ANÁLISE DE ALGORÍTMOS PARA O PROBLEMA DE MINIMIZAÇÕES SEM RESTRIÇÕES

MARIA AMÉLIA BIAGIO

Orientador:
Prof. Dr. JOSÉ MÁRIO MARTÍNEZ

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

Julho/1981

Aos meus pais,

Maria e Egídio.

AGRADECIMENTOS

Ao Prof. José Mário Martínez, por ter-me proporcionado força e orientação em todo tempo que trabalhamos juntos.

Ao Tarcísio, por ter-me ajudado nos trabalhos finais de computação.

A todos que estiveram ao meu lado e que, de alguma forma, me incentivaram e me ajudaram a prossequir.

Ao CNPq e à FAPESP, pela ajuda financeira.

ÍNDICE

INTRODUÇÃO..... 06

CAPÍTULO I - OS MÉTODOS DE MINIMIZAÇÃO "NEWTON", "QUASE-NEWTON" E "GRADIENTES CONJUGADOS"

1) O Método de Newton..... 09
   1.1) O Algorítmo do Método de Newton..... 09
   1.2) Observações, Teorema da Convergência..... 11
   1.3) O Método de Newton para Hessiano Discretizado..... 14
2) Os Métodos Quase-Newton..... 15
   2.1) Formas Recursivas para o Cálculo da "Matriz Aproximada" 15
   2.2) Observações, Fórmula de Sherman-Morrison..... 18
   2.3) Teorema da Convergência, Algorítmos para "Quase-Newton" 19
3) O Método dos Gradientes Conjugados..... 21
   3.1) O Algorítmo para Funções Quadráticas, Teorema da Convergência..... 21
   3.2) Generalização para Funções Não-Quadráticas..... 26

CAPÍTULO II - FORMAS DE RESOLUÇÃO PARA O PROBLEMA DA NÃO-POSITIVIDADE DA MATRIZ HESSIANA PARA O MÉTODO DE NEWTON

2.1) O Método de Newton com Fatorização de Choleski Modificada..... 29
   2.1.1) Considerações..... 29
   2.1.2) Como Contornar Problemas Numéricos - Teorema... 31
   2.1.3) Uma Alternativa: Direção de Curvatura; O Algorítmo e o Teorema da Convergência..... 34
2.2) O Método de Newton Canalizado..... 37
   2.1.1) Definições..... 37
   2.2.2) A Direção de Curvatura Negativa..... 38
   2.2.3) O Algorítmo..... 39
   2.2.4) A Fatorização de Choleski..... 41
   2.2.5) Exemplos Ilustrativos..... 45
   2.2.6) Diagrama em Blocos para Programação Quadrática. 57

CAPÍTULO III - ESTIMATIVAS A PRIORI DA PERFORMANCE DE ALGORÍTMOS

3.1) Desenvolvimento Similar ao que conduz à Noção de Eficiência de Ostrowski..... 60
3.2) Cálculo do Trabalho por Iteração dos Métodos de Newton e Quase-Newton e a Função de Decisão..... 61
3.3) Uma Aplicação..... 66

CAPÍTULO IV - TESTES
4.1) Descrição dos Programas Utilizados..... 72
4.2) Funções-Teste Utilizadas..... 74
4.3) Considerações..... 75
4.4) Tabelas e Observações..... 77

CONCLUSÕES..... 105

APÊNDICE..... 109

REFERÊNCIAS BIBLIOGRÁFICAS..... 113

INTRODUÇÃO

Dizemos que um algorítmo é tão eficiente quanto for menor o seu trabalho computacional para atingir convergência.

O trabalho computacional está diretamente relacionado com o número de iterações necessárias para um algorítmo convergir e o seu trabalho por iteração. Podemos predizer, para certos algorítmos, qual deve ser este número de iterações se o problema for tratado numa vizinhança pequena da solução.

O assunto desta pesquisa situa-se na *análise de algorítmos para minimização de funções sem restrições*.

