PENALIZAÇÃO E LAGRANGEANO AUMENTADO

HERMÍNIO SIMÕES GOMES

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

UNIVERSIDADE ESTADUAL DE CAMPINAS

INSTITUTO DE MATEMÁTICA, ESTATÍSTICA E CIÊNCIA DA COMPUTAÇÃO

CAMPINAS - SÃO PAULO
BRASIL

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

COORDENAÇÃO DOS CURSOS DE PÓS-GRADUAÇÃO

AUTORIZAÇÃO PARA QUE A UNICAMP POSSA FORNECER, A PREÇO DE CUSTO, CÓPIAS DA TESE A INTERESSADOS

Nome do Aluno: HERMINIO SIMÕES GOMES
Nº de Identificação: 795129
Endereço para Correspondência: AV. MAGALHÃES BARATA nº 1050 apto 201-A 66000-BELÉM-PARÁ
Curso: MATEMATICA APLICADA
Nome do Orientador: JOSE MARIO MARTÍNEZ P.
Título da Dissertação ou Tese: PENALIDADE E LAGRANGEANO AUMENTADO

Data proposta para a Defesa: 03 de agosto de 1981

(O Aluno deverá assinar um dos 3 ítens abaixo)

1) Autorizo a Universidade Estadual de Campinas a partir desta data, a fornecer, a preço de custo, cópias de minha Dissertação ou Tese a interessados.

21/07/1981 — Herminio Simões Gomes

Data — assinatura do aluno

2) Autorizo a Universidade Estadual de Campinas, a fornecer, a partir de dois anos após esta data, a preço de custo, cópias de minha Dissertação ou Tese a interessados.

/ /

Data — assinatura do aluno

3) Solicito que a Universidade Estadual de Campinas me consulte, dois anos após esta data, quanto à minha autorização para o fornecimento de cópias de minha Dissertação ou Tese, a preço de custo, a interessados.

/ /

Data

DE ACORDO

Orientador

assinatura do aluno

PENALIZAÇÃO E LAGRANGEANO AUMENTADO

HERMÍNIO SIMÕES GOMES

Orientador

Prof. Dr. José Mário Martínez

Dissertação apresentada no Instituto de Matemática, Estatística e Ciência da Computação, UNICAMP, como requisito parcial para obtenção do título de Mestre em Matemática Aplicada.

Campinas, Julho, 1981.

A meu pai (falecido) e

a minha mãe.

AGRADECIMENTOS

Ao Martínez, pela orientação, pela amizade e pelo apoio sem o qual seria impossível compor este trabalho.

A minha esposa, pelo seu grande apoio e companheirismo.

A meus irmãos, particularmente ao Hélio, que me ajudou enormemente durante o transcorrer do curso de mestrado.

A meu tio Preto, que sempre me incentivou.

A Silvia, pelo trabalho de datilografia e pela grande amizade.

A todos os colegas da UNICAMP que de alguma forma me ajudaram neste trabalho.

A CAPES/PICD pelo auxílio financeiro.

SUMÁRIO

INTRODUÇÃO

Na resolução de problemas de programação não-linear, a dimensão do problema ou sua estrutura pode, em certos casos, se constituir em um obstáculo difícil de ser transposto. Muitos algoritmos foram desenvolvidos para programação não-linear e, na sua maioria, procuram usar a informação das derivadas segundas (matrizes hessianas) , e devido a isso, as suas aplicabilidades são, em geral, limitadas a problemas de médio porte. Devido a problema de memória ou dificuldade de se codificar os hessianos, a necessidade de se desenvolver algoritmos que não usam matrizes se evidenciou. Certos problemas não apresentam estruturas particulares que possam ser aproveitadas para facilitarem as suas resoluções, ou,se apresentam, estas são difíceis ou até impossíveis de serem aproveitadas.

Algoritmos que usam basicamente a informação de derivadas primeiras(gradientes), são, em geral, mais fáceis de serem implementados e reduzem consideravelmente a necessidade de memória.

Os algoritmos que usam Penalização e Lagrangeano Aumentado se enquadram dentro das necessidades expostas.

Em problemas de grande porte, o lagrangeano aumentado,é particularmente bom, e pode se empregado com êxito em problemas que não necessitam de uma precisão elevada.

Nosso trabalho se propõe a fornecer uma base sobre a teoria da função Penalidade e Lagrangeano Aumentado, e também, poder fornecer a interessados, programas com os métodos e suas aplicações.

Uma comparação entre os métodos Penalidade e Lagrangeano Aumentado é fornecida, e diversos aspectos sobre a aplicação dos algoritmos são evidenciados. Um número satisfatório de problemas testes mostra o desempenho dos algoritmos. Uma aplicação a um problema de grande porte é feita utilizando o lagrangeano aumentado.

Não nos detivemos nos diversos tipos de funções Penalidade ou funções aumentadas; tratamos das formas que achamos mais importantes.

CAPÍTULO I

MÉTODOS DE PENALIDADE

1.1 - INTRODUÇÃO - Neste capítulo veremos como tratar o problema (P) dado a seguir,

$$\begin{array}{lll} \text{Minimizar } f(x) & f: R^n \to R & \\ \text{s/a} \quad x \in S & S \subseteq R^n & \end{array} \qquad (P)$$

utilizando uma classe de métodos conhecidos como MÉTODOS DE PENALIDADE.

No problema (P), o conjunto S é, geralmente, um conjunto de restrições funcionais, podendo incluir restrições de igualdades e/ou desigualdades. Algumas considerações devem ser feitas sobre o problema (P); em princípio a função f deve ser contínua. Veremos mais adiante que para a implementação do método por computador necessitaremos que f e as restrições funcionais de S sejam de classe C^1.

A idéia geral dos métodos é modificar o problema (P) para uma classe de problemas sem restrições, e resolver uma sequência de problemas sem restrições, o que é, em geral, mais fácil de ser programado para o computador.

1.2 - DEFINIÇÃO - Definimos FUNÇÃO PENALIDADE OU FUNÇÃO DE PENALIZAÇÃO associada ao problema (P), como qualquer função da seguinte forma [1]

---

(1) Alguns autores adotam o termo penalidade ou função de penalidade só para a função P(x), veja Avriel [2]

$$q(\mu,x) = f(x) + \mu P(x)$$

onde μ é uma constante(2) positiva que chamamos parâmetro de penalização, e $P(x)$ é uma função que satisfaz os seguintes axiomas:

(i) P é contínua em R^n

(ii) $P(x) \geq 0$ para todo $x \in R^n$

(iii) $P(x) = 0$ se e só se $x \in S$

Sendo assim, há uma infinidade de funções do tipo $P(x)$.

Por exemplo, se o conjunto S fosse definido por um conjunto de restrições de igualdade,

$$S = \{x \mid h_i(x) = 0 \quad , i=1,2,\ldots \ell\} \quad , h_i(x): \; R^n \to R$$

poderíamos ter para uma função $P(x)$ o seguinte:

$$P(x) = \sum_{i=1}^{\ell} h_i(x)^2 \tag{1.2.1}$$

Se, por outro lado, S fosse definido por um conjunto de restrições de desigualdade,

$$S = \{x \mid g_i(x) \leq 0 \quad , i=1,2,\ldots \ell\} \quad , g_i(x): \; R^n \to R$$

poderíamos ter para uma função $P(x)$ o que se segue:

$$P(x) = \sum_{i=1}^{m} \{\max\, [0, g_i(x)]\}^2 \tag{1.2.2}$$

---

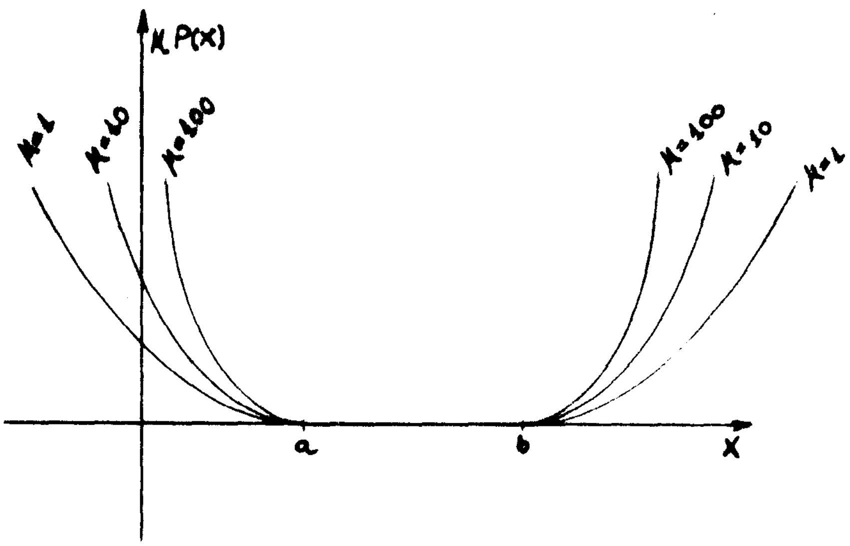
(2) Ao tratarmos com uma generalização de $q(\mu,x)$, veremos que μ será um vetor

Nota-se que em qualquer dos casos, os axiomas para $P(x)$ são satisfeitos.

Obviamente, se S contiver os dois tipos de restrições, $P(x)$ conterá parcelas do tipo (1.2.1) e parcelas do tipo (1.2.2)

1.3 - <u>O MÉTODO DE PENALIDADE</u> - Antes de introduzirmos o método, consideremos dois exemplos a seguir:

<u>Exemplo 1</u> - Consideremos o caso unidimensional em que $S = \{x \mid g_1(x) = x - b \leq 0,\ g_2(x) = a - x \leq 0\}$, então a parcela $\mu P(x)$ da função $q(\mu,x)$ apresenta o seguinte aspecto:



Com o aumento de μ, o mínimo de uma função $Q(\mu,x)$ se situará em uma região onde P é pequeno. Sendo assim, o problema sem restrição a seguir:

$$\text{Minimizar} \quad q(\mu,x) = f(x) + \mu P(x) \qquad (1.3.1)$$

tem o mínimo "próximo" de um mínimo do problema (P), desde que μ seja grande.

<u>Exemplo 2</u> - Aqui mostramos um exemplo numérico, bem simples, para que se tenha noção do que ocorre quando μ cresce.

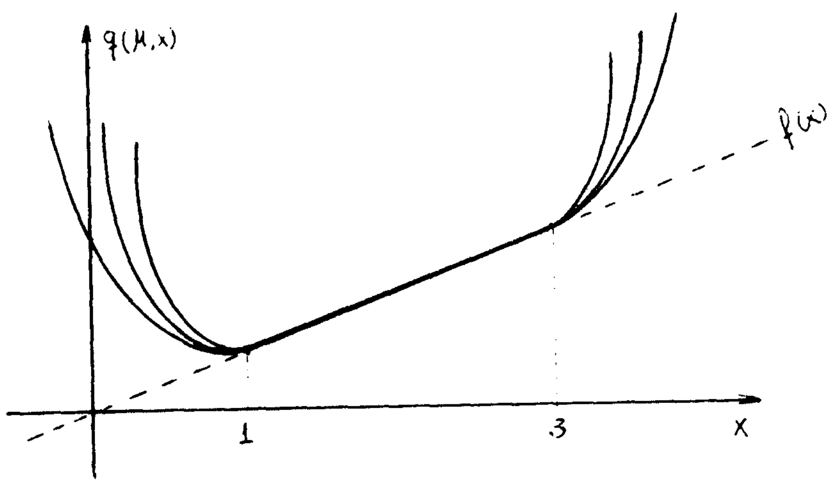
Minimizar $x/2$

s/a $\quad g_1(x) = x - 3 \leq 0 \;, \quad g_2(x) = 1 - x \leq 0$

Assim,

$$q(\mu,x) = 2x + \mu\{\max[0,x-3]\}^2 + \mu\{\max[o,1-x]\}^2$$

O gráfico para $q(\mu,x)$ para diversos valores de μ se apresenta como segue:



Nota-se que quando μ cresce, o mínimo de $q(\mu,x)$ tende ao mínimo da função f sujeita às restrições. Os diferentes valores de μ geram diferentes mínimos para q, neste exemplo o mínimo de q é da do por $x = 1 - \frac{1}{2\mu}$. É imediato que $\lim_{\mu\to\infty} x = x^* = 1$, como era de se esperar.

Como vemos, minimizar q para μ grande nos conduz a um ponto x próximo ao ponto ótimo do problema (P). Porém, querer se aproximar do ideal, minimizando q com um valor muito grande de μ, conduz em geral a erros devido a problemas numéricos e devido a própria expressão da função q.

O recomendável é, portanto, fazer uma SEQUÊNCIA DE MINIMIZAÇÕES **para crescentes** valores de μ. Fazendo-se assim, a possibilidade de se cometer erros é menor, pois cada nova minimização tem como ponto inicial o ponto que foi obtido na minimização anterior . A precisão é conseguida, portanto, com o preço da sequência de minimizações.

Seja $\{\mu_k\}$, k=1,2,..., uma sequência tendendo a infinito com $\mu_k > 0$ e $\mu_{k+1} > \mu_k$ para todo k, o método de penalidade pode ser resumindo como se segue:

1) selecione um $\mu = \mu_k > 0$, $k = 0$
2) minimize $q(\mu_k, x) = f(x) + \mu_k P(x)$
   obtendo x_k como ponto ótimo
3) teste a convergência, se convergir pare.
4) selecione um novo $\mu = \mu_{k+1}$ maior que o μ_k anterior
5) $k \leftarrow k+1$ tome x_k como novo ponto inicial
   e vá para 2

O número de minimizações necessárias, depende da sequência $\{\mu_k\}$, do critério de convergência e, é lógico, do problema em si. No capítulo III trataremos com mais detalhes deste algoritmo.

A seguir veremos um lema para chegarmos ao Teorema de Convergência do método.

1.4 LEMA 1 - Valem as desigualdade abaixo, para μ_k ,k=1,2... como definido acima

a) $q(\mu_k, x_k) \leq q(\mu_{k+1}, x_{k+1})$

b) $P(x_k) \geq P(x_{k+1})$

c) $f(x_k) \leq f(x_{k+1})$

PROVA:

a) $q(\mu_{k+1}, x_{k+1}) = f(x_{k+1}) + \mu_{k+1}P(x_{k+1}) \geq$

$\geq f(x_{k+1}) + \mu_k P(x_{k+1}) \geq f(x_k) + \mu_k P(x_k) = q(\mu_k, x_k)$

b) Como $f(x_k) + \mu_k P(x_k) \leq f(x_{k+1}) + \mu_k P(x_{k+1})$ e

$f(x_{k+1}) + \mu_{k+1}P(x_{k+1}) \leq f(x_k) + \mu_{k+1}P(x_k)$ temos,

somando as duas expressões acima:

$$\mu_k P(x_k) + \mu_{k+1}P(x_{k+1}) \leq \mu_k P(x_{k+1}) + \mu_{k+1}P(x_k)$$

ou

$$\mu_{k+1} [P(x_{k+1}) - P(x_k)] \leq \mu_k [P(x_{k+1}) - P(x_k)]$$

sendo necessário que $P(x_{k+1}) - P(x_k)$ seja menor ou igual a zero , para não contradizer a hipótese $\mu_{k+1} > \mu_k$. Assim, fica demonstrado o ítem (b).

c) Decorrente de (a), temos:

$$f(x_{k+1}) + \mu_k P(x_{k+1}) \geq f(x_k) + \mu_k P(x_k)$$

ou

$$f(x_{k+1}) - f(x_k) \geq \mu_k [P(x_k) - P(x_{k+1})] \geq 0$$

Logo

$$f(x_{k+1}) \geq f(x_k).$$

1.5 LEMA 2 - Se x^* é solução do probelam (P), então

$$f(x^*) \geq q(\mu_k, x_k) \geq f(x_k) \quad \text{para todo } k.$$

PROVA:

$$f(x^*) = f(x^*) + \mu_k P(x^*) \geq f(x_k) + \mu_k P(x_k) \geq f(x_k).$$

1.6 - TEOREMA DE CONVERGÊNCIA - Seja $\{x_k\}$ uma sequência gerada pelo método de penalidade, como descrito em 1.3, então, qualquer ponto limite da sequência é uma solução para o problema (P).

DEMONSTRAÇÃO: Tomemos uma subsequência convergente $\{x_k\}$ da sequência acima, com $k \in K$, tendo como limite $\bar{x}$. Pela continuidade de f, temos:

$$\lim_{k \in K} f(x_k) = f(\bar{x}) \tag{1.6.1}$$

Seja f^* o valor ótimo associado ao problema (P). Então, de acordo com os lemas 1 e 2, a sequência de valores $q(\mu_k, x_k)$ é não-decrescente e limitada superiormente por f*. Então

$$\lim_{k \in K} q(\varepsilon_k, x_k) = q^* \leq f^* \qquad (1.6.2)$$

Subtraindo (1.6.1) de (1.6.2), temos

$$\lim \mu_k P(x_k) = q^* \leq f(\bar{x}) \qquad (1.6.3)$$

Como $P(x_k) \geq 0$ e μ_k tende a infinito, implica que

$$\lim P(x_k) = 0$$

Já que P é contínua, $P(\bar{x}) = 0$, sendo assim $\bar{x}$ é factível. Pelo Lema 2, $f(x_k) \leq f^*$, assim

$$\lim_{k \in K} f(x_k) = f(\bar{x}) \leq f^*$$

Como $\bar{x}$ é factível e $f(\bar{x})$ é um limitante inferior, vem que

$$f(\bar{x}) = f^*$$

1.7 - <u>COROLÁRIO</u> - A sequência $\{f(x_k)\}$ tende a f* por valores inferiores a f*. Ou seja, o método de penalidade gera uma sequência de pontos não-factíveis ao conjunto de restrição S.

1.8 - <u>CONSIDERAÇÕES SOBRE PENALIZAÇÃO</u> - Transformar um problema com restrições em um problema sem restrições, ganha-se por um lado e perde-se por outro. Ganha-se pela facilidade de programar o método, já que este necessita, primordialmente, de um algoritmo para minimização sem restrição; perde-se devido a problemas numéricos inerentes ao próprio método, e que não podem ser evitados. O principal problema numérico reside no fato de que se necessita minimizar funções $q(\mu,x)$ com parâmetros μ cada vez maiores.

Quando minimizamos $q(\mu,x)$ e não alcançamos a tolerância necessária, <u>penalizamos</u> todas as restrições (ou seja, aumentamos o parâmetro μ), e continuamos o processo. Imediatamente aparece a questão: Por que penalizar todas as restrições? Respondemos: Não é necessário penalizarmos todas as restrições; e mesmo aquelas que penalizamos, não necessitamos penalizá-las de uma mesma maneira. Por isso generalizamos a função penalidade, e com isso aprimoramos o método.

1.9 - <u>GENERALIZAÇÃO DA FUNÇÃO PENALIDADE</u> - Se a cada restrição associarmos um parâmetro de penalização, a função $q(\mu,x)$ para o problema (P) se apresenta como:

$$Q(\mu,x) = f(x) + \sum_{i=1}^{m} \mu_i\{\max[0,g_i(x)]\}^2 + \sum_{j=m+1}^{m+\ell} \mu_j[h_j(x)]^2 \qquad (1.9.1)$$

Assim, poderíamos tratar μ como um vetor de $(m+\ell)$ componentes. Fato relevante é que com a função (1.9.1) podemos fazer crescer os parâmetros de cada uma das restrições a "diferentes velocidades".

As mesmas considerações sobre a convergência, utilizando a função $q(\mu,x)$, valem para $Q(\mu,x)$.

A função $Q(\mu,x)$ em (1.2.1), foi utilizada para implementação para o computador de um método de penalidade. Veremos mais adiante o algoritmo utilizado, e detalhes sobre o seu uso.

CAPÍTULO II

MÉTODOS QUE USAM O LAGRANGEANO AUMENTADO

2.1 - INTRODUÇÃO - Assim como no caso da penalização, veremos neste capítulo como tratar o problema (P) utilizando uma técnica semelhante à penalização com algumas modificações.

Na verdade, os métodos que usam lagrangeano aumentado usam a penalização e mais alguma coisa, isto é, em vez de usar somente o termo quadrático em $Q(\mu,x)$, acrescenta-se um termo linear. Sob outro ponto de vista, o lagrangeano aumentado, como o próprio nome diz, é proveniente da função lagrangeano que foi aumentada pela inclusão do termo quadrático.

O primordial sobre os métodos que usam lagrangeano aumentado é que estes estão "qualificados" para resolver problemas não-convexos, enquanto que o lagrangeano simples não é eficaz para estes mesmos tipos de problemas.

Além do mais, os métodos do tipo lagrangeano aumentado, em geral, apresentam menos problemas de instabilidade numérica que os métodos de penalidade.

Ressaltamos resultados obtidos por Powell, Fletcher e Rockafellar, tão importantes para a nossa teoria como para a confecção dos algoritmos.

2.2 - O LAGRANGEANO AUMENTADO DE POWELL-HESTENES - Para que se tenha uma boa noção sobre o porquê do uso do lagrangeano aumentado, temos que recapitular alguns itens obtidos por Powell, Hestenes e Fletcher, até chegar ao lagrangeano aumentado de Rockafellar.

Para resolver o problema (P2) abaixo,

$$\begin{array}{llll} \text{Minimizar } f(x) & & f: R^n \to R & h: R^n \to R^\ell \\ \text{s/a} & h_i(x) = 0 & i= 1,2,\dots,\ell & \end{array} \qquad \text{(P2)}$$

com f e h de classe C^1, no intuito de se achar um número local x^*, Powell [11] (1969) sugeriu uma função penalidade do tipo

$$\phi(x,\theta,\tau)= f(x) + \frac{1}{2}(h(x)-\theta)^t M \; (h(x)-\theta)$$

$$= f(x) + \frac{1}{2} \sum_{i=1}^{\ell} \tau_i (h_i(x) - \theta_i)^2 \qquad (2.2.1)$$

onde θ é um vetor de R^ℓ e M é uma matriz diagonal com elementos $\tau_i > 0$. Como se ve, τ é o vetor de parâmetros de penalização na função ϕ, $\tau = (\tau_1,\dots\tau_\ell)$.

O uso da função ϕ (x,θ,τ) pode ser descrito da seguinte maneira: Dados os parâmetros θ e τ, obtém-se o ponto $x(\theta,\tau)$ por minimização de ϕ (x,θ,τ) na variável x; é então feita uma modificação em θ e τ, de tal maneira que, com sucessivas minimizações, $x(\theta,\tau)$ tenda a x^*.

Quando $\theta = 0$, temos a função penalidade clássica de Powell, é a mesma que vimos no capítulo anterior. Mas se θ=0 a convergência é assegurada quando τ_i tende a infinito, para $i=1,\dots\ell$. Mas o que se desejava era justamente obter a convergência sem fazer τ_i tender a infinito; ou seja, era desejável obter convergência sem que τ_i atingisse valores elevados, caso contrário, certamente, ocorreriam dificuldades numéricas.

A idéia de Powell era fazer uma iteração com θ, ou seja, uma modificação adequada, para se obter convergência sem a necessidade de τ_i tender a infinito. Um meio era tentar fazer θ tender a um vetor ótimo θ^*, mantendo M, a matriz dos parâmetros τ_i, constante. Um problema a ser resolvido residia no fato de escolher um bom vetor τ para começar, já que se τ fosse pequeno demais o mínimo de $\phi(x,\theta,\tau)$, independente de qualquer θ, não seria o mínimo do problema (P2); E o mais importante é que existe um τ, suficientemente grande, de tal modo que, para um vetor θ adequado, digamos θ^*, o mínimo de $\phi(x,\theta^*,\tau)$ é igual ao mínimo do problema (P2); esta afirmação é assegurada por um teorema devido a R. Fletcher que veremos adiante.

No caso de se ter um τ suficientemente grande, a idéia para se trabalhar com $\phi(x,\theta,\tau)$ consistia em modificar θ, fazendo-o tender a um θ^* ótimo; as componentes deste vetor iriam satisfazer as igualdades abaixo:

$$-\theta_i^*\tau_i = \lambda_i^* \qquad i = 1,2,\ldots\ell \qquad (2.2.2)$$

onde λ^* é o vetor de multiplicadores de Lagrange para a solução x^* do problema (P2):

As igualdade (2.2.2) se impõem do seguinte modo:

$$\phi(x,\theta,\tau) = f(x) + \frac{1}{2} \sum_{i=1}^{\ell} \tau_i (h_i(x) - \theta_i)^2$$

$$= f(x) + \frac{1}{2} \sum_{i=1}^{\ell} \tau_i \left[h_i(x)^2 - 2h_i(x)\theta_i + \theta_i^2\right]$$

$$= f(x) + \sum_{i=1}^{\ell} \frac{\tau_i}{2} h_i(x)^2 - \sum_{i=1}^{\ell} \tau_i h_i(x)\theta_i +$$

$$+ \sum_{i=1}^{\ell} \frac{\tau_i \theta_i}{2}$$

e

$$\nabla_x \phi(x,\theta,\tau) = \nabla f(x) - \sum_{i=1}^{\ell} \tau_i h_i(x) \nabla h_i(x) + \sum_{i=1}^{\ell} \tau_i \theta_i \nabla h_i(x)$$

No ponto x^* teremos $\nabla_x \phi(x^*,\theta^*,\tau) = 0$ e $h_i(x^*) = 0$, logo

$$\nabla f(x^*) - \sum_{i=1}^{\ell} \tau_i \theta_i \nabla h_i(x^*) = 0$$

e por definição dos multiplicadores de Lagrange chegamos à igualdades (2.2.2).

Veremos adiante como fazer a iteração para conseguir fazer θ tender a θ^*.

Pela mesma época dos estudos de Powell, e independentemente, Hestenes [7] (1969), levou avante seus estudos sobre o <u>método dos multiplicadores</u>. No seu trabalho ele sugeriu a seguinte função **aumentada** .

$$\psi(x,\lambda,\tau) = f(x) + \lambda^T h(x) + \frac{1}{2}\, h(x)^t\, M\, h(x) \qquad (2.2.3)$$

$$= f(x) + \sum_{i=1}^{\ell} \lambda_i h_i(x) + \frac{1}{2} \sum_{i=1}^{\ell} \tau_i h_i(x)^2$$

onde $\lambda \in R^{\ell}$ e M é como definida em (2.2.1) (3)

Se considerarmos as igualdades $-\theta_i \tau_i = \lambda_i$, para i=1,2,.. ℓ em (2.2.1), teremos a seguinte relação para as funções de Powell e Hestenes:

$$\phi(x,\theta,\tau) = \psi(x,\lambda,\tau) + \frac{1}{2} \sum_{i=1}^{\ell} \lambda_i^2/\tau_i \qquad (2.2.4)$$

Ve-se, de imediato, que a diferença entre ϕ e ψ independe de x, logo, uma minimização de ϕ ou ψ com relação em x, conduz ao mesmo ponto ou seja, $x(\lambda,\tau) = x(\theta,\tau)$ para qualquer τ. Embora os valores das funções $\phi(x(\theta,\tau),\theta,\tau)$ e $\psi(x(\lambda,\tau),\lambda,\tau)$ sejam diferentes, isso não importa.

Assim, as idéias de Powell e Hestenes eram fudamentalmente as mesmas.

Fletcher [4], após estudar os trabalhos de Powell e Hestenes, tentou modificar a função penalidade de Powell(2.2.1) para resolver o problema geral, incluindo desigualdades.

(3) Aqui tratamos de colocar as mesmas notações para os dois casos, e fizemos a necessária modificação de sinal dos multiplicadores para chegar ao nosso objetivo. Na verdade, as notações de Powell e Hestenes diferem substancialmente.

Para o problema

$$\begin{array}{lll} \text{Minimizar} & f(x) & \\ \text{s/a} & g_i(x) \leq 0 \quad , \; i = 1,2,\ldots,m & (2.2.5) \end{array}$$

com f e g de classe C^1, a função proposta por Fletcher era:

$$\Phi(x,\theta,\tau) = f(x) + \frac{1}{2}\sum_{i=1}^{m} \tau_i \left[(\theta_i - g_i(x))_- \right]^2 \qquad (2.2.6)$$

onde a função a_- é definida por:

$$a_- = \min(a,0) = \begin{cases} a & \text{se} \quad a \leq 0 \\ 0 & \text{se} \quad a \geq 0 \end{cases}$$

De fato, uma generalização do tratamento do problema com igualdades para um problema que contenha igualdades e/ou desigualdades é relativamente fácil, mesmo assim a função Φ proposta por Fletcher não era muito boa pois apresentava descontinuidades na derivada primeira e, mais ainda, para pontos próximos ao ótimo x^* as derivadas segundas apresentam descontinuidades não removíveis, tendo seus valores tendendo a infinito.

