5	100	100	13.59	10	7	1.54
7.6	100	90	27.79	10	10	3.07
7.7	100	51	4.45	10	10	1.17
7.8	100	19	1.57	10	10	1.15
7.9	100	22	50.72	10	10	27.70

Table III. Results for the Examples

	SQL-algorithm			MMM-algorithm			
Example	p	T_{av}	T_{eff}	p	T_{av}	T_{eff}	E
7.1	1.00	0.059	0.058	1.00	0.133	0.133	0.43
7.2	0.95	0.136	0.143	0.90	-0.113	0.125	1.14
7.3	0.91	0.743	0.816	0.80	0.658	0.823	0.99
7.4	0.76	0.427	0.547	1.00	0.303	0.303	1.00
7.5	1.00	0.135	0.135	0.70	0.220	0.314	0.42
7.6	0.90	0.308	0.342	1.00	0.307	0.307	1.11
7.7	0.51	0.087	0.170	1.00	0.117	0.117	1.45
7.8	0.19	0.103	0.545	1.00	0.115	0.115	4.73
7.9	0.22	2.309	10.450	1.00	2.771	2.771	3.77

$$x_1 = 0; x_2 = \sqrt{3} \tag{85}$$

$$\lambda_1 = \sqrt{3}/6 \tag{86}$$

the function (83) has the relative minimum $f = -\sqrt{3}$.

(ii) At the point defined by

$$x_1 = 0; x_2 = \sqrt{3} \tag{87}$$

$$\lambda_1 = -\sqrt{3}/6 \tag{88}$$

the function (83) has the relative minimum $f = \sqrt{3}$.

Example 7.8. Consider the problem of minimizing the function

$$f = -(x_1^2 + x_2^2 + x_3^2) \tag{89}$$

subject to the constraints

$$x_1 + 2x_2 + 3x_3 - 1 = 0; x_1^2 + x_2^2/2 + x_3^2/4 - 4 = 0 \tag{90}$$

(i) At the point defined by

$$x_1 = -0.1835; x_2 = -0.2423 E + 01; x_3 = 0.2014 E + 01 \tag{91}$$

$$\lambda_1 = 0.5260; \lambda_2 = 0.2403 E + 01 \tag{92}$$

the function (89) has the relative minimum $f = -0.9555 E + 01$.

(ii) At the point defined by

$$x_1 = 0.1847; x_2 = 0.2022 E + 01; x_3 = -0.1463 E + 01 \tag{93}$$

$$\lambda_1 = -0.4121; \lambda_2 = 0.2311 E + 01 \tag{94}$$

the function (89) has the relative minimum $f = -0.[illegible] E + 01$.

Example 7.9. Consider the problem of minimizing the function

$$f = -\sum_{i=1}^{10} x_i^2 \tag{95}$$

subject to the constraints

$$\sum_{i=1}^{10} (x_i/e_{ij}) - 1 = 0; j = 1, 2, \ldots, 8 \tag{96}$$

$$\sum_{i=1}^{10} (x_i^2/a_i^2) - 4 = 0 \tag{97}$$

where the coefficients e_{ij} and a_i are given by

$$e_{ij} = 1; i \neq j$$
$$e_{ij} = 2; i = j \tag{98}$$

and

$$a_i^2 = 1 + 3(i - 1)/9; i = 1, 2, \ldots, 10 \tag{99}$$

(i) At the point defined by

$$x_1 = -0.9155 E - 01; x_2 = -0.9155 E - 01;$$
$$x_3 = -0.9155 E - 01; x_4 = -0.9155 E - 01;$$
$$x_5 = -0.9155 E - 01; x_6 = -0.9155 E - 01;$$
$$x_7 = -0.9155 E - 01;$$
$$x_8 = -0.9155 E - 01; x_9 = -0.1814 E + 01;$$
$$x_{10} = 0.3501 E + 01; \tag{100}$$

and

$$\lambda_1 = -0.6332; \lambda_2 = -0.2779; \lambda_3 = -0.6480 E - 01;$$
$$\lambda_4 = 0.7732 E - 01; \lambda_5 = 0.1753; \lambda_6 = 0.2549;$$
$$\lambda_7 = 0.3141; \lambda_8 = 0.3616; \lambda_9 = 0.3579 E + 01 \tag{101}$$

the function (95) has the relative minimum $f = -0.1562 E + 01$.

(ii) At the point defined by

$$x_1 = 0.1064; x_2 = 0.1064; x_3 = 0.1064;$$
$$x_4 = 0.1064; x_5 = 0.1064; x_6 = 0.1064;$$
$$x_7 = 0.1064; x_8 = 0.1064; x_9 = 0.2843 E + 01;$$
$$x_{10} = -0.2642 E + 01 \tag{102}$$

and

$$\lambda_1 = 0.7254; \lambda_2 = 0.5156; \lambda_3 = 0.7462 E - 01;$$
$$\lambda_4 = -0.8202 E - 01; \lambda_5 = -0.2042; \lambda_6 = -0.2914;$$
$$\lambda_7 = -0.3592; \lambda_8 = -0.4134; \lambda_9 = 0.3219 E - 01 \tag{103}$$

the function (95) has the relative minimum $f = -0.1316 E + 01$.

8. Numerical Results and Conclusions

The examples described in section 7 were solved using both the standard quasilinearization algorithm (SQL) and the modified method of multipliers (MMM) in accordance with the experimental conditions outlined in section 6.

Table II shows, for each example and each algorithm, the total number of runs N_t, the number of successful runs N_s, and [illegible] N_s successful runs. Table III shows the [illegible] T_{av} and the [illegible] derived.

(i) [illegible] examples shows that the percentage [illegible] lower than 50% for MMM. On the other hand, for SQL the percentage of success became as low as 19% in example 7.8 and 22% in example 7.9.

On the upper side, MMM achieved a percentage of success of 100% in six examples, while SQL achieved a percentage of success of 100% in two examples (one of which is he obvious linear-quadratic example 7.1).

For the nine examples. the cumulative percentage of success was 92% for MMM and 74% for SQL. From all these data, the higher robustness of MMM is apparent.

(ii) Computer Time per Run. Inspection of the average computer time per run T_{av} shows that SQL is superior to MMM in five examples and inferior in four examples. However, when one looks at the effective computer time per run T_{eff}, the situation is just the opposite: MMM is superior to SQL in six examples and inferior in three examples.

(iii) Relative Efficiency Index. In section 6, the relative efficiency index E was introduced as a way of combining percentage of success with average computer time per run, while arriving at a parameter which is machine independent, to some degree. This efficiency index E was defined as follows

$$E = \frac{(T_{eff})_{SQL}}{(T_{eff})_{MMM}} = \frac{(T_{av})_{SQL}}{(T_{av})_{MMM}} \frac{(p)_{MMM}}{(p)_{SQL}} \qquad (104)$$

This relative efficiency index is defined so that $E < 1$ indicates superiority of SQL with respect to MMM, while $E > 1$ indicates inferiority of SQL with respect to MMM. Inspection of Table III shows that $E < 1$ in three examples and $E > 1$ in six examples. Thus, from the examples investigated, we conclude that MMM compares favorably with SQL.

(iv) The results of Tables II and III must be taken with a rain of salt, since computer times are precise only to 20%. This being the case, values of E in the range $0.8 \le E < 1.2$ are not significant. In particular, this applies to examples 7.2, 7.3, and 7.6.

If one excludes these examples, then conclusion (iii) must be modified as follows: $E < 0.8$ in two examples and $E > 1.2$ in four examples. Therefore, even accounting for possible imprecision in computer time measurements, MMM compares favorably with SQL.

(v) As shown by eq 31 and 32 and Table I, the relative advantage of MMM with respect to SQL should become more apparent for large systems (large n) involving many constraints (large q). Example 7.9, which includes $n = 10$ variables and $q = 9$ constraints supports this point of view (see Table III).

Acknowledgment

This research was supported by the National Science Foundation, Grant No. GP-41158. The authors are indebted to Professor A. Miele for stimulating discussions. This work was done while the first author held a post-doctoral position with the Aero-Astronautics Group of Rice University, Houston, Texas.

Literature Cited

Hestenes, M. R., *J. Optim. Theory Appl.*, 4 (5), 303–320 (1969).

Isaacson, E., Keller, H. B., "Analysis of Numerical Methods," pp 34–37, Wiley, New York, N.Y., 1966.

Luus, R., Jaakola, T. H. I., *Ind. Eng. Chem., Process Des. Dev.*, 12, 380–383 (1973).

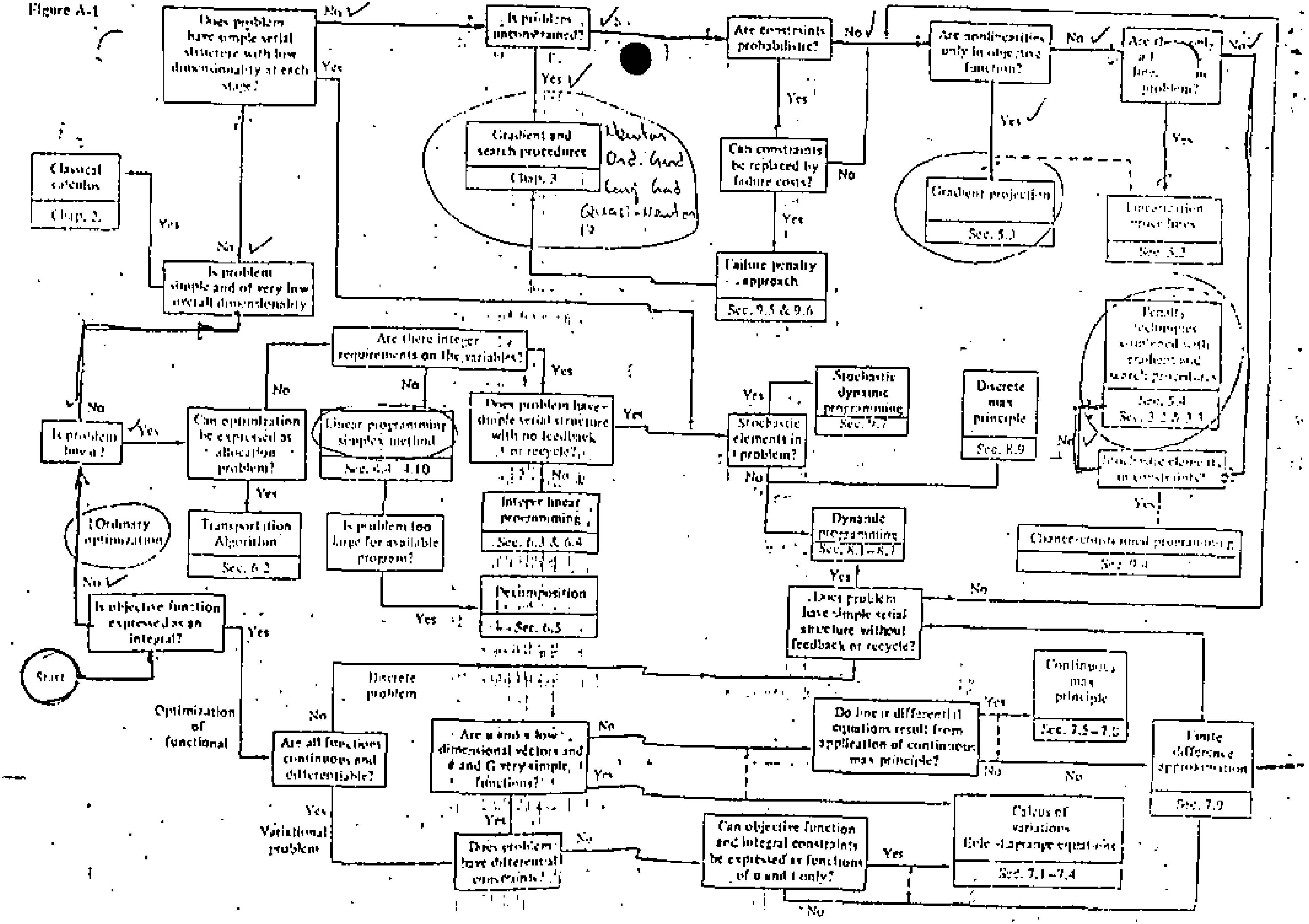
Miele, A., Huang, H. Y., Heideman, J. C., *J. Optim. Theory Appl.*, 4, (4), 213–343 (1969).

Miele, A., Levy, A. V., Iyer, R. R., Well, K. H., "Modified Quasilinearization Method for Mathematical Programming Problems and Optimal Control Problems," "Control and Dynamic Systems. Advances in Theory and Applications," Vol. 9, pp 239–307, C. T. Leondes, Ed., Academic Press, New York, N.Y., 1973.

Miele, A., Moseley, P. E., Levy, A. V., Coggins, G. M., *J. Optim. Theory Appl.*, 10, (1), 1–30 (1972).

Received for review August 26, 1974
Accepted May 14, 1975

Figure A-1


Start
Is objective function expressed as an integral?
No
Yes
Ordinary optimization
Is problem linear?
No
Yes
Is problem simple and of very low overall dimensionality?
Yes
No
Classical calculus
Chap. 2
Does problem have simple serial structure with low dimensionality at each stage?
No
Yes
Is problem unconstrained?
Yes
Gradient and search procedure
Chap. 3
Newton
Ord. Grad
Conj Grad
Quasi-Newton
Are constraints probabilistic?
No
Yes
Can constraints be replaced by failure costs?
No
Yes
Failure penalty approach
Sec. 9.5 & 9.6
Are nonlinearities only in objective function?
No
Yes
Gradient projection
Sec. 5.3
Are there only a few lines in problem?
No
Yes
Linearization
Sec. 5.2
Penalty techniques combined with gradient and search procedures
Sec. 5.4
Sec. 3.2 & 3.3
Stochastic elements in constraints?
Yes
No
Chance-constrained programming
Sec. 9.4
Can optimization be expressed as allocation problem?
No
Yes
Transportation Algorithm
Sec. 6.2
Are there integer requirements on the variables?
No
Yes
Linear programming simplex method
Sec. 4.4 – 4.10
Is problem too large for available program?
Yes
Decomposition
Sec. 6.5
Does problem have simple serial structure with no feedback or recycle?
Yes
No
Integer linear programming
Sec. 6.3 & 6.4
Stochastic elements in problem?
Yes
No
Stochastic dynamic programming
Sec. 9.7
Discrete max principle
Sec. 8.9
Dynamic programming
Sec. 8.1 – 8.7
Does problem have simple serial structure without feedback or recycle?
Yes
No
Optimization of functional
Discrete problem
Are all functions continuous and differentiable?
No
Yes
Variational problem
Are u and x low dimensional vectors and f and G very simple functions?
No
Yes
Does problem have differential constraints?
Yes
No
Do linear differential equations result from application of continuous max principle?
Yes
No
Continuous max principle
Sec. 7.5 – 7.8
Finite difference approximation
Sec. 7.9
Can objective function and integral constraints be expressed as functions of u and t only?
Yes
No
Calculus of variations
Euler-Lagrange equations
Sec. 7.1 – 7.4

Reference: Introduction to Optimization theory - B.S. Gottfried and J. Weisman

CAPITULO 8

Diseño óptimo mediante computadora y su aplicación a la ingeniería mecánica

Enrique Chicurel Uziel

8.1 El problema de diseño

El problema de diseño tradicional se refiere a la concepción y determinación de las características físicas de un sistema en forma tal que satisfaga básicamente los requisitos de funcionamiento cumpliendo con ciertas restricciones.

El sistema a diseñar puede ser, entre otros, una estructura, un aparato, una máquina, un sistema de control, un circuito, o bien una combinación de dos o más de los mismos.

Las restricciones se pueden referir a las limitaciones en las características mecánicas o eléctricas de los materiales, a los tamaños y capacidades de los componentes que se encuentran comercialmente, al espacio disponible, al medio ambiente, a consideraciones de seguridad y a muchas otras.

8.2 Necesidad de optimizar en diseño

Por lo general, no basta con satisfacer los requisitos de funcionamiento sino que, además, hay que satisfacerlos de acuerdo con algún criterio que depende del objeto que se persigue y la aplicación del sistema, como por ejemplo, el diseño debe ser lo más económico posible, o bien lo más eficiente, o lo más compacto, o lo más seguro. Es decir que, esencialmente, el problema de diseño es un problema de optimización, proceso que requiere una gran cantidad de operaciones aritméticas.

Cabe señalar que un diseño se mejora durante la evolución del producto, es decir, mediante el desarrollo de nuevos prototipos en donde se utilizan las experiencias logradas en la producción y aplicación de los anteriores. Este es un proceso lento y costoso pero inevitable. Sin embargo, las técnicas matemáticas de optimización y la computadora electrónica han hecho posible abreviar y abaratar dicho proceso.

Estas circunstancias, aunadas a la existencia de la fuerte competencia entre fabricantes de los países altamente industrializados ha traído como consecuencia un gran auge en el desarrollo de dichas técnicas.

Existen tres clases de optimización, a saber:

1. Optimización de magnitud
2. Optimización de forma
3. Optimización de configuración

Lo que comúnmente se entiende por optimización se refiere a la primera categoría. Sin embargo, todas son dignas de consideración.

8.3 Optimización de magnitud

Para concretar las ideas anteriores y lograr hacer optimizaciones prácticas a la mayor brevedad, echemos una ojeada a unos ejemplos de optimización de magnitud.

Ejemplo 1

Se desea diseñar una caja de cartón cilíndrica lo más económica en material posible para contener 1000 cm³ de cierto producto. Los anaqueles que se acostumbran utilizar para almacenar dichas cajas tienen una altura de 6 cm entre repisas.

Determinar las proporciones óptimas.

Solución

Formulación inicial:

$A = 2(\pi rh + \pi r^2)$	Criterio de optimización. (A, función a minimizar).
$V = \pi r^2 h$	Requisito de funcionalidad.
$h \leq h_{lim}$	Limitación.

Se especifican: V, $h_{lím}$

Variables: r, h

Variable restringida: h

Variable libre: r

Para obtener la formulación final, eliminamos la variable libre r

Formulación final:

$A = 2\,(\sqrt{\pi V}h^{\frac{1}{2}} + Vh^{-1})$ Criterio de optimización.

$h \leq h_{lím}$ Limitación.

Se especifican: V y $h_{lím}$

Variable: h

Recurriendo al cálculo se obtiene:

$$\frac{dA}{dh} = \sqrt{\pi V}\,h_c^{-\frac{1}{2}} - 2V\,h_c^{-2} = 0$$

$h_c = 1.085V^{\frac{1}{3}} = 10.85$ cm

puesto que $h_{lím} = 6$ cm

$h_c > h_{lím}$

Por tanto h_c no es la altura óptima y según podemos ver en la gráfica de la figura 1

$h_{opt} = h_{lím} = 6$ cm

y por lo tanto

$$r_{opt} = \sqrt{\frac{V}{\pi h_{opt}}} = \sqrt{\frac{1000}{\pi(6)}} = 7.25 \text{ cm}$$

$$A_{opt} = 2\,(\pi r_{opt}\,h_{opt} + \pi r^2_{opt}) = 2\,[\pi(7.25)\,(6) + \pi(7.25)^2] = 609 \text{ cm}^2$$

En la figura 1 apreciamos que cualquier punto en la región de diseño factible satisface el requisito de funcionalidad de la caja: contener 1000 cm³. Sin embargo, sólo cuando h = 6 cm se satisface el criterio de optimización: máxima economía.

Un diseño en donde no se utilizara ningún criterio de optimización podría resultar pésimo; considérese, por ejemplo, la parte izquierda de la región de diseño factible

El método que se ilustra es debido al profesor R. C. Johnson, del Instituto Politécnico de Worcester.[17]

Ejemplo 2

Una viga de acero simplemente apoyada de 100" de largo, de sección rectangular, con 3" de peralte y 1" de ancho, soporta una carga concentrada de 720 lb en el centro.

1. Calcular σ máximo, σ = esfuerzo normal a una sección.
2. Calcular τ_t máximo, τ_t = esfuerzo cortante transversal.

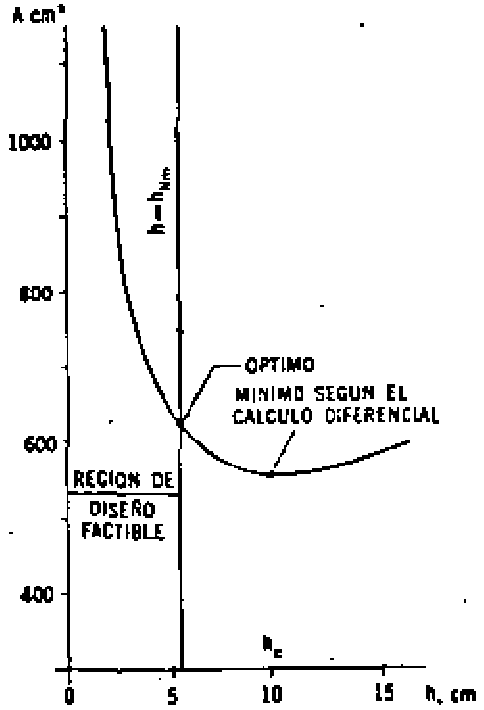


Figura 1. Optimización de caja de cartón cilíndrica.

3. Establecer una relación entre el esfuerzo cortante máximo τ en función de σ y τ_t para cualquier punto.
4. Comparar los valores numéricos obtenidos en los incisos 1 y 2 y, a la luz de ello, simplificar la relación que se estableció en el inciso 3.
5. Considerar ahora que las dimensiones de la sección de la viga no han sido determinadas. Utilizando la relación del inciso 4 determinar las dimensiones de la sección, tal que:

   El costo (peso) sea mínimo
   τ no exceda a 8000 lb/in²
   El peralte h no exceda a 3"
   La deflexión máxima no exceda 0.45"

Proceder de la siguiente manera:

a) Obtener la formulación inicial completa.
b) A partir de la inicial obtener la formulación final completa.
c) Dibujar a escala la región de diseño factible.
d) Calcular los valores óptimos de: el área seccional (índice del costo), las dimensiones, y el esfuerzo τ.

6. Si se exige ahora que la deflexión no exceda a 0.2",

a) Dibujar la nueva región de diseño factible.
b) Recalcular los valores requeridos en el inciso 5d.

olución

Nomenclatura

A = área seccional de la viga.
a = área seccional de la viga arriba (o abajo) del punto para el cual se calcula τ.
b = ancho de la viga.
c = distancia medida desde el eje neutro al punto más alejado del mismo.
Δ = deflexión máxima de la viga.
E = módulo de Young.
h = peralte de la viga.
I = momento de inercia del área seccional de la viga.
L = longitud de la viga.
M = momento flexionante.
P = carga transversal sobre la viga.
Q = primer momento del área "a" respecto al eje neutro.
V = fuerza cortante.
$\bar{y}$ = distancia centroidal del área "a" medida desde el eje neutro.
$(\ )_L$ = valor límite.