Na literatura sobre o tema acham-se, muitas vezes, afirmações sobre a eficácia relativa a algorítmos, as quais são pouco - precisas e de duvidosa interpretação. Por exemplo, é sabido que para problemas grandes, isto é, problemas cujas funções possuem muitas variáveis, o método do tipo *Gradientes Conjugados* deve ser usado. Esta afirmação é clara no que se refere à "memória" requerida; porém, ainda quando a memória não é problema, o método dos Gradientes Conjugados pode ser competitivo em relação aos métodos com uma ordem maior de convergência devido ao "baixo custo" (overhead) que tem para calcular uma direção de decréscimo. Por outro lado, para um número pequeno de variáveis, o método de Newton é mais eficiente que os métodos de Gradientes Conjugados e Quase-Newton.

Ainda assim, o fato de que um método seja preferível a outros depende das características da função (tempo de computação da função e de seu gradiente) e do número de iterações necessárias para chegar à resolução do problema.

Nosso trabalho se resume em estudarmos o comportamento dos algorítmos de Newton (e Newton Discreto), Quase-Newton e Gradientes Conjugados na resolução de problemas de minimização de funções sem restrições.

Propomo-nos a construir uma *função de decisão* que dependa dos tempos de computação de funções e gradientes, do número de variáveis (as quais geram o "overhead") e da ordem de convergência (que ajuda a predizer o número relativo de iterações) para determinarmos qual dos métodos deve ser usado em cada caso, Newton ou Quase-Newton.

Nosso interesse, neste trabalho, está também em encontrarmos um algorítmo eficiente que contorne o problema da não-positividade da matriz hessiana, empregada na busca unidimensional do algorítmo de Newton, sem se utilizar da *direção de máxima descida*. Para isso, estudaremos versões de Gill e Murray para obtenção de *direções de curvatura negativa*.

Dessa forma, no Capítulo 1 faremos um rápido estudo dos algorítmos de Newton, Quase-Newton e Gradientes Conjugados. Para o algorítmo Quase-Newton veremos as formas recursivas D.F.P. (Davidon-Fletcher-Powell) e B.F.G.S. (Broyden-Fletcher-Goldfarb-Shanno) para o cálculo das *matrizes aproximadas* do hessiano da função em questão, utilizadas na busca unidimensional deste algorítmo.

No Capítulo 2 estudaremos as versões de Gill-Murray, que implementam o algorítmo de Newton e, no Capítulo 3, descreveremos uma teoria semelhante à que conduz à noção de eficiência de Ostrowski e construiremos a *função de decisão*, dando alguns exemplos de como aplicá-la.

Já no Capítulo 4, faremos comentários sobre os programas utilizados para os vários algorítmos e forneceremos tabelas contendo os resultados obtidos através das nossas experiências e que nos mostrarão, de certa forma, o que podemos esperar desses algorítmos em termos de eficiência e a validade da função de decisão construída.

CAPÍTULO I

OS MÉTODOS DE MINIMIZAÇÃO "NEWTON", "QUASE-NEWTON" E "GRADIENTES CONJUGADOS"

1) O MÉTODO DE NEWTON:

1.1) Seja $f : \mathbb{R}^n \to \mathbb{R}$ uma função diferenciável.

Nosso problema, aqui, se resume em acharmos um *mínimo* para a função "f", onde este mínimo é um ponto de $\mathbb{R}^n$; ou seja, queremos encontrar uma solução, $x^* \in \mathbb{R}^n$, para o problema

$$\begin{aligned} &\text{Min} \quad f(x) \\ &\text{s.a} \quad x \in \mathbb{R}^n \end{aligned}$$

e sabemos que o gradiente desta função aplicado a x^* deve ser nulo, isto é, $\nabla f(x^*) = 0$.

Chamamos de V uma vizinhança de x_0 (uma vizinhança de - "raio" pequeno), $x_0 \in \mathbb{R}^n$ e próximo a x. Como f é diferenciável, podemos escrevê-la da forma

$$f(x) = f(x_0) + \langle \nabla f(x_0), x-x_0 \rangle + \ldots$$

$$\ldots \frac{1}{2} (x-x_0)^t \langle \nabla^2 f(x_0), x-x_0 \rangle + 0(\|x-x_0\|^3)$$

para todo $x \in V$. Logo

$$f(x) \approx f(x_0) + \langle \nabla f(x_0), x-x_0 \rangle + \frac{1}{2} (x-x_0)^t \langle \nabla^2 f(x_0), x-x_0 \rangle$$

para todo $x \in V$.