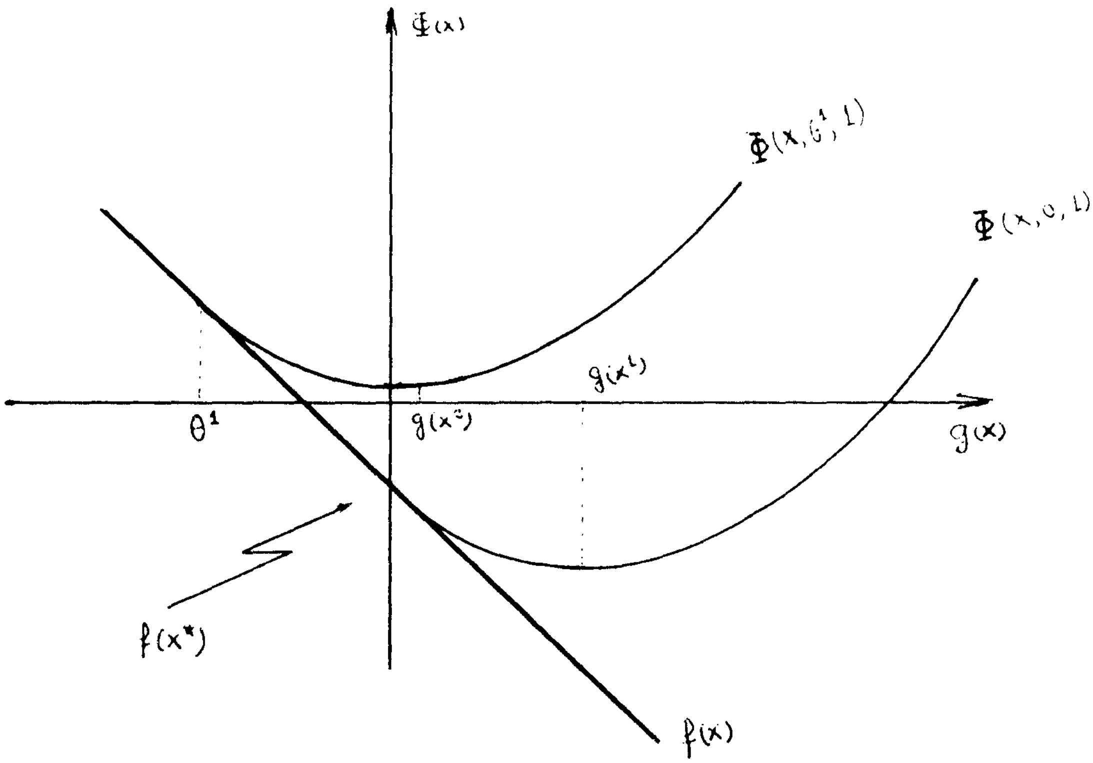
A maneira de se fazer a iteração com θ, fazendo θ tender a θ^*, se assemelha a um método dual simples em que λ , o multiplicador, é atualizado na direção da restrição. Vendo a fundo, as atualizações de θ ou λ no lagrangeano aumentado são iguais as feitas no dual simples, com a única diferença em que τ é variável, como teremos oportunidade de ver nos algoritmos em outro capítulo. Para exemplificar a iteração com θ, vejamos um pequeno exemplo usando a função proposta por Fletcher.

EXEMPLO

Minimizar $f(x)$

s/a $\quad g(x) \leq 0$

com $x \in R$ e $f(x)$, para simplificar, uma reta;



os valores de $g(x)$ (veja figura) são dados no eixo horizontal, os de Φ são no eixo vertical. Suponhamos que para uma escolha ini_ cial tomemos $\theta = 0$ e $\tau = 1$, supondo que este τ seja suficientemente grande, então a penalidade só é eficaz para $g(x) > 0$ (é claro, se $g(x) \leq 0$, $\Phi = f(x)$), e a curva $\Phi(x,0,1)$ tem mínimo em x^1, com $g(x^1) > 0$, ou seja, não factível. Fazendo a iteração com θ, como sugerido por Powell e Hestenes,

$$\theta^1 = \theta - g(x^1)$$

obtemos um novo θ^1, cujo mínimo de $\Phi(x,\theta^1,1)$ já está mais próximo do ótimo.

Voltando a falar sobre o problema das descontinuidades , se assumirmos que $f(x)$ e $g_i(x)$ $i = 1,2,...,m$ são de classe C^2, então nestas circunstâncias, Φ como função de x, será duas vezes continuamente diferenciável, excetuando-se os pontos em que $g_i(x) = \theta_i$, onde a segunda derivada tem uma descontinuidade de primeira espécie. Embora esta descontinuidade seja limitada sobre o conjunto S das restrições é, em geral, longe do mínimo; não afetando a convergência no processo de minimização.

2.3 - O LAGRANGEANO AUMENTADO DE ROCKAFELLAR - Devido a dificuldades numéricas, a função de Powell/Hestenes não se enquadra bem nos algoritmos de minimização, já que estes necessitam em geral de uma função suave, que não possua as descontinuidades que apresentam as funções de Powell/Hestenes. Expondo melhor, se utilizarmos um algoritmo do tipo "Gradientes Conjugados", este necessitará das derivadas primeiras contínuas; se utilizarmos um algoritmo ti po Newton, este necessitará de derivadas segundas contínuas; sendo portanto não recomendável a aplicação das funções de Powell/Hestenes para algoritmos desse tipo.

R.T. Rockafellar [12]propôs uma melhora na função de Hestenes que não apresenta o problema de descontinuidades que apresentam as anteriores. A função aumentada de Rockafellar tem derivada primeira contínua e apresenta como vantagem poder ser tratada por algoritmos de minimização que usam derivadas segundas (4), como por exemplo, os tipo Newton, e não somente os que usam primeiras derivadas como os do tipo gradientes-conjugados.

Considerando-se o problema (2.2.5) a função lagrangeano aumentado de Rockafellar se apresenta como:

$$\Psi(x,\lambda,\tau) = f(x) + \begin{cases} \displaystyle\sum_{i=1}^{m} \lambda_i g_i(x) + \frac{1}{2}\sum_{i=1}^{m} \tau_i g_i(x)^2 & \text{se } g_i(x) \geq \dfrac{-\lambda_i}{\tau_i} \\ \displaystyle -\frac{1}{2}\sum_{i=1}^{m} \frac{\lambda_i^2}{\tau_i} & \text{se } g_i(x) \leq \dfrac{-\lambda_i}{\tau_i} \end{cases} \tag{2.3.1}$$

Ve-se de imediato que $\Psi(x,\lambda,\tau)$ é contínua, e também apresenta derivadas primeiras contínuas, sendo portanto uma boa função para fins computacionais.

No caso de o problema apresentar restrições de desigual-

(4) Assumindo-se que f, g e/ou h sejam duas vezes diferenciáveis.

dade e restrições de igualdade a generalização é imediata, e a função se apresentaria como:

$$\Psi(x,\lambda,\rho,\tau,\mu) = f(x) + \begin{cases} \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{\ell} \rho_i h_i(x) + \frac{1}{2} \sum_{i=1}^{m} \tau_i g_i(x)^2 + \\ + \frac{1}{2} \sum_{i=1}^{m} \mu_i h_i(x)^2 \qquad \text{se} \quad g_i(x) \geq -\frac{\lambda_i}{\tau_i} \\ \\ -\frac{1}{2} \sum_{i=1}^{m} \frac{\lambda_i^2}{\tau_i} + \sum_{i=1}^{\ell} \rho_i h_i(x) + \frac{1}{2} \sum_{i=1}^{\ell} \mu_i h_i(x)^2 \\ \qquad \text{se} \quad g_i(x) \leq -\frac{\lambda_i}{\tau_i} \end{cases} \tag{2.3.2}$$

As funções $\Psi(x,\lambda,\tau)$ e $\Phi(x,\theta,\tau)$ apresentam a mesma relação (2.2.4) que apresentam as funções $\phi(x,\theta,\tau)$ e $\psi(x,\lambda,\tau)$.

A seguir, veremos um teorema devido a Fletcher, sobre um fato muito importante, tanto do lado teórico como do lado prático (computacional). O teorema aborda o fato de não ser necessário que o parâmetro τ em $\phi(x,\lambda,\tau)$, $\Phi(x,\lambda,\tau)$, $\psi(x,\lambda,\tau)$, ou $\Psi(x,\lambda,\tau)$ tenda a infinito para assegurar a convergência. Para simplificar, usaremos somente a função $\phi(x,\lambda,\tau)$ para a demonstração, não havendo perda de generalidade. Antes de realmente abordar o Teorema em questão, o leitor poderá ver a definição de função lagrangeano e os resultados (teoremas) sobre as condições de otimalidade no apêndice A.

2.4- TEOREMA(Fletcher [4])- Se as condições de segunda ordem (2.4.3) abaixo, são satisfeitas, então existe uma matriz M' diagonal, de elementos τ_i', i= 1,...,m tal que para toda matriz M diagonal de elementos τ_i, i= 1,...,m com $M \geq M'$ (isto é, $\tau_i \geq \tau_i'$), x^* é um mínimo local restrito de $\phi(x,\theta^*,\tau)$ e $\psi(x,\lambda^*,\tau)$. (Como já dissemos, o resultado é valido para as funções Φ e Ψ)

DEMONSTRAÇÃO - Devemos mostrar em primeiro plano, que as condições necessárias $h_i(x^*) = 0$ e $\nabla f(x^*) + \sum_{i=1}^{\ell} \theta_i \nabla h_i(x^*) = 0$ para o problema(P2), imlpicam que $\nabla\phi(x^*,\theta^*,\tau) = 0$, que são as condições necessárias para que x^* seja um mínimo local de ϕ. Este resultado resulta diretamente da expressão do gradiente de ϕ.

$$\nabla_x \phi(x,\theta,\tau) = \nabla f(x) + \sum_{i=1}^{\ell} -\tau_i (h_i(x) - \theta_i) \nabla h_i(x) \qquad (2.4.1)$$

$$\nabla_x \phi(x^*,\theta,\tau) = \nabla f(x^*) + \sum_{i=1}^{\ell} \tau_i \theta_i \nabla h_i(x^*) = 0$$

Assim fica demonstrada a primeira parte, para a segunda parte temos que mostrar a suficiência, ou seja, temos que mostrar que $\nabla^2\phi(x^*,\theta^*,\tau)$ é positiva definida.

De (2.4.1) tiramos

$$\nabla^2\phi(x^*,\theta^*,\tau) = L^* + N^* M N^{*T}$$

onde $N^* = N(x^*) = |\nabla h_1(x^*), \nabla h_2(x^*), \ldots, \nabla h_\ell(x^*)|$, devido a condição de regularidade $N(x^*)$ deve ter posto completo, e

$L^* = \nabla^2 f(x^*) + \sum_{i=1}^{\ell} \nabla^2 h_i(x^*)$ é o hessiano do lagrangeano.

Considere qualquer vetor u tal que $||u||_2 = 1$ e seja u escrito como uma componente v ortogonal às colunas de N*, mais uma componente que é uma combinação linear das colunas de N*. Isso poderá ser escrito da forma

$$u = v + (N^{*+})^T w$$

onde $N^{*+} = (N^{*T} N^*)^{-1} N^{*T}$. Então

$$u^T \nabla^2 \phi u = v^T L^* v + 2v^T L^* (N^{*+})^T w + w^T N^{*+} L^* (N^{*+})^T w + w^T M w \qquad (2.4.2)$$

Continuando, as condições suficientes para que x* seja solução do problema (P2) são que existe um a > 0 tal que

$$v^T L^* v \geq a \, ||v||_2^2 \quad \text{para todo v com } N^{*T} v = 0 \qquad (2.4.3)$$

Em outras palavras, L* deve ser positiva definida no subespaço tangente a nulidade[5] de h(x), ou seja

$$v^T L^* \, v \geq a \, ||v||_2^2 \qquad \text{para } v \in T = \{v: h'(x^*) v = 0\}$$

onde $h'(x^*) = (\nabla h_1(x)^T, \nabla h_2(x)^T, \ldots, \nabla h_\ell(x)^T)^T$.

Fazendo $||L^*(N^{*+})^T||_2 = b$ e $||N^{*+} L^*(N^{*+})^T||_2 = d$ teremos em (2.4.2) o seguinte:

---

(5) Se A é o subespaço gerado pelos gradientes de h(x) $\{\nabla h_i(x), \nabla h_2(x), \ldots, \nabla h_\ell(x)\}$ a nulidade de h(x) será $A = (A^T A)^{-1} A^T$

$$u^T\nabla^2\phi u \geq a\,||v||_2^2 - 2b||v||_2||w||_2 + (\min_i \tau_i - d)||w||_2^2$$

Considerando que u e w são não-nulos e tomando $M' = \tau' I$ (I = matriz identidade) com $\tau' > d + b^2/_a$, então para todo $M \geq M'$ segue-se que $u^T\nabla^2\phi u > 0$. Logo $\nabla^2\phi$ será positiva definida. O mesmo resultado é válido para $\psi(x,\lambda^*,\tau)$ pois a diferença entre ϕ e ψ independe de x como já vimos em (2.2.4).

2.5 - <u>ALGUNS COMENTÁRIOS IMPORTANTES</u> - Como vimos, o teorema de Fletcher é forte se o hessiano do lagrangeano for definido positivo, caso contrário um ponto x*,solução do problema original (P2),não poderá ser solução obtida por minimização da função ϕ (ou ψ). Um exemplo disto, devido a Hestenes, é o seguinte:

$$\begin{aligned} \text{Min} \quad f(x) &= x_1^4 + x_1\,x_2 \\ \text{s/a} \quad h(x) &= x_2 \end{aligned}$$

É imediato que $x^* = 0$ é uma solução deste problema, mas <u>não existe</u> nenhum valor de τ que minimize $\phi(x,0,\tau) = x_1^4 + x_1 x_2 + \frac{1}{2}\tau x_2^2$ no ponto x*. Muito embora x* seja um candidato, pois $\nabla\phi(x^*) = 0$, vem que $\nabla^2\phi(x^*)$ <u>não é definida positiva</u>, como sabemos, o ponto x* para a função ϕ é um " ponto de sela".

Resumindo, as condições necessárias são satisfeitas, mas não há suficiência.

A redução do perigo de se cair em um ponto que satisfaz as condições de Kuhn-Tucker de primeira ordem mas não satisfaz as condições suficientes de segunda ordem, reside no fato de se ter sempre uma melhora no valor da função, forçada pelos algoritmos de minimização sem restrição.

Esses algoritmos,em geral, obtêm sequência de pontos em que a condição $\phi(x^{k+1}) < \phi(x^k)$ (se $\phi(x)$ for a função que se minimiza) é obrigatória. Isso faz com se obtenha , em geral,um mínimo local de ϕ (x), que é um mínimo do problema sem restrição.

Suponhamos que um problema a ser resolvido seja do tipo (2.2.5), ou seja com restrições de desigualdade; se tomarmos a função $\psi(x,\lambda,\tau)$e fizermos $\nabla_x\psi(x,\lambda,\tau) = 0$ e admitindo λ como função de x, e τ constante suficientemente grande,ou seja:

$$\nabla_x\psi(x,\lambda,\tau) = \nabla_x\psi(x(\lambda),\lambda) = 0 \qquad (2.5.1)$$

teremos um sistema de equações não-lineares a ser resolvido; como $\nabla_x^2\psi(x^*(\lambda^*),\lambda^*)$ é positiva definida, segue-se que pelo Teorema da Função Implícita que existe uma vizinhança aberta Ω R^m em torno de λ^* e uma função $x(\lambda)$ definida em Ω tal que $x(\lambda)$ é contínua e diferenciável em Ω e que $\nabla^2\psi(x(\lambda),\lambda)$ é positiva definida para todo $\lambda \in \Omega$. Assim, $\psi(x,\lambda)$ tem um mínimo local. Logo, o sistema (2.5.1) tem solução e sua(s) solução(ões) é (são) mínimo(s) local(s) (ais) do problema (2.2.5), desde que também satisfaça a condição de que $\nabla^2\psi(x^*(\lambda^*),\lambda^*)$ seja definida positiva.

CAPÍTULO III

IMPLEMENTAÇÃO DE ALGORITMOS DE MINIMIZAÇÃO QUE USAM PENALIZAÇÃO E LAGRANGEANO AUMENTADO

3.1 - INTRODUÇÃO - Aqui trataremos de comentar e fazer uma pequena análise em alguns algoritmos, não nos deteremos em demonstrações e implicações teóricas sobre a convergência dos algoritmos; queremos fazer mais uma descrição do que propriamente uma análise.

Assumimos sempre que as funções do problema, (f, g e h), serão no mínimo de classe C^1 se a rotina de minimização sem restrições for do tipo gradientes conjugados (ou qualquer outro tipo que só use a informação de primeiras derivadas), e serão no mínimo de classe C^2 se a rotina de minimização sem restrição for do tipo Newton (ou qualquer outro que use a informação de segundas derivadas).

3.2 - DEFINIÇÃO - Definimos "MAIOR INFACTIBILIDADE" de um ponto $\bar{x} \in R^n$ em relação a uma região S não vazia, contida em R^n, com S definido por um conjunto de restrições funcionais do tipo $g(x) \leq 0$ e/ou $h(x) = 0$, a uma função real $R(\bar{x})$ dada por:

$$R(\bar{x}) = \max \{\min[0,-g_i(\bar{x})], [h_j(\bar{x})] \;,\; i=1,2,\ldots,m \;,\; j=1,2,\ldots,\ell\}$$

Se $\bar{x}$ é factível, R é igual a zero. O valor de R "mede o quanto $\bar{x}$ está longe da região S". Devido a natureza dos algoritmos que estudamos, os pontos obtidos como aproximação da solução ótima são infactíveis , sendo assim, $R(x)$ nos fornecerá a cada nova iteração uma medida de quão perto da região de factibilidade estamos.

3.3 - UM ALGORITMO PARA PENALIZAÇÃO - Uma das características do método de penalidade reside no fato de a medida que x^k tende a x^*, o ótimo, a função $Q(\mu,x)$ tende para a função $f(x)$ (por valores inferiores a $f(x)$). Essa característica pode servir como um meio para se testar a convergência. Basicamente temos três testes naturais para a convergência do método de penalidade:

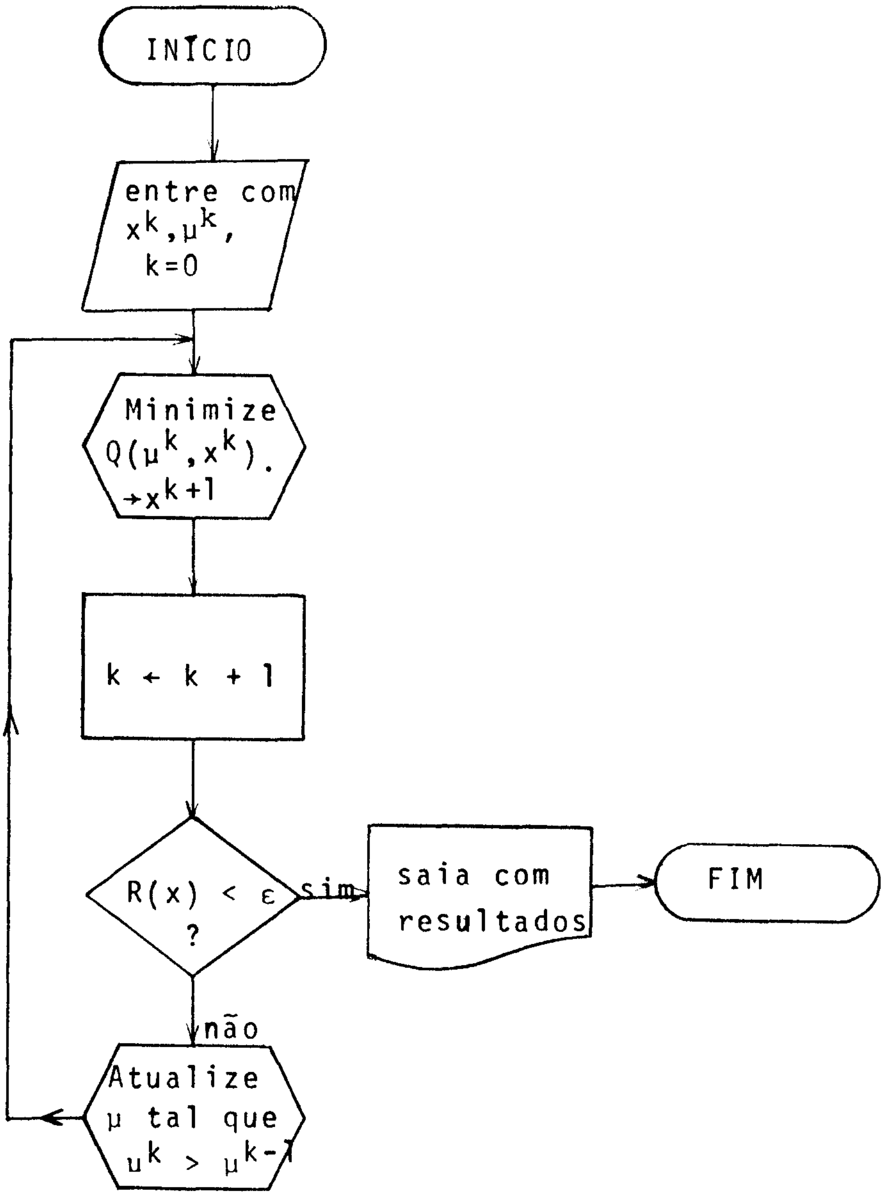
i) testar se $||x^k - x^{k-1}|| < \varepsilon$ (ε = uma dada tolerância)

ii) testar se $||Q(\mu^k,x^k)-f(x^k)|| < \varepsilon$

iii) testar se $||R(x^k)|| < \varepsilon$

O teste (i) é desaconselhável para casos em que o parâmetro de penalidade cresce lentamente, e deve ser evitado. O teste (ii) pode não ser muito bom quando a função $f(x)$ é "bastante deitada" na direção $\vec{d} = x^k-x^*$, e pior ainda, o prolongamento de $f(x)$ para fora da região de factibilidade poderia ser tal que $f(x^k)$ se aproximaria muito de $Q(\mu^k,x^k)$ e o algoritmo terminaria prematuramente. Pelas razões expostas preferimos o terceiro teste, já que este se preocupa com as restrições e não com a função objetivo.

A seguir mostramos um diagrama de blocos simplificado para o algoritmo que usamos :



OBS: As atualizações para μ são feitas para cada componente isoladamente, levando-se em consideração se é ou não necessário atualizar cada uma das componentes.

3.4 - UM ALGORITMO PARA LAGRANGEANO AUMENTADO - Diversos algoritmos foram desenvolvidos com o lagrangeano aumentado, e, em geral, a maior ou menor eficiência está ligada ao modo de se atualizar os parâmetros, tanto os das parcelas de primeiro grau(λ ou θ e/ou ρ) como os das parcelas de segundo grau (μ ou τ). O que se desejava era fazer um compromisso entre "não crescer muito" os parâmetros tipo μ ou τ e atualizar os parâmetros tipo λ ou θ com obtenção de aproximçõe**s** sucessivas dos multiplicadores de Lagrange. Se o problema é convexo, as atualizações de λ e θ conduzem ao ótimo sem que sejam necessárias as atualizações de μ ou τ; mas se o problema é não-convexo, as atualizações em μ ou τ serão provavelmente necessárias. Mesmo no caso do problema convexo, se atualizarmos somente λ ou θ (ou ρ) (movimentos duais) a convergência é, em geral, lenta, sendo assim,uma regrinha é usada no algoritmo:

Da iteração k para a iteração (k+1) as restrições "melhoram" o suficiente?

- Se sim, não aumente μ ou τ , atualize λ ou θ (faça o movimento dual).

- Se não, atualize λ ou θ (ou ρ) e aumente τ ou μ .

Existem variações sobre o procedimento acima, por exemplo, atualizar λ ou θ(ou ρ) depois ou antes de atualizar τ ou μ , como também o fato de se atualizar λ ou θ (ou ρ) em iterações diferentes das iterações de atualização de μ ou τ .

Sendo assim, uma série de algoritmos podem ser feitos e estudo sobre a eficiência particular de um sobre o outro foge ao nosso objetivo.

No caso da função $\psi(x,\lambda,\rho,\tau,\mu)$ em (2.3.2) a correção (atualização) dos parâmetros λ e ρ, para o problema (P), são como segue:

$$\lambda^{[k+1]} = \lambda^{[k]} - \tau^{[k]} \min\{ -g(x), \frac{\lambda^{[k]}}{\tau^{[k]}} \}$$

$$\rho^{[k+1]} = \rho^{[k]} + \mu^{[k]} . h(x)$$

A "melhora" de que falamos a pouco, pode ser medida do seguinte modo : seja RM a razão

$$RM = \left| \frac{g_i(x^{k+1})}{g_i(x^k)} \right| \qquad \begin{array}{l} i \in I \\ I = \{i : g_i(x^{k+1}) > 0\} \end{array}$$

Se RM for maior que um dado número (no nosso caso foi escolhido 0,25) então τ_i será corrigido (aumentado por um dado fator), e assim dizemos que a restrição g_i não melhorou bastante e por isso aumentou-se τ_i (penalizou-se) ,referente àquela restrição. Se a melhora fosse suficiente, o τ_i não seria atualizado e o movimento seria um dual simples, corrigindo-se apenas os multiplicadores das restrições de igualdade, só que não se necessita da condição de que $h_i(x^{k+1})$ seja positiva.

Note-se que se no caso em que $g_i(x^k)$ (ou $h_i(x^k)$) for menor que uma dada tolerância, o teste de razão de melhora não é feito pois g_i(ou h_i) já melhoraram o suficiente .

A seguir mostramos um algoritmo para lagrangeano aumentado. O algoritmo é bastante simplificado para que se tenha noção real do processo; maiores detalhes se encontram no apêndice C, nas listagens dos programas elaborados. Utilizou-se por motivos de simplificação, o problema abaixo,

$$\begin{array}{lll} \text{Min} & f(x) & \\ \text{s/a} & g_i(x) \leq 0 \qquad i= 1,2,\ldots,m & \qquad\qquad (P1) \end{array}$$

com a função $\Psi(x,\lambda,\tau)$ em (2.3.1)

1) Para k = 0 forneça x^k, $\lambda^k \geq 0$, $\tau^k > 0$.
2) Calcule $g_i(x^k)$ $\forall_i$ e guarde em gx
3) Minimize $\Psi(x, \lambda^k, \tau^k)$ obtendo x^{k+1}.
4) Calcule R(x) (maior **infactibilidade**) e teste a convergência. Se convergir, pare.
5) Atualize λ, $\lambda^{[k+1]} = \lambda^{[k]} - \tau \min\{ -g(x), \lambda^{[k]}/ \tau \}$
6) Atualize τ para as restrições que não melhoram o suficiente (RM > 0,25).
7) Remova $g(x^{k+1})$ para gx e vá para 3.

3.5 - <u>CONSIDERAÇÕES SOBRE A EFICIÊNCIA E PROBLEMAS NUMÉRICOS</u> - O aparecimento de um fator empírico (RM menor que 0,25) , não significa uma maneira má de se abordar a melhora das restrições de uma iteração para outra, pois, como sabemos, para cada problema (dentre os infinitos) e cada ponto inicial para o processo, a maneira como deverão decorrer as razões $g_i(x^{k+1}) \ / \ g_i(x^k)$ ou $h_j(x^{k+1})/ h_j(x^k)$, são totalmente imprevisíveis. O fato de usarmos 0,25(melhora de 4 vezes)[6] decorre de experiências já realizadas, fortalecendo o fator empírico utilizado.

A eficiência dos métodos penalidade simples e lagrangeano aumentado reside primordialmente no algoritmo de minimização sem restrições, se este for muito bom, dificilmente o processo falhará.

Os métodos de penalidade e lagrangeano aumentado constituem boa ferramenta quando não se necessita de uma precisão muito elevada. Esses métodos convergem muito lentamente em pontos próximos ao ótimo, e devido a isso podem realizar muitas iterações para passar de um ponto ao ponto ótimo, estando o primeiro já "próximo" do último. Testes mostram que em poucas iterações o ponto obtido já está próximo do ótimo, e o "gasto" das muitas iterações seguintes são, por assim dizer, "refinamentos caros". Um estudo preliminar do problema com respeito à precisão poderá baratear se, no caso, se verificar que não é necessária uma precisão tão alta.

---

(6) O valor 0,25 é proveniente de muitos testes computacionais, veja R.Fletcher [4]

CAPÍTULO IV

COMPARAÇÃO NUMÉRICA ENTRE OS MÉTODOS : PENALIDADE / LAGRANGEANO AUMENTADO

4.1 - INTRODUÇÃO - Alguns resultados obtidos pelo computador fornecem uma comparação entre os métodos de penalidade e lagrangeano aumentado. Foi utilizado um conjunto de funções testes para a comparação, 17 no total, que inclui vários problemas clássicos. Este conjunto já foi amplamente testado, veja [3], e contém problemas de até 20 variáveis com 18 restrições. Os problemas clássicos são 7 no total, e acham-se em J. Asaadi [1]. Veremos, em outro capítulo, o emprego dos algoritmos para problemas maiores.

4.2 - TABELA DE COMPARAÇÕES - Mostramos adiante uma tabela com as comparações entre os dois métodos.

Para cada linha de um problema, duas especificações são dadas: a primeira é para o método de penalidade, a segunda para o lagrangeano aumentado. A primeira coluna da tabela contém o número do problema, as colunas seguintes contêm : o número de variáveis, número de restrições de desigualdade e número de restrições de igualdade (N,M,L); o número de problemas parciais; o número total de avaliações de função e gradiente realizados dentro da sub-rotina de minimização sem restrições; o número total de iterações dentro da sub-rotina de minimização sem restrições; a maior infac-

Nº DO PROBLEMA	Nº DE PROBLEMAS PARCIAIS — N, M, L		Nº TOTAL DE ITERAÇÕES DENTRO DA SUB. DE MINIM S/R — Nº DE AVALIAÇÕES DE F. E GRAD.		TEMPO DE C.P.U. — DIAGNÓSTICO — MAIOR INFACTIBILIDADE		
1	2, 3, 0	6	378	31	0.150E-4	Ø	2,241
		3	855	35	0.409E-5	Ø	1,874
2	2, 0, 1	4	31	12	0.500E-4	Ø	0,244
		3	50	14	0.709E-6	Ø	0,192
3	2, 3, 0	4	420	38	0.186E-4	Ø	1.921
		4	201	25	0.403E-5	Ø	0,971
4	2, 3, 0	5	430	46	0.150E-4	Ø	1,897
		7	237	40	Ø.	Ø	0,969
5	2, 3, 0	4	111	18	0.800E-4	Ø	0,513
		5	438	16	0.327E-4	Ø	1,027
6	2, 3, 0	1	3	1	Ø.	Ø	0,114
		1	3	1	Ø.	Ø	0,107
7 *	2, 0, 0	1	60	25	Ø.	Ø	0,251
		1	60	25	Ø.	Ø	0,134
8	2, 2, 0	5	100	25	0.200E-4	Ø	0,467
		5	38	15	0,581E-6	Ø	0,27
9	4, 3, 0	5	1376	69	0.102E-4	Ø	5,661
		7	2080	73	0.593E-4	Ø	5,817
10	3, 4, 0	4	695	31	0.111E-4	Ø	2,551
		4	464	19	0.176E-4	Ø	1,193
11	5, 0, 3	3	141	63	0.259E-4	Ø	1,154
		2	100	46	0.171E-4	Ø	0,768
12	7, 4, 0	4	1266	106	0.588E-4	Ø	7,596
		6	1652	132	Ø.	Ø	6,958
13	10, 8, 0	4	1557	174	0.820E-4	Ø	16,95
		6	1323	141	0.362E-4	Ø	10,82
14	20,17, 0	4	1370	550	0.844E-4	Ø	35,88
		9	2527	489	0.227E-3	1	61,18
15	2, 2, 1	5	107	42	0.203E-4	Ø	0,539
		4	103	41	0,722E-4	Ø	0,463
16	2, 5, 0	4	1127	37	0.486E-4	Ø	7,391
		6	1011	32	0.849E-6	Ø	3,462
17	2, 3, 0	4	68	24	0.172E-4	Ø	0,348
		4	53	18	0.475E-5	Ø	0,253

* O problema 7 é um problema sem restrições. Queríamos mostrar apenas que a sub-rotina implementada, também funciona com casos particulares. Neste caso, o mérito é da sub-rotina de gradientes conjugados.

tibilidade; o diagnóstico ($\emptyset$ = convergência , 1 = superou o número máximo de problemas parciais permitido sem atingir a precisão pedida); o tempo de processamento (tempo de C.P.U.) gasto pela execução do problema.