1. $$\sigma_{máx} = \frac{M_{máx}\,c}{I} \qquad c = \frac{h}{2} \qquad I = \frac{bh^3}{12}$$

$$M_{máx} = \frac{PL}{4} = \frac{(720)(100)}{4} = 18\,000 \text{ lb-in.}$$

$$\therefore \sigma_{máx} = 6\,\frac{M_{máx}}{bh^2} = \frac{6(18\,000)}{(1)(3)^2} = 12\,000 \text{ lb/in}^2$$

2. $$\tau_{t\,máx} = \frac{V_{máx}\,Q_{máx}}{I\,b}$$

$$V_{máx} = \frac{P}{2}$$

$$Q_{máx} = a\,\bar{y} = \left(\frac{bh}{2}\right)\frac{h}{4} = \frac{bh^2}{8}$$

$$\tau_{t\,máx} = \frac{3}{4}\,\frac{P}{bh} = \frac{3}{4}\,\frac{(720)}{(1)(3)} = 180 \text{ lb/in}^2$$

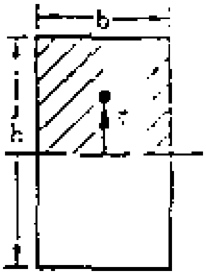


Figura 2. Ejemplo 2. Área "a" y su distancia centroidal.

3. Para cualquier punto y mediante el círculo de Mohr se obtiene

$$\tau = \sqrt{\left(\frac{\sigma}{2}\right)^2 + \tau_t^2}$$

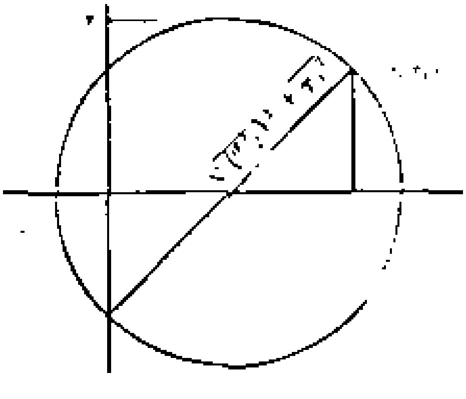


Figura 3. Ejemplo 2. Círculo de Mohr.

4. $\tau_1 <<< \sigma$ y además

cuando $y = c$ $\quad \sigma$ es máximo y $\tau_1 = 0$

cuando $y = 0$ $\quad \sigma = 0$ $\quad$ y τ_1 es máximo

$\therefore$ τ_1 se puede considerar nulo y

$$\tau = \frac{\sigma}{2}$$

5. Consideraciones preliminares:

$$\tau = \frac{\sigma}{2} = \frac{3M}{bh^2}$$

$$\Delta = \frac{PL^3}{48EI} = \frac{ML^2}{Ebh^3}$$

Formulación inicial

$A = bh$ Criterio de optimización (A, función a minimizar)

$$\left.\begin{aligned} \tau &= 3\,\frac{M}{bh^2} \\ \Delta &= \frac{ML^2}{Ebh^3} \end{aligned}\right\}\text{Requisitos de funcionalidad}$$

$$\left.\begin{aligned} \tau &\leq \tau_L \\ \Delta &\leq \Delta_L \\ h &\leq h_L \end{aligned}\right\}\text{Limitaciones}$$

Se especifican: E, M, L, τ_L, Δ_L, h_L.

Variables: b, h, τ, Δ

Variables libres: b

Nótese que *las variables libres son sencillamente las que no se indican como limitadas.*

Eliminando las variables libres se obtiene la:

Formulación final

$A = \dfrac{3\,M}{\tau\,h}$ Criterio de optimización (A, función a minimizar)

$= \dfrac{L^2\,\tau}{3\,Eh}$ Requisito de funcionalidad.

$$\left.\begin{aligned} \tau &\leq \tau_L \\ \Delta &\leq \Delta_L \\ h &\leq h_L \end{aligned}\right\}\text{Limitaciones}$$

Se especifican: los parámetros E, M, L y los valores límites τ_L, Δ_L, h_L.

Variables independientes: τ, h

Variable dependiente: Δ

Observamos que, a diferencia del ejemplo 1, ahora tenemos dos variables independientes y que, por lo tanto, tendrémos una región de diseño factible en dos dimensiones.

Determinemos pues las fronteras de dicha región.

Las fronteras referentes a las variables independientes quedan definidas inmediatamente por las limitaciones, de donde se obtienen sus ecuaciones:

$$\tau = \tau_L = 8\,000$$

$$h = h_L = 3$$

La frontera referente a la variable dependiente se obtiene sustituyendo la limitación correspondiente en la ecuación referente al requisito de funcionalidad

$$\tau = \frac{3\,E\Delta_L}{L^2}\,h = \frac{3\,(30 \times 10^6)\,(0.45)}{(100)^2}\,h$$

$$\tau = 4\,060\ h$$

La región de diseño factible se muestra en la figura 4.

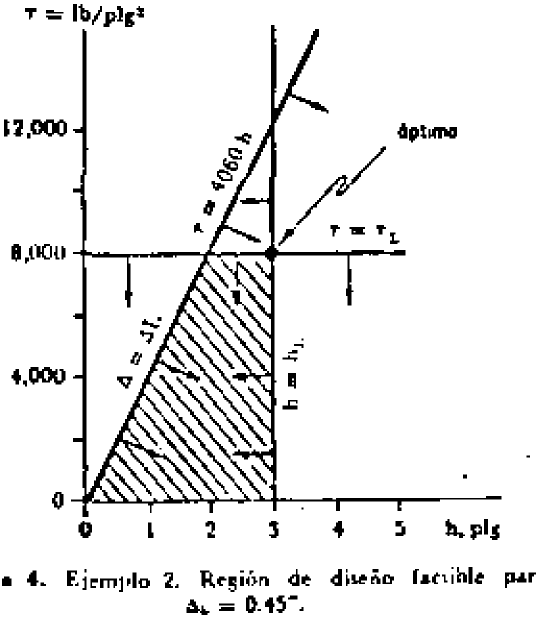


Figura 4. Ejemplo 2. Región de diseño factible para $\Delta_L = 0.45''$.

Puesto que la función objetiva es

$$A = \frac{3\,M}{\tau\,h} = \frac{54\,000}{\tau\,h}$$

A es mínimo cuando h y τ son ambas máximas dentro de la región de diseño factible o sea h = 3", τ = 8000 lb/in².

$$\therefore A_{\text{óptimo}} = \frac{54\,000}{3(8\,000)} = 2.23 \text{ in}^2$$

y el valor correspondiente de b es:

$$b = \frac{A}{h} = \frac{2.23}{3} = 0.75''$$

6. Para $\Delta_L = 0.2''$

$$\tau = 1\,800\,h$$

y la nueva región de diseño factible se muestra en la figura 5.

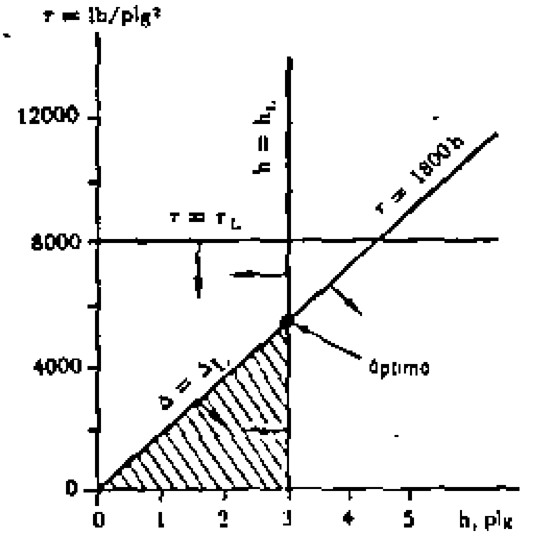


Figura 5. Ejemplo 2. Región de diseño factible para $\Delta_L = 0.2''$.

Una vez más el óptimo se obtiene de la relación:

$$A_{\text{opt}} = \frac{54\,000}{\tau_{\text{MAX}}\,h_{\text{MAX}}}$$

como se aprecia en la figura 5

$$h_{\text{MAX}} = 3''$$

y τ_{MAX} se obtiene de la intersección de las fronteras:

$$\tau = 1\,800\,h$$

$$h = h_L = 3''$$

es decir

$$\tau_{\text{MAX}} = 1\,800\,h_L = 5\,400 \text{ lb/in}^2$$

$$\therefore A_{\text{opt}} = \frac{54\,000}{(5\,400)(3)} = 3.33''$$

$$b_{\text{opt}} = \frac{A}{h} = \frac{3.33}{3} = 1.11''$$

Resumiendo, los valores óptimos son:

Δ_L, in	A, in²	h, in	b, in	Δ, in	τ, lb/in²
0.45"	2.23	3	0.75	0.296	8 000
0.20"	3.33	3	1.11	0.2	5 400

8.4 Consideraciones algebraicas

Si en los anteriores ejemplos se consideraron exclusivamente los requisitos de funcionalidad, se tiene un sistema con un número mayor de variables que de ecuaciones.

A continuación se resumen estas cantidades en relación a los dos primeros ejemplos.

Ejemplo	*Formulación*	*N° de variables*	*N° de ecuaciones requisito de funcionalidad*
1	Inicial	2	1
1	Final	1	0
2	Inicial	4	2
2	Final	3	1

Lo anterior quiere decir que existe un número infinito de soluciones de dicho sistema.

Si se considera ahora el conjunto de las ecuaciones referentes a los requisitos de funcionalidad y las desigualdades referentes a las limitaciones, se tiene que se han reducido los valores que pueden asumir las variables pero el número de soluciones sigue siendo infinito.

En diseño tradicional, no se define ningún criterio para seleccionar una sola de las soluciones posibles.

En cambio, en diseño óptimo sí se fija con toda precisión dicho criterio. El proceso de optimización consiste en la búsqueda de dicha solución.

Ejemplo 3

Se requiere diseñar un resorte helicoidal para un convertidor de par.* Ya ha sido seleccionada la cana-

ptimo mediante computadora

resorte, así como la fuerza máxima de

a montado en una flecha cuyo diámetro
so lo que el diámetro interior D_1 queda
nas por consideraciones de espacio existe
no debe exceder el diámetro exterior

cionó el material, por lo cual ya queda-
dos el esfuerzo permisible τ_p y el módulo
Sin embargo, el alambre viene única-
diferentes diámetros d_i.
hacer mínima la longitud cerrada del re-
ndo todas las espiras están en contacto)
e a la carga máxima Q.
ones que se emplean en el diseño de re-
ales son:

$$\frac{G\,d^4}{8ND^3_m}$$

$$\frac{QD_mW}{\pi d^3}$$

$$\frac{4D_m - d}{(D_m - d)} + 0.615\,\frac{d}{D_m}$$

etro del alambre.
etro medio de la espira.
ero de espiras.
or de concentración de esfuerzo de Wahl.

la formulación inicial.

de la formulación inicial obtener la final.

dos los siguientes valores:

38 lb/in
418 lb
2 in
4 in
40,000 lb/in²
12 × 10⁶ lb/in²
$\frac{1}{32}, \frac{1}{16}, \frac{3}{32}, \ldots 1$ in

ción inicial

Criterio de optimización
(L_c función a minimizar)

n

$$\left.\begin{aligned} k &= \frac{Gd^4}{8D^3_mN} \\ \tau &= \frac{8QD_mW}{\pi d^3} \\ W &= \frac{4D_m - d}{4(D_m - d)} + 0.615\,\frac{d}{D_m} \\ D^*_m &= \frac{Di + De}{2} \end{aligned}\right\}\ \text{Requisitos de funcionalidad}$$

$$\left.\begin{aligned} De &= Di + 2d \\ Di &\geqslant D_f \\ \tau &\leq \tau_p \\ De &\leq D_{e\,\max} \\ d &= d_1, d_2, \ldots d_N \end{aligned}\right\}\ \text{limitaciones}$$

Se especifican: k, Q, G, D_f, τ_p, $D_{e\,\max}$

Variables: N, d, τ, D_m, W, Di, De

Variables libres: N, W, D_m.

Eliminando las variables libres se obtiene:

Formulación final

$$L_c = \frac{Gd^5}{8k(D_i + d)^3} \qquad \text{Criterio de optimización. } L_c\text{, función de minimizar.}$$

$$\left.\begin{aligned} \tau &= \frac{4Q(D_i + d)}{\pi d^3}\left[\frac{4D_i + 3d}{2D_i} + 1.23\,\frac{d}{D_i + d}\right] \\ D_e &= D_i + 2d \end{aligned}\right\}\ \text{Requisitos de funcionalidad}$$

$$\left.\begin{aligned} D_i &\geq D_f \\ \tau &\leq \tau_p \\ D_e &\leq D_{e\,\max} \\ d &= d_1, d_2, d_3, \ldots d_N \end{aligned}\right\}\ \text{Limitaciones}$$

Se especifican los parámetros: k, Q, G, así como los valores límite: D_f, τ_p, $D_{e\,\max}$, $d_1, d_2, \ldots d_N$

Variables independientes: D_i, d

Variables dependientes: τ, D_e

Substituyendo valores numéricos obtenemos:

Formulación final

$$L_c = 3.84 \times 10^4\,\frac{d^5}{(D_i + d)^3}$$

Criterio de optimización
(L_c, función a optimizar)

$$\left.\begin{aligned} \tau &= 531\,\frac{(D_i + d)}{d^3}\left[\frac{4D_i + 3d}{2D_i} + \frac{1.23d}{D_i + d}\right] \\ D_e &= D_i + 2d \end{aligned}\right\}$$

$$\left.\begin{aligned} D_i &\geq 2 \\ \tau &\leq 4 \times 10^4 \\ D_e &\leq 4 \\ d &= \frac{1}{32}, \frac{1}{16}, \frac{3}{32}, \ldots 1 \end{aligned}\right\} \text{Limitaciones}$$

Variables independientes: D_i, d

Variables dependientes: τ, D_e

Determinamos primero la región de diseño factible en el sistema de coordenadas D_i, d. Se obtiene una frontera de la limitación en D_i y es la recta

$$D_i = 2$$

Nótese que por referirse a una variable independiente esta frontera es constante.

Las otras fronteras se obtienen de substituir las limitaciones de las variables dependientes en las ecuaciones referentes a los requisitos de funcionalidad.

$$4 = D_i + 2d$$

$$4 \times 10^4 = 531\,\frac{(D_i + d)}{d^3}\left(\frac{4D_i + 3d}{2D_i} + \frac{1.23d}{D_i + d}\right).$$

Como estas fronteras se refieren a las variables dependientes, no son constantes, y, en general, son curvas. Las fronteras se muestran en la figura 6. Analizando las limitaciones se concluye que la región de diseño factible es la que se muestra encerrada por las líneas sólidas.

Puesto que la función a optimizar ya no es tan sencilla, no se puede determinar el punto óptimo mediante una mera inspección de la ecuación referente al criterio de optimización.

Si no se tiene idea de optimización, lo primero que se ocurre es obtener los contornos de nivel de L_e. La ecuación se obtiene de la relación correspondiente al criterio de optimización

$$D_i = \left(3.84 \times 10^4\,\frac{d^3}{L_e}\right)^{\frac{1}{2}} - d$$

En la figura 6 se muestran cuatro contornos correspondientes a los valores de L_e = 40, 70, 150, 400 plg.

Si imaginamos a la región de diseño factible llena de contornos, llegamos a la conclusión que el óptimo se encuentra en el extremo derecho inferior de la región de diseño factible, es decir, d ≃ 0.48 plg.

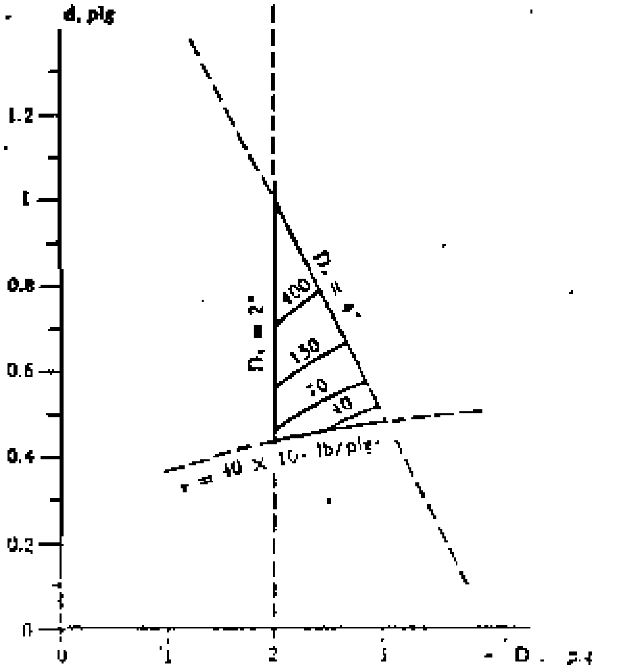


Figura 6. Ejemplo 3. Región de diseño factible y contornos de nivel de L_e.

Pero consideremos ahora a d como realmente es: una variable discreta. Esto significa que la región de diseño factible consta realmente de los segmentos de recta que se muestran en la figura 7.

Como los valores de L_e suben más lentamente a lo largo de la frontera inferior que a lo largo de la frontera derecha y como

$$\frac{15''}{32} < d < \frac{1''}{2}$$

el óptimo debe quedar en el extremo derecho del segmento inferior, por lo que las proporciones óptimas son:

$$\begin{aligned} d &= 15/32'' \\ D_i &= 2.7'' \\ D_e &= 3.6'' \\ L_e &= 28'' \end{aligned}$$

calculadas a partir de la formulación final

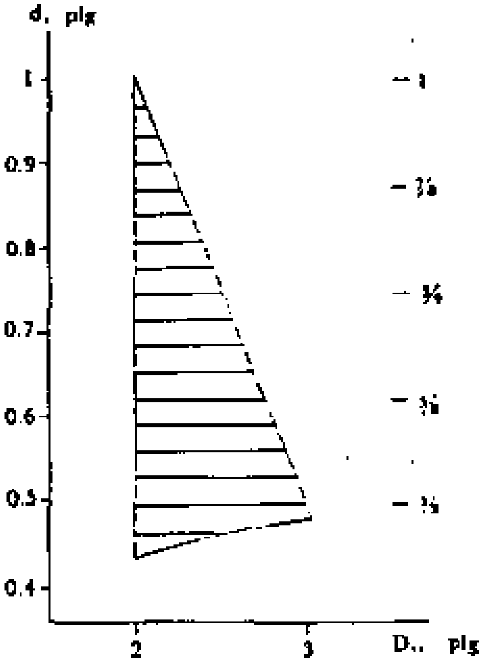


Figura 7. Ejemplo 3. Región de diseño factible tomando en cuenta los valores discretos de d.

6.6 Métodos de exploración local

La solución anterior, aunque muy ilustrativa, es sumamente ineficiente pues requiere de una gran cantidad de cómputos: por un lado, para determinar las fronteras (de hecho, la frontera inferior impuesta por la limitación en τ se calculó por tanteos); y, por otro lado, para determinar suficientes curvas de nivel para obtener una idea del comportamiento de la función a optimizar. El método sería muy difícil y laborioso si tuviéramos que lidiar con tres variables independientes y prácticamente imposible si fueran más.

Todas estas dificultades resultan del hecho que se trató de examinar la topografía de la región completa.

Hay métodos mucho más eficientes. Para entender la idea fundamental recurramos a una analogía.

Baldomero y Agripina juegan. Baldomero se halla con los ojos vendados en un punto en el interior de un predio grande, cercado, en una región montañosa y árida. Agripina le pide que, sin quitarse la venda, encuentre el punto más bajo del predio.

Baldomero se sienta en el suelo y con sus manos palpa el terreno en derredor suyo y escoge la dirección a lo largo de la cual parece descender más el terreno. Camina un trecho como de dos metros y se vuelve a [illegible]ar para palpar su derredor una vez más, repitiendo el proceso varias veces hasta encontrar un punto tal que en todas direcciones sube el terreno o bien se topa con cerca. Eufórico, le grita a Agripina que ya encontró el lugar señalado. Agripina le responde que no es cierto, pero lo conduce a otro punto inicial para darle una oportunidad más. Baldomero procede una segunda vez de idéntica manera hasta llegar a otro punto desde donde pregunta a Agripina si ése es el más bajo del predio. Agripina le responde afirmativamente y premia a Baldomero con un chocolate.

Evidentemente, en su primer intento, Baldomero llegó a un mínimo relativo. Está claro que, dadas suficientes oportunidades, Baldomero siempre podrá encontrar el mínimo absoluto, y esto, sin haber visto nunca el terreno.

Baldomero empleó un método que podríamos llamar de exploración local.

Una técnica muy útil de exploración local en optimización es el método de Box.[3]

Hemos utilizado una terminología propia de diseño pero que difiere de la utilizada por los especialistas en optimización. Así es que, primero que nada, presentamos un pequeño glosario de términos equivalentes.

Diseño		*Optimización*
Función a optimizar	=	Función objetivo
Requisitos de funcionalidad	=	Restricciones de igualdad
Limitaciones	=	Restricciones de desigualdad
Variables independientes	=	Variables de decisión
Limitaciones a las variables independientes	=	Restricciones explícitas
Limitaciones a las variables dependientes	=	Restricciones implícitas

Examinemos ahora el método de Box para dos variables de decisión. Supongamos concretamente que la optimización se refiere a minimización.

Se escogen cuatro puntos que satisfagan todas las restricciones, es decir, que se encuentren dentro de la región factible. Se aísla el punto más alto (el de valor más alto de la función objetivo). Se encuentra el centroide de los tres puntos restantes y se "refleja" el punto aislado a través de dicho centroide.

El punto reflejado substituye ahora al punto inicialmente aislado. De los cuatro puntos que quedan se vuelve a aislar el más alto y se repite todo el proceso anterior. De esta manera el conjunto de puntos o "Simplex" se va desplazando hacia abajo, substituyendo un punto del conjunto con cada desplazamiento. Si un punto reflejado viola una restricción explícita, se regresa a la frontera; es decir, que, a la variable que excedió su limitación, se le reasigna su valor limitativo. Si un punto reflejado viola una restricción implícita, se regresa medio camino.

DISEÑO ÓPTIMO DE FILTROS DIGITALES

Horacio Martínez C.

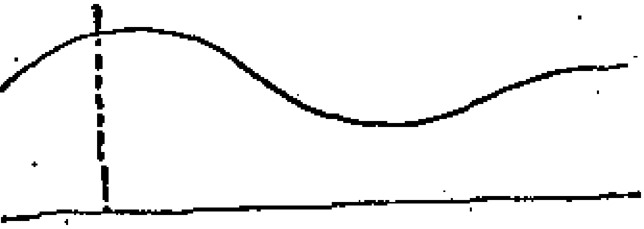
Abril, 82

Procesamiento digital de señales [1, 2]

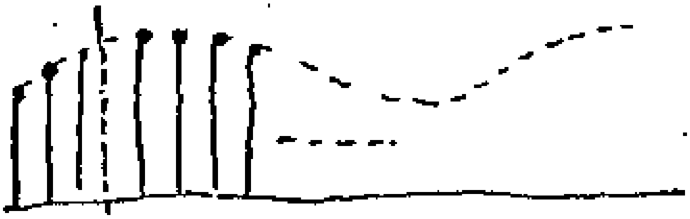
Representación de señales por medio de secuencias de números y el efectuar algunas transformaciones sobre ellas para obtener algún resultado deseado.

Ejs: interpolación
integración
diferenciación
separar distintas bandas de frecuencia
quitar ruido
estimar parámetros
estimación de espectro
identificación de sistemas

Aplicaciones en: Comunicaciónes
señales de voz
señales de audio
bioingeniería
simulación
señales sísmicas
radar
sonar



señal continua: definida para todos los valores de la variable independiente.