Isto é, $f(x)$ é uma função que "se aproxima" de uma *forma quadrática* numa vizinhança de x_0; ou seja, podemos, ainda, escrever a nossa função da seguinte maneira:

$$f(x) = \frac{1}{2} (x-x_0)^t G(x-x_0) + F(x-x_0) + C$$

onde $x \in V$, $G(x) = \nabla^2 f(x)$ e $F(x) = \nabla F(x)$.

Como sabemos que a solução para o problema de minimização está em resolver a equação $f'(x) = 0$, então a nossa "solução" está em resolver

$$G(x-x_0) + F(x_0) = 0 , \quad x \in V$$

ou seja,

$$x = x_0 - G^{-1}F(x_0) , \quad x \in V .$$

Isto é, x é tal que $f'(x) \approx 0$ ($f'(x)$ "se aproxima" de zero);portanto, também é um ponto que se aproxima da solução do problema e esta aproximação *deverá ser melhor* do que a de x_0 .

Chamaremos $x = x_1$ o ponto obtido a partir de x_0, $x = x_2$ o ponto obtido a partir de x_1, e assim sucessivamente até alcançarmos uma aproximação desejada $x = x_{k+1}$, ou seja ,

$$x_{k+1} = x_k - G^{-1}F(x_k) , \quad k > 0$$

e a sequência de pontos obtida $(x_0, x_1, \ldots, x_{k+1})$ deverá convergir para a solução do problema, ou ainda, $\nabla f(x_{k+1}) \to 0$ quando $k \to \infty$.

Antes de provarmos este resultado, devemos tentar responder a uma pergunta que surge aqui: "o que nos garante, com este *método de busca*, que estamos caminhando, passo a passo, na direção do mínimo da função, visto que para pontos "de máximo" e "de sela" o gradiente da função também se anula?"

Vimos acima que a *direção de Newton* para a busca do ponto seguinte é $(-G^{-1}F) = [-\nabla^2 f]\nabla f$ e sabemos que o hessiano, $\nabla^2 f(x_k)$, só pode garantir uma "busca" eficiente se for uma matriz positiva semi-definida, isto é, se todos os seus autovalores forem positi-

vos ou nulos (*). Caso contrário, isto é, se algum autovalor de $\nabla^2 f(x_k)$ for negativo, então devemos deixar a "direção de Newton" e utilizar a *direção de máxima descida*, o que é razoável.

Dessa forma, podemos agora construir os "PASSOS" do algorítmo para o método de Newton para minimização de funções sem restrições:

PASSO 1: x_0 arbitrário; $k = 0$.

PASSO 2: Se $\|\nabla f(x_k)\| \leq \varepsilon$, pare. Caso contrário, vá para o "PASSO 3".

PASSO 3: Se x_k é tal que $\nabla^2 f(x_k)$ é positiva definida, vá para o "PASSO 5". Caso contrário, vá para o "PASSO 4".

PASSO 4: $z = -\nabla f(x_k)$.

PASSO 5: Resolver $[\nabla^2 f(x_k)]z = -\nabla f(x_k)$.

PASSO 6: $x_{k+1} = x_k + \lambda_k z$, onde $f(x_k+\lambda_k z) = \min\{f(x_k+\lambda z),\ \lambda \geq 0\}$.

PASSO 7: $k = k+1$ e vá para "PASSO 2".

1.2) OBSERVAÇÕES:

1.2.a) ε é um número tão pequeno quanto "se queira", ou quanto for a necessidade de "precisão" na solução a ser encontrada.