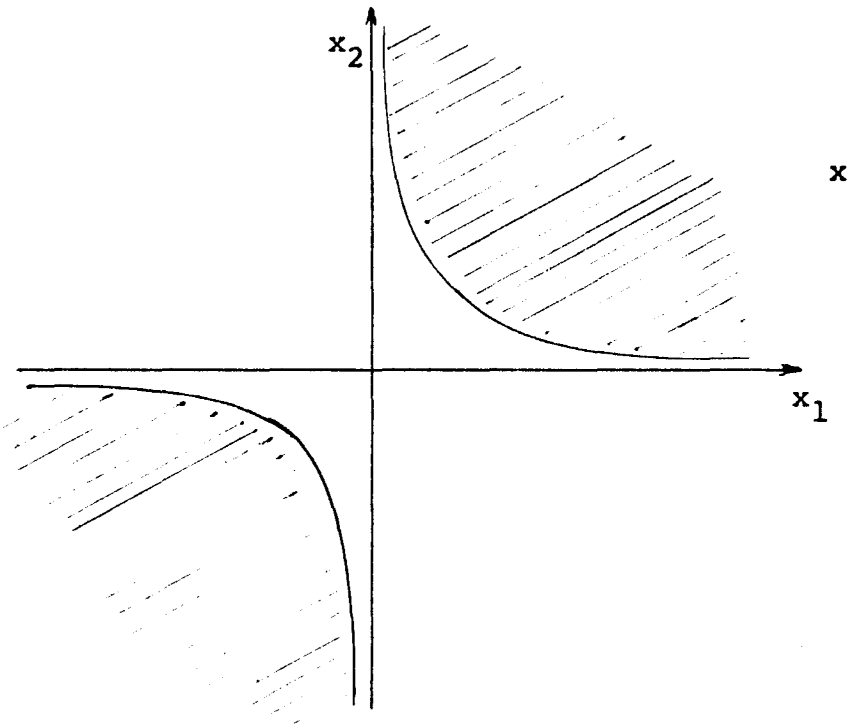
A relação dos problemas encontram-se adiante, e a mesma contém também os pontos iniciais e os pontos ótimos. Os pontos iniciais que foram fornecidos são os mesmos dos testes feitos por[3].

Cada chamada da sub-rotina de minimizações sem restrições chamamos problema parcial, no nosso caso o número de chamadas (problemas parciais) máximo permitido foi de 9(nove). Vemos no problema nº 14 que o lagrangeano aumentado não atingiu a tolerância desejada(10^{-4}).

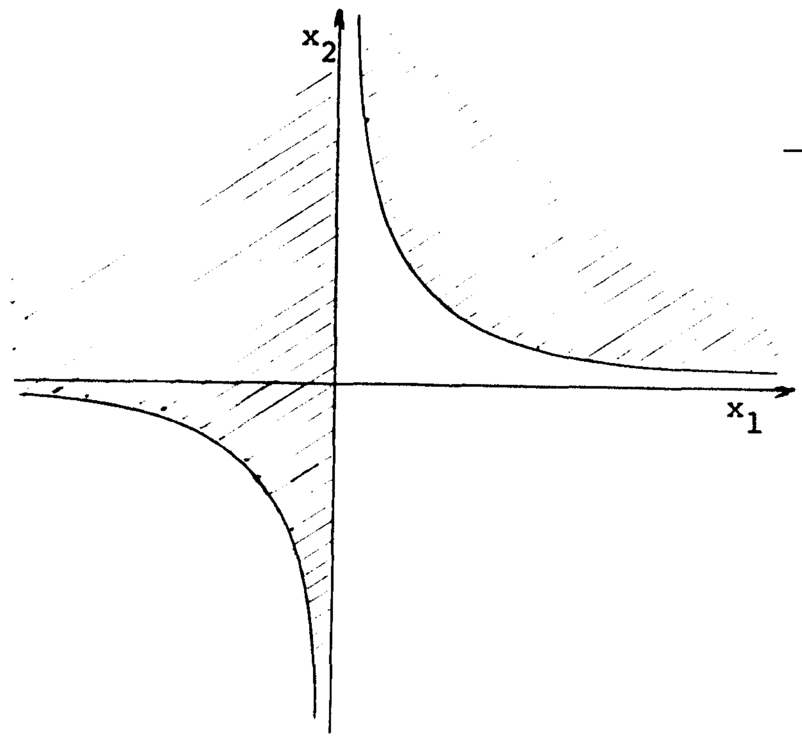
Em todos os casos,os parâmetros μ^0, τ^0 ou ρ^0 iniciaram com valor 10,0 para todas as componentes. O parâmetro λ foi iniciado com valor 0,01, no lagrangeano aumentado, para todas as componentes, com exceção do problema nº 1 que começou com $\lambda^0 = 0$.

As razões para a pequena mudança no problema nº 1 são as seguintes: a restrição $g_3(x) = \frac{1}{x_1} - x_2 \leq 0$ se apresenta, de propósito, de uma maneira numericamente ruim; é lógico que $1 - x_1x_2 \leq 0$ apresentaria nemos problemas numéricos, mesmo assim, observando mais atentamente $g_3(x)$, vemos que a região que verifica a desigualdade é diferente da região para $1 - x_1x_2 \leq 0$, veja figura, e com isso, pontos que se aproximam de (0,0), com $x_1 < 0$, estão se aproximando do ótimo, que é diferente de $x^* = (0,577 ; 1,732)$. Observando a região de factibilidade do problema nº 1, notamos que a semi-reta

$x_1 = 0$, $x_2 \geq 0$ faz parte da região, no limite, embora $g_3(x)$ não

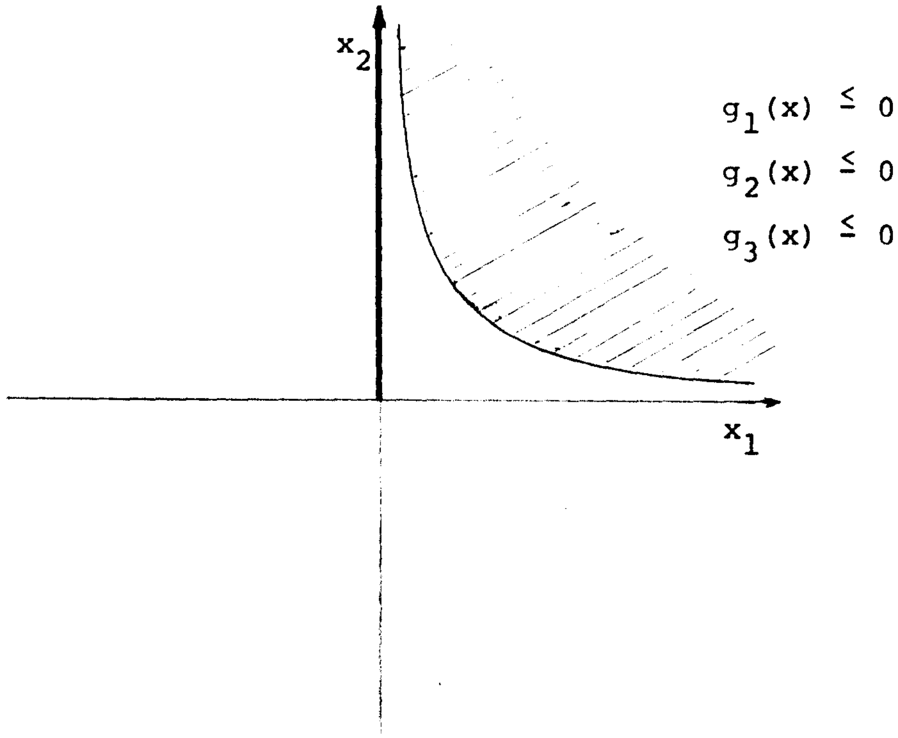


$x_1 x_2 \geq 1$



$\frac{1}{x_1} - x_2 \leq 0$

seja definida para a semi-reta. Queremos dizer que quando $x_1 \to 0^-$ g_3 é verificada e $x_1 \geq 0$ e $x_2 \geq 0$ são verificados com precisão ε se $x^k = (-\varepsilon\ , 0)$.



Com o exposto queremos dizer que o lagrangeano aumentado chegou a convergir para um ponto próximo de (0,0), que é, na verdade, um limite para o ótimo global.

Os resultados foram obtidos quando iniciamos o processo com $\lambda^0 = 0{,}01$ e são dados a seguir:

- o melhor ponto obtido: x = (-0,3638E-11;-0,1418E-7)
- número total de avaliação de função e gradiente: 346
- número de problemas parciais: 4
- número total de iterações dentro da sub-rotina de minimiza

ção sem restrições: 31

- diagnóstico : Ø
- tempo de C.P.U. : 1,552 segundos
- maior infactibilidade : 0,1418E-7

A situação exposta acontece com a penalidade, e a aplicação do algoritmo conduz a um ponto próximo de (0,0), os resultados obtidos são os da tabela. É claro que o ponto $x^*=(0{,}577;\ 1{,}732)$ pode também ser atingido pelo método com o mesmo ponto inicial fornecido, bastando que comecemos com um parâmetro de penalização adequado ($\mu = 100$ é um exemplo).

4.3 - CONCLUSÕES SOBRE A COMPARAÇÃO - Uma conclusão que se tira da tabela é que o lagrangeano aumentado é, em geral, mais rápido do que o método de penalidade. Esta diferença não é muito acentuada nos problemas expostos, contudo, o número de problemas parciais são mais ou menos idênticos para os dois, sendo assim a minimização da função de penalização é, em geral, mais cara do que a minimização do lagrangeano aumentado.

A própria natureza da função penalidade $Q(\mu,x)$ com μ grande faz com que a sub-rotina de minimizações exerça um trabalho maior. Nem sempre porém, a penalidade perde para o lagrangeano aumentado, principalmente nos casos em que um μ, não muito alto, seja suficiente para a tolerância pedida. Em alguns casos, com o aumento da precisão pedida , a penalidade fornece pontos com um erro significante, enquanto que o lagrangeano aumentado tem melhores condições com o aumento da precisão. Nos problemas testes, algumas componentes do parâmetro μ chegaram a atingir valores elevadíssimos (1.000.000), nesses testes o preço da minimização sem restrições era bem maior que os apresentados pelo lagrangeano aumentado.

Observando o problema nº 14, em que o lagrangeano aumentado não conseguiu atingir a precisão de 10^{-4} e chegou até 0.227E-3, vê-se que o número total de iterações dentro da sub-rotina de minimizações foi 489, que é menor que 550 da penalidade. Observando ainda, o número de avaliações de função e gradiente foram bem maiores para o lagrangeano aumentado. Isso é explicado pelo fato de o lagrangeano ter feito mais trabalho por iteração dentro da sub-rotina de minimizações. No problema nº 13 o lagrangeano aumentado realizou mais problemas parciais (chamadas à sub-rotina), porém o seu trabalho médio por iteração foi bem menor, idem para os problemas nºs 2, 3, 4, 8, 10, 11, 12, 15, 16 e 17.

Assim, o lagrangeano aumentado fez menos trabalho médio por iteração.

4.4 - RESULTADOS OBTIDOS POR M.A.B.DINIZ [3] - Fato relevante reside no fato de os métodos que tratamos não usarem a informação de segundas derivadas, e devido a isso a perda de tempo computacional seja em parte compensada pelo ganho de tempo em não ter que se codificar os hessianos da função objetivo e das restrições. M.A. Diniz implementou dois métodos de minimização com restrições (LAPRO e NEWPQ) que utilizam os hessianos das funções dos problemas, portanto, mais informações. Uma tabela com os resultados

Tabela dos resultados obtidos por M.A.B. Diniz.

	Problema	Nº Iterações	Nº Avaliações de f	Tempo de exec. (s)
LAPRO	1	8	19	0.24
NEWPQ		28	53	1.67
LAPRO	2	4	8	0.10
NEWPQ		4	6	0.16
LAPRO	3	8	9	0.20
NEWPQ		8	9	0.49
LAPRO	4	7	18	0.15
NEWPQ		-	-	-
LAPRO	5	3	4	0.07
NEWPQ		3	4	0.15
LAPRO	6	1	2	0.01
NEWPQ		2	3	0.07
LAPRO	7	21	27	0.31
NEWPQ		21	27	0.50
LAPRO	8	2	3	0.04
NEWPQ		2	3	0.10
LAPRO	9	5	8	0.20
NEWPQ		7	8	1.07
LAPRO	10	2	3	0.07
NEWPQ		2	3	0.24
LAPRO	11	2	4	0.16
NEWPQ		6	10	1.48
LAPRO	12	7	13	0.67
NEWPQ		-	-	-
LAPRO	13	7	13	1.72
NEWPQ		4	5	5.97
LAPRO	14	16	28	18.92
NEWPQ		5	6	51.57
LAPRO	15	5	10	0.07
NEWPQ		4	5	0.22
LAPRO	16	12	13	0.18
NEWPQ		2	4	0.12
LAPRO	17	2	3	0.06
NEWPQ		2	3	0.12

obtidos por M.A.B. Diniz mostram a diferença de "preço" entre os métodos com ou sem uso de hessianos. Nota-se nas tabelas que os tempos de C.P.U. são bem menores para os algoritmos de M.A.B.Diniz , e a justificação para uso dos métodos sem hessiano está na sua aplicação a problemas de grande porte que não requeiram alta precisão nos resultados e nos quais a utilização dos hessianos seja difícil devido a problemas de memória ou dificuldade devido a um trabalho excessivo na codificação dos hessianos .

A sub-rotina LAPRO de Diniz implementa um método do tipo hessiano do lagrangeano projetado para porte médio e exige que o ponto inicial para o problema a ser resolvido, seja factível, já os métodos de penalidade e lagrangeano aumentado não necessitam desta exigência. Na realidade, para se obter um ponto factível para a sub-rotina LAPRO resolve-se um problema a parte, que fornece um ponto factível. Isso, em termos de problemas maiores poderá ser difícil ou trabalhoso. Uma análise do problema a ser resolvido irá indicar que tipo de método pode ou deve ser aplicado.

Conclui-se que, um estudo particular para cada problema, em termos de tempo de codificação, tempo de C.P.U., memória disponível, tamanho do problema conduz a uma escolha satisfatória.

A seguir mostramos a relação dos problemas testes.

PROBLEMAS TESTES

PROBLEMA 1

$f(x) = 3x_1 + x_2$

$g_1(x) = - x_1$

$g_2(x) = - x_2$

$g_3(x) = 1/x_1 - x_2$

$x^0 = (1Ø;Ø,\ 2)$

$x^* = (Ø,\ 577;\ 1{,}732)$

PROBLEMA 2

$f(x) = x_2 - x_1^2$

$h(x) = x_2 - 2x_1^2$

$x^0 = (1\ ;\ 2)$

$x^* = (Ø\ ;\ Ø)$

PROBLEMA 3

$f(x) = x_1 + x_2$

$g_1(x) = 1 - x_1 x_2$

$g_2(x) = 1 - x_1 - 2x_2$

$g_3(x) = 2 + x_1 - 4x_2$

$x^0 = (4{,}6\ ;\ 1,\ 8)$

$x^* = (Ø,\ 5858\ ;\ 1,\ 7071)$

PROBLEMA 4

$f(x) = 9x_1 + x_2$

As mesmas restrições do problema 3

$x^0 = (4,\ 6\ ;\ 1,\ 8)$

$x^* = (1/3\ ;\ 3)$

PROBLEMA 5

$f(x) = x_1^2 + x_2^2$

As mesmas restrições do problema 3

$x^0 = (4,\ 6\ ;\ 1,\ 8)$

$x^* = (4/5\ ;\ 8/5)$

PROBLEMA 6

$f(x) = (x_1 - 2{,}5)^2 + (x_2 - 2{,}5)^2$

As mesmas restrições do problema 3

$x^0 = (4{,}6\ ;\ 1{,}8)$

$x^* = (2{,}5\ ;\ 2{,}5)$

PROBLEMA 7

$$f(x) = 100(x_2-x_1^2)^2 + (1-x_1)^2$$

$x^0 = (-1,2 ; 1)$

$x^* = (1 ; 1)$

PROBLEMA 8

$$f(x) = \frac{1}{3}(x_1+1)^3 + x_3$$

$$g_1(x) = -x_1 + 1$$

$$g_2(x) = -x_2$$

$x^0 = (1,125; 0,125)$

$x^* = (1 ; 0)$

PROBLEMA 9

$$f(x) = x_1^2+x_2^2+ 2x_3^2+x_4^2-5x_1- 5x_2- 21x_3+ 7x_4$$

$$g_1(x) = x_1^2+ x_2^2+ x_3^2+ x_4^2+ x_1- x_2+ x_3- x_4- 8$$

$$g_2(x) = x_1^2+ 2x_2^2+ x_3^2+ 2x_4^2- x_1- x_4-10$$

$$g_3(x) = 2x_1^2+ x_2^2+ x_3^2+ 2x_1- x_2- x_4- 5$$

$x^0 = (0; 0; 0; 0)$ $\quad$ $x^* = (0; 1; 2; -1)$

PROBLEMA 10

$$f(x) = 9 -8x_1-6x_2-4x_3+2x_1^2+2x_2^2+x_3^2 + 2x_1x_2+2x_1x_3$$

$$g_1(x) = -x_1$$

$$g_2(x) = -x_2$$

$$g_3(x) = -x_3$$

$$g_4(x) = x_1+x_2+2x_3-3$$

$x^0=(0,5;0,5;0,5)$

$x^*=(4/3 ; 7/9 ; 4/9)$

PROBLEMA 11

$$f(x) = \exp(x_1.x_2.x_3.x_4.x_5)$$

$$h_1(x) = x_1^2+ x_2^2+ x_3^2+ x_4^2+x_5^2-10$$

$$h_2(x) = x_2x_3 - 5x_4x_5$$

$$h_3(x) = x_1^3 + x_2^3 + 1$$

$x^0 = (-2,2,2,-1,1)$

$x^*=(-1,7171;1,5957;$
$1,8272;-0,7636;$
$-0,7636)$

PROBLEMA 12

$$f(x) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

$$g_1(x) = 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 - 127$$

$$g_2(x) = 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 - 282$$

$$g_3(x) = 23x_1 + x_2^2 + 6x_6^2 - 8x_7 - 196$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7$$

$$x^0 = (1;\ 2;\ 0;\ 4;\ 0;\ 1;\ 1)$$

$$x^* = (2{,}30;\ 1{,}95;\ -0{,}47;\ 4{,}37;\ 0{,}51;\ 1{,}03;\ 1{,}58)$$

PROBLEMA 13

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45$$

$$g_1(x) = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120$$

$$g_2(x) = 5x_1^2 + 8x_2 + (x_3-6)^2 - 2x_4 - 4$$

$$g_3(x) = \frac{1}{2}(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30$$

$$g_4(x) = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6$$

$$g_5(x) = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105$$

$$g_6(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8$$

$$g_7(x) = -3x_1 + 6x_2 + 12(x_9-8)^2 - 7x_{10}$$

$$g_8(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12$$

$x^0 = (2;\ 3;\ 5;\ 5;\ 1;\ 2;\ 7;\ 3;\ 6;\ 10)$

$x^* = (2{,}17;\ 2{,}36;\ 8{,}77;\ 5{,}09;\ 0{,}99;\ 1{,}43;\ 1{,}32;\ 9{,}82;$
$8{,}28;\ 8{,}37)$

<u>PROBLEMA 14</u>

$$f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2$$
$$(x_5-3)^2 + 2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2$$
$$+ (x_{10}-7)^2 + (x_{11}-9)^2 + 10(x_{12}-1)^2 + 5(x_{13}-7)^2 +$$
$$+ 4(x_{14} - 14)^2 + 27(x_{15} - 1)^2 + x_{16}^4 + (x_{17}-2)^2 +$$
$$+ 13(x_{18} - 2)^2 + (x_{19} - 3)^2 + x_{20}^3 + 95$$

$$g_1(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120$$

$$g_2(x) = 5x_1^2 + 8x^2 + (x_3-6)^2 - 2x_4 - 40$$

$$g_3(x) = \frac{1}{2}(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30$$

$$g_4(x) = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6$$

$$g_5(x) = 4x_1 + 5x_2 - 3x_7 + 9x_8 - 105$$

$$g_6(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8$$

$$g_7(x) = 3x_1 + 6x_2 + 12(x_9-8)^2 - 7x_{10}$$

$$g_8(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12$$

$$g_9(x) = x_1 + x_2 + 4x_{11} - 21x_{12}$$

$$g_{10}(x) = x_1^2 + 15x_{11} - 8x_{12} - 28$$

$$g_{11}(x) = 4x_1 + 9x_2 + 5x_{13}^2 - 9x_{14} - 87$$

$$g_{12}(x) = 3x_1 + 4x_2 + 3(x_{13}-6)^2 - 14x_{14} - 10$$

$g_{13}(x) = 14x_1^2 + 35x_{15} - 79x_{16} - 92$

$g_{14}(x) = 15x_2^2 + 11x_{15} - 61x_{16} - 54$

$g_{15}(x) = 5x_1^2 + 2x_2 + 9x_{17}^4 - x_{18} - 68$

$g_{16}(x) = x_1^2 - x_2 + 19x_{19} - 20x_{20} + 19$

$g_{17}(x) = 7x_1^2 + 5x_2^2 + x_{19}^2 - 30x_{20}$

$x^0 = (2;\ 3;\ 5;\ 1;\ 2;\ 7;\ 3;\ 6;\ 10;\ 2;\ 2;\ 6;\ 15;\ 1;\ 2;\ 1;\ 2;\ 1;\ 3)$

$x^* = (2{,}17;\ 2{,}35;\ 8{,}76;\ 5{,}06;\ 0{,}98;\ 1{,}43;\ 1{,}32;\ 9{,}83;\ 8{,}28;\ 8{,}37;\ 2{,}27;\ 1{,}35;\ 6{,}07;\ 14{,}17;\ 0{,}99;\ 0{,}65;\ 1{,}46;\ 2{,}0;\ 1{,}04;\ 2{,}06)$

PROBLEMA 15

$f(x) = -12x_1 - 7x_2 + x_2^2$ $\qquad x^0 = (1\ ;\ 0)$

$g_1(x) = -x_1$

$g_2(x) = -x_2$ $\qquad x^* = (0{,}7175;\ 1{,}4698)$

$h_1(x) = -2x_1^4 - x_2 + 2$

PROBLEMA 16

$f(x) = x_2 - x_1$

$g_1(x) = x_2 - x_1 - 5$ $\qquad x^0 = (0{,}3\ ;\ 5{,}2)$

$g_2(x) = -3x_1 - x_2 + 6$ $\qquad x^* = (9{,}2857;\ 2{,}8571)$

$g_3(x) = x_1/5 - x_2 + 1$

$g_4(x) = 10x_1 - x_2 - 90$

$g_5(x) = x_1/2 + x_2 - 8$

PROBLEMA 17 $\quad f(x) = \frac{1}{16} x_1^2 + \frac{1}{9} x_2^2$

$g_1(x) = x_2 - x_1$ $\qquad x^0 = (6 \; ; \; 5)$

$g_2(x) = 1 - x_2$ $\qquad x^* = (1 \; ; \; 1)$

$g_3(x) = x_1/3 + x_2 - 7$

CAPÍTULO V

O PROBLEMA DOS PEIXES

5.1 - INTRODUÇÃO - Com o intuito de empregar o lagrangeano aumentado para problemas grandes, resolvemos testar o algoritmo para dois problemas de algum cunho prático. Aqui não entramos na questão de se empregar, na prática, os problemas; queremos apenas mostrar a performance de um algoritmo.

Os dois problemas são semelhantes, mudando apenas no número de variáveis e restrições, estes problemas tratam de maximizar os produtos de uma pescaria em um dado horizonte de tempo, Explicações com mais detalhes serão vistas mais adiante.

5.2 - QUESTÃO DA PRECISÃO - Como já dissemos, para algoritmos do tipo que tratamos, é importante o fator precisão, ou seja, com o aumento da precisão o número de problemas parciais necessérios pode aumentar muito e o tempo de C.P.U. necessário pode ser grande. Pelo exposto, deve-se pedir a precisão estritamente necessária, já que pedir uma precisão maior do que a necessária pode-se pagar um preço alto. Outro fator importante que deve-se fazer, é limitar o número de problemas parciais em um número relativamente pequeno; em muitos casos, como já dissemos, uma boa parte dos problemas parciais são "refinamentos caros", devendo-se, na medida do possível, evitar-se atingir o critério de convergência fazendo um número elevado de problemas parciais, e em diversos casos existe a possibilidade de nunca se atingir a precisão pedida, já que a própria precisão da máquina limita o número de decimais exatas .

5.3 - O PROBLEMA DO CRESCIMENTO POPULACIONAL EM UM MEIO LIMITADO - Seja uma espécie (por exemplo: peixes) que tenha provisões de alimento ilimitada e "habitat" em meio ilimitado e que não sofra influências de agentes externos. Sob estas condições o crescimento populacional pode ser descrito matemáticamente pela equação diferencial:

$$\frac{dy(t)}{dt} = ky(t) \qquad , k > 0$$

onde k é uma constante, k é a razão entre a taxa de crescimento e a quantidade presente em um dado tempo t. Isso quer dizer que o crescimento populacional é diretamente proporcional a quantidade presente(população). Por outro lado, se o meio onde vive a espécie é limitado, ou a provisão de alimentos é limitada, a descrição matemática muda e passa a ser da seguinte forma:

$$\frac{dy(t)}{dt} = k_1 y(t) - k_2 y(t)^2 \qquad k_1, k_2 > 0$$

onde k_1 e k_2 são constantes positivas que aparecem considerando a hipótese, agora, de que k seja função linear de quantidade presente, $k = f(y(t)) = k_1 - k_2 y(t)$, de maneira que com o aumento da quantidade presente y(t), a taxa de crescimento (k) diminua linermente com y(t).

Consideremos agora que não só o meio e os alimentos constituem fator para uma alteração na taxa de crescimento de uma população, mas, também, a presença de outra ou outras espécies no mesmo meio disputando ou não o mesmo alimento, havendo ou não o caso de uma espécie devorando a outra. As influências, vamos considerar, são, em intensidade, diretamente proporcional às quantidades presentes das espécies. Para simplificar, mostramos a seguir um exemplo do que falamos. Não queremos fazer uma análise completa deste estudo, já que foge ao nosso objetivo, contudo, uma idéia será dada: Considere 2 espécies A e B, com influências de A sobre B e vice-versa. Então uma descrição matemática da situação englobando as situações anteriores é a seguinte:

$y_1(t)$ - espécie A (população)

$y_2(t)$ - espécie B (população)

$$\frac{dy_1(t)}{dt} = a_1y_1(t) + b_1y_1(t)^2 + c_1y_1(t)y_2(t)$$

$$\frac{dy_2(t)}{dt} = a_2y_2(t) + b_2y_2(t)^2 + c_2y_2(t)y_2(t)$$

$a_1, b_1, c_1, a_2, b_2, c_2$ constantes ,

$b_1, b_2 < 0$

i) se c_1 e c_2 forem ambos negativos é porque a presença de uma das espécies é prejudicial a outra, digamos, comportilham os mesmos alimentos.

ii) se $c_1 > 0$ e $c_2 < 0$ significa que a presença da espécie B é benéfica para a espécie A, embora a espécie B seja prejudicada com a espécie A; digamos que, neste caso, a espécie A se alimenta da espécie B.

iii) se c_1 e c_2 forem ambos positivos significa que as espécies se ajudam mutuamente, e a presença de uma no meio em que vive a outra, fortalece o seu cresimento populacional.

Como já dissemos, não trataremos os pormenores da questão das espécies, o leitor pode encontrar bastante dados em [10].

O sistema de equações diferenciais que aparece, dependendo das constantes e das condições iniciais, pode fornecer pontos de equilíbrio, curvas do crescimento ou decrescimento de uma espécie, etc. Mas, não estamos interessados em resolver um sistema de equações diferenciais, o que queremos é fazer uma pescaria das espécies (fizemos para 3 espécies), de modo a maximizar o lucro da pesca;o nosso caso,é maximizar o número de unidades pescadas dentro de um certo horizonte de tempo.