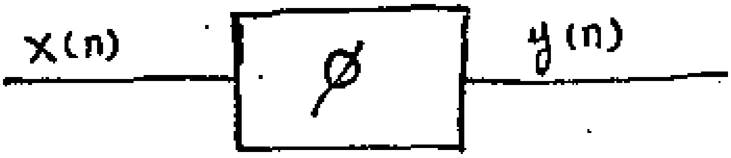
señal discreta: definida para valores discretos de la variable independiente. Generalmente el espaciamiento es uniforme.

Señal digital: tiempo y amplitud discretas

Señal analogica: tiempo y amplitud continuos

Teorema del muestreo: $\frac{1}{\Delta t} > 2f_{max}$

Sistema discreto



$$y(n) = \phi\,[x(n)]$$

Propiedades : Causalidad

Linealidad

Invariancia en el tiempo

Estabilidad.

La salida de un sistema lineal e invariante en el tiempo es

$$y(n) = x(n) * h(n)$$
$$= \sum_{m=0}^{n} x(n)\, h(n-m)$$

$h(\cdot) \triangleq$ respuesta a pulso

Relación entrada salida de los sistemas que vamos a estudiar

$$y(n) = -\sum_{i=1}^{n} b_i\, y(n-i) + \sum_{i=1}^{N} a_i\, x(n-i) \Rightarrow \underline{\text{filtro digital}}$$

Respuesta en frecuencia

$$y(n) = \sum_{m=0}^{n} h(m)\, x(n-m)$$ si la entrada es $x(n) = e^{j\omega n}$

$$y_i(n) = \sum_{m=0}^{n} h(m)\, e^{j\omega(m-n)} = e^{j\omega n} \sum_{m=0}^{n} h(m)\, e^{-j\omega m}$$

$$H(e^{j\omega}) \triangleq \sum_{m=0}^{n} h(m) e^{-j\omega m}$$

respuesta en frecuencia [illegible]

entrada senoidal → salida senoidal misma frecuencia con magnitud $|H(e^{j\omega})|$ y ángulo $\angle H(e^{j\omega})$

$$\text{Retraso de grupo} \triangleq -\frac{d \angle H(e^{j\omega})}{d\omega}$$

Transformada z

$$X(z) = \sum_{n=0}^{\infty} x(n) z^{-n} \qquad x(n) = \frac{1}{2\pi j} \oint X(z) z^{n-1} dz$$

Propiedades:

$$z[x(n)] = X(z) \quad \Rightarrow z[x(n-K)] = z^{-K} X(z)$$

$$z[y(n)] = z[h(n) * x(n)] \Rightarrow Y(z) = H(z) X(z)$$

$H(z)$ función de transferencia

$$y(n) = -\sum_{i=1}^{N} b_i y(n-i) + \sum_{i=1}^{M} a_i x(n-i)$$

$$= \left(-\sum_{i=1}^{N} b_i z^{-i}\right) Y(z) + \left(\sum_{i=1}^{M} a_i z^{-i}\right) X(z)$$

$$\frac{Y(z)}{X(z)} = \frac{\sum_{i=0}^{M} a_i z^{-i}}{\sum_{i=0}^{N} b_i z^{-i}} \quad \begin{matrix} \leftarrow \text{ceros} \\ \leftarrow \text{polos} \end{matrix}$$

Filtros Respuesta a Impulso de duración Finita

(RIF) $$H(z) = \sum_{n=0}^{L} h(n) z^{-n}$$

Filtros Respuesta a Impulso de duración Infinita

(RII) $$H(z) = \frac{\sum_{i=0}^{M} a_i z^{-i}}{\sum_{i=0}^{N} b_i z^{-i}}$$

Problema de aproximación [3,4]

$f(x)$	función aproximada
$F(A,X)$	función aproximante
A	vector de parametros
P	espacio de los parametros A $\quad A \subset P$
ρ	función de distancia cumple con las sigs. propiedades $\rho[g(x)] \geq 0$ y $\rho[g(x)] = 0$ s.y.s $g(x) \equiv 0$ $\rho[c\,g(x)] = \lvert c \rvert \rho[g(x)]$ para cualquier c real $\rho[g(x)+h(x)] \leq \rho[g(x)] + \rho[h(x)]$
X	intervalo de aproximación

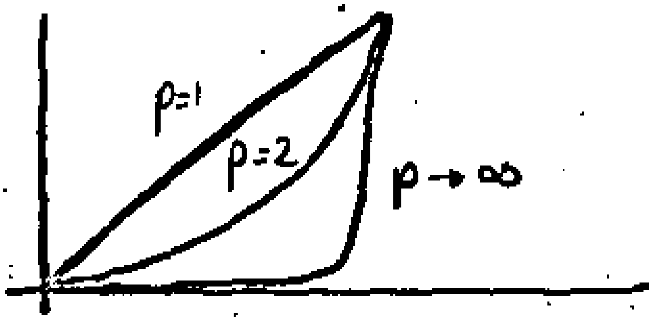
≡ Sea $f(x)$ una función real continua en X, y $F(A,X)$ una función real continua en X que depende de n parámetros A. Dada la función ρ determinar los parámetros $A^* \in P$ tales que

$$\rho[F(A^*,x), f(x)] \leq \rho[F(A,x), f(x)]$$

$\forall\ A \in P$

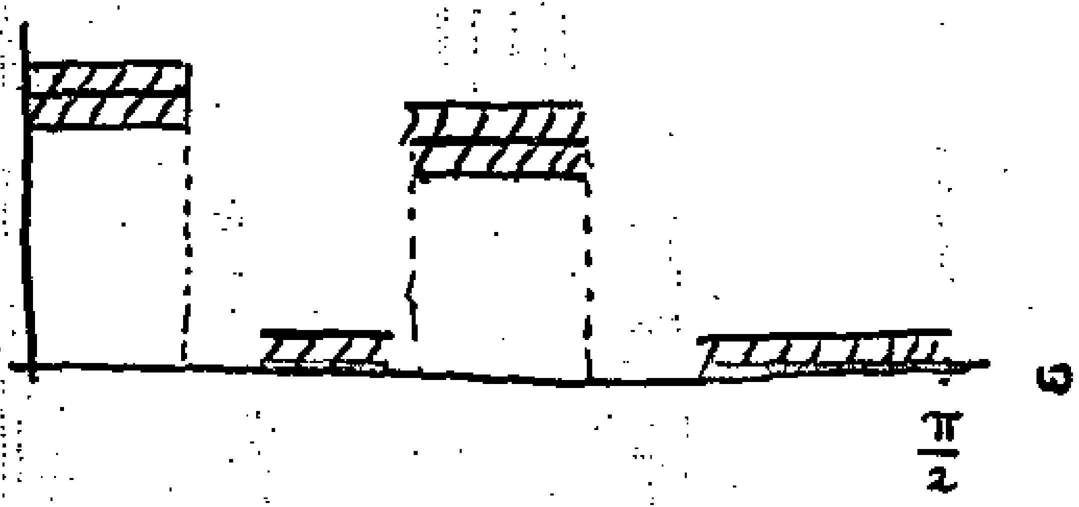
Normas L_p

$$L_p(f) = \left[\int_0^1 |f(x)|^p dx \right]^{1/p} \quad p \geq 1$$



Comportamiento de X^p

Diseño de filtros en el dominio de la frecuencia



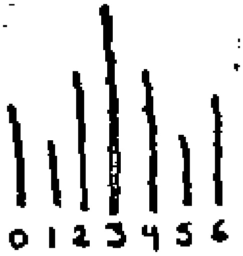
$$L_\infty = \min_{A \in P} \max_{x \in X} \rho \left[F(A,x), f(x) \right]$$

Filtros RIF [5]

$$H(z)=\sum_{n=0}^{L-1} h(n) z^{-n}$$

L impar

$$h(l) = h(L-1-l)$$



$$H(e^{j\omega}) = \sum_{n=0}^{\frac{L-1}{2}} a(n) \cos \omega n$$

Teorema de caracterización:

Si $P(\omega)$ es una función de la forma $\sum_{n=0}^{r} a(n) \cos(\omega n)$ una condición necesaria y suficiente para que $P(\omega)$ sea la mejor aproximación (en la norma ∞) en el intervalo $X \in [0,\pi]$ (X es compacto) a una función $D(\omega)$ continua en X es que el error tenga por lo menos $(r+2)$ valores extremos en X, es decir, debe haber por lo menos $r+2$ puntos ω_i que cumplan con las sigs. condiciones

$0 \leq \omega_1 < \omega_2 \ldots\ldots < \omega_{r+1} < \pi$

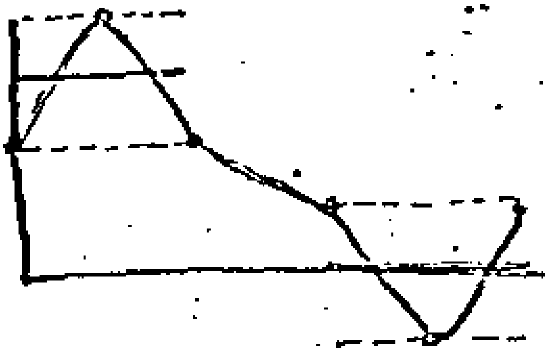
$\omega_i \in X$

y $E(\omega_i) = [D(\omega_i) - p(\omega_i)] = \max_{\omega \in X} |E(\omega)|$

$E(\omega_i) \cdot E(\omega_{i+1}) < 1 \quad i = 0, 1, \ldots r+1$

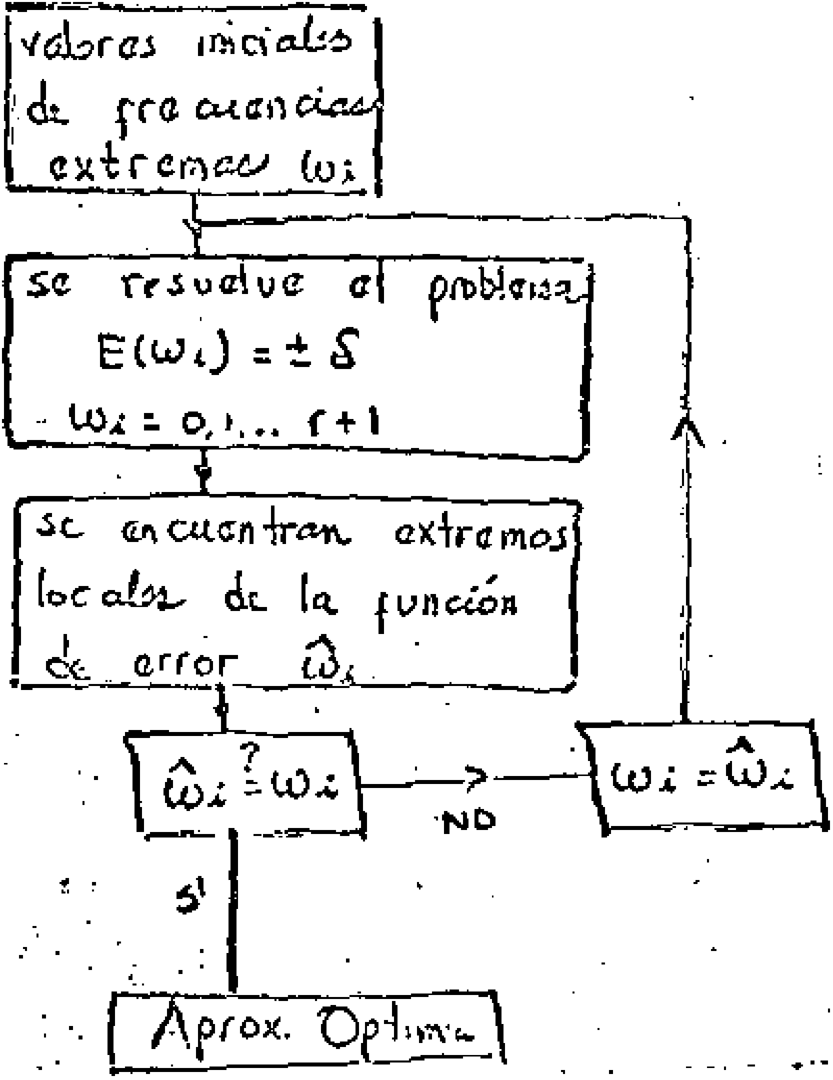
Ejemplos

L = 9



6 extremos

Algoritmo de Remez (2°)


valores iniciales de frecuencias extremas ωi
se resuelve el problema E(ωi) = ± δ ωi = 0, 1 ... r+1
se encuentran extremos locales de la función de error ω̂i
ω̂i ?= ωi
NO
ωi = ω̂i
SI
Aprox. Optima

$$E(\omega_i) = D(\omega_i) - P(\omega_i) = \pm \rho$$

$$\begin{bmatrix} 1 & \cos\omega_0 & \cos 2\omega_0 & \cdots & \cos r\omega_0 & 1 \\ 1 & \cos\omega_1 & \cos 2\omega_1 & \cdots & \cos r\omega_1 & -1 \\ \vdots & & & & & \\ 1 & & & & & \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_r \\ \rho \end{bmatrix} = \begin{bmatrix} D(\omega_0) \\ D(\omega_1) \\ \vdots \\ D(\omega_{r+1}) \end{bmatrix}$$

Filtros RII

$$H(z) = \frac{\sum_{k=0}^{M} \tilde{a}_k z^{-k}}{1 + \sum_{k=1}^{N} \tilde{b}_k z^{-k}} = K \frac{\prod_{i=1}^{M} (z - z_i)}{\prod_{i=1}^{N} (z - p_i)}$$

Magnitud al cuadrado.

$$H(z)H(z^{-1}) = \frac{c(0) + \sum_{k=1}^{M} c(k)(z^{k} + z^{-k})}{d(0) + \sum_{k=1}^{N} d(k)(z^{k} + z^{-k})}$$

$$c(i) = \sum_{k=0}^{M-i} a(k)\, a(k+i)$$

$$d(i) = \sum_{k=0}^{N-i} b(k)\, b(k+i)$$

$$|H(e^{j2\pi f})|^2 = \frac{c(0) + 2\sum_{k=1}^{M} c(k)\cos(2\pi f k)}{d(0) + 2\sum_{k=1}^{N} d(k)\cos(2\pi f k)}$$

$$= \frac{\hat{c}(0) + \sum_{k=1}^{M} \hat{c}(k)\cos^k(2\pi f)}{1 + \sum_{k=1}^{N} \hat{a}(k)\cos^k(2\pi f)}$$

Minimización d normas L_p [6]

la norma L_p $\left\{\int_X |F(A,\omega) - f(\omega)|^p \, d\omega\right\}^{1/p}$

se aproxima por

$$L_{2p} = \sum_{k=1}^{K} |(F(A,\omega_k) - f(\omega_k)|^{2p}$$

Algoritmo de Polya

$$\lim_{p\to\infty} A_p^* = A^*$$

donde $F(A^*, X)$ es la solución óptima en la norma de Chebychou.

Forma de la función de transferencia

$$H(z) = k_0 \prod_{i=1}^{N} \frac{z^2 + a_{1i} z + a_{2i}}{z^2 + b_{1i} z + b_{2i}}$$

evaluando en el círculo unitario

$$\alpha(A\omega) = H(e^{j\omega}) = k_0 \prod_{i=1}^{N} \frac{1+a_{2i})\cos\omega + a_{1i} + j(1-a_{2i})\operatorname{sen}\omega}{(1+b_{2i})\cos\omega + b_{1i} + j(1-b_{2i})\operatorname{sen}\omega}$$

$$\tau(A,\omega) = \sum_{i=1}^{N} \Big[\operatorname{Re}\left(\frac{2\cos\omega + b_{1i} + j2\operatorname{sen}\omega}{(1+b_{2i})\cos\omega + b_{1i} + j(1-b_{2i})\operatorname{sen}\omega} \right)$$

$$- \operatorname{Re}\left(\frac{2\cos\omega + a_{1i} + j2\operatorname{sen}\omega}{(1+a_{2i})\cos\omega + a_{1i} + j(1-a_{2i})\operatorname{sen}\omega} \right)$$

más conveniente usar coordenadas polares

Se puede aproximar simultaneamente magnitud y fase

p. ej. $$L_{2p,2q}^{\alpha,\tau}(A,\tau_0) = \delta \sum_{k=1}^{K} \left(\alpha(A,\omega_k) - \alpha_d(\omega_k)\right)^{2p}$$

$$+ (1-\delta) \sum_{r=1}^{S} \left(\tau(A,\omega_r) - \tau_d(\omega_r) - \tau_0\right)^{2q}$$

Uso de programación lineal [7]

Se trabaja con la magnitud al cuadrado

$$\left[c_0 + \sum_{i=1}^{n} 2c_i \cos(\omega i)\right] \Big/ \left[d_0 + \sum_{i=1}^{n} 2d_i \cos(\omega i)\right]$$

$$= \hat{N}(\omega) / \hat{D}(\omega)$$

$F(\omega)$ función deseada

$$-\epsilon(\omega) \le \frac{\hat{N}(\omega)}{\hat{D}(\omega)} - F(\omega) \le \epsilon(\omega)$$

$\epsilon(\omega)$ función de tolerancia

$$\hat{N}(\omega) - \hat{D}(\omega) F(\omega) \le \epsilon(\omega) \hat{D}(\omega)$$

$$-\hat{N}(\omega) + \hat{D}(\omega) F(\omega) \le \epsilon(\omega) \hat{D}(\omega)$$

$$\Downarrow$$

$$\hat{N}(\omega) - \hat{D}(\omega) [F(\omega) + \epsilon(\omega)] \le 0$$

$$-\hat{N}(\omega) + \hat{D}(\omega) [F(\omega) - \epsilon(\omega)] \le 0$$

$$-\hat{N}(\omega) \le 0$$

$$-\hat{D}(\omega) \le 0$$

Se resta la variable auxiliar ν de las ecs. anteriores y se minimiza

$$Z = \nu$$

Solución si $\nu = 0$

si $\nu > 0$ hay que modificar $E(\omega)$ o $F(\omega)$

Algoritmo de Remez para el caso racional

$$\sum_{i=0}^{m} c_i \cos \omega i \Big/ \sum_{i=0}^{n} b_i \cos \omega i - f(\omega) = \pm \delta$$

$$\sum_{i=0}^{m} c_i \cos \omega i - f(\omega) \left[b_0 + \sum_{i=1}^{n} b_i \cos \omega i \right] =$$

$$(-1)^i \delta \left[b_0 + \sum_{i=1}^{n} b_i \cos \omega i \right]$$

Teorema de caracterización [4]

Sea $R = P/Q$ donde P es un polinomio de grado m, Q es un polinomio de grado n, P/Q es irreducible y $Q > 0$ en X. R es una mejor aproximación a $f \in C[X]$ si la función de error tiene al menos $m+n+2$ extremos en X

Solución iterando entre diseño de numerador y denominador [8]

Referencias:

[1] L. Rabiner + B. Gold, Theory and applications of digital signal processing, Prentice-Hall, 75

[2] A. Oppenheim + R Schafer, Digital Signal Processing, Prentice-Hall, 75

[3] J. R. Rice, The approximation of fonctions, Addison Wesley, 1964

[4] E.W. Cheney, Introduction to approximation theory Mc. Graw-Hill, 1966

[5] L. R. Rabiner, J.H. Mc Clellan + T.W. Parks, FIR digitall filter design techniques using weighted Chebyshev approximation, Proc IEEE, Abril 1975.

[6] A.G. Deczky, Synthesis of recursive Digital Filters Using the minimum p- Error Criterion, IEEE Trans on Audio Electroacoust, vol AU-20, Oct. 72

[7] L.R. Rabiner, N.Y. Graham + H.O. Helms, Linear programming design of IIR Digital Filters with arbitrary magnitude function, IEEE Trans. on ASSP, Abril 74

[8] H. G. Martinez + T.W. Parks, "Design of recursive filters with optimum magnitude and attenuation poles on the unit circle", IEEE Trans. on ASSP, Abril 78

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

CONDICIONES DE OPTIMALIDAD

Dr. Antonio Montalvo

MARZO, 1982

CONDICIONES DE OPTIMALIDAD

I. EL PROBLEMA GENERAL DE PROGRAMACION NO LINEAL

En el sentido más amplio, el problema general no lineal es el de encontrar un extremo (máximo o mínimo) de una función objetivo sujeta a restricciones de igualdad y/o desigualdad, las cuales pueden ser lineales y/o no lineales. Sin embargo, en las siguientes secciones de estas notas han quedado excluidos los dos siguientes problemas:

a) Las variables estan restringidas a valores enteros (programación no lineal entera)

b) Las restricciones incluyen al parámetro tiempo en la forma de una ecuación diferencial (control óptimo).

En lo siguiente se supondrá que la función objetivo $f(x)$ es continua, $h_1(x), \ldots, \overset{h_m}{hm}(x)$ denotan las restricciones de igualdad y $g_{m+1}(x), \ldots, g_p(x)$ las restricciones de desigualdad, donde: $x = (x_1, \ldots, x_n)^T$ es un vector columna de componentes $x_1, \ldots, x_n$ en un espacio euclidiano n-dimensional. (Las variables $x_1, x_2, \ldots, x_n$ pueden ser parámetros de diseño, ajuste de controles, lecturas de

instrumentos, etc., mientras que la función objetivo podría representar el costo, peso, ganancias, etc.; finalmente, las restricciones pueden representar requerimientos técnicos, condiciones de operación, etc., del proceso).