1.2.b) Para resolvermos o sistema do *PASSO 5*, utilizaremos a decomposição de Choleski para matrizes e isto trará facilidades para o algorítmo, visto que uma matriz só é decomposta pelo

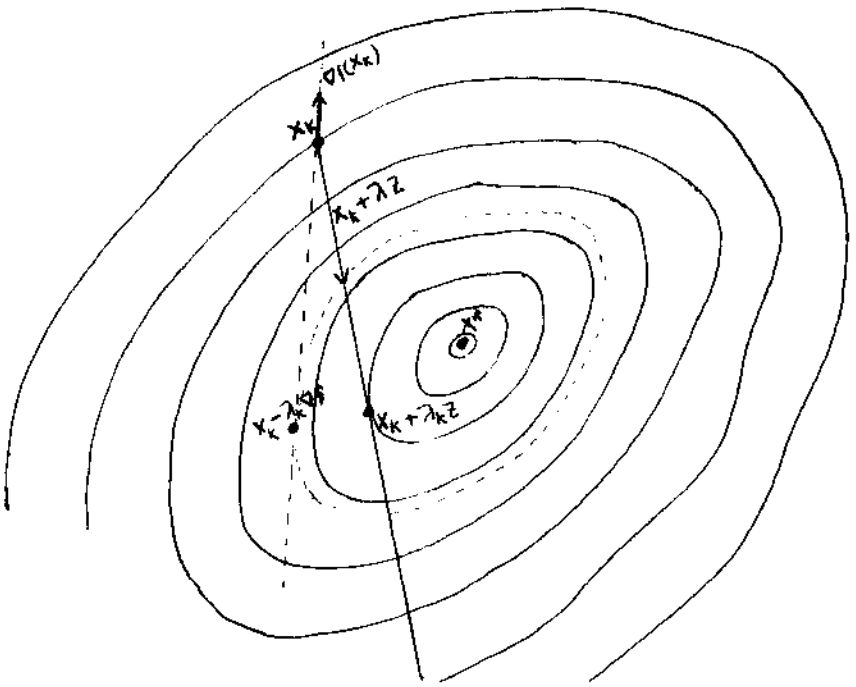
---

(*) Se todos os autovalores de uma matriz dada, num ponto x, forem positivos (ou nulos), então x é ponto de mínimo global (local) se $\nabla f(x) = 0$.

método de Choleski se ela for positiva definida.

1.2.c) No *PASSO 6*, simplesmente deixamos de usar $\lambda_k = 1$, $k=1,\ldots,$ para resolvermos, aí, outro problema de minimização, o - qual nos garantirá o melhor valor para λ_k (e, consequentemente, para x_{k+1}) na k-ésima iteração.

Logo, o algorítmo, construído desta maneira, nos garantirá que, a cada iteração k, devemos ter $f(x_{k+1}) < f(x_k)$.



Demonstraremos agora o teorema de convergência para o método de Newton, mas para isso, devemos assinalar uma das condições necessárias para tal, que é a

CONDIÇÃO 1: Seja $f(x)$ uma função duas vezes diferenciável, satisfazendo $m\|y\|^2 \leq \langle f''(x)y, y\rangle \leq M\|y\|^2$, $m > 0$, $M > 0$ e $x, y \in E^n$.

TEOREMA: Seja $f(x)$ uma função que satisfaz a condição 1 e tal

que $\|f''(x) - f''(y)\| \leq R\|x-y\|$, $x,y \in E^n$, isto é, $f(x)$ satisfaz, também, a condição de Lipschitz.

Seja x_0 o ponto inicial dado. Se a sequência dos pontos $(x_0,x_1,\ldots,x_k)$, dada acima pelo algorítmo, converge para o mínimo x^*, então esta convergência é quadrática, isto é,

$$\|x_{k+1} - x^*\| \leq \frac{R}{m} \|x_k - x^*\|^2, \quad m > 0$$

PROVA: Sabemos, a priori, que $f(x)$ é uma função diferenciável em todo o seu domínio. Pela fórmula de Lagrange (*) nós podemos escrever a seguinte igualdade:

$$\langle (f_k'')^{-1} f_k', x_{k+1} - x^* \rangle = \langle (f_k'')^{-1} (f_k' - f_*'), x_{k+1} - x^* \rangle$$

onde $f_k = f(x_k)$, $f_* = f(x^*)$ e $f_*' = f'(x^*) = 0$. Ou seja,

$$\langle (f_k'')^{-1} f_k', x_{k+1} - x^* \rangle = \langle (f_k'')^{-1} f_{kc}'' (x_k - x^*), x_{k+1} - x^* \rangle$$