5.4 - FORMULAÇÃO DO PROBLEMA DOS PEIXES - Considere 3 espécies A,B,C que vivem em um mesmo meio e que se interferem mutuamente. Desejamos fazer uma pescaria das espécies ao longo de T períodos de tempo,de modo a colher o número máximo de unidades de peixes(o problema poderia considerar diferentes preços para os peixes). Sejam as equações abaixo a descrição matemática de sua evolução:

$$\frac{dy_1(t)}{dt} = a_1y_1(t) + b_1y_1(t)^2 + c_1y_1(t)y_2(t) + d_1y_1(t)y_2(t)$$

$$\frac{dy_2(t)}{dt} = a_2y_2(t) + b_2y_2(t)^2 + c_2y_2(t)y_1(t) + d_2y_2(t)y_3(t) \qquad (5.4.1)$$

$$\frac{dy_3(t)}{dt} = a_3y_3(t) + b_3y_3(t)^2 + c_3y_3(t)y_1(t) + d_3y_3(t)y_2(t)$$

a_i, b_i, c_i, d_i constantes

$b_i < 0$

Como o processo é dificilmente descrito com precisão, ou seja, dificilmente conseguimos saber o número exato das espécies e suas taxas evolutivas ($a_{is}, b_{is}, c_{is}, d_{is}$), podemos tratar o problema discreto, sem corrermos o risco de termos um modelo não significativo da realidade. Assim, o sistema de equações diferenciais (5.4.1) pode ser tratado mudando as derivadas pelas correspondentes razões incrementais:

$$\frac{dy_i(t)}{dt} \simeq \frac{y_i(t+h) - y_i(t)}{h} \qquad i = 1,2,3$$

Considerando as razões expostas, podemos tomar $h = 1$ (isso é razoável), e assim obtermos um sistema de <u>equações recursivas</u>, ou sejam, equações da forma $f_{t+1} = f_t(x^t)$, em que cada instante t+1 é função do instante t.

Temos, portanto, 3 equações recursivas abaixo:

$$y_1(t+1) = y_1(t)+a_1y_1(t)+b_1y_1(t)^2+c_1y_1(t)y_2(t)+d_1y_1(t)y_3(t)$$

$$y_2(t+1) = y_2(t)+a_2y_2(t)+b_2y_2(t)^2+c_2y_2(t)y_1(t)+d_2y_2(t)y_3(t)$$

$$y_3(t+1) = y_3(t)+a_2y_3(t)+b_3y_3(t)^2+c_3y_3(t)y_1(t)+d_3y_3(t)y_2(t)$$

Finalmente, após cada período de tempo teremos uma pescaria de cada espécie (que pode ser nula). E se considerarmos T períodos de tempo, completamos a formulação do problema:

$$\text{Maximizar} \sum_{k=0}^{T-1} \text{custo}_1 u_1(k)+\text{custo}_2 u_2(k)+\text{custo}_3 u_3(k)$$

sujeito a

$$y_1(t+1)=y_1(t)+a_1y_1(t)+b_1y_1(t)^2+c_1y_1(t)y_2(t)+d_1y_1(t)y_3(t)-u_1(t)$$

$$y_2(t+1)=y_2(t)+a_2y_2(t)+b_2y_2(t)^2+c_2y_2(t)y_1(t)+d_2y_2(t)y_3(t)-u_2(t)$$

$$y_3(t+1)=y_3(t)+a_3y_3(t)+b_3y_3(t)^2+c_3y_3(t)y_1(t)+d_3y_3(t)y_2(t)-u_3(t)$$

$$y_1(t) \geq 0,$$
$$y_2(t) \geq 0,$$
$$y_3(t) \geq 0, \quad t = 1,2,\ldots,T$$

$$u_1(t) \geq 0,$$
$$u_2(t) \geq 0,$$
$$u_3(t) \geq 0, \quad t = 0,1,2,\ldots,(T-1)$$

$y_1(0)$, $y_2(0)$, $y_3(0)$ dados.

5.5 - O PROBLEMA DOS PEIXES I - Para facilitar a visualização dos resultados de um problema maior, resolvemos implementar um problema pequeno do tipo descrito em 5.4 para $T= 5$, havendo somente 5 pescarias; mesmo assim, o problema conta com 15 restrições de igualdade e 30 restrições do tipo $x \geq 0$ (canalizações); nos nossos programas estas restrições são consideradas como restrições de desigualdade, não havendo distinção entre uma restrição propriamente dita e uma canalização.

As constantes usadas foram:

$a_1 = 0{,}05$	$b_1 = -5{,}0E-5$	$c_1 = 2{,}0E-5$	$d_1 = 1{,}25E-5$
$a_2 = 0{,}05$	$b_2 = -0{,}333E-5$	$c_2 = 7{,}5E-5$	$d_2 = 2{,}5E-5$
$a_3 = 0{,}08$	$b_3 = -8{,}0E-5$	$c_3 = -1{,}6E-5$	$d_3 = -1{,}6E-5$

$custo_1 = custo_2 = custo_3 = 1$ unidade

$y_i(0) = 1000$ unidades, $i = 1,2,3$.

O ponto inicial para nosso problema foi obtido por uma sub-rotina que gera um "ponto inicial bom" para começar o processo de minimização. As listagens do problema e das sub-rotinas usadas estão no apêndice C .

Aqui mostramos o ponto inicial fornecido e os resultados finais obtidos. Os y_s são os estados, os u_s são as variáveis de controle (pesca).

O valor da função objetivo é negativo porque trabalhamos com minimização; este valor nos fornece o número de unidades pescadas no decorrer do horizonte.

[illegible] :

300.0000	300.0000	300.0000	300.0000	300.00
732.5000	460.2679	180.7836	-109.7834	-416.4333
200.0000	200.0000	200.0000	200.0000	200.0000
746.6700	526.7907	326.5925	135.0783	-57.75985
200.0000	200.0000	200.0000	200.0000	200.0000
768.0000	564.0780	374.8411	190.5450	2.806853

O ponto é disposto, computacionalmente, da seguinte maneira:

$X = (U_1, Y_1, U_2, Y_2, U_3, Y_3)$ onde U_1, U_2, U_3 vão de $\emptyset$ a T-1 e

Y_1, Y_2, Y_3 vão de 1 a T.

[illegible] : 50
[illegible] : 23
[illegible] : 6

[illegible]

[illegible] 5.01[illegible]

[illegible]

[illegible] : -3033.7[illegible]

Observe na tabela " Valores dos controles e estados" que os últimos estados, para as tres espécies, são aproximadamente nulos. Isto era de se esperar, pois ao final a "pesca" deveria ser "fulmi - nante".

[illegible]

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Y2(1)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1?7.	Y2(2)=	531.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	1?3.	Y2(3)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Y2(4)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Y2(5)=	1.

U3(0)=	200.	Y3(1)=	768.
U3(1)=	200.	Y3(2)=	564.
U3(2)=	200.	Y3(3)=	375.
U3(3)=	200.	Y3(4)=	190.
U3(4)=	200.	Y3(5)=	2.

5.6 - O PROBLEMA DOS PEIXES II - Como dissemos em 5.5, o problema dos peixes I foi implementado para se ter uma noção de um resultado de um problema maior. Com a experiência do problema anterior, implementamos um problema grande, que é do mesmo tipo descrito em 5.4. Neste novo problema usamos $T = 50$ e com isso as variáveis de controle passam a ser 50 para cada espécie e 150 no total. Assim, o novo problema passa a ter 300 variáveis e 150 restrições tipo $x_i \geq 0$ (canalizações).

O ponto inicial fornecido foi gerado, como no caso anterior, por uma sub-rotina que fornece um ponto inicial bom. Esta sub-rotina (a mesma aplicada ao problema anterior), gera uma política para o problema. A política gerada(sempre satisfazendo as restrições de igualdade) pode ser super-ótima ou sub-ótima, conforme os valores das variáveis de controle. A política será sub-ótima se satisfizer todas as restrições de desigualdade e ainda assim pudermos "pescar" mais sem violar nenhuma delas. A política será super-ótima quando"pescamos" acima do possível e violamos restrições do tipo $y_i(t) \geq 0$ (pescamos além do que existe disponível). Experiências mostraram que os dois problemas implementados apresentam muitos máximos locais e, em geral, se forneclamos políticas sub-ótimas o algoritmo convergia para pontos que evidentemente não eram uma política boa, mas satisfaziam a precisão pedida e as condições do teste de convergência. Porém, ao fornecermos uma política super-ótima, o algoritmo convergia a uma política "factível" que deveria ser uma "boa" política. Não queremos dizer que a política assim obtida, seja a <u>melhor</u>, pois como já dissemos, o problema deve apresentar muitos máximos.

Voltamos a repetir que não estivemos interessados em considerar fatores para a aplicabilidade do problema e nosso objetivo era a de testar o algoritmo para um problema grande. Bem sabemos que a dificuldade de se conseguir dados e um modelo para representar a realidade é evidente. O modelo apresentado, bem simplificado, atendeu aos nosso objetivos.

Mostramos abaixo, o ponto inicial fornecido (uma política super-ótima), e os resultados obtidos. Como no caso anterior, trabalhamos com minimização e a função objetivo tem, portanto, sinal negativo; também como no problema anterior, a função objetivo nos fornece o número de unidades pescadas.

50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000
50.00000	50.00000	50.00000	50.00000	50.00000

9[illegible]2.5906	962.8921	941.5097	918.6307	894.4853
8[illegible]9.2626	843.1168	816.1713	788.5239	760.2492
7[illegible]1.4925	702.0219	672.1301	641.7369	610.8397
5[illegible]9.4253	547.4702	514.9417	481.7978	447.9876
4[illegible]3.4511	373.1194	341.9139	304.7160	266.5159
2[illegible]7.1122	180.4103	144.2710	100.5388	55.03997
7.579916	-42.05445	-31.12299	-148.8860	-206.6586
-2[illegible]7.7901	-332.0719	-491.7578	-475.5399	-554.5849
-6[illegible]9.5235	-731.9565	-829.9528	-837.0391	-1053.175
-1[illegible]7[illegible].201	-1315.455	-1163.532	-1622.275	-1799.785

continua

continuação

30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000

9[illegible]6.0700	840.6623	771.0503	707.0457	647.9703
5[illegible]3.2347	542.3228	494.7801	450.2036	408.2342
3[illegible]8.5501	330.8698	294.4027	260.4347	227.2343
1[illegible]5.0945	163.8203	133.2262	103.1333	73.36660
4[illegible].75220	14.11521	-15.72412	-45.95256	-76.76777
-1[illegible]8.3325	-141.0265	-174.9568	-210.1587	-247.8557
-2[illegible]7.5182	-329.8745	-375.4245	-424.7574	-478.5735
-5[illegible]7.7121	-603.1834	-676.2423	-758.3937	-851.5325
-9[illegible]8.0235	-1033.855	-1223.338	-1391.835	-1591.390
-1[illegible]3[illegible].770	-2121.217	-2377.782	-2920.917	-3473.681

30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000
30.00000	30.00000	30.00000	30.00000	30.00000

9[illegible]8.0[illegible]00	8[illegible]1.1[illegible]07	8[illegible]6.8303	794.8239	757.1962
7[illegible]3.2172	692.3079	654.0928	637.9235	613.7582
5[illegible]1.2463	579.1757	559.3599	531.6415	513.8873
4[illegible]6.9311	[illegible]45.8216	455.3291	450.3982	435.9862
4[illegible]2.0215	4[illegible]3.4478	395.2140	382.2729	369.5813
3[illegible]7.0965	341.7455	332.6098	320.5305	309.5175
2[illegible]6.5361	281.5521	272.5321	260.4405	218.2394
2[illegible]5.8[illegible]	223.3507	210.5723	197.5028	134.0817
1[illegible]0.2368	155.8[illegible]09	133.9373	125.2545	108.6871
91.13590	72.0[illegible]	51.34983	[illegible]8.48508	2.76958[illegible]

VALORES DOS CONTROLES E ESTADOS

U1(0)=	50.	Y1(1)=	992.	U2(0)=	30.	Y2(1)=	917.
U1(1)=	50.	Y1(2)=	963.	U2(1)=	30.	Y2(2)=	841.
U1(2)=	50.	Y1(3)=	941.	U2(2)=	30.	Y2(3)=	771.
U1(3)=	50.	Y1(4)=	919.	U2(3)=	30.	Y2(4)=	707.
U1(4)=	50.	Y1(5)=	894.	U2(4)=	30.	Y2(5)=	648.
U1(5)=	50.	Y1(6)=	869.	U2(5)=	30.	Y2(6)=	593.
U1(6)=	50.	Y1(7)=	843.	U2(6)=	30.	Y2(7)=	542.
U1(7)=	50.	Y1(8)=	815.	U2(7)=	30.	Y2(8)=	495.
U1(8)=	50.	Y1(9)=	788.	U2(8)=	30.	Y2(9)=	450.
U1(9)=	50.	Y1(10)=	760.	U2(9)=	30.	Y2(10)=	408.
U1(10)=	50.	Y1(11)=	731.	U2(10)=	30.	Y2(11)=	369.
U1(11)=	50.	Y1(12)=	702.	U2(11)=	30.	Y2(12)=	331.
U1(12)=	50.	Y1(13)=	672.	U2(12)=	30.	Y2(13)=	295.
U1(13)=	50.	Y1(14)=	642.	U2(13)=	30.	Y2(14)=	260.
U1(14)=	50.	Y1(15)=	611.	U2(14)=	30.	Y2(15)=	227.
U1(15)=	50.	Y1(16)=	579.	U2(15)=	30.	Y2(16)=	195.
U1(16)=	50.	Y1(17)=	547.	U2(16)=	30.	Y2(17)=	164.
U1(17)=	50.	Y1(18)=	515.	U2(17)=	30.	Y2(18)=	133.
U1(18)=	50.	Y1(19)=	482.	U2(18)=	30.	Y2(19)=	104.
U1(19)=	50.	Y1(20)=	448.	U2(19)=	29.	Y2(20)=	75.
U1(20)=	50.	Y1(21)=	413.	U2(20)=	28.	Y2(21)=	47.
U1(21)=	50.	Y1(22)=	378.	U2(21)=	24.	Y2(22)=	24.
U1(22)=	50.	Y1(23)=	342.	U2(22)=	15.	Y2(23)=	10.
U1(23)=	50.	Y1(24)=	305.	U2(23)=	5.	Y2(24)=	4.
U1(24)=	50.	Y1(25)=	267.	U2(24)=	2.	Y2(25)=	2.
U1(25)=	50.	Y1(26)=	227.	U2(25)=	1.	Y2(26)=	2.
U1(26)=	50.	Y1(27)=	187.	U2(26)=	0.	Y2(27)=	2.
U1(27)=	50.	Y1(28)=	146.	U2(27)=	-0.	Y2(28)=	2.

[illegible]1(28)=	[illegible]8.	Y1(29)=	1[illegible].	[illegible]2(28)=	-0.	Y2(29)=	2.
[illegible]1(29)=	45.	Y1(30)=	6[illegible].	[illegible]2(29)=	-0.	Y2(30)=	2.
[illegible]1(30)=	37.	Y1(31)=	23.	[illegible]2(30)=	-0.	Y2(31)=	2.
[illegible]1(31)=	18.	Y1(32)=	13.	[illegible]2(31)=	-0.	Y2(32)=	2.
[illegible]1(32)=	7.	Y1(33)=	7.	[illegible]2(32)=	-0.	Y2(33)=	2.
[illegible]1(33)=	3.	Y1(34)=	5.	[illegible]2(33)=	0.	Y2(34)=	3.
[illegible]1(34)=	1.	Y1(35)=	4.	[illegible]2(34)=	-0.	Y2(35)=	3.
[illegible]1(35)=	0.	Y1(36)=	4.	[illegible]2(35)=	-0.	Y2(36)=	3.
[illegible]1(36)=	0.	Y1(37)=	4.	[illegible]2(36)=	-0.	Y2(37)=	3.
[illegible]1(37)=	0.	Y1(38)=	4.	[illegible]2(37)=	-0.	Y2(38)=	4.
[illegible]1(38)=	-0.	Y1(39)=	5.	[illegible]2(38)=	-0.	Y2(39)=	4.
[illegible]1(39)=	-0.	Y1(40)=	5.	[illegible]2(39)=	-0.	Y2(40)=	4.
[illegible]1(40)=	-0.	Y1(41)=	5.	[illegible]2(40)=	-0.	Y2(41)=	5.
[illegible]1(41)=	0.	Y1(42)=	5.	[illegible]2(41)=	-0.	Y2(42)=	5.
[illegible]1(42)=	-0.	Y1(43)=	6.	[illegible]2(42)=	-0.	Y2(43)=	6.
[illegible]1(43)=	-0.	Y1(44)=	6.	[illegible]2(43)=	-0.	Y2(44)=	7.
[illegible]1(44)=	-0.	Y1(45)=	7.	[illegible]2(44)=	-0.	Y2(45)=	7.
[illegible]1(45)=	0.	Y1(46)=	7.	[illegible]2(45)=	-0.	Y2(46)=	8.
[illegible]1(46)=	0.	Y1(47)=	7.	[illegible]2(46)=	-0.	Y2(47)=	9.
[illegible]1(47)=	0.	Y1(48)=	3.	[illegible]2(47)=	-0.	Y2(48)=	10.
[illegible]1(48)=	-0.	Y1(49)=	3.	[illegible]2(48)=	1.	Y2(49)=	10.
[illegible]1(49)=	3.	Y1(50)=	6.	[illegible]2(49)=	5.	Y2(50)=	6.

U3(0)=	30.	Y3(1)=	938.
U3(1)=	30.	Y3(2)=	884.
U3(2)=	30.	Y3(3)=	836.
U3(3)=	30.	Y3(4)=	794.
U3(4)=	30.	Y3(5)=	757.
U3(5)=	30.	Y3(6)=	723.
U3(6)=	30.	Y3(7)=	692.
U3(7)=	30.	Y3(8)=	663.
U3(8)=	30.	Y3(9)=	637.
U3(9)=	30.	Y3(10)=	613.
U3(10)=	30.	Y3(11)=	591.
U3(11)=	30.	Y3(12)=	570.
U3(12)=	30.	Y3(13)=	550.
U3(13)=	30.	Y3(14)=	531.
U3(14)=	30.	Y3(15)=	513.
U3(15)=	30.	Y3(16)=	496.
U3(16)=	30.	Y3(17)=	480.
U3(17)=	30.	Y3(18)=	465.
U3(18)=	30.	Y3(19)=	450.
U3(19)=	30.	Y3(20)=	435.
U3(20)=	30.	Y3(21)=	421.
U3(21)=	30.	Y3(22)=	408.
U3(22)=	30.	Y3(23)=	395.
U3(23)=	30.	Y3(24)=	382.
U3(24)=	30.	Y3(25)=	369.
U3(25)=	30.	Y3(26)=	356.
U3(26)=	30.	Y3(27)=	344.
U3(27)=	29.	Y3(28)=	331.
U3(28)=	29.	Y3(29)=	319.
U3(29)=	29.	Y3(30)=	307.
U3(30)=	29.	Y3(31)=	295.
U3(31)=	29.	Y3(32)=	283.
U3(32)=	28.	Y3(33)=	270.
U3(33)=	28.	Y3(34)=	258.
U3(34)=	28.	Y3(35)=	245.
U3(35)=	27.	Y3(36)=	233.
U3(36)=	27.	Y3(37)=	220.
U3(37)=	27.	Y3(38)=	207.
U3(38)=	26.	Y3(39)=	193.
U3(39)=	26.	Y3(40)=	180.
U3(40)=	26.	Y3(41)=	166.
U3(41)=	26.	Y3(42)=	151.
U3(42)=	25.	Y3(43)=	136.
U3(43)=	25.	Y3(44)=	120.
U3(44)=	25.	Y3(45)=	103.
U3(45)=	25.	Y3(46)=	85.
U3(46)=	25.	Y3(47)=	66.
U3(47)=	25.	Y3(48)=	46.
U3(48)=	26.	Y3(49)=	23.
U3(49)=	25.	Y3(50)=	0.

Mostramos abaixo alguns dados do problema dos peixes II. Observe o tempo de C.P.U. e o número de problemas parciais; a média

```
NUMERO TOTAL DE AVALIACOES DE FUNCAO E GRADIENTE :   55
NUMERO TOTAL DE ITERACOES DENTRO DA MINSS :   25
NUMERO DE CHAMADAS A MINSS (PROBLEMAS PARCIAIS ) :  4

DIAGNOSTICO : 1    MAIOR INFACTIBILIDADE :   0.4851367
EPS :    0.1000000E-01    TOL :    0.1000000E-01    FATOR :    10.00000



TEMPO TOTAL DE CPU PARA O EXEMPLO   2  -->        332.9760000 SEGUNDOS
   VALOR DA FUNCAO OBJETIVO :    -3667.683
   VALOR DA FUNCAO LAGRANGEANO AUMENTADO :    -3629.927
```

de tempo por problema parcial não chega a 1 minuto e meio. O diagnóstico 1 não significa um mau resultado, pois a maior infactibilidade é suficientemente pequena para um problema como este.

5.7 - CONSIDERAÇÕES SOBRE O DESEMPENHO DO ALGORITMO - Diversas experiências foram realizadas com os problemas dos peixes e, com o aumento da precisão, o tempo de C.P.U. aumentava drasticamente. Com a necessidade de pouca precisão nos problemas, usamos $\varepsilon = 0{,}01$ o suficiente. Observando ambos os problemas, chegamos a conclusão que a última pescaria é "fulminante" e deveremos ter para últimos estados (últimas componentes de y_1, y_2, y_3) valores nulos. No nosso problema isto praticamente se verificou já que os últimos estados ou são nulos ou perto disto. O problema de erros de arredondamento também deve ser considerado, já que a função aumentada contém o seguinte esquema de parcelas para o problema:

$$(300 \times 6) \text{ parcelas} + (300 \text{ parcelas})^2 +$$

$$+(300 \text{ parcelas}) + (300 \text{ parcelas})^2 \text{ aprox.}$$

Este número é o máximo de parcelas, podendo a função ter menos parcelas. Mas, ao começar o processo (com uma política super-ótima) todas estas parcelas aparecem na função aumentada. Assim o erro poderá ser grande; porém, como não se usou o processo recursivo (é claro, queríamos evitar isso), e cada novo ponto obtido como produto de um problema parcial é recalculado, os erros não se propagam além do cálculo da função. Ou melhor: o cálculo de uma função aumentada não é recursivo.

O desempenho do algoritmo chega a ser excelente quando o testamos para um ponto inicial muito ruim e assim mesmo, ele forne ceu uma boa política. Este ponto inicial foi $y_1(t) = y\ (t) = y_3(t)$ $= u_1(t-1) = u_2(t-1) = u_3(t-1) = 600$, $t = 1,\ldots,T$. Observe que todas as variáveis iniciam com valor 600, o que é bem diferente de uma política. Neste caso, permitimos um número máximo de 25 problemas parciais; todos os 25 problemas parciais foram realizados, tendo um tempo total de cerca de 19 minutos de C.P.U., ou seja, aproximadamente 45 segundos por iteração. Porém, o fato mais importante é que no terceiro problema parcial, o ponto obtido já era suficientemente bom, embora não atingisse a precisão de 0,01. Observe que o ponto inicial é bastante longe da solução, mas, mesmo assim, não causou impedimento para o algoritmo. O número total de avaliações de função e gradiente (para os 25 problemas parciais) foi 160 e o número total de iterações dentro da sub-rotina de minimização sem restrições foi de 64. Mostramos a seguir a política obtida; observe que é uma política factível e que não apresenta todos os estados nulos no final; voltamos a achar que o problema deve ter diversos máximos locais ou pontos de não-regularidade em que seja impossível para o algoritmo evitá-los.

A seguir mostramos a política obtida para este caso. O custo da política (função objetivo) é de 3585 unidades.

[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Y2(1)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	Y2(7)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	Y2(11)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	Y2(13)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	Y2(15)=	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	Y2(16)=	591.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	591.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	591.
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	3.	[illegible]	[illegible]

H3(0)=	457.	Y3(1)=	[illegible]	H3(28)=	0.	Y3(29)=	-1.
H3(1)=	217.	Y3(2)=	[illegible]	H3(29)=	0.	Y3(30)=	-0.
H3(2)=	126.	Y3(3)=	125.	H3(30)=	0.	Y3(31)=	-0.
H3(3)=	57.	Y3(4)=	[illegible]	H3(31)=	0.	Y3(32)=	-0.
H3(4)=	55.	Y3(5)=	32.	H3(32)=	0.	Y3(33)=	-0.
H3(5)=	19.	Y3(6)=	16.	H3(33)=	0.	Y3(34)=	-0.
H3(6)=	9.	Y3(7)=	8.	H3(34)=	0.	Y3(35)=	-0.
H3(7)=	4.	Y3(8)=	4.	H3(35)=	0.	Y3(36)=	-0.
H3(8)=	2.	Y3(9)=	3.	H3(36)=	0.	Y3(37)=	-0.
H3(9)=	1.	Y3(10)=	2.	H3(37)=	0.	Y3(38)=	-0.
H3(10)=	1.	Y3(11)=	1.	H3(38)=	0.	Y3(39)=	-0.
H3(11)=	0.	Y3(12)=	1.	H3(39)=	0.	Y3(40)=	-0.
H3(12)=	0.	Y3(13)=	1.	H3(40)=	0.	Y3(41)=	-0.
H3(13)=	0.	Y3(14)=	0.	H3(41)=	0.	Y3(42)=	-0.
H3(14)=	0.	Y3(15)=	0.	H3(42)=	0.	Y3(43)=	-0.
H3(15)=	0.	Y3(16)=	0.	H3(43)=	0.	Y3(44)=	-0.
H3(16)=	0.	Y3(17)=	0.	H3(44)=	0.	Y3(45)=	-0.
H3(17)=	0.	Y3(18)=	0.	H3(45)=	0.	Y3(46)=	-0.
H3(18)=	0.	Y3(19)=	0.	H3(46)=	-0.	Y3(47)=	-0.
H3(19)=	0.	Y3(20)=	0.	H3(47)=	-0.	Y3(48)=	-0.
H3(20)=	0.	Y3(21)=	0.	H3(48)=	-0.	Y3(49)=	0.
H3(21)=	0.	Y3(22)=	0.	H3(49)=	0.	Y3(50)=	0.
H3(22)=	0.	Y3(23)=	0.				
H3(23)=	0.	Y3(24)=	0.				
H3(24)=	0.	Y3(25)=	0.				
H3(25)=	0.	Y3(26)=	0.				
H3(26)=	0.	Y3(27)=	0.				
H3(27)=	0.	Y3(28)=	-0.				

CONCLUSÃO

Neste trabalho, obtivemos resultados acima dos esperados, e nossos objetivos foram atingidos plenamente.

As comparações entre os métodos foram bastante satisfatórias. Os resultados do Problema dos Peixes II, um problema grande, foram, podemos dizer, excelentes.

A utilização de uma rotina para minimização sem restrições que utilize a informação de canalizações melhoraria, em muito, a eficiência dos algoritmos. As sub-rotinas que usamos tratam das canalizações como restrições propriamente ditas. Propomos, a partir de nossas experiências neste trabalho, que os interessados que queiram melhorar o desempenho dos algoritmos, utilizem uma sub-rotina de minimizações sem restrições com canalizações.

BIBLIOGRAFIA

[1] Asaadi,J. A computational comparison of some non-linear programs. *Mathematical programming*, *4*(2):144-54, 1973.

[2] Avriel,M. *Nonlinear programming;* analysis and methods. Englewood Cliffs, Prentice-Hall, 1976.

[3] Diniz,M.A.B. *Algoritmos para minimização de funções com restrições não lineares.* Campinas, IMECC/UNICAMP, 1980. (Tese de Mestrado).

[4] Fletcher,R. An Ideal penalty function for constrained optimization. *Journal of the Institute of mathematics and its applications*, *15*(3):319-42, 1975.

[5] ___________ An exact penalty function for nonlinear programming with inequality constraints. *Mathematical programming*, *5*(2): 129-50, 1973.

[6] Hadley,G. *Nonlinear and dynamic programming.* Reading, Addison Wesley, 1964.

[7] Hestenes,M.R. Multiplier and gradient methods. *Journal of optimization theory and its applications*, *4*(5):303-20,1973.

[8] Jittorntrum,K. Accelerated convergence for the Powell/Hestenes multiplier method. *Mathematical programming*,*18*(2):197-214, 1980.

[9] Luenberger,D.G. *Introdu tion to linear and nonlinear programming.* Reading, Addison-Wesley, 1973.

[10] Pielou, E.C. *Mathematical ecology.* New York, Wiley, 1977.

[11] Powell,M.J.D. A method for nonlinear constraints in minimi zation. In: Fletcher,R.,ed. - *Optimization.* London, Aca demic, 1969. p.283-98.

[12] Rockafellar, R.T. The multiplier method of Hestenes and Po- well applied to convex programming. *Journal of optimiza- tion theory and its applications, 12*(6):555-62,1973.

APÊNDICE A

APÊNDICE A

Aqui daremos algumas definições importantes e apresentaremos alguns resultados sem demonstrações. Caso o leitor necessite de alguma demonstração, poderá te-la nas bibliografias fornecidas.

DEFINIÇÃO 1 - Um ponto x^* que satisfaz as restrições h e g,ou seja, $h_i(x^*) = 0$ e $g_i(x^*) \leq 0$ é dito ponto regular para estas restrições se os gradientes $\nabla h_i(x^*)$, $\nabla g_j(x^*)$, $i = 1,2,\ldots,\ell$ e $j \in J$, $J = \{j \mid g_j(x^*) = 0\}$, são linearmente independentes.

DEFINIÇÃO 2 - Definimos função lagrangeano a função $L(x,\lambda,\theta)$: $R^{n+m+\ell} \rightarrow R$ associada ao problema geral (P), que tem a seguinte expressão:

$$L(x,\lambda,\rho) = f(x) + \lambda^T g(x) + \rho^T h(x)$$
$$= f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{\ell} \rho_i h_i(x)$$

onde os vetores λ e ρ são chamados vetores dos multiplicadores de Lagrange.

TEOREMA 1[3] - (Condições de Kuhn-Tucker): Se x^* é um ponto mínimo local para o problema (P) e se x^* é um ponto regular para as restrições de (P), então existe um vetor $\lambda \in R^m$ e um vetor $\rho \in R^\ell$ com $\lambda_i \geq 0$, $i= 1,2,\ldots,m$ tais que

$$f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) + \sum_{i=1}^{\ell} \rho_i \nabla h_i(x^*) = 0 \quad \text{e} \quad \lambda^T g(x^*) = 0$$

Assim, uma condição <u>necessária</u> para que x* seja um mínimo local, é que cumpra as condições de Kuhn-Tucker acima; estas condições são conhecidas como condições de Kuhn-Tucker de primeira ordem.