El problema de programación no lineal se puede establecer formalmente como:

$$\text{Minimizar: } \quad f(x), \; x \in E^n \tag{1.1}$$

sujeta a m restricciones de igualdad, lineales $^y/_o$ no lineales,

$$h_j(x) = 0 \qquad j = 1, \ldots, m \tag{1.2}$$

y (p - m) restricciones de desigualdad, lineales $^y/_o$ no lineales,

$$g_j(x) \geq 0 \qquad j = m+1, \ldots, p \tag{1.3}$$

O en forma alterna como:

$$\text{Minimizar: } \{f(x) | x \in R\} \tag{1.4}$$

donde R es el dominio de x para el cual (1.2) y (1.3), se satisfacen, es decir:

$$R = \{x | h_j(x) = 0, \; g_j(x) \geq 0, \text{ para toda } j\} \tag{1.5}$$

Un ejemplo sencillo de programación no lineal es el que se muestra en la figura 1, y está dado por:

Minimice: $f(x) = x_1^{2} + x_2^{2} + 2\overset{x_2}{x_2}$

sujeto a: $h_1(x) = x_1^{2} + x_2^{2} - 1 = 0$

$g_2 = x_1 + \underset{x_2}{\cancel{x_2}} - \frac{1}{2} \geq 0$

$g_3 = x_1 \geq 0$

$g_4 = x_2 \geq 0$

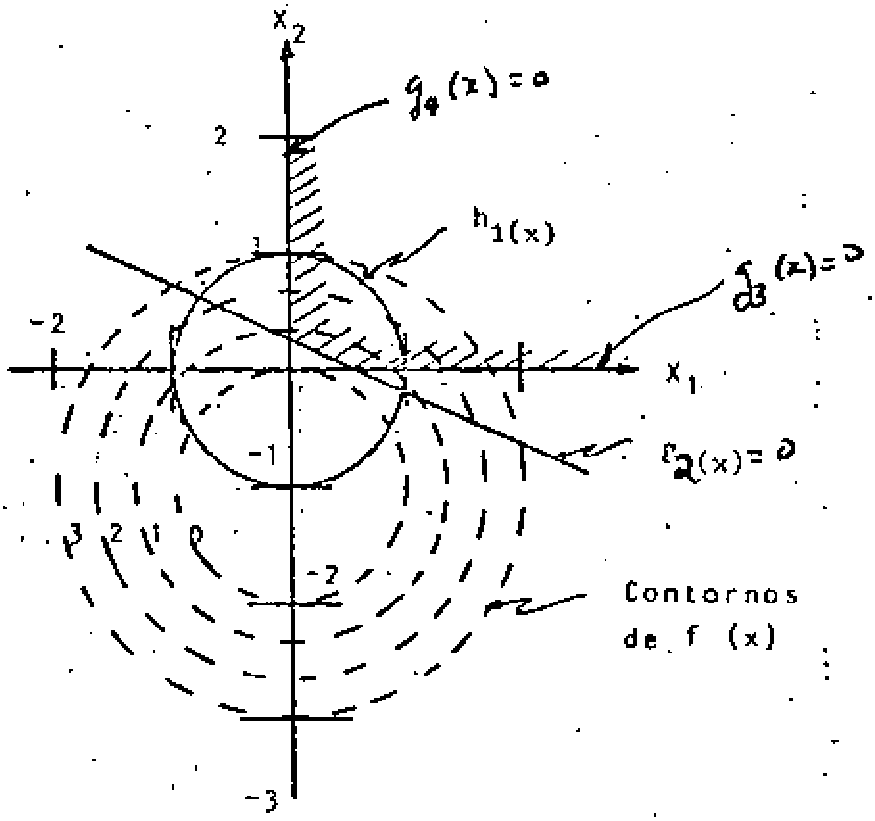


FIGURA 1. Representación geométrica de un problema de programación no lineal

II. NOTACION Y TERMINOLOGIA

El vector columna $x^* = (x_1^*, \ldots, x_n^*)^T$ que satisface (1.1) - (1.3) se denomina punto óptimo, y el valor de $f(x^*)$ que le corresponde se denomina valor óptimo de la función objetivo. La pareja, x^*, $f(x^*)$ constituye la solución óptima. Para algunos problemas, pueden existir varias categorias de soluciones óptimas si la función objetivo no es unimodal (exhibe mas de un punto extremo) como se ilustra en la figura 2. La solución global óptima representa el valor más pequeño de $f(x)$, mientras que una solución óptima local (o relativa) representa el valor más pequeño de $f(x)$ en una cierta vecindad del vector x: es decir,

óptimo global x^* satisface $f(x^*) < f(x) \; \forall x \in R$

óptimo local x^* satisface $f(x^*) < f(x), \; ||x-x^*|| \leq \delta(x^*)$

2.1. Concavidad y Convexidad

Los conceptos de concavidad y convexidad ayudan a determinar bajo que condiciones una solución óptima local es tambien solución óptima global.

Una función $\phi(x)$ se dice que es convexa en el dominio R, si para cualesquiera dos vectores x_1 y x_2 $\in R$,

$$\phi\left(\theta x_1 + (1-\theta)x_2\right) \le \theta\phi(x_1) + \phi(x_2)(1-\theta) \qquad (1.6)$$

donde θ es un escalar $0 \le \theta \le 1$. Además, $\phi(x)$ es <u>estrictamente</u> convexa si, para $x_1 \neq x_2$, el signo $\le$ en (1.6) se puede remplazar por el signo de desigualdad $(<)$. Si en (1.6) la desigualdad contraria es la válida, se dice que la función $\phi(x)$ es cóncava $(\ge)$ o <u>estrictamente</u> cóncava $(>)$. Note que si $\phi(x)$ es cóncava (cónvexa), $-\phi(x)$ es cónvexa (cóncava). (Las funciones lineales son, simultáneamente, convexas y cóncavas).

Una función convexa diferenciable posee las siguientes propiedades.

a) $\phi(x_2) - \phi(x_1) \ge \nabla^T\phi(x_1)(x_2-x_1)$ para toda x_1 y x_2.

b) La matriz de segundas derivadas parciales de $\phi(x)$ con respecto a x (matriz Hessiana) es positiva definida (o positiva semidefinida) para toda x si $\phi(x)$ es estrictamente convexa (o convexa).

c) Sobre el dominio de R; $\phi(x)$ posee un sólo mínimo.

(1.7)

Un conjunto de puntos (o región) se define como <u>conjunto</u> convexo en un espacio n-dimensional si, para toda pareja de puntos x_1 y x_2 en el conjunto, la linea recta que los une pertenece completamente al conjunto. Es decir, R es convexo si para toda x_1 y x_2 ϵ R

$x = \theta x_1 + (1-\theta)x_2 \in R$

De los conceptos de convexidad emerge un resultado importante en programación matemática: para el problema de programación no lineal conocido como el problema de programación convexa

$$\text{Minimizar}: \quad f(x)$$
$$\text{sujeta a}: \quad g_j(x) \geq 0 \qquad j = 1, \ldots, p$$
$$x \geq 0$$

en el cual (a), f (x) es una función convexa y (b) cada restricción de desigualdad es una función cóncava (las restricciones forman un conjunto convexo), se puede demostrar el siguiente resultado: el mínimo local también es mínimo global (Usando argumentos opuestos, el resultado opuesto, máximo, también es cierto).

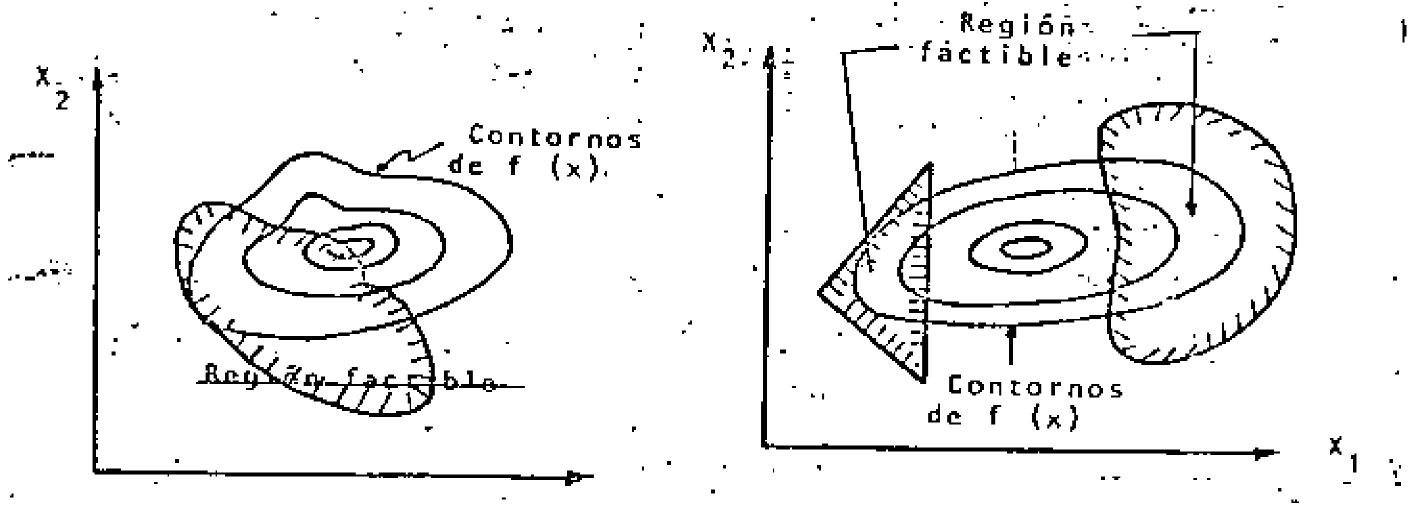
2.2. Factibilidad

Cualquier vector x que satisface tanto las restricciones de desigualdad como las de igualdad se llama punto factible. El conjunto de todos los puntos que satisfacen las restricciones constituyen el dominio factible de f (x), y se denotará por R; cualquier punto no en R se llama punto no factible.

Un óptimo restringido es uno para el cual el óptimo local cae en la frontera de la región factible. Si las restricciones son unicamente de igualdad, un punto x factible debe caer en la intersección de todas las hipersuperficies que satisfacen $h_j(x) = 0$

Con respecto a las restricciones de desigualdad, un punto x se puede clasificar como punto interior (factible), punto frontera (factible) o punto exterior (no factible). Los puntos interiores son aquellos para los cuales $\bar{g}_j(x) > 0$; para un punto frontera, g_j $g_j(x) = 0$ para al menos una restricción; y un punto exterior, $g_j(x) < 0$ para al menos una restricción. Las restricciones se llaman activas (o de atadura) si $g_j(x) = 0$.

Una región R de vectores admisibles puede ser convexa o no convexa, según se describió con anterioridad, pero además puede ser simplemente conexa ó no-simplemente conexa (ver Figura 2).



(a) simplemente conexa
(no convexa)

(b) no simplemente conexa
(obviamente no convexa)

FIGURA 2. Ejemplos de tipos de región

2.3. El Gradiente

El conjunto de puntos para los cuales una función $f(x)$ exhibe un valor constante, se llaman contornos de $f(x)$. Si la función $f(x)$ es continua y diferenciable, el gradiente de la función existe y está definido como el vector columna formado por las primeras derivadas parciales de $f(x)$ con respecto a x es decir:

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} \qquad (1.8)$$

Se puede demostrar que en el espacio métrico euclidiano, el gradiente de una función escalar apunta en la dirección de máximo incremento en el valor de la función, máximo ascenso, y que es, además, ortogonal a las líneas de contorno. El negativo del gradiente apunta en la dirección del máximo descenso de $f(x)$. Finalmente, cualquier vector V, ortogonal a $\nabla f(x)$, tal como la superficie tangente a $f(x)$, está definido por

$V^T \nabla f(x) = 0$ (Figura 3)

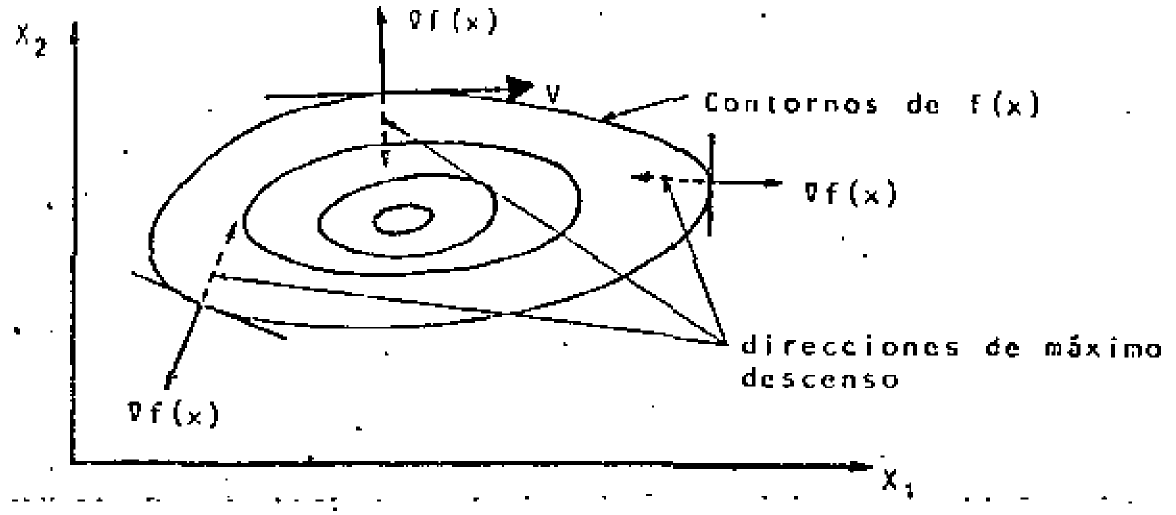


FIGURA 3. El gradiente, y la dirección de máximo descenso.

2.4. Aproximación de funciones

Algunos de los procedimientos de programación matemática que se discutirán más tarde requieren de aproximaciones lineales o cuadráticas para $f(x)$, $g(x)$ y $h(x)$.

Una aproximación lineal, o de primer orden, para una función $f(x)$, se puede hacer truncando la serie de Taylor, alrededor de un punto x_0, como

$$f(x) \approx f(x_0) + \nabla^T f(x_0)(x-x_0) \tag{1.9}$$

Para obtener una aproximación cuadrática, en la misma serie de Taylor se pueden despreciar los términos mayores e iguales a tercer orden, obteniéndose

$$f(x) = f(\bar{x}_0) + \nabla^T f(x_0)(x-x_0) + \frac{1}{2}(x-x_0)^T \nabla^2 f(x_0)(x-x_0) \qquad (1.10)$$

donde $\nabla^2 f(x)$ es la <u>matriz Hessiana</u> ~~x~~ de $f(x)$, $H(x)$, es decir

$$H(x_0) = \{h_{ij}(x_0)\} \quad i,j = 1, \ldots, n \qquad (1.11a)$$

donde

$$h_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j} \qquad (1.11b)$$

Se observa facilmente que $H(x)$ es una matriz simétrica.

2.5. Condiciones Necesarias y Suficientes para que Una Solución sea Solución Optima

Para algunas clases especiales del problema general de programación no lineal [Ecs (1.1) - (1.3)] ha sido posible establecer criterios de optimalidad. Sin embargo, para funciones completamente generales, no ha sido posible establecer criterios de optimalidad precisos. En consecuencia, unicamente se describirán algunos casos especiales, los cuales, sin embargo, son bastante comunes y de importancia práctica. Las condiciones que determinan si un vector x resuelve o no el problema de programación no lineal serán presentadas en una serie de teoremas los cuales no serán demostrados (las demostraciones están fuera de los objetivos de este curso).

2.5.1. Programación no-lineal sin restricciones

El problema es el siguiente:

$$\text{Minimizar:} \quad f(x),\ x \in E^n \qquad (1.12)$$

Las condiciones necesarias para que x^* sea un mínimo local del problema (1.12) son:

1. $f(x)$ diferenciable en x^*.

2. $\nabla f(x^*) = 0$, es decir, existe un punto estacionario de $f(x)$ en x^*

Las condiciones suficientes para que x^* sea un mínimo local del problema (1.12) son:

3. $\nabla^2 f(x) > 0$; es decir, la matriz Hessiana es positiva definida.

(Las condiciones para la existencia de un máximo son iguales, excepto que el hessiano deberá ser negativo definido).

2.5.2. Programación no-lineal con restricciones de Igualdad y Desigualdad

En este caso el problema es el siguiente:

$$\left.\begin{array}{llll} \text{Minimizar:} & f(x) & x \in E^n & \\ \text{Sujeta a:} & h_j(x) = 0 & j = 1, \ldots, m & \\ & g_j(x) \geq 0 & j = m+1, \ldots, p & \end{array}\right) \quad (1.13)$$

Las condiciones necesarias para que x^* sea un mínimo local se establecen en dos teoremas, el primero de los cuales (teorema 2) puede ser llamado condiciones de primer orden (debido a que las funciones que intervienen se consideran una vez diferenciables). El segundo teorema (teorema 3) se le denomina condiciones de segundo orden (se considera que las funciones son dos veces diferenciables).

Para establecer las condiciones necesarias, empezaremos co el siguiente concepto: si x^* es un mínimo local de $f(x^*)$, ésta no puede decrecer a lo largo de ningun arco "suave" dirigido desde x^* hacia la región factible. Sea el vector V tangente al arco que empieza en x^*. Usando los conceptos de Fiacco y Mc Cormick, se asignan tres diferentes categorias o clases al vector V, donde cada conjunto V_i incluye el conjunto de tales que: (ver tabla I).

Todas las posibles perturbaciones de x^* caen en la unión de V_1 y V_2 y si $V \in V_2$, f (x) decrece, mientras que si $V \overset{\in}{\notin} V_1$, f (x) se incrementa o es constante. En esencia, las condiciones necesarias de primer orden imponen el requerimiento de que el conjunto V_2 esté vacio.

	RESTRICCIONES				
CLASE	DESIGUALDAD		IGUALDAD		FUNCION OBJETIVO
V_1	$\left\{ v^T \nabla g_j (x^*) \geq 0 \right.$ para las restricciones activas $\left. \right\}$	y	$\left\{ v^T \nabla h_j (x^*) = 0 \right.$ $\left. j = 1,2, \ldots, m \right\}$	y	$\left\{ v^T \nabla f (x^*) \geq 0 \right\}$
V_2	$\left\{ v^T \nabla g_j (x^*) \geq 0 \right.$ para las restricciones activas $\left. \right\}$	y	$\left\{ v^T \nabla h_j (x^*) = 0 \right.$ $\left. j = 1,2, \ldots, m \right\}$	y	$\left\{ v^T \nabla f (x^*) < 0 \right\}$
V_3	$\left\{ v^T \nabla g_j (x^*) < 0 \right.$ para, al menos, una restricción activa $\left. \right\}$	o	$\left\{ v^T \nabla h_j (x^*) \neq 0 \right.$ para, al menos, una restricción $\left. \right\}$		

TABLA I. Clasificación de los conjuntos V_i

Si V_2 está vacio, se puede demostrar la existencia de los multiplicadores de Lagrange, resultando el siguiente teorema.

TEOREMA I.

Si (a) x^* satisface el problema (1.13), (b) las funciones $f(x)$, $g_j(x)$ son una vez diferenciables, y (c) en x^* V_2 está vacio, entonces existen los vectores u^* y w^* (multiplicadores de Lagrange) tales que, (u^*, w^*, x^*) satisfacen

(1) $h_j(x^*) = 0$ $\qquad j = 1, \ldots, m$

(2) $g_j(x^*) \geq 0$ $\qquad j = m + 1, \ldots, p$

(3) $u_j^* g_j(x) = 0$ $\qquad j = m + 1, \ldots, p$

(4) $u_j^* \geq 0$ $\qquad j = m + 1, \ldots, p$

(5) $\nabla L(x^*, u^*, w^*) = 0$

donde la función

$$L(x, u, w) = f(x) + \sum_{j-1}^{m} w_j h_j(x) - \sum_{j=m+1}^{p} u_j g_j(x)$$

puede ser considerada como una función Lagrangiana generalizada, asociada al problema (1.13).

Con el propósito de establecer bajo que circunstancias el conjunto V_2 está vacio, se necesita calificar, a primer orden, a las restricciones.

Sea x^* un punto factible del problema (1.13) y supóngase que $h_1(x), \ldots,$ ~~h_n~~ h_m (x), $g_{m+1}(x), \ldots, g_p(x)$ son funciones una vez diferenciables. La calificación a primer orden de las restricciones es una condición que se impone unicamente sobre las restricciones (sin importar la función objetivo) y que consiste en que para cada punto frontera formado por el conjunto de restricciones de igualdad y las activas de desigualdad, debe de existir una curva suave que termine en el punto frontera y que pertenezca completamente al conjunto de las restricciones. Si x^* es un mínimo local de $f(x)$, ésta no puede decrecer a lo largo de tal curva, dirigida desde x^* hacia la región factible. Una condición suficiente para la calificación a primer orden de las restricciones que debe cumplirse es que todos los gradientes de las restricciones de desigualdad activas y los gradientes de las restricciones de igualdad evaluados en x^*, sean linealmente independientes. Esto último se establece en el siguiente teorema.

TEOREMA 2.

Si las funciones $h_1(x), \ldots,$ ~~h_n~~ h_m (x), $g_{m+1}(x), \ldots, g_p(x)$ son una vez diferenciables en x^*, y si la calificación a primer orden de las restricciones es válida en x^*, entonces la condición necesaria para que x^* sea un mínimo local del problema (1.7), es que existan multiplicadores de Lagrange u^* y w^* tales que (u^*, w^*, x^*) satisfagan las ecuaciones (1) - (5) del Teorema 1.

Para tomar en cuenta la curvatura de las funciones en el problema (1.13), Mc Cormick estableció las condiciones necesarias de segundo orden para que x^* sea un mínimo local. Supongase que las funciones $f(x)$, $h_1(x), \ldots, h_m(x)$, $g_{m+1}(x), \ldots, g_p(x)$ son dos veces diferenciables en x^*, un punto que satisface al problema (1.13). Sea V cualquier vector no cero tal que

$V^T \nabla g_j(x) = 0$ para las restricciones de desigualdad activas.

$V^T \nabla h_j(x) = 0$. para las restricciones de igualdad

Entonces, si V es la tangente a una curva $\psi(\theta)$, $\theta \geq 0$, dos veces diferenciable, a lo largo de la cual $g_j(\psi(\theta)) = 0$ para todas las restricciones de desigualdad activas, y $h_j(\psi(\theta)) = 0$ para todas las restricciones de igualdad, la calificación a segundo orden de las restricciones en x^* es válida. Una condición suficiente para la calificación a segundo orden de las restricciones que debe cumplirse es que los gradientes de las restricciones de desigualdad activas en x^* y los gradientes de las restricciones de igualdad en x^* sean linealmente independientes.

Las condiciones necesarias de segundo orden se pueden establecer como sigue.

TEOREMA 3.

(a) Si las funciones $f(x)$, $h_1(x)$, ..., $h_m(x)$, $g_{m+1}(x)$, ..., $g_p(x)$ son dos veces diferenciables en x^*, y (b) si la calificación a primer orden de las restricciones es válida en x^*, entonces las condiciones necesarias para que x^* sea un mínimo local del problema (1.13) son que existan u^* y w^* tales que (c) ecuaciones (1) - (5) del Teorema 1 se satisfagan, y (d) para cada vector no cero V, para el cual $V^T \nabla g_j(x^*) = 0$, para las restricciones de desigualdad activas, y $V^T \nabla h_j(x^*) = 0$, para las restricciones de igualdad, se cumple lo siguiente:

$$(6) \quad V^T \nabla^2 L \, (x^*,\ u^*;\ w^*) \ V \geq 0$$

Las condiciones suficientes para que x^* sea un mínimo local aislado del problema (1.13) son las mismas (a), (b) y (c) del Teorema 3, excepto la parte (d) (ecuación (6)), la cual debe ser substituida por:

(d') Para cada vector V no cero para el cual $V^T \nabla g_j(x^*) = 0$ para las restricciones de desigualdad activas, $V^T \nabla g_j(x^*) \geq 0$ para las restricciones de desigualdad no activas y $V^T \nabla h_j(x^*) = 0$ para las restricciones de igualdad, lo siguiente es verdadero:

$$(6') \quad V^T \nabla^2 L \, (x^*,\ u^*,\ v^*) \ V > 0$$

Ejemplo 1. Condiciones necesarias y suficientes con restricciones de desigualdad.

