ASPECTOS SOBRE A APLICAÇÃO DE MÉTODOS DE GRADIENTES CONJUGADOS EM OTIMIZAÇÃO IRRESTRITA DIFERENCIÁVEL

Paulo Roberto Torres Borges

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.) EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Paulo Roberto de Oliveira
(Presidente)

Nélson Maculan Filho

Maurício Vieira Kritz

RIO DE JANEIRO, RJ - BRASIL

JANEIRO DE 1985

FICHA CATALOGRÁFICA

BORGES, PAULO ROBERTO TORRES

Aspectos sobre a aplicação de Métodos de Gradientes Conjugados em Otimização Irrestrita Diferenciável (Rio de Janeiro) 1985.

IX , 137 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1985)

Tese - Universidade Federal do Rio de Janeiro, COPPE.

1. PROGRAMAÇÃO NÃO-LINEAR 2. OTIMIZAÇÃO IRRESTRITA 3. MÉTODOS DE GRADIENTES CONJUGADOS COM REINICIALIZAÇÕES 4. ALGORITMOS DE BUSCA LINEAR INEXATA I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

- A meus pais, pela formação básica.

- Aos professores do Programa de Engenharia de Sistemas da COPPE e do Instituto de Matemática da UFRJ, pelos ensinamentos transmitidos durante o curso; em particular, os mestres e amigos:
  - Paulo Roberto de Oliveira, pela orientação neste trabalho;
  - Nélson Maculan Filho, Antonio A.F. de Oliveira e Luis Paulo Vieira Braga, pela amizade e atenção dedicadas.

- Ao CNPq e à CAPES , pelas bolsas de estudos.

- Ao Laboratório de Computação Científica (LCC) do CNPq, nas figuras de:
  - Marco Antonio Raupp e Carlos A. de Moura, pelo apoio institucional;
  - Maurício V. Kritz, Paulo César M. Vieira, Leon R. Sinay e Hélios M.O. Freires Filho, pelas valiosas sugestões fornecidas;
  - Marília P. Braga, Fani G.P. Rodrigues Valle e Sônia Brandão pelo carinho demonstrado na datilografia dos manuscritos;
  - Paulo César Faria, cujos desenhos enriqueceram bastante este trabalho.

- À minha esposa, pela compreensão, apoio, paciência e incentivo constantes, sem os quais este trabalho seria impossível.

- A todos, amigos e familiares que, direta ou indiretamente, contribuíram para a realização do curso e deste trabalho, e que foram involuntariamente omitidos.

DEDICATÓRIA

À *memória de minha avó*,
OLGA AMADOR TORRES ;

À *minha esposa* LINDALVA *e*
ao nosso filhote RAFAEL.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ASPECTOS SOBRE A APLICAÇÃO DE MÉTODOS DE GRADIENTES CONJUGADOS EM OTIMIZAÇÃO IRRESTRITA DIFERENCIÁVEL

Paulo Roberto Torres Borges

Janeiro,1985

Orientador: *Paulo Roberto de Oliveira*
Programa: *Engenharia de Sistemas e Computação*

RESUMO

Neste trabalho, apresentamos os fundamentos teóricos, a evolução histórica e aspectos computacionais de algoritmos de gradientes conjugados. Tais métodos formam uma família de algoritmos em Otimização Irrestrita que geram direções sem a utilização de quaisquer matrizes. Esta característica torna-se bem atraente para problemas onde não se conhece a Hessiana da função-objetivo e/ou para problemas em dimensões elevadas, onde o armazenamento e manipulação desta matriz forem problemáticas. Finalmente, introduzimos conceitos básicos sobre buscas lineares e estudamos recentes estratégias para a solução destes problemas.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

ASPECTS ON THE APPLICATION OF CONJUGATE GRADIENT METHODS IN DIFFERENTIABLE UNCONSTRAINED OPTIMIZATION

Paulo Roberto Torres Borges

January 1985

Chairman: *Paulo Roberto de Oliveira*
Department: *Systems Engineering and Computer Science*

ABSTRACT

In this work, the theoretical foundations, the evolution and computational aspects of conjugate gradient algorithms are presented. Such methods belong to a family of unconstrained optimization algorithms that do not use any matrix for generating directions. This feature becomes rather attractive in problems where the Hessian matrix of the objective-function is not known and/or in large scale unconstrained optimization, whenever the storage and manipulation of that matrix becomes troublesome. Besides that, basic concepts on line searches are introduced and recent strategies to solve this problem are discussed.

ÍNDICE

NOTAÇÃO E DEFINIÇÕES USADAS

$\mathbb{R}^n$ - espaço vetorial euclideano n-dimensional

n - dimensão do domínio da função f

x - variável no $\mathbb{R}^n$

f - função objetivo em problemas de Otimização Irrestrita (O.I.)

g ou ∇f ou f' - gradiente de f

r ou -g ou -f' - antigradiente de f

H ou $\nabla^2 f$ - matriz-Hessiana de f

A - matriz-Hessiana de uma função quadrática

x^* - solução ótima de problemas de O.I.

k - ordem de iteração de algum método de O.I.

ε - escalar positivo arbitrariamente pequeno

i,j - índices

x_{k+1} - ponto do $\mathbb{R}^n$ alcançado na k-ésima iteração de algum método de O.I.

d_k - direção considerada na k-ésima iteração de algum método de O.I.

β_k - coeficiente de conjugação considerado na k-ésima iteração de métodos de gradientes conjugados.

μ,a - escalares de uso geral

t - transposição (como superíndice)

λ_k - passo a ser usado na k-ésima iteração de algum método de O.I.

r - comprimento de um ciclo para algoritmos de gradientes conjugados

Θ - símbolo representando ângulo (uso geral)

m,M - escalares positivos

b - { vetor de constantes n x 1 (capítulos I a III)
escalar entre 0 e 1 (capítulo IV)

$c, \lambda, \hat{\lambda}, \lambda_E \ \lambda_D$ - escalares

DEFINIÇÕES

A_{nxn} é dita matriz definida positiva se e somente se,

$$x^t Ax > 0 \ , \ \forall x \neq 0 \ , \ x \in \mathbb{R}^n \ .$$

Dados $f: \mathbb{R}^n \to \mathbb{R}$ e $\hat{x} \in \mathbb{R}^n$, sempre que $x^t \nabla^2 f(\hat{x}) x > 0$, diremos, por extensão, que f é definida positiva no ponto $\hat{x}$.

O conceito de "taxa de convergência" da sequência $\{x_k\}$ gerada por algum método de O.I. é estabelecido da seguinte forma: seja $\{x_k\}$ uma sequência convergindo para x^*, onde $x_k \neq x^*, \forall k>0$. Se existirem p inteiro positivo e C real e não-nulo, tais que:

$$\lim_{k\to\infty} \frac{\|x_{k+1} - x^*\|}{\|x_k - x^*\|^p} = C \ ,$$

então p é denominado ordem de convergência de $\{x_k\}$ e a taxa de convergência de $\{x_k\}$ é dita:

- LINEAR, se $p = 1$.
- SUPERLINEAR, se $1 < p < 2$.
- QUADRÁTICA, se $p = 2$.

Em particular, a ocorrência da conhecida "taxa de convergência quadrática a cada n iterações" é caracterizada pela existência de um real C não-nulo, tal que:

$$\lim_{k\to\infty} \frac{\|x_{(j+1)n} - x^*\|}{\|x_{jn} - x^*\|^2} = C$$

CAPÍTULO I

INTRODUÇÃO E REVISÃO BIBLIOGRÁFICA

"Recent work in unconstrained minimization has involved the consolidation of existing ideas rather than the dramatic development of new techniques."

(K.W.Brodlie)

Considere o problema geral de otimização não-linear irrestrita diferenciável:

$$\min f(x) \text{ , sujeito a } x \in \mathbb{R}^n \quad , \qquad (1.1)$$

onde $f:\mathbb{R}^n \to \mathbb{R}$, $f \in C^2$. Procuramos portanto,se existir,um ponto x^* tal que $f(x^*) \leq f(x)$, $\forall x \in \mathbb{R}^n$.

Atualmente, a grande parte dos métodos usados na resolução deste problema pode ser classificada em , pelo menos, uma das seguintes famílias:

i) Método do Gradiente Puro ou de Cauchy
ii) Métodos de Direções Conjugadas
iii) Métodos Quasi-Newton
iv) Método de Newton

Devido a duas características muito atraentes, quais sejam, confiabilidade bem razoável e facilidade de programação , os métodos destas famílias se constituem atualmente nas melhores opções na abordagem do problema (1.1) .

A característica comum aos métodos das quatro famílias acima citadas reside em sua natureza iterativa. Geralmente,todos eles evoluem no domínio de f, iteração após iteração, no sentido de alcançar um ponto $\hat{x}$ tal que: $\{f_k\} \to f(\hat{x})$ e $\{\|g_k\|\} \to \|g(\hat{x})\| = 0$.

Para tanto, o mecanismo de evolução de um ponto x_k para o próximo, x_{k+1}, representado esquematicamente por:

$$x_{k+1} = x_k + \lambda_k d_k,\ \lambda_k \in \mathbb{R}^+ \text{ e } d_k \in \mathbb{R}^n, \qquad (1.2)$$

é tal que, dentro de um enfoque de minimização, mantém conveni entemente a chamada propriedade de descida, qual seja:

$$f(x_{k+1}) \leq f(x_k)\ ,\ \forall k>0.$$

Por outro lado, podemos afirmar intuitivamente que o fator que distingue os métodos entre si e que nos leva a classificá-los nas diversas famílias consiste na forma de se obter a direção de descida d_k. Enquanto o método do gradiente puro evolui sempre na direção de maior decréscimo local de f, ou seja, $d_k=-\nabla f(x_k)$, métodos de direções conjugadas evoluem, *ipso facto*, por direções mutuamente conjugadas. Estas famílias utilizam explicitamente apenas informações sobre derivadas de primeira ordem. Por sua vez, o método de Newton progride por direções do tipo $d_k=-[\nabla^2 f_k]^{-1}.\nabla f_k$, isto é, através de projeções de $-\nabla f_k$, enquanto que métodos Quasi-Newton (que aproximam o método de Newton) evoluem por direções do tipo $d_k=-H_k^{-1}\nabla f_k$, onde $\{H_k\}\to\nabla^2 f(x^*)$.

Feita esta breve introdução à Otimização Irrestrita, devemos expressar em algumas palavras a importância dos MGC nessa área, bem como os objetivos deste trabalho. Desenvolvido originariamente por HESTENES [1] em 1952 para a resolução de sistemas lineares, o MGC foi usado pela primeira vez em O.I. por volta de 1964, por FLETCHER e REEVES [2]. O sucesso de tal uso é contemporâneo ao surgimento de métodos Quasi-Newton como o de Davidon (1959), aperfeiçoado em 1963 por Fletcher e Powell, e conhecido (ver referências em [33],p.ex.) como DFP (iniciais dos autores). Se,por um lado,já existia o método do gradiente puro (desde 1847), extremamente simples mas lento na convergência, o método de Newton, embora muito rápido, era operacionalmente complexo, visto necessitar da inversão da Hessiana de f a cada iteração; e,além disso, nem sempre tal Hessiana é explicitamente conhecida. Neste contexto, os MGC e os métodos Quasi-Newton surgiram como alternativas para aqueles métodos, com velocidade de convergência e complexidade de cálculos intermediárias.

Métodos Quasi-Newton, por "aproximarem" mais fielmente o método de Newton, têm se mostrado na prática um pouco mais rápidos que os MGC [13]. Por sua vez, os MGC se baseiam fortemente no fato (geralmente consistente) de que, em uma vizinhança de um mínimo local, toda função pode ser razoavelmente bem aproximada por uma função quadrática. Em termos de complexidade de cálculos, ambos são semelhantes. O grande ponto a favor dos MGC reside no fato de a memória por estes requerida crescer a uma taxa cada vez menor que a dos métodos Quasi-Newton, conforme aumenta a dimensão n do problema. Esta economia de memória torna-se bem atraente em implementações de algoritmos em microcomputadores ou em computadores onde a quantidade de memória disponível for fator restringente. A partir de problemas de médio porte (n>70), pode ser problemático o armazenamento e a manipulação da aproximação da inversa da Hessiana. Sob este aspecto, e não nos esquecendo que as taxas de convergência do MGC e dos métodos Quasi-Newton não são muito diferentes, os MGC devem ser visualizados como a opção ideal.

Neste contexto, nosso trabalho visa não só apresentar a evolução histórica dos MGC, mas, principalmente, dissecar sua idéia básica, seus mecanismos de funcionamento e os aspectos práticos de implementações computacionais de algoritmos de GC. Assim, a apresentação evolui didaticamente sempre que possível e, ao longo de todo o texto, utilizamos fortemente a visão geométrica dos fatos, sendo raras e suscintas as demonstrações.

No segundo capítulo, são apresentados aspectos teóricos sobre a aplicação do MGC à minimização de funções quadráticas da forma:

$$f(x) = \frac{1}{2} x^t A x - b^t x + c \ ,$$

onde A_{nxn} é simétrica, definida positiva, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^n$ e $c \in \mathbb{R}$. Veremos adiante que este problema é equivalente ao de se resolver o sistema linear Ax=b, problema para o qual HESTENES e STIEFEL [1], em 1952, desenvolveram o MGC.

O terceiro capítulo se concentra no estudo de particularidades da aplicação do MGC à minimização de funções não-quadráticas uniformemente convexas. São então apresentadas mo-

dificações e generalizações de conceitos do caso quadrático, necessárias à adaptação do MGC a essa nova classe de funções. O conceito de *reinicialização/"restart"* (o que é, como e quando fazê-la) é apresentado em algumas de suas formas, propiciando o surgimento de novas versões do MGC.

O quarto capítulo versa sobre experiências e idéias obtidas em pesquisas sobre a aplicação do MGC em casos reais, nos quais o uso dos recursos finitos de computadores impõe limites ao MGC, tanto em relação às interrupções (testes de parada) dos processos iterativos inerentes ao método, como em relação a erros acumulados em cálculos intermediários, fatos que, se não contornados adequadamente, podem deteriorar drasticamente não só a velocidade do método, como também a própria convergência do algoritmo. No final, é abordado o problema da minimização unidimensional ou busca linear. São apresentados, dentro de um enfoque computacional, aspectos sobre o que se tem usualmente denominado *"técnicas modernas de buscas lineares"*.

O quinto capítulo apresenta algumas experiências computacionais efetuadas com diferentes versões do MGC aplicadas a um conjunto de funções-teste, como também algumas considerações sobre os resultados observados.

Finalmente, indicamos algumas referências onde os principais aspectos sobre algoritmos de O.I. (como origem , fundamentos teóricos, apresentação dos algoritmos e resultados de convergência) podem ser encontrados. Cronologicamente,temos os textos de MURRAY [16] (1972), LUENBERGER [19] (1973), PSHENICHNY/DANILIN [30] (1975), AVRIEL [25] (1976), BAZARAA/SHETTY [33] (1979) e GILL/MURRAY [40] (1981) que cobrem de uma forma bem abrangente os tópicos de O.I. .

No que diz respeito exclusivamente a métodos de direções conjugadas, nos ativemos aos dois trabalhos de HESTENES, [1] e [38]. Em particular, a evolução histórica dos MGC pode ser acompanhada pelos trabalhos dos seguintes autores: HESTENES e STIEFEL [1], FLETCHER e REEVES [2], POLAK e RIBIÈRE [8], BEALE [17], POWELL [27] e SHANNO [31] e [32]. Embora se constituam apenas em um pequeno sub-conjunto do que tem sido publicado

sobre os MGC, estes trabalhos formam a espinha dorsal da evolução destes métodos,sendo citados na maioria dos demais trabalhos nesta área.

Sobre buscas lineares, são importantes, além dos textos clássicos de O.I. já citados, os trabalhos de LEMARECHAL [34] e de COHEN [39], onde são analisadas as chamadas técnicas modernas de busca linear.

CAPÍTULO II

MGC: MINIMIZAÇÃO DE FUNÇÕES QUADRÁTICAS

Inicialmente, devemos responder à seguinte pergunta: "Por que o estudo do comportamento do MGC em funções quadráticas?" A importância deste estudo se origina basicamente em três fatos:

- funções quadráticas compõem a classe mais simples de funções que podem ser minimizadas (já que as funções lineares, a menos da função constante, são ilimitadas).

- o comportamento de um algoritmo de O.I. em funções quadráticas é, em geral, uma aproximação razoável do comportamento deste algoritmo em uma vizinhança da solução para funções de classe C^2, visto estas poderem ser aproximadas localmente por uma quadrática.

- o MGC tem, como se verá, uma propriedade intrinsecamente ligada à sua aplicação em funções quadráticas, qual seja, a geração progressiva do espaço $\mathbb{R}^n$ pelas direções obtidas.

II.1 - Origem dos MGC:

Os Métodos de Gradientes Conjugados (MGC) foram introduzidos por HESTENES e STIEFEL [1], em 1952, para resolver o sistema linear $Ax = b$, onde:

A_{nxn} - matriz real simétrica def.pos.

x , b - vetores reais nx1

A equivalência do problema acima com a resolução de:

$$\min f(x) = \frac{1}{2} x^t Ax - b^t x + c ,$$

$$x \in \mathbb{R}^n \text{ e } A \text{ simétrica, def. pos.} \qquad (2.1)$$

é estabelecida pelas condições necessárias e suficientes a serem satisfeitas por pontos de mínimo em funções convexas [19]. Será dada agora uma breve noção sobre tal equivalência; aspectos mais detalhados poderão ser encontrados em [16].

O gradiente de f em um ponto x é dado por $g(x) = Ax - b$, e a matriz Hessiana em x é dada por $H(x) = A$. Como um ponto crítico x de f é tal que $g(x) = 0$, temos que:

x é ponto crítico da função quadrática f	<=>	x é solução de Ax = b

O sistema acima, sendo A simétrica def. pos., tem uma única solução x^*, onde $x^* = A^{-1}b$ e, consequentemente, x^* é o único ponto crítico de f. Tomando-se a expansão de Taylor de f em torno de x, dada por:

$$f(x + p) = f(x) + p^t(Ax - b) + \frac{1}{2} p^t Ap , \qquad (2.2)$$

onde o resto é identicamente nulo e, sendo x^* ponto crítico de f, se trocarmos $x \leftarrow x^*$ e $p \leftarrow x - x^*$ em (2.2), teremos:

$$f(x) = f(x^*) + \frac{1}{2}(x-x^*)^t A(x-x^*) . \qquad (2.3)$$

Do ponto de vista geométrico, (2.3) nos mostra que, neste caso, o ponto crítico x^* é o centro das superfícies quadráticas $f(x) = \alpha$, $\alpha \in \mathbb{R}$. Vejamos:

$$f(x) = \alpha = f(x^*) + \frac{1}{2}(x-x^*)^t A(x-x^*) \Rightarrow$$

$$\Rightarrow \alpha - f(x^*) = \text{CONSTANTE} = \frac{1}{2}(x-x^*)^t A(x-x^*) ,$$

que representa uma quadrática centrada em x^*. As superfícies de nível $f(x) = \alpha$ de uma função quadrática definida positiva são

então elipsóides, tendo simultaneamente $x^* = A^{-1}b$ como centro.

Conclusão Final: O problema (2.1) é algebricamente equivalente a se resolver o sistema linear $Ax = b$ e geometricamente equivalente ao de se encontrar o centro de um elipsóide.

II.2 - Idéia Básica dos MGC:

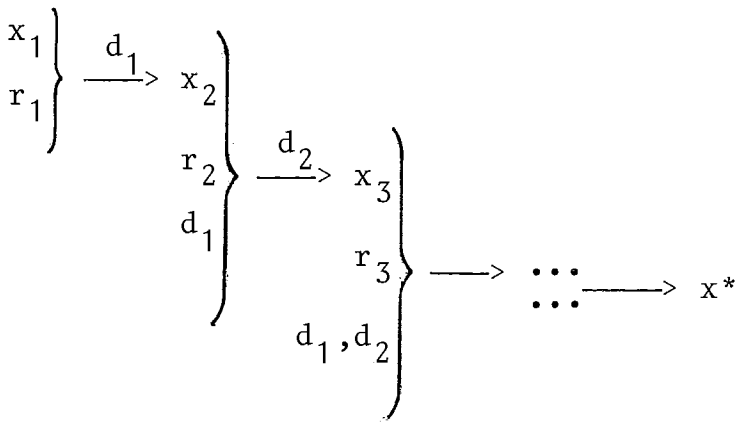
A idéia básica dos MGC pode ser resumida no fato de que usam informações acumuladas, em geral, desde o início da execução do algoritmo para continuarem a progredir em direção do mínimo desejado.

Sabemos que no $\mathbb{R}^n$ existem, no máximo, n vetores não nulos ortogonais entre si. O MGC, quando gera uma direção conjugada às anteriores, gera também um (anti)gradiente ortogonal aos anteriormente gerados (a ser demonstrado no Teorema 4, na próxima seção). Ora, quando já tiverem sido obtidos n (anti)gradientes, o MGC, ao gerar o (n+1)-ésimo, chegará ao vetor nulo, associado a um ponto crítico de f. Como f é quadrática definida positiva, este ponto é o ótimo global [19], podendo então ser terminado o algoritmo.

A partir de agora, denominaremos por $r_k = r(x_k) = -f'(x_k)$ o antigradiente de f no k-ésimo ponto da sequência $\{x_k\}$ gerada pelo MGC (k>0). Cabe também citarmos que, ao longo de todo este capítulo, é suposta a utilização de aritmética exata e que, enquanto nada for explicitamente mencionado, as buscas lineares serão consideradas exatas. Por último, adiantamos que o conceito de conjugação pode ser por enquanto entendido como uma generalização do conceito de ortogonalidade.

O procedimento usado pelo MGC para acumular informações e utilizá-las em sua evolução pode ser assim descrito: inicialmente, dados um ponto inicial x_1 e seu antigradiente r_1, escolhemos a primeira direção de busca $d_1 = r_1$, ao longo da qual será feita uma busca linear para alcançarmos um novo ponto, x_2. Agora, com as informações d_1, r_1 e r_2, calculamos uma nova direção de busca d_2, conjugada a d_1, ao longo da qual buscaremos um novo ponto, x_3, e assim, iterativamente, até qua se atinja a so

lução ótima x*. Esquematicamente, temos:



Dentro de um enfoque de minimização, as direções geradas devem ser de descida. Logo, é conveniente que o antigradiente no ponto atual tenha um peso razoável no cálculo desta direção. Como o método utiliza direções anteriormente obtidas para este cálculo, parece razoável que, por exemplo, estando-se no ponto x_{k+1}, a próxima direção de busca d_{k+1} possa ser escrita da seguinte forma:

$$d_{k+1} = r_{k+1} + \beta_k d_k \qquad , k > 0, \tag{2.4}$$

onde β_k é um escalar escolhido de forma que d_{k+1} seja conjugada a todas as direções já geradas. Desenvolvendo (2.4), concluimos que cada direção pode ser também vista como uma combinação linear dos antigradientes já obtidos, como se nota abaixo:

$$d_1 = r_1$$

$$d_2 = r_2 + \beta_1 d_1 = r_2 + \beta_1 r_1$$

$$d_3 = r_3 + \beta_2 d_2 = r_3 + \beta_2 (r_2 + \beta_1 r_1) =$$

$$= r_3 + \beta_2 r_2 + \beta_2 \beta_1 r_1$$

$$\vdots$$

$$d_{k+1} = \sum_{i=1}^{k+1} \alpha_i r_i \ , \ k > 0$$

Consequentemente, concluímos que no cálculo da direção d_{k+1} estão envolvidas todas as direções encontradas anteriormente. As formas de se calcular β_k, em (2.4), geraram várias versões do MGC, como veremos mais tarde.

Podemos, a esta altura, descrever um algoritmo correspondente a uma versão bem geral do MGC. Vejamos:

ALGORITMO 1: (GERAL)

Passo 0: Dado x_1
$d_1 \leftarrow r_1 \leftarrow -f'(x_1)$
$k \leftarrow 1$

Passo 1: Se $||r_k|| = 0$, pare. O ponto x_k é a solução procurada.

Passo 2: A partir de x_k, fazer uma busca linear ao longo da direção d_k, ou seja, encontrar um passo $\lambda_k \in \mathbb{R}$ tal que este seja solução ótima do problema de minimização unidimensional de f. Este problema equivale a se determinar:

$$\lambda_k \mid f(x_k + \lambda_k d_k) = \min_{\lambda \in \mathbb{R}} f(x_k + \lambda d_k)$$

Passo 3: $x_{k+1} \leftarrow x_k + \lambda_k d_k$ e $r_{k+1} \leftarrow -f'(x_{k+1})$

Passo 4: Calcular o coeficiente de conjugação β_k, de forma que d_{k+1} seja conjugada às direções anteriores.

Passo 5: $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$

Passo 6: $k \leftarrow k+1$
Volte para Passo 1.

II.3 - Relações fundamentais nos MGC:

Definição 1: Um conjunto de vetores d_i, $i = 1,2,\ldots,k (k > 1)$, é dito A-ortogonal ou conjugado em relação à matriz A associada à função quadrática $f(x) = \frac{1}{2} x^t Ax - b^t x + c$ se:

$$d_i^t A d_j = 0, \quad i \neq j, \quad i,j = 1,\ldots,k \quad . \qquad (2.5)$$

Neste capítulo, por simplicidade, sempre que a expressão direções conjugadas (ou A-conjugadas) for usada, subentenderemos que estas são conjugadas em relação à matriz Hessiana simétrica def.pos. A, associada à função quadrática a ser minimizada.

Exemplo: Seja f uma função quadrática tal que:

$$A = \begin{bmatrix} 4 & 0 \\ 0 & 9 \end{bmatrix}$$

Então, por exemplo, $d_1 = (18,1)^t$ e $d_2 = (1/18, -4/9)^t$ são conjugadas, pois $d_1^t A d_2 = 0$.

Vejamos agora os principais resultados teóricos sobre o MGC (no caso quadrático) enunciados pelos seguintes quatro teoremas:

Teorema 1: Vetores conjugados são linearmente independentes.

Seja $\{d_1,d_2,\ldots,d_k\}$ um conjunto de k vetores A-conjugados, $d_i \in \mathbb{R}^n$, $i = 1,\ldots,k$. Então $d_1,d_2,\ldots,d_k$ são l.i. .

Demonstração:

Suponhamos que existam escalares α_i, $i = 1,\ldots,k$, , tais

que :

$$\alpha_1 d_1 + \alpha_2 d_2 + \ldots + \alpha_k d_k = 0 \quad . \tag{2.6}$$

Pré-multiplicando (2.6) por $d_i^t A$, $1 \leq i \leq k$, teremos

$$\alpha_1 d_i^t A d_1 + \ldots + \alpha_i d_i^t A d_i + \ldots + \alpha_k d_i^t A d_k = 0 \quad .$$

Devido à hipótese de conjugação (ver(2.5)), todas as parcelas do lado esquerdo, a menos da i-ésima, são nulas. Temos então: $\alpha_i d_i^t A d_i = 0$.

Como, por hipótese, A é definida positiva, concluímos que $\alpha_i = 0$. Logo, $d_1, d_2, \ldots, d_k$ são linearmente independentes.

■

Corolário 1: Se $d_1, d_2, \ldots, d_k$ são vetores conjugados, então podemos definir o conjunto:

$$S_k = \{x \in \mathbb{R}^n \mid x = \sum_{j=1}^{k} \mu_j d_j, \ \mu_j \in \mathbb{R}, \ (\forall_j)\},$$

que é um subespaço k-dimensional do $\mathbb{R}^n$. Ainda mais, dado $\bar{x} \in \mathbb{R}^n$, o conjunto

$$L_{\bar{x}}(d_1, d_2, \ldots, d_k) = \{x \in \mathbb{R}^n \mid x = \bar{x} + \sum_{j=1}^{k} \mu_j d_j\}$$ é uma variedade afim k-dimensional, paralela a S_k, gerada pelo ponto $\bar{x}$ e pelos vetores $d_1, d_2, \ldots, d_k$.

Definição 2: Dizemos que um algoritmo de otimização irrestrita tem "terminação quadrática" se, quando aplicado a funções no $\mathbb{R}^n$, ele converge para uma solução em, no máximo, n iterações.

Teorema 2: O MGC, quando aplicado a funções quadráticas definidas positivas, possui a propriedade de terminação quadrática. Isto é, assumindo as seguintes hipóteses:

Hipóteses:

H1) Sejam as direções $d_1, d_2, \ldots, d_n$ A-conjugadas ,

H2) A função a ser minimizada é quadrática com sua matriz Hessiana A_{nxn} simétrica def. pos. ,

H3) λ_k é solução ótima da k-ésima busca linear (k = 1, ..., n) ,

H4) x_{k+1} deve ser obtido a partir de x_k por meio de uma busca linear exata ao longo de uma direção conjugada às anteriores, isto é, $x_{k+1} = x_k + \lambda_k d_k$,

provaremos as seguintes teses:

Teses: Para k = 1, ..., n, temos:

T1) $r_{k+1}^t d_j = 0$, j = 1, ..., k (cada antigradiente é ortogonal às direções já obtidas)

T2) $r_1^t d_k = r_k^t d_k$

T3) x_{k+1} é uma solução ótima para o problema: min f(x) sujeito a $x \in L_{x_1}(d_1, \ldots, d_k) = \{x | x = x_1 + \sum_{j=1}^{k} \mu_j d_j\}$

Em particular, x_{n+1} é o mínimo global de f no $\mathbb{R}^n$.

Demonstração:

T1) Para se provar T1 , note que $h(\lambda) = f(x_k + \lambda d_k)$, pela hipótese H3, tem seu valor mínimo em λ_k se e somente se:

$$h'(\lambda_k) = g\,(x_k + \lambda_k d_k)^t \cdot d_k = 0 \Rightarrow$$

$$\Rightarrow r_{k+1}^t \cdot d_k = 0 \qquad (2.7)$$

Para $0 \leq j < k$, observe que:

$$r_{k+1} = b - Ax_{k+1} = b - A(x_{j+1} + \sum_{i=j+1}^{k} \lambda_i d_i) =$$

$$= b - Ax_{j+1} - A \sum_{i=j+1}^{k} \lambda_i d_i \quad , \text{ e, portanto:}$$

$$r_{k+1} = r_{j+1} - A \sum_{i=j+1}^{k} \lambda_i d_i \quad . \tag{2.8}$$

Por H1 , temos $d_k^t A d_i = 0$, $i = 1, \ldots, k-1$.

Pós-multiplicando ambos os termos de (2.8) por d_j, temos:

$$r_{k+1}^t d_j = \underbrace{r_{j+1}^t d_j}_{\substack{\| \\ 0 \\ \text{por(2.7)}}} + \underbrace{\sum_{i=j+1}^{k} \lambda_i d_i^t A d_j}_{\substack{\| \\ 0 \\ \text{por(H1)}}} = 0, \quad j < k \quad . \tag{2.9}$$

Logo, a partir de (2.7) e (2.9), concluímos a validade de T1 .

T2) Para k=1, T2 é trivial. Trocando k por k-1 e tomando j = 0 em (2.8), temos:

$$r_k = r_1 - A \sum_{i=1}^{k-1} \lambda_i d_i, \quad k \geq 2 \ . \tag{2.10}$$

Pós-multiplicando (2.10) por d_k, temos:

$$r_k^t d_k = r_1^t d_k - \sum_{i=1}^{k-1} \lambda_i d_i^t A d_k,$$

onde o somatório, pela hipótese H1, vale zero. Então:

$$r_1^t d_k = r_k^t d_k, \quad k \geq 2.$$

Logo, T2 vale.

T3) Dividiremos esta demonstração em 4 partes, a saber:

i) obtenção do valor de f(x), para um ponto qualquer $x \in L_{x_1}(d_1,d_2,\ldots,d_k)$.

ii) obtenção do valor $f(x_{k+1})$, onde $x_{k+1} \in L_{x_1}(d_1,d_2,\ldots,d_k)$ é o (k+1)-ésimo ponto da sequência gerada pelo MGC.

iii) processo que nos permitirá estabelecer que $f(x_{k+1}) \leq f(x)$.

iv) generalização do resultado de (iii) para k=n.

Usando o corolário 1 do Teorema 1 e a hipótese H1, podemos, para um ponto genérico $x \in L_{x_1}(d_1,d_2,\ldots,d_k)$, escrever:

$$x = x_1 + \sum_{j=1}^{k} \mu_j d_j \ , \quad \mu_j \in \mathbb{R} \ . \tag{2.11}$$

Pela definição da função f,

$$f(x) = f(x_1 + \sum_{j=1}^{k} \mu_j d_j) =$$

$$= \frac{1}{2}(x_1 + \sum_{j=1}^{k} \mu_j d_j)^t A(x_1 + \sum_{j=1}^{k} \mu_j d_j) - b^t(x_1 + \sum_{j=1}^{k} \mu_j d_j) + c =$$

$$= f(x_1) + (Ax_1 - b)^t \sum_{j=1}^{k} \mu_j d_j + \frac{1}{2} \sum_{j=1}^{k} \mu_j^2 d_j^t A d_j \ ,$$

devido à hipótese de conjugação H1. Logo, temos:

$$f(x) = f(x_1) - r_1^t \sum_{j=1}^{k} \mu_j d_j + \frac{1}{2} \sum_{j=1}^{k} \mu_j^2 d_j^t A d_j \ . \tag{2.12}$$

Por construção do algoritmo, podemos expressar x_{k+1} de forma análoga à de (2.11), ou seja,

$$x_{k+1} = x_1 + \sum_{j=1}^{k} \lambda_j d_j \ , \ \lambda_j \in \mathbb{R} \ , \tag{2.13}$$

onde os valores de λ_j satisfazem à hipótese H3.

Aplicando (2.12) para x_{k+1}, obtemos:

$$f(x_{k+1}) = f(x_1) - r_1^t \sum_{j=1}^{k} \lambda_j d_j + \frac{1}{2} \sum_{j=1}^{k} \lambda_j^2 d_j^t A d_j \ . \tag{2.14}$$

Passaremos agora à fase (iii) da demonstração. Inicialmente, pela hipótese H3, temos :

$$f(x_j + \lambda_j d_j) \leq f(x_j + \mu_j d_j), \ \forall \mu_j \in \mathbb{R} \text{ e } j \leq n ,$$

que, através do mesmo desenvolvimento de (2.12) a ambos os seus lados, pode ser re-escrita como abaixo:

$$f(x_j) - \lambda_j r_j^t d_j + \lambda_j^2 d_j^t A d_j \leq f(x_j) - \mu_j r_j^t d_j + \mu_j^2 d_j^t A d_j . \quad (2.15)$$

Pela tese T2, temos $r_j^t d_j = r_1^t d_j$, $\forall j \leq n$. Fazendo esta substituição em (2.15), obtemos:

$$- \lambda_j r_1^t d_j + \lambda_j^2 d_j^t A d_j \leq - \mu_j r_1^t d_j + \mu_j^2 d_j^t A d_j . \quad (2.16)$$

Tomando (2.16) com $j=1,\ldots,k$ e somando-os, temos:

$$- r_1^t \left(\sum_{j=1}^{k} \lambda_j d_j \right) + \frac{1}{2} \sum_{j=1}^{k} \lambda_j^2 d_j^t A d_j \leq - r_1^t \left(\sum_{j=1}^{k} \mu_j d_j \right) + \frac{1}{2} \sum_{j=1}^{k} \mu_j^2 d_j^t A d_j . \quad (2.17)$$

Adicionando o termo $f(x_1)$ a ambos os lados de (2.17) e por inspecção de (2.12) e (2.14), concluimos diretamente que:

$$f(x_{k+1}) \leq f(x) , \ \forall \ x \in L_{x_1}(d_1, d_2, \ldots, d_k).$$

Em particular, $L_{x_1}(d_1, d_2, \ldots, d_n) = \mathbb{R}^n$, $\forall \ x_1 \in \mathbb{R}^n$, e daí:

x_{n+1} minimiza f sobre o $\mathbb{R}^n$.