<u>TEOREMA 2 [3]</u>- (das condições suficientes de segunda ordem): Consições suficientes para que x^*, ponto regular, seja um ponto mínimo local para o problema (P), são que existam $\lambda \in R^m$ e $\theta \in R^\ell$ tais que as condições do Teorema 1 sejam satisfeitas e a matriz $\nabla^2 L(x^*,\lambda^*,\theta^*)$ (hessiano do lagrangeano) abaixo, seja definida positiva no subespaço T dado mais abaixo:

$$\nabla^2 L(x^*,\lambda^*,\rho^*) = \nabla^2 f(x^*) + \lambda^T \ \nabla^2 g(x^*) + \rho^{*T} \nabla^2 h(x^*)$$

$$T = \{y: \nabla h_i(x^*)^T y = 0 \ , \ i=1,2,\dots,\ell, \ \nabla g_i(x^*)^T y = 0$$

para todo $j \in J'\}$

$$J' = \{j : g_j(x^*) = 0, \ \lambda_j > 0 \}$$

No caso de o problema apresentar somente restrições de igualdade o lagrangeano apresenta-se como:

$$L(x,\rho) = f(x) + \rho^T h(x)$$

e no Teorema 2, o subespaço T seria

$$T = \{y: \nabla h_i(x^*)^T y = 0 \ , \ i= 1,2,\dots, \ell\} .$$

APÊNDICE B

APÊNDICE B

DESCRIÇÃO DAS SUB-ROTINAS

Aqui descreveremos as sub-rotinas usadas no nosso trabalho. Aqui faremos um trabalho breve; não entraremos em pormenores sobre o uso das sub-rotinas, pois, no apêndice C(listagens) são incluidos os programas principais, que servem como exemplo para o uso.

B.1 - SUB-ROTINAS PARA O MÉTODO DE PENALIDADE -

PENA - é a sub-rotina global do método.

- argumentos -

X - vetor ponto inicial de dimensão N .

N - número de variáveis .

M - número de restrições de desigualdade.

L - número de restrições de igualdade.

F - valor da função objetivo na saída.

GFO - vetor gradiente da função objetivo na saída (N componentes.

Q - valor da função penalidade na saída.

AMI - vetor de parâmetros de penalização, na entrada deve ser fornecido com todas as componentes maiores que zero. Na saida é a última atualização que recebeu.

GNORM - parâmetro para efeito de diagnósticos, é a norma euclidiana ao quadrado para as restrições de desigualdade que são violadas pelo ponto x.

HNORM - o mesmo que GNORM para restrições de igualdade.

IDEV - número da máquina para saída dos resultados.

MXF - número máximo de problemas parciais permitidos.

FATOR - fator de aumento dos parâmetros de penalização. É parâmetro de entrada. Em geral FATOR = 10.

KONT - contador de problemas parciais.

IER - diagnóstico, IER = Ø significa que houve convergência para um ponto que satisfaz ER < EPS; IER = 1 significa que a função objetivo já não pode melhorar entre duas iterações consecutivas(só para IK ≠ 0); IER = 2 significa que o número máximo de iterações foi atingido sem que ER fosse menor que EPS.

EPS - tolerância pedida para convergência.

IOUT - se IOUT = Ø a sub-rotina MINSR nada imprime; se IOUT = k > Ø a sub-rotina MINSR imprimirá a cada k iterações completas.

KFUN - número total de avaliações de função e gradiente realizadas dentro da sub-rotina MINSR.

KTER - número total de iterações completas dentro da sub-rotina MINSR.

GRAG - vetor para trabalho de M componentes, que conterá o gradiente de uma restrição de desigualdade.

GRAH - o mesmo que GRAG com L componentes, para restrição de igualdade.

GRAQ - vetor gradiente da função penalidade.

MXFUN - número máximo de avaliações de função e gradiente permitidas dentro da sub-rotina MINSR.

W - vetor de trabalho para a sub-rotina MINSR, a ser dimensionado com no mínimo (5N+2) posições.

GAUX - vetor que conterá os valores das restrições de desigualdade, deve ser dimensionado com M componentes.

HAUX - o mesmo que GAUX para restrições de igualdade, deve ser dimensionado com L componentes.

ICH - vetor de chaves para passagem da informação se $g_i(x)$ é ou não violada. Se $g_i(x)$ é violada ICH(I) = Ø, caso contrário ICH(I) = 1. Deve ter dimensão M.

ER - maior infactibilidade, serve para testar a convergência e na saída serve para se verificar em quanto foi o valor da norma máximo das restrições que são violadas pelo ponto.

IMP - se IMP = Ø nada é impresso pela sub-rotina PENA (podendo ou não ter impressão pela sub-rotina MINSR); se IMP = 1 são impressos IER, KONT, F, X, KFUN, KTER, ER, somente; se IMP = 2, IMPRI é chamada e uma impressão detalhada é fornecida, sendo impressas quase a totalidade das variáveis de saída.

IK - se IK = 0 não é feito o teste de melhora ou não da função objetivo em relação a função Q. Se IK = 1 o teste é feito.

Sub-rotinas que são chamadas pela sub-rotina PENA:

a) MINSR - tem a finalidade de fazer a minimização sem restrições da função penalidade. Esta sub-rotina implementa um método semelhante a gradientes conjugados com uso de pseudos-hessianos.

Chamada da sub-rotina:

CALL MINSR(N, M, L, X, Q, GRAG, AMI, IFUN, ITER, EPSG, NFLAG , MXFUN, F, GFO, W, IOUT, IDEV, ACC, GNORM, HNORM, GAUX, HAUX, GRAG, GRAH, ICH)

- argumentos não descritos na sub rotina PENA -

IFUN - número de avaliações de função e gradiente realizadas nesta sub-rotina.

ITER - número de iterações completas realizadas nesta sub-rotina.

EPSG - é a tolerância pedida para convergência na minimização sem restrições.

NFLAG - diagnóstico da MINSR. Se NFLAG = ∅ houve convergência; se NFLAG = 1 houve falha na busca unidimensional do método, não se conseguindo melhorar o valor da função ; se NFLAG = 2 o número máximo de avaliações de função e gradiente foi atingido sem que houvesse sido atingida a tolerância EPSG.

ACC - uma aproximação para a precisão da máquina em ponto flutuante. No PDP-10 da UNICAMP, ACC = 2^{-27}.

b) ATUAMI- tem a finalidade de atualizar os parâmetros de penalização.

Chamada:

CALL ATUAMI (AMI, M, L, EPS, FATOR, ICH, HAUX)

c) IMPRI - tem a finalidade de imprimir os resultados finais obtidos

Chamada:

CALL IMPRI(N, M, L, X, F, GFO, Q, AMI, FATOR, KONT, IER, EPS, KFUN, KTER, GAUX, HAUX, GRAQ, ER, IDEV)

d) FPEN - calcula o valor da função penalidade e seu gradiente.

Chamada:

CALL FPEN(X,N,M, L, Q, GRAQ, AMI, AGNORM, HNORM, F, GFO, GAUX, HAUX, GRAG, GRAH, ICH)

e) FUN, GE, HAGA - são 3 sub-rotinas que compõem o conjunto de problemas testes. A sub-rotina FUN calcula uma especificada função objetivo ou o seu gradiente; a sub-rotina GE calcula uma especificada restrição de desigualdade ou seu gradiente; a sub-rotina HAGA é semelhante a GE, só que para restrições de igualdade.

Chamadas:

CALL FUN (X, F, GF, L)

CALL GE (I, X, G, GRAG, L)

CALL HAGA (I, X, H, GRAH, L)

- argumentos -

X - ponto fornecido.

F - valor de uma função objetivo.

GF - gradiente de uma função objetivo.

L - se L =1 a função é calculada; se L = 2 o gradiente da função é calculada.

I - é a i-ésima restrição considerada.

G - valor da restrição de desigualdade.

GRAG - gradiente da restrição de desigualdade.

H - valor da restrição de igualdade.

GRAH - gradiente da restrição de igualdade.

Área em COMMON; COMMON/NOPROB/NP

Se NP = i, o i-ésimo problema teste está sendo considerado.

B.2 - SUB-ROTINAS PARA O MÉTODO LAGRANGEANO AUMENTADO

PNLØ1 - é a sub-rotina global do método.

- argumentos -

X - mesmo que na sub-rotina PENA.

N - mesmo que na sub-rotina PENA.

M - mesmo que na sub-rotina PENA.

L - mesmo que na sub-rotina PENA.

XLAM - vetor aproximação para os multiplicadores de Lagrange; na entrada, se não é fornecido, a sub-rotina assume o valor 0,01 para todas as componentes de XLAM, se alguma componente apresenta valor maior que 0,01, esse valor é assumido. Deve ser M-dimensionado.

XSIG - vetor de parâmetros para os termos de segundo grau das restrições de desigualdade. Deve ser M-dimensioando.

TETA - o mesmo que XLAM para restrições de igualdade. Pode , suas componentes, assumir qualquer valor real. Deve ser L-dimensioando.

XMI - o mesmo que XSIG para restrições de igualdade. Deve ter dimensão L.

FOBJ - valor da função objetivo da saída.

FPSI - valor da função lagrangeano aumentado na saída.

GX - vetor de trabalho de dimensão M.

HX - vetor de trabalho de dimensão L.

GFO - gradiente da função objetivo na saída.

W - vetor de trabalho. Deve ter no mínimo (5N+2) posições

GRAG - vetor de trabalho que conterá um gradiente de uma restrição de desigualdade.

GRAH - o mesmo que GRAG para uma restrição de igualdade.

AK - vetor de atualização dos parâmetros XLAM.

KFUN - número total de avaliações de função e gradiente na saída. É o somatório dos IFUN.

EPS - precisão pedida para convergência.

MXFUN - o mesmo que na sub-rotina MINSR.

MAXF - número máximo de problemas parciais permitidos.

IOUT - o mesmo que na sub-rotina MINSR.

ACC - o mesmo que na sub-rotina MINSR.

TOL - se uma restrição de desigualdade for menor que TOL, esta restrição não passrá pelo teste de razão de melhora; se uma restrição de igualdade em módulo for menor que TOL, esta não passará pelo teste acima.

IDEV - o mesmo que na sub-rotina MINSR.

KONT - o mesmo que na sub-rotina PENA.

KTER - o mesmo que na sub-rotina PENA.

IER - se IER = Ø o método convergiu para um ponto em que a sua maior infactibilidade é menor que EPS; se IER = 1 foi atingido o número máximo de problemas parciais permitidos sem obtenção da precisão desejada; se IER = 2 o problema só contém restrições de desigualdade e o ponto obtido é factível(só para IOP ≠ Ø); se IER = 3 não houve melhora na função objetivo em duas iterações consecutivas (só para ID ≠ Ø); se IER = 1 o problema é um caso particular em que M = L = Ø e sua convergência deve ser analisada pelo parâmetro NFLAG da sub-rotina MINSS.

GAUX - vetor que contém os valores das restrições de desigualdade. Deve ser M-dimensionado.

HAUX - o mesmo que GAUX para restrições de igualdade. Deve ter dimensão L.

FATOR - fator de multiplicação dos parâmetros XSIG para atualização.

EPSG - é a tolerância para a sub-rotina MINSS. No primeiro problema parcial EPSG = min { Ø,1; 10xEPS } após isso EPSG assumirá o valor EPS; para ter um EPSG diferente do especificado, a linha EPSG = AMIN 1(Ø.1, 10*EPS) deve ser removida.

IMP - se IMP = 0 nada é impresso; se IMP = 1 cada iteração é impressa com ponto obtido, o valor da função objetivo e resultados finais detalhados; se IMP = 2 cada iteração é impressa com bastante detalhes mais os resultados finais detalhados.

ID - se ID = Ø o teste de melhora da função objetivo não é feito; se ID ≠ 0 o teste é feito.

Área em COMMON : COMMON/SFACTI/IOP. (Veja IER)

Sub-rotinas que são chamadas pela sub-rotina PNLØ1:

a) MINSS - é a mesma que MINSR para o método de penalidade.

Chamada da sub-rotina:

CALL MINSS(X, N, M, L, FPSI, GPSI, XLAM, XSIG, TETA, XMI, IFUN, ITER, EPSG, NFLAG, MXFUN, FOBJ, GFO, W, IOUT, IDEV, ACC, GRAG, GRAH, GAUX, HAUX)

b) PSI - tem a finalidade de clacular a função lagrangeano aumentado e seu gradiente.

Chamada:

CALL PSI(X, N, M, L, XLAM, XSIG, TETA, XMI, FPSI, GPSI, GFO, GRAG, GRAH, FOBJ, GAUX, HAUX)

c) ITERA - tem a finalidade de imprimir os resultados finais do processo.

Chamada:

CALL ITERA (X, N, M, L, KONT, FOBJ, FPSI, GPSI, HAUX, GAUX, IFUN, ITER, NFLAG, ER, IDEV, IMP)

d) SFINAL- tem a finalidade de imprimir os resultados finais do processo.

Chamada:

CALL SFINAL (X, N, M, L, FOBJ, EPSI, GFO, GAUX, HAUX, XLAM, TETA, KFUN, KTER, KONT, IER, ER, EPS, TOL, FATOR, IMP, IDEV)

e) FNORMA- é uma função real de argumentos A e N, que tem a finalidade de calcular a norma infinito de um vetor A de N componentes.

f)FUN, - tem a mesma finalidade já descrita em 6.2.
GE,
HAGA

B.3 - PROGRAMAS PARA IMPLEMENTAÇÃO DOS PROBLEMAS DOS PEIXES - Além dos programas principais, no apêndice das listagens, encontram-se dois arquivos com sub-rotinas tipo FUN, GE e HAGA para os dois problemas dos peixes; duas sub-rotinas suplementares são também incluidas, a sub-rotina P1 de parâmetros N e X que tem a finalidade de criar uma política para os problemas ;a sub-rotina SA1 de parâmetros N,X,IT com a finalidade de imprimir no formato F7.Ø os valores dos controles e estados obtidos ao final do processo no problema dos peixes. A variável IT especifica o número de estados; se IT = 5 é o problema dos peixes I; se IT = 50 é o problema dos peixes II.

B.4 - ALGUMAS OBSERVAÇÕES IMPORTANTES - No apêndice C , das listagens, encontram os mais importantes programas principais usados. Não é nosso objetivo mostrar os seus usos. Queremos mostrar resultados obtidos e poder oferecer a interessados uma série de programas que podem ser úteis como ferramenta na solução de problemas de otimização.