Minimizar : $f(x) = x_1^2 + x_2$

Sujeta a : $g_1(x) = \overset{\downarrow}{\boxed{-}}(x_1^2 + x_2^2) + \overset{9}{\cancel{}} \geq 0$

$g_2(x) = -x_1 - x_2 + \overset{1}{\cancel{}} \geq 0$

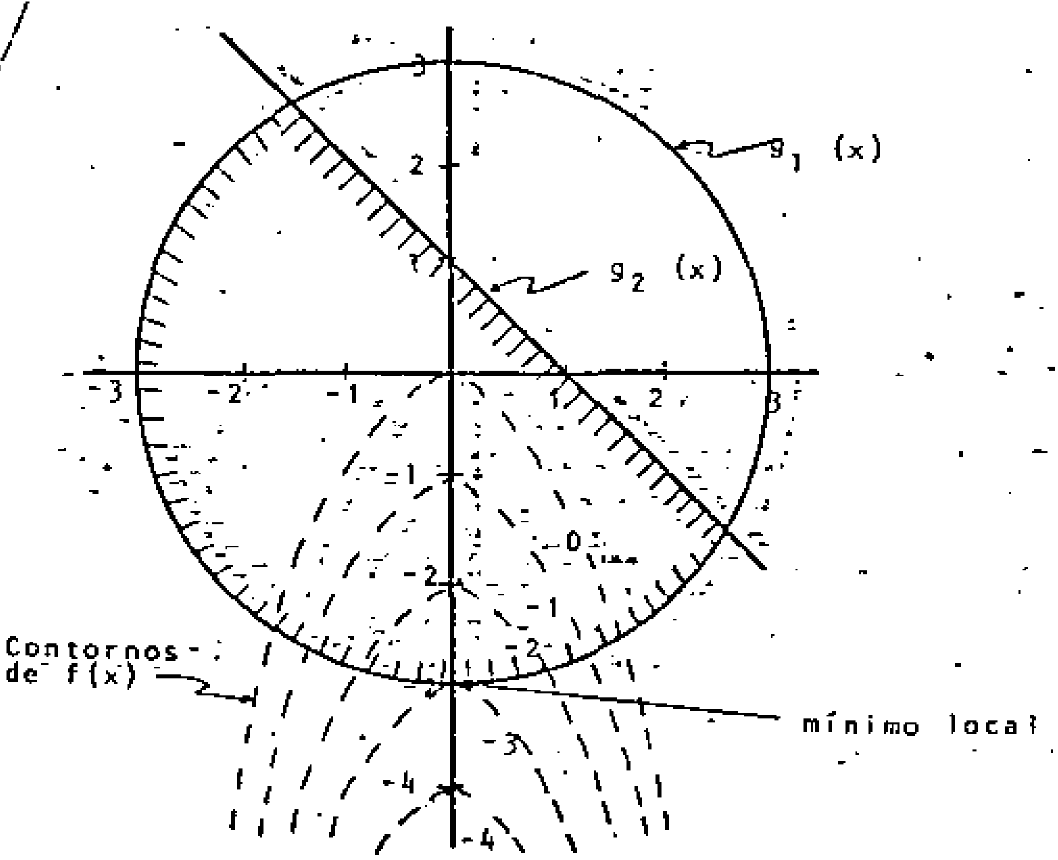


FIGURA 4.- Región admisible y curvas de contorno del Ejemplo 1. Se observa que $g_1(x)$ es una restricción activa mientras que $g_2(x)$ no lo es.

Ya que sólo una de las restricciones está activa, no se necesita comprobar la calificación a primer - y segundo orden de las restricciones (Note que $f(x)$, $g_1(x)$ y $g_2(x)$ son dos veces diferenciables).

De acuerdo a los Teoremas (1) y (2) se necesita demostrar que existen u^* y x^* tales que

(2) $g_j(x^*) \geq 0$ $\qquad$ $-x_1^{*2} - \overset{4}{x_2^{*2}} + \overset{9}{1} \geq 0$

$-x_1^* - x_1^* + 1 \geq 0$

(3) $u_j^* \, g_j(x^*) = 0$ $\qquad$ $u_1^* \, (-x_1^{*2} - x_2^{*2} - \overset{9}{1}) = 0$

$u_2^* \, (-x_1^* - x_2^* + 1) = 0$

(4) $u_j^* \geq 0$ $\qquad$ $u_1^* \geq 0$

$u_2^* \geq 0$

(5) $\nabla L(x^*, u^*) = 0$ $\qquad$ $(L = f(x) - u_1 \, g_1(x) - u_2 \, g_2(x))$

$$\begin{pmatrix} 2x_1^* \\ 1 \end{pmatrix} - u_1^* \begin{pmatrix} -2x_1^* \\ -2x_2^* \end{pmatrix} - u_2^* \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Resolviendo las ecuaciones anteriores se puede verificar que :

$$x^* = (0, \ -3)^T$$

y

$$u^* = \left(\frac{1}{6}, \ 0\right)^T$$

La condición de segundo orden que debe ser satisfecha es :

$$V^T \ \nabla g_j(x^*) = 0 \qquad g_j(x^*) \ : \ \text{restricción activa}$$

es decir:

$$(V_1, \ V_2) \begin{pmatrix} -2x_1^* \\ -2x_2^* \end{pmatrix} = V_1(0) + V_2(6) = 0$$

de donde V_1 puede tomar cualquier valor, y $V_2 = 0$, substituyendo V en (6), se tiene

(6) $$V^T \ \nabla^2 L \ (x^*, \ u^*) \ V \geq 0$$

donde $$\nabla^2 L = \begin{bmatrix} 2(1 + u_1) & 0 \\ 0 & 2u_2 \end{bmatrix}$$

Ejemplo 2. Condiciones necesarias y suficientes con restricciones de Igualdad y de Desigualdad

Minimi~~z~~ce : $f(x) = x_1^2 + x_2$

Sujeta a : $h_1(x) = x_1^2 + x_2^2 - \overset{1}{\cancel{9}} = 0$

$g_2(x) = -(x_1 + x_2^2) + 1 \geq 0$

$g_3(x) = -(x_1 + x_2) + 1 \geq 0$

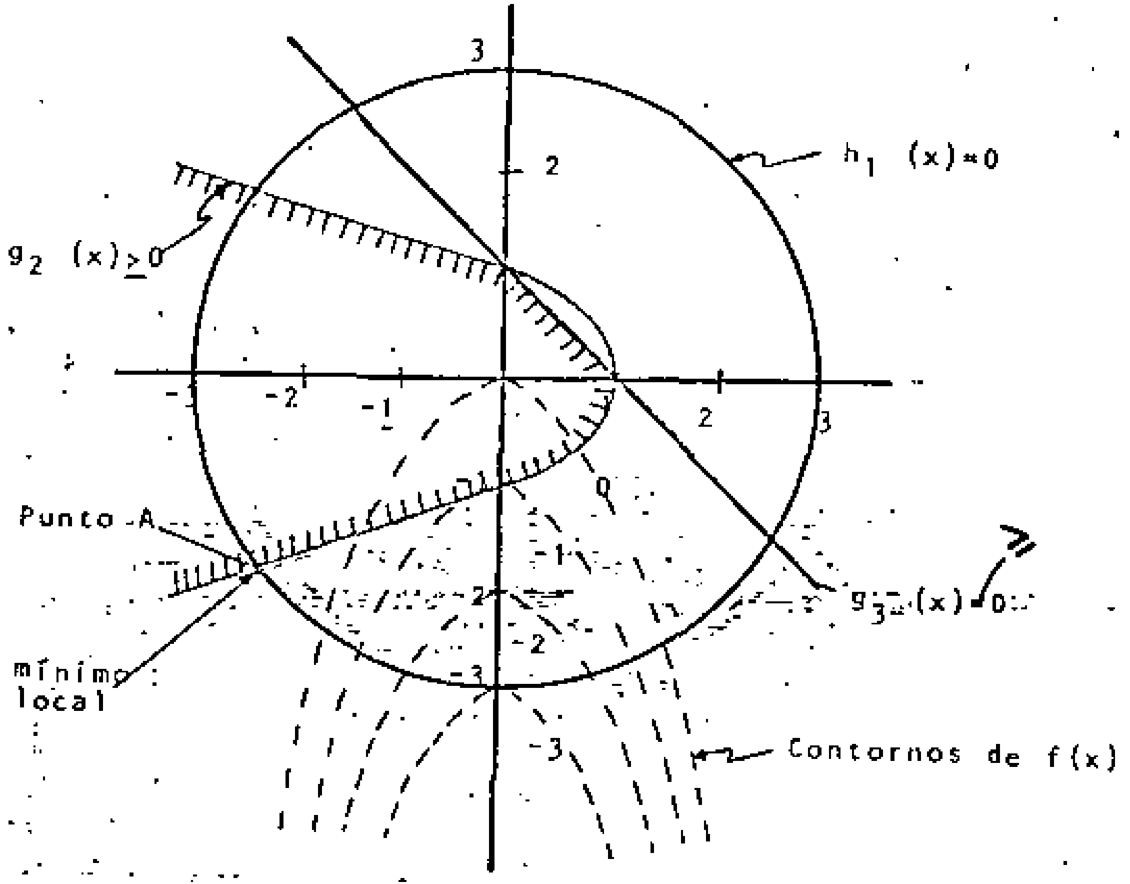


FIGURA 5. Región admisible y curvas de nivel del Ejemplo 2.

De acuerdo al Teorema 3, $h_1(x)$, $g_2(x)$ y $g_3(x)$ deben ser dos veces diferenciables. Además, los gradientes de las restricciones activas deberán ser linealmente independientes para que satisfagan la calificación a primer- y segundo orden. Suponga que A, localizado en $x^* = (-2.37, -1.84)^T$ en la intersección de $h_1(x^*) = g_2(x^*) = 0$ sea un candidato a mínimo local. Si se forma una combinación lineal entre los gradientes de $h_1(x)$ y $g_2(x)$, en x^*, se tiene que, para que sean linealmente independientes

$$C_1 \nabla h_1 (x^*) + C_2 \nabla g_2 (x^*) = 0 \Rightarrow \cancel{a_1 = a_2 = 0}$$

$$\hookrightarrow C_1 = C_2 = 0$$

o

$$C_1 \begin{pmatrix} 2x_1^* \\ 2x_2^* \end{pmatrix} + C_2 \begin{pmatrix} -1 \\ -2x_2^* \end{pmatrix} = 0$$

y como

$$\det \begin{vmatrix} 2x_1^* & -1 \\ 2x_2^* & -2x_2^* \end{vmatrix} \neq 0 \Rightarrow C_1 = C_2 = 0$$

es decir, los gradientes de las restricciones activas son linealmente independientes.

De acuerdo a los Teoremas (1) y (2) se debe cumplir que.

(1) $h_1 (x^*) = 0$ $\qquad$ $x_1^{*2} + x_2^{*2} = 0$.

(2) $g_j (x^*) = 0$ $\qquad$ $-(x_1^* + x_2^{*2}) + 9 \geq 0$

$\qquad$ $-(x_1^* + x_2^*) + 1 \geq 0$

(3) $u_j^* \; g_j (x^*) = 0$ $\qquad$ $u_2^* \left[-(x_1^{*2} + x_2^{*2}) + 9 \right] = 0$

$\qquad$ $u_3^* \left[-(x_1^* + x_2^*) + 1 \right] = 0$

(4) $u_j^* \geq 0$ $\qquad$ $u_2^* \geq 0$

$\qquad$ $u_3^* \geq 0$

(5) $\nabla L(x^*, u^*, w^*) = 0$ $L = f(x^*) + w_1^* h_1(x^*) - u_2^* g_2(x^*) - u_3^* g_3 (x$

$$\begin{pmatrix} 2x_1^* \\ 1 \end{pmatrix} + w_1^* \begin{pmatrix} 2x_1^* \\ 2x_2^* \end{pmatrix} - u_2^* \begin{pmatrix} -1 \\ -2x_2^* \end{pmatrix} - u_3^* \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

de donde se obtiene que...

$v_1^* = -0{,}779$

$u_1^* = 1{,}05$

$u_3^* = 0$

a

De acuerdo al Teorema 3, se debe encontrar V tal que $V^T \nabla g_j(x^*) = 0$ para las restricciones de desigualdad activas, y además $V^T \nabla h_1(x^*) = 0$, es decir

$$(v_1, v_2) \begin{pmatrix} -1 \\ -2x_2^* \end{pmatrix} = 0 \qquad V_1 + 2x_2^* V_2 = 0$$

$$[v_1, v_2] \begin{bmatrix} 2x_1^* \\ 2x_2^* \end{bmatrix} = 0 \qquad 2x_1^* V_1 + 2x_2^* V_2 = 0$$

de donde se obtiene que $V = (v_1, v_2)^T = (0, 0)^T$ y entonces existe una única solución al problema en la intersección de $g_2(x)$ y $h_1(x)$.

Note que en este problema en particular

$$V^T \nabla^2 L(x^*, u^*, w^*) V \geq 0$$

para toda $V \neq 0$, es decir $\nabla^2 L(x^*, u^*, w^*)$ es una matriz positiva definida. (condición (6')).

Los dos ejemplos analizados en este capitulo tienen el propósito de ilustrar las condiciones de optimalidad de primero y segundo orden. Ahora bien, los problemas que se pueden tratar analiticamente son demasiado sencillos, sin que esto quiera decir que para problemas muy grandes y/o complicados lo anterior no sea válido.

BIBLIOGRAFIA

1. A.V. Fiacco and G.P. Mc Cormick. "Non linear Programming", John Wiley and Sons, Inc., New York, 1968.

2. W.W. Kuhn and A. W. Tucker, "Non linear Programming. Proc. 2nd. Berdeley Symp. on Mathematical Statistics and Probability, Univ. of Calif. Press, Berkeley, 1951, pp. 481-492.

3. G.P. Mc. Cormick. SIAM J. Appl. Math., 15: 641 (1967)

4. O.L. Mangasarian, "Non linear Programming"; Mc Graw-Hill, New York, 1969.

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

DISEÑO OPTIMO DE SISTEMAS DE INGENIERIA

METODOS DE OPTIMACION CON RESTRICCIONES

Dra. Susana Gómez Gómez

MARZO, 1982

M E T O D O L O G I A

I. BUSQUEDAS UNIDIRECCIONALES

Casi la totalidad de las técnicas de minimización que se describirán en otras secciones más adelante, requieren de técnicas de minimización unidimensionales las cuales tienen como propósito el localizar el mínimo local de una función de una variable. La implementación de los métodos mencionados requieren del conocimiento previo de un cierto intervalo $\underline{\Delta}^{(o)}$ e cual contiene al mínimo de la función objetivo $f(x)$, y además se supone que en el intervalo prescrito la función es unimodal. En la Tabla II se mencionan algunos de los métodos más conocidos para lograr la minimización deseada; todos los cuales tienen como própesito reducir el tamaño del intervalo $\Delta^{(o)}$ hasta un tamaño $\Delta^{(n)}$. Para comparar la rapidez relativa de los métodos, Wilde define una eficiencia para n evaluaciones de la función como :

E

$$\text{Eficiencia} = E = \frac{\Delta^{(n)}}{\Delta(o)} \quad \Delta^{(o)}$$

En la tabla $\overset{\text{II}}{\text{I}}$ se comparan los valores de E para varios métodos.

Métodos no-secuenciales	E	Secuenciales	E
Búsqueda uniforme	$\frac{2}{n+1}$	Búsqueda secuencial (Dicotomus)	$\frac{1}{2^{n/2}}$
Búsqueda uniforme (Dicotomus)	$\frac{1}{\left(\frac{n}{2}\right)+1}$	Búsqueda Fibonacci	$\frac{1}{F_n}$ *
		Sección Dorada	$(0.618)^{1-n}$

(*) : Número de Fibonacci para n evaluaciones

TABLA II. Eficiencia de Técnicas de Búsqueda Unidimensional

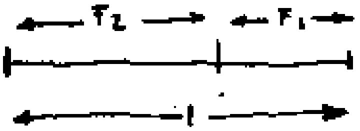
En la Tabla III se compara el número de evaluaciones de la función que se requiere para reducir un intervalo inicial de $5 \times \overset{10^{-1}}{\cancel{15}}$ a uno menor.

	NO SECUENCIAL		SECUENCIAL		
$\Delta^{(n)}$	Uniforme	Dicotomus	Dicotomus	Fibonacci	Sección Dorada
$5 * 10^{-3}$	199	198	14	11	11
$5 * 10^{-5}$	19,999	19,998	28	21	21

TABLA III. Número de evaluación de la función para reducir $\Delta^{(o)}$ a 5×10^{-1}

A continuación se describe el método de la sección dorada. Este método esta basado en la división de una linea en dos segmentos tales que la relación del tamaño original de la linea al segmento mayor es la misma

que la relación del segmento mayor al menor, es decir :



$$F_1 + F_2 = 1$$

$$\frac{1}{F_2} = \frac{F_2}{F_1} \quad ; \quad F_2^{\,2} = F_1$$

de donde

$$F_1 = \frac{3 - \sqrt{5}}{2} \simeq 0.38$$

$$F_2 = \frac{\sqrt{5} - 1}{2} \simeq 0,62$$

Para iniciar la búsqueda del mínimo de $f(x)$, se necesita especificar (o averiguar) en que dirección ésta se llevará a cabo. (Se supondrá que se conoce).

Como primer paso se debe encontrar un intervalo Δ donde se encuentra el mínimo de $f(x)$ usando, por ejemplo, una serie de pasos cada vez más grande sobre la variable independiente. Suponga que esto se ha hecho y que los últimos tres puntos obtenidos en x son los siguientes : $x_3^{(o)}$, el último, $x_2^{(o)}$ y $x_1^{(o)}$, donde $f(x_3^{(o)}) \geq f(x_2^{(o)})$ y sea $\Delta^{(k)} = x_3^{(k)} - x_1^{(k)}$ (Fig. 6). Lo anterior Una vez hecho la k-ésima etapa modifica el intervalo de la siguiente forma.

$$y_1^{(k)} = x_1^{(k)} + F_1 \; \Delta^{(k)}$$

$$y_2^{(k)} = x_1^{(k)} + F_2 \; \Delta^{(k)} = x_3^{(k)} - F_1 \; \Delta^{(k)}$$

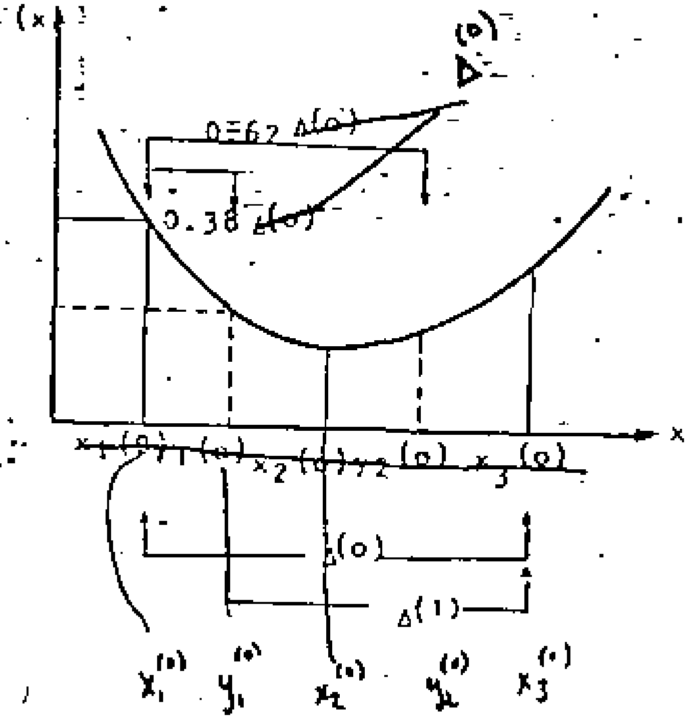
si $f\,(y_1^{(k)}) < f\,(y_2^{(k)})$: $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)})$, y $x_1^{(k+1)} = x_1^{(k)}$, $x_3^{(k+1)} = y_2^{(k)}$

si $f\,(y_1^{(k)}) > f\,(y_2^{(k)})$: $\Delta^{(k+1)} = (x_3^{(k)} - y_1^{(k)})$, y $x_1^{(k+1)} = y_1^{(k)}$, $x_3^{(k+1)} = x_3^{(k)}$

si $f\,(y_1^{(k)}) = f\,(y_2^{(k)})$: $\Delta^{(k+1)} = (y_2^{(k)} - x_1^{(k)}) = (x_3^{(k)} - y_1^{(k)})$, y

$x_1^{(k+1)} = x_1^{(k)}$, $x_3^{(k+1)} = y_2^{(k)}$, o bien

$x_1^{(k+1)} = y_1^{(k)}$, $x_3^{(k+1)} = x_3^{(k)}$



$$y_1^{(o)} = x_1^{(o)} + 0.38 \;\; \Delta^{(o)}$$

$$y_2^{(o)} = x_1^{(o)} + \overset{0}{\cancel{1}}.62 \;\; \Delta^{(o)}$$

FIGURA 6. Búsqueda mediante sección Dorada.

Otra clase de métodos para búsquedas unidireccionales localizan un punto x, cercano a x^* (el mínimo) mediante interpolación y extrapolación. En estos métodos se usan interpolaciones cuadráticas y cúbicas para aproximar el valor de la función. A continuación se describen un par de algoritmos que al usarlos en forma conjunta generan un algoritmo bastante poderoso para la localización de mínimos locales en una dirección: estos dos algoritmos son los siguientes

a) Davis-Swann - Compey (DSC) para definir el intervalo Δ donde se encuentra el mínimo, y

b) Powell, para definir la localización "exacta" del mínimo.

En el método DSC, se toman pasos de tamaño cada vez mayor hasta que se sobrepasa la localización del mínimo. A partir de ese momento se usa interpolación cuadrática (Figura 7). Los pasos que se siguen en este algoritmo son los siguientes

1. Evaluar $f(x)$ en el punto inicial $x^{(o)}$. ~~Si $f(x^{(o)}$ ...~~

   Si $f(x^{(o)} + \Delta x) \leq f(x^{(o)})$ se continua con el paso 2.

   Si $f(x^{(o)} + \Delta x) > f(x^{(o)})$ se define $\Delta x = -\Delta x$ (se cambia la dirección de búsqueda) y se continua con el paso ~~1~~. 2

2. $x^{(k+1)} = x^{(k)} + \Delta x$

3. Calcular $f(x^{(k+1)})$

4. Si $f(x^{(k+1)}) \leq f(x^{(k)})$, se duplica el tamaño de paso Δx y se repite el procedimiento desde el paso 2. Si $f(x^{(k+1)}) > f(x^{(k)})$ sea $x^{(m)} = x^{(k+1)}$, $x^{(m-1)} = x^{(k)}$, etc., redúzcase Δx a la mitad y regrese a los pasos 2 y 3 una vez más.

5. De los cuatro puntos igualmente espaciados x, en el conjunto $\{x^{(m+1)}, x^{(m)}, x^{(m-1)}, x^{(m-2)}\}$, se eliminan o $x^{(m)}$ o $x^{(m-2)}$, el que este más lejos de la x con el valor de la función más baja. Los tres puntos restantes se denotan como $x^{(\overset{a}{\cancel{c}})}$, $x^{(b)}$ y $x^{(c)}$, donde $x^{(b)}$ es el punto central y $x^{(a)} = x^{(b)} - \Delta x$ y $x^{(c)} = x^{(b)} + \Delta x$.

6. Use interpolación cuadrática para estimar x^*,

$$x^* = \bar{x}^* = x^{(b)} + \frac{\Delta x\,[f(x^{(a)}) - f(x^{(c)})]}{2[f(x^{(a)}) - 2f(x^{(b)}) + f(x^{(c)})]}$$

Los pasos anteriores completan la primera etapa del método DSC. Para continuar, se reinicia en $\bar{x}^*$ o $x^{(c)}$, si $f(x^{(\overset{c}{\cancel{c}})}) < f(\bar{x}^*)$, se reduce Δx y se empieza desde el paso 1 nuevamente.