■

Teorema 3: Seja a função a ser minimizada $f(x) = \frac{1}{2}x^tAx - b^tx + c$, onde A_{nxn} é simétrica, def.pos. e $x \in \mathbb{R}^n$.
Então, o valor λ_k ($k = 1, \ldots, n$) do passo ótimo na k-ésima busca linear nos MGC é dada por:

$$\lambda_k = \frac{d_k^t r_k}{d_k^t A d_k} = \frac{\|r_k\|^2}{d_k^t A d_k} \qquad (2.18)$$

Demonstração:

Pelo teorema 2, existem $\lambda_k \in \mathbb{R}$ tais que:

$$x^* - x_1 = \lambda_1 d_1 + \lambda_2 d_2 + \ldots + \lambda_n d_n \quad . \qquad (2.19)$$

Pré-multiplicando (2.19) por $d_k^t A$, temos:

$$d_k^t A(x^* - x_1) = \lambda_k d_k^t A d_k \Rightarrow \lambda_k = \frac{d_k^t A(x^* - x_1)}{d_k^t A d_k} \quad . \qquad (2.20)$$

Por construção do algoritmo, podemos escrever:

$$x_k - x_1 = \lambda_1 d_1 + \lambda_2 d_2 + \ldots + \lambda_{k-1} d_{k-1} \quad . \qquad (2.21)$$

Pré-multiplicando (2.21) por $d_k^t A$, temos:

$$d_k^t A(x_k - x_1) = 0 \quad . \qquad (2.22)$$

A partir de (2.20), podemos escrever:

$$\lambda_k = \frac{d_k^t A(x^* - x_1)}{d_k^t A d_k} = \frac{d_k^t A(x^* - x_k + (x_k - x_1))}{d_k^t A d_k} =$$

$$= \frac{d_k^t A(x^*-x_k)+d_k^t A(x_k-x_1)}{d_k^t\, Ad_k} \overset{\uparrow}{=}_{(2.22)}$$

$$= \frac{d_k^t A(x^*-x_k)}{d_k^t\, Ad_k} = \frac{d_k^t (b-Ax_k)}{d_k^t\, Ad_k} = \frac{d_k^t r_k}{d_k^t\, Ad_k} \quad .$$

Concluindo,

$$d_k^t r_k = (r_k + \beta_{k-1} d_{k-1}) r_k^t = r_k^t r_k + \underbrace{\beta_{k-1} d_{k-1}{}^t r_k}_{\text{''}\,0} \quad , \text{ de onde,}$$

pela tese T1 do teorema 2, temos

$$\boxed{d_k^t r_k = \| r_k \|^2} \quad \blacksquare$$

Teorema 4: No caso quadrático, os (anti)gradientes gerados pelos M.G.C. são ortogonais.

Demonstração:

Mostraremos, por indução, que, $1 \leq k \leq n$ e fixo, $r_{k+1}^t r_j = 0, j=1,\ldots,k$. Para $j=1$, a prova é trivial, devido ao Teorema 2. Supondo-se válido o Teorema 4 para $j=i-1$ $(i \leq k)$, isto é:

$$r_{k+1}^t d_{i-1} = 0 \quad , \qquad (2.23)$$

provaremos a sua validade para $j=i$. De fato, temos:

$$r_{k+1}^t r_i = r_{k+1}^t (b-Ax_i) = r_{k+1}^t (b-A(x_{i-1}+\lambda_{i-1} d_{i-1})) =$$

$$= r_{k+1}^t (r_{i-1}-\lambda_{i-1} Ad_{i-1}) = r_{k+1}^t r_{i-1} - \lambda_{i-1} r_{k+1}^t Ad_{i-1}) =$$

$$\overset{(2.23)}{=} -\lambda_{i-1} r_{k+1}^t Ad_{i-1} \overset{(2.18)}{=} -\frac{\| r_{i-1} \|^2}{d_{i-1}^t Ad_{i-1}} \cdot r_{k+1}^t Ad_{i-1} =$$

$$= -\frac{r_{i-1}^t (r_{i-1}^t r_{k+1}) Ad_{i-1}}{d_{i-1}^t Ad_{i-1}} \overset{(2.23)}{=} 0. \quad \text{Logo, temos:}$$

$$r_{k+1}^t r_j = 0, \ (j = 1, \ldots, k) \ , \quad 1 \leq k \leq n \ ,$$

isto é:

os antigradientes são mutuamente ortogonais. ■

Este teorema nos mostra que o MGC, ao progredir por direções conjugadas, gera antigradientes mutuamente ortogonais.Os resultados que apresentamos podem ser esquematicamente dispostos conforme abaixo e cabe lembrarmos que as demonstrações expostas não são únicas.

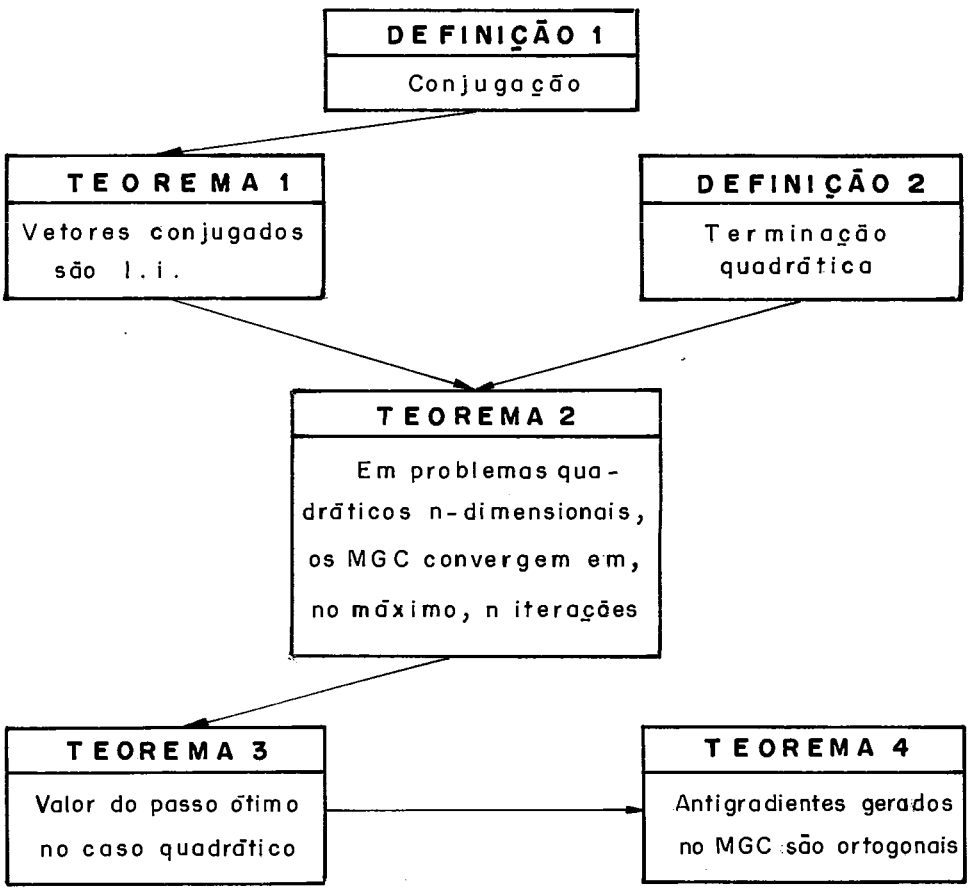


Fig.II.1

A questão que agora se levanta é a respeito das formas de se obter direções conjugadas e de descida.

II.4 - Conjugando direções:

Pelo que foi apresentado e, principalmente, revendo-se a definição de conjugação, podemos visualizar o processo de conjugação de vetores como uma generalização do processo de ortogonalização; os métodos de gradientes conjugados trabalham com bases conjugadas ao invés de ortogonais. Logo,o problema fundamental é o do desenvolvimento de algoritmos que construam vetores mutuamente conjugados.

Como já observado em II.2, a nova direção de descida d_{k+1}, a partir do último ponto alcançado (x_{k+1}),pode ser definida como uma combinação linear de todas as direções anteriormente geradas (representadas na última direção d_k) com o antigradiente r_{k+1}, obtido neste ponto (x_{k+1}). Temos:

Geometricamente:

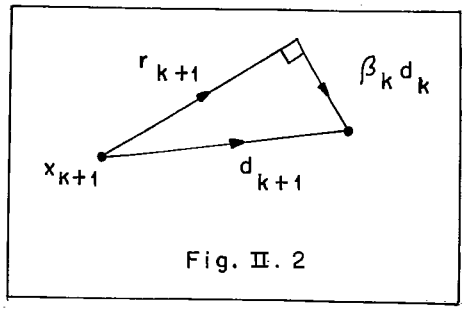


Fig. II. 2

Analiticamente:

$$d_{k+1} = r_{k+1} + \beta_k d_k, \ \beta_k \in \mathbb{R} \qquad (2.24)$$

O escalar β_k, que define a combinação linear acima, é escolhido de forma a garantir que a nova direção seja conjugada a todas as anteriores, ou, equivalentemente à última direção gerada (d_k). Logo, devemos ter:

$$d_{k+1}^t \, Ad_k = r_{k+1}^t \, Ad_k + \beta_k d_k^t \, Ad_k = 0 \quad . \qquad (2.25)$$

De (2.25), temos a fórmula de Daniel [25]:

$$\boxed{\beta_k = - \frac{r_{k+1}^t \, Ad_k}{d_k^t \, Ad_k}} \qquad (2.26)$$

Definindo $y_{k+1} = r_{k+1} - r_k$, podemos escrever:

$$y_{k+1} = (b - Ax_{k+1}) - (b - Ax_k) = A(x_k - x_{k+1}) =$$

$$= A(x_k - x_k - \lambda_k d_k) \Rightarrow$$

$$\Rightarrow \quad y_{k+1} = - \lambda_k \, Ad_k \quad . \qquad (2.27)$$

Multiplicando e dividindo o lado direito de (2.26) por $- \lambda_k$ e, usando (2.27), temos:

$$\boxed{\beta_k = - \frac{r_{k+1}^t \, y_{k+1}}{d_k^t \, y_{k+1}}} \quad , \qquad (2.28)$$

conhecida como a fórmula de Hestenes-Stiefel, segundo LENARD [23]; no entanto, AVRIEL em [25], atribui (2.28) a Sorenson e Wolfe. Aqui, nos referimos a (2.28) sempre de primeira forma.

Podemos simplificar (2.28) se usarmos buscas linea - res exatas, pois a tese T1 do Teorema 2 nos permite escrever:

$$d_k^t r_k = (r_k + \beta_{k-1} d_{k-1})^t r_k = r_k^t r_k + \underbrace{\beta_{k-1} d_{k-1}^t \, r_k}_{\| \; 0} \Rightarrow$$

$$\Rightarrow d_k^t \, r_k = \| r_k \|^2 \quad ,$$

o que nos leva a:

$$\beta_k = - \frac{r_{k+1}^t y_{k+1}}{(\underbrace{d_k^t r_{k+1}}_{0} - d_k^t r_k)} = - \frac{r_{k+1}^t y_{k+1}}{- \| r_k \|^2} \quad \Rightarrow$$

$$\Rightarrow \quad \boxed{\beta_k = \frac{r_{k+1}^t y_{k+1}}{\| r_k \|^2}} \quad , \qquad (2.29)$$

que foi proposta por POLAK e RIBIÈRE, em [6].

A última simplificação baseia-se no fato de a função ser quadrática, o que, aliado à condição de termos $d_1 = r_1$, nos permite usar o Teorema 4 da seguinte forma:

$$\beta_k = \frac{r_{k+1}^t r_{k+1} - \overbrace{r_{k+1}^t r_k}^{0}}{\| r_k \|^2} \quad \Rightarrow \quad \boxed{\beta_k = \frac{\| r_{k+1} \|^2}{\| r_k \|^2}}, \qquad (2.30)$$

fórmula usada por FLETCHER e REEVES [2]. Repare que o uso de (2.30) implica em assumirmos o uso de buscas lineares exatas.

II.5 - Um Algoritmo de Gradientes Conjugados Para Funções Quadráticas Definidas Positivas:

Podemos agora apresentar um algoritmo relativo à versão dos MGC destinada a minimizar uma função quadrática estritamente convexa. Daqui em diante, este algoritmo será denominado Algoritmo 2. Dada uma função f, como definida em (2.1), temos:

Passo 0: Dados $x_1 \in \mathbb{R}^n$, $r_1 \leftarrow -f(x_1) (= b - Ax_1)$ e $d_1 \leftarrow r_1$

Passo 1: Para k = 1 até n

faça

1.1: Se $\| r_k \|^2 = 0$, pare. $x^* = x_k$ é o mínimo desejado.

1.2: $\lambda_k \leftarrow \| r_k \|^2 / (d_k^t A d_k)$

1.3: $x_{k+1} \leftarrow x_k + \lambda_k d_k$

1.4: $r_{k+1} \leftarrow b - Ax_{k+1}$

1.5: $\beta_k \leftarrow \| r_{k+1} \|^2 / \| r_k \|^2$

1.6: $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$

Passo 2: Pare. $x^* = x_{n+1}$ é o mínimo desejado.

Observações sobre o Algoritmo 2:

i) Sobre os pontos de parada:

- a propriedade de terminação quadrática é usada, ao final do algoritmo, como indicador de garantia total de, no caso quadrático e com a utilização de aritmética exata, se ter alcançado o mínimo global da função f.

- sendo f quadrática def. pos., a condição $\nabla f(x_k) = 0$ (equivalente a $\| r_k \|^2 = 0$) é necessária e suficiente de otimalidade para o ponto x_k e, automaticamente, detecta uma possivel convergência precoce do algoritmo.

ii) O passo λ_k é dado de forma explícita, como obtido no Teorema

3, fazendo com que se evite o uso de processos iterativos para efetuar buscas lineares.

iii) Podemos obter 1.4 de outra forma, visto que:

$$r_{k+1} = b - Ax_{k+1} = b - A(x_k + \lambda_k d_k) = r_k - \lambda_k A d_k \ .$$

iv) A fórmula escolhida para β_k no passo 1.5 foi a de Fletcher-Reeves, visto ela ser a que melhor utiliza as características da função quadrática definida positiva a ser minimizada e as propriedades dos MGCs.

II.6 - MGC versus Método do Gradiente: Avaliação de desempenho:

Após avaliar em termos absolutos o desempenho dos MGCs através dos resultados do Teorema 2, relacionaremos agora este desempenho com o do tradicional Método do Gradiente, ou Método de Cauchy, quando aplicados a funções quadráticas definidas positivas.

Sem perda de generalidade, assumamos a função f a ser minimizada do tipo $f(x) = \frac{1}{2}x^t Ax$, $x \in \mathbb{R}^n$, A_{nxn} sendo simétrica, definida positiva, e seu antigradiente, em ponto x_k, dado por $r_k = -g(x_k) = -Ax_k$. Ambos os métodos operam com as seguintes relações:

$$x_{k+1} = x_k + \lambda_k d_k \ , \quad \lambda_k = \| r_k \|^2 / (d_k^t A d_k)$$

e, consequentemente:

$$f(x_{k+1}) = \frac{1}{2} \left(x_k + \frac{\| r_k \|^2}{d_k^t A d_k} \ d_k\right)^t \left(Ax_k + \frac{\| r_k \|^2}{d_k^t A d_k} \ Ad_k\right)$$

O resultado da comparação do desempenho de uma iteração do MGC (s.p.g., Algoritmo 2) e do Método do Gradiente pode ser mostrado através da seguinte relação, demonstrada após seu enunciado:

> A partir de um mesmo ponto x_k, a redução do valor da função por uma iteração do MGC é sempre maior ou igual que a redução obtida por uma iteração do Método do Gradiente, isto é:
>
> $$f(x_{k+1}^{MGC}) \leq f(x_{k+1}^{MG}) \qquad (2.31)$$

Demonstração:

Suponha que ambos os métodos, na k-ésima iteração, estejam em um mesmo ponto x_k, a partir do qual evoluirão através da direção d_k dada por:

$$d_k = \begin{cases} r_k, \text{ se Método do Gradiente,} \\ r_k + \beta_{k-1} d_{k-1}, \text{ se MGC,} \end{cases}$$

alcançando, respectivamente, os pontos x_{k+1}^{MG} e x_{k+1}^{MGC}.

Inicialmente, vamos avaliar $f(x_{k+1}^{MG})$:

$$f(x_{k+1}^{MG}) = \frac{1}{2}(x_k + \frac{\| r_k \|^2}{r_k^t A r_k} r_k)^t (-r_k + \frac{\| r_k \|^2}{r_k^t A r_k} Ar_k) =$$

$$= \frac{1}{2}(-x_k^t r_k + \frac{\| r_k \|^2}{r_k^t A r_k} x_k^t A r_k) =$$

$$= \frac{1}{2}\left(r_k^t A^{-1} r_k - \frac{\| r_k \|^2}{r_k^t A r_k} \cdot \| r_k \|^2\right) \Rightarrow$$

$$f(x_{k+1}^{MG}) = \frac{1}{2}\left(r_k^t A^{-1} r_k - \frac{\| r_k \|^4}{r_k^t A r_k}\right) .$$

Analogamente, para $f(x_{k+1}^{MGC})$, temos:

$$f(x_{k+1}^{MGC}) = \frac{1}{2}(x_k + \lambda_k d_k)^t (Ax_k + \lambda_k A d_k) =$$

$$= \frac{1}{2}(-A^{-1} r_k + \lambda_k d_k)^t (-r_k + \lambda_k A d_k) =$$

$$= \frac{1}{2}(r_k^t A^{-1} r_k - 2\lambda_k r_k^t d_k + \lambda_k^2 d_k^t A d_k) =$$

$$= \frac{1}{2}\left(r_k^t A^{-1} r_k - 2 \frac{d_k^t r_k}{d_k^t A d_k} \cdot r_k^t d_k + \frac{(d_k^t r_k)^2}{d_k^t A d_k}\right) =$$

$\uparrow$ def. de λ_k

$$= \frac{1}{2}\left(r_k^t A^{-1} r_k - \frac{(d_k^t r_k)^2}{d_k^t A d_k}\right) \Rightarrow$$

$$\Rightarrow f(x_{k+1}^{MGC}) = \frac{1}{2}\left(r_k^t A^{-1} r_k - \frac{\| r_k \|^4}{d_k^t A d_k}\right) .$$

Assim, para verificar a validade de (2.31), basta provarmos que:

$$r_k^t A r_k \geq d_k^t A d_k \ .$$

Porém,

$$d_k^t A d_k = (r_k + \beta_{k-1} d_{k-1})^t A d_k =$$

$$= r_k^t A d_k + \underbrace{\beta_{k-1} d_{k-1}^t A d_k}_{\| \; 0} =$$

$$= r_k^t A (r_k + \beta_{k-1} d_{k-1}) =$$

$$= r_k^t A r_k + \beta_{k-1} r_k^t A d_{k-1} ,$$

o que nos permite re-escrever a desigualdade a ser demonstrada como abaixo:

$$r_k^t A r_k \geq r_k^t A r_k + \beta_{k-1} r_k^t A d_{k-1},$$

ou, equivalentemente:

$$0 \geq \beta_{k-1} r_k^t A d_{k-1},$$

que pode ser facilmente verificada, pois:

$$\beta_{k-1} r_k^t A d_{k-1} = \beta_{k-1} (d_k - \beta_{k-1} d_{k-1})^t A d_{k-1} =$$

$$= \beta_{k-1} \underbrace{(d_k^t A d_{k-1}}_{\| \; 0} - \beta_{k-1} d_{k-1}^t A d_{k-1}) =$$

$$= -\beta_{k-1}^2 d_{k-1}^t A d_{k-1} \leq 0 .$$

Logo, (2.31) vale.

Corolário: Visto o Método do Gradiente ter uma taxa de convergência linear, concluímos que o MGC (Algoritmo 2) terá taxas de convergência, no pior caso, lineares, quando aplicados a funções quadráticas definidas positivas.

II.7 - Relação entre o M.G.C. e o Método de Newton:

Para definirmos o chamado Método Puro de Newton (MPN) aplicado à minimização de uma função genérica f, exigiremos , a princípio, $f \in C^2$. Poderemos então, escrever sua aproximação quadrática em uma vizinhança de x_k em termos da expansão de Taylor::

$$f(x) \cong F^k(x)=f(x_k) + (x-x_k)^t g_k + \frac{1}{2}(x-x_k)^t H_k (x-x_k) \quad , \qquad (2.32)$$

onde: g_k denota o gradiente em x_k , H_k é a Hessiana de f no mesmo ponto e x, $x_k \in \mathbb{R}^n$.

Derivando (2.32) em relação a x, temos:

$$f'(x) = g(x) \cong g_k + H_k(x-x_k) \, . \qquad (2.33)$$

Sabendo que uma condição necessária (e em certos casos suficiente) para existência de um mínimo x^* de f é de que $g(x^*) = 0$, buscaremos um ponto x_{k+1} tal que:

$$0 = g(x_{k+1}) = g_k + H_k (x_{k+1} - x_k) \, , \qquad (2.34)$$

ou, equivalentemente, assumindo H_k não singular:

$$x_{k+1} = x_k - H_k^{-1} g_k \qquad (2.35)$$

O MPN consiste na aplicação iterativa de (2.35) a partir de um ponto $x_1 \in \mathbb{R}^n$.

A cada iteração precisamos calcular o gradiente e a inversa da Hessiana no ponto x_k.. Na verdade, não se inverte diretamente H_k, resolvendo-se, em seu lugar, o sistema linear dado por (2.34).

Em dimensões altas e/ou em problemas complexos, os cál

culos dos valores de $f(x_k)$, $g(x_k)$ e, principalmente, da inversa de H_k podem ser custosos, no que diz respeito a tempo (quantidade de cálculos). Um outro problema bastante frequente na aplicação do MPN a funções não-quadráticas é o de que não se pode garantir sempre que $f(x_{k+1}) < f(x_k)$, problema este que tem, entre outras causas, a de que H_k pode não ser def.pos. ou de que x_{k+1}, obtido por (2.35), estar suficientemente afastado de x_k de modo a tornar a aproximação quadrática de f, em uma região onde se encontre x_k, uma aproximação ruim para $f(x_{k+1})$. Soluções para contornar estes problemas tem sido propostas (ver [36] e [38]), em particular o uso de busca linear para cada direção $- H_k^{-1} g_k$.

Sendo f quadrática def. pos., o MPN adquire a propriedade atraente de convergir para o mínimo global x* em apenas uma iteração, a partir de qualquer ponto $x_1 \in \mathbb{R}^n$. Note que, se $f(x) = \frac{1}{2} x^t Ax - bx + c$, aplicando o MPN, temos:

$$x_2 = x_1 - A^{-1} g_1 ,$$

e, consequentemente:

$$g(x_2) = Ax_2 - b = Ax_1 - g_1 - b = 0 \quad \Rightarrow$$

$$\Rightarrow x_2 = x^* \text{ é o mínimo global de f.}$$

O MPN deve ser visto como uma sequência de minimizações de aproximações quadráticas da função f a ser minimizada. À k-ésima iteração de Newton corresponde a obtenção do mínimo x_{k+1} da aproximação quadrática F^k de f.

Conclusão Final:

Comparando o MPN e o MGC, e tendo em vista o Teorema 2, concluímos que, para funções quadráticas def. pos., uma iteração de Newton corresponde a um ciclo de (no máximo) n buscas lineares ao longo de n direções mutuamente conjugadas.

CAPÍTULO III

M.G.C.: MINIMIZAÇÃO DE FUNÇÕES NÃO-QUADRÁTICAS UNIFORMEMENTE CONVEXAS

III.1 - Apresentação do Problema:

Geralmente, um algoritmo assume e/ou utiliza propriedades da classe de problemas para a qual foi desenvolvido. No nosso caso, os MGCs se baseiam em características próprias de funções quadráticas convexas (continuamente diferenciáveis). Diversos trabalhos têm sido desenvolvidos ao longo dos últimos 20 anos no sentido de aplicá-los a funções mais gerais, tendo-se alcançado algum sucesso principalmente em funções que sejam razoavelmente bem aproximadas por uma quadrática em uma certa vizinhança do mínimo procurado. Este fato justifica-se em que o MGC, como descrito no capítulo II, faz parte de uma classe de algoritmos de Otimização Irrestrita que operam baseados em informações sobre derivadas de segunda ordem. Isto fará com que o MGC, aplicado a funções não-quadráticas, evolua fortemente baseado na curvatura de sua aproximação quadrática.

Torna-se necessário então definirmos precisamente a classe de funções reais não-quadráticas para a qual apresentaremos reformulações (ou extensões) dos conceitos, algoritmos e resultados de convergência, de forma a podermos analisar a aplicação do MGC à minimização desta nova classe de funções.

Neste capítulo, assumiremos a função a ser minimizada f na classe das uniformemente convexas e duas vezes continuamente diferenciáveis no $\mathbb{R}^n$ e, por definição, satisfazendo:

$$m \|x\|^2 \leq x^t H(\bar{x}) x \leq M \|x\|^2 ,$$

$$(M \geq m > 0) \; \forall x, \bar{x} \in \mathbb{R}^n , \qquad (3.1)$$

onde (3.1) indica que os autovalores de $H(\bar{x})$ pertencem ao intervalo [m,M], implicando,equivalentemente, em que f satisfaça a condição de convexidade uniforme. Considere também x^* o mínimo global de f, e a aproximação quadrática de f para todo x em uma certa vizinhança de x^* sendo dada por F, tal que:

$$F(x) = f(x^*) + (x-x^*)^t g^* + \frac{1}{2}(x-x^*)^t H^*(x-x^*), \qquad (3.2)$$

onde $H^* = \nabla^2 f(x^*)$.

Sendo f uniformemente convexa, temos de imediato H^* def.pos. Logo, em uma certa vizinhança de x^*, o problema $\min f(x)$ é bem aproximado pelo de minimizar uma quadrática. Justifica-se então a expectativa de que os MGCs, convergentes em problemas quadráticos em um número limitado de iterações, sejam eficientes também para funções uniformemente convexas.

III.2 - A contribuição de Fletcher e Reeves:

O sucesso dos MGCs em funções quadráticas levou diversos pesquisadores, principalmente a partir da década de 60, a testá-los em funções não-quadráticas. Neste sentido, em 1964, FLETCHER e REEVES [2] formularam uma versão do MGC onde não se necessita explicitamente de derivadas de segunda ordem (em contraste com o Algoritmo 2) exigindo, porém, uma forma indireta de se realizar buscas lineares exatas. O algoritmo resultante, denominado Algoritmo de Fletcher-Reeves sem Reinicializações, o qual nomearemos a partir de agora Algoritmo 3, é apresentado abaixo:

Passo 0: Dados $f:\mathbb{R}^n \to \mathbb{R}$ e $x_1 \in \mathbb{R}^n$, faça $r_1 \leftarrow -f'(x_1)$ e $k \leftarrow 1$.

Passo 1: $d_k \leftarrow r_k$

Passo 2:
- 2.1: Se $\| r_k \|^2 = 0$, pare; $x^* = x_k$ é o mínimo desejado.
- 2.2: $x_{k+1} \leftarrow x_k + \lambda_k d_k$,

onde $\lambda_k = \min_{\bar{\lambda}>0} \bar{\lambda} \mid f(x_k + \bar{\lambda} d_k) < f(x_k + \lambda d_k), \forall \lambda > 0.$

2.3: $r_{k+1} \leftarrow -f'(x_{k+1})$

2.4: $\beta_k \leftarrow \| r_{k+1} \|^2 / \| r_k \|^2$

2.5: $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$

2.6: $k \leftarrow k + 1$
Vá para 2.1.

Assumindo aritmética exata, este algoritmo, quando aplicado a funções quadráticas def. pos., é idêntico ao Algoritmo 2, possuindo a propriedade de terminação quadrática.

Por outro lado, no caso de funções não-quadráticas, o cálculo do passo ótimo λ_k em 2.2, impossível de ser efetuado diretamente como no caso quadrático (ver Teorema 3), é feito através de uma busca linear (processo iterativo) onde se tenta alcançar a condição $d_k^t r_{k+1} = 0$, de forma similar ao caso quadrático. Aqui, as direções geradas correspondem à atual aproximação quadrática local da função e a taxa de convergência depende fortemente da sensibilidade do algoritmo à precisão da aproximação quadrática em cada iteração.

Na prática, porém, é em geral impossível efetuarmos buscas lineares exatas. São usados algoritmos iterativos (veremos alguns no capítulo IV) que fornecem, com acurácia desejada, uma aproximação $\bar{\lambda}_k$ para λ_k (onde $\lambda_k \mid d_k^t r_{k+1} = 0$). Logo, o processo de busca linear, sem ser a única, é a fase nos MGCs que mais gera erros que, acumulados a cada iteração, vão distorcendo os resultados de cálculos futuros nos MGCs e, consequentemente, as propriedades por eles assumidas. A solução adotada por Fletcher e Reeves para contornar este problema indesejado foi a reinicialização ("restart") do algoritmo a cada n iterações, for

necendo como nova direção inicial de busca $d_{kn+1} = r_{kn+1}$, k=1,2,... .O "restart" é então uma estratégia para se evitar que erros acumulados em iterações anteriores prejudiquem o bom comportamento dos MGCs quando estes tiverem alcançado uma região onde a função seja bem aproximada por uma quadrática.

O "restart" a cada n iterações, da forma acima descrita, é melhor entendido geometricamente, ao relembrarmos que os MGCs vão evoluir baseados na aproximação quadrática local da função a ser minimizada , e que n (ou até menos) iterações do MGC correspondem à minimização desta aproximação, obtendo x_{n+1}, centro do elipsóide F_1. Esquematicamente, teríamos:

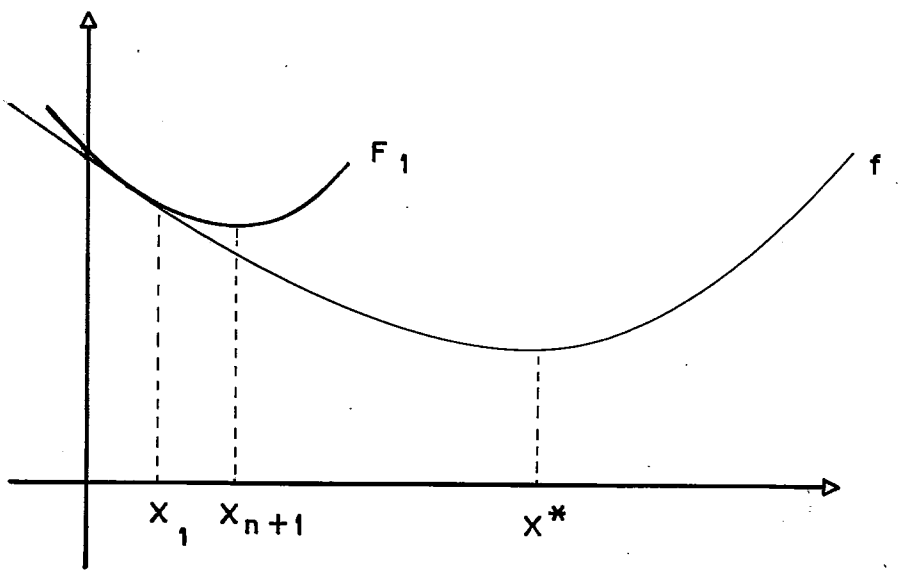


Fig. III.1

Alcançado x_{n+1} , que, embora seja o mínimo de F_1, está longe de poder ser apontado mínimo de f, é muito mais interessante para o MGC, a partir de agora, ignorar F_1 e construir uma nova aproximação de f, digamos F_2, centrada em x_{n+1}, para cujo mínimo - s.p.g., x_{2n+1} - o MGC deve evoluir. Esta construção equivale algebricamente a se reinicializar o algoritmo. Teríamos ago-

ra a seguinte situação:

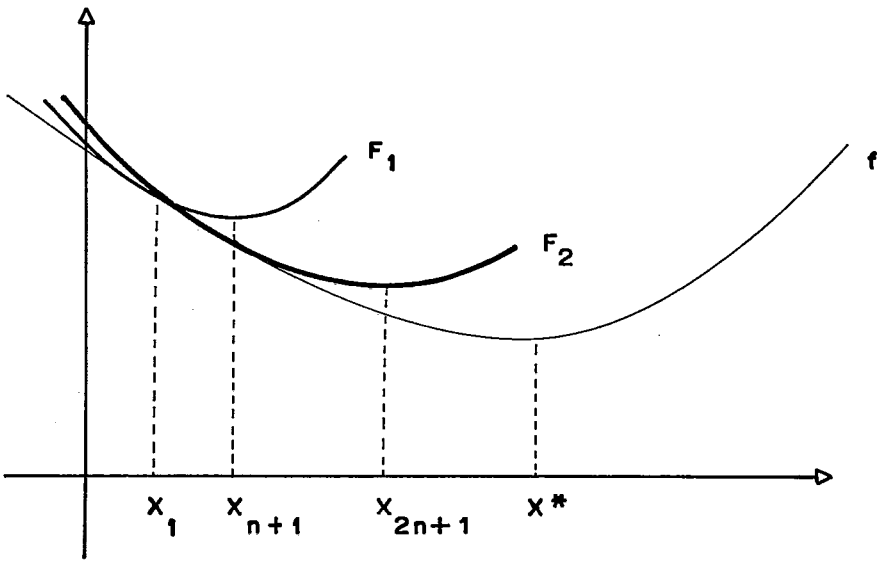


Fig. III.2

Assim, gerando sucessivas aproximações de f cada vez mais próximas de uma vizinhança de x*, o método converge em uma velocidade, em geral, bem maior para uma aproximação bastante satisfatória de x*, do que seria a do MGC sem "restart".

Fletcher e Reeves, em experiências com funções diversas, utilizaram este processo com ciclos de n+1 iterações em vez de n. Segundo eles [2], uma iteração a mais (por ciclo) foi adicionada "por analogia com o uso dos MGCs na resolução de sistemas lineares [1], onde foi visto que uma iteração adicional é benéfica para compensar a acumulação de erros de arredondamento gerados durante o ciclo de n iterações". Como observação, vale citar que o sub-problema da minimização unidimensional foi resolvido, de forma bastante satisfatória, pelo algoritmo de interpolação cúbica de Davidon, ao qual voltaremos a nos referir no capítulo IV.

III.3 - A contribuição de Polak e Ribière:

Em 1964, POLAK e RIBIÈRE [6] apresentaram uma contribuição significativa ao estudo da aplicação dos MGC a funções não-quadráticas (principalmente), onde propuseram uma fórmula alternativa para o coeficiente de conjugação β_k (à qual já nos referimos em II.4). Para facilitar a compreensão, re-escrevemos as fórmulas para β_k segundo Fletcher-Reeves, (2.30), e Polak-Ribière, (2.29).