APÊNDICE C

```
C       PROGRAMA PRINCIPAL PARA MINIMIZACAO POR FUNCAO PENALIDADE
C
        DIMENSION X(500),GFO(500),AMI(1000),GRAG(500),GRAH(500),GRAO(500)
          1     ,GAUX(500),W(2502),HAUX(500),ICH(500),Y(100)
        COMMON/NOPROB/NP
          DATA IK/0/
          DATA IOUT,MAFUN,IDEV,IXE/3,600,22,10/
        EPS=1.E-4
          FATOR = 10.
          IMP = 2
77        DO 66 I=1,(5*N + 2)
66        W(I) = 0.
        TYPE 1
1       FORMAT(5X,'NUMERO DO EXEMPLO :',$)
        ACCEPT 2,NP
2       FORMAT(I)
        TYPE 3
3       FORMAT(5X,'NUMERO DE VARIAVEIS, NUMERO DE RESTRICOES DE DESIGUALDA
       *DES ',/,20X,'E NUMERO DE RESTRICOES DE IGUALDADE :',$)
        ACCEPT 4,N,M,L
4       FORMAT(3I)
        TYPE 5
5       FORMAT(5X,'VALOR DE MI INICIAL : ',$)
        ACCEPT 7,XMI
7       FORMAT(G)
          DO 13 I=1,(M+L)
13        AMI(I) = XMI
        TYPE 9
9       FORMAT(5X,' PONTO INICIAL : ',$)
          DO 55 I=1,N
55        ACCEPT 7, X(I)
          CALL GETRUN(0)
          CALL PENA(X,N,M,L,F,GFO,G,AMI,GNORM,HNORM,IDEV,IXE,FATOR,
          1 KONT,IFE,EPS,IOUT,KFUN,ITER,GRAG,GRAH,GRAO,MXFUN,W,
          2 GAUX,HAUX,ICH,FN,IMP,IK)
          CALL RUNTIM(IPC)
          TEMPO = FLOAT(IPC)*[illegible]
          WRITE(IDEV,20) NP,TEMPO
20        FORMAT(///,5X,'TEMPO DE CPU PARA O EXEMPLO ',I2,'  ---> ',
          1   G,///)
          WRITE(IDEV,25)
25        FORMAT(////)
         TYPE 15
15      FORMAT(5X,'SE DESEJA OUTRO EXEMPLO BATA 1 ',$)
        ACCEPT 18, J
18      FORMAT(I)
        IF(J.EQ.1) GO TO 77
        STOP
        END
```

```
C-----------------------------------------------------------------------
C
      SUBROUTINE IMPRI(N,M,L,X,F,GFO,Q, AMI,FATOR,KONT,
     * IFR,EPS,KFUN,KTER,GAUX,HAUX,GRAQ,ER,IDEV)
      DIMENSION X(1),AMI(1),GAUX(1),HAUX(1),GRAQ(1),GFO(1)
      COMMON/NUPROB/NP
      WRITE(IDEV,100) NP
      WRITE(IDEV,102) N,M,L,IGR
      WRITE(IDEV,104) F,Q,(X(I),I=1,N)
      WRITE(IDEV,105)
      IF(M.NE.0) WRITE(IDEV,106) (GAUX(I),I=1,M)
      IF(L.NE.0) WRITE(IDEV,108) (HAUX(I),I=1,L)
      WRITE(IDEV,110) (GFO(I),I=1,N)
      WRITE(IDEV,112) (GRAQ(I),I=1,N)
      ML=M+L
      WRITE(IDEV,114) (AMI(I),I=1,ML)
      WRITE(IDEV,116) KONT,KFUN,KTER
      WRITE(IDEV,120) EPS,ER,FATOR
      RETURN
C
100   FORMAT(1H1,10X,'R E S U L T A D O S   F I N A I S   P A R A   O',
     *'   E X E M P L O   ',I2,//)
102   FORMAT(3X,'NUMERO DE VARIAVEIS :',I3,3X,'NUMERO DE RESTRICOES DE
     * DESIGUALDADE :',I3,3X,'NUMERO DE RESTRICOES DE IGUALDADE : ',I3
     *,/,3X,'DIAGNOSTICO : ',I2,/)
104   FORMAT(3X,'VALOR DA FUNCAO OBJETIVO : ',G,5X,'VALOR DA FUNCAO PENA
     *LIDADE : ',G,/,3X,30('-'),' MELHOR PONTO OBTIDO ',30('-'),/,3X,
     * 50(5G,/,3X))
105   FORMAT(3X,81('-'),//)
106   FORMAT(10X,'VALORES DAS RESTRICOES DE DESIGUALDADE ',/,3X,
     * 50(5G,/,3X))
108   FORMAT(10X,'VALORES DAS RESTRICOES DE IGUALDADE',/,3X,50(5G,/,3X)
     *)
110   FORMAT(10X,'GRADIENTE DA FUNCAO OBJETIVO ',/,3X,50(5G,/,3X))
112   FORMAT(10X,'GRADIENTE DA FUNCAO PENALIDADE ',/,3X,50(5G,/,3X))
114   FORMAT(10X,'VETOR DE PARAMETROS DE PENALIZACAO ',/,3X,50(5G,/,3X))
118   FORMAT(3X,'NUMERO DE CHAMADAS (ITERACOES) DA SUBROTINA HLASR : ',
     * I3,/,3X,'NUMERO TOTAL DE AVALIACOES DE FUNCAO E GRADIENTE : ',I4,
     * /,3X,'NUMERO TOTAL DE ITERACOES DENTRO DA SUBROTINA HLASR : ',I4,
     * /)
120   FORMAT(3X,'PRECISAO PEDIDA : ',G,/,3X,'MAIOR INFACTIBILIDADE : ',
     * G,5X,'FATOR DE MULTIPLICACAO DOS PARAMETROS DE PENALIZACAO : ',G)
      END
C
C-----------------------------------------------------------------------




C-----------------------------------------------------------------------
C
      SUBROUTINE FPEN(X,N,M,L,Q,GRAQ,AMI,GNORM,HNORM,F,GFO,
     * GAUX,HAUX,GRAG,GRAH,ICH)
      DIMENSION X(N),GRAQ(N),GRAH(N),AMI(1),GRAG(N),GFO(N),
     * GAUX(M),ICH(M),HAUX(L)
C     O VETOR AMI DEVE TER NO MAXIMO (M+L) COMPONENTES
      DO 2 I=1,N
2     GRAQ(I)=0.
      GNORM = 0.
```

```
      HNORM=0.
C     OBTENCAO DA PRIMEIRA PARCELA DA FUNCAO PENALIDADE,
C     ESTA PARCELA E' A FUNCAO OBJETIVO
      CALL FUN(X,F,GFO,1)
      Q=F
C     CALCULO DAS PARCELAS DE PENALIZACAO DAS RESTRICOES DE DESIGUALD.
      IF(M.EQ.0) GO TO 4
      DO 3 I=1,M
C     TORNANDO TODAS AS CHAVES IGUAIS A ZERO
      ICH(I)=0
      CALL GE(I,X,G,GRAG,1)
C     SE A RESTRICAO DE DESIGUALDADE E' SATISFEITA ENAO SUA CHAVE E' UM
      IF(G.GE.0.) ICH(I)=1
C     OBTIDO O VALOR DE UMA RESTRICAO, GUARDA-SE SEU VALOR EM GAUX
      GAUX(I)=G
C     CALCULO DE UMA PARCELA DE PENALIZACAO E ADICAO A GNORM
      T=(AMAX1(0.,G))**2
      GNORM=GNORM+T
      Q=Q+AMI(I)*T
C     CALCULO DAS PARCELAS DO GRADIENTE DA FUNCAO PENALIDADE,
C     REFERENTE AS RESTRICOES DE DESIGUALDADE
C
C     SE A CHAVE E' IGUAL A 1, A PARCELA E' ZERO, CASO CONTRARIO
C     ESTA SERA' CALCULADA.
      IF(ICH(I).EQ.1) GO TO 3
      CALL GE(I,X,G,GRAG,2)
      DO 5 K=1,N
5     GRAD(K)=GRAD(K)+2.*AMI(I)*GAUX(I)*GRAG(K)
3     CONTINUE
C
C     CALCULO DAS PARCELAS DE PENALIZACAO DAS RESTRICOES DE IGUALDADE
4     IF(L.EQ.0) GO TO 6
      DO 7 I=(M+1),(M+L)
      J=I-M
      CALL HAGA(J,X,H,GRAH,1)
      HAUX(J) = H
C     CALCULO DA PARCELA DE PENALIZACAO E ADICAO AO ACUMULADOR HNORM.
      T= H*H
      Q= Q + AMI(I)*T
      HNORM = HNORM + T
C     CALCULO DAS PARCELAS DO GRADIENTE DA FUNCAO PENALIDADE,
C     REFERENTE AS RESTRICOES DE IGUALDADE
      CALL HAGA(J,X,H,GRAH,2)
      DO 8 K=1,N
8     GRAD(K)=GRAD(K)+2.*AMI(I)*HAUX(J)*GRAH(K)
7     CONTINUE
C     CALCULO DA PARCELA DO GRADIENTE DA FUNCAO OBJETIVO
C     PARA SER ADICIONADA AO GRADIENTE DA FUNCAO PENALIDADE
6     CALL FUN(X,F,GFO,2)
      DO 9 K=1,N
9     GRAD(K)=GRAD(K)+GFO(K)
      RETURN
      END
C
C---------------------------------------------------------------------
```

```
C----------------------------------------------------------------------
C
      SUBROUTINE PENA(X,N,M,L,F,GEQ,Q,AMI,GNORM,HNORM,IDEV,MXF,
     * FATOR,KONT,IER,EPS,IOUT,KFUN,KTER,GRAG,GRAH,GRAQ,MXFUN,
     * W,GAUX,HAUX,ICH,ER,IMP,IW)
      DIMENSION X(1),AMI(1),GRAG(1),GRAH(1),GRAQ(1),GAUX(1),
     * HAUX(1),GEQ(1),W(1),ICH(1)
      ACC=1.E-7
      FANT=-1.E+30
      EPSG=AMIN1(0.1,10.*EPS)
      KONT=0
      KTER=0
      KFUN=0
      IF(IMP.EQ.2) IOUT = 0
C     CHAMADA DA SUBROTINA DE MINIZACOES SEM RESTRICOES
7     CALL MINSR(N,M,L,X,Q,GRAQ,AMI,IFUN,ITER,EPSG,NFLAG,MXFUN,
     *F,GEQ,W,IOUT,IDEV,ACC,GNORM,HNORM,GAUX,HAUX,GRAG,GRAH,ICH)
      CALL FPEN(X,N,M,L,Q,GRAQ,AMI,GNORM,HNORM,F,GEQ,GAUX,HAUX,
     1  GRAG,GRAH,ICH)
      EPSG=EPS
      KFUN=KFUN+IFUN
      KTER=KTER+ITER
      KONT=KONT+1
C     IMPRESSAO DA ITERACAO OBTIDA
      IF(IMP.NE.0) WRITE(IDEV,352) KONT,ITER,F,(X(I),I=1,N)
      ER=0.
      IF(M.EQ.0) GO TO 2
      DO 3 I=1,M
C     ER SERA' O VALOR DA MAIOR INFACTIBILIDADE
      IF(GAUX(I).GT.0.0) ER=AMAX1(GAUX(I),ER)
3     CONTINUE
2     IF(L.EQ.0) GO TO 5
      DO 4 I=1,L
4     ER=AMAX1(ER,ABS(HAUX(I)))
C     OS TESTE PARA CONVERGENCIA A SEGUIR
5     IF(ER.LT.EPS) GO TO 97
C     TESTE PARA SABER SE HOUVE MELHORA NA FUNCAO OBJETIVO.
C     NOTE QUE TAMBEM E' UM TESTE DE CONVERGENCIA. (OPCIONAL)
        IF([illegible]) GO TO 71
      IF(FANT.GE.F) GO TO 97
      FANT=F
C     TESTE PARA SABER SE FOI SUPERADO O NUMERO MAXIMO
C     DE CHAMADAS DA SUBROTINA MINSR
71      IF(KONT.GE.MXF) GO TO 96
C     ATUALIZACAO DO PARAMETROS DE PENALIZACAO
      CALL ATUAMI(AMI,M,L,EPS,FATOR,ICH,HAUX)
      GO TO 7
C
C     SAIDA DOS RESULTADOS
C
C     1) [illegible] DE CONVERGENCIA
97    IER=0
      IF(IMP.EQ.0) RETURN
105   GO TO (101,102),IW
101   WRITE(IDEV,354) IER,KONT,F,(X(I),I=1,N)
      WRITE(IDEV,355) KFUN,KTER,ER
      RETURN
102   CALL IMPRI(N,M,L,X,F,GEQ,Q,AMI,FATOR,KONT,IER,EPS,KFUN,
     * KTER,GAUX,HAUX,GRAQ,ER,IDEV)
      RETURN
```

```
C
C      2) COM A OBTENCAO DE UM PONTO EM QUE A FUNCAO OBJETIVO
C      JA' NAO PODE SER MELHORADA  (ALGUMAS VEZES TRATA-SE DE CONVERGENCIA)
97     IER=1
       IF(IMP.EQ.0) RETURN
       WRITE(IDEV,358)
       GO TO 105
C
C      3) COM KONT SUPERANDO MXF
98     IER=2
       IF(IMP.EQ.0) RETURN
       WRITE(IDEV,360)
       GO TO 105
C
352    FORMAT(/,3X,'ITERACAO --> ',I2,/,3X,'NO. DE ITERACOES DA MINSR :',
      *I4,/,3X,'VALOR DA FUNCAO OBJETIVO : ',G,/,15X,'P O N T O
      *   O B T I D O',/,3X,50(5G,/,3X))
354    FORMAT(/,3X,'--> CONVERGENCIA ',5X,'DIAGNOSTICO : ',I2,5X,
      *'NO. DE CHAMADAS DA MINSR : ',I2,/,3X,'VALOR DA FUNCAO OBJETIVO :'
      *,G,/,3X,12('-'),'MELHOR  PONTO  OBTIDO',12('-'),/,3X,50(5G,/,3X))
356    FORMAT(/,3X,'NUMERO TOTAL DE ITERACOES DE FUNCAO E GRADIENTE : ',
      * I4,/,3X,'NUMERO TOTAL DE ITERACOES REALIZADAS PELA MINSR : ',I4,
      */,3X,'MAIOR INFACTIBILIDADE : ',G)
358    FORMAT(/,3X,'DESTE PONTO JA''NAO SE PODE MELHORAR A FUNCAO OBJETIV
      *O  ',/)
360    FORMAT(/,3X,'SUPEROU O NUMERO MAXIMO DE CHAMADAS DA MINSR',/)
       END
C
C-------------------------------------------------------------------------
C-------------------------------------------------------------
C
         SUBROUTINE ATUAMI(AMI,M,L,EPS,FATOR,ICH,HAUX)
         DIMENSION AMI(1),ICH(1),HAUX(1)
         IF(L.EQ.0) GO TO 9
         DO 7 I=1,L
         IF(ABS(HAUX(I)).GT.EPS) AMI(M+I) = AMI(M+I)*FATOR
7        CONTINUE
9        IF(M.EQ.0) RETURN
         DO 8 I=1,M
         IF(ICH(I).EQ.0) AMI(I) =  AMI(I)*FATOR
8        CONTINUE
         RETURN
         END


C
C-------------------------------------------------------------
```

```
      SUBROUTINE MINSR(N,M,X,F,G,AMI,IFUN,ITER,EPS,NFLAG,MXFUN,
     1 FO,GFO,W,IOUT,IDEV,ACC,GNORM,HNORM,GAUX,HAUX,GRAG,GRAH,ICH)
      DIMENSION X(1),G(1),AMI(1) ,W(1),ICH(1)
      DIMENSION GRAG(1), GRAH(1),GAUX(1),HAUX(1),GFO(1)
      LOGICAL NOPOS
C
C     INICIALIZAR ITER, IFUN E IOUTK
C
      ITER=0
      IFUN=1
      IOUTK=0
C
C
C     COLOCAR PARAMETROS PARA SUBDIVIDIR W.W(I) CONTEM O VETOR DE
C     BUSCA. W(NX+I) CONTEM A MELHOR ESTIMATIVA PARA O MINIMO.
C     W(NG+I) CONTEM O GRADIENTE NO PONTO ATUAL.
C
      DG=0.
C
      NX=N
C
      NG=NX+N
C
C
      NRY=NG+N
      NRD=NRY+N
      NCONS=5*N
      CALL FPCN(X,N,1,1,F,G,AMI,GNORM,HNORM,FO,GFO,GAUX,HAUX,
     1 GRAG,GRAH,ICH)
      NFLAG=0
      NRST=N
C
C
C     CALCULO DA DIRECAO INICIAL DE BUSCA, A SOMA DE X AO QUA-
C     DRADO,A SOMA DE G AO QUADRADO."DG1" E' A ATUAL DERIVA-
C     DA DIRECIONAL,ENQUANTO QUE "XSQ" E "GSQ" SAO NORMAS
C     AO QUADRADO
C
      DG1=0.
      XSQ=0.
      DO 3 I=1,N
      W(I)=-G(I)
      XSQ=XSQ+X(I)*X(I)
    3 DG1=DG1-G(I)*G(I)
      GSQ=-DG1
C
C     TESTE SE O PONTO INICIAL E' O PONTO MINIMO.
C
      AUX1=EPS*EPS*AMAX1(1.0,XSQ)
      IF(GSQ. GT. AUX1)GO TO 4
      IF(IOUT.EQ.0)RETURN
      WRITE(IDEV,30)
      RETURN
C
C
C
    4 FMIN=F
C
C     TESTAR SE O NUMERO MAXIMO DE AVALIACOES FOI USADO
C
```

```
      IF(IFUN.LT.MXFUN)GO TO 45
40    NFLAG=1
      IF(IOUT.EQ.0)RETURN
      WRITE(IDEV,600)
      RETURN
45    NCALLS=IFUN
C
C     TESTAR SE DEVE HAVER IMPRESSAO NESTA ITERACAO.
C
      IF(IOUT .EQ. 0)GO TO 60
      IF(IOUTK .NE. 0)GO TO 50
      WRITE(IDEV,500)ITER,IFUN,FMIN,GSQ
50    IOUTK=IOUTK+1
      IF(IOUTK .EQ. IOUT)IOUTK=0
C
C     COMECO DA BUSCA UNIDIRECIONAL.ALPHA E' O COMPRIMENTO
C     DO PASSO.
60    ALPHA=ALPHA*DG/DG1
C
C     ALPHA=1.0
C
      IF(NRST.EQ.1)  ALPHA=1.0
C
C
C     SE ESTA E' A PRIMEIRA ITERACAO,COLOQUE ALPHA=1.0/SQRT(GSQ),
C     O QUE VAI DETERMINAR QUE O VETOR DE BUSCA INICIAL SEJA UNITARIO.
C
      IF(ITER .EQ. 0)ALPHA=1.0/SQRT(GSQ)
C
C
C     A BUSCA UNIDIRECIONAL AJUSTA UMA CUBICA A "F" E A "DAL"(A
C     FUNCAO E SUA DERIVADA EM ALPHA); E A "FP" E A"DP"(A FUNCAO
C     E A DERIVADA NO PONTO EXPERIMENTAL "AP").INICIALIZAR AP,FP,DP.
      AP=0.
      FP=FMIN
      DP=DG1
C
C
C     GUARDAR A ATUAL DERIVADA,DETERMINAR O PROXIMO VETOR DE BUSCA.
      DG=DG1
C
C
C     ATUALIZE A ITERACAO.
C
      ITER=ITER+1
C
C
C     CALCULAR O ATUAL COMPRIMENTO DO PASSO E GUARDAR O ATUAL "X"
C     E "G"
C
      STEP=0.
      DO 70 I=1,N
      STEP=STEP+W(I)*W(I)
      W(NX+I)=X(I)
70    W(NG+I)=G(I)
      STEP=SQRT(STEP)
C
C
C     COMECE A BUSCA UNIDIRECIONAL DESTA ITERACAO.TESTE PARA
C     VERIFICAR POSSIVEL TERMINACAO DA BUSCA UNIDIRECIONAL.
```

```
C
80        A1=ALPHA*STEP
          IF(A1 .GT. ACC)GO TO 90
          NFLAG=2
          IF(IOUT.EQ.0)RETURN
          WRITE(IDEV,590)
          RETURN
C
C
C         CALCULO DO PONTO EXPERIMENTAL.
C
90        DO 100 I=1,N
100       X(I)=W(NX+I)+ALPHA*S(I)
C
C
C         CALCULO DA FUNCAO NESTE PONTO.
C
          CALL FPEN(X,N,M,L,F,G,[illegible]I,GNORM,HNORM,FO,GFO,GAUX,HAUX,
          1 GRAG,GRAH,ICH)
          IFUN=IFUN+1
          IF(IFUN.LT.MXFUN)GOTO110
          DO 105 I=1,N
          X(I)=W(NX+I)
105       G(I)=W(NG+I)
          F=FMIN
          GOTO42
C         COMPUTAR A DERIVADA DE "F" EM ALPHA.
C
110       DAL=0.0
          DO 120 I=1,N
120       DAL=DAL+G(I)*S(I)
C
C         TESTAR SE FUNCAO NESTE PONTO TEM INCLINACAO NEGATIVA,MAS TEM
C         VALOR MAIOR QUE EM ALPHA=0.NESTE CASO A BUSCA PASSOU POR UM
C         PONTO DE MAXIMO LOCAL,E ESTA' SE DIRIGINDO PARA UM PONTO DE
C         MINIMO LOCAL.
C
          IF((F .GT. FMIN) .AND. (DAL .LT. 0.))GO TO 160
C
C         TESTE SE OS CRITE'RIOS PARA O COMPRIMENTO DO PASSO ESTAO
C         SENDO VERIFICADOS.
C
          AUX1=0.0001*ALPHA*DG
          AUX1=AUX1+FMIN
          AUX2=ABS(DAL/DG)
          AUX3=0.9
          IF(F .GE. AUX1)GO TO 130
          IF(AUX2 .LE. AUX3)GO TO 180
C
C
C         FORAM VERIFICADOS,[illegible] SE [illegible] PONTOS TEM SIDO EXPERIMENTADOS
C
          AUX1=IFUN-[illegible]
          AUX2=ABS(DAL/DG)
          IF((AUX1 .GE. 5.) .AND. (AUX2 .GT. EPS) )
          1GO TO 130
          GO TO 170
C
C
C         PROVER UM NOVO PONTO [illegible] INTERPOLACAO CUBICA PARA ACHAR O
```

```
C      PONTO EXPERIMENTAL.
C
130    U1=DP+DAL-3.0*(DP-F)/(AP-ALPHA)
       U2=U1*U1-DP*DAL
       IF(U2 .LT. 0)U2=0.
       U2=SQRT(U2)
       AT=ALPHA-(ALPHA-AP)*(DAL+U2-U1)/(DAL-DP+2.*U2)
C
C
C      TESTAR SE TEM SIDO LOCALIZADO UM INTERVALO ONDE ESTA' O MI-
C      NIMO.
C
       AUX1=DAL/DP
       IF(AUX1 .GT. 0.)GO TO 140
C
C
C      O INTERVALO TEM SIDO ENCONTRADO,TESTAR SE O PONTO EXPERIMEN-
C      TAL PERTENCE AO INTERVALO DETERMINADO.SE NAO,ESCOLHA O PON-
C      TO MEDIO DO INTERVALO.
C
       AUX1=1.01*AMIN1(ALPHA,AP)
       AUX2=0.99*AMAX1(ALPHA,AP)
       IF((AT .LT. AUX1) .OR. (AT .GT. AUX2))AT=(ALPHA+AP)/2.
       GO TO 150
C
C
C      O INTERVALO AINDA NAO FOI ENCONTRADO,TESTE SE AMBOS OS PON-
C      TOS SAO MAIORES QUE O MINIMO,E O PONTO EXPERIMENTAL E'SUFI-
C      CIENTEMENTE MENOR QUE OS DOIS.
C
C
140    AUX1=0.99*AMIN1(AP,ALPHA)
       IF((DAL .GT. 0.) .AND. (0. .LT. AT) .AND. (AT .LT. AUX1))
       1GO TO 150
C
C
C      TESTE SE AMBOS OS PONTOS SAO MENORES QUE O MINIMO,E O PON-
C      TO EXPERIMENTAL E' SUFICIENTEMENTE GRANDE.
C
       AUX1=1.01*AMAX1(AP,ALPHA)
       IF((DAL .LE. 0.) .AND. (AT .GT. AUX1))GO TO 150
C
C
C      SE O PONTO EXPERIMENTAL E' PEQUENO DEMAIS,DUPLIQUE O MAIOR
C      PONTO ANTERIOR.
C
C
       IF(DAL .LE. 0.)AT=2.*AMAX1(AP,ALPHA)
C
C
C      SE O PONTO EXPERIMENTAL E' GRANDE DEMAIS,DIVIDA POR 2
C      O MENOR PONTO ANTERIOR.
C
C
       IF(DAL .GT. 0.)AT=AMIN1(AP,ALPHA)/2.0
C
C
C      CALCULAR AP=ALPHA,ALPHA=AT E CONTINUE A BUSCA.
C
150    AP=ALPHA
```

```
      FP=F
      DP=DAL
      ALPHA=AT
      GO TO 80
C
C
C     O PONTO MAXIMO RELATIVO TEM SIDO PASSADO.DIVIDA ALPHA POR 2
C     E REINICIE A BUSCA.
C
160   ALPHA=ALPHA/2.
      AP=0.
      FP=FMIN
      DP=DG
      GO TO 80
C
C
C     A BUSCA UNIDIRECIONAL TEM CONVERGIDO.TESTAR A CONVERGENCIA
C     DO ALGORITMO.
C
170   GSQ=0.0
      XSQ=0.0
      DO 180 I=1,N
      GSQ=GSQ+G(I)*G(I)
180   XSQ=XSQ+X(I)*X(I)
      AUX1=EPS*EPS*AMAX1(1.0,XSQ)
      IF(GSQ .GT. AUX1)GO TO 185
      IF(IOUT.EQ.0)RETURN
      WRITE(IDEV,500) ITER,IFUN,F,GSQ
      RETURN
C
C
C     A BUSCA CONTINUA.COLOCAR W(I)=ALPHA*W(I).
C
185   DO 190 I=1,N
190   W(I)=ALPHA*W(I)
C
C
C     COMPUTAR O NOVO VETOR DE BUSCA.TESTAR QUE METODO ESTA' SEN-
C     DO USADO.
C
C
C
C     GRADIENTE CONJUGADO E' ATUALIZADO NESTA SECAO.TESTAR SE E'
C     INDICADO UM REINICIO DE "POWELL".
C
      RTST=0.
      DO 200 I=1,N
200   RTST=RTST+G(I)*W(NX+I)
      IF(ABS(RTST/GSQ) .GT. 0.2)NRST=N
C
C
C     SE UM REINICIO FOR INDICADO,SALVAR O ATUAL"D" E"Y",COMO VETO-
C     RES DE UM REINICIO DE "POWELL";E"D'Y" E "Y'Y" EM W(NCONS+1) E
C     W(NCONS+2) RESPECTIVAMENTE.
C
      IF(NRST .NE. N)GO TO 220
      W(NCONS+1)=0.
      W(NCONS+2)=0.
      DO 210 I=1,N
      W(NRY+I)=G(I)-W(NG+I)
```

```
      W(NRD+I)=W(I)
      W(NCONS+1)=W(NCONS+1)+W(NRY+I)*W(NRY+I)
210   W(NCONS+2)=W(NCONS+2)+W(I)*W(NRY+I)
C
C
C     CALCULO DA HESSIANA VEZES O GRADIENTE ATUAL.
C
220   U1=0.0
      U2=0.0
      DO 230 I=1,N
      U1=U1-W(NRD+I)*G(I)/W(NCONS+1)
230   U2=U2+W(NRD+I)*G(I)*2./W(NCONS+2)-W(NRY+I)*G(I)/W(NCONS+1)
      U3=W(NCONS+2)/W(NCONS+1)
      DO 240 I=1,N
240   W(NX+I)=-U3*G(I)-U1*W(NRY+I)-U2*W(NRD+I)
C
C
C     SE ESTA E' UMA ITERACAO REINICIADA,W(NX+I) CONTEM O NOVO
C     VETOR DE BUSCA.
C
      IF(NREST .EQ. 0)GO TO 300
C
C     SE NAO E' UMA ITERACAO REINICIADA, CALCULO DA HESSIANA REINI-
C     CIADA VEZES O ATUAL "Y".
C
250   U1=0.
      U2=0.
      U3=0.
      U4=0.
      DO 260 I=1,N
      U1=U1-(G(I)-W(NG+I))*W(NRD+I)/W(NCONS+1)
      U2=U2-(G(I)-W(NG+I))*W(NRY+I)/W(NCONS+1)
     1+2.0*W(NRD+I)*(G(I)-W(NG+I))/W(NCONS+2)
260   U3=U3+W(I)*(G(I)-W(NG+I))
      STEP=0.
      DO 270 I=1,N
      STEP=(W(NCONS+2)/W(NCONS+1))*(G(I)-W(NG+I))
     1+U1*W(NRY+I)+U2*W(NRD+I)
      U4=U4+STEP*(G(I)-W(NG+I))
270   W(NG+I)=STEP
C
C     CALCULO DA HESSIANA 2 ATUALIZADA,VEZES O GRADIENTE ATUAL PARA
C     OBTER O VETOR DE BUSCA.
C
      U1=0.
      U2=0.
      DO 280 I=1,N
      U1=U1-W(I)*G(I)/U3
280   U2=U2+(1.0+U4/U3)*W(I)*G(I)/U3-W(NG+I)*G(I)/U3
      DO 290 I=1,N
290   W(NX+I)=W(NX+I)-U1*W(NG+I)-U2*W(I)
C
C     CALCULO DA DERIVADA AO LONGO DO NOVO VETOR DE BUSCA.
C
300   DG1=0.
      DO 310 I=1,N
      W(I)=W(NX+I)
310   DG1=DG1+W(I)*G(I)
C
C     SE A NOVA DIRECAO NAO FOR DE DESCIDA ;PARE.
```

```
C
      IF(DG1 .GE. 0.)GO TO 320
C
C     ATUALIZE "NRST" PARA ASSEGURAR UM REINICIO A CADA N ITERA-
C     COES.
C
      IF(NRST .EQ. N)NRST=0
      NRST=NRST+1
      GO TO 40
C
C     ERROS DE ARREDONDAMENTO, PRODUZIRAM UMA MA' DIRECAO.
C
320   NFLAG=3
      IF(IOUT.EQ.0)RETURN
500   FORMAT(' O PONTO INICIAL E'' SOLUCAO ')
540   FORMAT(  '      FRACASSO  NFLAG=2')
600   FORMAT('    FRACASSO NFLAG=1')
560   FORMAT(    '       ITERACAO ' ,I5,17H    AVALIACOES   :        ,I5,
     1/'    F =   ',G, '   G-QUADRADO =   ',G)
610   FORMAT('    FRACASSO NFLAG=3')
      WRITE(IOUT,610)
      RETURN
      END
```

```
C     ARQUIVO DE FUNCOES TESTE
C     CONSTA DE TRES SUBROTINAS - UMA PARA FUNCOES OBJETIVOS
C                               - OUTRA PARA RESTRICOES DE DESIGUALDADE
C                                      - OUTRA PARA RESTRICOES DE  IGUALDADE
      SUBROUTINE FUN(X,F,GF,L)
C     SE L=1 A FUNCAO E' CALCULADA
C     SE L=2 O GRADIENTE DA FUNCAO E' CALCULADO
      DIMENSION X(2),GF(2)
      COMMON/NOPROB/NP
      GO TO (1,2,3,4,5,6,7,31,32,33,34,35,36,37,38,39,300)NP
1     IF(L.EQ.2)GO TO 100
      F=3.*X(1)+X(2)
      RETURN
100   GF(1)=3.
      GF(2)=1.
      RETURN
C
2     IF(L.EQ.2) GO TO 101
      F=X(2)-X(1)**2
      RETURN
101   GF(1)=-2.*X(1)
      GF(2)=1.
      RETURN
C
3     IF(L.EQ.2) GO TO 102
      F=X(1)+X(2)
      RETURN
102   GF(1)=1.
      GF(2)=1.
      RETURN
C
4     IF(L.EQ.2) GO TO 103
      F=9.*X(1)+X(2)
      RETURN
103   GF(1)= 9.
      GF(2)= 1.
      RETURN
C
5     IF(L.EQ.2) GO TO 104
      F=X(1)*X(1)+X(2)*X(2)
      RETURN
104   GF(1)=2.*X(1)
      GF(2)=2.*X(2)
      RETURN
C
6     IF(L.EQ.2) GO TO 105
      F=(X(1)-2.5)**2 + (X(2)-2.5)**2
      RETURN
105   GF(1)=2.*(X(1)-2.5)
      GF(2)=2.*(X(2)-2.5)
      RETURN
C
7     IF(L.EQ.2) GO TO 106
      F=100.*(X(2)-X(1)*X(1))**2 + (1.-X(1))**2
      RETURN
106   GF(1)=-400.*X(1)*(X(2)-X(1)*X(1))-2.*(1.-X(1))
      GF(2)= 200.*(X(2)-X(1)*X(1))
      RETURN
C
31    IF(L.EQ.2) GO TO 107
```

```
      F=1./3.*(X(1)+1.)**3 +X(2)
      RETURN
107   GF(1)=(X(1)+1.)**2
      GF(2)=1.
      RETURN
C
32    IF(L.EQ.2) GO TO 108
      F= X(1)**2+X(2)**2+2.*X(3)**2+X(4)**2-5.*X(1)
     1 -5.*X(2)-21.*X(3)+7.*X(4)
      RETURN
108   GF(1)= 2.*X(1)-5.
      GF(2)= 2.*X(2)-5.
      GF(3)=4.*X(3)-21.
      GF(4)=2.*X(4)+7.
      RETURN
C
33    IF(L.EQ.2) GO TO 109
      F=9.-8.*X(1)-6.*X(2)-4.*X(3)+2.*X(1)**2+2.*X(2)**2
     1 +X(3)**2+2.*X(1)*X(2)+2.*X(1)*X(3)
      RETURN
109   GF(1)=4.*X(1)+2.*X(2)+2.*X(3)-8.
      GF(2)=2.*X(1)+4.*X(2)-6.
      GF(3)=2.*X(1)+2.*X(3)-4.
      RETURN
C
34    PROD=X(1)*X(2)*X(3)*X(4)*X(5)
      IF(L.EQ.2) GO TO 110
      F= EXP(PROD)
      RETURN
110   T= EXP(PROD)
      GF(1)=PROD/X(1)*T
      GF(2)=PROD/X(2)*T
      GF(3)=PROD/X(3)*T
      GF(4)=PROD/X(4)*T
      GF(5)=PROD/X(5)*T
      RETURN
C
35    IF(L.EQ.2) GO TO 111
      F=(X(1)-10.)**2+5.*(X(2)-12.)**2+X(3)**4+3.*(X(4)-11.)**2
     1 +10.*X(5)**6 +7.*X(6)**2+X(7)**4-4.*X(6)*X(7)-10.*X(6)-8.*X(7)
      RETURN
111   GF(1)=2.*(X(1)-10.)
      GF(2)=10.*(X(2)-12.)
      GF(3)=4.*X(3)**3
      GF(4)=6.*(X(4)-11.)
      GF(5)=60.*X(5)**5
      GF(6)=14.*X(6)-4.*X(7)-10.
      GF(7)=4.*X(7)**3-4.*X(6)-8.
      RETURN
C
36    IF(L.EQ.2) GO TO 112
      F=X(1)**2+X(2)**2+X(1)*X(2)-14.*X(1)-16.*X(2)+(X(3)-10.)**2
     1 +4.*(X(4)-5.)**2+(X(5)-3.)**2+2.*(X(6)-1.)**2+5.*X(7)**2
     2 +7.*(X(8)-11.)**2+2.*(X(9)-10.)**2+(X(10)-7.)**2+45.
      RETURN
112   GF(1)=2.*X(1)+X(2)-14.
      GF(2)=X(1)+2.*X(2)-16.
      GF(3)=2.*(X(3)-10.)
      GF(4)=8.*(X(4)-5.)
      GF(5)=2.*(X(5)-3.)
```

```
      GF(6)=4.*(X(6)-1.)
      GF(7)=10.*X(7)
      GF(8)=14.*(X(8)-11.)
      GF(9)=4.*(X(9)-10.)
      GF(10)=2.*(X(10)-7.)
      RETURN
C
37    IF(L.EQ.2) GO TO 113
      F=X(1)**2+X(2)**2+X(1)*X(2)-14.*X(1)-16.*X(2)+(X(3)-10.)**2
     1 +4.*(X(4)-5.)**2+(X(5)-3.)**2+2.*(X(6)-1.)**2+5.*X(7)**2+
     2 7.*(X(8)-11.)**2+2.*(X(9)-10.)**2+(X(10)-7.)**2+(X(11)-9.)**2+
     3 10.*(X(12)-1.)**2+5.*(X(13)-7.)**2+4.*(X(14)-14.)**2+27.*(X(15)
     4 -1.)**2+X(16)**4+(X(17)-2.)**2+13.*(X(18)-2.)**2+(X(19)-3.)**2+
     5 X(20)**2 +95.
      RETURN
113   GF(1)=2.*X(1)+X(2)-14.
      GF(2)= X(1)+2.*X(2)-16.
      GF(3)=2.*(X(3)-10.)
      GF(4)=8.*(X(4)-5.)
      GF(5)=2.*(X(5)-3.)
      GF(6)=4.*(X(6)-1.)
      GF(7)=10.*X(7)
      GF(8)=14.*(X(8)-11.)
      GF(9)=4.*(X(9)-10.)
      GF(10)=2.*(X(10)-7.)
      GF(11)=2.*(X(11)-9.)
      GF(12)=20.*(X(12)-1.)
      GF(13)=10.*(X(13)-7.)
      GF(14)= 8.*(X(14)-14.)
      GF(15)=54.*(X(15)-1.)
      GF(16)=4.*X(16)**3
      GF(17)=2.*(X(17)-2.)
      GF(18)=26.*(X(18)-2.)
      GF(19)=2.*(X(19)-3.)
      GF(20)=2.*X(20)
      RETURN
C
38    IF(L.EQ.2) GO TO 114
      F=-12.*X(1)-7.*X(2)+X(2)**2
      RETURN
114   GF(1)=-12.
      GF(2)=2.*X(2)-7.
      RETURN
C
39    IF(L.EQ.2) GO TO 115
      F= X(2)-X(1)
      RETURN
115   GF(1)=-1.
      GF(2)=1.
      RETURN
C
310   IF(L.EQ.2) GO TO 116
      F=1./16.*X(1)**2+1./9.*X(2)**2
      RETURN
116   GF(1)=1./8.*X(1)
      GF(2)=2./9.*X(2)
      RETURN
      END
```

```
C         SUBROTINA PARA RESTRICOES DE DESIGUALDADES
      SUBROUTINE GE(I,X,G,GRAG,L)
      DIMENSION X(2),GRAG(2)
      COMMON/NOPROB/NP
      IF(L.EQ.2) GO TO 499
       GO TO(1,99,20,20,20,20,99,31,32,33,34,35,36,37,38,39,300)NP
1     GO TO(3,3,5)I
3     G=-X(I)
      RETURN
5     G=(1./X(1))-X(2)
99    RETURN
20    GO TO(21,22,23)I
21    G=1.-X(1)*X(2)
      RETURN
22    G=4.-X(1)-2.*X(2)
      RETURN
23    G=2.+X(1)-4.*X(2)
      RETURN
31    GO TO(41,42)I
41    G=-X(1)+1.
      RETURN
42    G=-X(2)
      RETURN
32    GO TO (51,52,53) I
51    G=X(1)**2+X(2)**2+X(3)**2+X(4)**2+X(1)-X(2)+X(3)-X(4)-8.
      RETURN
52    G=X(1)**2+2.*X(2)**2+X(3)**2+2.*X(4)**2-X(1)-X(4)-10.
      RETURN
53    G=2.*X(1)**2+X(2)**2+X(3)**2+2.*X(1)-X(2)-X(4)-5.
34    RETURN
33      IF(I.NE.4)    GO TO 3
        G= X(1) + X(2) + 2.*X(3) -3.
        RETURN
35      GO TO (61,62,63,64) I
61      G=2.*X(1)**2+3.*X(2)**4+X(3)+4.*X(4)**2+5.*X(5)-127.
        RETURN
62      G=7.*X(1)+3.*X(2) +10.*X(3)**2+X(4) -X(5) -282.
        RETURN
63      G=23.*X(1) + X(2)**2 + 6.*X(6)**2 -8.*X(7) -196.
        RETURN
64      G= 4.*X(1)**2 + X(2)**2-3.*X(1)*X(2)+2.*X(3)**2+5.*X(6)-11.*X(7)
        RETURN
C
C
36    GO TO (71,72,73,74,75,76,77,78)I
71    G=3.*(X(1)-2.)**2+4.*(X(2)-3.)**2+2.*X(3)**2  -7.*X(4)-120.
      RETURN
72    G=5.*X(1)**2+8.*X(2)+(X(3)-6.)**2-2.*X(4)-40.
      RETURN
73    G=0.5*(X(1)-8.)**2+2.*(X(2)-4.)**2+3.*(X(5))**2-X(6)-30.
      RETURN
74    G=X(1)**2+2.*(X(2)-2.)**2-2.*X(1)*X(2)+14.*X(5)-6.*X(6)
      RETURN
75    G=4.*X(1)+5.*X(2)-3.*X(7)+9.*X(8)-105.
      RETURN
76    G=10.*X(1)-8.*X(2)-17.*X(7)+2.*X(8)
      RETURN
```

```
77      G=-3.*X(1)+6.*X(2)+12.*(X(9)-8.)**2-7.*X(10)
        RETURN
78      G=-8.*X(1)+2.*X(2)+5.*X(9)-2.*X(10)-12.
        RETURN
C
37      GO TO (81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97)I
81      G=3.*(X(1)-2.)**2+4.*(X(2)-3)**2+2.*X(3)**2-7.*X(4)-120.
        RETURN
82      G=5.*X(1)**2+8.*X(2)+(X(3)-6.)**2-2.*X(4)-40.
        RETURN
83      G=0.5*(X(1)-8.)**2+2.*(X(2)-4.)**2+3.*X(5)**2-X(6)-30.
        RETURN
84      G=X(1)**2+2.*(X(2)-2.)**2-2.*X(1)*X(2)+14.*X(5)-6.*X(6)
        RETURN
85      G=4.*X(1)+5.*X(2)-3.*X(7)+9.*X(8)-105.
        RETURN
86      G=10.*X(1)-8.*X(2)-17.*X(7)+2.*X(8)
        RETURN
87      G=3.*X(1)+6.*X(2)+12.*(X(9)-8.)**2-7.*X(10)
        RETURN
88      G=-8.*X(1)+2.*X(2)+5.*X(9)-2.*X(10)-12.
        RETURN
89      G=X(1)+X(2)+4.*X(11)-21.*X(12)
        RETURN
90      G=X(1)**2+15.*X(11)-8.*X(12)-28.
        RETURN
91      G=4.*X(1)+9.*X(2)+5.*X(13)**2-9.*X(14)-87.
        RETURN
92      G=3.*X(1)+4.*X(2)+3.*(X(13)-6.)**2-14.*X(14)-10.
        RETURN
93      G=14.*X(1)**2+35.*X(15)-79.*X(16)-92.
        RETURN
94      G=15.*X(2)**2+11.*X(15)-61.*X(16)-54.
        RETURN
95      G=5.*X(1)**2+2.*X(2)+9.*X(17)**4-X(18)-68.
        RETURN
96      G=X(1)**2-X(2)+19.*X(19)-20.*X(20)+19.
        RETURN
97      G=7.*X(1)**2+5.*X(2)**2+X(19)**2-30.*X(20)
        RETURN
C
360     GO TO (361,362,363,364,365,366,367,368)I
361     GRAG(1)=6.*(X(1)-2.)
        GRAG(2)=8.*(X(2)-3.)
        GRAG(3)=4.*X(3)
        GRAG(4)=-7.
187     DO 188 K=5,10
188     GRAG(K)=0.
        RETURN
362     GRAG(1)=10.*X(1)
        GRAG(2)=8.
        GRAG(3)=2.*(X(3)-6.)
        GRAG(4)=-2.
        GO TO 187
363     GRAG(1)=X(1)-8.
        GRAG(2)=4.*(X(2)-4.)
        GRAG(3)=0.
        GRAG(4)=0.
        GRAG(5)=6.*X(5)
        GRAG(6)=-1.
```

```
177   DO 178 K=7,10
178   GRAG(K)=0.
      RETURN
364   GRAG(1)=2.*X(1)-2.*X(2)
      GRAG(2)=4.*(X(2)-2.)-2.*X(1)
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=14.
      GRAG(6)=-6.
      GO TO 177
365   GRAG(1)=4.
      GRAG(2)=5.
      GRAG(7)=-3.
      GRAG(8)=9.
      GRAG(9)=0.
      GRAG(10)=0.
      DO 168 K=3,6
168   GRAG(K)=0.
      RETURN
366   GRAG(1)=10.
      GRAG(2)=-8.
      GRAG(7)=-17.
      GRAG(8)=2.
      DO 158 K=3,6
158   GRAG(K)=0.
      GRAG(9)=0.
      GRAG(10)=0.
      RETURN
367   GRAG(1)=-3.
      GRAG(2)=6.
      GRAG(9)=24.*(X(9)-8.)
      GRAG(10)=-7.
153   DO 157 K=3,8
157   GRAG(K)=0.
      RETURN
368   GRAG(1)=-8.
      GRAG(2)=2.
      GRAG(9)=5.
      GRAG(10)=-2.
      GO TO 153
C
370   GO TO(471,472,473,474,475,476,477,478,479,480,481,482,483,484,485,
     * 486,487)I
471   GRAG(1)=6.*(X(1)-2.)
      GRAG(2)=8.*(X(2)-3.)
      GRAG(3)=4.*X(3)
      GRAG(4)=-7.
800   DO 801 K=5,20
801   GRAG(K)=0.
      RETURN
472   GRAG(1)=10.*X(1)
      GRAG(2)=8.
      GRAG(3)=2.*(X(3)-6.)
      GRAG(4)=-2.
      GO TO 800
473   GRAG(1)=X(1)-8.
      GRAG(2)=4.*(X(2)-4.)
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=6.*X(5)
```

```
      GRAG(6)=-1.
      JJ=7
899   DO 802 K=JJ,20
802   GRAG(K)=0.
      RETURN
474   GRAG(1)=2.*X(1)-2.*X(2)
      GRAG(2)=4.*(X(2)-2.)-2.*X(1)
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=14.
      GRAG(6)=-6.
      JJ=7
      GO TO 899
475   GRAG(1)=4.
      GRAG(2)=5.
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=0.
      GRAG(6)=0.
      GRAG(7)=-3.
      GRAG(8)=9.
      JJ=10
      GO TO 899
476   GRAG(1)=10.
      GRAG(2)=-8.
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=0.
      GRAG(6)=0.
      GRAG(7)=-17.
      GRAG(8)=2.
      JJ=9
      GO TO 899
477   GRAG(1)=3.
      GRAG(2)=6.
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=0.
      GRAG(6)=0.
      GRAG(7)=0.
      GRAG(8)=0.
      GRAG(9) = 24.*(X(9)-8.)
      GRAG(10)=-7.
      JJ=11
      GO TO 899
478   GRAG(1)=-8.
      GRAG(2)=2.
      DO 870 K=3,20
870   GRAG(K)=0.
      GRAG(9)=5.
      GRAG(10)=-2.
      RETURN
479   GRAG(1)=1.
      GRAG(2)=1
      DO 871 K=3,20
871   GRAG(K)=0.
      GRAG(11)=4.
      GRAG(12)=-21.
      RETURN
4?0   GRAG(1)=2.*X(1)
```

```
      DO 872 K=2,20
872   GRAG(K)=0.
      GRAG(11)=15.
      GRAG(12)=-8.
      RETURN
481   GRAG(1)=4.
      GRAG(2)=9.
      DO 180 K=3,20
180   GRAG(K)=0.
      GRAG(13)=10.*X(13)
      GRAG(14)=-9.
      RETURN
482   GRAG(1)=3.
      GRAG(2)=4.
      DO 805 K=3,20
805   GRAG(K)=0.
      GRAG(13)=6.*(X(13)-6.)
      GRAG(14)=-14.
      RETURN
483   GRAG(1)=28.*X(1)
      DO 806 K=2,20
806   GRAG(K)=0.
      GRAG(15)=35.
      GRAG(16)=-79.
      RETURN
484   GRAG(1)=0.
      GRAG(2)=30.*X(2)
      DO 807 K=3,20
807   GRAG(K)=0.
      GRAG(15)=11.
      GRAG(16)=-61.
      RETURN
485   GRAG(1)=10.*X(1)
      GRAG(2)=2.
      DO 808 K=3,20
808   GRAG(K)=0.
      GRAG(17)=36.*X(17)**3
      GRAG(18)=-1.
      RETURN
486   GRAG(1)=2.*X(1)
      GRAG(2)=-1.
      DO 809 K=3,20
809   GRAG(K)=0.
      GRAG(19)=19.
      GRAG(20)=-20.
      RETURN
487   GRAG(1)=14.*X(1)
      GRAG(2)=10.*X(2)
      DO 901 K=3,20
901   GRAG(K)=0.
      GRAG(19)=2.*X(19)
      GRAG(20)=-30.
      RETURN
C
C
499    GO TO(120,99,130,130,130,130,99,310,320,330,340,350
        1    ,360,370,380,390,3000) NP
120   GO TO(201,202,203)I
201   GRAG(1)=-1.
      GRAG(2)=0.
```

```
      RETURN
202   GRAG(1)=0.
      GRAG(2)=-1.
      RETURN
203   GRAG(1)=-1./X(1)**2
      GRAG(2)=-1.
      RETURN
130   GO TO(131,132,133)I
131   GRAG(1)=-X(2)
      GRAG(2)=-X(1)
      RETURN
132   GRAG(1)=-1.
      GRAG(2)=-2.
      RETURN
133   GRAG(1)=1.
      GRAG(2)=-4.
      RETURN
310   GO TO(311,312)I
311   GRAG(1)=-1.
      GRAG(2)=0.
      RETURN
312   GRAG(1)=0.
      GRAG(2)=-1.
      RETURN
320   GO TO(321,322,323)I
321   GRAG(1)=2.*X(1)+1.
      GRAG(2)=2.*X(2)-1.
      GRAG(3)=2.*X(3)+1.
      GRAG(4)=2.*X(4)-1.
      RETURN
322   GRAG(1)=2.*X(1)-1.
      GRAG(2)=4.*X(2)
      GRAG(3)=2.*X(3)
      GRAG(4)=4.*X(4)-1.
      RETURN
323   GRAG(1)=4.*X(1)+2.
      GRAG(2)=2.*X(2)-1.
      GRAG(3)=2.*X(3)
      GRAG(4)=-1.
340   RETURN
330     IF(I.EQ.4) GO TO 331
        GRAG(I) = -1.
        RETURN
331               GRAG(1) = 1.
        GRAG(2) = 1.
        GRAG(3) = 2.
      RETURN
C
350   GO TO(351,352,353,354)I
351   GRAG(1)=4.*X(1)
      GRAG(2)=12.*X(2)**3
      GRAG(3)=1.
      GRAG(4)=8.*X(4)
      GRAG(5)=5.
      GRAG(6)=0.
      GRAG(7)=0.
      RETURN
352   GRAG(1)=7.
      GRAG(2)=3.
      GRAG(3)=20.*X(3)
```

```
      GRAG(4)= 1.
      GRAG(5)=-1.
      GRAG(6)=0.
      GRAG(7)=0.
      RETURN
353   GRAG(1)=23.
      GRAG(2)=2.*X(2)
      GRAG(3)=0.
      GRAG(4)=0.
      GRAG(5)=0.
      GRAG(6)=12.*X(6)
      GRAG(7)=-8.
      RETURN
354   GRAG(1)=8.*X(1)-3.*X(2)
      GRAG(2)=2.*X(2)-3.*X(1)
      GRAG(3)=4.*X(3)
      GRAG(4)=0.
      GRAG(5)=0.
      GRAG(6)=5.
      GRAG(7)=-11.

      RETURN

38      G=-X(I)
      RETURN
39      GO TO (931,932,933,934,935) I
931   G= X(2)-X(1)-5.
      RETURN
932   G=-3.*X(1)-X(2)+6.
      RETURN
933   G= 0.2*X(1)-X(2)+1.
      RETURN
934   G=10.*X(1)-X(2)-90.
      RETURN
935   G=0.5*X(1)+X(2)-8.
      RETURN
300      GO TO (936,937,938) I
936   G=X(2)-X(1)
      RETURN
937   G=1.-X(2)
      RETURN
938   G=(1./3.)*X(1)+X(2)-7.
      RETURN
380   DO 777 J=1,4
777   GRAG(J)=0.
      GRAG(I)=-1.
      RETURN
390   GO TO (971,972,973,974,975) I
971   GRAG(1)=-1.
      GRAG(2) = 1.
      RETURN
972   GRAG(1)=-3.
      GRAG(2)=-1.
      RETURN
973   GRAG(1)=0.2
      GRAG(2)=-1.
      RETURN
974   GRAG(1)=10.
      GRAG(2)=-1.
      RETURN
```

```
975    GRAG(1)=0.5
       GRAG(2)=1.
       RETURN
3000    GO TO (921,922,923)         I
921    GRAG(1)=-1.
       GRAG(2)= 1.
       RETURN
922    GRAG(1)=0.
       GRAG(2)= -1.
       RETURN
923    GRAG(1)=1./3.
       GRAG(2)=1.
       RETURN
       END
       SUBROUTINE HAGA(I,X,H,GRAH,L)
       DIMENSION X(2),GRAH(2)
       COMMON/NOPROB/NP
       GO TO (1,2,1,1,1,1,1,1,1,1,34,1,1,1,38,1,1) NP
2      IF(L.EQ.2) GO TO 9
       H=X(2)-2.*X(1)**2
1      RETURN
9      GRAH(1)=-4.*X(1)
       GRAH(2)=1.
       RETURN
34     GO TO (11,12,13) I
11     GO TO (14,15)L
14     H=X(1)**2+X(2)**2+X(3)**2+X(4)**2+X(5)**2 -*0.
       RETURN
15     DO 17 J=1,5
17       GRAH(J)=2.*X(J)
       RETURN
12     GO TO (20,21) L
20     H=X(2)*X(3)-5.*X(4)*X(5)
       RETURN
21     GRAH(1)=0.
       GRAH(2)=X(3)
       GRAH(3)=X(2)
       GRAH(4)=-5.*X(5)
       GRAH(5)=-5.*X(4)
       RETURN
13     GO TO (24,25)L
24     H=X(1)**3+X(2)**3 +1.
       RETURN
25     GRAH(1)=3.*X(1)**2
       GRAH(2)=3.*X(2)**2
       GRAH(3)=0.
       GRAH(4)=0.
       GRAH(5)=0.
       RETURN
38     GO TO (40,29)L
40     H=-2.*X(1)**4 -X(2)+2.
       RETURN
29     GRAH(1)=-8.*X(1)**3
       GRAH(2)=-1.
       RETURN
       END
C
```

```
C         ATUALIZACAO D[illegible] 28 DE [illegible] DE 81
C
C         PROGRAMA PRINCIPAL PARA MINIMIZACAO POR FUNCAO
C         LAGRANGEANO AU[illegible]
C
          DIMENSION X(5[illegible]),[illegible](500),XSIG(500),GX(500),HX(500),
          1 TETA(500),X[illegible](5[illegible]),[illegible](2502),GRAG(500),
          2 GRAH(500),AK(5[illegible]),GPSI(50[illegible]),GPU(500),GAUX(500),HAUX(500)

          COMMON/NOPROB/N[illegible]
          COMMON/SFACIT/ I[illegible]
          COMMON/[illegible]/[illegible]
          DATA IOUT,IDEV,[illegible],[illegible],ACC/0,22,400,9,1.E-7/
          DATA KD,I[illegible]/[illegible],[illegible]/
          DATA TOL,EPSG,FATOR,EPS/1.E-[illegible],1.E-3,10.,1.E-4/
77        DO 55 I=1,[illegible]
55        XLAM(I) = 0.
          DO 44 I =1,L
44        TETA(I) = 0.
          TYPE 1
1         FORMAT(5X,'[illegible] EXEMPLO : ',$)
          ACCEPT 2,[illegible]
2         FORMAT(I)
          TYPE 88
88        FORMAT(2X,' CODIGO PARA IMPRESSAO ? ',$)
          ACCEPT 2,I[illegible]
          TYPE 21
21        FORMAT(5X,'SE [illegible] NOVOS VALORES PARA TOL,EPSG,FATOR,EPS,
          1 BATA 1 ' )
          ACCEPT 2,ID
          IF(ID.NE.1) GO TO 23
          TYPE 24
24        FORMAT(5X,'DE-[illegible] TOL ,EPSG,FATOR E EPS ',/)
          ACCEPT 26,TOL,EPSG,FATOR,EPS
26        FORMAT(4G)
23        TYPE 3
3         FORMAT(5X,'VALORES DE [illegible] : ',$)
          ACCEPT 4, [illegible]
4         FORMAT(3I)
          IF([illegible].EQ.0) GO TO [illegible]
          TYPE 5
5         FORMAT(5X,'[illegible] : ',$)
          ACCEPT 6,[illegible]
6         FORMAT(G)
          DO 17 J=1,L
17        X[illegible]I(J) = [illegible]
1[illegible]        IF([illegible].EQ.0) GO TO [illegible]
          TYPE 9
9         FORMAT(5X,'VALOR INICIAL DE XSIG : ' ,$)
          ACCEPT 6,[illegible]
          DO 1[illegible] J=1,[illegible]
1[illegible]        X[illegible](J)= [illegible]
1[illegible]        TYPE 10
1[illegible]        FORMAT(5X,'[illegible] : ',/)
          DO 15 J=1,[illegible]
15        ACCEPT 6 ,X(J)
          WRITE(IDEV,[illegible]) ([illegible](I) ,I=1,[illegible] )
9[illegible]        FORMAT(////,[illegible],'[illegible] : ',/,3X,100(5G,/),///)
          I[illegible] = 0
          CALL [illegible]([illegible])
```

```
      CALL PNL01(X,G,H,E,XLAM,XSIG,TETA,XMI, FOBJ,FPSI,GPSI,GX,
     1  HX,GFO,W,GRAG,GRGH,AK,KFUN,EPS,MXFUN,MAXF,IOUT,ACC,TOL,IDEV,
     2  KONT,KTER,IER,GAUX,HAUX,FATOR,EPSG,IMP,KD)
      CALL RUNTIME(ITE)
      TEMPO = FLOAT(ITE)*.001
      WRITE(IDEV,188) NP,TEMPO
188   FORMAT(//,'  TEMPO TOTAL DE CPU PARA O EXEMPLO ',I3,'  -->   '
     1  ,F,' SEGUNDOS ' ,/////)
      TYPE 14
14    FORMAT(5X,' SE DESEJA NOVO EXEMPLO , BATA 1 : ',$)
      ACCEPT 2, NJ
      IF(NJ.EQ.1) GO TO 77
      STOP
      END
```

```
C-----------------------------------------------------------------------
C
C      FINALIDADE : ESTA SUB-ROTINA RESOLVE O PROBLEMA GERAL DE
C                        [illegible], MINIMIZAR F(X) SUJEITO A G(X)
C                  [illegible] IGUAL A ZERO E H(X) IGUAL A ZERO.
C
C                  MIN  F(X1,X2,....,XN) ,      X=(X1,X2,....,XN)
C
C                  S/A  G1(X1,X2,....,XN) <=0
C                       G2(X1,X2,....,XN) <=0
C                       .
C                       .
C                       .
C                       GM(X1,X2,....,XN) <= 0
C                       H1(X1,X2,....,XN) =  0
C                       H2(X1,X2,....,XN) =  0
C                       .
C                       .
C                       .
C                       HL(X1,X2,....,XN) =  0
C
C-----------------------------------------------------------------------
C      METODO USADO : O METODO EMPREGADO E' DO TIPO
C                     LAGRANGEANO AUMENTADO.
C
C-----------------------------------------------------------------------
C
C
C      DESCRICAO DAS VARIAVEIS
C
C      X- VETOR PONTO INICIAL NA ENTRADA, NA SAIDA E' O MELHOR PONTO.
C
C      N- NUMERO DE VARIAVEIS
C
C      [illegible] DESIGUALDADES
C
C      [illegible] IGUALDADES
C
C      [illegible] NEGATIVAS,
C         [illegible]
C         [illegible] DESIGUALDADES.
C
C      [illegible]
C         [illegible]
C         [illegible] DESIGUALDADES.
C
C      [illegible] IGUALDADE.
C
C      [illegible] IGUALDADES.
C
C      [illegible]
C
C      [illegible] SAIDA.
C
C      [illegible] AUMENTADO;
C         [illegible] POSITIVO.
C
C      [illegible]
C
C      [illegible] SAIDA:DEVE SER
```