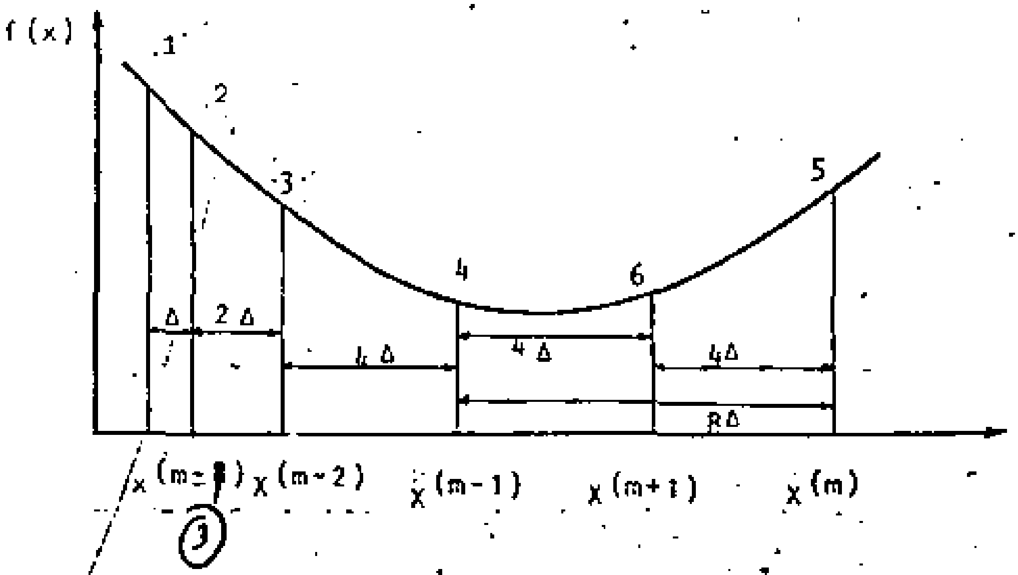


FIGURA 7. Método DSC para minimización unidimensional.

En el método de Powell, se usa una aproximación cuadrática usando los tres primeros puntos obtenidos en la dirección de búsqueda dada. La x correspondiente al mínimo de la función cuadrática se usa para efectuar una nueva aproximación y se continua de esta forma hasta localizar el mínimo de f (x) (Figura 8).

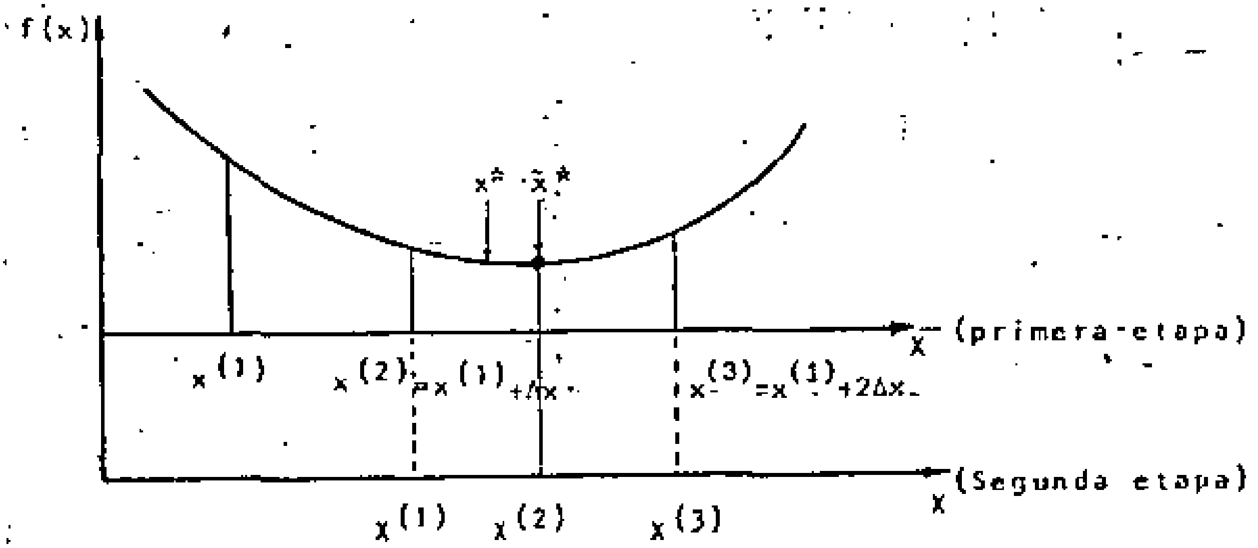


FIGURA 8. Método de Powell para minimización unidimensional.

Los pasos que deben seguirse para implementar el método de Powell son:

1. dados $x^{(1)}$ y Δ, calcule $x^{(2)} = x^{(1)} + \Delta$

2. calcule $f(x^{(1)})$ y $f(x^{(2)})$

3. Si $f(x^{(1)}) > f(x^{(2)})$, $x^{(3)} = x^{(1)} + 2\Delta$

   Si $f(x^{(1)} \leq f(x^{(2)})$, $x^{(3)} = x^{(1)} - \Delta$

4. Calcule $f(x^{(3)})$

5. Estime el valor de x en el mínimo de $f(x)$, $\bar{x}^*$, como

$$x^* \simeq \bar{x}^* = \frac{1}{2}\,\frac{\left[(x^{(2)})^2 - (x^{(3)})^2\right] f(x^{(1)}) + \left[(x^{(3)})^2 - (x^{(1)})^2\right] f(x^{(2)}) + \left[(x^{(1)})^2 - (x^{(2)})^2\right] f(x^{(3)})}{(x^{(2)} - x^{(3)})\, f(x^{(1)}) + (x^{(3)} - x^{(1)})\, f(x^{(2)}) + (x^{(1)} - x^{(2)})\, f(x^{(3)})}$$

6. Si $\bar{x}^*$ o alguna de entre $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ que corresponda al valor más pequeño de $f(x)$, difiere en menos que la exactitud pres~~enta~~crita para x, o la exactitud en el valor de $f(x)$, se termina la búsqueda. En caso contrario, evalúe $f(\bar{x}^*)$ y se elimina del conjunto $\{x^{(1)}, x^{(2)}, x^{(3)}\}$ ~~d~~el que tenga el valor más alto de la función $f(x)$, a menos que se pierda el intervalo de x donde se encuentra el mínimo, en cuyo caso se debe eliminar una x tal que el intervalo adecuado no se pierda. Se repita el procedimiento desde el paso ~~1~~ 5.

~~1~~ 2. MINIMIZACION SIN RESTRICCIONES USANDO DERIVADAS

El problema general de minimización sin restricciones se puede plantear como :

$$\text{Minimizar} : \quad f(x), \quad \cancel{x \in E^n} \quad x \in E^n \qquad (1)$$

donde $f(x)$ es la función objetivo. Según se vió en el capitulo anterior, se pretende encontrar un punto x^* tal que $\nabla f(x^*) = 0$. En la presente sección se atacará el problema definido en (1) mediante el uso de métodos que hagan ~~uso de las~~ empleen primeras y segundas derivadas parciales $(\nabla f(x)$ y $\nabla^2 f(x))$ de la función objetivo.

2.1. Método del Máximo Descenso (Gradiente).

Según se recordará del capitulo anterior, el gradiente de la función objetivo $f(x)$, en cualquier punto x, es un vector dirigido en la dirección del máximo incremento local en $f(x)$. Resulta obvio que se puede escoger como dirección de búsqueda, para minimizar $f(x)$, la dirección opuesta al gradiente, $-\nabla f(x)$, esto es, en la dirección de máximo descenso. Si se escoge como dirección de búsqueda la ya señalada, resultará lo siguiente.

a) sea $\cancel{s_k} \; s^k = -\nabla f(x_k)$ la dirección de búsqueda en el k-ésimo paso del algoritmo.

b) Si $\Delta x_k = + \alpha_{\cancel{A}}^{k} S_k = - \alpha_k \nabla f(x_k)$, es el desplazamiento del punto x_k al x_{k+1}, es decir

$$x_{k+1} = x_k + \Delta x_k \qquad (\alpha_k \text{ un escalar positivo})$$

c) Entonces, la aproximación a primer orden de $f(x)$ quedará como

$$f(x_k + \Delta x_k) = f(x_k) + \nabla^T f(x_k) \Delta x_k$$

o bien

$$f(x_k + \Delta x_k) - f(x_k) = - \alpha_k \nabla^T f(x_k) \nabla f(x_k) < 0$$

es decir, se puede garantizar que para α_k suficientemente pequeño, el valor de la función en x_{k+1} decrecerá con respecto al valor previo en x_k, siempre y cuando $\nabla f(x_k) \neq 0$. (En los puntos (a) – (c), α_k se le conoce como tamaño de paso).

Existen varias alternativas para escoger el tamaño de paso $\alpha_{\cancel{}}\,k$ de las cuales se pueden mencionar las siguientes

a) Fijar el valor de α_k de antemano e igual para todas las iteraciones del método. La desventaja de proceder de esta forma es que no se puede garantizar que de una iteración a la siguiente, el valor de la función decrezca, ya que esta propiedad sólo es válida cuando $\alpha_k \longrightarrow 0$.

Debido a lo anterior, el método puede presentar las siguientes desventajas. En primer lugar, que si α_k no es lo "suficientemente pequeña" el método ascilará. Por otro lado, si α_k se escoge "demasiado pequeña" la convergencia puede volverse demasiado ~~pequeña~~ lenta

b) Una segunda alternativa es escoger α_k en cada iteración de forma tal que el valor de la función objetivo se reduzca para algún cierto valor de α_k. Para lograr lo anterior se puede proceder de la siguiente forma en cada etapa

o) se escoge ~~$\alpha ref > 0$,~~ $\alpha_{ref} > 0$ y $0 < \pi < 1$ un factor multiplicativo

i) se fija $\beta_0 =$ ~~αref~~ α_{ref}, $i = 0$

ii) $y^i = x_k + \beta_0 \Delta x_k$

iii) calcular $f(y^i)$

iv) si $f(y^i) < f(x_k)$ continuar con el paso vii)

v) $i \longleftarrow i + 1$

vi) $\beta_{i+1} = \pi\beta_i$; repetir el procedimiento desde (ii)

vii) $x_{k+1} = y^{(i)}$; $\alpha_k = \beta_i$

$f(x_{k+1}) = f(y^i)$

El procedimiento anterior garantiza que, si se permite un número ilimitado de iteraciones $\beta_{i+1} = \pi\beta_i$, en cada etapa del método de máximo descenso, se obtendrá un descenso en la función objetivo. Tiene el defecto a que puede consumir demasiado tiempo en la búsqueda de un tamaño de paso adecuado α_k.

c) Una tercera alternativa es la siguiente. Supongase ~~que de~~ conocida ~~alguna manera~~ la dirección de búsqueda S_k, es decir el nuevo iterando x_{k+1} caerá sobre la ecuación de un parámetro

$$x_{k+1} = x_k + \alpha S_k$$

siendo α el parámetro. La diferencia entre este procedimiento y los dos anteriores es que en el presente se pide que el tamaño de paso α sea tal que $f(x_{k+1}) = f(x_k + \alpha S_k)$ adquiera su mínimo valor ~~0~~ σ, formalmente

$$\frac{d\, f(x_k + \alpha S_k)}{d\alpha} = 0$$

Por ejemplo, si $f(x)$ es una función cuadrática

$$f(x) = a + b^T x + \tfrac{1}{2} x^T Q x \quad (Q: \text{positiva definida simétrica})$$

entonces

$$S_k = -\nabla f(x_k) = -b - Qx_k$$

$$x_{k+1} = x_k - \alpha_k \; (b + Q\,x_k)$$

$\dfrac{df\left[x_k - \alpha_k(b+Qx_k)\right]}{d\alpha}$ ~~$df(x_k - \alpha_k(b+Qx_k))$~~ $= 0 = \nabla^T f(x_k)\; S_k + S_k^T\, Q\,(\alpha_k\, S_k)$

de donde

$$\alpha_k = -\,\frac{\nabla^T f(x_k) S_k}{S_k^T Q S_k}$$

Si se supone que $f(x)$ no es una función cuadrática de x, entonces se pueden emplear cualquiera de las técnicas de búsquedas unidimensionales descritas en la sección anterior.

Una característica interesante de este procedimiento de minimización es que el gradiente en el nuevo punto, $\nabla f(x_{k+1})$ es ortogonal a la dirección de búsqueda empleada para localizar x_{k+1}, es decir :

$\nabla^T f(x_{k+1})\; S_k = 0$, lo cual se demuestra como sigue. Supóngase la misma función cuadrática ya empleada entonces

$$\nabla f(x_k) = b + Q\,x_k$$

y de la expresión para $\dfrac{d\,f(\alpha)}{d\alpha} = 0$ se obtiene

$$(b + Q\,x_k)^T\; S_k + S_k^T\; Q\;\alpha_k\; S_k = 0$$

y como $a x_{k+1} - x_k = \alpha_k s_k$, entonces

$$s_k^T (b + Q x_k) + s_k^T Q (x_{k+1} - x_k) = 0.$$

o bien

$$s_k^T (b + H x_{k+1}) = s_k^T \nabla f(x_{k+1}) \equiv 0.$$

Lo anterior se ilustra en la Figura 9.

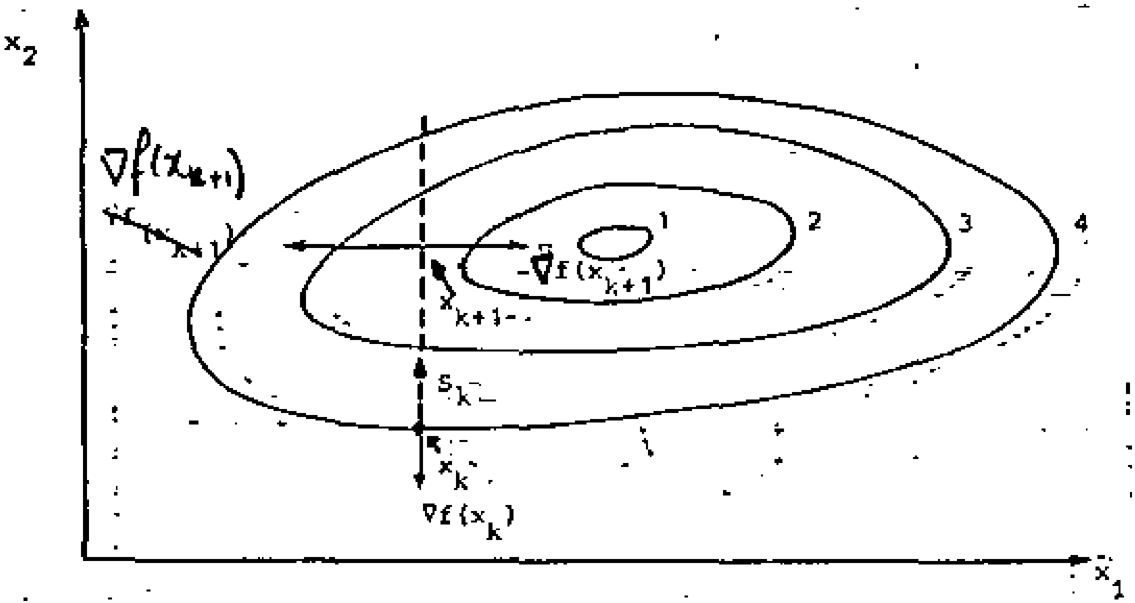


FIGURA 9. Ortogonalidad de las direcciones de Búsqueda.

La implementación práctica del algoritmo de máximo descenso, con cualquiera de las tres alternativas discutidas para determinar α_k, quedaría de la siguiente forma.

1. $k = 0$

2. Estimar x_o (punto de arranque).

3. Calcular $f(x_k)$, $\nabla f(x_k)$

4. Si $\nabla^T f(x_k) \quad \nabla f(x_k) \leq$ tolerancia, se ha encontrado un punto estacionario de la función $f(x)$

5. $S_k = - \nabla f(x_k)$

6. Cálculo de a_k (ver texto)
   como resultado se calcula x_{k+1} y $f(x_{k+1})$

7. Calcular $\nabla f(x_{k+1})$.

8. $k \longrightarrow k+1$; se repite el procedimiento desde el paso 4.

Para finalizar la discusión sobre el método del máximo descenso es conveniente hacer notar que bajo algunas condiciones, por cierto no muy frecuentes, el algoritmo puede ser atraido por un punto silla ya que también en esta clase de puntos se satisface que $\nabla^T f(x) \quad \nabla f(x) = 0$, no existiendo forma de detectar a priori que tal cosa sucederá. Ahora bien, para clasificar el tipo de punto donde se detuvo el algoritmo, es necesario analizar la matriz de segundas derivadas, de acuerdo a :

i) H, positiva definida — mínimo

ii) H, negativa definida — máximo

iii) H, semi-positiva o semi-negativa definida —— punto silla.

2.2. Método de Newton

El método de Newton que a continuación se presenta esta basado en una aproximación a segundo orden de la función objetivo f (x), es decir, usa información de segundo orden (matriz hessiana), de aqui que se le clasifique como método de segundo orden.

Considérese la aproximación a segundo orden de f (x)

$$f(x + \Delta x) = f(x) + \nabla^T f(x)\,\Delta x + \tfrac{1}{2}\,\Delta x^T\,\nabla^2 f(x)\,\Delta x$$

Si se supone que la aproximación anterior es buena, como de hecho lo es en la vecindad de un mínimo y de un máximo, entonces, al derivar parcialmente f (x + Δx) con respecto a cada uno de los elementos de Δx se obtiene

$$\frac{\partial f(x+\Delta k)}{\partial \Delta x} = \nabla f(x) + \nabla^2 f(x)\Delta x = 0$$

de donde

$$\Delta x = - H^{-1}(x)\nabla f(x)$$

donde $H(x) = \nabla^2 f(x)$ es la matriz hessiana $\left(hij = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}\right)$.

y ΔX será la dirección del desplazamiento desde un punto x_k a x_{k+1} ,

es decir

$$x_{k+1} = x_k - H^{-1}(x_k)\nabla f(x_k)$$

El empleo de la última fórmula para generar los iterandos del método de Newton puede provocar los siguientes problemas si es que el método se está empleando para minimizar una función f (x). El problema es el siguiente. Note que tanto un máximo como un mínimo, asi como los puntos silla, satisfacen $\overset{\nabla f = 0}{\cancel{\partial f(x + \Delta x)/\partial(x) = 0}}$, . por lo que no se puede garantizar que el método converga a un mínimo, si no se modifica adecuadamente Esta modificación consiste en lo siguiente :

sea $\Delta x_k = -H^{-1}(x_k)\nabla f(x_k)$ la dirección de avance del punto x_k al x_{k+1} y considérase la aproximación a primer orden de f (x) como sigue :

$$f(x_k + \Delta x_k) = f(x_k) + \alpha_k(\nabla^T f(x_k)\,\Delta x_k)\rho_k$$

o bien

$$\Delta f_k = f(x_k + \Delta x_k) - f(x_k) = \alpha_k(\nabla^T f(x_k)\,\Delta x_k)\,\overset{\rho_k}{\cancel{\rho k}}$$

donde α_k es el tamaño de paso, descrito en el inciso anterior, y ρ_k es el sentido de la dirección de búsqueda, el cual se determina como sigue : ya que se desea que $\Delta f_k < 0 \; (f(x_{k+1}) < f(x_k))$ y como $\alpha_k > 0$, entonces

$$\rho_k = \begin{cases} 1 & \text{si } \nabla^T f(x_k)H^{-1}(x_k)\nabla f(x_k) > 0 \\ -1 & \text{si } \nabla^T f(x_k)H^{-1}(x_k)\nabla f(x_k) < 0 \end{cases}$$

Y con esto se garantiza que si $\alpha_k(>0)$ es suficientemente pequeña entonces $f(x_{k+1}) < f(x_k)$ y el método de Newton convergerá a un mínimo, o en el peor de los casos a un punto silla, pero nunca a un máximo (se omite la discusión sobre el tamaño de paso α_k, por ser idéntica a la presentada con anterioridad).

Tomando en cuenta los puntos anteriores, la implementación del método de Newton queda como sigue

1. $k = 0$

2. Estimar x_0 (punto de arranque)

3. Calcular $f(x)$,

4. Calcular $\nabla f(x_k)$;

5. Si $\nabla^T f(x_k) \quad \nabla f(x_k) \leq$ tolerancia, se ha encontrado un punto estacionario de $f(x)$ (mínimo o punto silla).

6. Calcular $H(x_k)$ y $H^{-1}(x_k)$.

7. $$\rho_k = \begin{cases} 1 & \text{si. } \nabla^T f(x_k) \; H^{-1}(x_k) \; \nabla f(x_k) > 0 \\ -1 & \text{si } \nabla^T f(x_k) \; H^{-1}(x_k) \; \nabla f(x_k) < 0 \end{cases}$$

8. $\Delta x_k = - \overset{(k}{\cancel{\rho_k}} H^{-1}(x_k) \nabla f(x_k)$

9. Calcular α_k (inciso 2.1)

   como resultado se obtiene x_{k+1}, $f(x_{k+1})$

10. $k \longleftarrow k+1$. Se repite el procedimiento desde (4).

Es fácil ver que si la función objetivo es cuadrática, p. ej.

$$f(x) = a + b^T x + \tfrac{1}{2} x^T Q x$$

el método de Newton converge en una sola iteración, ya que

$$\nabla f(x_k) = b + Q x_k$$

y $$\nabla^2 f(x_k) = Q$$

entonces

$$x_{k+1} = x_k - \left[\nabla^2 f(x)\right]^{-1} \nabla f(x_k) = x_k - Q^{-1}(b + Qx_k)$$

$$\boxed{Q^{-1} b = x^*}$$

convergencia cuadrática

2.3. Conjugancia y Direcciones Conjugadas

Como se verá más adelante, una función objetivo cuadrática de n-variables que exhibe un mínimo, puede ser minimizada en n pasos (o menos) si estos pasos se toman en lo que se ha llamado direcciones conjugadas. En el inciso siguiente se limitará la discusión a funciones cuadráticas del tipo

$$f(x) = a + b^T x + \tfrac{1}{2} x^T H x \qquad (o)$$

donde H es una matriz positiva definida.

2.3.1. Conjugancia

Supóngase que la minimización de $f(x)$ empieza en x_o en la dirección $\bar{S}^o$, escogida arbitrariamente (o por algún algoritmo); se supondrá $(\bar{S}_o)^T \bar{S}_o = 1.0$. El siguiente punto generado por el algoritmo será

$$x_1 = x_o + \lambda_o \bar{S}_o \qquad (1)$$

donde el tamaño de paso λ_o se determina minimizando $f(x_o + \lambda_o \bar{S}_o)$ con respecto a λ, es decir

$$\frac{df(x_o + \lambda_o \bar{S}_o)}{d\lambda} = 0 = \nabla^T f(x_o)\bar{S}_o + (\bar{S}_o)^T \nabla^2 f(x_o)\left(\bar{S}_o \lambda_o\right)$$

de donde

$$\lambda_o = - \frac{\nabla^T f(x_o)\bar{S}_o}{\bar{S}_o^T \nabla^2 f(x_o)\bar{S}_o} \qquad (2)$$

Una vez que se encuentra el siguiente iterando, x_1, se deberá seleccionar una nueva dirección de búsqueda para la minimización de $f(x)$. La nueva dirección $\bar{S}_1$ se dice que es conjugada con $\bar{S}_o$ si $(\bar{S}_1)^T \nabla^2 f(x_o)\bar{S}_o = 0$. (En general, un conjunto de n direcciones independientes de búsqueda S_o, S_1, ... S_{n-1} son conjugadas con respecto a una matriz Q, positiva definida, si

$$S_i^T Q S_j = 0 \qquad 0 \le i \neq j \le n-1 \tag{3}$$

Q, podría ser, por ejemplo, la matriz hessiana de la función objetivo H. Note además que si $Q = I$, la matriz unitaria, conjugancia y ortogonalidad son sinónimos).