Temos então:

$$\beta_k^{FR} = \frac{\| r_{k+1} \|^2}{\| r_k \|^2} \qquad \text{e} \qquad \beta_k^{PR} = \frac{r_{k+1}^t (r_{k+1} - r_k)}{\| r_k \|^2}$$

O termo adicional $r_{k+1}^t r_k$ em β_k^{PR} tem um significado imediato como minimizador do erro ocorrido quando a fórmula de Fletcher-Reeves considera os antigradientes gerados como exatamente ortogonais. Este erro é causado principalmente por uma fraca aderência da função com sua aproximação quadrática local tomada na k-ésima iteração. Porém, a discussão em torno de qual fórmula deva ser usada em que tipo(s) de problemas está em aberto, motivando uma série de estudos. Em particular, o trabalho de POWELL [27] deve ser considerado de suma importância, visto ter confrontado diretamente as duas versões de β_k em uma situação bastante viável em aplicações reais; sua conclusão para este tipo de situação, como veremos agora, é a de que se deve optar por β_k^{PR}.

A figura ao lado nos mostra a definição de $d_k (k > 1)$, o ângulo θ_k entre d_k e r_k e também a ortogonalidade entre d_{k-1} e r_k.

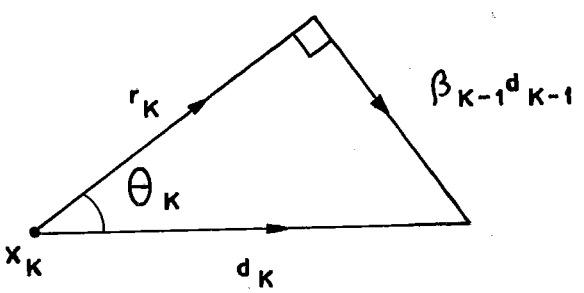


Fig. III.3

A partir da figura, temos:

$$\cos \theta_k = \frac{\| r_k \|}{\| d_k \|} \Rightarrow \| d_k \| = \| r_k \| \cdot \sec \theta_k \quad . \quad (3.3)$$

Por outro lado, se trocarmos k por k+1 na Figura III.3, poderemos escrever:

$$\tan \theta_{k+1} = \beta_k \cdot \| d_k \| / \| r_{k+1} \| \quad , \quad (3.4)$$

A partir de (3.3) e (3.4), concluimos:

$$\boxed{\tan \theta_{k+1} = \frac{\| r_k \|}{\| r_{k+1} \|} \cdot \beta_k \cdot \sec \theta_k} \quad , \quad (3.5)$$

o que nos mostra de forma clara a dependência do ângulo θ_{k+1} a ser obtido na (k+1)-ésima iteração com θ_k, obtido na k-ésima.

Tomemos agora o MGC, segundo Fletcher e Reeves. Substituindo β_k por β_k^{FR} em (3.5), temos:

$$\tan \theta_{k+1} = \frac{\| r_{k+1} \|}{\| r_k \|} \sec \theta_k > \frac{\| r_{k+1} \|}{\| r_k \|} \tan \theta_k \quad . \quad (3.6)$$

A situação proposta por Powell foi a seguinte: suponha que o MGC, com β_k^{FR}, tenha efetuado a k-ésima iteração, obtendo uma direção d_k tal que $\theta_k \cong \pi/2$ (localmente d_k não é uma boa direção de descida). Teremos, então, uma pequena variação pontual $\lambda_k \| d_k \|$ que, por sua vez, implicará em um valor muito pequeno para $\| r_{k+1} - r_k \|$. Consequentemente, o raio $\| r_{k+1} \| / \| r_k \|$ será muito próximo de 1. Isto fará com que θ_{k+1} também fique próximo de $\pi/2$, ocasionando um novo pequeno progresso na próxima iteração. Este mau desempenho do método pode continuar por vá-

rias iterações, degradando seriamente sua velocidade de convergência. É verdade que este problema, quando ocorre, é sempre resolvido, mais cedo ou mais tarde, por uma reinicialização do MGC na direção do antigradiente.

Por outro lado, ao substituirmos β_k por β_k^{PR} em (3.5), teremos:

$$\tan \theta_{k+1} \leq \frac{\| r_{k+1} - r_k \|}{\| r_k \|} \sec \theta_k \quad . \qquad (3.7)$$

Suponha o MGC com a fórmula β_k^{PR} e que a mesma situação ($\theta_k \cong \pi/2$) tenha ocorrido na k-ésima iteração. Como já vimos, isto implica em que $\| r_{k+1} \| / \| r_k \| \to 1$. Porém, agora teremos $\| r_{k+1} - r_k \| << \| r_k \|$, e, consequentemente $\tan \theta_{k+1} << \sec \theta_k$, que pode ser re-escrita na forma $\tan \theta_{k+1} << (\tan \theta_k)/\text{sen } \theta_k$. (Valendo sempre a hipótese de que $\text{sen } \theta_k \cong 1$). Isto nos leva à conclusão de que, em usando β_k^{PR}, o MGC se recupera rapidamente da má iteração k ao reaproximar a direção d_{k+1} de r_{k+1}, através da diminuição de θ_{k+1} em relação a θ_k.

Concluindo, o trabalho de POWELL [27] recomenda que, principalmente em funções não-quadráticas (onde este tipo de situação pode ocorrer mais facilmente), se dê preferência ao uso de β_k^{PR} em relação a β_k^{FR}.

III.4 - Resultados de Convergência:

Pretendemos apresentar aqui uma breve evolução histórica de resultados de convergência do MGC. Podemos iniciar citando o resultado da terminação quadrática obtido em HESTENES [1]

em 1952, válido para a versão Hestenes-Stiefel do MGC (Algoritmo 2 com β_k^{HS}) em funcionais quadráticos. Com o surgimento das versões Fletcher-Reeves (em 1964) e Polak-Ribière (em 1969) verificou-se a terminação quadrática para toda uma classe de MGC. Em 1972, CROWDER e WOLFE [11] no entanto mostraram que, se a primeira direção de descida tomada for diferente do antigradiente da função no ponto inicial, o MGC aplicado à funcionais quadráticos pode não convergir finitamente. Os melhores resultados por eles obtidos indicaram taxas de convergência apenas lineares. Este resultado é de grande importância no estudo dos MGCs em funções mais gerais pois indica claramente que as versões sem reinicializações ("restarts") do MGC aplicadas a funções (não-quadráticas) quaisquer podem não atingir sequer uma taxa superlinear. Por esta razão, focalizaremos, a partir de agora, apenas as versões do MGC com "restart".

Em 1972, COHEN [14] demonstrou que o MGC (Daniel, Fletcher-Reeves e Polak-Ribière) tem convergência quadrática a cada n iterações se a hessiana da função a ser minimizada é lipschitizianamente contínua, uniformemente definida positiva e se são usadas buscas lineares exatas. Definimos a seguir o MGC com "restart" considerado por Cohen no seu trabalho.

ALGORITMO 4

Passo 0: Escolha $r \geq n$, onde r determina de quantas em quantas iterações será efetuada uma reinicialização (r será o comprimento a ser usado do ciclo do MGC). Dado $x_1 \in \mathbb{R}^n$, faça $k \leftarrow 1$ e $d_1 \leftarrow r_1 \leftarrow -\nabla f(x_1)$.

Passo 1: Se $\| r_k \|^2 = 0$, pare. x_k é o ponto desejado.

Passo 2: Obtenha $\lambda_k \in \mathbb{R}_+$ tal que:

$$f(x_k + \lambda_k d_k) = \min_{\lambda>0} f(x_k + \lambda d_k)$$

Passo 3: $x_{k+1} \leftarrow x_k + \lambda_k d_k$

$$r_{k+1} \leftarrow - \nabla f(x_{k+1})$$

$$d_{k+1} \leftarrow r_{k+1} + \beta_k d_k \ ,$$

$$\text{onde } \beta_k \leftarrow \begin{cases} 0 \ , \text{ se } k = jr \ , \ \forall j \in \{1,2,3,\ldots\} \\ \text{uma das expressões} \\ (2.28), (2.29) \text{ ou } (2.30), \text{ se } k \neq jr \end{cases}$$

Passo 4: Faça $k \leftarrow k+1$ e volte ao Passo 1.

KLESSIG e POLAK [15], em 1972, foram os primeiros a provar que não eram necessárias buscas lineares exatas para se obter a taxa de convergência quadrática a cada n iterações. A única exigência imposta ao processo de busca linear foi a de que o ponto encontrado não fosse muito afastado do correspondente à busca linear exata, isto é:

$$\frac{| r_{k+1}^t d_k |}{\| r_{k+1} \| \, . \, \| d_k \|} \leq \min(\| r_k \| , s), \text{ onde } 0 < s < 1 \ .$$

Este tipo de estratégia visa manter sob controle o erro nas buscas lineares ao exigir que o cosseno entre r_{k+1} e d_k tenda a 0, quando $k \to \infty$.

Através de um procedimento semelhante, KAWAMURA e VOLTZ [18] em 1973 obtiveram as mesmas taxas obtidas por Klessig e Polak. As exigências feitas sobre as buscas lineares foram as seguintes:

- existência de um limite inferior positivo para o passo obtido;
- o passo obtido deve fazer com que o valor da função decresça;
- existência de um limite superior para o módulo da derivada direcional de f no final de cada busca linear.

Além disto, cabe citarmos que eles consideraram a minimização de funções de classe três vezes continuamente diferenciável.

Em 1974, McCORMICK e RITTER [20], usando as condições impostas à função-objetivo por Cohen, apresentaram uma nova forma de demonstração da taxa de convergência quadrática a cada n iterações, relaxando a exigência de buscas lineares exatas. Estes mesmos autores em [13] mostram uma abordagem comparativa das taxas de convergência do MGC e de alguns métodos Quasi-Newton. A conclusão deste trabalho é a de que um algoritmo, em geral, não deve ser escolhido entre outros apenas por apresentar uma maior taxa (velocidade) de convergência; outros aspectos como a quantidade de cálculos por iteração ("esforço computacional") e o espaço de memória requerido para a implementação do algoritmo em computadores devem ser também considerados.

Concluindo, podemos afirmar que é recente (início da década de 70) o grande avanço dos estudos teóricos sobre a convergência de métodos de otimização irrestrita aplicados a funções quaisquer. As pesquisas atuais têm sido dirigidas principalmente à consolidação das idéias existentes do que à construção de novas classes de métodos. Técnicas alternativas têm sido usadas e o desempenho global dos algoritmos é observado. Assim, tenta-se estabelecer propriedades que dêem origem a algoritmos mais estáveis e rapidamente convergentes.

CAPÍTULO IV

MINIMIZAÇÃO DE FUNÇÕES MAL-COMPORTADAS E IMPLEMENTAÇÃO COMPUTACIONAL DE ALGORITMOS DE GC PARA FUNÇÕES REAIS CONTINUAMENTE DIFERENCIÁVEIS QUAISQUER

IV.1 - Introdução:

A partir do sucesso do trabalho de FLETCHER e REEVES [2] em 1964, diversos estudos ([6], [8], [9], [15], [17], [18], [27], [29], [31], [35], [36] e [41], por exemplo) vêm sendo efetuados ao longo dos últimos 20 anos no sentido de propor contribuições à teoria e desenvolvimento de versões do MGC que sejam confiáveis e convergentes na minimização de funções reais continuamente diferenciáveis. Estes estudos foram motivados pelo desejo de se acelerar a típica convergência linear do Método do Gradiente sem que se precise trabalhar com a (calcular e inverter, ou aproximar a inversa da) Hessiana da função-objetivo a cada iteração, como é feito nos métodos Quasi-Newton e Newton. O Método do Gradiente, quando aplicado em funções não-quadráticas apresenta, em geral, um comportamento gradativamente sofrível. Perto de um ponto de mínimo, o gradiente normalmente aponta para uma região onde a função logo aumentará seu valor, razão pela qual o passo λ_k escolhido deve ser muito pequeno para garantir um decréscimo no valor da função. Tema principal deste trabalho, os aspectos envolvidos na aplicação do MGC a este tipo de problemas nos interessam fundamentalmente, devido a três características bem atraentes do MGC. São elas:

i) O MGC utiliza de forma explícita somente os valores da função objetivo e da sua primeira derivada em qualquer ponto da sequência $\{x_k\}$ gerada;

ii) do ponto de vista computacional, a memória necessária ao MGC é relativamente pequena se comparada à requerida pelos métodos Quasi-Newton e Newton . Já a partir de problemas de médio

porte ($n > 70$), começa a se tornar problemático, em geral, o armazenamento da aproximação da inversa da Hessiana (nos métodos Quasi-Newton) ou da própria inversa (no método de Newton) na memória disponível do computador. Por seu lado, o MGC só lida com alguns vetores, (em implementações mais sofisticadas, como veremos, tem-se no máximo 5 vetores com n posições cada: x_k, r_k, d_k, r_c e d_c) não requerendo espaço para o armazenamento de qualquer matriz. Essa economia de memória é particularmente interessante em implementações de algoritmos de GC em microcomputadores ou em computadores onde houver restrições quanto à memória disponível. Logo, sob o aspecto da memória, o MGC se constitui em uma boa alternativa aos métodos Quasi-Newton.

iii) as sequências $\{x_k\}$, $\{f_k\}$ e $\{\|g_k\|\}$ em geral convergem razoavelmente rápido e não são, comparativamente, muito mais lentas que as geradas por métodos Quasi-Newton.

Neste capítulo, após evidenciar alguns problemas envolvendo a convergência de algoritmos de Otimização Irrestrita em problemas mal-condicionados, pretendemos mostrar os principais resultados envolvendo a aplicabilidade do MGC em funções reais não-quadráticas continuamente diferenciáveis, bem como a implementação computacional destes algoritmos. Por último, cabe lembrar que, por considerarmos neste capítulo apenas como pré-requisito da função-objetivo a diferenciabilidade contínua, sempre estaremos nos referindo à convergência do MGC a pontos de mínimo locais.

IV.2- Discussão de alguns problemas relacionados com a minimização irrestrita de funções mal-comportadas:

Como já afirmamos em III.1, um algoritmo de otimização costuma, de alguma forma, utilizar propriedades da classe de problemas para a qual foi desenvolvido. No caso do MGC, somente para funções com comportamento quadrático na vizinhança de um mínimo podemos garantir a convergência do método para este ponto. Ao ampliarmos o conjunto de funções a ser abrangido pelo MGC,

em princípio não se garante a convergência para um ponto de mínimo; porém, como o método é de descida, a expectativa é a de que, pelo menos, o algoritmo correspondente atinja um ponto-de-sela da função-objetivo. E se, além disto, a função puder ser razoavelmente bem aproximada pelos três primeiros termos da sua série de Taylor (até a sua segunda derivada), espera-se [2] que a convergência de um método como o MGC, que aproxima o método de Newton, seja "bastante rápida" em uma vizinhança de um ponto de mínimo da função.

Adiada a discussão de problemas relacionados com a implementação computacional de algoritmos numéricos (discutidos em IV.3),temos ainda alguns efeitos indesejados causados na convergência de algoritmos de otimização irrestrita devidos a um mau comportamento da função-objetivo ("ill-behaved functions") caracterizado por um mau condicionamento da Hessiana da função em torno de um mínimo x^*. Esta característica pode prejudicar o desempenho de certas partes mais sensíveis destes algoritmos como, por exemplo, o processo de busca linear, a escolha da nova direção de descida e a validade dos testes de parada. Problemas mal-condicionados devem ser abordados por algoritmos contendo o maior número possível de salvaguardas que os previnam contra os efeitos dessa característica. Evidenciaremos a seguir alguns destes efeitos, sugerindo, sempre que possível, medidas que os contornem satisfatoriamente:

A importância da escolha do ponto inicial x_1 do algoritmo:

Em muitos casos, um algoritmo de Otimização Irrestrita não converge para uma solução quando a função não apresentar um bom comportamento na região onde se encontra o ponto inicial fornecido. Sempre que possível for, escolhemos, com o máximo de informações disponíveis sobre a função , um ponto de partida conveniente para o algoritmo.

A estimativa $\hat{x}$ alcançada pelo algoritmo está distante da solução ótima x^*:

Para evidenciar melhor este fenômeno em funções mal-comportadas, considere o problema:

$$\min f(x), \quad f : \mathbb{R}^n \to \mathbb{R} \quad e \quad f \in C^2. \tag{4.1}$$

Sendo x^* ponto de mínimo local estrito (ou forte) de f, temos necessariamente $g(x^*) = 0$ e $\nabla^2 f(x^*) = H(x^*)$ definida positiva. Além disto, denominando:

- $\hat{x}$ a aproximação de x^* obtida por algum algoritmo de otim. irrestrita, e
- ε ($\varepsilon>0$) a distância máxima permitida entre $f(x^*)$ e o valor de f em uma aproximação qualquer alcançada por este algoritmo, temos que, se:

$$|f(x^*)-f(\hat{x})| \leq \varepsilon \tag{4.2}$$

então teoricamente nenhuma outra aproximação de x^* deverá ser melhor que $\hat{x}$, visto $f(\hat{x})$ estar suficientemente próximo de $f(x^*)$. Fixando-se um valor para ε, definimos qualquer ponto satisfazendo (4.2) como uma aproximação satisfatória da solução x^* do problema (4.1). O que mostraremos a seguir é que, mesmo que $\hat{x}$ obedeça a (4.2), não é sempre verdade que o afastamento $\|x^*-\hat{x}\|$ seja próximo de zero!

Tomando a série de Taylor de f em uma vizinhança de x^*, podemos escrever (para $\mu \in \mathbb{R}^+$ e $d \in \mathbb{R}^n$):

$$f(\hat{x}) = f(x^*+\mu d) = f(x^*) + \underbrace{\mu d^t g(x^*)}_{0} + \frac{1}{2}\mu^2 d^t H(x^*) d + 0(\mu^3) \Rightarrow$$

$$\Rightarrow f(\hat{x}) = f(x^*) + \frac{1}{2}\mu^2 d^t H(x^*) d + 0(\mu^3) \quad , \tag{4.3}$$

onde, s.p.g., admitimos $\|d\| = 1$ e, consequentemente, $\mu = \|x^*-\hat{x}\|$. Por estarmos considerando $\hat{x}$ em uma pequena vizinhança de x^*, podemos escolher um valor tal para $\hat{x}$ que, a partir de (4.2) e (4.3), nos permita escrever:

$$\frac{1}{2}\mu^2 |d^t H(x^*) d| \cong |f(x^*) - f(\hat{x})| = \varepsilon \Rightarrow$$

$$\Rightarrow \mu^2 \cong \frac{2\varepsilon}{d^t H(x^*) d} \quad , \tag{4.4}$$

que, pela definição de μ nos leva finalmente a:

$$\boxed{\|x^*-\hat{x}\|^2 \cong \frac{2\varepsilon}{d^t H(x^*) d}} \qquad (4.5)$$

À luz de (4.5), procuraremos compreender como o afastamento $\|x^*-\hat{x}\|$ pode ser influenciado por problemas onde $H(x^*)$ for mal-condicionada(,ou até mesmo bem-condicionada, porém com autovalores muito próximos de zero). Para isto, explicitaremos a relação entre $\|x^*-\hat{x}\|$ e os autovalores λ_i (com respectivos autovetores v_i) de $H(x^*)$. Supondo os escalares $a_i (i=1,\ldots,n)$, nem todos nulos, tais que $d = \sum_i a_i v_i$, podemos re-escrever (4.5) da seguinte forma:

$$\|x^*-\hat{x}\|^2 \cong \frac{2\varepsilon}{(\sum_i a_i v_i)^t \sum_i a_i H(x^*) v_i} = \frac{2\varepsilon}{(\sum_i a_i v_i)^t \sum_i a_i \lambda_i v_i} =$$

$$= \frac{2\varepsilon}{\sum_{j=1}^{n} \sum_{i=1}^{n} a_i a_j \lambda_j v_i v_j}$$

A título de ilustração, vejamos duas situações distintas:

i) se d puder ser escrita como combinação linear dos autovetores de $H(x^*)$ associados aos seus maiores autovalores, então $\|x^*-\hat{x}\|$ poderá ser razoavelmente pequeno; são os casos onde estes autovalores sejam pelo menos $O(1)$.

ii) se, porém, d for combinação linear dos autovetores de $H(x^*)$ associados aos seus menores autovalores, então $\|x^*-\hat{x}\|$ poderá ser razoavelmente grande; são os casos onde estes autovalores sejam no máximo $O(\varepsilon)$.

CONCLUSÃO 1

O erro $\|x^*-\hat{x}\|$ em uma solução satisfatória $\hat{x}$ em alguns problemas pode ser relativamente grande em certas direções. Logo, deve-se evitar a

utilização isolada da condição $\|x^*-\hat{x}\| \cong 0$ como critério de parada em algoritmos de otimização irrestrita, principalmente em problemas com as características antes citadas.

A norma do gradiente na estimativa $\hat{x}$ alcançada pelo algoritmo não é necessariamente próxima de zero:

Como é usual usar-se a norma $\|g(x)\|$ como indicador em testes de parada de algoritmos de otimização irrestrita, devemos evidenciar aqui como um problema mal-condicionado (no sentido já exposto) ou mesmo um bem-condicionado com auto-valores bem maiores que zero pode apresentar $\|g(\hat{x})\| >> 0$. Expandindo g na sua série de Taylor em uma vizinhança de x^*, podemos escrever:

$$g(\hat{x}) = g(x^*+\mu d) = \underbrace{g(x^*)}_{\|\ 0} + \mu H(x^*)d + \mathcal{O}(\mu^2) \quad \Rightarrow$$

$$\Rightarrow g(\hat{x}) \cong \mu H(x^*)d \Rightarrow \|g(\hat{x})\| \cong \mu^2 \|H(x^*)d\|^2,$$

e, usando (4.4), temos:

$$\boxed{\|g(\hat{x})\|^2 \cong 2\varepsilon \frac{\|H(x^*)d\|^2}{d^t H(x^*)d}} \tag{4.6}$$

Tomando a_i, λ_i e v_i $(i=1,\ldots,n)$ como antes definidos, podemos re-escrever (4.6) como:

$$\|g(\hat{x})\|^2 \cong 2\varepsilon \frac{\|\sum_i a_i H(x^*)v_i\|^2}{(\sum_i a_i v_i)^t (\sum_i a_i H(x^*)v_i)} =$$

$$= 2\varepsilon \frac{(\sum_i a_i \lambda_i v_i)^t (\sum_i a_i \lambda_i v_i)}{(\sum_i a_i v_i)^t (\sum_i a_i \lambda_i v_i)} =$$

$$= 2\varepsilon \frac{\sum_{i=1}^{n} a_i \lambda_i v_i \sum_{j=1}^{n} a_j \lambda_j v_j}{\sum_{i=1}^{n} a_i v_i \sum_{j=1}^{n} a_j \lambda_j v_j} \tag{4.7}$$

Apenas para evidenciar os limites desta dependência, tomemos o caso extremo onde a direção $\underline{d}$ é múltipla do autovetor de $H(x^*)$ associado ao seu maior autovalor λ_{max}. Para este caso teremos $\|g(\hat{x})\|^2 = 0(\varepsilon\ \lambda_{max})$, o que nos leva a concluir que, em problemas com $\lambda_{max} \geq 0(100)$, podemos ter até $\|g(\hat{x})\| = 0(10\sqrt{\varepsilon})$ ao invés do esperado $\|g(\hat{x})\| \cong 0(\sqrt{\varepsilon})$, segundo (4.5). Se por um lado este inconveniente pode ser contornado em problemas bem-comportados através de técnicas de "scaling" na função objetivo (ver [40]), não há como evitá-lo em problemas mal-condicionados, onde $0<\lambda_{min} <<< \lambda_{max}$.

CONCLUSÃO 2

A medida $\|g(\hat{x})\|$ em uma solução satisfatória $\hat{x}$ para alguns problemas pode ser relativamente grande em certas direções. Logo, cuidados devem ser tomados na utilização isolada da condição $\|g(\hat{x})\| \cong 0$ como critério de parada em algoritmos de otimização irrestrita, principalmente em problemas com as características antes citadas.

Critérios para a escolha de um conjunto suficientemente robusto de testes de parada em problemas mal-condicionados e/ou com características particulares:

Principalmente em problemas assim, deve-se dedicar a maior atenção possível à análise e definição de critérios confiáveis e robustos a serem obedecidos nas soluções alcançadas por algoritmos de otimização irrestrita, a serem usados tanto com aritmética exata como com as de ponto-flutuante em computadores. Discutiremos estes critérios com mais profundidade em IV.4 dentro de um enfoque computacional, após apresentarmos algumas considerações sobre o uso de aritméticas de ponto-flutuante em computadores, em IV.3.

IV.3- Sobre computadores e aritméticas de ponto-flutuante:

Quando desejamos executar um algoritmo em um computador, construímos um programa que nada mais é do que uma versão desse algoritmo que leva em consideração os recursos finitos do computador. Este fato se torna de suma importância na implementação de algoritmos numéricos definidos sobre valores reais (ou complexos), já que o computador, por ter uma quantidade finita de memória para o armazenamento de informações, representa certos valores reais (na concepção matemática de números reais) internamente por valores aproximados. Na prática, o computador não trabalha com números reais e sim com um conjunto de números de ponto-flutuante. Este conjunto, como se sabe, é discreto e contém apenas uma "pequena parte" (finita) do conjunto dos racionais. Isto pode fazer com que certos trechos de um programa que usem variáveis do tipo REAL (ponto-flutuante) forneçam, para os mesmos dados, diferentes resultados em diferentes computadores, devido à falta de padronização das técnicas para tratar arredondamentos (e truncamentos) bem como dos algoritmos para avaliação de funções definidas por séries trigonométricas e exponenciais, por exemplo . Algoritmos numéricos iterativos, em geral, estão envolvidos significativamente com acumulação (ou propagação) de erros gerados em cálculos intermediários que, por sua vez, podem levá-los, em certas situações, a uma não-convergência teoricamente inesperada.

CONCLUSÃO 3

Do ponto de vista matemático, métodos de minimização irrestrita são construídos para resolver o problema:

$$\min f(x),\ f:\mathbb{R}^n \to \mathbb{R},\ f(x) \in \mathbb{R},\ x \in \mathbb{R}^n,$$

enquanto que, do ponto de vista computacional, quando implementamos um algoritmo para resolver o problema acima em um computador, estamos, na verdade, resolvendo o seguinte problema:
min F(x), onde F é uma função $\mathbb{F}^n \to \mathbb{F}$, $F(X) \in \mathbb{F}$ e $X \in \mathbb{F}^n$,

onde $\mathbb{F}$ é o conjunto de números de ponto-flutuante existente no computador usado.

IV.4- Sobre testes de parada:

A importância da presença de testes de parada em algoritmos de otimização (irrestrita ou não) deve-se basicamente a duas causas principais:

i) por questões lógicas, deve haver um mecanismo que decida se um ponto alcançado pelo algoritmo é (ou não) uma aproximação satisfatória para a solução do problema; e,

ii) mesmo não alcançando uma solução aceitável, testes de parada são úteis para indicar:

   a. se deve-se ou não interromper o algoritmo caso ocorra, durante algumas iterações, uma evolução muito lenta onde praticamente não haja progresso; neste caso, pode-se interromper o algoritmo, evitando que uma boa quantidade (às vezes bem grande) de cálculos inúteis seja efetuada;

   b. a ocorrência de ciclagem do algoritmo, por problemas diversos, impedindo qualquer progresso do mesmo;

   c. que não existe uma aproximação aceitável.

Logo, os principais objetivos de testes de parada devem fundamentalmente residir em garantir realmente a obtenção de uma aproximação satisfatória e em evitar cálculos desnecessários.

Segundo GILL e MURRAY [40], a decisão sobre se um ponto $\hat{x}$ é ou não uma aproximação satisfatória para x^* consiste em duas ações simultâneas:

- verificar se $\hat{x}$ satisfaz "razoavelmente" as condições suficientes de otimalidade, e
- confirmar se $\{x_k\}$ aparenta apresentar convergência.

Antes de indicar alguns testes, cabe lembrarmos que nenhum conjunto de testes de parada conhecido é totalmente robusto (ou confiável) para todas as funções (e para todos os algoritmos)). Um determinado teste pode ser satisfeito por uma aproximação satisfatória de uma função e pode não ser por uma aproximação

satisfatória de outra função, computadas pelo mesmo algoritmo (ver IV.2).

Da análise feita em IV.2, podemos concluir que uma aproximação satisfatória $\hat{x}$ para funções localmente muito bem-comportadas (i.é, com a Hessiana possuindo todos os seus autovalores $O(1)$, numa vizinhança de x^* contendo $\hat{x}$), deve obedecer simultaneamente a:

i) $f(\hat{x}) - f(x^*) \leq \varepsilon$,

ii) $\|\hat{x}-x^*\| \cong O(\sqrt{\varepsilon})$,

iii) $\|g(\hat{x})\| \cong O(\sqrt{\varepsilon})$.

Porém, o desconhecimento dos valores (na maioria dos casos) de x^* e $f(x^*)$ nos leva naturalmente a um conjunto de testes que, pelo menos, mantenha uma vigilância constante sobre a taxa de convergência, bem como sobre as próprias grandezas, das sequências $\{x_k\}$,$\{f_k\}$ e $\{\|g_k\|\}$. Neste sentido, escolhemos o seguinte conjunto mínimo (com pequenas variações) de testes a serem usados para indicar se um algoritmo de otimização irrestrita deve ser interrompido:

T1. $f_k - f_{k+1} \leq \varepsilon$

T2. $\|x_k - x_{k+1}\| \leq \sqrt{\varepsilon}$

T3. $\|g_{k+1}\| \leq \sqrt{\varepsilon}$

Enquanto os testes T1 e T2 confirmam se $\{x_k\}$e, consequentemente, $\{f_k\}$ aparentam convergir, o teste T3 obriga o atendimento à condição necessária de otimalidade $\|g(x^*)\|=0$.Deve-se ter cuidado em casos onde $|f(x^*)| >> 0$ ou $\|x^*\| >> 0$, onde então é conveniente substituir-se os testes T1 e T2 (erro absoluto) por outros, envolvendo erros relativos, como por exemplo:

T1a. $(f_k-f_{k+1})/|f_{k+1}| \leq \varepsilon$

T2a. $\|x_k-x_{k+1}\|/\|x_{k+1}\| \leq \sqrt{\varepsilon}$

Os testes T1 a T3 devem ser usados simultaneamente; caso contrário, podemos ter avaliações individuais sofríveis. Vejamos os possíveis inconvenientes de se usar isoladamente:

i) T1- Neste caso, se o algoritmo atingir uma região onde a função apresente uma "planície" (Fig. IV.1), pode ser interrompido precocemente;

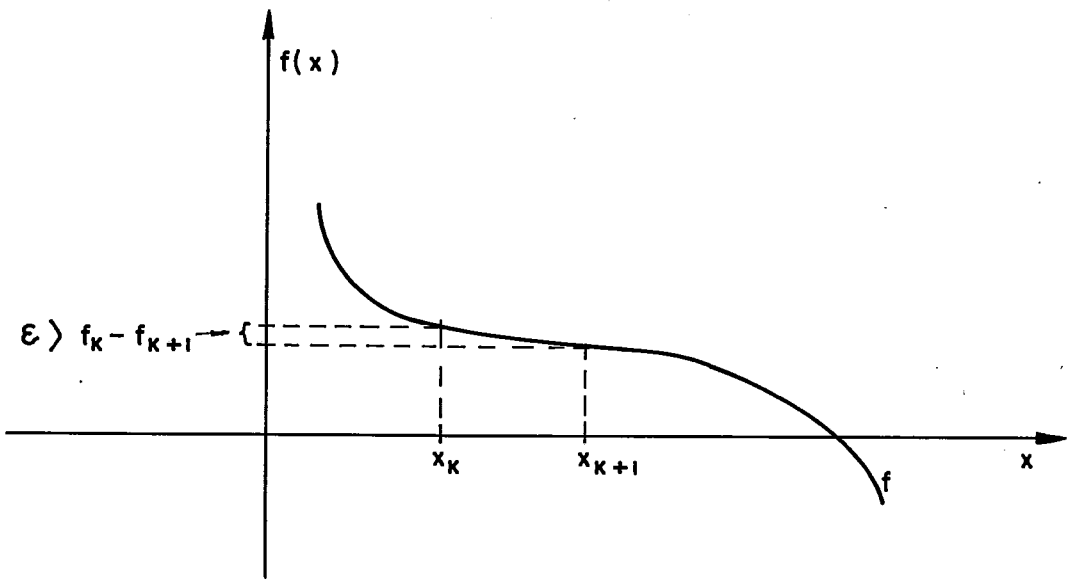


Fig. IV.1

ii) T2- Neste caso, se o algoritmo atingir uma região onde a função apresente um "declive muito íngreme", (Fig.IV.2) pode ser interrompido precocemente;

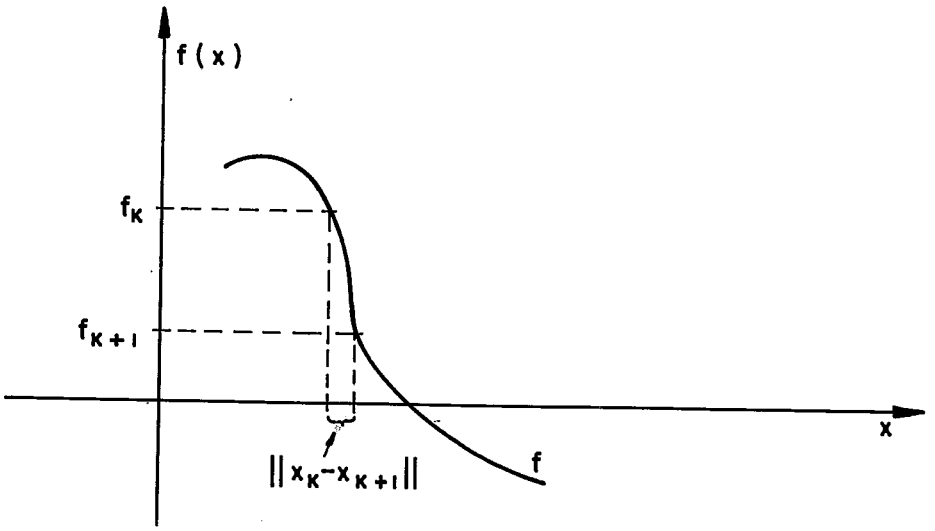


Fig.IV.2

iii) T3- Neste caso, se o algoritmo atingir uma vizinhança de um ponto de mínimo do tipo "planície", pode tentar seguir adiante em busca de uma aproximação melhor, insensível ao fato de que não será mais possível obter reduções significantes no valor da função (veja Fig. IV.3.a). O uso isolado de T3 pode causar interrupção precoce no algoritmo nas seguintes situações, por exemplo:

- em um ponto-de-sela, embora esteja ocorrendo uma variação pontual $\|x_k - x_{k+1}\|$ significativa, e,
- em pontos como os ilustrados na Fig.IV.3.b.