[illegible] PRATICO.

[illegible]- VETOR [illegible] N+2 POSICOES

G[illegible]- VETOR DE [illegible] POSICOES, PARA CONTER AS COMPONENTES DO GRADIENTE [illegible] RESTRICOES DE DESIGUALDADE.

G[illegible]- [illegible] N+2 POSICOES [illegible] PARA RESTRICOES DE IGUALDADE.

A[illegible]- VETOR AUXILIAR DE [illegible] POSICOES

[illegible]- NUMERO TOTAL DE AVALIACOES DE FUNCAO E GRADIENTE

[illegible] PELA SUB-ROTINA MINSS.

[illegible]- PRECISAO [illegible] PARA CONVERGENCIA

[illegible]- NUMERO MAXIMO DE ITERACOES COMPLETAS QUE A SUB-ROTINA MINSS PODE REALIZAR

[illegible]- NUMERO MAXIMO DE VEZES QUE A SUB-ROTINA MINSS PODERA' SER CHAMADA

[illegible]- SE [illegible] = [illegible] SUB-ROTINA MINSS NAO IMPRIME ; SE [illegible] = K> [illegible] SUB-ROTINA MINSS IMPRIMIRA' A CADA K ITERACOES

[illegible]- [illegible] DA MAQUINA E [illegible] (CDC [illegible] (CPD [illegible]). [illegible] 10**(-7)

[illegible]- [illegible] MENOR QUE TOL, [illegible] RAZAO DE MUDANCA [illegible] (VIA DE REGRA).

I[illegible]- [illegible] PARA SAIDA DOS RESULTADOS.

[illegible]- [illegible] DAS MATRIZES(COLUNAS [illegible])

[illegible]- NUMERO TOTAL DE ITERACOES FEITAS PELA SUB-ROTINA MINSS

I[illegible]- [illegible] DE SAIDA
[illegible]=-1 [illegible] RESTRICAO E A CONVERGENCIA DEVE SER ANALISADA PELOS PARAMETROS DA SUB-ROTINA [illegible] ([illegible] O VALOR DE IFLAG)
[illegible]=0 [illegible] CONVERGENCIA [illegible] FORNECIDO
[illegible]=1 [illegible] NUMERO MAXIMO DE CHAMADAS A SUB-ROTINA [illegible] ATINGIDO [illegible] PRECISAO PEDIDA.
[illegible] SAIDA [illegible]=1 SIGNIFICA QUE [illegible] PROBLEMA [illegible] INICIAIS.
I[illegible]=2 [illegible] RESTRICOES DE DESIGUALDADE [illegible] POSSIVEL
I[illegible]=3 [illegible] PARA A FUNCAO OBJETIVO ([illegible] I=1)