Debido a que los vectores $\overset{S_i}{\cancel{\alpha_i}}$, son linealmente independientes además de conjugados, cualquier vector $V \in E^n$, se puede representar en términos de aquellos como:

$$V = \sum_{j=0}^{n-1} \overset{\nu_j}{\cancel{\alpha}}_j \, S_j$$

$$\overset{\nu}{\cancel{\alpha}}_j = \frac{S_j^T \, H(x) \, V}{S_j^T \, H(x) \, S_j}$$

Otra relación importante que se utilizará más adelante es la siguiente. Considerese la matriz P definida por

$$P = \sum_{j=0}^{n-1} o_j \, S_j \, S_j^T$$

Resulta obvio que si las o_j se escogan de manera tal que

$$P H s_k = s_k$$ mayúsculas

entonces $P = H^{-1}$. Ahora bien.

$$P\,H\,S_k = \left(\sum_{j=0}^{n-1} \alpha_j S_j S_j^T\right) H S_k = \sum_{j=0}^{n-1} \alpha_j S_j (S_j^T\, H S_k)$$

$$= \alpha_k\, S_k\, (S_k^T\, H\, S_k)$$

de donde si $\alpha_k = (S_k^T\, H\, S_k)^{-1}$, entonces $P = H^{-1}$,

es decir $$\boxed{H^{-1} = \sum_{j=0}^{n-1} \frac{S_j\, S_j^T}{S_j^T\, H\, S_j}}$$

Si las direcciones de búsqueda empleadas en la minimización de $f(x)$ se escogen conjugadas (esto se demostrará más adelante) a continuación se demuestra que : cualquier función cuadrática de n variables que exhibe un mínimo, puede ser minimizada en n pasos, si se emplean direcciones conjugadas, una dirección diferente en cada paso. Además, el orden en que se usan las direcciones de búsqueda es irrelevante para alcanzar el mínimo.

Demonstración Sea $f(x) = a + b^T x + \frac{1}{2} x^T H x$, $\nabla f(x) = b + Hx$

$\nabla^2 f(x) = H$, y en el mínimo de x, $\nabla f(x^*) = 0$, es decir $x^* = -H^{-1} b$

Para la n-ésima etapa se tiene, usando (1) y (2), que

$$x_n = x_o + \sum_{k=o}^{n-1} \lambda_k \bar{S}_k$$

y como en cada etapa se usó el valor óptimo de λ_k dado por la ecuación (2),

$$x_n = X_o - \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T \nabla f(x_k)}{(\bar{S}_k)^T H \bar{S}_k} \bar{S}_k \qquad (5a)$$

Por otro lado

$$\begin{aligned}(\bar{S}_k)^T \nabla f(x_k) &= (\bar{S}_k)^T (Hx_k + b) \\ &= (\bar{S}_k)^T \left\{ H(X_o + \sum_{i=1}^{n-1} \lambda_i \bar{S}_i) + b \right\} \\ &= (\bar{S}_k)^T (H_o + b) \Big\} \text{ por conjugancia de las } \bar{S}_i .\end{aligned}$$

Entonces

$$x_n = x_o - \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T (H x_o + b) \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k} \qquad (5\ b)$$

Usando la relación (4 a) se obtiene que

$$x_o = \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T H x_o}{(\bar{S}_k)^T H \bar{S}_k} \bar{S}_k$$

y por lo tanto

$$x_n = \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T b \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k} = - \sum_{k=o}^{n-1} \frac{(\bar{S}_k)^T H (H^{-1} b) \bar{S}_k}{(\bar{S}_k)^T H \bar{S}_k}$$

y usando (4 a) nuevamente, se obtiene

$$\underline{x_n = - H^{-1} b}$$ q.e.d.

Un método para el cual se garantiza que alcanza el mínimo de una función objetivo cuadrática en un número especifico de pasos, se dice que tiene la propiedad de <u>terminación cuadrática</u>. (El método de gradiente conjugado necesita de n pasos, mientras que el de Newton uno sólo).

2.3.2. <u>Método de Gradiente conjugado</u>

El método de Fletcher - Reeves de gradiente conjugado, que a continuación se describe, genera una secuencia de direcciones de búsqueda que son combinación lineal de $- \nabla f(x_k)$, -la dirección de máximo descenso en el último punto, y de las k direcciones de búsqueda anteriores, s_o, s_1, ..., s_{k-1}, -usando factores de peso α_k tales que S_k sea conjugada a las direcciones anteriores.

Para ilustrar el método, sea $S_o = - \nabla f\ (x_o)$, y

$x_1 = x_o + \lambda^*_o S_o$; sea

$$S_1 = - \nabla f\ (x_1) + \alpha_1 S_o \qquad (6)$$

donde α_1 se escoge de forma tal que S_o y S_1 sean conjugadas con respecto a H, es decir

$$S_o^T H S_1 = 0 \qquad (7)$$

Para eliminar S_o, considérese la aproximación a primer orden del gradiente, es decir

$$\begin{aligned}\nabla f(x_1) - \nabla f(x_o) &= \nabla^2 f(x_o)(x_1 - x_o) \\ &= \lambda_o^* H S_o\end{aligned} \qquad (8)$$

$$S_o = \frac{1}{\lambda_o^*} H^{-1}\left[\nabla f(x_1)\nabla f(x_o)\right]$$

y como H es simétrica, entonces.

$$S_o^T = \frac{\left[\nabla f(x_1)-\nabla f(x_o)\right]^T H^{-1}}{\lambda_o^*} \qquad (9)$$

Substituyendo (6) y (9) en (7) se obtiene

$$\left[\nabla f(x_1) - \nabla f(x_o)\right]^T \left[-\nabla f(x_1) + \alpha_1 S_o\right] = 0 \qquad (10)$$

ya que, según se vió con anterioridad $\nabla^T f(x_o)\nabla f(x_1) = \nabla^T f(x_1) S_o = 0$, entonces

$$\alpha_1 = -\frac{\nabla^T f(x_1)\nabla f(x_1)}{\nabla^T f(x_o) S_o}$$

o bien,

$$\alpha_1 = -\frac{\nabla^T f(x_1)\nabla f(x_1)}{\nabla^T f(x_o)\nabla f(x_o)} \qquad (11)$$

La dirección de búsqueda S_2 se forma como una combinación lineal de $-\nabla f(x_2)$, S_1 y S_o, y se fuerza a que sea conjugada a S_1 y S_o

con lo que se obtiene la siguiente expresión para los factores de peso α_k

$$\alpha_k = \frac{\nabla^T f(x_k) \; \nabla f(x_k)}{\nabla^T f(x_{k-1}) \; \nabla f(x_{k-1})} \tag{12}$$

La implementación del algoritmo de gradiente conjugado de Fletcher - Reeves incluye los siguientes pasos :

1. Estimar x_o (punto de arranque)

2. $S_o = - \nabla f(x_o)$

3. En la k - ésima etapa del algoritmo, se determina el mínimo unidireccional de $f(x)$, a lo largo de la dirección de búsqueda S_k. Con esto se localiza x_{k+1}.

4. La nueva dirección de búsqueda se determina como

$$S_{k+1} = - \nabla f(x_{k+1}) + \frac{\nabla^T f(x_{k+1}) \; \nabla f(x_{k+1})}{\nabla^T f(x_k) \; \nabla f(x_k)} \; S_k$$

   Después de (n+1) iteraciones (k = n), se empieza un nuevo ciclo del algoritmo, es decir, x_{n+1} se convierte en x_o.

5. La búsqueda se da por terminada cuando en alguna iteración sucede que

$$S_k^T \; S_k \leq \text{tolerancia}$$

Note que al igual que en el método de gradiente ordinario, no se necesita la inversión de matriz alguna, lo cual es una ventaja.

2.4. Métodos de Métrica Variable

Los métodos de Métrica Variable o Cuasi-Newton son métodos que aproximan el hessiano, o su inversa, usando unicamente información acerca del gradiente. La mayoria de estos métodos usan direcciones conjugadas conforme avanzan, lo cual hacen siguiendo el esquema general

$$x_{k+1} = x_k + \lambda_k S_k = x_k - \alpha_k \eta(x_k)\nabla f(x_k) \qquad (1)$$

donde $\eta(x_k)$ representa una aproximación a $H^{-1}(x_k)$. (En el método de Gradiente ordinario $\eta(x_k) = I$, mientras que el método de Newton toma $\eta(x_k) = H^{-1}(x_k)$, con la desventaja de que hay que invertir el hessiano).

En una serie de métodos de Cuasi-Newton, $H^{-1}(x_{k+1})$ se aproxima de la información disponible en la k-ésima etapa, como

$$H^{-1}(x_{k+1}) \approx \omega\eta_{k+1} = \omega(\eta_k + \Delta\eta_k) \qquad (2)$$

donde η es una aproximación a H^{-1}, $\Delta\eta_k$ es una matriz que se especifica de acuerdo al método, y ω una constante de escalamiento que frecuentemente se fija en 1. La selección de $\Delta\eta_k$ determina, esencialmente, el método de método variable. Para garantizar convergencia, $\omega\eta_{k+1}$ debe ser positiva definida y debe satisfacer la siguiente relación cuando reemplaza a H^{-1}

$$x_{k+1} - x_k = H^{-1}(x_k)\left[\nabla f(x_{k+1}) - \nabla f(x_k)\right] \qquad (3\ a)$$

que es una aproximación a primer orden del gradiente.

En la $(k+1)$ - ésima etapa de cualquier método se conocen

x_k, $\nabla f(x_k)$, x_{k+1}, $\nabla f(x_{k+1})$ y η_k , y se desea calcular η_{k+1}

De la relación obtenida con (2) y (3) como

$$\eta_{k+1}\,\Delta g_k = \frac{1}{\omega}\Delta x_k \qquad (3\ b)$$

donde $\Delta g_k = \nabla f(x_{k+1}) - \nabla f(x_k)$. Sea $\Delta\eta_k = \eta_{k+1} - \eta_k$, por lo que la ecuación

$$\Delta\eta_k \Delta g_k = \frac{1}{\omega}\Delta x_k - \eta_k \Delta g_k \qquad (3\ c)$$

debe ser resuelta para $\Delta\eta_k$, y esta se obtiene como sigue. Si el lado derecho de (3 c) se multiplica y divide por $y^T \Delta g_k$, el primer término, y $Z^T \Delta g_k$ el segundo término se obtiene que:

$$\left[\Delta\eta_k - \left(\frac{1}{\omega}\,\frac{\Delta x_k y^T}{y^T \Delta g_k} - \frac{\eta_k \Delta g_k Z^T}{Z^T \Delta g_k}\right)\right]\Delta g_k = 0$$

o bien,

$$\Delta\eta_k = \frac{1}{w}\left[\frac{\Delta x_k y^T}{y^T \Delta g_k} - \frac{\eta_k \Delta g_k}{Z^T \Delta g_k} Z^T\right] \qquad (4)$$

donde los vectores columna y, Z son arbitrarios, al igual que w. Si por ejemplo se escogen

$$w = 1$$

$$y = Z = \Delta x_k - \eta_k \Delta g_k$$

se genera el algoritmo de Broyden, mientras que si se escogen

$$y = \Delta x_k$$

$$Z = \eta_k \Delta g_k$$

entonces la matriz η_{k+1} se actualiza de acuerdo al método de Davidon - Fletcher - Powell. Ya que los vectores y, Z son arbitrarios se pueden efectuar varias selecciones, las que se discuten más adelante. Si los pasos Δx_k se determinan mediante minimizaciones unidireccionales de $f(x)$ en la dirección S_k, todos los métodos que calculan una η_{k+1} simétrica que satisfagan (3 b), generan direcciones que son mutuamente ortogonales (para funciones cuadráticas)..

2.4.1. $\Delta\eta_k$ de Rango 1

Broyden demontró que si $\Delta\eta_k$ es simétrica con rango 1, la relación $\eta_{k+1}\,\Delta g_k = \Delta x_k$ se satisface, la única posibilidad de escoger $\Delta\eta_k$ es

$$\Delta\eta_k = \frac{[\Delta x_k - \eta_k \Delta g_k][\Delta x_k - \eta_k \Delta g_k]^T}{[\Delta x_k - \eta_k \Delta g_k]^T \Delta g_k} \qquad (5)$$

El algoritmo funcionaría de la siguiente forma. Se escogen x_o y $\eta_o > 0$ y se usan en ~~usa una~~ forma secuencial (1), (5) y (2) hasta que por ejemplo , $\nabla^T f(x_k) \nabla f(x_k) \leq \varepsilon$. Por otro lado, si se usan minimizaciones unidireccionales, el método genera direcciones conjugadas y bajo algunas condiciones más o menos restrictivas, se puede demostrar que el algoritmo converge a la solución. Una característica atractiva de este método es que α_k en (1) no necesariamente tiene que ser un parámetro que minimize $f(x)$ a lo largo de S_k. El mismo Broyden demuestra que α puede tomar cualquier valor con la única condición de que no provoque que η se haga singular (denominador en (5)).

Si la función objetivo no es cuadrática, algunos de los aspectos poco satisfactorios al usar (5), son los siguientes :

1. η puede dejar de ser positiva definida, en cuyo caso es necesario recurrir a alguna otra estrategia que lo garantice.

2. La corrección $\Delta\eta_k$ puede no quedar acotada (generalmente por errores de redondeo, incluso para funciones cuadráticas)

3. Si por coincidencia $\Delta x_k = -\alpha_k \eta(x_k) \nabla f(x_k)$ queda en la dirección del paso anterior, $\eta(x_{k-1})$ se vuelve singular.

E consecuencia, en el algoritmo de Broyden, si sucede que

$$\eta_k \Delta g_k = \Delta x_k$$

o

$$(\eta_k \Delta g_k - \Delta x_k)^T \Delta g_k^* = 0$$

se fuerza a que

$$\eta_{k-1} = \eta_k \qquad (\Delta \eta_k = 0)$$

2.4.2. Método de Davidon – Fletcher – Powell

En este método la matriz $\Delta\eta$ se escoge que tenga rango 2. La η inicial normalmente se toma como $\eta = I$. (se puede usar cualquier otra matriz simétrica positiva definida), con lo que el método arranca con la dirección del máximo descenso. Conforme el método avanza, va existiendo un cambio del método de gradiente a Newton con lo que se obtiene una gran ventaja al usar las mejores características de ambos métodos.

Como se mencionó con anterioridad la relación para $\Delta\eta_k$ en el método de Davidon – Fletcher – Powell, $y_k = \Delta x_k$ y $Z_k = \eta_k \Delta g_k$ con lo que al substituir en (4) se obtiene

$$\begin{aligned}\eta_{k-1} &= \eta_k + A_k + B_k \\ &= \eta_k + \frac{\Delta x_k (\Delta x_k)^T}{(\Delta x_k)^T \Delta x_k} - \frac{\eta_k \Delta g_k (\Delta g_k)^T \eta_k^T}{(\Delta g_k)^T \eta_k \Delta g_k}\end{aligned} \tag{5}$$

en donde las matrices A_k y B_k son simétricas y si, además, η_k es también simétrica, entonces η_{k+1} también lo será. La relación anterior (Ec. (5)) produce resultados satisfactorios en la práctica siempre y cuando

1. El error al evaluar $\nabla f(x_k)$ no sea grande

2. η_k no se haga mal-condicionada

El papel de la matriz A_k en la ecuación (5) es garantizar que $\eta \longrightarrow H^{-1}$, mientras que la matriz B_k garantiza que η_{k+1} sea positiva definida en todos los pasos, y en el límite se cancela con η_o. Esto se puede ver como sigue

$$\eta_1 = I + A_o - B_o$$

$$\eta_2 = \eta_1 + A_1 - B_1 = I + (A_o + A_1) - (B_o - B_1)$$

$$\eta_{k+1} = I + \sum_{i=0}^{k} A_k - \sum_{i=0}^{k} B_k$$

Para una función cuadrática la suma de las matrices A_i debe ser igual a H^{-1} cuando $k = n - 1$, y la suma de las matrices B_i deberá cancelar η_o (I en este caso), se puede decir que el método de Davidon – Fletcher – Powell refleja, en cierta forma, toda la información ganada en iteraciones anteriores, a través de η.

Debe señalarse que el método que se está ~~descubriendo~~ describiendo usa direcciones conjugadas si la función objetivo es cuadrática. Para que la última dirección, S_{n-1}, sea conjugada a todas las anteriores, se debe

cumplir que : $X_{n-1}^T H S_{n-1} = 0$

~~$X_{n-1} H_{n-1} s_{n-1} = 0$~~

si se substituye que $S_{n-1} = -\eta_{n-1} \nabla f(x_{n-1})$, entonces

$$X_{n-1}^T H \eta_{n-1} \nabla f(x_{n-1}) = 0 .$$

~~$X_{n-1}^T H_{n-1} f(x_{n-1}) = 0$~~ (6)

donde $X_{n-1} = \left[\Delta x_0, \Delta x_1, \ldots, \Delta x_{n-1}\right]$. Si $H \eta_{n-1} = I$ ($\eta_{n-1} = H^{-1}$), entonces $\nabla f(x_{n-1})$ es conjugada a todas las direcciones de búsqueda anteriores dadas por $\Delta x_0, \Delta x_1, \ldots, \Delta x_{n-1}$. Sabiendo que todas las direcciones de búsqueda son conjugadas, se puede demostrar que $\sum_{i=0}^{n-1} A_i = H^{-1}$, como sigue. Ya que ~~Como~~ $\Delta g_k = H \Delta x_k$, entonces el numerador ~~[illegible]~~ y denominador de cada A_i es

$$(\Delta x_k)(\Delta x_k)^T = (\alpha_k s_k)(\alpha_k s_k)^T = \alpha_k^2 s_k s_k^T$$

$$(x_k)^T g_k = (\alpha_k s_k)^T (H \alpha_k s_k) = x^2 s_k^T H s_k$$

de donde

$$\sum_{i=0}^{n-1} A_i = \sum_{i=0}^{n-1} \frac{s_i s_i^T}{s_i^T H s_i} = H^{-1} \qquad (7)$$

que es la fórmula (4 b). Sec. 2,3, obtenida con anterioridad.

Para terminar la presentación de este método, se hacen dos comentarios sobre la implementación práctica del mismo

1. En algunos problemas, los métodos de métrica variable fallan en alcanzar el mínimo de la función objetivo si el grado de precisión en la búsqueda unidimensional no es suficientemente fina. Se recomienda que la precisión en la búsqueda unidireccional sea al menos equivalente que en que se requiere para detener al algoritmo completo.

2. La búsqueda por el mínimo se debe detener, si al evaluar los vectores $-\eta_k \Delta f(x_k)$ y $-\alpha_k \eta_k \nabla f(x_k)$ ocurre cualquiera de los dos siguientes puntos.

   a) Cada componente en ambos vectores es menor que una tolerancia dada.

   b) Las longitudes predichas al mínimo, de cualquiera de los dos vectores es inferior a una cierta tolerancia.

2.4.3. Algorifmos de Pearson

Pearson propusó una serie de algoritmos para calcular η, usando direcciones que fueran conjugadas. Los algoritmos de Pearson se pueden obtener empleando diferentes vectores y, Z en la ecuación (4) del inciso 2.4., según se muestra a continuación

1. Pearson No. 2 Sea $y = Z = \Delta x_k$ y $w = 1$.

Entonces

$$\eta_{k+1} = \eta_k + \frac{(\Delta x_k - \eta_k \Delta g_k)(\Delta x_k)^T}{(\Delta x_k)^T \Delta g_k}$$

$$\eta_o = R_o$$

donde R_o es cualquier simétrica positiva definida. Este algoritmo generalmente conduce a matrices mal condicionadas.

2. Pearson No. 3 Sea $y = Z = \eta_k \Delta g_k$, con $w = 1$.

Entonces

$$\eta_{k+1} = \eta_k + (\Delta x_k - \eta_k \Delta g_k) \frac{(\eta_k \Delta g_k)^T}{(\Delta g_k)^T \eta_k \Delta g_k}$$

$$\eta_o = R_o$$

Este algoritmo se comporta bastante parecido al de Davidon - Fletcher - Powell, excepto que el tamaño del paso es, en general, inferior al de este último.

3. Newton - Raphson Proyectado.

Pearson propuso este otro algoritmo al cual se puede obtener haciendo que $\overset{\omega}{\cancel{x}} \longrightarrow \infty$ y $Z = \eta_k \ \Delta g_k$, con lo que se obtiene.

$$\eta_{k+1} = \eta_k - \frac{(\eta_k \ \Delta g_k)(\eta_k \ \Delta g_k)^T}{(\Delta g_k)^T \ \eta_k \ \Delta g_k}$$

$$\eta_o = R_o$$

Este método incluye la siguiente regla de reinicio cada n etapas, donde n es el número de variables independientes, sobre la matriz η_k.

$$R_{k\overset{+}{\downarrow}1} = R_k + \frac{(\Delta x_k - R_k \Delta g_k)(\eta_k \Delta g_k)^T}{(\Delta g_k)^T \eta_k \Delta g_k}$$

es decir, cada n etapas se toma $\eta_k = R_k$.

BIBLIOGRAFIA

1. C.G. Broyden, Math. Computation, 21 : 368 (1967)

2. M.J. Box, D. Davis, and W.H. Swann. "Non-linear Optimization Techniques", Chemical Industries Monograph 5, Oliver and Boyd, Edinburgh, 1970

3. G.E. Coggins, Univariate Search Methods, Imperial chemical Industries, Ltd., Central Instr. Lab. Res., Note 64/11, 1964

4. W.C. Davidon, USAEC Doc. ANL - 5990 (Rev.), Nov., 1959

5. W.C. Davidon, Computes J., 10 : 406 (1908) ; Chap. 2 in R. Fletcher (ed.), "Optimization" Academic Press Inc. N.Y., 1969

6. R. Fletcher and M.J.D. Powell, Computer J., 6 : 163 (1963)

7. R. Fletcher and C.M. Reeves, Computer J., 7 : 149 (1964)

8. M.R. Hestenes, The Conjugate Gradient Method for Solving Linear Systems, in Proceedings of the Symposium on Applied Mathematics, Vol. VI, Mc. Graw-Hill Book Company, New York, 1956, pp. 83-102.