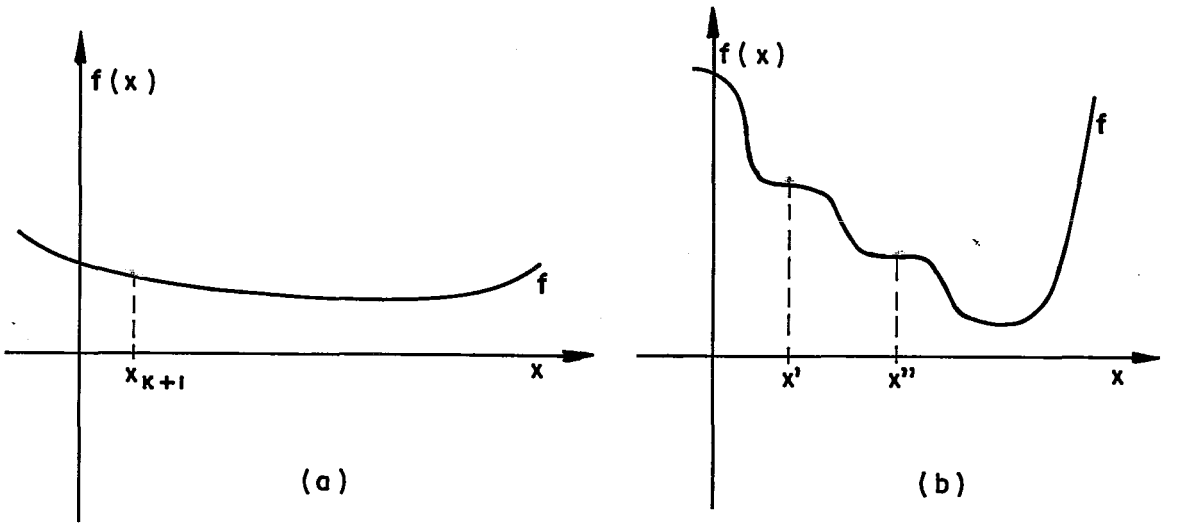


Fig. IV.3

Por razões já discutidas no início deste item, é conveniente incluirmos um quarto critério ao nosso conjunto, a saber:

T4. (número de iterações efetuadas) > (número máximo de iterações a priori fixado).

Finalizando, devemos lembrar que o conjunto proposto deve ser visualizado como um ponto de partida, podendo ser modificado convenientemente no que diz respeito à natureza e a quantidade de testes a serem considerados. Muitas propostas alternativas vêm sendo obtidas e aperfeiçoadas a partir de experiências empíricas (ver por exemplo, [40]), porém não podemos ainda garantir um conjunto que funcione muito bem em todos os casos.

IV.5- A questão da reinicialização, a proposta de Beale e os critérios de Powell:

Conforme já discutido em III.2, a reinicialização do MGC a cada n iterações fundamenta-se na sua propriedade de terminação quadrática, quando aplicado em funções quadráticas, e no fato de que um ciclo de n iterações do MGC corresponde diretamente a uma iteração do Método de Newton. Para funções não-quadráticas, o conceito de "direções conjugadas" deve ser redefinido, isto é, devemos saber em relação a que matriz as direções devem ser conjugadas. Como sabemos que o MGC, aplicado a funções não quadráticas, evoluirá sempre baseado na sua aproximação quadrática local, a resposta natural à nossa pergunta está na Hessiana de f no ponto de mínimo $\bar{x}$ desta aproximação local (que pode ser associado, s.p.g., ao ponto x_{n+1} da figura III.1).

Revista a conjugação, passemos agora à questão da reinicialização do MGC. A forma tradicional de "restart" consiste em se escolher a próxima direção de descida d_{k+1} como:

$$d_{k+1} \leftarrow \begin{cases} r_{k+1} + \beta_k d_k, \text{ se } k \neq jn \ (j=0,1,2,\ldots) \\ r_{k+1}, \text{ se } k=jn \end{cases}$$

Como já vimos em III.2, o uso desta estratégia representa uma melhoria significativa no desempenho do MGC; porém ela tem um ponto fraco, qual seja, o de desprezarmos a partir de x_{k+1}, todas as informações de segunda ordem (curvatura) acumuladas pelo MGC até atingir x_{k+1}. Para ter uma idéia das conseqüências deste "desprezo", POWELL, em [27], apresentou uma função não-quadrática com n=3 onde o MGC foi usado sem reinicialização e com a tradicional reinicialização a cada 1,2,3,4 e 5 iterações. Os resultados por ele obtidos, mostraram claramente que a redução local do valor da função quando se reinicializa com o antigradiente é geralmente menor da respectivamente obtida na mesma iteração, se não se tivesse reinicializado o MGC.

A nossa conclusão é a de que, embora exista um consenso geral de que reinicializações periódicas são bastante úteis na prática, devemos estabelecer versões do MGC que não tomem somente o antigradiente como direção de descida em iterações de

reinicialização do método. Neste sentido, BEALE [17], em 1972, propôs que o MGC fosse usado com

$d_1 = r_1$ e $d_{k+1} = r_{k+1} + \beta_k^{HS} d_k$ $(0 < k < n+1)$,

isto é: deve-se usar a última expressão para se obter também a direção d_{n+1} correspondente à primeira reinicialização do MGC; toma-se provisoriamente $c \leftarrow n+1$ (c representa a iteração onde ocorreu a última reinicialização) e, a partir daí, o cálculo das direções, neste segundo ciclo de n iterações, considera a influência da direção d_c usada na reinicialização mais recente da segunda forma: para o MGC (s.p.g., o algoritmo 4 - ver III.4 - com r=n) na k-ésima iteração $(k > c)$ teremos

$$d_{k+1} = r_{k+1} + \beta_k d_k + \gamma_{k,c} d_c, \tag{4.8}$$

onde :

$$\gamma_{k,c} = - \frac{r_{k+1}^t (r_c - r_{c-1})}{d_c^t (r_c - r_{c-1})} . \tag{4.9}$$

A expressão (4.9), proposta em [17], se originou do seguinte problema: "Se d_c é uma direção qualquer de descida usada em uma reinicialização anterior do MGC, se f é quadrática e se d_{k+1} $(k > c)$ deve resultar de uma combinação linear de d_c e dos antigradientes $r_{c+1}, r_{c+2}, \ldots, r_{k+1}$, então que combinação linear manterá as direções $d_c, c_{c+1} \ldots, d_{k+1}$ mutuamente conjugadas?" Beale demonstrou que este problema é resolvido pela expressão (4.9) para $\gamma_{k,c}$ e pela expressão de Hestenes-Stiefel (2.28) para β_k. A idéia da demonstração apresentada por Beale em [17] reside em que, se a direção usada na última reinicialização foi outra que não o antigradiente, então o uso de (4.8) sem a última parcela envolvendo $\gamma_{k,c}$ não garante que as novas direções $d_{c+1}, d_{c+2}, \ldots$ sejam conjugadas a d_c.

Esta técnica foi inicialmente experimentada **in natura** em funções não quadráticas (ver comentário em [31]) e causou divergência do MGC, através da geração de direções que não eram de descida, mesmo com o uso de buscas lineares exatas. Em 1977, POWELL [27] apresentou dois critérios a serem usados simultaneamente para garantir que a direção de reinicialização gerada

(diferente de r_{k+1}) fosse realmente de descida e propôs uma nova versão do MGC.

O primeiro critério de Powell se fundamenta no fato de que, quando a técnica de Beale é usada com aritmética exata, pode-se detectar automaticamente a necessidade de uma nova reinicialização, já que, com o uso de (4.8), sabe-se [17] que os antigradientes r_i, (i=c+1,c+2,...) são mutuamente ortogonais. Neste sentido, Powell sugere que um instante adequado para uma nova reinicialização é tal que:

$$\frac{r_k^t r_{k+1}}{\|r_{k+1}\|^2} > > 0 \qquad (4.10)$$

Powell lembra que, se a técnica de Beale for usada com o critério (4.10), este é mais do que um mecanismo para detectar e evitar uma convergência a um ponto errado, pois (4.10) se baseia no controle, a níveis toleráveis, da manutenção da ortogonalidade dos antigradientes, propriedade fundamental do MGC quando aplicado em funções quadráticas. Logo, podemos concluir que (4.10) é de grande valia no desempenho do MGC na aproximação quadrática local de funções não-quadráticas*.

Na prática, deve-se definir uma grandeza, digamos ε_1, a ser utilizada em (4.10). Em vista de que ε_1 deve indicar se a ortogonalidade entre r_k e r_{k+1} foi "significativamente deteriorada ou não", Powell comenta em [27] que valores de ε_1 tomados no intervalo (0.1,0.9) foram considerados empiricamente satisfatórios, sugerindo, em particular, 0.2. Logo, o primeiro critério de Powell pode ser assim escrito:

$$\boxed{\text{Se } \frac{|r_k^t r_{k+1}|}{\|r_{k+1}\|^2} > 0.2 \text{ então } \underline{\text{reinicialize o MGC.}}} \qquad (4.11)$$

Uma breve análise do teste acima é capaz de nos mostrar a dinâmica do seu funcionamento. Observe que, quanto maior

* Por outro lado, o uso de (4.10) com aritmética exata em funções quadráticas não interfere de forma alguma no desempenho do MGC, já que, neste caso, $r_{k+1} r_k = 0, (\forall\, k > c)$.

for a redução na ordem de grandeza entre $\|r_k\|$ e $\|r_{k+1}\|$ (indicando que localmente o MGC evoluiu rápido, provavelmente pela boa aderência entre a função e sua aproximação quadrática local),mais o teste será restrito, obrigando fortemente o processo de descida a manter satisfeita a propriedade fundamental da ortogonalidade dos antigradientes. Por outro lado, se as ordens de grandeza de $\|r_{k+1}\|$ e $\|r_k\|$ forem semelhantes, o teste torna-se mais relaxado, fazendo com que o MGC, em uma região de uma provável má aproximação quadrática local da função, não se preocupe tanto em respeitar fortemente uma condição obrigatória para funções quadráticas. Este teste, segundo POWELL [27], adicionalmente evita que $\{\|r_k\|\}$ tenda a um limite não-nulo. Powell lembra que apesar de o primeiro critério agir como um mecanismo de vigilância sobre a "quase-ortogonalidade" dos antigradientes gerados, ele ainda não garante que a direção de reinicialização de Beale seja de descida. Para auxiliar o primeiro critério, Powell, ainda em [27], sugeriu seu segundo critério, que visa manter sob controle o ângulo entre d_{k+1} e r_{k+1}, da seguinte forma:

> Se d_{k+1} não satisfizer a
> $0.8\|r_{k+1}\|^2 \leq d_{k+1}{}^t r_{k+1} \leq 1.2\|r_{k+1}\|^2$, (4.12)
> então reinicialize o MGC.

Para uma direção que satisfaça a este critério não só está garantido que é de descida, mas principalmente que é uma direção razoável de descida ("sufficiently downhill"). Novamente, uma análise pragmática baseada na comparação entre as ordens de grandeza de $\|d_{k+1}\|$ e $\|r_{k+1}\|$ nos mostra que, se $O(\|d_{k+1}\|) \leq O(\|r_{k+1}\|)$,então o teste (4.12) torna-se cada vez mais restritivo, sendo respeitado apenas para direções muito próximas do antigradiente r_{k+1}. Este mecanismo pode ser melhor entendido se considerarmos que, sendo razoavelmente grande o valor da norma do antigradiente, este estará voltado para uma direção tal que a redução local do valor da função, além de ser a máxima possível, deverá ser significativamente grande. É então bem coerente considerarmos como direções "suficientemente de descida" apenas as que se aproximem bastante do antigradiente. Por outro lado, se (e somente se) $O(\|d_{k+1}\|) > O(\|r_{k+1}\|)$, o teste (4.12) define um cone de direções gradativamente mais perto de ser ortogonal a r_{k+1}. Essa característica de

(4.12) se justifica exatamente pelo fato de um valor pequeno de $\|r_{k+1}\|$ em relação a $\|d_{k+1}\|$ indicar que, heuristicamente, faz sentido ignorar a direção r_{k+1} em prol de uma redução funcional maior em qualquer direção do cone.

Em relação a testes, SHANNO em [31] e BOLAND e KOWALIK [36] efetuaram algumas comparações usando a versão tradicional do MGC (Algoritmo 4 com r=n e a fórmula (2.29) de Polak-Ribière) e acrescida com a técnica de Beale e os critérios de Powell . Shanno, em particular, mostrou um terceiro algoritmo correspondente à versão tradicional descrita acrescida somente da técnica de Beale. Suas conclusões afirmam que "a técnica de Beale, na maioria dos casos melhora a velocidade de algoritmos de gradientes conjugados, e que a inclusão dos critérios de Powell é realmente útil em qualquer caso" [31]. Por sua vez, Boland e Kowalik apresentaram conclusões semelhantes; com o uso de buscas lineares exatas, conseguiram, em algumas comparações, uma economia em torno de 30% no total de iterações, a favor da versão com a estratégia Beale-Powell.

Por último, devemos comentar que a inclusão da técnica de Beale necessita (a princípio) 7 vetores, a saber:

$$x_{k+1}\ ,\ x_k\ ,\ r_{k+1},\ r_k\ ,\ d_k\ ,\ d_c \text{ e } y_c = r_{c+1} - r_c,$$

porém, na prática é possível economizar-se pelo menos o espaço para 2 vetores: x_{k+1} e r_{k+1}. Estes vetores podem ser armazenados respectivamente, sobre x_k e r_k, já que estes últimos não serão mais usados após o cálculo de x_{k+1} e r_{k+1}. Logo, temos um total de 5 vetores contra os 3 (x_{k+1}, r_{k+1}, d_k) usados fisicamente pelas versões tradicionais, o que não representa em geral um aumento significativo na memória requerida; no entanto, quando o espaço for por demais crítico, cabe ponderarmos sobre o uso ou não da técnica de Beale.

IV.6 - Sobre buscas lineares:

IV.6.1 - Introdução ao problema:

Desejamos aqui, além de introduzir suscintamente conceitos básicos relacionados com buscas lineares, apresentar alguns modernos algoritmos que são finitos e suficientemente robustos (no sentido de serem capazes de atingir uma aproximação razoável a partir de qualquer estimativa inicial do passo ótimo) para realizar uma busca linear, frequentemente usados em algoritmos de minimização irrestrita. Dados x_k, $d_k \in \mathbb{R}^n$, $g_k = \nabla f(x_k)$ e d_k satisfazendo $d_k^t g_k < 0$ (d_k é direção de descida), uma minimização unidimensional na direção d_k a partir do ponto x_k é representada pelo seguinte problema:

$$\begin{aligned} &\text{determinar } \lambda_k \in \mathbb{R}_+ \text{ tal que} \\ &f(x_k + \lambda_k d_k) = \min_{\lambda>0} f(x_k + \lambda d_k) \end{aligned} \tag{4.13}$$

Cronologicamente, os primeiros algoritmos usados (décadas de 50 e 60) na resolução do problema (4.13) se dividiam em duas principais famílias, a saber:

i) métodos que não usam informações sobre as derivadas de f: aqui se incluem o método de Fibonacci, o da secante, o da dicotomia e o da secção áurea (ver descrições destes métodos, por exemplo , em [30] ou [33]);

ii) métodos que usam derivadas, como o método da bissecção, o de Newton e alguns métodos de aproximações polinomiais(com destaque para o método de interpolação cúbica de Davidon[2]).

Tais métodos, criados para a determinação de raízes de funções, são usados no nosso caso para calcular,iterativamente, pontos de derivada direcional nula, isto é, definindo-se as relações a seguir (para a k-ésima iteração de algum método de otimização):

$$h(\lambda) = f(x_k + \lambda d_k) \; ; \quad \lambda \geq 0 \tag{4.14}$$

$$h'(\lambda) = \langle g(x_k + \lambda d_k), d_k \rangle \quad , \tag{4.15}$$

e, consequentemente,

$$h'(0) = g_k^t d_k < 0 \tag{4.16}$$

Os algoritmos acima citados (tanto em (i) como em (ii)) são denominados métodos de busca linear <u>exata</u> e se caracterizam por procurarem determinar um valor λ_k para λ (ou o menor valor de λ_k) tal que:

$$h'(\lambda_k) = 0 \qquad (4.17)$$

Isto fará com que o novo ponto obtido $x_{k+1}=x_k + \lambda_k d_k$ cause o máximo decréscimo possível de f na direção d_k. Considerando f uma função convexa, teríamos geometricamente a seguinte situação final para um processo de busca linear <u>exata</u>:

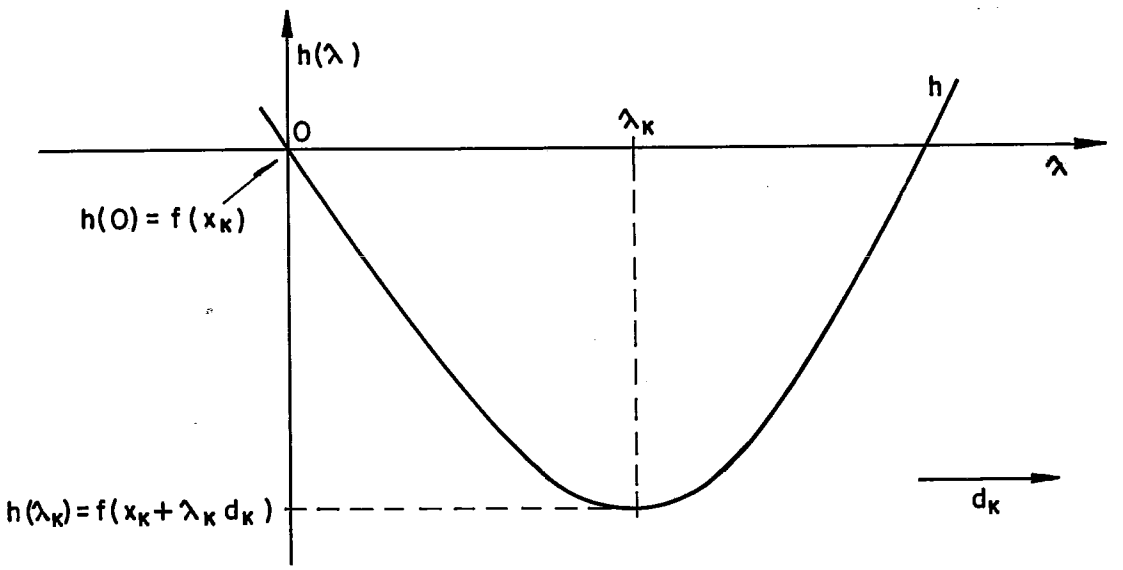


Fig. IV.4

Algoritmos de busca linear exata apresentam uma característica bastante inconveniente na prática: possuem em geral convergência assintótica; esta propriedade pode tornar esses algoritmos extremamente lentos em certos casos. E aí existem dois aspectos intrinsecamente relacionados:

i) <u>tempo</u> (ou esforço computacional) - não podemos nos esquecer que um algoritmo de otimização irrestrita deve resolver, a cada iteração, um sub-problema de busca linear; logo, devemos usar um método rápido para resolver este problema;

ii) <u>precisão da solução</u> - justamente para resolver o dilema: convergência assintótica (número muitas vezes grande de iterações necessárias) X tempo finito disponível (número

limitado de iterações), é que, na prática, processos de busca linear exata geram, em contraste com seus nomes, soluções inexatas (!) para o problema (4.13).

Afora estes aspectos não devemos esquecer que o desempenho dos métodos de b.l.e. depende quase sempre das propriedades da função a ser minimizada. Por exemplo, ao se usar um algoritmo de interpolação em uma função cujo comportamento não tem localmente uma boa aderência com a aproximação polinomial utilizada, a busca linear pode ser interrompida em uma aproximação sofrível do passo ótimo. Em particular, a unimodalidade é uma propriedade tipicamente exigida de funções para garantir-se convergência de processos de busca exata.

No sentido de evitar estes inconvenientes das buscas lineares "exatas" (a esta altura podemos usar os apóstofros), diversos estudos foram efetuados a partir do final da década de 60 (por exemplo, Armijo em 1966, Goldstein em 1967 e Wolfe em 1969, ver referências em Cohen [39]) no sentido de se obter algoritmos aproximativos para a resolução da minimização unidimensional, chamados métodos de busca linear inexata. Tais métodos correspondem simplesmente a algoritmos (por definição, finitos) que obtenham uma aproximação $\bar{\lambda}$ do passo ótimo λ^* tal que a condição

$$f(x_k + \bar{\lambda} d_k) < f(x_k) \qquad (4.18)$$

seja satisfeita. Em outras palavras, estes métodos buscam apenas um ponto tal que haja um decréscimo (sempre que possível, significativo) no valor da função na k-ésima iteração.

Neste ponto, cabe colocarmos aqui dois fatos até hoje indiscutíveis, verificados na prática:

1º-Não é verdade que, em benefício da convergência global de um método de minimização irrestrita, buscas lineares exatas sempre sejam preferíveis às inexatas, isto é, nem sempre uma melhor minimização local levará à uma melhor (mais rápida e precisa) minimização global;

2º-Embora a intuição possa em um primeiro momento levar-nos a a acreditar que uma busca exata deve oferecer algoritmos mais "eficientes" na obtenção de novas direções, verificamos na prática que uma busca linear inexata é na maioria das vêzes mais interessante do ponto de vista do volume de cálculos envolvido e até mesmo da velocidade real de convergência em alguns casos. THOMPSON, em [28], apresenta casos para os quais o uso de buscas lineares no MGC faz inclusive com que este não convirja, sem fazer comentários sobre buscas inexatas.

Embora a teoria do MGC apresentada no capítulo II esteja baseada no uso de buscas lineares exatas, sabemos como são custosas ou até mesmo irrealizáveis na prática. Em geral, uma busca exata pode ser substituída por outro processo iterativo (b.l. inexata) que vise obter uma aproximação tal que d_k e r_{k+1} sejam "quase" ortogonais. O uso criterioso de buscas inexatas não afeta a convergência do MGC, se pudermos garantir que o processo iterativo seja de descida (ver, por ex.,[18], [20], [23] e [31]).

Uma vez justificada a nossa preferência pelas buscas lineares inexatas, apresentaremos a seguir os algoritmos de Armijo, Goldstein e Wolfe acima citados. Estes métodos aproximativos constituem a chamada família de "técnicas modernas de busca linear" [34], e diferem das buscas exatas pelo fato de que, enquanto estas tentam calcular (na prática, aproximar) o passo ótimo, as inexatas buscam determinar e refinar (os limites de)um intervalo de incerteza contendo um conjunto de estimativas do qual tomando-se qualquer elemento, tenhamos assegurada uma redução (sempre que possível)significativa no valor da função-objetivo.

Antes de apresentar os algoritmos propriamente ditos, cabe citarmos a importância da existência de critérios de parada independentes do algoritmo a ser utilizado. Embora cada um dos algoritmos tenha seus próprios testes de convergência, é importante que incluamos neles testes adicionais (salvaguardas) que:

i) limitem o número de buscas a serem efetuadas na minimização unidimensional, e que

ii) interrompam a busca no caso em que, embora não detectado pelos testes existentes, o intervalo de incerteza se tornar muito pequeno ou, equivalentemente, menor que uma constante muito pequena; em algoritmos computacionais, esta constante pode ser da ordem da precisão de máquina.

IV.6.2 - O algoritmo de Armijo:

Desenvolvido em 1966, este algoritmo se constitui em um processo aproximativo bem simples para gerar e refinar um intervalo de incerteza. Segundo LEMARECHAL [34], este algoritmo (como os de Goldstein e Wolfe) opera com dois objetivos em relação ao λ_k escolhido:

a) não deve ser grande demais pois, conforme a função, a direção só poderá ser de descida localmente. Logo, é conveniente que x_{k+1} não esteja muito distante de x_k.

b) não deve ser, no entanto, pequeno demais, pois neste caso, considerando-se $\|d_k\|=1$, teremos duas consequências inconvenientes:

   i) $x_{k+1} \approx x_k$, o que implica em uma progressão irrisória para o MGC;

   ii) $\|r_{k+1}-r_k\| \approx 0$, que, no caso de usarmos a expressão (2.29) de Polak-Ribière para o cálculo do coeficiente de conjugação β_k, fará com que $\beta_k \approx 0$ e, portanto a próxima direção de busca, d_{k+1}, seja bem próxima do antigradiente r_{k+1}. Isto fará com que o MGC se porte na próxima iteração praticamente como o Método do Gradiente, reduzindo possivelmente a velocidade de convergência.

Para resolver (a), a seguinte estratégia é adotada: considerando-se as definições (4.14) a (4.16), escolhemos um escalar $a \in (0,1)$ e a condição a seguir:

$$h(\lambda) \leq h(0) + ah'(0)\lambda , \qquad (4.19)$$

que, geometricamente, equivale a definirmos um conjunto de valores para λ (não necessariamente contínuo) tal que o gráfico de h, nestes valores, esteja abaixo de uma reta de inclinação $ah'(0)$, isto é, uma fração da inclinação inicial $h'(0)$, como na Fig.IV.5 .

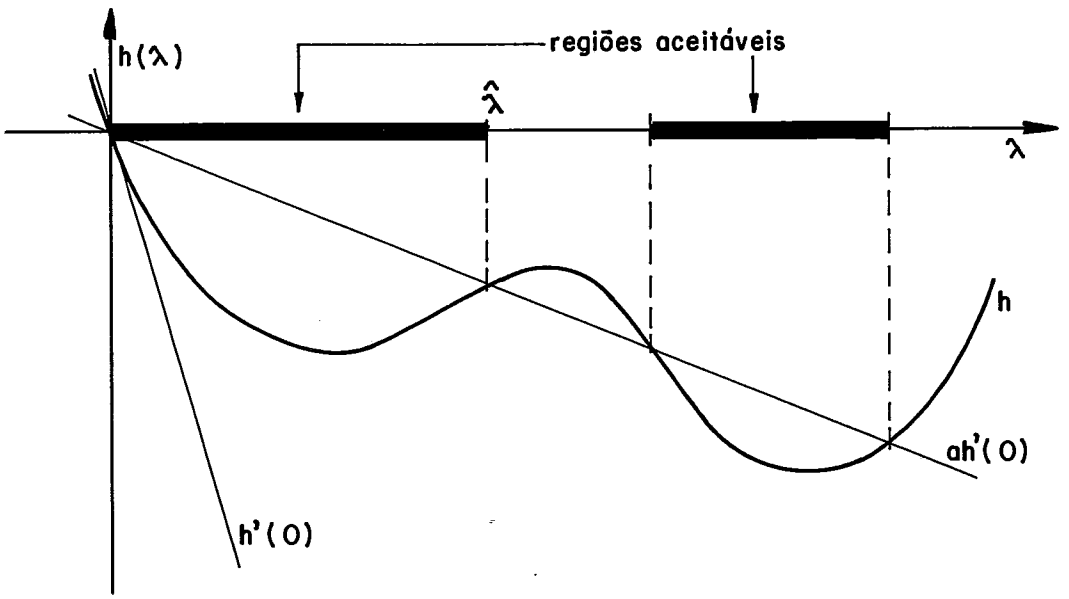


Fig.IV.5

Sendo d_k direção de descida e h limitada inferiormente, vê-se claramente que o conjunto de valores de λ que obedece (4.19) é necessariamente não-vazio (contendo pelo menos o intervalo $[0,\hat{\lambda}]$, onde $\hat{\lambda} > 0$) e limitado, e que além disto, qualquer desses valores garante uma redução do valor da função.

O próximo passo consiste em se escolher um valor inicial de λ, denominado λ^o. Se λ^o não satisfizer (4.19), podemos considerá-lo um limite λ_D à direita de tal forma que procuremos outro valor λ^1 no intervalo $[0,\lambda^o]$. Este processo é repetido até que se encontre um valor que satisfaça (4.19).

É importante frisarmos que, primeiro, se por um lado este processo gera uma sequência $\{\lambda^i\}$, onde $\lambda^{i+1} < \lambda^i$, no sentido de satisfazer (a), por outro, o simples fato de o processo parar no primeiro λ^i que obedecer (4.19) deve ser visualizado como uma salvaguarda que evite a escolha de um $\lambda^i \approx 0$, isto é, o algoritmo tenta obter um valor λ_k que satisfaça (b). Apresentamos então o algoritmo puro de Armijo:

Passo 0: Dados x_k, d_k, $h(0)=f(x_k)$ e $h'(0) = \langle g_k, d_k \rangle$, escolhe-se $0 < a < 1$ e $\lambda^0 > 0$ fixos para todas as iterações k de algum algoritmo de min.irrest. . Faça $\lambda \leftarrow \lambda^0$.

Passo 1:
$$\begin{cases} x \leftarrow x_k + \lambda d_k \\ f(x) \leftarrow f(x_k + \lambda d_k) \end{cases}$$

Passo 2: Se $f(x) \leq f(x_k) + a \langle g_k, d_k \rangle \lambda$

então

$$\begin{cases} \lambda_k \leftarrow \lambda \\ \text{vá para o Passo 3.} \end{cases}$$

senão

$$\begin{cases} \lambda_D \leftarrow \lambda \\ \lambda \leftarrow \text{INTERPOL}(\lambda_D) \\ \text{vá para o Passo 1.} \end{cases}$$

Passo 3: Pare. λ_k é a aproximação obtida.

Observações sobre o algoritmo de Armijo:

i) Por conveniência, esta versão (pura) do algoritmo de Armijo não foi apresentada com os testes de salvaguarda (número máximo de buscas e tamanho mínimo do intervalo $[0,\lambda_D]$) antes citados. É importante incluí-los.

ii) No passo 2, a função INTERPOL (λ_D) é efetuada por qualquer fracionamento de λ_D, por exemplo, $\lambda \in [0.1\ \lambda_D, 0.9\lambda_D]$ e, em particular, $\lambda \leftarrow 0.7\lambda_D$ parece-nos satisfatório.

iii) Observe que, quanto mais $a \to 0$, maior será o domínio onde se procurará um valor λ que satisfaça (4.19), logo a probabilidade de se encontrar rapidamente uma aproximação satisfatória para λ_k pode aumentar, conforme a natureza da função e da magnitude da estimativa inicial fornecida.

iv) O esforço computacional de cada iteração do algoritmo de Armijo é essencialmente dado pelo cálculo de f(x) no Passo 1 e pelo teste de desigualdade no Passo 2.

IV.6.3 - O algoritmo de Goldstein :

Este algoritmo, apresentado em 1967, pode ser visto como uma versão aperfeiçoada do algoritmo de Armijo. A diferença básica entre ambos consiste na forma de atingir o objetivo (b), descrito em IV.6.2. Enquanto o algoritmo de Armijo opera sempre com um intervalo de incerteza $[0,\lambda_D]$, o algoritmo de Goldstein (e também o de Wolfe, como veremos) tenta obter uma região do tipo $[\lambda_E,\lambda_D]$, $0 < \lambda_E < \lambda_D$, onde o cálculo de λ_E (limite à esquerda da região aceitável) é feito durante a busca linear,no sentido de obter uma estimativa final do passo não muito próxima de zero. No caso de Goldstein, este problema é resolvido da seguinte forma: considerando novamente as definições (4.14) a (4.16) e a condição (4.19), escolhe-se um escalar $b \in (0,1)$, tal que $0 < a < b < 1$, e define-se a condição a seguir (lembre que $h'(0) < 0$):

$$h(\lambda) \geq h(0) + bh'(0)\lambda \qquad (4.20)$$

que, usada conjuntamente com(4.19),define um intervalo $[\lambda_E,\lambda_D]$ contendo a união das regiões de estimativas aceitáveis. Tomando um exemplo convexo, temos:

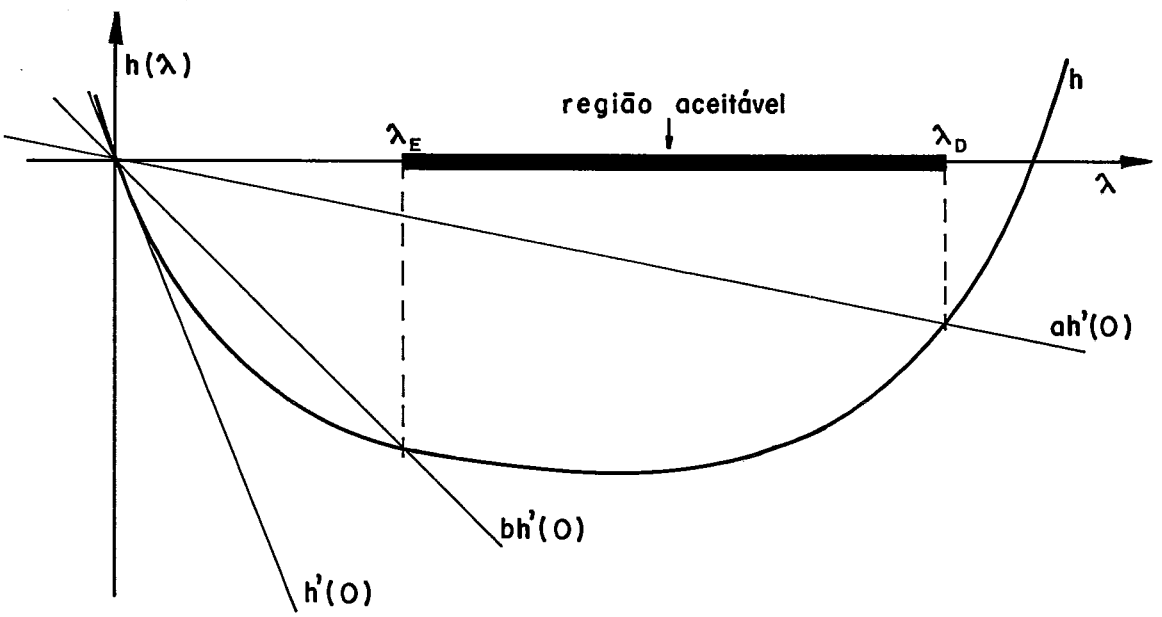


Fig.IV.6

Note que, enquanto (4.19) obriga que o conjunto de regiões aceitáveis esteja à esquerda de λ_D(impedindo estimativas muito afastadas de zero), a condição (4.20) impede que se escolha

estimativas muito pequenas, proibindo a busca no intervalo $(0,\lambda_E)$. A condição (4.20) é baseada em um "esperado bom comportamento" de h, isto é, se $h(\lambda)$ é "suficientemente diferente" de $h(0)$, então λ, por sua vez, também deve ser "suficientemente diferente" de 0.

A busca linear de Goldstein consiste basicamente em:

1º) gerar λ_D que garanta (4.19);
2º) gerar λ_E que garanta (4.20);
3º) escolher como aproximação final $\lambda_k \in [\lambda_E,\lambda_D]$.

Na prática, dada uma estimativa $\lambda^0>0$ e $\lambda_D=\lambda_E=0$, o algoritmo verifica se λ^0 já obedece simultaneamente a (4.19) e a (4.20). Se isso acontecer, a busca é terminada. Porém, se λ^0 desobedecer a (4.19)(ou a (4.20)),ele é considerado como o novo λ_D (ou o novo λ_E) e é então gerado um novo valor λ^1, interpolado do intervalo $[\lambda_E,\lambda_D]$ (ou extrapolado a partir de λ_E). Apresentamos a seguir o algoritmo puro de Goldstein:

Passo 0: Dados x_k, d_k, $h(0)=f(x_k)$ e $h'(0)=\langle g_k,d_k\rangle$, escolhemos $0<a<b<1$ e $\lambda^0>0$, fixos em todas as iterações k.
Faça $\lambda_E \leftarrow \lambda_D \leftarrow 0$ e $\lambda \leftarrow \lambda^0$.