onde $x_{kc} = x_k + \theta(x_k - x^*)$, $\theta \in [0,1]$. Consequentemente,

$$\|x_{k+1} - x^*\|^2 = \langle x_{k+1} - x^*, x_{k+1} - x^* \rangle = \langle x_k - x^* - (f_k'')^{-1} f_k', x_{k+1} - x^* \rangle =$$

$$= \langle (x_k - x^*) - (f_k'')^{-1} f_{kc}'' (x_k - x^*), x_{k+1} - x^* \rangle =$$

$$= \langle (I - (f_k'')^{-1} f_{kc}'')(x_k - x^*), x_{k+1} - x^* \rangle =$$

$$= \langle (f_k'')^{-1} (f_k'' - f_{kc}'')(x_k - x^*), x_{k+1} - x^* \rangle \leq \quad \text{(Pela Condição 1)}$$

$$\leq \frac{1}{m} \|f_k'' - f_{kc}''\| \, \|x_k - x^*\| \, \|x_{k+1} - x^*\|, \quad m > 0 .$$

(*) *f diferenciável, $x,y,h \in E^n$.*
$\langle f(x+h) - f(x), y \rangle = \langle f'(x+\theta h)h, y \rangle$, $\theta \in (0,1)$

Escrevendo

$$\lambda_k = \frac{1}{m} \|f_k'' - f_{kc}''\|$$

vem que

$$\|x_{k+1}-x^*\|^2 \leq \lambda_k \|x_k-x^*\| \; \|x_{k+1}-x^*\|$$

ou seja

$$\|x_{k+1}-x^*\| \leq \lambda_k \|x_k-x^*\|$$

Pela condição de Lipschitz, temos que

$$\|f''(x)-f''(y)\| \leq R\|x-y\|, \quad x,y \in E^n, \quad R > 0 \; .$$

Desta forma, podemos obter

$$\lambda_k = \frac{1}{m} \|f_k'' - f_{kc}''\| \leq \frac{R}{m} \|x_k-x^*\| \; .$$

Portanto,

$$\|x_{k+1}-x^*\| \leq \lambda_k \|x_k-x^*\| \leq \frac{R}{m}\|x_k-x^*\|^2 \; .$$

Assim,

$$\|x_{k+1}-x^*\| \leq \frac{R}{m} \|x_k-x^*\| , \quad m,R > 0$$

c.q.d.

1.3) <u>O MÉTODO DE NEWTON PARA HESSIANO DISCRETO - "NEWTON DISCRETO"</u>:

Vimos que, para utilizarmos o método usual de Newton, precisamos calcular o hessiano da função em questão, ou seja, precisamos também fazer os cálculos das *segundas derivadas* dessa função - o que nos é bastante penoso. Veremos então, como podemos tratar o hessiano, calculando apenas as primeiras derivadas, isto é, fazendo uma aproximação das derivadas segundas por "diferenças finitas"; por exemplo:

$$\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \approx \left[\frac{\frac{\partial f}{\partial x_i}(x+he_j) - \frac{\partial f}{\partial x_i}(x)}{h} + \frac{\frac{\partial f}{\partial x_j}(x+he_i) - \frac{\partial f}{\partial x_j}(x)}{h} \right] \Big/ 2$$

onde $e_k = (0,\ldots,0,\underset{\uparrow}{1},0,\ldots,0)$ e h "pequeno" .
k-ésimo

Assim, o cálculo do hessiano é substituído pelo cálculo de n gradientes em pontos "auxiliares".

Quanto à convergência do algorítmo assim modificado, podemos dizer que, como estamos tratando de uma "aproximação" para os elementos do hessiano da função, então, provavelmente, conseguiremos, a partir daí, uma matriz "aproximada" de $\nabla^2 f(x)$ e que se comportará como tal. Dessa maneira, a menos dos riscos de "arredondamento", este algorítmo deverá convergir e o teorema acima demonstrado também será verdadeiro para este caso.

Pode ser provado, com efeito, que se $h_k = 0(\|g(x_k)\|)$ a ordem de convergência de Newton discreto é também igual a 2.