[illegible]- VETOR [illegible] DAS RESTRICOES DE DESIGUALDADES

[illegible]- [illegible] PARA RESTRICOES DE IGUALDADE, DEVE SER [illegible].

```
C              [illegible] FUNCAO [illegible] [illegible]NTADO.
C
C       EPSG- [illegible] CONSTANTE POSITIVA PARA A CONVERG NCIA DENTRO DA
C             SUB-ROTINA [illegible].
C
C       IMP- CODIGO PARA IMPRESSAO
C             IMP= 0 [illegible] IMPRESSAO
C             IMP= 1 IMPRIME CADA ITERACAO COM PONTO,VALOR DA FUNCAO OBJE-
C                    TIVO [illegible] RESULTADOS FINAIS.
C             IMP= 2 IMPRIME CADA ITERACAO COM BASTANTE DETALHES,MAIS OS
C                    RESULTADOS FINAIS
C
C       ID- CHAVE, SE ID=1 [illegible] TESTE DE MELHORA OU NAO DA FUNCAO
C                  OBJETIVO [illegible] PARCIAL.
C                  SE ID=2 [illegible] TESTE.
C
C-----------------------------------------------------------------------------
C
C          PROGRAMADORES [illegible] -DEEC/UNICAMP
C
C
C-----------------------------------------------------------------------------
C
      SUBROUTINE [illegible](X,[illegible],XSIG,TETA,X[illegible]T,FOBJ,EPSI,GPSI,
     * [illegible],[illegible],[illegible],[illegible],[illegible],[illegible],EPS,[illegible],[illegible],IMP,ACC,TOL,
     * [illegible],[illegible],[illegible],[illegible],[illegible],[illegible],[illegible],EPSG,IMP,ID)
      DIMENSION X(1), [illegible](1),XSIG(1),TETA(1),X[illegible]T(1),GPSI(1),[illegible](1),
     * [illegible](1),[illegible](1),[illegible](1),[illegible](1),[illegible](1),[illegible](1), IAUX(1),JAUX(1)
        COMMON/[illegible]/ [illegible]
      LOGICAL [illegible]
      K[illegible]=
      [illegible]=
      K[illegible]=
        [illegible]= [illegible](1.,1.[illegible])
      [illegible]=-1.[illegible]
C.*   [illegible]
      IF([illegible]) GO TO 6
      DO 5 J=1,[illegible]
5     X[illegible](J)=[illegible](1.,X[illegible](J))
6     IF([illegible]) GO TO [illegible]
      DO 7 J=1,[illegible]
      XSIG(J)=[illegible](1.,[illegible](J))
7     [illegible](J)=[illegible](1.,[illegible](J))
C
C *   [illegible] DE MELHORA
C
      [illegible] 11 J=1,
      [illegible]
C
C *   [illegible]/[illegible](J)) [illegible] A [illegible]
C *   [illegible] ([illegible](I)/GPSI(I)) . [illegible]
C *   [illegible] APROXIMADA [illegible]
C *   [illegible] VALOR
C *   [illegible] ESTIVER VIOLADA.
C
      [illegible]
1     [illegible]
C
C *   [illegible] TIPO [illegible].
C
```

```
C**   SE [illegible] PARTICULAR EM QUE M=0 E L=0, OU SEJA,
C**   PROBLEMA [illegible] O TESTE DE CONVERGENCIA MAIS A SEGUIR
C**   E' [illegible]
C
3     IF((L.EQ.0).[illegible]) GO TO 333
      IF(L.[illegible]) GO TO [illegible]
      DO 13 I=1,L
      CALL [illegible]
13    [illegible]
C
C**   [illegible] DAS RESTRICOES DE IGUALDADE.
C
      GO TO 333
C
C**   [illegible] TESTE DE CONVERGENCIA . SE AS RESTRICOES DE DESIGUALDADE QUE
C**   SAO VIOLADAS (G>0),FOREM TAIS QUE A SUA NORMA INFINITO
C**   E' MENOR QUE EPS, A PRIMEIRA PARTE DO TESTE E' SATISFEITO. E SE
C**   AS RESTRICOES DE IGUALDADE [illegible] NORMA INFINITO MENOR QUE EPS, A
C**   SEGUNDA PARTE DO TESTE E' SATISFEITO.
C
4       IF([illegible]) GO TO 333
      [illegible]
      [illegible]
      IF([illegible]) GO TO 700
C
C**   [illegible] SEM RESTRICOES.
C
333   CALL [illegible],BETA,XMI,[illegible],EPSG,
     * [illegible],ACC,GRAG,GRAH,GAUX,[illegible])
C
C **  [illegible]
C**   [illegible] OS VALORES [illegible]
C**   [illegible]
C**   [illegible]
C
      [illegible]
C
C
      [illegible] + 1
      [illegible]
        [illegible] + [illegible]
      IF([illegible]) [illegible]
        CALL [illegible],EPSI,GDSI,[illegible],GRAG,[illegible],
        [illegible])
      IF([illegible]) GO TO [illegible]
        [illegible]
        IF([illegible]) GO TO [illegible]
        [illegible](GAUX,[illegible])
5[illegible]     IF([illegible]) GO TO [illegible]
        [illegible] = 1,[illegible]
        IF([illegible](I).[illegible]) GO TO [illegible]
        [illegible](I))
5[illegible]     [illegible]
5[illegible]     CALL [illegible],GDSI,[illegible],GAUX,[illegible],
     * [illegible])
      IF([illegible]) GO TO [illegible]
      [illegible]
      DO [illegible] I=1,[illegible]
C
C *   [illegible] (GAUX) [illegible] MAIOR QUE [illegible]
```

```
C*      [illegible] VALOR FALSO.
C
        IF(G[illegible](I).[illegible]) [illegible]
3[illegible]    CONTINUE
C
C*      S[illegible] ESPACOS DE IGUALDADE, O
C*      P[illegible] (LI[illegible] IGUAL).
          IF([illegible]) [illegible]
C
        IF([illegible]) [illegible]
C
C*      S[illegible] A FUNCAO OBJETIVO DO PROBLEMA,
C*      M[illegible] VERIFICADO, E SE NAO HOUVER
C*      M[illegible] RETORNARA'.
C*      DEVIDO A [illegible] PRECISAO PEDIDA PELO USUARIO
C*      PARA [illegible] FUNCAO PODERA' OU NAO "MELHORAR".
C
5[illegible]      IF(I[illegible]) [illegible]
        IF([illegible]) [illegible]
        F[illegible]=[illegible]
3[illegible]    DO 15 [illegible]=1,[illegible]
        G=[illegible](J)
        A[illegible](J)=[illegible](J)/[illegible](J)))
C
C*      S[illegible] ( [illegible] ) [illegible] VALOR ABSOLUTO
C*      [illegible], [illegible] PA SARA' PELO PR[illegible]
C*      D[illegible].
C
        IF(ABS(G[illegible](I)).[illegible]) [illegible]
        R[illegible]=[illegible](I/[illegible](J))
        X[illegible](J)=X[illegible](J)-[illegible](J)*[illegible](J)
C
C*      S[illegible] ) [illegible] QUE 0.25 E G FOR MAIOR [illegible]
C*      ZERO [illegible] CORRESPONDENTE COMPONENTE [illegible]
C*      PE[illegible] VIOLADA [illegible]
C*      S[illegible].
C
        IF([illegible]) [illegible](J)=[illegible](J)*FATOR
1[illegible]    G[illegible](J)=[illegible]
1[illegible]    CONTINUE
2[illegible]    IF([illegible]) [illegible]
        [illegible] I=1,[illegible]
        [illegible](J)
        IF(ABS([illegible](J)).[illegible]) [illegible]
C
C*      [illegible] FEITAS NO CASO
C*      [illegible]
C
        R[illegible](I/[illegible](I))
        [illegible](J)=[illegible](J)+[illegible]
C
        IF([illegible]) [illegible](J)=[illegible](J)*[illegible]
1[illegible]    [illegible](J)=[illegible]
1[illegible]    CONTINUE
        G[illegible]
7[illegible]    I[illegible]
[illegible]     F[illegible]
        IF([illegible]) [illegible]
        F[illegible]=[illegible]
[illegible]     IF([illegible]) [illegible]
```

```
      DO [illegible] I=1,[illegible]
      IF([illegible](I).[illegible]) [illegible]
      [illegible]=[illegible]([illegible], [illegible](I))
6 [illegible]   CONTINUE
C
6 [illegible]   CALL [illegible](X, [illegible], [illegible], XTAB, [illegible],
     * [illegible], FAIDF, IMP, IDEV)
      RETURN
7[illegible]    I[illegible]=1
         CALL [illegible](X, [illegible], FORJ, FPSI, GPSI, HAUX, GAUX,
        1    IFU ,IFLR, [illegible], IDEV, IMP)
      GO TO 6[illegible]
8 [illegible]   I[illegible]=2
      GO TO 6[illegible]
8 [illegible]   I[illegible]=3
      GO TO 6[illegible]
9 [illegible]   I[illegible]=-1
      GO TO 6[illegible]
      END
C
      SUBROUTINE [illegible](X, [illegible], FORJ, FPSI, GPSI, HAUX, GAUX, IFU,
     * IFLR, [illegible], IDEV, IMP)
      DIMENSION X(1), GPSI(1), [illegible](1), GAUX(1)
      IF([illegible]) RETURN
      WRITE(I[illegible], [illegible]) [illegible]
      WRITE(I[illegible], [illegible])
      WRITE(I[illegible], [illegible]) (X(I), I=1, [illegible])
      IF([illegible].2) [illegible]
      WRITE(I[illegible], [illegible]) [illegible]
      WRITE(I[illegible], [illegible])
      WRITE(I[illegible], [illegible])([illegible](I), I=1, [illegible])
      IF([illegible]) 2
      WRITE(I[illegible], [illegible])
      WRITE(I[illegible], [illegible])([illegible](I), I=1, [illegible])
2     IF([illegible]) [illegible]
      WRITE(I[illegible], [illegible])
      WRITE(I[illegible], [illegible]) ([illegible](I), I=1, [illegible])
3     WRITE(I[illegible], [illegible]) [illegible]
      RETURN
2 [illegible]   FORMAT(///,[illegible],'[illegible] ---> ',I2,1X,'[illegible] = ',
     * [illegible],/)
2 [illegible]   FORMAT([illegible],'[illegible]',//)
2 [illegible]   FORMAT([illegible],/)
2 [illegible]   FORMAT(//,[illegible],'[illegible] = ',[illegible],//)
2 [illegible]   FORMAT(/,[illegible],'[illegible]',//)
2 [illegible]   FORMAT(//,[illegible],'[illegible] LOADF',//)
2 [illegible]   FORMAT(//,[illegible],'[illegible] SIGUALDADE',//)
2 [illegible]   FORMAT(//,[illegible],'[illegible]',/,5X,'[illegible]
     *[illegible] = ',[illegible],/,[illegible],'[illegible]
     *[illegible] = ',[illegible],/,[illegible],'[illegible] DA MINGS = ',I1,/,
        1 [illegible],'[illegible] = ',[illegible],///)
      END
C
C-- -----------------------------------------------------------------------
C
      SUBROUTINE [illegible](X, [illegible], HAUX, XBAR, [illegible],
     *[illegible], IMP, IDEV)
      [illegible]
      DIMENSION [illegible](1), [illegible](1), [illegible](1), [illegible](1), [illegible](1), TETA(1)
      IF([illegible]) [illegible]
```

```
C *     SUB[illegible] PARA IMPRESSAO DOS RESULTADOS FINAIS
        WRITE(IOF7,2[illegible]) [illegible]
        WRITE(IOF7,24)
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
        WRITE(IOF7,32) [illegible]
        WRITE(IOF7,35)
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
        IF([illegible]) GO TO [illegible]
        WRITE(IOF7,[illegible])
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
        WRITE(IOF7,[illegible])
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
2       IF([illegible]) GO TO [illegible]
        WRITE(IOF7,[illegible])
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
        WRITE(IOF7,[illegible])
        WRITE(IOF7,28) ([illegible](I),I=1,[illegible])
5       WRITE(IOF7,[illegible]) [illegible]
        WRITE(IOF7,[illegible]) [illegible],FATOR
        RETURN
2[illegible] FORMAT(101,15('-'),' [illegible] F I N A I S   P A R
       *[illegible] ',I2,1X,15('-'),//)
2[illegible] FORMAT(5X,1[illegible]('-'),' [illegible] OBJETIVO ',10('-'),/)
2[illegible] FORMAT(/,1[illegible](X,[illegible],/),//)
3[illegible] FORMAT(5X,'VALOR DA FUNCAO OBJETIVO : ',G,3X,'VALOR DA FUNCAO LAGR
       *ANGEANA [illegible] . ',[illegible],//)
3[illegible] FORMAT(5X,1[illegible]('-'),' [illegible] DA FUNCAO OBJETIVO ',1[illegible]('-'),/)
4[illegible] FORMAT(5X,1[illegible]('-'),' [illegible] DAS RESTRICOES DE DESIGUALDADE ',1[illegible]('-')
       * ,/)
4[illegible] FORMAT(5X,1[illegible]('-'),' [illegible]. P/ MULTIPLICADORES DE [illegible] PA
       *R[illegible] ',1[illegible]('-'),/)
4[illegible] FORMAT(5X,1[illegible]('-'),' [illegible] RESTRICOES DE IGUALDADE ',1[illegible]('-'),
       * /)
5[illegible] FORMAT(5X,1[illegible]('-'),' [illegible]. P/ MULTIPLICADORES DE [illegible]
       *R[illegible] ',1[illegible]('-'),/)
5[illegible] FORMAT([illegible],' [illegible] : ',
       *[illegible],/,5X,'[illegible] NA VIDA : ',I[illegible],/,5X,
       *'[illegible] DA CIAIS ) : ',I[illegible],/)
6[illegible] FORMAT(5X,'[illegible]',[illegible],' [illegible] : ',[illegible],/,
       *5X,'[illegible] : ',[illegible],3X,'[illegible] : ',[illegible],3X,'[illegible] : ',[illegible],///)
        END
C
C
C
          [illegible]
          [illegible](1)
          IF([illegible]) [illegible]
          [illegible]((1))
          [illegible](-1)
          IF([illegible]) [illegible]
          [illegible]
1         [illegible]
          RETURN
          END
C *       [illegible] PARA CALCULAR [illegible] AUMENTADA
C *       [illegible].
C *
          [illegible],PSIP,GRADS,[illegible],
        1 [illegible]
```

```
      DIMENSION X(1),A[illegible](1) ,[illegible]SIG(1),BETA(1),X[illegible]I(1),GRAPS(1)
     1 ,GF(1),GRAD(1),[illegible](1),[illegible](1),GRAH(1)
      CALL FD (X,F,GF,1)
      PSIF = F
      CALL FD(X,[illegible],GF,[illegible])
      DO 3[illegible] K=1,[illegible]
3     GRAPS(K) = GF(K)
      IF(I.EQ.[illegible]) GO TO [illegible]
      DO 1 I=1,[illegible]
      CALL GR(I,X,G,[illegible],1)
      GRAD(I)= [illegible]
      G[illegible]= G
      CALL GR(I,X,[illegible],[illegible],2)
      T = -A[illegible](I)/A[illegible](I)
      IF(T.GT.T) GO TO 3
      PSIF = PSIF + [illegible].5*A[illegible](I)*T
      GO TO 1
3     PSIF = PSIF + A[illegible](I)*G[illegible] + .5*XSIG(I)*[illegible]*G[illegible]
      DO 7 K=1,[illegible]
7     GRAPS(K) = GRAPS(K) + A[illegible](I)*G[illegible]AG(K) + XSIG(I)*G[illegible]*GR[illegible]G(K)
1     CONTINUE
2     IF(J.EQ.[illegible]) [illegible]
      DO 1[illegible] I=1,[illegible]
      CALL GR[illegible]A(I,[illegible],[illegible],[illegible],1)
      GRAD(I) = [illegible]
      PSIF = PSIF + [illegible](I)*[illegible] + [illegible].[illegible]XMI(I)*[illegible]
      CALL GR[illegible]A(I,X,[illegible],[illegible],2)
      DO 1[illegible] K=1,[illegible]
1     GRAPS(K) = GRAPS(K) + [illegible](I)*G[illegible]H(K) + XMI(I)*[illegible]*GR[illegible](K)
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE MINSS(X,N,NL,LL,F,G,XLAM,XSIG,TETA,XMI,IFUN,ITER,
     1 EPS,NFLAG,MXFUN,FOB,GFO,W,IOUT,IDEV,ACC,GRAG,GRAH,GAUX,HAUX)
      DIMENSION X(1),G(1),W(1),XLAM(1),XSIG(1),TETA(1),XMI(1)
      DIMENSION GRAG(1), GRAH(1),GAUX(1),HAUX(1),GFO(1)
      LOGICAL NOPOS
C
C     INICIALIZAR ITER, IFUN E IOUTK
C
      ITER=0
      IFUN=1
      IOUTK=0
C
C
C     COLOCAR PARAMETROS PARA SUBDIVIDIR W.W(I) CONTEM O VETOR DE
C     BUSCA. W(NX+I) CONTEM A MELHOR ESTIMATIVA PARA O MINIMO.
C     W(NG+I) CONTEM O GRADIENTE NO PONTO ATUAL.
C
      DG=0.
C
      NX=N
C
      NG=NX+N
C
C
      NRY=NG+N
      NRD=NRY+N
      NCONS=5*N
      CALL PSI(X,N,NL,LL,XLAM,XSIG,TETA,XMI,F,G,GFO,GRAG,GRAH,FOB,
     1 GAUX,HAUX)
C
      NFLAG=0
      NRST=N
C
C
C     CALCULO DA DIRECAO INICIAL DE BUSCA, A NORMA DE X AO QUA-
C     DRADO,A NORMA DE G AO QUADRADO."DG1" E' A ATUAL DERIVA-
C     DA DIRECIONAL,ENQUANTO QUE "XSQ" E "GSQ" SAO NORMAS
C     AO QUADRADO
C
      DG1=0.
      XSQ=0.
      DO 30 I=1,N
      W(I)=-G(I)
      XSQ=XSQ+X(I)*X(I)
 3    DG1=DG1-G(I)*G(I)
      GSQ=-DG1
C
C     TESTE SE O PONTO INICIAL E' UM PONTO MINIMO.
C
      AUX1=EPS*EPS*AMAX1(1.0,XSQ)
      IF(GSQ. GT. AUX1)GO TO 40
      IF(IOUT.EQ.0)RETURN
C     WRITE(IDEV,580)  ************( PODERA' SER INCLUIDO SE QUISER IMPRI
      RETURN
C
C
C
 4    FMIN=F
C
C     TESTAR SE O NUMERO MAXIMO DE AVALIACOES FOI USADO
```

```
C
      IF(IFUN.LT.MXFUN)GO TO 45
4     NFLAG=1
      IF(IOUT.EQ.0)RETURN
      WRITE(IDEV,600)
      RETURN
45    NCALLS=IFUN
C
C     TESTAR SE DEVE HAVER IMPRESSAO NESTA ITERACAO.
C
      IF(IOUT .EQ. 0)GO TO 60
      IF(IOUTK .NE. 0)GO TO 50
      WRITE(IDEV,500)ITER,IFUN,FMIN,GSQ
50    IOUTK=IOUTK+1
      IF(IOUTK .EQ. IOUT)IOUTK=0
C
C     COMECO DA BUSCA UNIDIRECIONAL.ALPHA E' O COMPRIMENTO
C     DO PASSO.
60    ALPHA=ALPHA*DG/DG1
C
C     ALPHA=1.0
C
      IF(NRST.EQ.1)  ALPHA=1.0
C
C
C     SE ESTA E' A PRIMEIRA ITERACAO,COLOQUE ALPHA=1.0/SQRT(GSQ),
C     O QUE VAI DETERMINAR QUE O VETOR DE BUSCA INICIAL SEJA UNITARIO.
C
      IF(ITER .EQ. 0)ALPHA=1.0/SQRT(GSQ)
C
C
C     A BUSCA UNIDIRECIONAL AJUSTA UMA CUBICA A "F" E A "DAL"(A
C     FUNCAO E SUA DERIVADA EM ALPHA); E A "FP" E A"DP"(A FUNCAO
C     E A DERIVADA NO PONTO EXPERIMENTAL "AP").INICIALIZAR AP,FP,DP.
      AP=0.
      FP=FMIN
      DP=DG1
C
C
C     GUARDAR A ATUAL DERIVADA,DETERMINAR O PROXIMO VETOR DE BUSCA.
      DG=DG1
C
C
C     ATUALIZE A ITERACAO.
C
      ITER=ITER+1
C
C
C     CALCULAR O ATUAL COMPRIMENTO DO PASSO E GUARDAR O ATUAL "X"
C     E "G"
C
      STEP=0.
      DO 70 I=1,N
      STEP=STEP+W(I)*W(I)
      W(NX+I)=X(I)
70    W(NG+I)=G(I)
      STEP=SQRT(STEP)
C
C
C     COMECO DA BUSCA UNIDIRECIONAL NESTA ITERACAO.TESTE PARA
```

```
C         VERIFICAR POSSIVEL FRACASSO NA BUSCA UNIDIRECIONAL.
C
8         A1=ALPHA*STEP
          IF(A1 .GT. ACC)GO TO 90
          NFLAG=2
          IF(IOUT.EQ.0)RETURN
          WRITE(IDEV,592)
          RETURN
C
C
C         CALCULO DO PONTO EXPERIMENTAL.
C
9         DO 100 I=1,N
100       X(I)=W(NX+I)+ALPHA*W(I)
C
C
C         CALCULO DA FUNCAO NESTE PONTO.
C
          CALL PSI(X,N,M,LL,XLAM,XSIG,TETA,XMI,F,G,GFO,GRAG,GRAH,FOB,
          1 GAUX,HAUX)
C
          IFUN=IFUN+1
          IF(IFUN.LT.MXFUN)GOTO110
          DO 105 I=1,N
          X(I)=W(NX+I)
105       G(I)=W(NG+I)
          F=FMIN
          GOTO42
C         COMPUTAR A DERIVADA DE "F" EM ALPHA.
C
110       DAL=0.0
          DO 120 I=1,N
120       DAL=DAL+G(I)*W(I)
C
C         TESTAR SE FUNCAO NESTE PONTO TEM INCLINACAO NEGATIVA,MAS TEM
C         VALOR MAIOR QUE EM ALPHA=0. NESTE CASO A BUSCA PASSOU POR UM
C         PONTO DE MAXIMO LOCAL,E ESTA' SE DIRIGINDO PARA UM PONTO DE
C         MINIMO LOCAL.
C
          IF((F .GT. FMIN) .AND. (DAL .LT. 0.))GO TO 160
C
C         TESTE SE OS CRITERIOS PARA O COMPRIMENTO DO PASSO ESTAO
C         SENDO VERIFICADOS.
C
          AUX1=0.0001*ALPHA*DG
          AUX1=AUX1+FMIN
          AUX2=ABS(DAL/DG)
          AUX3=0.9
          IF(F .GT. AUX1)GO TO 130
          IF(AUX2 .GT. AUX3)GO TO 130
C
C
C         FORAM VERIFICADOS;TESTAR SE DOIS PONTOS TEM SIDO EXPERIMENTAD
C
          AUX1=IFUN-NCALLS
          AUX2=ABS(DAL/DG)
          IF((AUX1 .LE. 1.) .AND. (AUX2 .GT. EPS) )
          1GO TO 130
          GO TO 170
C
```

```
C
C         PROVAR UM NOVO PONTO.USAR INTERPOLACAO CUBICA PARA ACHAR O
C         PONTO EXPERIMENTAL.
C
130       U1=DP+DAL-3.0*(FP-F)/(AP-ALPHA)
          U2=U1*U1-DP*DAL
          IF(U2 .LT. 0)U2=0.
          U2=SQRT(U2)
          AT=ALPHA-(ALPHA-AP)*(DAL+U2-U1)/(DAL-DP+2.*U2)
C
C
C         TESTAR SE TEM SIDO LOCALIZADO UM INTERVALO ONDE ESTA' O MI-
C         NIMO.
C
          AUX1=DAL/DP
          IF(AUX1 .GT. 0.)GO TO 140
C
C
C         O INTERVALO TEM SIDO ENCONTRADO,TESTAR SE O PONTO EXPERIMEN-
C         TAL PERTENCE AO INTERVALO DETERMINADO.SE NAO,ESCOLHA O PON-
C         TO ME'DIO DO INTERVALO.
C
          AUX1=1.01*AMIN1(ALPHA,AP)
          AUX2=0.99*AMAX1(ALPHA,AP)
          IF((AT .LT. AUX1) .OR. (AT .GT. AUX2))AT=(ALPHA+AP)/2.
          GO TO 150
C
C
C         O INTERVALO AINDA NAO FOI ENCONTRADO,TESTE SE AMBOS OS PON-
C         TOS SAO MAIORES QUE O MINIMO,E O PONTO EXPERIMENTAL E'SUFI-
C         CIENTEMENTE MENOR QUE OS DOIS.
C
C
140       AUX1=0.99*AMIN1(AP,ALPHA)
          IF((DAL .GT. 0.) .AND. (0. .LT. AT) .AND. (AT .LT. AUX1))
          GO TO 155
C
C
C         TESTE SE AMBOS OS PONTOS SAO MENORES QUE O MINIMO,E O PON-
C         TO EXPERIMENTAL E' SUFICIENTEMENTE GRANDE.
C
          AUX1=1.01*AMAX1(AP,ALPHA)
          IF((DAL .LE. 0.) .AND. (AT .GT. AUX1))GO TO 155
C
C
C         SE O PONTO EXPERIMENTAL E' PEQUENO DEMAIS,DUPLIQUE O MAIOR
C         PONTO ANTERIOR.
C
C
          IF(DAL .LE. 0.)AT=2.0*AMAX1(AP,ALPHA)
C
C
C         SE O PONTO EXPERIMENTAL E' GRANDE DEMAIS,DIVIDA POR 2
C         O MENOR PONTO ANTERIOR.
C
C
          IF(DAL .GT. 0.)AT=AMIN1(AP,ALPHA)/2.0
C
C
C         COLOCAR AP=ALPHA,ALPHA=AT E CONTINUE NA BUSCA.
```

```
C
150     AP=ALPHA
        FP=F
        DP=DAL
        ALPHA=AT
        GO TO 80
C
C
C       UM PONTO MAXIMO RELATIVO TEM SIDO PASSADO.DIVIDA ALPHA POR 2
C       E REINICIE A BUSCA.
C
160     ALPHA=ALPHA/2.
        AP=0.
        FP=FMIN
        DP=DG
        GO TO 80
C
C
C       A BUSCA UNIDIRECIONAL TEM CONVERGIDO.TESTAR A CONVERGENCIA
C       DO ALGORITMO.
C
170     GSQ=0.0
        XSQ=0.0
        DO 180 I=1,N
        GSQ=GSQ+G(I)*G(I)
180     XSQ=XSQ+X(I)*X(I)
        AUX1=EPS*EPS*AMAX1(1.0,XSQ)
        IF(GSQ .GT. AUX1)GO TO 185
        IF(IOUT.EQ.0)RETURN
        WRITE(IDEV,500) ITER,IFUN,F,GSQ
        RETURN
C
C
C       A BUSCA CONTINUA.COLOCAR W(I)=ALPHA*W(I).
C
185     DO 190 I=1,N
190     W(I)=ALPHA*W(I)
C
C
C       COMPUTAR O NOVO VETOR DE BUSCA.TESTAR QUE METODO ESTA' SEN-
C       DO USADO.
C
C
C
C       GRADIENTE CONJUGADO E' ATUALIZADO NESTA SECAO.TESTAR SE E'
C       INDICADO UM REINICIO DE "POWELL".
C
        RTST=0.0
        DO 200 I=1,N
200     RTST=RTST+G(I)*W(NG+I)
        IF(ABS(RTST/GSQ) .GT. 0.2)NRST=N
C
C
C       SE UM REINICIO FOR INDICADO,GUARDE O ATUAL"D" E"Y",COMO VETO-
C       RES DE UM REINICIO DE "BEALE";E "D'Y" E "Y'Y" EM W(NCONS+1) E
C       W(NCONS+2) RESPECTIVAMENTE.
C
        IF(NRST .NE. N)GO TO 220
        W(NCONS+1)=0.
        W(NCONS+2)=0.
```

```
      DO 210 I=1,N
      W(NRY+I)=G(I)-W(NG+I)
      W(NRD+I)=W(I)
      W(NCONS+1)=W(NCONS+1)+W(NRY+I)*W(NRY+I)
210   W(NCONS+2)=W(NCONS+2)+W(I)*W(NRY+I)
C
C
C     CALCULO DA HESSIANA VEZES O GRADIENTE ATUAL.
C
220   U1=0.0
      U2=0.0
      DO 230 I=1,N
      U1=U1-W(NRD+I)*G(I)/W(NCONS+1)
230   U2=U2+W(NRD+I)*G(I)*2./W(NCONS+2)-W(NRY+I)*G(I)/W(NCONS+1)
      U3=W(NCONS+2)/W(NCONS+1)
      DO 240 I=1,N
240   W(NX+I)=-U3*G(I)-U1*W(NRY+I)-U2*W(NRD+I)
C
C
C     SE ESTA E' UMA ITERACAO REINICIADA,W(NX+I) CONTEM O NOVO
C     VETOR DE BUSCA.
C
      IF(NRST .EQ. N)GO TO 300
C
C     SE NAO E' UMA ITERACAO REINICIADA, CALCULO DA HESSIANA REINI-
C     CIADA VEZES O ATUAL "Y".
C
250   U1=0.
      U2=0.
      U3=0.
      U4=0.
      DO 260 I=1,N
      U1=U1-(G(I)-W(NG+I))*W(NRD+I)/W(NCONS+1)
      U2=U2-(G(I)-W(NG+I))*W(NRY+I)/W(NCONS+1)
     1+2.0*W(NRD+I)*(G(I)-W(NG+I))/W(NCONS+2)
260   U3=U3+W(I)*(G(I)-W(NG+I))
      STEP=0.
      DO 270 I=1,N
      STEP=(W(NCONS+2)/W(NCONS+1))*(G(I)-W(NG+I))
     1+U1*W(NRY+I)+U2*W(NRD+I)
      U4=U4+STEP*(G(I)-W(NG+I))
270   W(NG+I)=STEP
C
C     CALCULO DA HESSIANA ATUALIZADA,VEZES O GRADIENTE ATUAL PARA
C     OBTER O VETOR DE BUSCA.
C
      U1=0.
      U2=0.
      DO 280 I=1,N
      U1=U1-W(I)*G(I)/U3
280   U2=U2+(1.0+U4/U3)*W(I)*G(I)/U3-W(NG+I)*G(I)/U3
      DO 290 I=1,N
290   W(NX+I)=W(NX+I)-U1*W(NG+I)-U2*W(I)
C
C     CALCULO DA DERIVADA AO LONGO DO NOVO VETOR DE BUSCA.
C
300   DG1=0.
      DO 310 I=1,N
      W(I)=W(NX+I)
310   DG1=DG1+W(I)*G(I)
```

```
C
C         SE A NOVA DIRECAO NAO FOR DE DESCIDA ;PARE.
C
          IF(DG1 .GE. 0.)GO TO 320
C
C         ATUALIZE  "NRST" PARA ASSEGURAR UM REINICIO A CADA N ITERA-
C         COES.
C
          IF(NRST .EQ. N)NRST=0
          NRST=NRST+1
          GO TO 40
C
C         ERROS DE ARREDONDAMENTO, PRODUZIRAM UMA MA' DIRECAO.
C
320       NFLAG=3
          IF(IOUT.EQ.0)RETURN
C 8       FORMAT(' O PONTO INICIAL E'' SOLUCAO ')  *******( SE QUISER INCLU
5 0       FORMAT(  '     FRACASSO  NFLAG=2')
6 0       FORMAT('   FRACASSO NFLAG=1')
5 0       FORMAT(    '      ITERACAO ' ,I5,17H   AVALIACOES  :       ,I6,
          1/'    F =  ',G, '  G-QUADRADO =  ',G)
610       FORMAT('   FRACASSO.NFLAG=3')
          WRITE(IDEV,610)
          RETURN
          END
```

```
C         [illegible] 1
C
C
          S[illegible] G[illegible](I,X,[illegible],G[illegible])
          [illegible] X(3[illegible]),G[illegible]
          G[illegible] (3,[illegible]) [illegible]
3         G=-X(1)
          RETURN
4         DO 1 K=1,[illegible]
1         G[illegible](K) =[illegible].
          G[illegible](1)=-1.
          RETURN
          END
C
C         --------------------------------------------------------------
C
          S[illegible](X,[illegible],[illegible],[illegible])
          [illegible] X(3[illegible]),G[illegible]
          COMMON /[illegible]/[illegible],[illegible],C[illegible]3
          COM[illegible]/[illegible]/GPP([illegible])
          G[illegible] (1,2) [illegible]
1         [illegible]=[illegible].
          [illegible]
3         [illegible]-[illegible]*X(K+1)-[illegible]*X(K+11)-COSTO3*Y(K+21)
          RETURN
2         [illegible]=1,3[illegible]
4         [illegible] = GPP(K)
          RETURN
          END

C
C         --------------------------------------------------------------
C
          [illegible](I,X,[illegible],[illegible],L)
          [illegible] X(3[illegible]),[illegible],[illegible],D2(0/4),D3(0/4)
          [illegible] Y1([illegible]),Y[illegible],Y3([illegible])
          [illegible]/Y[illegible]
          [illegible]/[illegible],B1,C1,[illegible],B2,[illegible],D2/[illegible]3/[illegible],B3,C3,D3
          [illegible]=A1+B1*Z1+[illegible]**2 + [illegible] +
          [illegible] = [illegible]
          [illegible]=-1.-[illegible]*Z1-T3*Z2-[illegible]*Z3
          [illegible]=[illegible]
          [illegible]=X(K+1)
          [illegible]=X(K+11)
          [illegible]=X(K+21)
          Y1(K+1)=X(K+[illegible])
          Y2(K+1)=Y(K+1[illegible])
          Y3(K+1)=X(K+2[illegible])
4         [illegible]
          [illegible]
          [illegible]
          [illegible]
          [illegible]=[illegible](Z1[illegible],Z2[illegible],B1,C1,[illegible],D1([illegible]))
          [illegible]
2         [illegible]
          [illegible]Y1(K+1)-[illegible](Z1(K),[illegible](K),Z3(K),[illegible],B1,C1,[illegible],D1(K))
          [illegible]
3         [illegible](1,[illegible],1[illegible]) [illegible]
          [illegible](1,[illegible],[illegible]) [illegible]
          [illegible](1)-[illegible](Z2[illegible],[illegible],D2([illegible]))
```

```
C
      SUBROUTINE AM1(Y1,Y2,Y3,U1,U2,U3,IT)
      DIMENSION Y1(0/50),Y2(0/50),Y3(0/50)
      DIMENSION U1(0/49),U2(0/49),U3(0/49)
      COMMON/K1/A1,B1,C1,D1/K2/A2,B2,C2,D2/K3/A3,B3,C3,D3
      COMMON/K0/Y10,Y20,Y30
      FF(Z1,Z2,Z3,V1,V2,V3,V4,W)=Z1 + V1*Z1+V2*Z1**2 + V3*Z1*Z2
     1  + V4*Z1*Z3 - W
      Y1(0) =Y10
      Y2(0)=Y20
      Y3(0)=Y30
      DO 1 K=1,IT
      J=K-1
      Y1(K)=FF(Y1(J),Y2(J),Y3(J),A1,B1,C1,D1,U1(J))
      Y2(K)=FF(Y2(J),Y1(J),Y3(J),A2,B2,C2,D2,U2(J))
1     Y3(K)=FF(Y3(J),Y1(J),Y2(J),A3,B3,C3,D3,U3(J))
      RETURN
      END
C
C     ------------------------------------------------------------------
C
      SUBROUTINE P1(IT,X)
      DIMENSION Y1(0/50),Y2(0/50),Y3(0/50),X(300)
      DIMENSION U1(0/49),U2(0/49),U3(0/49)
      COMMON/K0/Y10,Y20,Y30
      Y1(0)=Y10
      Y2(0)=Y20
      Y3(0)=Y30
      TYPE 2
2     FORMAT(2X,'VALORES DE U1,U2,U3 ? ')
      ACCEPT 3,U1(0),U2(0),U3(0)
3     FORMAT(3G)
      DO 7 K=1,(IT-1)
      U1(K)=U1(0)
      U2(K)=U2(0)
      U3(K)=U3(0)
7     CONTINUE
      CALL AM1(Y1,Y2,Y3,U1,U2,U3,IT)
      J=IT-1
      WRITE(23,8)(U1(K),K=0,J),(Y1(K),K=1,IT),
     1 (U2(K),K=0,J),(Y2(K),K=1,IT),(U3(K),K=0,J),
     2 (Y3(K),K=1,IT)
      DO 20 K=0,J
      X(K+1) = U1(K)
      X(IT+K+1)= Y1(K+1)
      X(2*IT+K+1) = U2(K)
      X(3*IT+K+1) = Y2(K+1)
      X(4*IT + K+1) =U3(K)
      X(5*IT+K+1)  = Y3(K+1)
20    CONTINUE
8     FORMAT(6(5G,/))
      RETURN
      END
```

```
        SUBROUTINE HAGA(I,X,H,GRAH,L)
        DIMENSION X(300),GRAH(300),U1(0/49),Y1(0/50)
        1 ,U2(0/49),Y2(0/50),U3(0/49),Y3(0/50)
        COMMON/K0/Y10,Y20,Y30
        COMMON/K1/A1,B1,C1,D1/K2/A2,B2,C2,D2/K3/A3,B3,C3,D3
        F(Z1,Z2,Z3,T1,T2,T3,T4,W)=Z1+T1*Z1+T2*Z1**2+T3*Z1*Z2 +
        1  T4*Z1*Z3-W
        DO 1 K=0,49
        U1(K)=X(K+1)
        U2(K)=X(K+101)
        U3(K)=X(K+201)
        Y1(K+1)=X(K+51)
        Y2(K+1)=X(K+151)
1       Y3(K+1)=X(K+251)
        GO TO (3,40) L
3       Y1(0)=Y10
        Y2(0)=Y20
        Y3(0)=Y30
        IF(I.GT.50) GO TO 2
        H=Y1(I)-F(Y1(I-1),Y2(I-1),Y3(I-1),A1,B1,C1,D1,U1(I-1))
        RETURN
2       IF(I.GT.100) GO TO 5
        H=Y2(I-50)-F(Y2(I-51),Y1(I-51),Y3(I-51),A2,B2,C2,D2,U2(I-51))
        RETURN
5       H=Y3(I-100)-F(Y3(I-101),Y1(I-101),Y2(I-101),A3,B3,C3,D3,U3(I-101))
        RETURN
C
C       CALCULO DOS GRADIENTES
C
40      DO 35 K=1,300
35      GRAH(K)=0.
        IF(I.GT.50)GO TO 7
        IF(I.GT.1) GO TO 8
        GRAH(1) = 1.
        GRAH(51)=1.
        RETURN
8       GRAH(I+49)=-(1.+A1+2.*B1*Y1(I-1)+C1*Y2(I-1)+D1*Y3(I-1))
        GRAH(I+149)=-C1*Y1(I-1)
        GRAH(I+249)=-D1*Y1(I-1)
        GRAH(I)=1.
        GRAH(I+50)=1.
        RETURN
7       IF(I.GT.51)GO TO 9
        GRAH(151) = 1.
        GRAH(101)=1.
        RETURN
9       IF(I.GT.100) GO TO 10
        GRAH(I+99)=-(1.+A2+2.*B2*Y2(I-51)+C2*Y1(I-51)+D2*Y3(I-51))
        GRAH(I-1)=-C2*Y2(I-51)
        GRAH(I+199)=-D2*Y2(I-51)
        GRAH(I+50)=1.
        GRAH(I+100)=1.
        RETURN
1       IF(I.GT.101)GO TO 11
        GRAH(201)=1.
        GRAH(251)=1.
        RETURN
11      GRAH(I+49)=-(1.+A3+2.*B3*Y3(I-101)+C3*Y1(I-101)+D3*Y2(I-101))
        GRAH(I-51)=-C3*Y3(I-101)
        GRAH(I+49)=-D3*Y3(I-101)
```

```
      GRAH(I+100)=1.
      GRAH(I+150)=1.
      RETURN
      END
      SUBROUTINE GE(I,X,G,GRAG,L)
      DIMENSION X(300),GRAG(300)
      GO TO (1,2)L
1     G=-X(I)
      RETURN
2     DO 3 K=1,300
3     GRAG(K)=0.
      GRAG(I)=-1.
      RETURN
      END

      SUBROUTINE FUN(X,F,GF,L)
      DIMENSION X(300),GF(300)
      COMMON/GRADF/GFF(300)
      COMMON/CUSTO/CUSTO1,CUSTO2,CUSTO3
      GO TO (1,2)L
1     F=0.
      DO 3 K=1,50
3     F=F-(X(K)*CUSTO1+X(K+100)*CUSTO2+X(K+200)*CUSTO3)
2     DO 7 K=1,300
7     GF(K) = GFF(K)
      END
```

```
      SUBROUTINE SAI(J,X,IP)
      DIMENSION X(300)
      WRITE(26,7)
    7 FORMAT(1H0,3X,'VALO [illegible] CONTROLE E GRADOS'//)
      DO 1 KC=1,IP
      KK=KC-1
      K1=KK+IP
      K2=K1+IP
      K3=K2+IP
      K4=K3+IP
      K5=K4+IP
    1 WRITE(26,2)KC,X(KC),KC,X(K1),KC,X(K2),KC,X(K3),KC,X(K4),KC,X(K5)
    2 FORMAT(2X,'U1(',I2,')=',F7.4,2X,'Y1(',I2,')=',F7.4,2X,'U2(',I2,'
     1)=',F7.4,2X,'Y2(',I2,')=',F7.4,2X,'U3(',I2,')=',F7.4,2X,'Y3(',I2,
     2')=',F7.4,/)
      RETURN
      END
```