Passo 1: Faça $x \leftarrow x_k + \lambda d_k$ e $f(x) \leftarrow f(x_k + \lambda d_k)$.

Passo 2: Se $f(x) > f(x_k) + a\lambda\langle g_k,d_k\rangle$
então
    $\lambda_D \leftarrow \lambda$
    Vá para o Passo 5.

Passo 3: Se $f(x) < f(x_k) + b\lambda\langle g_k,d_k\rangle$
então $\lambda_E \leftarrow \lambda$
senão
    $\lambda_k \leftarrow \lambda$
    Pare. O valor λ_k é a aproximação final.

Passo 4: Se $\lambda_D = 0$
então
    $\lambda \leftarrow$ EXTRAPOL(λ_E)
    Vá para o Passo 1.

Passo 5: $\lambda \leftarrow$ INTERPOL(λ_E,λ_D)
Vá para o Passo 1.

Observações sobre o algoritmo de Goldstein:

i) Como no de Armijo, não incluímos os testes de salvaguarda.

ii) No Passo 1, é efetuado o cálculo de $h(\lambda)$.

iii) No Passo 2, é verificado se λ obedece a (4.19). Se obedecer, o algoritmo segue em frente (para verificar se λ obedece a (4.20). Se não obedecer, fixa-se λ_D e procura-se (Passo 5) novo valor para λ tal que $0 \leq \lambda_E < \lambda \leq \lambda_D$

iv) Ao atingir o Passo 3, sabe-se que λ obedece a (4.19). Verifica-se então se λ obedece também a (4.29); se obedecer, fim. Senão, fixa-se λ_E e procura-se novo valor para λ tal que $\lambda > \lambda_E$.

v) No Passo 4, já se tem $\lambda_E > 0$ e, se $\lambda_D = 0$, extrapola-se λ baseado apenas em λ_E; caso contrário ($\lambda_D \neq 0$), segue-se para uma interpolação em $[\lambda_E, \lambda_D]$.

vi) Ao atingir o Passo 5, já se tem $0 \leq \lambda_E < \lambda_D$. A interpolação (função INTERPOL) é efetuada por uma escolha qualquer do tipo $\lambda \in [\lambda_E + 0.1(\lambda_D - \lambda_E),\ \lambda_E + 0.9(\lambda_D - \lambda_E)]$ e, em especial, $\lambda \leftarrow 0.5(\lambda_D + \lambda_E)$ parece-nos satisfatório.

vii) A extrapolação (função EXTRAPOL) no Passo 4 pode ser efetuada através de uma atribuição do tipo $\lambda \leftarrow j\lambda_E$, $j = \{2,3,\ldots\}$, por exemplo; advertimos que se evite tomar $j >> 0$.

IV.6.4- O algoritmo de Wolfe:

Divulgado em 1969, este algoritmo é uma variante do anterior, diferindo deste justamente pela estratégia usada para atingir o objetivo (b) - ver IV.6.2. Enquanto Goldstein tenta obter um valor de λ tal que $h(\lambda)$ seja "suficientemente diferente" de $h(0)$, Wolfe trabalha com a derivada de h e procura determinar um valor de λ tal que $h'(\lambda)$ seja "suficientemente diferente" de $h'(0)$. Devido à continuidade de h', neste caso espera-se que x_{k+1} seja "suficientemente diferente" de x_k [34]. Em síntese, Wolfe considera como uma região aceitável qualquer conjunto de valores para λ que satisfaça simultaneamente a (4.19) e a:

$$h'(\lambda) \geq b\, h'(0), \text{ onde } 0 < b < 1 \qquad (4.21)$$

Para melhor visualizar a importância de (4.21), tomemos uma estimativa λ^i que não a satisfaça; logo, na vizinhança de λ^i, $h(\lambda)$ está decrescendo a uma taxa maior que $|bh'(0)|$. Uma atitude sensata é a de que deva-se tomar λ^{i+1} maior (=mais à direita) que λ^i. Vemos que, devido à continuidade de h' ,(4.21) não poderá ser satisfeita por valores muito pequenos de λ e, consequentemente, teremos o objetivo (b) alcançado. Considerando por exemplo o gráfico de h abaixo, teríamos as regiões 1 e 2 satisfazendo (4.21), mas não necessariamente (4.19):

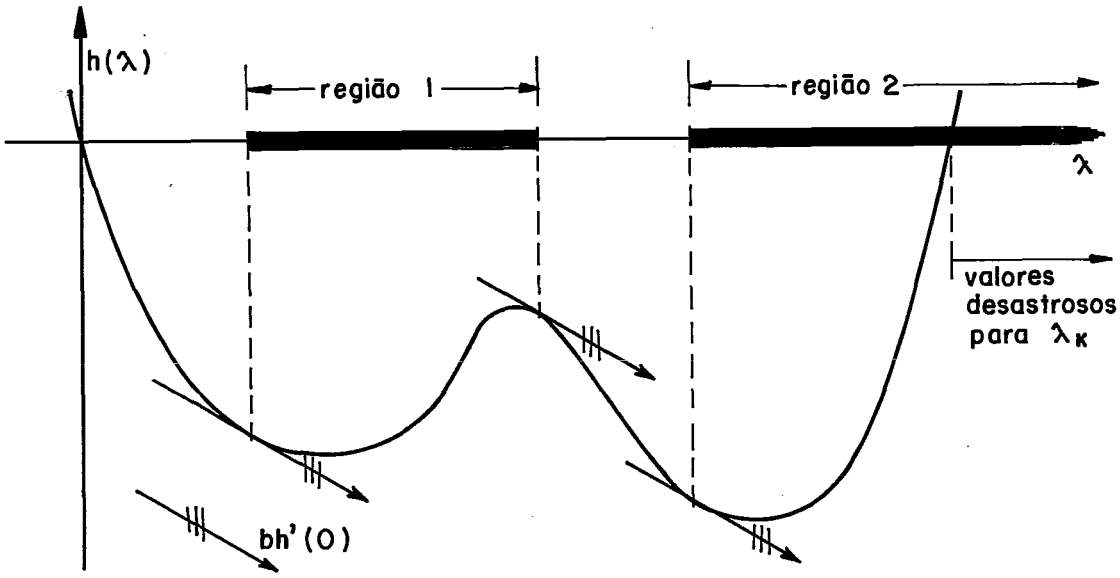


Fig.IV.7

Neste gráfico-exemplo, podemos ver que, entre os valores de λ satisfazendo (4.21), existem alguns que (se escolhidos) fariam com que o MGC perdesse sua propriedade de ser de descida. Entretanto, a condição (4.19) com $a > 0$, impede tal desastre.

A busca linear de Wolfe pode ser resumida na seguinte sequência de operações:

1º) Dada uma estimativa inicial λ, enquanto (4.19) não for atendida, diminuímos λ.

2º) Sendo (4.19) atendida, testamos (4.21);enquanto não for atendida, aumentamos λ. Quando(4.21)for atendida,pare e faça $\lambda_k \leftarrow \lambda$.

A convergência deste processo é garantida [34] pelo seguinte resultado:

"Se $0 < a < b < 1$ e h for continuamente diferenciável e limitada inferiormente, então sempre existirá um intervalo não vazio de estimativas aceitáveis para λ_k (na definição de Wolfe). E mais, se λ não satisfizer (4.19), então existe uma região aceitável à sua esquerda, e, se λ satisfizer (4.19) mas não (4.21), então existe uma região aceitável à sua direita".

Segue-se o algoritmo puro de Wolfe:

Passo 0: Dados x_k, d_k, $h(0) = f(x_k)$ e $h'(0) = g_k^t d_k$, escolha $0 < a < b < 1$ e $\lambda^0 > 0$ fixos para todas as iterações k, e faça $\lambda_E \leftarrow \lambda_D \leftarrow 0$ e $\lambda \leftarrow \lambda^0$

Passo 1: $\left[\begin{array}{l} x \leftarrow x_k + \lambda d_k \\ f(x) \leftarrow f(x_k + \lambda d_k) \end{array}\right.$

Passo 2: Se $f(x) > f(x_k) + a \langle g_k, d_k \rangle \lambda$
então
$\left[\begin{array}{l} \lambda_D \leftarrow \lambda \\ \text{Vá para o Passo 5.} \end{array}\right.$

Passo 3: Se $\langle g(x), d_k \rangle < b \langle g_k, d_k \rangle$
então $\lambda_E \leftarrow \lambda$
senão
$\left[\begin{array}{l} \lambda_k \leftarrow \lambda \\ \text{Pare. } \lambda_k \text{ é a aproximação obtida.} \end{array}\right.$

Passo 4: Se $\lambda_D = 0$
então
$\left[\begin{array}{l} \lambda \leftarrow \text{EXTRAPOL } (\lambda_E) \\ \text{Vá para o Passo 1.} \end{array}\right.$

Passo 5: $\left[\begin{array}{l} \lambda \leftarrow \text{INTERPOL } (\lambda_E, \lambda_D) \\ \text{Vá para o Passo 1.} \end{array}\right.$

Observações sobre o algoritmo de Wolfe:

i) Como no de Armijo, não incluímos os testes de salvaguarda.
ii) No Passo 1, é calculado $h(\lambda)$.
iii) Idem a (iii) de Goldstein, trocando-se (4.20) por (4.21).
iv) Idem a (iv) de Goldstein, trocando-se (4.20) por (4.21).
Verifique que no Passo 3 de Wolfe, embora implícito,

devemos inicialmente calcular $h'(\lambda) \leftarrow \langle g(x_k+\lambda d_k), d_k \rangle$.

v),vi),vii) Idem às observações (v), (vi) e(vii) de Goldstein.

viii)Cabe colocar a opinião de LEMARECHAL em [34], que vem utizando técnicas modernas de buscas lineares em seus trabalhos na área de Otimização Diferenciável e Não-Diferenciável. Em relação ao algoritmo de Wolfe, em particular, ele afirma que "esta técnica é atualmente (1979) uma das mais eficientes entre todos os métodos de descida existentes que se utilizam (apenas) dos valores de f(x) e g(x)". Os argumentos para esta afirmação são os seguintes:

- os teoremas mais recentes sobre a convergência de algoritmos práticos (isto é, sem b.l.exatas) utilizam as condições (4.19) e (4.21).
- o algoritmo de Wolfe não depende de convexidade ou da unimodalidade (ou nada deste tipo) de f.
- em Otimização Não-Diferenciável, este algoritmo, com alguma generalização,apresenta bons resultados.

ix) A grande deficiência do algoritmo é sua sensibilidade a possíveis erros de arredondamento; se o cálculo de $h'(\lambda)$, em aritméticas de ponto-flutuante, não refletir razoavelmente a derivada de h, o algoritmo pode divergir.

IV.7 - Um algoritmo de GC para funções gerais:

Desejamos, nesta seção,apresentar uma versão implementável do MGC, aplicável à minimização de funções gerais. Por conveniência de quem for usá-la, deixamos em aberto:

- a escolha do coeficiente de conjugação β_k, embora devamos sugerir, pelo que foi apresentado, as propostas por Hestenes - Stiefel (2.28) e por Polak-Ribière (2.29);
- os testes de convergência, já discutidos em IV.4;
- o algoritmo de busca linear; a nossa sugestão recai nos métodos de Wolfe e de Goldstein apresentados em IV.6. Consequentemente, os parâmetros:
  - frações a e b de inclinação,
  - número máximo de buscas lineares em uma iteração,

tamanho mínimo permitido do intervalo $|\lambda_D - \lambda_E|$, devem ser escolhidos também pelo usuário, como melhor lhe convier.

A versão apresentada contém a reinicialização a cada n iterações aliada à técnica de Beale, aos critérios de Powell e os testes de parada apresentados em IV.4. Denominaremos a partir de agora esta versão de Algoritmo 5.

Passo 0:
- Dados
  - n, $f(x)$, $f'(x)$, x_1
  - MAXITER, ε, (parâmetros para convergência)
  - a, b, NMAXBL (parâmetros para busca linear)
- tome
  - $k \leftarrow 1$; $f_1 \leftarrow f(x_1)$; $d_1 \leftarrow r_1 \leftarrow -f'(x_1)$
  - $c \leftarrow 1$

Passo 1:

1.1 - Se CONVERGÊNCIA (x_k, f_k, r_k, k) então pare. $x^* = x_k$ é a solução procurada.

1.2 - BUSCA LINEAR$(x_k, f_k, r_k, d_k, a, b, \text{NMAXBL}, \text{ERRO}, \lambda_k)$
Se ERRO então pare.
senão
- $x_{k+1} \leftarrow x_k + \lambda_k d_k$
- $f_{k+1} \leftarrow f(x_{k+1})$
- $r_{k+1} \leftarrow -f'(x_{k+1})$

1.3 - Se $|r_k^t r_{k+1}| \geq 0.2\|r_{k+1}\|^2$ ou $k = c+n-1$
então
- $c \leftarrow k+1$
- $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$

senão
- $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k + \gamma_{k,c} d_c$
- Se NOT$(0.8\|r_{k+1}\|^2 \leq d_{k+1}^t r_{k+1} \leq 1.2\|r_{k+1}\|^2)$
- então
  - $c \leftarrow k+1$
  - $d_{k+1} \leftarrow r_{k+1} + \beta_k d_k$

1.4 - $k \leftarrow k+1$
Volte para 1.1

IV.8 - Sobre convergência de algoritmos de GC em funções não-quadráticas:

A maior parte dos resultados existentes sobre a convergência destes algoritmos nesses casos está baseada em particularidades da função-objetivo, no uso de aritmética exata, na realização de buscas lineares exatas ou "praticamente" exatas (ver[8], [11], [14], [18], [22], [23], [24], [26], [30] e [32]) e sobre o uso ou não de alguma técnica de "restart" (bem como de qual a técnica usada). Porém, alguns resultados importantes obtidos nesta área, além dos já citados em III.4, devem ser mencionados[40]:

1. Para funções suficientemente suaves, o MGC (Algoritmo 4, com r=n e coeficiente de conjugação definido por (2.29)) usado com aritmética e busca linear exatas apresenta, em geral (caso fortemente convexo, por exemplo), convergência superlinear a cada n iterações, isto é, existe $1 < p < 2$ tal que:

$$\lim_{j\to\infty} \frac{\|x^* - x_{nj+n}\|}{\|x^* - x_{nj}\|^p} = 0$$

Esta taxa, no entanto, é extremamente sensível a mudanças na estratégia de reinicialização usada pelo algoritmo de GC em questão.

2. Por outro lado, algoritmos computacionais de GC sofrem, na prática, influência da acumulação de erros de arredondamento; esse fenômeno pode degradar as taxas superlineares citadas acima. GILL e MURRAY [40] afirmam que, em suas experiências com algoritmos de GC, verificaram, na grande parte das funções-teste, a ocorrência de taxas lineares, tanto em algoritmos com como sem técnicas de reinicialização. Eles citam que "as exceções ocorrem em circunstâncias muito especiais tais como casos onde f era uma função quadrática bem-comportada, e que, quando n for muito grande, a verificação da taxa de convergência (linear ou superlinear a cada n iterações) torna-se em geral impraticável, devido ao grande número de iterações necessárias à aplicação dos resultados teóricos de convergência assintótica".

3. Em casos particulares, onde $H(x^*)$ tem autovalores agrupados ("clustered") em determinados conjuntos compostos por autovalores de mesma magnitude, o problema pode ser resolvido em bem menos do que n iterações. Por sua vez, problemas sem esta característica requerem em geral um número de iterações entre n e $5n$,

com uma média de 2n iterações. A conclusão final é a de que qualquer implementação satisfatória do MGC deve atingir uma aproximação significativamente precisa de uma solução em, no máximo, 5n iterações.

Por sua vez, a análise de SHANNO em [31], após uma série de testes com várias versões do MGC, mostra basicamente que o uso do "restart" de Beale melhora, na grande maioria dos casos, a velocidade do MGC. Além disto, este uso permite que se possa relaxar bastante a exatidão das buscas lineares feitas e, mesmo assim, a velocidade de convergência ainda chega a aumentar significativamente! Finalizando, Shanno afirma categoricamente em relação aos resultados de seus testes que as buscas lineares inexatas se mostram nitidamente preferíveis às exatas, no que diz respeito à velocidade global de convergência dos MGCs.

CAPÍTULO V

EXPERIÊNCIAS COMPUTACIONAIS E CONCLUSÕES

V.1 - Algoritmos, funções, estratégias dos testes e metodologia de comparação utilizadas:

Nesta seção, desejamos estabelecer as ferramentas com as quais trabalharemos neste capítulo, como também procuraremos justificar até certo ponto o porquê da escolha dos algoritmos, das funções-teste, da estratégia estabelecida para os testes e da metodologia empregada na avaliação e comparação do desempenho desses algoritmos. Cabe-nos, de antemão, lembrar que os testes aqui efetuados foram escolhidos de forma a ilustrar, para alguns poucos casos reais, o restante deste trabalho. Quaisquer resultados obtidos, principalmente para funções não-quadráticas, não nos devem levar a pragmatismos.

V.1.1 - Algoritmos escolhidos:

Em vista do que foi apresentado nos últimos dois capítulos, nos concentramos básicamente em duas famílias de algoritmos de GC, representadas por:

i) o ALGORITMO 4, apresentado em III.4, com r=n;
este algoritmo representa a família clássica dos MGC com reinicializações a cada n iterações; e

ii) o ALGORITMO 5, apresentado em IV.7, que nada mais é que uma extensão do anterior, incorporando a técnica de Beale e os critérios de Powell, já apresentados em IV.5.

Para ambos os algoritmos construiremos algumas variantes, obtidas quando combinarmos diferentes técnicas de buscas lineares com diferentes coeficientes de conjugação. Primeiramente,em relação a buscas lineares, concentrar-nos-emos no uso dos algoritmos de Goldstein e de Wolfe, em vista do exposto em IV.6. Quanto aos coeficientes de conjugação, alternaremos as fórmulas de Hestenes-Stiefel (2.28), de Polak-Ribière (2.29) e de Fletcher-Reeves (2.30), apresentadas em II.4, visto só utilizarem informações sobre derivadas de primeira ordem. A descrição das

variantes usadas, bem como os valores estabelecidos para os parâmetros necessários à otimização (p.ex.: tolerância para os testes de parada e as inclinações para o algoritmo de busca linear) serão apresentados junto com a exposição da estratégia dos testes, em V.1.3 .

Alertamos o leitor para o fato de que os algoritmos de busca linear usados foram programados com as seguintes características:

i) sem limite no número de tentativas, pois desejamos apresentar o esforço máximo do sub-algoritmo para convergir. A ausência desse limite não causará problemas nas nossas experiências, já que todas as funções dadas são limitadas inferiormente.

ii) se a busca linear gerar um passo insignificante na direção d_k, esta direção será substituída pela do antigradiente r_k, através da qual será efetuada outra busca linear.

iii) ao final de qualquer busca linear, é realizado um teste para verificar se a busca feita degradou significativamente a ortogonalidade entre d_k e r_{k+1}. Neste caso, é executada uma interpolação quadrática, sendo um novo passo gerado e comparado com o anteriormente obtido. O algoritmo escolhe aquele que garante a maior redução funcional. Embora, a princípio, o comportamento desta salvaguarda possa sugerir uma busca "mais exata", o objetivo de sua inserção é o de tentar preservar duas características teóricas fundamentais do MGC no caso quadrático, quais sejam:

   - a ortogonalidade entre d_k e r_{k+1} ;
   - considerar as informações de segunda ordem (curvatura) contidas em uma aproximação quadrática da projeção da função-objetivo nessa direção.

Em relação aos testes de parada, usamos o conjunto apresentado em IV.4. Os testes absolutos T1 e T2 serão substituídos respectivamente por T1a T2a sempre que $O(|f_{k+1}|) \geq O(10)$ e/ou $O(\|x_{k+1}\|) \geq O(10)$. Consideramos $\varepsilon = 10^{-6}$ para todos os problemas, exigindo consequentemente uma precisão de, pelo menos, 6 dígitos entre $f(\hat{x})$ e $f(x^*)$ (ou f^*).

Todas as experiências foram processadas no computador IBM/370-158 do Laboratório de Computação Científica (LCC) do CNPq , usando-se precisão dupla e tendo-se programado todos os algoritmos em FORTRAN-IV.

V.1.2 - Conjunto de funções-teste:

Apresentamos a seguir o conjunto de funções usado nos testes efetuados. Na próxima seção, descreveremos como pretendemos usá-las na estratégia de comparação dos algoritmos.

Basicamente, as funções usadas podem ser classificadas em três grandes classes, a saber:

A - funções quadráticas (bem ou mal-comportadas),
B - funções não-quadráticas convexas,
C - funções não-quadráticas quaisquer,

sendo que, para todas estas classes, apresentaremos funções em dimensões baixas (n=2,4) e altas (n=40,50,100,200, por exemplo). Todos os problemas são de minimização. Seguem-se as funções:

A.1) Funções quadráticas no $\mathbb{R}^2$:

Muito bem-comportada:

I) $f(x)=x_1^2 + 1.2x_2^2$

$x^1=(1000,1)^t$, $x^*=(0,0)^t$, $f^*=0$

Muito mal-comportada:

II) $f(x)=x_1^2 + 1000x_2^2$

$x^1=(1000,1)^t$, $x^*=(0,0)^t$, $f^*=0$

A.2) Funções quadráticas em dimensões mais altas:

Muito bem-comportada:

III) $f(x)=\sum_{i=1}^{n}(a_i x_i^2)$, n=100

$a_i=1$, se i é par, $a_i=1.5$, se i é ímpar

$x^1=(5,-5,5,-5,\ldots,5,-5)^t$

$x^*=(0,0,\ldots,0)^t$, $f^*=0$

IV) Idêntica a III, com n=200

Muito mal-comportadas:

V) função de Oren-Spedicato alongada

$f(x)=\sum_{i=1}^{n}(i^2 x_i^2)$, n=100

$x^1=(1,1,\ldots,1)^t$, $x^*=(0,0,\ldots,0)^t$, $f^*=0$

VI) idêntica a V, com n=200

B) Função não-quadrática convexa:

VII) $f(x)= \sum_{i=1}^{n} (x_i-1)^4$, n=50

$x^1 = (0,0,\ldots,0)^t$, $x^* = (1,1,\ldots,1)^t$, $f(x^*)=0$

VIII) idêntica a VII, com n=200

C) Funções não-quadráticas gerais [36]:

IX) função de Rosenbrock no $\mathbb{R}^2$:

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

$x^1= (-1.2,1)^t$, $x^* = (1,1)^t$, $f^* = 0$

X) função de Wood no $\mathbb{R}^4$ (ou função de Rosenbrock no $\mathbb{R}^4$):

$f(x)=100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_3^2 - x_4)^2 +$

$+ (1-x_3)^2 + 10.1[(x_2-1)^2 + (x_4-1)^2] + 19.8(x_2-1)(x_4-1)$

$x^1 = (-3,-1,-3,-1)^t$, $x^k=(1,1,1,1)^t$, $f^*=0$

XI) função extendida de Powell no $\mathbb{R}^n$:

$f(x)= \sum_{i=1}^{n/4} [(x_{4i-3}+10x_{4i-2})^2 + 5(x_{4i-1}-x_{4i})^2 +$

$+ (x_{4i-2}+2x_{4i-1})^4 +10(x_{4i-3}-x_{4i})^4]$, n=40

$x^1 = (3,-1,0,3,3,-1,0,3,\ldots,3,-1,0,3)^t$

$x^* = (0,0,\ldots,0)^t$, $f^*= 0$

XII) idêntica a XI, com n=200

V.1.3 - Estratégia dos testes:

Apresentados os 2 algoritmos de GC e o conjunto de funções-teste, pretendemos, com as experiências computacionais a seguir, ilustrar significativamente o desempenho do MGC em diferentes tipos de funções; em particular, queremos confrontar resultados obtidos com as alternativas na implementação destes métodos, a saber:

- a técnica de Beale e os critérios de Powell,
- o sub-algoritmo de busca linear,
- as frações de inclinação usadas na busca linear, e
- o coeficiente de conjugação a ser usado no MGC.

Neste sentido, a nossa estratégia de testes consistirá em abordar, com os 2 algoritmos, as 12 funções dadas, conforme explicado abaixo. Citamos também que f e f' são calculadas analiticamente.

Para os testes, consideramos as seguintes opções:

- quanto aos métodos de busca linear: Goldstein (G) e Wolfe (W).
- quanto ao coeficiente de conjugação: Hestenes-Stiefel (HS), Fletcher-Reeves (FR) e Polak-Ribière (PR).
- quanto às frações a e b de inclinação, os pares: (a,b) = (0.1,0.9), (0.4,0.6) e (0.5,0.9).

Logo, teremos para cada função:

2 algoritmos de gradientes conjugados
2 buscas lineares
3 pares de frações de inclinação
x 3 coeficientes de conjugação

36 minimizações,

fornecendo um total de (36 x 12 funções) 432 testes.

Com esta estratégia, não procuramos conlcusões pragmáticas e definitivas, mas sim, mostrar a importância de se tentar abordar um determinado problema através de uma boa quantidade de alternativas, sempre que possível. Lembramos que outras alternativas como, por exemplo:

- variação do ponto inicial,
- outros conjuntos de testes de parada,
- outros algoritmos de busca linear,

não foram incluídas neste trabalho , devendo, porém, ser também consideradas em casos reais.

V.1.4 - Metodologia de comparação:

Como já advertimos, neste capítulo desejamos ilustrar com algumas experiências computacionais o desempenho de algoritmos de gradientes conjugados na otimização de funções reais continuamente diferenciáveis. Neste sentido, não buscamos, por meios empíricos, provar definitivamente que tal versão é melhor que outra, mas sim fornecer parâmetros que possam vir a ser usados para se dizer se um determinado algoritmo é (e o quanto é) eficiente na otimização de determinadas (classes de) funções.

Consequentemente, os resultados obtidos, a serem apresentados na próxima seção, serão expostos da seguinte forma: dados um algoritmo e um problema, temos:

(a) o "grau de otimização", isto é, aonde o algoritmo foi interrompido e o quanto otimizou o problema, e

(b) o esforço empreendido para alcançar este grau.

Se, por um lado, a mensuração de (a) torna-se trivial através da esposição do ponto alcançado $\hat{x}$, se tivermos a solução x^*, e os valores $f(\hat{x})$ e $f(x^*)$, por outro, não existe uma só forma (ou mesmo a melhor forma) para medir e representar o esforço dispendido, ou seja, o quanto custou ao algoritmo para atingir determinado grau de otimização de um problema. No trabalho de BRAGA [43], temos uma consideração que reflete bem a não-uniformidade (e até mesmo a enorme subjetividade) existente no uso de critérios para avaliação de desempenho, quando afirma que "algoritmos podem ser avaliados por uma grande variedade de critérios, desde considerações puramente matemáticas do tipo 'velocidade de convergência' até observações absolutamente pragmáticas do tipo 'o algoritmo é bom porque resolve problemas de dimensão n=10 em apenas 2.5 segundos'."

Em geral, a verificação da complexidade de um algoritmo é efetuada [43]:

i) sobre um modelo de máquina abstrata onde o algoritmo a ser avaliado é representado e sua complexidade é medida através do número de operações elementares (desse modelo) necessárias à execução do algoritmo.

ii) através do estabelecimento de características intrínsecas do problema. No caso de problemas envolvendo métodos numéricos, a avaliação é tratada como complexidade aritmética.

iii) sobre o programa correspondente ao algoritmo, isto é, mede se geralmente o tempo total de processamento, as operações sobre o programa-fonte ou então faz-se a contagem de operações elementares sobre o programa-objeto, por exemplo.

Destas três formas, constata-se atualmente um maior desenvolvimento em estudos sobre a forma ii). Recomendamos o trabalho de BRAGA [43] sobre avaliação de algoritmos em Programação Não-

Linear, onde poderão ser encontradas referências e maiores detalhes sobre este assunto. Em particular, lá são também abordadas as características, vantagens e inconvenientes próprios às três formas de avaliação apresentadas.

Esta breve introdução sobre avaliação e comparações de algoritmos foi aqui colocada no sentido de alertar o leitor quanto às dificuldades existentes e aos problemas em aberto na área de Análise de Algoritmos; ao mesmo tempo, nela baseados, desejamos justificar,a seguir,os critérios pelos quais avaliaremos e compararemos o desempenho dos algoritmos nos testes efetuados.

O princípio básico que iluminou nossa escolha consistiu em buscar-se necessariamente critérios para determinação da complexidade que sejam independentes, o máximo possível, do contexto da máquina (ambiente computacional usado). Neste sentido, procuramos imediatamente descartar o tempo total de processamento, gasto em determinado computador por um programa correspondente a um algoritmo testado, como fator indicativo da complexidade computacional. Este parâmetro é,em geral, muito influenciado por fatores que transcendem o algoritmo, envolvendo o ambiente computacional que está sendo usado como, por exemplo:

- processamento sequencial ou em paralelo;
- tempo compartilhado ou dedicado;
- tempo gasto em gerenciamento de memória ("overlays" e processo de paginação("page-in/page-out"));
- linguagem de codificação utilizada, no que tange à natureza do compilador usado, capaz ou não de gerar um programa-objeto otimizado.

Por outro lado, temos o número total de iterações efetuadas pelo algoritmo como um indicador comumente usado em métodos numéricos em que o esforço computacional (ou complexidade aritmética) é bem definido e constante para qualquer iteração. Porém, no contexto de otimização irrestrita de funções gerais , não temos uma "iteração" bem definida em termos de cálculos efetuados. Basta lembrar que o número de buscas lineares varia em geral.

O conjunto de indicadores por nós escolhido para avaliar o desempenho do MGC será composto por:

1 - o número total de avaliações da função-objetivo;
2 - o número total de avaliações do gradiente da função-objetivo;
3 - o esforço computacional global do sub-algoritmo de busca linear usado, isto é, conhecido o número t(k), representando o número de tentativas efetuadas na k-ésima iteração pelo sub-algoritmo de b.l. para convergir, obtemos, ao final do processo de otimização, o número t = t(1)+t(2)+...+t(L), sendo L o número total de iterações efetuadas pelo MGC;
4 - a razão entre o número de reinicializações e o número total de iterações efetuadas pelo MGC para convergir.

Inicialmente, os indicadores 1 e 2 foram escolhidos por refletirem diretamente uma característica intrínseca ao algoritmo, isto é, o número de vezes que foi a ele necessário calcular os valores da função-objetivo e do seu gradiente para alcançar o grau de otimização *a priori* fornecido.

O indicador 3 nos auxilia bastante na estimativa da da eficiência média do sub-algoritmo de busca linear escolhido. Com este indicador podemos, após usar diferentes algoritmos (uniformemente calibrados), concluir para um determinado problema, se tal busca linear é preferível ou não a outra. Além disto, também este indicador reflete uma característica inerente ao algoritmo, já que buscas lineares são executadas várias vezes durante um processo de otimização irrestrita.

Quanto ao indicador 4, tivemos o cuidado (já mencionado) em evitar, no nosso contexto, o uso isolado do número total de iterações. Nossa intenção, com este indicador, foi a de tentar "medir qualitativamente" o desempenho global de algoritmos de GC. Se, ao final do algoritmo, obtivermos valores próximos de 1 para o indicador 4 sugerido, podemos afirmar que, na maioria das vezes durante a otimização, a função-objetivo não apresentou uma aderência satisfatória com suas aproximações quadráticas locais (fato principalmente causado pela propagação de erros computacionais), obrigando assim o MGC a reinicializar constantemente o processo de descida; este fato nos leva a inferir que o MGC se comporta aproximadamente como o Método do Gradiente Puro, tendo uma baixa velocidade de convergência. Se, por outro lado, o indicador 4 apresentar valores pequenos, digamos abaixo de 0.5, pode-se concluir que, para esta função, o MGC apresenta uma velocidade de

convergência significativa, causada principalmente pela natureza da função. Além disto, este indicador(4) também mede uma característica intrínseca ao algoritmo, i.é, a sua capacidade (ou "esperteza") de, ao alcançar regiões de pouca aderência entre a função e sua aproximação quadrática local, se recuperar e reinicializar o processo, descendo através de direções mais convenientes.

V.2 - Resultados numéricos:

Conforme a estratégia dos testes (V.1.3) e a metodologia de comparação (V.1.4) propostas, apresentamos nesta seção os resultados obtidos nas experiências computacionais efetuadas. Tais resultados são apresentados da seguinte forma: seguem-se 12 tabelas, uma para cada função, contendo os resultados obtidos nas 36 minimizações desta função. Cada tabela é dividida funcional - mente em 3 grandes setores contendo respectivamente:

1) os parâmetros usados (definidos abaixo),
2) os resultados obtidos pelo Algoritmo 1, e
3) os resultados obtidos pelo Algoritmo 2.

O primeiro setor (parâmetros) é composto por 4 colunas, indicando:

(i) β_k - o coeficiente de conjugação usado: HS = Hestenes-Stiefel, PR = Polak-Ribière e FR = Fletcher-Reeves,

(ii) B.L. - a busca linear usada: W = Wolfe e G = Goldstein.

(iii) a e (iv) b - os valores fornecidos para as frações de inclinação usadas no algoritmo de busca linear.

Tanto o segundo como o terceiro setor possuem a mesma estrutura, composta por 4 colunas, indicando:

(i) NCALCF - o número total de avaliações da função-objetivo efetuadas na sua minimização,
(ii) NCALCG - o número total de avaliações do (anti)gradiente da função-objetivo efetuadas na sua minimização,
(iii) Nº B.L. - número total de tentativas efetuadas no processo de otimização pelo algoritmo de busca linear usado,
(iv) R/I - número de reinicializações (R) e o total de iterações (I).

Seguem-se as 12 tabelas.