9. M.R. Hestenes and E.L. Steifel, J. Res. Nat. Bur. Std., B 49 : 409 (1952)

10. J.D. Pearson, Computer J., 13 : 171 (1969)

11. M.J.D. Powell, Computer J., 7 : 155 (1964)

12. M.J.D. Powell; Rank One Methods for Unconstrained Minimization,

AERE Rept., TP 372, 1969

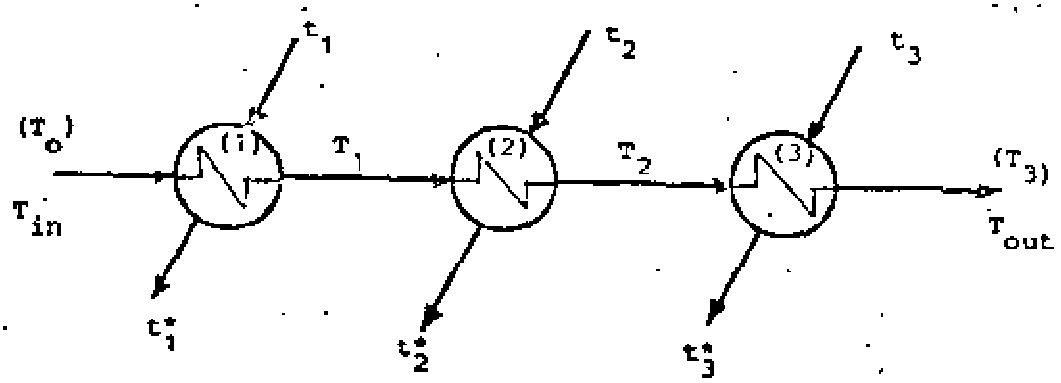
13. D.J. Wilde, "Optimum Seeking Methods", Prentice Hall, Inc.,

Englewood Cliffs, N.J., 1964.

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

OPTIMIZACION DE UN TREN DE CAMBIADORES DE CALOR

Se desea calentar un fluido desde una temperatura T_{in} hasta una temperatura de salida T_{out}, mediante intercambio de calor con tres corrientes liquidas calientes, en un sistema de tres cambiadores de calor que operan en contra corriente, según se muestra en la figura



Para el proceso anterior se especifican los siguientes parámetros :

WC_p : producto del flujo en masa por la capacidad calorifica (se supone idéntica en todas las corrientes)

t_i : temperatura de entrada de las corrientes calientes (i = 1, 2, 3)

U_i : coeficiente total de transferencia de calor en cada uno de los tres cambiadores (i = 1, 2, 3)

Si se supone que la inversión total requerida para el sistema de cambiadores es proporcional al area total de los mismos, el problema se puede plantear como el de seleccionar las areas A_i (i = 1,2,3) de manera tal que :

$$A_T = A_1 + A_2 + A_3$$

sea mínima.

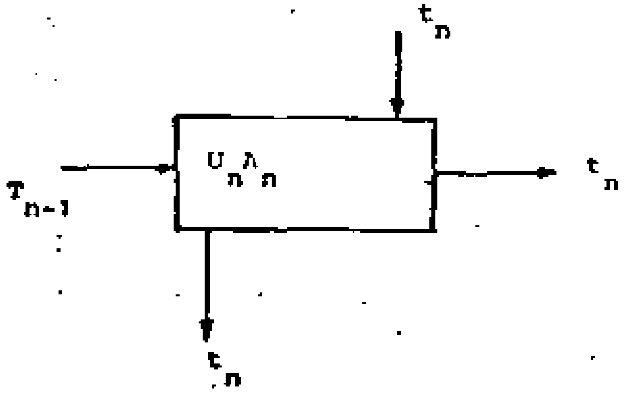
1°) Formular el problema como una optimización en estado estacionario. Identifique la función objetivo, variables de decisión y restricciones. Los siguientes parámetros se consideran fijos

Tin = To = 100	$t_1 = 300$	$U_1 = 120$
Tout = T_3 = 500	$t_2 = 400$	$U_2 = 80$
WC = 100,000	$t_3 = 600$	$U_3 = 40$

2°) Suponga que en el problema anterior las corrientes de salida caliente de los cambiadores 1 y 2 deben mezclarse y la temperatura resultante no deberá exceder los 230° . Plantee este nuevo problema tomando en cuenta la restricción planteada.

Modelo del cambiador de calor

Considerése el cambiador de calor a contra corriente que se muestra a continuación



Las ecuaciones que describen su funcionamiento son :

Q_n : rapidez de transferencia de calor en el n-ésimo cambiador

$$= WC \; (T_n - T_{n-1})$$

$$= WC \; (t_n - t_n^*)$$

$$= U_n A_n \; (t_n - T_n)$$

$$= U_n A_n \; (t_n^* - T_{n-1})$$

siendo las temperaturas de salida las siguientes :

$$T_n = \frac{T_{n-1} + \alpha \quad t}{1 + \alpha \quad n}$$

donde $\cancel{\alpha_n}\ \alpha_n = \frac{U_n A_n}{WC}$

$$y \qquad t_n^* = t_n - (T_n - T_{n-1})$$

Función objetivo

Según se mencionó con anterioridad, se trata de minimizar el area total del sistema

$$A_T = A_1 + A_2 + A_3$$

Variables de decisión y restricciones

1- Sin restricciones en el mezclado (Parte 1)

a) T_1 y T_2 como variables de decisión

[illegible]

$$Q_1 = W\,C\,(T_1 - T_o) \qquad A_1 = \theta_1/U_1\,(t_1 - T_1)$$

$$Q_2 = W\,C\,(T_2 - T_1) \qquad A_2 = Q_2/U_2\,(t_2 - T_2)$$

$$Q_3 = W\,C\,(T_3 - T_2) \qquad A_3 = Q_3/U_3\,(t_3 - T_3)$$

Restricciones sobre las variables independientes

$$T_o \leq T_1 \leq t_1$$

$$T_1 \leq T_2 \leq t_2$$

Restricciones sobre las variables dependientes

$$Q_i \geq 0 \qquad i = 1, 2, 3$$

o equivalentemente

$$A_i \geq 0 \qquad i = 1, 2, 3$$

b) ~~Areas~~ Areas A_1 y A_2 como variables independientes

$$T_1 = \frac{T_o + a_1 t_1}{1 + a_1} \quad ; \quad a_1 = U_1 A_1 / W C$$

$$T_2 = \frac{T_1 + a_2 t_2}{1 + a_2} \quad ; \quad a_2 = U_2 A_2 / W C$$

Q_1, Q_2 y Q_3 como en el caso anterior, al igual que A_3.

Restricciones en las variables independientes

$$0 \leq A_1 \leq A_1^* \quad \text{(dato)}$$

$$0 \leq A_2 \leq A_2^* \quad \text{(dato)}$$

restricciones en las variables dependientes : no hay.

c) Tomar términos Q_1 y Q_2 como variables de decisión :

$$T_1 = Q_1 / WC + T_o \quad ; \quad A_1 = Q_1 / U_1 \, (t_1 - T_1)$$

$$T_2 = Q_2 / WC + T_1 \quad ; \quad A_2 = Q_2 / U_2 \, (t_2 - T_2)$$

$$Q_3 = WC \, (T_3 - T_2) \quad ; \quad A_3 = Q_3 / U_3 \, (t_3 - T_3)$$

Restricciones sobre las variables independientes

$$0 \leq Q_1 \leq WC \, (t_1 - T_o)$$

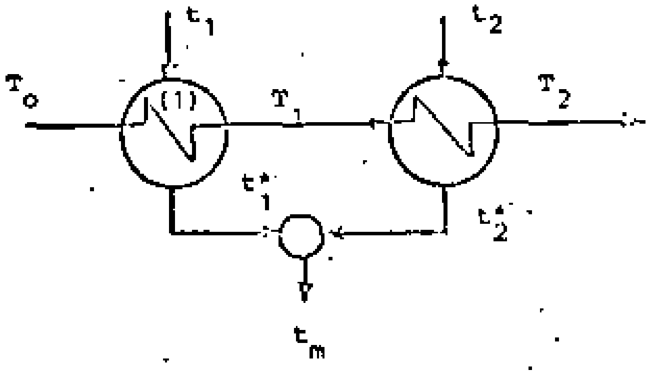
$$0 \leq Q_2 \leq WC \, (t_2 - T_o)$$

Restricciones sobre las variables dependientes

$$0 \leq Q_1 + Q_2 \leq WC \, (t_2 - T_o)$$

2- Con restricciones en la corriente de mezcla (Parte).

Las ecuaciones y restricciones analizadas en la parte 1 siguen siendo válidas aunque en este caso existe la necesidad de añadir otra restricción, según se analiza a continuación



Variable dependiente : $t_m^* = \dfrac{t_1^* + t_2^*}{2}$

restricciones sobre t_m :

Si $A_1 = \infty$ y $A_2 = \infty$ $\Longrightarrow$ $t_1^* = T_o$, $T_1 = t_1$

$t_2^* = T_1$, $T_2 = t_2$

entonces $\frac{t_1 + T_o}{2} \leq t_m^*$

y por lo tanto $\frac{t_1 + T_o}{2} \leq t_m^* \leq 230$ (restriccioón del problema)

(restricción por ar~~c~~ea infinita en A_1 y A_2)

Las soluciones óptimas resultan ser :

<u>Primera parte</u> Sin restricción en t_m^*

$T_1 = 186.2$, $T_2 = 292.7$, $Q_1 = 8.62 * 10^6$, $Q_2 = 10.64 * 10^6$, $Q_3 = 20.73 \cdot 10^6$

~~$t_m^* = 631.8$~~ $A_2 = 1239.1$, $A_3 = 5183.8$, $A_T = 7054.8$

$A_1 = 631.8$

<u>Segunda parte</u> Con restricción en t_m^*

$T_1 = 210.0$, $T_2 = 340.1$, $Q_1 = 11.00 * 10^8$, $Q_2 = 13.01 * 10^6$, $Q_3 = 16.01 \cdot 10^6$

$t_m^* = 229.9$, $A_1 = 1017.9$, $A_2 = 2715.7$, $A_3 = 4009.3$, $A_T = 7743.0$

AMPLIACION DE UNA PLANTA QUIMICA

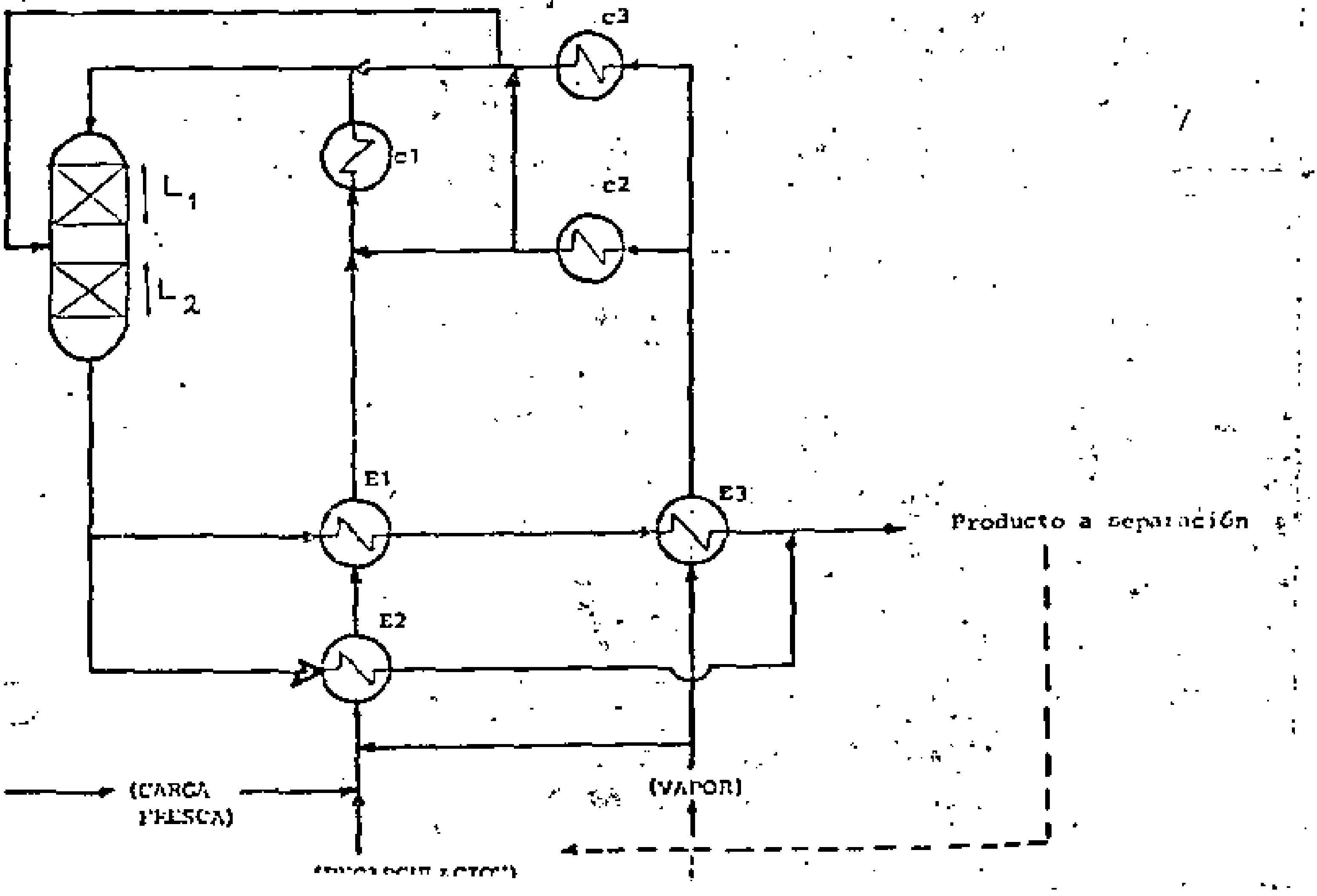
El presente estudio tiene el propósito de diseñar un nuevo reactor catalítico y fijar nuevas condiciones de operación para aumentar la capacidad de una planta que produce ES.

La planta fue originalmente diseñada para producir 91 T/D de ES, y se pretende aumentar la capacidad para producir 140 T/D. Dentro de las varias alternativas analizadas se pretende efectuar la optimización sobre el diagrama que se muestra a continuación :

C1, C2, C3 : calentadores

EI, E2, E3 : cambiadores

R : reactor catalítico



Bases de diseño

1. Composición de la carga fresca (%)

B	0.43		
T	0.86		
E B	98.46	←	(materia prima)
P E B	0.22		

2. Composición de la carga total al reactor (sin vapor)

B	0.166		
T	1.866		
E B	95.217		
E S	2.668	←	(producto final)
P E B	0.082		

Descripción del Flujo

El EB fresco se une con la corriente de recirculación proveniente de otra sección de la planta. La corriente resultante se bombea al sistema de precalentamiento de carga, constituido por los cambiadores E1 y E2 ; antes de entrar a estos cambiadores la carga combinada se mezcla con aproximadamente el 9% del vapor total usado en la reacción. La mezcla, parcialmente vaporizada, se alimenta al cambiador E2 a una temperatura de 316° F , saliendo del mismo a 692° F , completamente vaporizada. Del cambiador E2 pasa al E1 de donde sale a 1092° F . El calentamiento final de la mezcla vapor-hidrocarburos se suministra en el calentador C1, previa adición de vapor adicional proveniente del cambiador E3 y del calentador C2, de donde sale a una temperatura de 1150° F.

El 91% restante del vapor requerido por el proceso se precalienta en el cambiador E3. Este vapor entra a 366° F y sale del cambiador E3 a 748° F. El vapor así calentado, parte se alimenta al calentador C3, de donde sale a 1300° F y el resto se alimenta al C2 para más tarde mezclarse con la corriente de vapor-hidrocarburos antes de entrar al calentador C1. Por otro lado, el vapor que sale del calentador C3 se divide en dos corrientes, a saber: una parte sirve para dar la temperatura final y la relación (vapor/hidrocarburos) a la entrada del reactor R, mientras que la otra se usa para elevar la temperatura de los gases de reacción entre los lechos catalíticos del reactor.

La mezcla vapor - hidrocarburos que sale del reactor R a 1119° F se aprovecha para precalentar los hidrocarburos y el vapor que entran al proceso.

(Nota: La mayoría de los datos dados con anterioridad son resultado de la optimización que se describe más adelante).

Datos

Carga fresca : cantidad máxima disponible, composición y temperatura

[illegible]

Recirculación : Cantidad , composición y temperatura

Calentador C3 : temperatura de salida

Vapor : temperatura de entrada

ES en producto a separación : cantidad (140 T/D)

Se requiere calcular lo siguiente:

1.. Calcular el volúmen de catalizador en el lecho L 1

2.. Calcular el volúmen de catalizador en el lecho L 2

3.. La cantidad (%) del flujo de salida del reactor R que pasa por los cambiadores E 1 y E 3

4.. La cantidad total de vapor

5.. La cantidad de vapor que se inyecta al cambiador E 2

6.. La cantidad de vapor que pasa por el calentador C 2 para que la mezcla a la salida del calentador C 1 sea de 1150° F

7.. La relación (vapor/hidrocarburos) a la entrada del lecho L 1 del reactor R y a la entrada del lecho L 2 del mismo reactor.

<u>El objetivo es el siguiente</u>

1. Minimizar la cantidad total de catalizador en el reactor R (lechos L1 y L2

2. Que la producción sea ~~la más cercana a~~ de 140 T/D.

3. Que el consumo de carga fresca sea la menor posible (C)

4. Que el producto a separación esté lo más frío posible (T).

Con lo anterior se puede plantear la siguiente función objetivo :

$$F = K_1 (L_1 + L_2) + K_2 (\text{prod} - 1\overset{4\ ok}{\cancel{5}}0)^2 + K_3 (G) + K_4 (T)$$

donde K_1, K_2, K_3 y K_4 son constantes que se usan para "condicionar adecuadamente" la función objetivo.

Restricciones

Debido a condiciones de operación de los equipos, se deben tomar en cuenta las siguientes restricciones :

1000° F $\leq$ Temp. entrada al reactor $\leq$ 1280° F

Temp. salida del primer lecho (L1) $\leq$ Temp. entrada al segundo lecho (L2) $\leq$ 1250° F

1.0 $\leq$ Relación (Vap./hidrocarburos) entrada al primer lecho $\leq$ 3.0

Relación (Vap./hidrocarburos) entrada al 1er lecho $\leq$ Relación (Vap./hidrocarburos) entrada al 2ª lecho $\leq$ 3.0

Modelo de los equipos

Los equipos que es necesario simular y/o dimensionar son los siguientes :

1. Calentadores (C 1, C 2, C 3)

2. Cambiadores de calor

   a) Gas - Gas (E 1 y E 3)

   b) Gas - Liquido con evaporación (E 2)

3. Reactor químico catalítico (R)

De los equipos mencionados sólo se describirá el modelo usado para el reactor catalítico, por ser este el equipo más importante de la planta. Para el resto de los equipos los modelos matemáticos son bastante convencionales (Ver., p.ej., D.D. Kern (Process Heat Transfer", Mc Graw-Hill Book Co., New York).

Cinética del sistema reaccionante

Se considera que en los lechos catalíticos L 1 y L 2 se llevan a cabo las siguientes reacciones.. (A : vapor de agua).

1) $EB \rightleftarrows ES + H$ (catalítica)

2) $EB \longrightarrow B + E$ (catalítica)

3) $H + EB \longrightarrow T + M$ (catalítica)

4) $M + A \longrightarrow CO + H$ (catalítica)

5) $CO + A \longrightarrow CO_2 + H$ (catalítica)

6) $E \longrightarrow AC + H$ (descomposición en fase vapor)

Las expresiones para la velocidad y generación de calor de las reacciones anteriores, son las siguientes

$$V_1 = \left(P_{EB} - \frac{P_{ES}\ P_H}{K_p} \right) \exp \left(- \frac{5715}{T} - 6.16 \right)$$

$$V_2 = \exp \left(- \frac{25600}{T} + 12.8 \right) P_{EB}$$

$$V_3 = \exp \left(- \frac{11000}{T} - 1.8 \right) P_{EB}\ P_H$$

$$V_4 = \exp \left(- \frac{7900}{T} - 3.36 \right) P_M$$

$$V_5 = P \exp \left(- \frac{8850}{T} + 3.8 \right) P_{co}\ P_A$$

$$V_6 = 2.5 * 10^{11} \exp \left(- \frac{38000}{T} \right) P_E / T$$

donde:

$$K_p = T^{0.549} \exp \left(- \frac{14516}{T} + 11.41 \right)$$

P : Presión en atmósferas

T : Temperatura (°K)

P_i : Presión parcial a cada componente

El calor liberado por cada una de las reacciones se tomó igual a

$$\Delta H_1 = 28,843 + 1.09\ T$$

$$\Delta H_2 = 25,992 - 1.09\ T$$

$$\Delta H_3 = 12,702 - 3.15\ T$$

$$\Delta H_4 = 22,015 + 3.96\ T$$

$$\Delta H_5 = 10,802 + 2.5\ T$$

$$\Delta H_6 = 38,278 + 11.45\ T$$

Con la anterior información, es posible establecer las relaciones que describen la variación, con la longitud, de cada uno de los componentes así como de la presión y temperatura, las cuales tendrán la forma general

$$\frac{d\ n_i}{d\ Z} = f_i\ (n_1, n_2, \ldots, n_{NC}, T, P) \qquad i = 1,2, \ldots, N^o \text{ de componentes } (NC)$$

$$\frac{d\ T}{d\ Z} = g\ (n_1, n_2, \ldots, n_{NC}, T, P)$$

$$\frac{d\ P}{d\ Z} = h\ (n_1, n_2, \ldots, n_{NC}, T, P)$$

con sus condiciones iniciales asociadas ($Z = 0$)

Hay que tomar en cuenta que el volúmen de catalizador en cada uno de los lechos es una incognita, por lo que el límite superior de integración en cada lecho ($Z = Z_1$ y $Z = Z_2$) son variables, además de que entre ambos lechos existe una sección de mezclado con vapor.

Resultados

Los resultados que produjeron un valor mínimo de la función objetivo fueron los siguientes :

Temperatura de entrada a la primera cama	1243° F
Temperatura de entrada a la segunda cama	1145° F
Longitud de la primera cama	211.4 cm
Longitud de la segunda cama	157.1 cm
(Vapor/hidrocarburos) primera cama	2,738 lb/lb
Gasto de hidrocarburos en la primera cama	249.31 lbmol/h
(Vapor/hidrocarburos) segunda cama	2,923 lb/lb
ES a purificación	150.4 T / D
Conversión total	65.06 %
Selectividad total	81.84 %
Fracción de la salida del reactor a E1 y E3	64 %
Temperatura del producto a separación	562° F
Fracción del vapor total al cambiador E 2	8.3 %
Fracción del vapor total al calentador C 3	0 %

Comentarios Finales

El problema anteriormente descrito resulta ser, desde el punto de vista de minimización de funciones, uno de los más complejos ya que una sola evaluación de la función objetivo requiere de los siguientes cálculos

a) Solución de sistemas de ecuaciones diferenciales ordinarias no lineales

b) Cálculos iterativos en los cambiadores de calor ya que estos existen de antemano y los flujos y temperaturas de operación deben ajustarse al diseño mecánico que tienen.

c) Existe dentro del proceso un paso de recuperación de energía (cambiadores de calor), lo cual también genera que se hagan cálculos iterativos dentro de cada evaluación de la función objetiva.

En opinión del autor de este ejemplo, el modificar las condiciones de operación de una planta para ajustarlas a nuevos requerimientos es, con mucho, bastante más complejo que el diseño de una nueva planta ya que en este último caso se tienen muchos grados de libertad.