FUNÇÃO: I - FUNÇÃO QUADRÁTICA NO $\mathbb{R}^2$ MUITO BEM-COMPORTADA											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	29	28	25	1 / 3	12	10	6	2 / 3
HS	W	0.4	0.6	29	28	25	1 / 3	12	10	6	2 / 3
HS	W	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3
HS	G	0.1	0.9	29	28	25	1 / 3	12	10	6	2 / 3
HS	G	0.4	0.6	29	28	25	1 / 3	12	10	6	2 / 3
HS	G	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3
PR	W	0.1	0.9	29	28	25	1 / 3	12	10	6	2 / 3
PR	W	0.4	0.6	29	28	25	1 / 3	12	10	6	2 / 3
PR	W	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3
PR	G	0.1	0.9	29	28	25	1 / 3	12	10	6	2 / 3
PR	G	0.4	0.6	29	28	25	1 / 3	12	10	6	2 / 3
PR	G	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3
FR	W	0.1	0.9	29	28	25	1 / 3	11	10	6	2 / 3
FR	W	0.4	0.6	29	28	25	1 / 3	11	10	6	2 / 3
FR	W	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3
FR	G	0.1	0.9	29	28	25	1 / 3	11	10	6	2 / 3
FR	G	0.4	0.6	29	28	25	1 / 3	11	10	6	2 / 3
FR	G	0.5	0.9	34	31	28	1 / 3	16	13	9	1 / 3

Tabela V.1

FUNÇÃO: II - FUNÇÃO QUADRÁTICA NO IR^2 MUITO MAL-COMPORTADA

PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	86	82	78	2 / 5	66	63	56	4 / 6
HS	W	0.4	0.6	86	82	78	2 / 5	54	51	45	3 / 5
HS	W	0.5	0.9	87	83	79	2 / 5	55	52	46	3 / 5
HS	G	0.1	0.9	86	82	78	2 / 5	66	63	56	4 / 6
HS	G	0.4	0.6	86	82	78	2 / 5	54	52	46	3 / 5
HS	G	0.5	0.9	87	83	79	2 / 5	55	52	46	3 / 5
PR	W	0.1	0.9	85	82	78	2 / 5	132	129	119	8 /10
PR	W	0.4	0.6	85	82	78	2 / 5	193	187	169	15/17
PR	W	0.5	0.9	107	103	97	3 / 7	125	120	108	9 /11
PR	G	0.1	0.9	85	82	78	2 / 5	132	129	119	8 /10
PR	G	0.4	0.6	85	82	78	2 / 5	192	188	171	14/16
PR	G	0.5	0.9	107	103	97	3 / 7	125	120	108	9 /11
FR	W	0.1	0.9	108	104	98	3 / 7	32227	32199	30200	1996/1998
FR	W	0.4	0.6	109	105	99	3 / 7	858	846	775	67/70
FR	W	0.5	0.9	110	106	100	3 / 7	901	864	787	74/76
FR	G	0.1	0.9	108	104	98	3 / 7	32227	32199	30200	1996/1998
FR	G	0.4	0.6	109	106	100	3 / 7	4505	4504	4198	302/305
FR	G	0.5	0.9	109	106	100	3 / 7	901	864	787	74/76

Tabela V.2

FUNÇÃO: III - FUNÇÃO QUADRÁTICA NO IR^{100} MUITO BEM-COMPORTADA											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_k	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
HS	W	0.4	0.6	15	12	8	0 / 3	19	15	10	2 / 4
HS	W	0.5	0.9	16	14	10	0 / 3	21	17	12	2 / 4
HS	G	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
HS	G	0.4	0.6	20	17	12	0 / 4	20	17	12	3 / 4
HS	G	0.5	0.9	16	14	10	0 / 3	21	17	12	2 / 4
PR	W	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
PR	W	0.4	0.6	15	12	8	0 / 3	18	15	10	2 / 4
PR	W	0.5	0.9	16	14	10	0 / 3	21	17	12	2 / 4
PR	G	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
PR	G	0.4	0.6	26	22	16	0 / 5	26	22	16	4 / 5
PR	G	0.5	0.9	16	14	10	0 / 3	21	17	12	2 / 4
FR	W	0.1	0.9	13	10	6	0 / 3	37	28	18	7 / 9
FR	W	0.4	0.6	15	12	8	0 / 3	43	34	24	7 / 9
FR	W	0.5	0.9	16	13	9	0 / 3	46	37	27	7 / 9
FR	G	0.1	0.9	13	10	6	0 / 3	37	28	18	7 / 9
FR	G	0.4	0.6	26	22	16	0 / 5	29	25	18	4 / 6
FR	G	0.5	0.9	16	13	9	0 / 3	46	37	27	7 / 9

Tabela V.3

FUNÇÃO: IV - FUNÇÃO QUADRÁTICA NO $\mathbb{R}^{200}$ MUITO BEM-COMPORTADA											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
HS	W	0.4	0.6	15	12	8	0 / 3	18	15	10	2 / 4
HS	W	0.5	0.9	16	13	9	0 / 3	21	17	12	2 / 4
HS	G	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
HS	G	0.4	0.6	20	17	12	0 / 2	20	17	12	3 / 4
HS	G	0.5	0.9	16	13	9	0 / 3	21	17	12	2 / 4
PR	W	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
PR	W	0.4	0.6	15	12	8	0 / 3	18	15	10	2 / 4
PR	W	0.5	0.9	16	13	9	0 / 3	21	17	12	2 / 4
PR	G	0.1	0.9	13	10	6	0 / 3	17	13	8	2 / 4
PR	G	0.4	0.6	26	22	16	0 / 5	26	22	16	4 / 5
PR	G	0.5	0.9	16	13	9	0 / 3	21	17	12	2 / 4
FR	W	0.1	0.9	13	10	6	0 / 3	37	28	18	7 / 9
FR	W	0.4	0.6	15	12	8	0 / 3	43	34	24	7 / 9
FR	W	0.5	0.9	16	13	9	0 / 3	46	37	27	7 / 9
FR	G	0.1	0.9	13	10	6	0 / 3	37	28	18	7 / 9
FR	G	0.4	0.6	26	22	16	0 / 5	29	25	18	4 / 6
FR	G	0.5	0.9	16	13	9	0 / 3	46	37	27	7 / 9

Tabela V.4

FUNÇÃO: V - FUNÇÃO QUADRÁTICA NO IR^{100} MUITO MAL-COMPORTADA											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	14288	14062	13089	9 / 980	14164	14016	13056	733/ 959
HS	W	0.4	0.6	15402	15150	14113	10 /1040	23422	23202	21665	1136/1536
HS	W	0.5	0.9	19558	19029	17740	12 /1292	26434	25969	24262	1289/1706
HS	G	0.1	0.9	14288	14062	13089	9 / 980	14164	14016	13056	733/ 959
HS	G	0.4	0.6	12076	12071	11295	7 / 780	19566	19564	18326	822/1237
HS	G	0.5	0.9	19558	19029	17740	12 /1292	26434	25969	24262	1289/1706
PR	W	0.1	0.9	13044	12806	11938	8 / 876	15212	15000	13984	789/1018
PR	W	0.4	0.6	18957	18654	17385	12 /1272	22985	22733	21216	1239/1516
PR	W	0.5	0.9	21106	20568	19182	13 /1390	24834	24333	22727	1325/1605
PR	G	0.1	0.9	13044	12806	11938	8 / 876	15212	15000	13984	789/1018
PR	G	0.4	0.6	14642	14623	13672	9 / 955	22991	22979	21500	1131/1478
PR	G	0.5	0.9	21106	20568	19182	13 /1390	24834	24333	22727	1325/1605
FR	W	0.1	0.9	22856	22789	21397	14 /1405	34482	34457	32471	1983/1985
FR	W	0.4	0.6	20025	19954	18757	12 /1205	43160	43144	40765	2376/2378
FR	W	0.5	0.9	22141	22093	20793	13 /1305	32010	31966	30203	1759/1762
FR	G	0.1	0.9	22856	22789	21397	14 /1405	34482	34457	32471	1983/1985
FR	G	0.4	0.6	17941	17939	16842	11 /1107	23677	23676	22352	1321/1323
FR	G	0.5	0.9	22141	22093	20793	13 /1305	32010	31966	30203	1759/1762

Tabela V.5

FUNÇÃO: VI - FUNÇÃO QUADRÁTICA NO $\mathbb{R}^{200}$ MUITO MAL-COMPORTADA											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	33316	32927	30916	10 /2020	51908	51446	48339	2406/3106
HS	W	0.4	0.6	35697	35268	33139	10 /2135	78026	77431	72888	3454/4542
HS	W	0.5	0.9	49861	48736	45820	14 /2923	88033	86668	81629	3806/5038
HS	G	0.1	0.9	33316	32927	30916	10 /2020	51908	51446	48339	2406/3106
HS	G	0.4	0.6	35002	34997	33003	10 /2003	55801	55797	52648	2162/3148
HS	G	0.5	0.9	49861	48736	45820	14 /2923	88033	86668	81629	3806/5038
PR	W	0.1	0.9	45781	45105	42440	13 /2705	58926	58230	54787	2686/3464
PR	W	0.4	0.6	60712	59971	56380	18 /3603	67021	66359	62455	3179/3903
PR	W	0.5	0.9	64259	62904	59158	18 /3755	74079	72795	68550	3502/4244
PR	G	0.1	0.9	45781	45105	42440	13 /2705	58926	58230	54787	2686/3464
PR	G	0.4	0.6	48410	48382	45589	14 /2804	69935	69909	65960	3013/3962
PR	G	0.5	0.9	64259	62904	59158	18 /3755	74079	72795	68550	3502/4244
FR	W	0.1	0.9	41055	40888	38667	11 /2267	43394	43266	40948	2321/2364
FR	W	0.4	0.6	41854	41738	39481	11 /2286	43954	43875	41558	2316/2354
FR	W	0.5	0.9	46383	46127	43631	12 /2528	67357	67064	63549	3510/3573
FR	G	0.1	0.9	41055	40888	38667	11 /2267	43394	43266	40948	2321/2364
FR	G	0.4	0.6	36447	36437	34450	10 /2002	43311	43292	41038	2253/2321
FR	G	0.5	0.9	46383	46127	43631	12 /2528	67357	67064	63549	3510/3573

Tabela V.6

FUNÇÃO: VII - FUNÇÃO NÃO-QUADRÁTICA CONVEXA NO $\mathbb{R}^{50}$											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_k	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
HS	W	0.4	0.6	380	365	363	0 /15	380	365	363	14 / 15
HS	W	0.5	0.9	423	405	402	0 /18	423	405	402	17 / 18
HS	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
HS	G	0.4	0.6	286	276	274	0 /11	286	276	274	10 / 11
HS	G	0.5	0.9	431	414	412	0 /17	431	414	412	16 / 17
PR	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
PR	W	0.4	0.6	138	123	107	0 /15	138	123	107	14 / 15
PR	W	0.5	0.9	142	124	105	0 /18	142	124	105	17 / 18
PR	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
PR	G	0.4	0.6	124	114	101	0 /12	124	114	101	11 / 12
PR	G	0.5	0.9	144	127	109	0 /17	144	127	109	16 / 17
FR	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
FR	W	0.4	0.6	128	113	97	0 /15	128	113	97	14 / 15
FR	W	0.5	0.9	137	119	100	0 /18	137	119	100	17 / 18
FR	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
FR	G	0.4	0.6	110	100	88	0 /11	110	100	88	10 / 11
FR	G	0.5	0.9	143	126	108	0 /17	143	126	108	16 / 17

Tabela V.7

FUNÇÃO: VIII - FUNÇÃO NÃO-QUADRÁTICA CONVEXA NO $\mathbb{R}^{200}$											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
HS	W	0.4	0.6	318	309	304	0 /16	318	309	304	15 / 16
HS	W	0.5	0.9	381	371	361	0 /19	381	371	361	18 / 19
HS	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
HS	G	0.4	0.6	213	207	205	0 / 9	213	207	205	8 / 9
HS	G	0.5	0.9	326	317	309	0 /16	326	317	309	15 / 16
PR	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
PR	W	0.4	0.6	67	58	47	0 /10	67	58	47	9 / 10
PR	W	0.5	0.9	68	57	44	0 /12	68	57	44	11 / 12
PR	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
PR	G	0.4	0.6	90	83	72	0 /10	90	83	72	9 / 10
PR	G	0.5	0.9	76	66	53	0 /12	76	66	53	11 / 12
FR	W	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
FR	W	0.4	0.6	71	62	50	0 /11	71	62	50	10 / 11
FR	W	0.5	0.9	66	55	42	0 /12	66	55	42	11 / 12
FR	G	0.1	0.9	5	5	3	0 / 1	5	5	3	0 / 1
FR	G	0.4	0.6	72	66	56	0 / 9	72	66	56	8 / 9
FR	G	0.5	0.9	67	57	45	0 /11	67	57	45	10 / 11

Tabela V.8

FUNÇÃO: IX - FUNÇÃO DE ROSENBROCK NO $\mathbb{R}^2$											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_k	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	308	299	280	12/24	242	235	211	23/24
HS	W	0.4	0.6	332	323	297	14/28	310	302	275	27/28
HS	W	0.5	0.9	342	332	304	14/29	388	375	339	33/36
HS	G	0.1	0.9	308	299	280	12/24	242	235	211	23/24
HS	G	0.4	0.6	472	459	439	15/30	268	260	235	24/25
HS	G	0.5	0.9	342	332	304	14/29	388	375	339	33/36
PR	W	0.1	0.9	547	530	500	20/40	224	212	192	18/21
PR	W	0.4	0.6	600	586	551	22/44	390	381	347	32/35
PR	W	0.5	0.9	516	495	469	17/34	401	386	352	32/35
PR	G	0.1	0.9	547	530	500	20/40	224	212	192	18/21
PR	G	0.4	0.6	417	404	387	13/26	316	308	280	27/28
PR	G	0.5	0.9	923	894	837	34/68	414	399	358	38/40
FR	W	0.1	0.9	635	613	586	21/42	1426	1410	1299	110/112
FR	W	0.4	0.6	834	810	765	30/60	500	488	441	44/47
FR	W	0.5	0.9	698	676	639	24/48	1198	1151	1042	107/109
FR	G	0.1	0.9	637	615	587	21/42	1426	1410	1299	110/112
FR	G	0.4	0.6	583	569	546	17/35	343	335	307	26/28
FR	G	0.5	0.9	822	801	762	25/51	1097	1063	960	102/103

Tabela V.9

FUNÇÃO: X - FUNÇÃO DE WOOD NO $\mathbb{R}^4$

PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	2545	2489	2348	44 / 179	2368	2326	2103	174/ 222
HS	W	0.4	0.6	2519	2463	2284	49 / 197	1711	1686	1532	125/ 154
HS	W	0.5	0.9	2124	2077	1911	43 / 175	1999	1947	1768	155/ 178
HS	G	0.1	0.9	1959	1911	1796	35 / 142	1564	1534	1385	122/ 148
HS	G	0.4	0.6	2949	2922	2740	53 / 215	2503	2488	2270	155/ 217
HS	G	0.5	0.9	2118	2073	1913	42 / 171	2068	2020	1836	157/ 184
PR	W	0.1	0.9	4543	4441	4180	81 / 325	2426	2372	2167	178/ 212
PR	W	0.4	0.6	3578	3488	3232	71 / 284	2321	2278	2073	171/ 207
PR	W	0.5	0.9	4806	4653	4279	99 / 399	2441	2371	2153	183/ 218
PR	G	0.1	0.9	4543	4441	4180	81 / 325	2688	2632	2389	205/ 245
PR	G	0.4	0.6	4949	4905	4605	88 / 355	2195	2178	1995	148/ 185
PR	G	0.5	0.9	4526	4368	4002	96 / 386	2495	2423	2202	187/ 220
FR	W	0.1	0.9	1651	1615	1525	28 / 114	10900	10879	10139	738/ 740
FR	W	0.4	0.6	3527	3471	3231	65 / 263	46510	46496	43791	2704/2708
FR	W	0.5	0.9	3849	3788	3530	68 / 273	12868	12837	12002	833/ 837
FR	G	0.1	0.9	2050	2008	1898	34 / 138	11152	11127	10365	760/ 762
FR	G	0.4	0.6	3314	3285	3086	57 / 231	7087	7077	6601	475/ 480
FR	G	0.5	0.9	3842	3749	3470	73 / 294	8231	8198	7648	546/ 552

Tabela V.10

FUNÇÃO: XI - FUNÇÃO EXTENDIDA DE POWELL NO $\mathbb{R}^{40}$											
PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_k	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	347	337	295	1 / 42	622	604	525	62 / 79
HS	W	0.4	0.6	426	416	362	1 / 54	339	330	287	36 / 42
HS	W	0.5	0.9	621	599	516	2 / 82	658	631	560	58 / 74
HS	G	0.1	0.9	347	337	295	1 / 42	653	635	555	65 / 81
HS	G	0.4	0.6	421	417	367	1 / 50	639	635	557	54 / 77
HS	G	0.5	0.9	621	599	516	2 / 82	658	631	560	58 / 74
PR	W	0.1	0.9	285	275	238	0 / 36	818	794	696	85 /100
PR	W	0.4	0.6	554	540	467	1 / 72	691	676	589	72 / 86
PR	W	0.5	0.9	1066	1027	896	3 /130	1185	1150	1005	126 /145
PR	G	0.1	0.9	285	275	238	0 / 36	818	794	696	85 /100
PR	G	0.4	0.6	372	367	321	1 / 45	504	502	442	46 / 59
PR	G	0.5	0.9	1066	1027	896	3 /130	1185	1150	1005	126 /145
FR	W	0.1	0.9	405	395	345	1 / 50	6621	6605	6021	581 /583
FR	W	0.4	0.6	757	734	641	2 / 93	2481	2465	2223	236 /241
FR	W	0.5	0.9	1312	1284	1137	3 /146	8343	8311	7630	672 /680
FR	G	0.1	0.9	359	348	304	1 / 44	6621	6605	6021	581 /583
FR	G	0.4	0.6	792	786	698	2 / 89	3130	3127	2836	285 /290
FR	G	0.5	0.9	1312	1284	1137	3 /146	8343	8311	7630	672 /680

Tabela V.11

FUNÇÃO: XII - FUNÇÃO EXTENDIDA DE POWELL NO IR^{200}

PARÂMETROS				ALGORITMO 1				ALGORITMO 2			
β_K	B.L.	a	b	NCALCF	NCALCG	Nº B.L.	R / I	NCALCF	NCALCG	Nº B.L.	R / I
HS	W	0.1	0.9	421	412	356	0 / 55	685	669	582	70/87
HS	W	0.4	0.6	458	447	388	0 / 59	591	578	502	64/75
HS	W	0.5	0.9	540	522	450	0 / 71	659	632	561	58/74
HS	G	0.1	0.9	421	412	356	0 / 55	685	669	582	70/87
HS	G	0.4	0.6	488	486	424	0 / 61	1138	1134	994	104/139
HS	G	0.5	0.9	540	522	450	0 / 71	659	632	561	58/74
PR	W	0.1	0.9	344	333	288	0 / 44	845	822	724	88/101
PR	W	0.4	0.6	448	437	378	0 / 58	909	889	773	98/115
PR	W	0.5	0.9	875	845	734	0 / 110	1682	1621	1424	157/197
PR	G	0.1	0.9	344	333	288	0 / 44	845	822	724	88/101
PR	G	0.4	0.6	465	461	402	0 / 58	767	765	673	73/91
PR	G	0.5	0.9	875	845	734	0 / 110	1682	1621	1424	157/197
FR	W	0.1	0.9	678	665	586	0 / 78	12942	12923	11821	1099/1101
FR	W	0.4	0.6	2075	2053	1839	1 / 213	1780	1766	1577	182/188
FR	W	0.5	0.9	1411	1366	1191	0 / 174	1875	1842	1632	199/209
FR	G	0.1	0.9	678	665	586	0 / 78	12942	12923	11821	1099/1101
FR	G	0.4	0.6	1293	1289	1131	0 / 157	5203	5200	4724	470/475
FR	G	0.5	0.9	1411	1366	1191	0 / 174	1875	1842	1632	199/209

Tabela V.12

V.3 - Discussão dos resultados e considerações finais:

Dividiremos nossa discussão conforme as 3 classes de funções descritas em V.1.2 . Posteriormente, faremos algumas considerações sobre certos fatores diretamente relacionados com a qualidade dos resultados obtidos.

Para funções quadráticas, em termos de comparação das versões do MGC, podemos esquematizar os melhores desempenhos da seguinte forma:

	Bem-comportadas	*Mal-comportadas*
Dimensões baixas:	*Alg.2*	*Alg.2*
Dimensões elevadas:	*Alg.1*	*Alg.1*

Tabela V.13

Para as funções quadráticas bem-comportadas, a propriedade de terminação quadrática foi, em geral, alcançada por ambos os algoritmos. Na verdade, isso só não aconteceu para a função I devido ao teste de parada referente à variação pontual entre iterações que, baseado na evolução pontual significativa entre x_2 e x_3, impediu que ambos os algoritmos fossem interrompidos neste último ponto, obrigando a realização de uma iteração adicional. Particularmente, este caso nos mostra a importância de um possível conhecimento prévio de regiões onde estejam localizados pontos de mínimo. Em todas as experiências, não fornecemos pontos de partida propositadamente favoráveis aos algoritmos, mas reutilizamos pontos já usados em testes disponíveis na literatura,([31] e [36], por exemplo) ou mesmo aleatoriamente escolhidos.

A aplicação do MGC a funções quadráticas mal-comportadas mostrou como estas afetam, na prática, não só a terminação quadrática, mas principalmente a velocidade de convergência destes algoritmos. Em todos esses casos, o MGC evoluiu iterativamente, como se estivesse otimizando uma função não-quadrática.

Em relação aos coeficientes de conjugação usados, notamos:

- em problemas mal-comportados, uma certa homogeneidade

entre as 3 fórmulas. Este fato atesta, na prática, a comprovação da teoria apresentada em II.4;

- em problemas mal-comportados, uma certa vantagem no uso da fórmula HS, principalmente em dimensões mais baixas. Tal vantagem é justificada pelo fato de os termos $d_k^t r_{k+1}$, considerado nulo por PR e FR, e $r_{k+1}^t r_k$, considerado nulo por FR, apresentarem valores constantemente diferentes de zero, degradando a utilização destas fórmulas neste tipo de problema.

Por sua vez, os algoritmos de busca linear apresentaram, em geral, comportamentos semelhantes, principalmente quando usamos os pares (0.1,0.9) e (0.4,0.6) para as frações de inclinação. Estes pares sempre propiciaram uma convergência global mais rápida a ambos os algoritmos de busca linear. O par (0.5,0.9), usada no sentido de forçar os sub-algoritmos a obter passos bem diferentes de zero, só aumentou o esforço computacional, sem contribuir para uma convergência mais rápida. Além disso, esta última dupla degradou mais sensivelmente o algoritmo de Goldstein do que o de Wolfe.

Para as funções convexas não-quadráticas, os resultados não nos permitem comparar efetivamente o desempenho das 2 versões do MGC, já que o Algoritmo 2(devido ao número de reinicializações verificadas) se portou identicamente ao Alg.1 (ver Tabs.V.7 e V.8). Em relação aos coeficientes de conjugação usados, os melhores resultados foram obtidos com FR, seguido por PR. Quanto à busca linear, notamos:

- o melhor desempenho de ambos os sub-algoritmos, quando usamos (0.1,0.9). Neste caso, W e G apresentaram a mesma eficiência;
- quando do uso de (0.4,0.6), G convergiu, em geral, um pouco mais rápido que W;
- quando do uso de (0.5,0.9), W convergiu, em geral, um pouco mais rápido que G.

Por termos utilizado apenas 2 funções desta classe, não podemos inferir considerações globalmente válidas para todos os casos. Porém, a constância dos resultados verificados nos leva a recomendar o uso das frações (0.1,0.9) em qualquer

dos 2 sub-algoritmos, isto é, deixando-os livres em um domínio relativamente grande. Tal estratégia foi responsável pelos melhores resultados verificados para essa classe de funções.

Para as funções não-quadráticas gerais, obtivemos os melhores resultados para:
- o Alg.2, em dimensões baixas,
- o Alg.1, em dimensões elevadas.

Novamente, devido à pequena quantidade de funções abordadas, relacionaremos apenas os resultados, sem inferir pragmatismos.

Em relação aos coeficientes de conjugação, verificamos uma alternância das 3 fórmulas nos melhores desempenhos dos 2 algoritmos, como pode ser visto abaixo:

Função	*Dimensão*	*Algoritmo 1*	*Algoritmo 2*
IX	*2*	HS	PR
X	*4*	FR	HS
XI	*40*	PR	HS
XII	*200*	PR	HS

Tabela V.14

Estatisticamente, a fórmula HS favoreceu um melhor desempenho do Algoritmo 2.

Quanto às buscas lineares, houve uma leve vantagem para o algoritmo de Wolfe e novamente as duplas (0.1,0.9) e (0.4,0.6), alternadamente, propiciaram uma convergência mais rápida a ambos os sub-algoritmos de busca linear.
Os melhores resultados podem ser assim resumidos:

Função	*Dimensão*	*Algoritmo 1*	*Algoritmo 2*
IX	*2*	W/G-(0.1,0.9)	W/G-(0.1,0.9)
X	*4*	W -(0.1,0.9)	G -(0.1,0.9)
XI	*40*	W/G-(0.1,0.9)	W -(0.4,0.6)
XII	*200*	W/G-(0.1,0.9)	W -(0.4,0.6)

Tabela V.15

Discutidos os resultados, colocamos agora as 3 principais conclusões globais, já esperadas de antemão e plenamente constatadas:

i) *Funções mal-comportadas afetam sensivelmente a velocidade de convergência dos MGC.*

ii) *Devido a diversos fatores que transcendem o MGC (como erros computacionais, natureza dos testes de parada e da busca linear usados), os resultados numéricos mostram que a teoria deve ser usada, na prática, como um instrumento apenas de previsão, fornecendo estimativas (nem sempre acuradas) para o desempenho destes algoritmos.*

iii) *Não existe uma versão do MGC globalmente mais eficiente. Qualquer problema deve ser abordado com o máximo de alternativas possíveis.*

Finalmente, convém evidenciarmos alguns fatores que, de alguma forma, influenciaram os resultados, a sua discussão e as nossas conclusões:

a. Os algoritmos de busca linear foram implementados de uma forma bem acadêmica, no sentido de nos permitir ter idéia do esforço máximo realizado para convergência. Ao inserirmos uma eventual interpolação quadrática no fim de cada busca linear e ao deixarmos ilimitado o número máximo de tentativas nesses sub-algoritmos, tentamos obrigá-los a efetuar b.l.inexatas bastante criteriosas (ver V.1.1). A definição de um limitante superior de tentativas deve ser considerada na prática, podendo fazer até com que um algoritmo de O.I. convirja mais rápido.

b. O conjunto de testes usado (ver V.1.1) foi derivado de uma análise feita sobre problemas muito bem-comportados, em IV.2. O uso desse conjunto em funções mal-comportadas, principalmente em dimensões elevadas, mostrou o quanto pode custar a um algoritmo de O.I. para satisfazê-lo (ver tabelas V.5 e V.6). Fica evidente a necessidade de se buscar conjuntos alternativos com testes para casos assim.

c. À luz dos fatores a e b, podemos considerar os resultados verificados como limitantes inferiores na qualidade do desempenho do MGC. Mudanças na implementação dos sub-algoritmos de busca linear e dos testes de convergência podem conduzir a melhorias bastante significativas.

d. Devido ao número estatisticamente pequeno de funções-testes não-quadráticas gerais, seria ousada a construção de inferências mais abrangentes sobre o desempenho do MGC para essa classe. As experiências realizadas serviram basicamente para ilustrar, com alguns casos reais, a teoria apresentada.

REFERÊNCIAS BIBLIOGRÁFICAS

"Knowledge is of two kinds. We know a subject ourselves or we know where we can find information upon it". [31]

(Samuel Johnson-1775)

[1] HESTENES,M.R. e STIEFEL,E., "Methods of Conjugate Gradients for Solving Linear Systems", J. of Research of the National Bureau of Standarts, vol.49, nº 6 , pp.409-436, (1952).

[2] FLETCHER,R. e REEVES,C.M., "Function minimization by conjugate gardients", Computer Journal, vol.7, nº2, pp.149-153, (1964).

[3] DANIEL,J.W., "The conjugate gradient method for linear and nonlinear operator equations", SIAM J. Num. An., vol.4, nº1, pp.10-26, (1967).

[4] KOWALIK,J. e OSBORNE,M.R., Methods for Unconstrained Optimization Problems, New York, American Elsevier Publishing Company, 1ª Edição, 1968.

[5] HESTENES,M.R., "Multiplier and Gradient methods", J.Optimization Theory Appl.(JOTA), vol.4, nº5, pp.303-320, (1969).

[6] POLAK,E. e RIBIÈRE,G., "Note sur la convergence de Methodes de directions conjuguées", Revue Française d'Inform. et Recherche Operationelle(RIRO), vol.16, pp.35-43, (1969).

[7] FLETCHER,R., "A review of methods for unconstrained optimization", em Optimization, editado por Fletcher,

R., London, Academic Press, 1ª Edição, 1969.

[8] RITTER,K., "A superlinearly convergent method for unconstrained minimization", em Nonlinear Programming, editado por Rosen, J.B., Mangasarian, O.L.e Ritter, K., London, Academic Press, 1ª edição, 1970.

[9] FRIED,I., "N-step conjugate gradient minimization scheme for nonquadratic functions", AIAA Journal, vol.9, nº11, pp.2286-2287, (1971).

[10] POLAK,E., Computational Methods in Optimization - a unified approach, New York, Academic Press, 1ª edição, 1971.

[11] CROWDER,H.P. e WOLFE,P., "Linear convergence of the Conjugate Gradient Method", IBM J.Res.Dev., vol. 16, pp.431-433, (1972).

[12] ORTEGA,J.M. e RHEINBOLDT,W.C., "A general convergence result for unconstrained minimization methods",SIAM J.Num.An., vol.9, nº1, pp.40-43, (1972).

[13] McCORMICK,G.P. e RITTER,K., "Methods of conjugate directions versus Quasi-Newton methods", Mathematical Programming, vol.3, pp.101-116, (1972).

[14] COHEN,A.I., "Rate of convergence of several conjugate gradient algorithms", SIAM J.Num.An., vol.9, nº 2, pp.248-259, (1972).

[15] KLESSIG,R. e POLAK,E., "Efficient implementations of the Polak-Ribière conjugate gradient algorithm", SIAM J.Control, vol.10, nº3, pp.524-549, (1972).

[16] MURRAY,W., Numerical methods for unconstrained optimization, London, Academic Press, 1ª Edição, 1972.

[17] BEALE,E.M.L., "A derivation of conjugate gradients", em Numerical Methods for Nonlinear Optimization, editado por Lootsma,F.A., pp.39-43, London e New York, Academic Press, 1972.

[18] KAWAMURA,K. e VOLTZ,R.A., "On the rate of convergence of the Conjugate Gradient Reset Method with Inaccurate Linear Minimizations", IEEE Transactions on Automatic Control, vol.AC-18, nº4, pp.360-366, (1973).

[19] LUENBERGER,D.G., Introduction to linear and nonlinear programming, Mento Park, California, Addison-Wesley, 1ª Edição, 1973.

[20] McCORMICK,G.P. e RITTER,K., "Alternative proofs of the convergence properties of the CG Method", JOTA, vol. 13, nº5, pp.497-518, (1974).

[21] HESTENES,M.R., Optimization Theory - the finite dimensional case, New York, John Wiley & Sons, 1ª Edição,1975.

[22] POWELL,M.J.D., "Convergence properties of a class of minimization algorithms", em Nonlinear Programming 2, editado por Mangasarian,O.L., Meyer,R.R. e Robinson,S.M., London, Academic Press, 1ª Edição, 1975.

[23] LENARD,M.L., "Convergence conditions for restarted conjugate gradient methods with inaccurate line searches", Mathematical Programming, vol.10, pp. 32-51, (1976).

[24] MEYER,R.R., "On the convergence of algorithms with restart", SIAM J.Num.Anal., vol.13, nº5, pp.696-703, (1976).

[25] AVRIEL,M., Nonlinear Programming: Analysis and Methods, New Jersey, Prentice-Hall, 1ª Edição, 1976.

[26] POWELL,M.J.D., "Some convergence properties of the conjugate gradient method", Mathematical Programming, vol.11, pp.42-49, (1976).

[27] POWELL,M.J.D., "Restart procedures for the conjugate gradient method", Mathematical Programming, vol.12, pp.241-254, (1977).

[28] THOMPSON,J.R., "Examples of non-convergence of conjugate descent algorithms with exact line-searches",Mathematical Programming, vol.12, pp.356-360, (1977).

[29] LENARD,M.L., "Accelerated conjugate direction methods for unconstrained optimization", JOTA, vol.25, nº1, pp.11-31, (1978).

[30] PSHENICHNY,B.N. e DANILIN,Yu.M., Numerical methods in extremal problems, Moscow, MIR Publishers, Edição em Inglês, 1978 (1ª Edição em russo, 1975).

[31] SHANNO,D.F., "Conjugate gradient methods with inexact searches", Mathematics of Operations Research, vol.3, nº3, pp.244-256, (1978).

[32] SHANNO,D.F., "On the convergence of a new conjugate gradient algorithm", SIAM J.Num.Anal., vol.15, nº6, pp.1247-1257, (1978).

[33] BAZARAA,M.S. e SHETTY,C.M., Nonlinear Programming - Theory and Algorithms, New York, John Wiley & Sons, 1ª Edição, 1979.

[34] LEMARECHAL,C. e PIRONNEAU,O., Nonlinear optimization methods and applications to some control problems in Physics, Rio de Janeiro, Instituto de Matemática da UFRJ, 1ª Edição, 1979.

[35] BOLAND,W.R., KAMGNIA,E.R. e KOWALIK,J.S., "A conjugate-gra

dient optimization method invariant to nonlinear scaling", JOTA, vol.27, nº2, pp.221-230, (1979).

[36] BOLAND, W. R. e KOWALIK, J. S.,"Extended Conjugate-Gradient Methods with Restarts", JOTA, vol.28, nº1, pp.1-9, (1979).

[37] BOLAND, W. R. e KOWALIK, J. S., "Some Computational Advances in unconstrained optimization", Applied Mathematics and Computation, vol.7, pp.55-69, (1980).

[38] HESTENES, M. R., Conjugate Direction Methods in Optimization, da série "Applications of Mathematics" (vol.12), editada por Balakrishnam, A. V., New York, Springer Verlag, 1ª Edição, 1980.

[39] COHEN, A. I., "Stepsize Analysis for Descent Methods", JOTA, vol.33, nº2, pp.187-205,(1981).

[40] GILL, P. E. , MURRAY, W. e WRIGHT, M. H., Practical Optimization, New York, Academic Press, 1ª Edição, 1981.

[41] SCHWEPEL, H-P, Numerical Optimization of Computer Models, Chichester, Great Britain, John Wiley & Sons, Edição em Inglês, 1981. (1ª Edição em alemão,1977).

[42] NAZARETH, L. e NOCEDAL, J., "Conjugate Direction Methods with variable storage", Mathematical Programming, vol.23, pp.326-340, (1982).

[43] BRAGA, L.P.V., "Avaliação e Comparação de Algoritmos em Programação Não-Linear", Anais do Simpósio Brasileiro de Pesquisa Operacional, 12.,pp.105-126, SOBRAPO, São Paulo, 1979.

APÊNDICE A: LISTAGEM DO PROGRAMA UTILIZADO

O programa construído contém os 2 algoritmos de GC usados nas experiências computacionais do capítulo V e é composto pelas seguintes rotinas:

1.MAIN - Módulo principal do programa, tem como principais funções:

- ler os dados de entrada (ver na própria listagem),
- obter o ponto inicial, ativando a rotina PTINIC,
- imprimir os dados lidos e gerados,
- selecionar o algoritmo de GC a ser utilizado, conforme opção do usuário, através da ativação de GCALG1 (Algoritmo 1) ou de GCALG2 (Algoritmo 2),
- e, finalmente, imprimir os resultados obtidos.

2.PTINIC - Seleciona o ponto inicial para o algoritmo de GC conforme a função-teste especificada.

3.VALFR - Dado um ponto no domínio da função-teste, VALFR avalia o valor da função e/ou do seu antigradiente nesse ponto.

4.GCALG1 - Rotina de otimização. Contém o Algoritmo 1 de GC definido para os testes.

5.GCALG2 - Rotina de otimização. Contém o Algoritmo 2 de GC definido para os testes.

6.TESTA - Ativada a cada iteração do MGC por uma das rotinas de otimização acima, TESTA verifica se os testes de convergência foram satisfeitos ou não.

7.GLDWLF - Rotina de busca linear. Ativada a cada iteração do MGC por uma das rotinas de otimização acima, GLDWLF resolve o sub-problema de minimização unidimensional.

8.BETAK - Calcula o valor do coeficiente de conjugação β_k, utilizando uma entre as expressões HS, PR e FR , conforme especificado pelo usuário.

9.NOREUC - Dado um vetor n-dimensional, essa rotina calcula o valor de sua norma euclideana.

Finalmente, a estrutura geral do programa pode ser resumida no fluxograma abaixo:

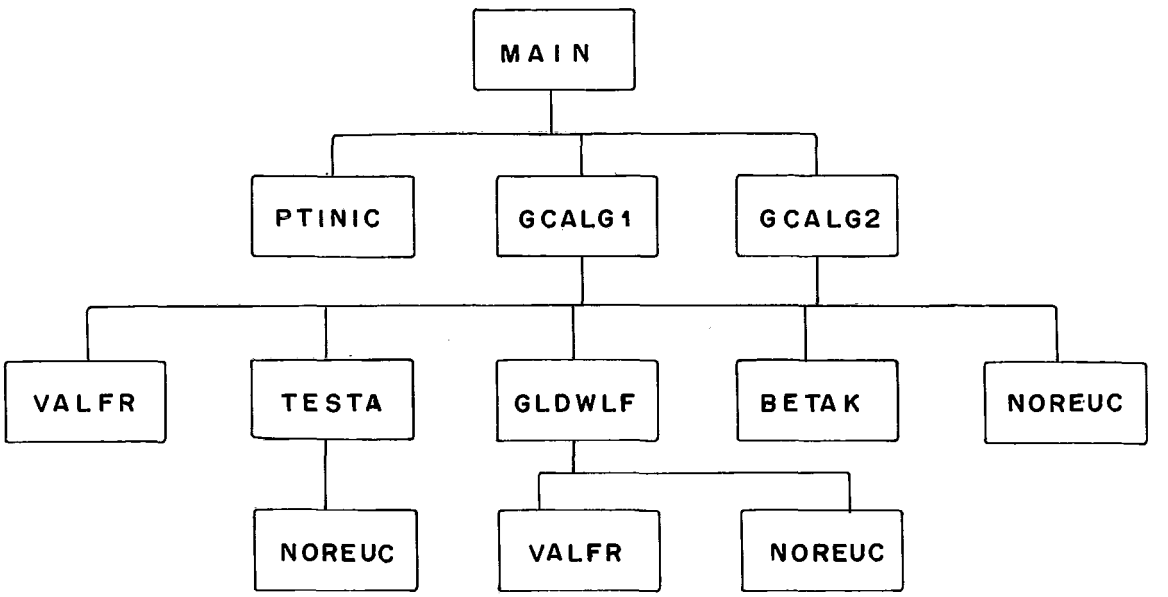


Fig. A.1

Segue-se a listagem completa do programa.

```
C=======+=========+=========+=========+=========+=========+=========+=========
C                                                                              =
C                                                                              =
C  A P L I C A C A O   D O   M E T O D O   D E   G R A D I E N T E S           =
C  ---------------------------------------------------------------             =
C       C O N J U G A D O S   'A   M I N I M I Z A C A O   D E                 =
C       -----------------------------------------------------                  =
C             F U N C O E S   C O N T I N U A M E N T E                        =
C             -----------------------------------------                        =
C                   D I F E R E N C I A V E I S                                =
C                   ---------------------------                                =
C                                                                              =
C                                                                              =
C      AUTORES: Lindalva Pereira Torres Borges (COPPE/UFRJ)                    =
C               Paulo Roberto Torres Borges    (COPPE/UFRJ)                    =
C                                                                              =
C      DATA   : julho de 1982                                                  =
C                                                                              =
C                 ===================================          =               =
C                 =                                                =           =
C=================  P R O G R A M A   P R I N C I P A L  ==================
C                 =                                                =           =
C                 ===================================          =               =
C                                                                              =
C     DEFINICAO DOS PARAMETROS DE ENTRADA:                                     =
C     ------------------------------------                                     =
C                                                                              =
C    ALFA ,                                                                    =
C    BETA   - Fracoes de inclinacao para a busca linear                        =
C                                                                              =
C    EPS    - Tolerancia para os testes de convergencia do MGC                 =
C                                                                              =
C    IMP    - Intervalo para impressao de resultados (em iteracoes)            =
C                                                                              =
C    ITIPOF - Codigo indicando a funcao-teste a ser minimizada                 =
C                                                                              =
C    IXBETA - Codigo associado 'a formula de conjugacao usada                  =
C             =1 - HESTENES-STIEFEL                                            =
C             =2 - POLAK-RIBIERE                                               =
C             =3 - FLETCHER-REEVES                                             =
C                                                                              =
C    IXGC   - Codigo associado ao algoritmo de GC usado                        =
C             =1 - Tradicional,reinicializado a cada N iteracoes               =
C             =2 - o anterior+tecnica de BEALE+criterios de POWELL             =
C                                                                              =
C    IXGERA - Codigo associado ao metodo usado para busca linear               =
C             =1 - WOLFE                                                       =
C             =2 - GOLDSTEIN                                                   =
C                                                                              =
C    N      - Dimensao do problema (numero de variaveis)                       =
C                                                                              =
C    NTRACE - Codigo indicando se o usuario deseja a impressao a               =
C             cada iteracao de XK,RK e F(XK):                                  =
C             =0 - nao havera' "TRACE"                                         =
C             =1 - havera' "TRACE"                                             =
C                                                                              =
C    NMAX   - Numero maximo de iteracoes permitido ao MGC                      =
C                                                                              =
C    XINIC  - Vetor contendo o ponto inicial para o MGC                        =
C
```

```
C
C     DEFINICAO DOS PARAMETROS DE SAIDA:
C     ----------------------------------
C
C    XFINAL - Vetor contendo a solucao obtida
C
C    VALFUN - Valor da funcao na solucao obtida
C
C    GRDFIN - Vetor-residual  na solucao obtida (Antigradiente)
C
C    ENORG  - Valor da norma euclideana de GRDFIN
C
C    NITER  - Numero de iteracoes necessarias p/ se obter a solucao
C
C    NREST  - Numero de iteracoes aonde foi necessario
C             reinicializar o MGC
C
C    NRESBL - Numero de vezes aonde foi necessario reinicializar
C             a busca linear
C
C    NTOTBL - Numero total de sub-iteracoes efetuadas pelo
C             algoritmo de busca linear usado
C
C    NCALCF - Numero de avaliacoes da funcao-objetivo pelo MGC
C
C    NCALCG - Numero de avaliacoes de antigradiente pelo MGC
C
C    LITEX  - Codigo 0/1 indicando se o MGC excedeu(1) ou nao(0)
C             o limite de iteracoes "NMAX" fornecido.
C
C-----------------------------------------------------------------
C
C
C
C
      COMMON /DADOS1/ EPS    , NMAX , N , IMP , NTRACE
     .       /DADOS2/ XINIC(500)
     .       /CONST/  IXBETA , ITIPOF , IXGERA , MAPABL   , LITEX
     .       /BUSCAL/ ALFA   , BETA   , PASSO  , FUNC     , ICOUNT
     .       /RESULT/ ENORG  , VALFUN , XFINAL(500)       ,
     .                GRDFIN(500)     , NITER  ,NREST,NRESBL,NTOTBL
     .       /ESTADO/ ICONBL
     .       /AVALIA/ NCALCF , NCALCG
C
      DOUBLE PRECISION ENORG  , FUNC   , GRDFIN , PASSO ,
     .                 XINIC  , XFINAL , VALFUN , EPS
      REAL MSG(17)
      DATA MSG/'WOLF','E   ','GOLD','STEI','N   ','HEST','ENES','-STI',
     .         'EFEL','POLA','K-RI','BIER','E   ','FLET','CHER','-REE',
     .         'VES '/
C
C
C     ---------------------------------------------------------
C                        Leitura dos parametros
C     ---------------------------------------------------------
      READ(5,1010) N,NMAX,IXBETA,IXGERA,ALFA,BETA,MAPABL,ITIPOF,IMP
      READ(5,1020) EPS,NTRACE,IXGC
      IF (IXGC.NE.1.AND.IXGC.NE.2) IXGC = 2
C
C
```

```
C     ----+---------+---------+---------+---------+---------+--------
C         Obtencao do ponto inicial conforme a funcao-teste
C     ------------------------------------------------------------------
      CALL PTINIC
C
C     ------------------------------------------------------------------
C                    Impressao dos dados lidos
C     ------------------------------------------------------------------
      IXIN  = 1
      IXFIN = 2
      IF ( IXGERA.EQ.1 ) GO TO 20
           IXIN  = 3
           IXFIN = 5
   20 WRITE(6,1030)ITIPOF, N, NMAX, EPS, (MSG(I),I = IXIN,IXFIN)
           IXIN  = 6
           IXFIN = 9
      IF ( IXBETA.NE.2 ) GO TO 40
           IXIN  = 10
           IXFIN = 13
      GO TO 60
   40 IF ( IXBETA.NE.3 ) GO TO 60
           IXIN  = 14
           IXFIN = 17
   60 WRITE(6,1040) ALFA , BETA , (MSG(I),I = IXIN , IXFIN) , IXGC
      WRITE(6,1050) (XINIC(I) , I = 1 , N)
C
C     ------------------------------------------------------------------
C                  Minimizacao do problema fornecido
C     ------------------------------------------------------------------
      IF (IXGC.EQ.1) CALL GCALG1
      IF (IXGC.EQ.2) CALL GCALG2
C
C     ------------------------------------------------------------------
C                       Impressao dos resultados
C     ------------------------------------------------------------------
      WRITE(8,1060)
C
C     Verifica se houve falha na busca linear
C     ---------------------------------------
      IF (ICONBL.NE.0) GO TO 80
      WRITE(8,1090)
C
C     Verifica se o numero maximo de iteracoes foi excedido
C     -----------------------------------------------------
   80 IF (LITEX.EQ.0) GO TO 100
      WRITE(8,2000)
  100 WRITE(8,1070) NITER , ((I,XFINAL(I)), I = 1 , N)
      WRITE(8,1080) VALFUN,ENORG,NCALCF,NCALCG,NTOTBL,NREST,NRESBL
C
      STOP
C
C
C
C     ------------------------------------------------------------------
C                          Formatos usados
C     ------------------------------------------------------------------
 1010 FORMAT(/,T3,I3,T12,I4,T24,I1,T33,I1,T40,F4.2,
     .       T50,F4.2,T62,I1,T71,I2,T78,I3)
 1020 FORMAT(/,T5,D7.1,T19,I1,T30,I1)
 1030 FORMAT('1',1X,50('*'),' G R A C O -  RELATORIO FINAL ',
```

```
     . 49('*'),/,'0',40X,'*** DADOS DO PROBLEMA ***',/,'0',
     . 10X,'CODIGO DA FUNCAO : ',I2,/,'0',
     . 10X,'DIMENSAO DO ESPACO-SOLUCAO : ',I4,/,'0',
     . 10X,'LIMITE FORNECIDO DE ITERACOES : ',I4,/,'0',
     . 10X,'TOLERANCIA PARA OS TESTES DE CONVERGENCIA : ',D7.1,/,'0'
     .,10X,'BUSCA LINEAR A SER USADA : ',3A4)
1040 FORMAT('0',10X,'FRACOES DE INCLINACAO FORNECIDAS PARA A BUSCA'
     .,' LINEAR : ',F4.2,5X,F4.2,/,'0',10X,'COEFICIENTE DE CONJUGA',
     .'CAO A SER USADO : ',4A4,/,'0',10x,
     .'ALGORITMO DE GC USADO (1=TRADICIONAL,2=BEALE+POWELL):',I1,
     ./,'0',10X,'PONTO INICIAL FORNECIDO : ')
1050 FORMAT(T18,D12.3,T31,D12.3,T44,D12.3,T57,D12.3)
1060 FORMAT('1',////,20X,'*** RESULTADOS DO PROBLEMA ***',///)
1070 FORMAT(1X,'SOLUCAO OBTIDA APOS ',I5,' ITERACOES',//,
     .       (1X,'X(',I4,') ',T10,'= ',D20.10))
1080 FORMAT(//,1X,'VALOR FINAL DA FUNCAO : ',D20.10,/,1X,
     .             'NORMA DO GRADIENTE : ',D20.10,/,1X,
     .             'TOTAL DE AVALIACOES DA FUNCAO : ',I6,/,1X,
     .             'TOTAL DE AVALIACOES DO GRADIENTE : ',I6,/,1X,
     .             'TOTAL DE TENTATIVAS NA B.L. : ',I6,/,1X,
     .             'NUMERO DE RESTARTS DO MGC : ',I5,/,1X,
     .             'NUMERO DE BUSCAS LINEARES DUPLAS : ',I6,/,1X,
     .//////,1X,48('*'),'  GRACO/2  -  FIM DE RELATORIO  ',48('*'))
1090 FORMAT(5X,'AVISO :  MINIMO NAO ENCONTRADO NA BUSCA LINEAR',//)
2000 FORMAT(5X,'AVISO :   PROGRAMA INTERROMPIDO POR NAO ',
     .'ALCANCAR UMA SOLUCAO NO NUMERO MAXIMO DE ITERACOES',///)
     END
```

```
      SUBROUTINE PTINIC
C     ---------- ------
C
C     ------------------------------------------------------
C     Fornece o ponto inicial 'XINIC' , conforme a funcao ,
C     para o MGC
C     ------------------------------------------------------
C
      COMMON /DADOS1/ EPS    , NMAX , N , IMP , NTRACE
     1       /DADOS2/ XINIC(500)
     2       /CONST/ , IXBETA , ITIPOF , IAGERA , MAPABL , LITEX
      DOUBLE PRECISION XINIC , EPS
C
C
      GO TO (10,10,20,20,30,30,40,40,50,60,70,70) , ITIPOF
C
C     ------------------------------------------------------
C     (I)   - Funcao quadratica no R2 (muito bem-comportada)
C     (II)  - Funcao quadratica no R2 (muito mal-comportada)
C     ------------------------------------------------------
   10 XINIC(1)    = 1000.0D0
      XINIC(2)    = 1.0D0
      GO TO 200
C
C     ------------------------------------------------------------
C     (III)-(IV) - Funcao quadratica no RN (muito bem-comportada)
C     ------------------------------------------------------------
   20 LIM      = N - 1
      DO 22 J = 1 , LIM , 2
         XINIC(J)   =  5.0D0
         XINIC(J+1) = -5.0D0
   22 CONTINUE
      GO TO 200
C
C     ------------------------------------------------------------
C     (V)-(VI) - Funcao quadratica no RN (muito mal-comportada)
C     ------------------------------------------------------------
   30 DO 32 J = 1 , N
         XINIC(J)   =  1.0D0
   32 CONTINUE
      GO TO 200
C
C     ------------------------------------------------------
C     (VII)-(VIII) - Funcao nao-quadratica convexa no RN
C     ------------------------------------------------------
   40 DO 42 J = 1 , N
         XINIC(J)   =  0.0D0
   42 CONTINUE
      GO TO 200
C
C     ------------------------------------------------
C     (IX) - Funcao nao-quadratica no R2 (ROSENBROCK)
C     ------------------------------------------------
   50 XINIC(1)    = -1.2D0
      DO 52  J    =  2 , N
         XINIC(J) =  1.0D0
   52 CONTINUE
      GO TO 200
C
```

```
C     ------------------------------------------
C     (X) - Funcao nao-quadratica no R4 (WOOD)
C     ------------------------------------------
      60 LIM = N - 1
         DO 62 J = 1 , LIM , 2
               XINIC(J)   = -3.0D0
               XINIC(J+1) = -1.0D0
      62 CONTINUE
         GO TO 200
C     ------------------------------------------------------
C     (XI)-(XII) - Funcao nao-quadratica no RN (POWELL)
C     ------------------------------------------------------
      70 DO 72 I    = 1 , N
         XINIC(I)   =  3.0D0
         XINIC(I+1) = -1.0D0
         XINIC(I+2) =  0.0D0
         XINIC(I+3) =  3.0D0
      72 CONTINUE
         GO TO 200
C
     200 RETURN
         END
```

```
      SUBROUTINE VALER (ITIPOF,N,X,F,R,I)
C     ---------- -----
C
C ================================================================
C                                                                =
C     OBJETIVO: Estimar o valor da funcao-objetivo e/ou do       =
C     --------  vetor-antigradiente em um ponto 'X' dado.        =
C                                                                =
C     ENTRADAS :                                                 =
C     --------                                                   =
C        ITIPOF    - Codigo indicando qual funcao-teste deve     =
C                    ser considerada.                            =
C                                                                =
C        N         - Dimensao do problema                        =
C                                                                =
C        X(*)      - Ponto onde sera' estimada a funcao e/ou o   =
C                    antigradiente.                              =
C                                                                =
C        I         - Codigo indicando se nesta ativacao sera(ao) =
C                    calculado(os):                              =
C                    =1 , somente o valor da funcao              =
C                    =2 , o valor da funcao e do antigradiente   =
C                    =3 , somente o antigradiente                =
C                                                                =
C                                                                =
C     SAIDA :                                                    =
C     -----                                                      =
C        F         - Valor da funcao estimado em X               =
C                                                                =
C        R(*)      - Vetor-antigradiente estimado em X           =
C                                                                =
C        NCALCF    - Total atualizado de avaliacoes da funcao    =
C                    necessarias 'a sua minimizacao.             =
C                                                                =
C        NCALCG    - Total atualizado de avaliacoes do antigra-  =
C                    diente necessarias 'a minimizacao da funcao=
C                                                                =
C ================================================================
C
      COMMON /AVALIA/ NCALCF , NCALCG
      DOUBLE PRECISION F , X(500) , R(500) , A(500)
      INTEGER I , ITIPOF , N
C
      F        = 0.0D0
C     ------------------------------------------------------
C     Se so' se desejar o calculo do antigradiente , desvia.
C     ------------------------------------------------------
      IF (I.EQ.3) GO TO 250
C
C
C
C     ---------------------------------
C     Selecao da funcao-objetivo a ser usada
C     ---------------------------------
      GO TO(10,20,30,30,40,40,50,50,60,70,80,80) , ITIPOF
C
C
C
C
```

```
C     ----+---------+---------+---------+---------+--------
C     (I) Funcao quadratica no R2 (muito bem-comportada)
C     ----------------------------------------------------
   10 F = X(1)**2 + 1.2D0*X(2)**2
      GO TO 200
C
C     ----------------------------------------------------
C     (II) Funcao quadratica no R2 (muito mal-comportada)
C     ----------------------------------------------------
   20 F = X(1)**2 + 1000.0D0*X(2)**2
      GO TO 200
C
C     ------------------------------------------------------------
C     (III)-(IV) Funcao quadratica no RN (muito bem-comportada)
C     ------------------------------------------------------------
   30 LIM        = N - 1
C
      DO 32 J    = 1,LIM,2
           A(J)  = 1.0D0
           A(J+1)= 1.5D0
   32 CONTINUE
C
      DO 34 J    = 1,N
           F     = F + A(J)*X(J)**2
   34 CONTINUE
      GO TO 200
C
C     ------------------------------------------------------
C     (V)-(VI) Fcao.quadratica no RN (muito mal-comportada)
C                 (Funcao de Oren-Spedicato alongada)
C     ------------------------------------------------------
   40 DO 42 J = 1 , N
           F = F + DFLOAT(J**2)*X(J)**2
   42 CONTINUE
      GO TO 200
C
C     ------------------------------------------------
C     (VII)-(VIII) Funcao nao-quadratica convexa no RN
C     ------------------------------------------------
   50 DO 52 J = 1 , N
           F = F + (X(J)-1.0D0)**4
   52 CONTINUE
      GO TO 200
C
C
C     ----------------------------------------
C     (IX) Funcao nao-quadratica no R2 (ROSENBROCK)
C     ----------------------------------------
   60 F = 100.0D0*(X(2)-X(1)**2)**2 + (1.0D0-X(1))**2
      GO TO 200
C
C     ------------------------------------
C     (X) Funcao nao-quadratica no R4 (WOOD)
C     ------------------------------------
   70 F = 100.0D0*(X(2)-X(1)**2)**2 + (1.0D0-X(1))**2    +
     .     90.0D0*(X(3)**2-X(4))**2 + (1.0D0-X(3))**2    +
     .     10.1D0*( (X(2)-1.0D0)**2 + (X(4)-1.0D0)**2 ) +
     .     19.8D0 * (X(2)-1.0D0) * (X(4)-1.0D0)
      GO TO 200
C
```

```
C
C     ------------------------------------------------------------
C     (XI)-(XII) Funcao nao-quadratica no RN (Fcao. de Powell)
C     ------------------------------------------------------------
   80 DO  82 J = 1 , N , 4
         F  = F + (X(J)+10.0D0*X(J+1))**2 + 5.0D0*(X(J+2)-X(J+3))**2
     .      +  (X(J+1)-2.0D0*X(J+2))**4 + 10.0D0*(X(J)-X(J+3))**4
   82 CONTINUE
      GO TO 200
C
C
C
  200 NCALCF = NCALCF + 1
C
C
C     ------------------------------------------------------------
C     Se so' o valor de F deveria ser calculado , retorna.
C     ------------------------------------------------------------
      IF(I.EQ.1) GO TO 600
C
C
C
  250 CONTINUE
C
C     ----------------------------------------
C     Selecao do antigradiente a ser calculado
C     ----------------------------------------
C
      GO TO(310,320,330,330,340,340,350,350,360,370,380,380) ,ITIPOF
C
C     ------------------------------------------------------
C     (I)  Funcao quadratica no R2 (muito bem-comportada)
C     ------------------------------------------------------
  310 R(1)  = -2.0D0*X(1)
      R(2)  = -2.4D0*X(2)
      GO TO 500
C
C     ------------------------------------------------------
C     (II)  Funcao quadratica no R2 (muito mal-comportada)
C     ------------------------------------------------------
  320 R(1)  = -2.0D0*X(1)
      R(2)  = -2000.0D0*X(2)
      GO TO 500
C
C     ------------------------------------------------------------
C     (III)-(IV) Funcao quadratica no RN (muito bem-comportada)
C     ------------------------------------------------------------
  330 IF (I.EQ.2) GO TO 334
C
          LIM         = N - 1
          DO 332 J    = 1,LIM,2
              A(J)    = 1.0D0
              A(J+1)  = 1.5D0
  332     CONTINUE
C
  334 CONTINUE
      DO 336 J     = 1,N
            R(J)  = -2.0D0*A(J)*X(J)
  336 CONTINUE
      GO TO 500
```

```
C
C     ------------------------------------------------------
C     (V)-(VI) Fcao.quadratica no RN (muito mal-comportada)
C                (Funcao de Oren-Spedicato alongada)
C     ------------------------------------------------------
  340 DO 342  J  = 1 , N
          R(J) = -2.0D0*DFLOAT(J**2)*X(J)
  342 CONTINUE
      GO TO 500
C
C     ------------------------------------------------
C     (VII)-(VIII) Funcao nao-quadratica convexa no RN
C     ------------------------------------------------
  350 DO 352  J  = 1 , N
          R(J) = -4.0D0*(X(J)-1.0D0)**3
  352 CONTINUE
      GO TO 500
C
C     ------------------------------------------
C     (IX) Funcao nao-quadratica no R2 (ROSENBROCK)
C     ------------------------------------------
  360 R(1) = 400.0D0*(X(2)-X(1)**2)*X(1) + 2.0D0*(1.0D0-X(1))
      R(2) =-200.0D0*(X(2)-X(1)**2)
      GO TO 500
C
C     ------------------------------------
C     (X) Funcao nao-quadratica no R4 (WOOD)
C     ------------------------------------
  370 R(1) = 400.0D0*(X(2)-X(1)**2)*X(1) + 2.0D0*(1.0D0-X(1))
      R(2) =-200.0D0*(X(2)-X(1)**2) - 20.2D0*(X(2)-1.0D0) -
     .        19.8D0*(X(4)-1.0D0)
      R(3) =-360.0D0*(X(3)**2-X(4))*X(3) + 2.0D0*(1.0D0-X(3))
      R(4) = 180.0D0*(X(3)**2-X(4)) - 20.2D0*(X(4)-1.0D0) -
     .        19.8D0*(X(2)-1.0D0)
      GO TO 500
C
C     ------------------------------------------------------
C     (XI)-(XII) Funcao nao-quadratica no RN (Funcao de Powell)
C     ------------------------------------------------------
  380 DO 382 J = 1 , N , 4
       R(J)   =-2.0D0*(X(J)+10.0D0*X(J+1)) - 40.0D0*(X(J)-X(J+3))**3
       R(J+1) = -20.0D0*(X(J)+10.0D0*X(J+1)) - 4.0D0*(X(J+1)-2.0D0*
     .                                                  X(J+2))**3
       R(J+2) =-10.0D0*(X(J+2)-X(J+3))+8.0D0*(X(J+1)-2.0D0*X(J+2))**3
       R(J+3) = 10.0D0*(X(J+2)-X(J+3)) + 40.0D0*(X(J)-X(J+3))**3
  382 CONTINUE
      GO TO 500
C
  500 NCALCG = NCALCG + 1
  600 RETURN
      END
```

```
      SUBROUTINE GCALG1
C     ---------- ------  (Algoritmo 1)
C
      COMMON /DADOS1/ EPS    , NMAX , N , IMP   , NTRACE
     .       /DADOS2/ XK(500)
     .       /CONST / IXBETA , ITIPOF , IXGERA , MAPABL,LITEX
     .       /BUSCA1/ ALFA   , BETA   , PASSO  , FUNC  ,ICONT
     .       /RESULT/ ENORG  , VALFUN , XFINAL(500)       ,
     .                GRDFIN(500) , NITER ,NREST,NRESBL,NTOTBL
     .       /ESTADO/ ICONDL
     .       /AVALIA/ NCALCF , NCALCG
C
      LOGICAL*1 RSTART(3) , IBUS(7) , CONVER
      DOUBLE PRECISION BET    , DIF    , ENORG  , PASSO,DIFNGX,
     .                 FUNC   ,          GRDFIN ,
     .                 PRDIN1 , PRDIN2 , RK(500), RKM1(500),XK,
     .                 XFINAL , DC(500), DK(500), DKM1(500)   ,
     .                 VALFUN , XKM1(500) , RC(500) ,RCM1(500),
     .                 OLDFUN ,             NORMEUC , BETAK   ,
     .                                               EPS
      DATA IBUS /' ' , 'E' , 'I' , 'F' , 'R' , 'N' , 'M'/
C
      EQUIVALENCE (XFINAL(1),XKM1(1)) , (GRDFIN(1),RKM1(1))
C
C
C     --------------
C     Inicializacoes
C     --------------
C
      NCALCF = 0
      NCALCG = 0
      NTOTBL = 0
      NRESBL = 0
      NREST  = 0
      I1     = 1
      I2     = 2
      I3     = 3
      LITEX  = 0
      K      = 0
      L      = 1
      NITER  = 0
      ICPONT = 1
      IPAGT1 = 1
      CONVER = .FALSE.
      LIMBIG = 100
C
C     ------------------------------------------------------------
C     Obtencao do valor da funcao e do antigradiente no pto.inicial
C     ------------------------------------------------------------
C
      CALL VALFUN(ITIPOF,N,XK,FUNC,RK,I2)
      ENORG  = NORMEUC(N,RK)
C
      WRITE(6,2000) IPAGT1
      WRITE(6,3010) FUNC   ENORG
C
```

```
C
C     ------------------------------------------------------
C     O laco 500 simula "infinitos" lacos de "N" iteracoes
C     ------------------------------------------------------
      DO 500 KKK = 1 , LIMDIG
C
C        ------------------------------------------------------
C        Obtencao da primeira direcao de busca (= -F'(XK))
C        ------------------------------------------------------
C
         DO 20 I = 1 , N
            DK(I) = PK(I)
   20    CONTINUE
C
C        ----------------------------------------
C        Laco do tipo "ENQUANTO-FACA" simulando um
C        ciclo de "N" iteracoes
C        ----------------------------------------
C
         KK = 1
   40    IF (KK.GT.N) GO TO 500
C
C             ---------------------------------
C             Ativacao da rotina 'TESTA' para
C             verificar convergencia do MGC
C             ---------------------------------
C
              VALFUN = FUNC
              IF (KKK.GT.1.AND.KK.EQ.1) NREST = NREST + 1
              K      = K + 1
              CALL TESTA(K,N,XK,XKM1,OLDFUN,FUNC,ENORG,EPS,CONVER)
              IF (.NOT.CONVER) GO TO 45
                  J = MOD(ICPONT,26)
                  IF ( J .EQ. 0 ) RETURN
                  WRITE(6,2080)
                  RETURN
C
   45         IF (NITER.LE.NMAX) GO TO 50
                  LITEX = 1
                  J     = MOD(ICPONT,26)
                  IF ( J .EQ. 0) RETURN
                  WRITE(6,2030)
                  RETURN
C
   50         OLDFUN       = FUNC
              IF (K.EQ.1) GO TO 60
                  DO 55 I  = 1 , N
                     XK(I) = XKM1(I)
                     PK(I) = PKM1(I)
                     DK(I) = DKM1(I)
   55             CONTINUE
C
C             ---------------------------------------
C             Busca linear ao longo da direcao 'DK'
C             ---------------------------------------
C
   60         CALL GLDWLF(XK,DK,PK)
              NTOTPL = NTOTPL + ICOUNT
              IF (ICONBI.EQ.0) RETURN
C
```

```
C
C     ------------------------------------------------------------
C     Obtencao de XKM1 , F(XKM1) , RKM1 e de // RKM1 //
C     (...e armazanamento temporario de //XKM1-XK// em RKM1)
C     ------------------------------------------------------------
      DO 70 I     = 1 , N
         XKM1(I)  = XK(I) + PASSO*DK(I)
         RKM1(I)  = XKM1(I) - XK(I)
   70 CONTINUE
      DIFNOX      = NORMUC(N,RKM1)
      CALL VALFR(ITIPOF,N,XKM1,FUNC,RKM1,I2)
      ENORG       = NORMUC(N,RKM1)
      IF (NITACE.EQ.0) GO TO 75
          WRITE(6,4010) (XKM1(I),I=1,N)
          WRITE(6,4010) (RKM1(I),I=1,N)
          WRITE(6,4010) FUNC
C
   75 NITER       = K
      KK          = KK + 1
      DIF         = FUNC - OLDFUN
C
C     ------------------------------------------------------------
C     Se esta iteracao nao for a ultima deste ciclo de "N",
C     calcula a direcao DKM1 conjugada 'as anteriores
C     ------------------------------------------------------------
      IF (KK.GT.N) GO TO 90
C
C         Obtem o coeficiente de conjugacao BET e DKM1
C         --------------------------------------------
          BET         = BETAK(DK,RK,RKM1)
          DO 80 I     = 1 , N
             DKM1(I)  = RKM1(I) + BET*DK(I)
   80     CONTINUE
          GO TO 110
C
C         Atualiza RK (proxima direcao de RESTART) com RKM1
C         -------------------------------------------------
   90     DO 100 I    = 1 , N
             RK(I)    = RKM1(I)
  100     CONTINUE
C
  110 CONTINUE
      IF ( ICPONT .LT. 25 ) GO TO 170
           WRITE(6,2080)
           ICPONT = 1
           IPAGI1 = IPAGI1 + 1
           WRITE(6,2060) IPAGI1
C
  170 IF(MOD(K,IMP).NE.0) GO TO 180
         WRITE(6,2020) NITER,FUNC,IBUS(ICONBL),PASSO,ICOUNT,
                       ENORG,DIFNOX,DIF
         ICPONT = ICPONT + 1
C
  180    GO TO 40
  500 CONTINUE
C
C
C
C
C
```

```
C
C
C
C     ---------------
C     Formatos usados
C     ---------------
C
 2020         FORMAT(1X,'I',113('-'),'I',/,1X,'I',T5,I6,T13,'I',I15,
     .        D18.11,T34,'I',T35,A1,
     .                       T36,D20.10,T57,'I',T59,I4,T64,'I',I66,
     .        D15.5,T82,'I',T83,D15.5,T99,'I',T100,D15.5,T116,'I')
 2060 FORMAT('1',/,1X,115('='),T119,'TABELA/1',/,
     .        1X,'I ITERACAO I  VALOR  DA  FUNCAO I',
     .        T39,'PASSO  ORTIDO',T57,'I NRO. I ',
     .        'VALOR DA NORMA',T81,' I  IIX  -  X II I',
     .        '  VARIACAO  DO',T116,'I',T120,'PAGINA',/,
     .        1X,'I',T13,'I',T34,'I',I44,'NA',I57,
     .        'I  DE  I',I67,'DO  GRADIENTE  I',T88,'K+1',
     .        T92,'  K  I',T103,'VALOR  DA',T116,'I',T122,I4,/,
     .        1X,'I',T13,'I',T17,'NESTA  ITERACAO  I',T39,
     .        'BUSCA  LINEAR',T57,'I B.L. I NESTA  ITERACAO I ',
     .        T99,'I',T104,'FUNCAO',T116,'I',/,1X,115('='))
 2080              FORMAT(1X,115('='))
 3010 FORMAT(1X,'I',T10,'0',T13,'I',T15,D18.11,T34,'I',T44,'---',
     .        I57,'I ---- I',T66,D15.5,T82,'I',I89,'----',T99,'I',
     .        T104,'--------',T116,'I')
 4010 FORMAT(/,5(2X,E10.4))
      END
```

```
      SUBROTINE GCALG2
C     ---------- ------ (Algoritmo 2)
C
      COMMON /DADOS1/ EPS    , NMAX , N , IMP , NTRACE
     .       /DADOS2/ XK(500)
     .       /CONST / IRETA , ITIPOF , INGERA , NACABL,LITEX
     .       /BUSCAL/ ALPA  , BETA   , PASSO  , FUNC  ,LCOUNT
     .       /RESULT/ ENORG , VALFUN , XFINAL(500)      ,
     .                GRDFIN(500) , NITER ,NRES1,NRESBL,NTOTBL
     .       /ESTADO/ ICONBL
     .       /AVALIA/ NCALCF , NCALCG
      INTEGER   C
      LOGICAL*1 ISTART(3) , IBUS(7) , CONVER
      DOUBLE PRECISION BET    , DIF    , ENORG  , PASSO,DIFNOX,
     .                 FUNC   , GAMA   , GRDFIN , P1   , P2    ,
     .                 PRDIN1 , PRDIN2 , RK(500), RKM1(500),XK,
     .                 XFINAL , DC(500), DK(500), DKM1(500)    ,
     .                 VALFUN , XKM1(500) , RC(500) ,RCM1(500),
     .                 OLDFUN , PRD1,PRD2 , NORMUC , BETAK    ,
     .                 PRDIN3 , COSENO,DA , ANGULO , ANG , EPS
C
      DATA IBUS /' ' , 'E' , 'I' , 'F' , 'R' , 'N' , 'M'/
C
      EQUIVALENCE (XFINAL(1),XKM1(1)) , (GRDFIN(1),RKM1(1))
C
C
C
C
C
C     --------------
C     Inicializacoes
C     --------------
C
      NCALCF = 0
      NCALCG = 0
      NTOTBL = 0
      NRESBL = 0
      NREST  = 0
      I1     = 1
      I2     = 2
      I3     = 3
      LITEX  = 0
      C      = 0
      K      = 1
      L      = 1
      NITER  = 1
      ICPONT = 1
      IPAGT1 = 1
      IPAGT2 = 1
      GAMA   = 0.0D0
      CONVER = .FALSE.
C
C     ------------------------------------------------------------
C     Obtencao do valor da funcao e do antigradiente em 'XK'
C     ------------------------------------------------------------
C
      CALL VALFP(ITIPOF,N,XK,FUNC,RK,I2)
      OLDFUN = FUNC
C
```

```
C
C     ------------------------------------------------------
C     Obtencao  da  primeira  direcao  de  busca (= -F'(XK))
C     ------------------------------------------------------
C
      ENORG   = NORMVC(N,GK)
      DO 20 I = 1 , N
         DK(I) = GK(I)
         RC(I) = FX(I)
         DC(I) = DF(I)
   20 CONTINUE
C
      WRITE(6,2060) IPAGT1
 2060 FORMAT('1',/,1X,115('='),T119,'TABELA/1',/,
     .       1X,'I ITERACAO I  VALOR  DA  FUNCAO I',
     .       T39,'PASSO  OBTIDO',T57,'I NRO. I ',
     .       'VALOR DA NORMA',T81,' I  IIX  -  X II I',
     .       '  VARIACAO  DO',T116,'I',T120,'PAGINA',/,
     .       1X,'I',T13,'I',T34,'I',T44,'NA',T57,
     .       'I  DE  I',T67,'DO  GRADIENTE  I',T88,'K+1',
     .       T92,'  K  I',T103,'VALOR  DA',T116,'I',T122,I4,/,
     .       1X,'I',T13,'I',T17,'NESTA  ITERACAO  I',T39,
     .       'BUSCA  LINEAR',T57,'I B.L. I NESTA  ITERACAO I ',
     .       T99,'I',T104,'FUNCAO',T116,'I',/,1X,115('='))
C
      WRITE(6,3010) FUNC , ENORG
 3010 FORMAT(1X,'I',T10,'0',T13,'I',T15,D18.11,T34,'I',T44,'---',
     .       T57,'I ---- I',T66,D15.5,T82,'I',T89,'----',T99,'I',
     .       T104,'--------',T116,'I')
C
      WRITE(7,3000) IPAGT2
 3000 FORMAT('1',/,T100,'TABELA/2 - PAGINA ',I4,//,1X,121('='),/,
     .       1X,'I',T13,'I',T31,'I',T53,'I',T58,'RESTART NECESSARIO',
     .    '? (T/F)  I',T104 , 'I',T122,'I',/,T2,'I',T13,'I',T31,'I',
     .     T53,'I PRIMEIRO',T75,'SEGUNDO',T86,'I',T104,'I',T122,'I',
     .    /,1X,'I ITERACAO I',T16,'I RK*RKM1 I',T31,'I',T35,'0.2*II',
     .       'RKM1II**2  I',T55,'CRITERIO  CICLO  CRITERIO  I',
     .       T90,'ANG(DK,RKM1)  I',T108,'ANG(RK,RKM1)  I',/,
     .       1X,'I',T13,'I',T31,'I',T53,'I',T58,'DE',T66,'DL  N',
     .       T77,'DF      I',T104,'I',T122,'I',/,
     .       1X,'I',T13,'I',T31,'I',T53,'I',T56,'POWELL  ITERACOES',
     .       '  POWELL ',T86,'I',T104,'I',T122,'I',/,1X,121('='))
   40 CONTINUE
C     ------------------------------------------------------
C     RSTART(1) - Primeiro criterio de POWELL
C     RSTART(2) - Ciclo de N iteracoes a partir do ultimo restart
C     RSTART(3) - Segundo  criterio de POWELL
C     ------------------------------------------------------
C
      RSTART(1) = .FALSE.
      RSTART(2) = .FALSE.
      RSTART(3) = .FALSE.
C
C     ------------------------------------------------------
C     Ativacao da rotina TESTA para verificar convergencia do NGC
C     ------------------------------------------------------
C
      VALFUN  = FUNC
      CALL TESTA(X,N,XK,XKM1,OLDFUN,FUNC,ENORG,EPS,CONVER)
      IF (.NOT.CONVER) GO TO 45
```

```
C
C           - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C           Foi verificada a convergencia do MGC para XK.
C           Se a ex-fetuca direcao (ja' calculada) tiver
C           sido do "RESTART",desconsidera-a na contabilizacao
C           de reinicializacoes.
C           - - - - - - - - - - - - - - - - - - - - - - - - - - - -
            IF (I.EQ.1.AND.K.GT.1) NREST = NREST - 1
            J = MOD(ICONT,26)
            IF ( J .EQ. 0 ) RETURN
            WRITE(6,2080)
            WRITE(7,2090)
            RETURN
   45 IF ( NITER . LT . NMAX     ) GO TO 50
            LATEX = 1
            J     = MOD(ICONT,26)
            IF ( J .EQ. 0) RETURN
            WRITE(6,2080)
            WRITE(7,2090)
            RETURN
C
C
C     ------------------------------------------------------------
C     Busca linear ao longo da direcao 'DK' (Obtencao do 'PASSO')
C     ------------------------------------------------------------
C
   50 OLDFUN        = FUNC
      IF (K.EQ.1)  GO TO 60
           DO 55 I  = 1 , N
              XK(I) = XKM1(I)
              RK(I) = RKM1(I)
              DK(I) = DKM1(I)
   55      CONTINUE
C
   60 CALL GLDWLF(XK,DK,RK)
      NTOTBL = NTOTBL + ICOUNT
      IF (NTRACE.EQ.0) GO TO 65
           WRITE(9,4010) (XK(I),I=1,N)
           WRITE(9,4010) (RK(I),I=1,N)
           WRITE(9,4010) FUNC
 4010      FORMAT(/,5(2X,E10.4))
C
   65 IF (ICONPL.EQ.0) RETURN
C
C     ---------------------------------------------
C     Calculo do novo ponto XKM1 . . .
C     ( e de  XKM1 - XK  , armazenado temp/ em RKM1)
C     ---------------------------------------------
C
      DO 70 I = 1 , N
         XKM1(I) = XK(I) + PASSO*DK(I)
         RKM1(I) = XKM1(I) - XK(I)
   70 CONTINUE
C
C     - - - - - - - - - - - - - - - - - - - - - - -
C     Armazenamento de || XKM1 - XK || em DIFNOX
C     - - - - - - - - - - - - - - - - - - - - - - -
C
      DIFNOX = NORFUC(N,RKM1)
C
```

```
C     ---+---------+---------+---------+---------+---------+---------+--
C     Obtencao do valor da funcao e do antigradiente em XKM1
C     ------------------------------------------------------------------
      CALL VALFG(ITMOD,N,XKM1,FUNC,GKM1,IZ)
C
      IF(L.EQ.0) GO TO 85
         DO 80 I = 1 , N
   80    RKM1(I) = DK(I)
C
   85 PRDIN1    = 0.0D0
      PRDIN2    = 0.0D0
      PRDIN3    = 0.0D0
      DO 90 I   = 1 , N
         PRDIN1 = PRDIN1 + RK(I)    * RKM1(I)
         PRDIN3 = PRDIN3 + DK(I)    * RKM1(I)
   90 CONTINUE
C
      ENORG  = NORMVC(N,RKM1)
      PRDIN2 = ENORG ** 2
      PRD2   = 0.2D0 * PRDIN2
      P1     = NORMVC(N,DK)
      COSENO = 0.0D0
      IF (ENORG.GE.1.0D-20) COSENO = PRDIN3 / (P1*ENORG)
      IF (COSENO.GT.1.0D0)  COSENO = 1.0D0
      IF (COSENO.LT.-1.0D0) COSENO =-1.0D0
      DA     = DACOS(COSENO)
      ANGULO = (DA*180.0D0) / 3.14159265360D0
C
C     ------------------------------------
C     Calculo do coeficiente de conjugacao
C     ------------------------------------
C
      BET    = BETAK(DK,RK,RKM1)
C
C     ------------------------------------------------------------------
C     Teste do primeiro criterio de POWELL e do ciclo de N iteracoes
C               (ORTOGONALIDADE ENTRE ANTIGRADIENTES)
C     ------------------------------------------------------------------
C
      IAUX      = C + N
      IF ( DABS(PRDIN1) . GE . PRD2 ) RSTART(1) = .TRUE.
      IF ( K      . EQ . IAUX ) RSTART(2) = .TRUE.
      IF ( (.NOT.RSTART(1)).AND.(.NOT.RSTART(2)) ) GO TO 120
C
C       ----------------------------------------------------------------
C       Calculo da nova direcao de busca 'RKM1' (Direcao de RESTART)
C       ----------------------------------------------------------------
C
        NREST       = NREST + 1
        C           = K
        L           = 1
C
        DO 100 I    = 1 , N
           DKM1(I) = RKM1(I) + BET*DK(I)
           DC(I)    = DKM1(I)
           RC(I)    = RKM1(I)
  100   CONTINUE
C
        GO TO 160
C
```

```
C     ---------+---------+---------+---------+---------+------
C     Calculo da nova direcao de busca 'DKM1' (Direcao de BEALE)
C     ---------+---------+---------+---------+---------+------
C
  120 CONTINUE
      P1       = 0.0D0
      P2       = 0.0D0
      L        = 0
C
C     -------------------------------
C     Calculo do coeficiente de BEALE
C     -------------------------------
C
      DO 140 I= 1 , N
         DIF   = GKM1(I) - GC(I)
         P1    = P1      + RKM1(I) * DIF
         P2    = P2      + DC(I)   * DIF
  140 CONTINUE
      GAMA     = -P1 / P2
C
      DO 150 I = 1 , N
  150    DKM1(I) = RKM1(I) + BET*DK(I) + GAMA*DC(I)
C
  160 NITER = K
      DIF   = FUNC - OLDFUN
      IF ( ICPONT .LE. 25 ) GO TO 170
C
            WRITE(6,2080)
 2080       FORMAT(1X,115('='))
            WRITE(7,2080)
 2080       FORMAT(1X,121('='))
            ICPONT = 1
            IPAGT1 = IPAGT1 + 1
            IPAGT2 = IPAGT2 + 1
            WRITE(6,2060) IPAGT1
            WRITE(7,3000) IPAGT2
C
  170 IF(MOD(K,IMP).NE.0) GO TO 175
            PRDIN3 = PRDIN1
            WRITE(6,2020)NITER,FUNC,IBUS(ICONBL),PASSO,
     .                   ICONJNT,ENORG,DIFNOX,DIF
 2020       FORMAT(1X,'I',113('-'),'I',/,1X,'I',T5,I6,T13,
     .           'I',T15,D18.11,T34,'I',T35,A1,T36,D20.10,
     .           T57,'I',T59,I4,T64,'I',T66,D15.5,T82,'I',
     .           T83,D15.5,T99,'I',T100,D15.5,T116,'I')
            ICPONT = ICPONT + 1
C
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C     Armazenamento temporario de ABS(RK*RKM1) em OLDFUN
C     - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
  175 OLDFUN = DABS(PRDIN1)
C
C
C     ------------------------------------------------
C     Se esta direcao e' de RESTART(L=1),e' de descida
C     ------------------------------------------------
C
      IF ( L.EQ.1 ) GO TO 210
C
```

```
C     ----------+---------+---------+---------+---------+---------+--------
C     Testa se a nova direcao (DF FINLP) e' suficiente/ de descida
C                    (SEGUNDO CRITERIO DE POWELL)
C     ------------------------------------------------------------------
C
          PRDIN1     = 0.0D0
          DO 180 I   = 1 , N
             PRDIN1 = PRDIN1 + DFM1(I)*RKM1(I)
  180     CONTINUE
          PRDIN2     = 1.2D0 * ENORG**2
          RSTART(3)  = (PRDIN1.GT.PRDIN2)
          IF ( RSTART(3) ) GO TO 200
          PRD1       = 0.2D0 * ENORG**2
          RSTART(3)  = (PRDIN1.LT.PRD1)
          IF ( .NOT.RSTART(3) ) GO TO 240
C
C
C
C
C     ------------------------------------------------------------------
C     Direcao obtida nao e' suficientemente de descida ; e' calcu-
C     lada uma nova direcao de busca 'DKM1'   (Direcao de RESTART)
C     ------------------------------------------------------------------
C
  200     L          = 1
          C          = K
          NREST      = NREST + 1
          DO  220  I = 1 , N
              DKM1(I)= RKM1(I) + BET*DK(I)
              DC(I)  = DKM1(I)
              RC(I)  = RKM1(I)
  220     CONTINUE
  240 IF(MOD(K,INF).NE.0) GO TO 260
      P1      = NORMUC(N,RK)
      COSENO = 0.0D0
      IF (ENORG.GT.10D-20) COSENO = PRDIN3 / (P1*ENORG)
      IF (COSENO.GT.1.0D0) COSENO = 1.0D0
      IF (COSENO.LT.-1.0D0)COSENO = -1.0D0
      DA      = DACOS(COSENO)
      ANG     = (DA*180.0D0) / 3.14159265358D0
      WRITE(7,2040) NITER,OLDFUN,PRD2,RSTART,ANGULO,ANG
 2040 FORMAT(1X,'I',T5,I6,T13,'I',T15,D15.5,T31,'I',      T35,D15.5,
     .       T53,'I',T58,L1,T68,L1,T78,L1,T86,'I',T88,D15.6,T104,'I'
     .      ,T106,D15.6,T122,'I')
  260 K       = K + 1
      GO TO 40
      END
```

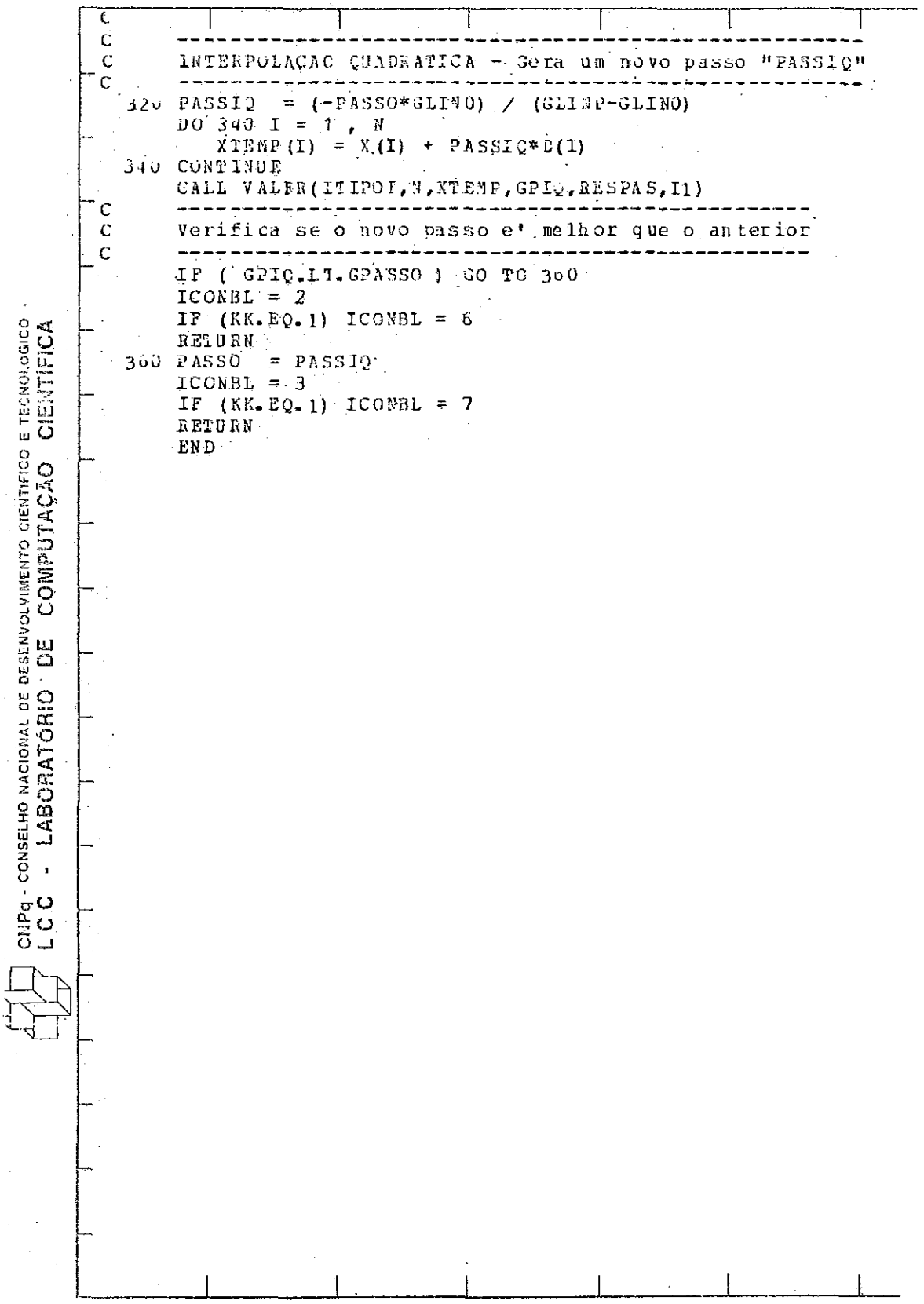
```
      SUBROUTINE TESTA(K,N,XK,XKM1,FXK,FXKM1,ENORG,EPS,CONVER)
C     ---------- ------------------------------------------
C
C ==========================================================================
C                                                                          =
C     OBJETIVO: Testar a convergencia do MGC, definida pela                =
C     --------- validade simultanea dos tres testes abaixo:                =
C               1-  Norma do gradiente <= SQRT(EPS)                        =
C               2-  Reducao da funcao  <= EPS                              =
C               3-  Variacao pontual   <= SQRT(EPS)                        =
C               OBS : Eventualmente, os testes 2 e/ou 3 podem              =
C               ---   ser absolutos ou relativos.                          =
C                                                                          =
C     ENTRADAS :                                                           =
C     --------                                                             =
C        K          - Numero da iteracao                                   =
C        N          - Dimensao do problema                                 =
C        XK(*)      - Ponto alcancado anteriormente (se K>1),              =
C                     Ponto atual (se K=1)                                 =
C        XKM1(*)    - Ponto atual (se K>1) , "DUMMY"(SE K=1)               =
C        FXK,FXKM1  - Valores da funcao em XK e XKM1 ; conforme            =
C                     o valor de K , se comportam analogamente             =
C                     a XK e XKM1                                          =
C        ENORG      - Valor da norma euclideana no ponto:                  =
C                     XK  --- se K=1                                       =
C                     XKM1 --- se K>1                                      =
C        EPS        - Tolerancia para os testes de convergencia           =
C                                                                          =
C     SAIDA :                                                              =
C     -----                                                                =
C        CONVER     = .FALSE. - Testes nao foram simultaneamente           =
C                               satisfeitos. Convergencia nao              =
C                               foi alcancada.                             =
C                   = .TRUE.  - Testes foram simultaneamente sa-           =
C                               tisfeitos.Convergencia alcancada.          =
C ==========================================================================
C
C
      INTEGER*4 K , N
      LOGICAL*1 CONVER
      REAL*8    XK(500) , XKM1(500), FXK , FXKM1 , ENORG , EPS ,
     .          VALUE   , DIFNOX   , DIFEMF , ENORT , ABSF ,
     .          DELTAX(500)        , SQRTEP , NOREUC
C
C     ----------------------------------
C     Inicialmente , assume convergencia.
C     ----------------------------------
C
      CONVER = .TRUE.
C
C     ----------------------------
C     TESTE 1 - Norma do gradiente
C     ----------------------------
C
      IF (ENORG.EQ.0.0D0)  RETURN
      SQRTEP = DSQRT(EPS)
      IF (ENORG.LE.SQRTEP) GO TO 10
         CONVER = .FALSE.
         RETURN
C
```

```
C     ---------+---------+---------+---------+---------+---------+-
C     Se o MGC estiver na primeira iteracao (K=1), o TESTE 1
C     satisfeito isoladamente implica em convergencia.
C     ------------------------------------------------------------
   10 IF (K.EQ.1) RETURN
C
      VALUE = 10.0D0
C     ------------------------------------
C     TESTE 2 - Reducao no valor da funcao
C     ------------------------------------
      DIFEMF = DABS(FXK-FXKM1)
      ABSF   = DABS(FXKM1)
      IF(ABSF.GE.VALUE) DIFEMF = DIFEMF / ABSF
      IF(DIFEMF.LE.EPS) GO TO 20
         CONVER = .FALSE.
         RETURN
C
   20 CONTINUE
C     --------------------------
C     TESTE 3 - Variacao pontual
C     --------------------------
      DO 30 I = 1 , N
         DELTAX(I) = XKM1(I) - XK(I)
   30 CONTINUE
      DIFNOX = NOREUC(N,DELTAX)
      ENORT  = NOREUC(N,XKM1)
      IF(ENORT.GE.VALUE) DIFNOX = DIFNOX / ENORT
      IF(DIFNOX.GT.SQRTEP) CONVER = .FALSE.
      RETURN
      END
```

```
      SUBROUTINE GLDSLF(X,D,RK)
C     ---------- ------
C
C     Esta rotina aplica o Metodo de GOLDSTEIN ou o de WOLFE
C     para a obtencao do "PASSO" da busca linear.Para aumentar a
C     robustez destes algoritmos,foram inseridas 2 salvaguardas
C     adicionais:
C     a. Se a busca linear na direcao DK gerar um passo insigni-
C        ficante,o algoritmo abandona esta direcao e desce na
C        direcao do antigraliente.
C     b. Em ambos os casos acima , apos o passo ser gerado , se
C        este degradar significativamente a ortogonalidade entre
C        DK e RK+1 (::= COSSENO >0.2 OU COSSENO <-0.2) , entao e'
C        executada uma interpolacao quadratica , usando-se os
C        valores F'(XK),F(XK+PASSO*D) e F'(XK+PASSO*D). Embora,
C        a principio,o comportamento desta salvaguarda possa
C        sugerir uma busca linear "mais exata",o objetivo de sua
C        insercao aqui e' o de preservar duas caracteristicas
C        teoricas fundamentais dos MGC , qual sejam:
C        - PRESERVACAO DE UMA QUASE-ORTOGONALIDADE ENTRE DK E RK+1
C        - NAO IGNORAR APROXIMACCES QUADRATICAS LOCAIS NA PROJECAO
C          DA FUNCAO NA DIRECAO 'DK' (OU 'D',COMO E' AQUI NOMEADA)
C-----------------------------------------------------------------
C     ICONBL - Variavel inteira que,apos a obtencao do PASSO,re-
C              torna um dos seguintes valores,conforme abaixo
C-----------------------------------------------------------------
C
C                ICONBL     SIMBOLO A SER          FATO
C                       IMPRESSO NA TABELA        OCORRIDO
C-----------------------------------------------------------------
C
C                  0            ---          MINIMO NA DIRECAO
C                                            ATUAL NAO FOI
C                                            ENCONTRADO
C
C                  1            ' '          NAO FOI PRECISO
C                                            INTERPOLACAO
C
C                  2            'E'          HOUVE INTERPOLACAO
C                                            SEM MELHORIA NO VALOR
C                                            DA FUNCAO
C
C                  3            'I'          HOUVE INTERPOLACAO
C                                            COM MELHORIA NO VALOR
C                                            DA FUNCAO
C
C                  4            'F'          DK=RK.ITERACAO RUIM ,
C                                             PASSO INSIGNIFICANTE
C
C                  5            'R'          DK=RK.NAO FOI PRECISO
C                                             INTERPOLACAO
C
C                  6            'N'          DK=RK.HOUVE INTERPOL.
C                                            SEM MELHORIA NO VALOR
C                                            DA FUNCAO
C
C                  7            'M'          DK=RK.HOUVE INTERPOL.
C                                            COM MELHORIA NO VALOR
C                                            DA FUNCAO
C-----------------------------------------------------------------
```

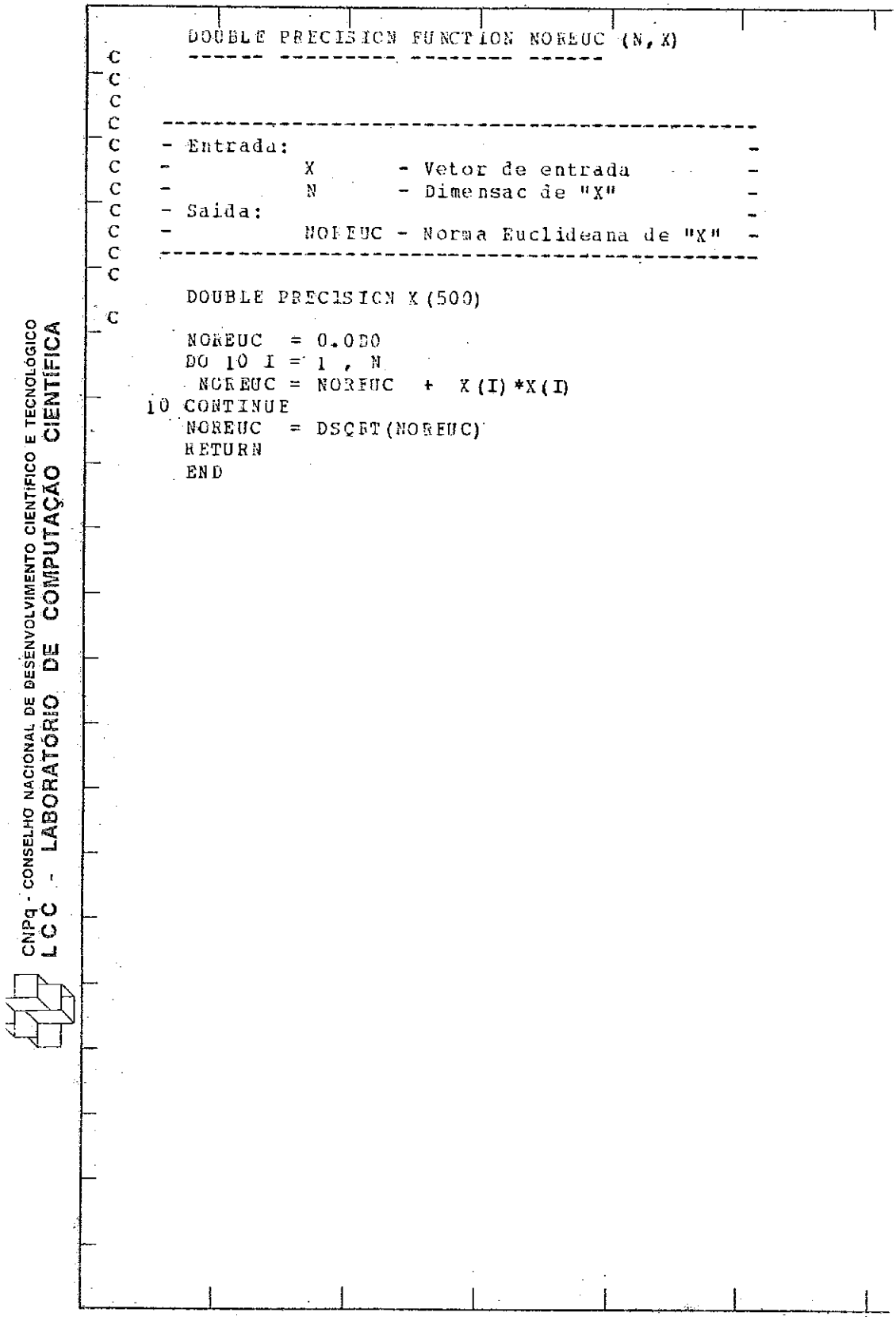
```
C
      COMMON /DADOS1/ EPS     , NMAX , N , IMP , NTFACE
     .       /BUSCAL/ ALFA    , BETA     , PASSO  , FUNC , ICOUNT
     .       /CONST/  IXBETA , ITIPOF    , IXGERA , MAPABL , LITEX
     .       /ESTADO/ ICONBL
      COMMON /RESULT/ BUFFER(1002),NITER , NGEST  , NRESBL , NTOTBL
      DOUBLE PRECISION  X(500) , D(500) , PASSO , ELAMBD  , FUNC ,
     .                  DLAMBD , GLINO  ,XTEMP(500) ,GPASSO,BUFFER,
     .                  RESPAS(500) , AUX , GLINP , RK(500) , DIF ,
     .                  DLIMF , CRKIDK , PASSIQ , GPIQ,NORMUC,EPS
      LOGICAL*1 INICIO
C
      INICIO = .TRUE.
      ICONBL = 1
      DLIMF  = -10.0D7
      GPIQ   = 0.0D0
      KK     = 0
      I1     = 1
      I2     = 2
      ICOUNT = 0
   10 PASSO  = 1.0D0
      GLINO  = 0.0D0
      DO 20 I = 1 , N
        GLINO = GLINO - RK(I)*D(I)
   20 CONTINUE
      ELAMBD  = 0.0D0
      DLAMBD  = 0.0D0
   40 ICOUNT = ICOUNT + 1
      IF (INICIO) GO TO 50
      DIF     = DLAMBD - ELAMBD
      IF (DABS(DIF).GT.1.0D-5) GO TO 50
      ICHAVE = 0
      GO TO 240
   50 DO 60 I = 1 , N
        XTEMP(I) = X(I) + PASSO*D(I)
   60 CONTINUE
      INICIO = .FALSE.
      ICHAVE = 1
      CALL VALFR(ITIPOF,N,XTEMP,GPASSO,RESPAS,I2)
      IF (GPASSO.LE.DLIMF) GO TO 200
      AUX     = FUNC  +  DBLE(ALFA)*PASSO*GLINO
      IF(GPASSO.LT.AUX) GO TO 80
        DLAMBD= PASSO
        PASSO = (ELAMBD + DLAMBD) / 2.0D0
        GO TO 40
C
   80 GO TO (100,140) , IXGERA
C
C     --------------
C     Teste de WOLFE
C     --------------
C
  100 GLINP   = 0.0D0
      DO 120 I= 1 , N
      GLINP   = GLINP - RESPAS(I) * D(I)
  120 CONTINUE
      AUX     = DBLE(BETA) * GLINO
      IF(GLINP.GT.AUX) GO TO 240
        GO TO 160
C
```

```
C
C     ------------------
C     Teste de GOLDSTEIN
C     ------------------
C
  140 AUX      = FUNC  +  DBLE(BETA)*PASSO*GLINO
      IF(GPASSO.GT.AUX) GO TO 240
C
  160   ELAMBD = PASSO
        IF(DLAMBD.NE.0.0D0)  GO TO 180
            PASSO = 2.0D0 * ELAMED + PASSO
            GO TO 40
  180   PASSO = (ELAMBD + DLAMBD) / 2.0D0
        GO TO 40
  200 ICONBL = 0
      RETURN
C
  240 CONTINUE
      IF (ICHAVE.EQ.1) GO TO 260
      IF ( KK.EQ.1 )   GO TO 255
      DO 250 I = 1 , N
          D(I) = PK(I)
  250 CONTINUE
      KK       = 1
      NRESBL   = NRESBL + 1
      INICIO   = .TRUE.
      GO TO 10
  255 PASSO    = (ELAMBD + DLAMBD) / 2.0D0
      ICONBL   = 4
      RETURN
C
C     -------------------
C     Fim da busca linear
C     -------------------
C
  260 CONTINUE
      IF (IXGERA.EQ.1) GO TO 300
      GLINP    = 0.0D0
      DO 280 I = 1 , N
         GLINP = GLINP - RESPAS(I) * D(I)
  280 CONTINUE
C
C     ------------------------------------------------------
C     Verifica se o passo gerado afetou significativamente
C     a ortogonalidade entre "D" e "RK+1"
C     ------------------------------------------------------
C
  300 CONTINUE
      AUX      = NOREUC(N,RESPAS)
      IF (AUX.GE.1.0D-10) GO TO 310
          IF(KK.EQ.1) ICONBL = 5
          RETURN
  310 AUX      = AUX * NOREUC(N,D)
      IF (AUX.GE.1.0D-10) GO TO 315
          IF(KK.EQ.1) ICONBL = 5
          RETURN
  315 CRK1DK = DABS(GLINP) / AUX
      IF (CRK1DK.GT.0.2D0) GO TO 320
          IF(KK.EQ.1) ICONBL = 5
          RETURN
```



```
C
C     ------------------------------------------------------
C     INTERPOLAÇAO QUADRATICA - Gera um novo passo "PASSIQ"
C     ------------------------------------------------------
  320 PASSIQ  = (-PASSO*GLIN0) / (GLINP-GLIN0)
      DO 340 I = 1 , N
         XTEMP(I) = X(I) + PASSIQ*D(1)
  340 CONTINUE
      CALL VALFR(ITIPOI,N,XTEMP,GPIQ,RESPAS,I1)
C     ------------------------------------------------
C     Verifica se o novo passo e' melhor que o anterior
C     ------------------------------------------------
      IF ( GPIQ.LT.GPASSO ) GO TO 360
      ICONBL = 2
      IF (KK.EQ.1) ICONBL = 6
      RETURN
  360 PASSO  = PASSIQ
      ICONBL = 3
      IF (KK.EQ.1) ICONBL = 7
      RETURN
      END
```

```
      DOUBLE PRECISION FUNCTION BETAK(DK,RK,RKM1)
C     ------ --------- -------- -----
C
C     ------------------------------------------------------------
C       Calcula o coeficiente de conjugacao utilizado no MGC
C     ------------------------------------------------------------
C
      COMMON /DADOS1/ EPS    , NMAX , N , IMP , NTRACE
     .       /CONST/  IXBETA , ITIPOF , IXGERA , MAPABL , LITEX
      DOUBLE PRECISION  DK(500) , RK(500), RKM1(500)    ,
                        DENOM   , RNUMER , TEMP , EPS
C
      RNUMER  = 0.0D0
      DENOM   = 0.0D0
      GO TO (10 , 30 , 50)   , IXBETA
C
   10 CONTINUE
C     ----------------------------
C       Formula de HESTENES-STIEFEL
C     ----------------------------
C
      DO 20 I = 1 , N
      TEMP    = RKM1(I) - RK(I)
      RNUMER  = RNUMER  + TEMP*RKM1(I)
      DENOM   = DENOM   + TEMP*DK(I)
   20 CONTINUE
      RNUMER  = -RNUMER
      GO TO 99
C
   30 CONTINUE
C     ------------------------
C       Formula de POLAK-RIBIERE
C     ------------------------
C
      DO 40 I = 1 , N
       TEMP   = RKM1(I) - RK(I)
       RNUMER = RNUMER + TEMP*RKM1(I)
       DENOM  = DENOM  + RK(I)**2
   40 CONTINUE
      GO TO 99
C
   50 CONTINUE
C     ----------------------------
C       Formula de FLETCHER-REEVES
C     ----------------------------
C
      DO 60 I = 1 , N
      RNUMER  = RNUMER + RKM1(I)**2
      DENOM   = DENOM  + RK(I)**2
   60 CONTINUE
C
   99 CONTINUE
      BETAK   = RNUMER / DENOM
      RETURN
      END
```



```
      DOUBLE PRECISION FUNCTION NOREUC (N, X)
C     ------ --------- -------- ------
C
C
C
C     ------------------------------------------------
C     - Entrada:                                     -
C     -           X       - Vetor de entrada         -
C     -           N       - Dimensao de "X"          -
C     - Saida:                                       -
C     -           NOREUC - Norma Euclideana de "X"   -
C     ------------------------------------------------
C
      DOUBLE PRECISION X(500)
C
      NOREUC  = 0.0D0
      DO 10 I = 1 , N
       NOREUC = NOREUC  +  X(I)*X(I)
   10 CONTINUE
      NOREUC  = DSQRT(NOREUC)
      RETURN
      END
```