2) OS MÉTODOS "QUASE-NEWTON":

2.1) Analogamente ao método de Newton, os métodos Quase-Newton tratam de problemas cujas funções são diferenciáveis no $\mathbb{R}^n$ e, portanto, aproximáveis localmente por formas quadráticas.

Então, da mesma forma, se $f : \mathbb{R}^n \to \mathbb{R}$ é diferenciável, podemos escrever

$$f(x) = \frac{1}{2} x^T G x + b^T x + c \ , \quad c \in \mathbb{R}^n$$

e temos um algorítmo para minimização de funções semelhante ao do método de Newton, contendo uma diferença em sua "busca unidimensional"; ou seja,

$$x_{k+1} = x_k - \lambda_k [B_k^{-1}] \nabla f(x_k), \quad x_k \in \mathbb{R}^n, \ k = 1,2,... \tag{1}$$

ou, alternativamente,

$$x_{k+1} = x_k - \lambda_k H_k \nabla f(x_k), \quad x_k \in \mathbb{R}^n, \ k = 1,2,... \tag{2}$$

onde devemos ter B_k^{-1} como uma matriz aproximada de $\nabla^2 f(x_k)$, ou, se for o caso, H_k como uma aproximação de $[\nabla^2 f(x_k)]^{-1}$ e podemos obter essas matrizes através de fórmulas recursivas, que citaremos abaixo.

A fim de que possamos garantir uma boa aproximação para B_k, ou H_k^{-1}, temos que preservar certas propriedades importantes que caracterizam o hessiano da função "f", quando f é quadrática:

"Se $y,z \in \mathbb{R}^n$ então $\nabla^2 f(y-z) = \nabla f(y) - \nabla f(z)$

e $[\nabla^2 f]^{-1}(\nabla f(y) - \nabla f(z)) = y - z$. "

Para isso, então, B_k e H_k devem satisfazer:

$$B_{k+1}(x_{k+1} - x_k) = g(x_{k+1}) - g(x_k) \tag{3}$$

$$H_{k+1}(g(x_{k+1}) - g(x_k)) = x_{k+1} - x_k \tag{4}$$

onde $x_{k+1}, x_k \in \mathbb{R}^n$ e $g = \nabla f$.

Existem muitas fórmulas recursivas, para o cálculo de ambas matrizes, mas falaremos apenas de algumas, aqui.

a) (Conhecida como "single-rank fórmula".)

Chamamos de $\delta = x_{k+1} - x_k$ e $\gamma = g(x_{k+1}) - g(x_k)$.

Por (3) desejamos que $B_{k+1}\delta = \gamma$, onde $B_{k+1} = B_k + \Delta B_k$, sendo ΔB_k uma matriz de *posto 1*.

De $B_{k+1}\delta = \gamma$ vem que $(B_k+\Delta B_k)\delta = \gamma$; ou seja, $\Delta B_k\delta = \gamma - B_k\delta$, donde podemos concluir que

$$\Delta B_k = \frac{(\gamma - B_k\delta)z^t}{z^t\delta},$$

z um vetor arbitrário de $\mathbb{R}^n$.

Se o nosso objetivo é *poupar memória*, então nos interessa que B_{k+1} seja simétrica, isto é,

$$\Delta B_k = \frac{(\gamma - B_k\delta)(\gamma - B_k\delta)^t}{(\gamma - B_k\delta)^t\delta}$$

b) Utilizando o mesmo raciocínio de (a), queremos que $H_{k+1}\gamma = \delta$, onde $H_{k+1} = H_k+\Delta H_k$, com ΔH_k de *posto 1*. Então

$$\Delta H_k = \frac{(\delta - H_k\gamma)(\delta - H_k\gamma)^t}{(\delta - H_k\gamma)^t}$$

c) Fórmula de Broyden-Flecher-Goldfarb-Shanno (BFGS).

Nesta fórmula requeremos que $B_{k+1} = B_k+\Delta B_{k1}+\Delta B_{k2}$, onde ΔB_{k1} e ΔB_{k2} são matrizes de *posto 1*.

Analogamente, para o caso (3), devemos ter

$$(B_k + \Delta B_{k1} + \Delta B_{k2})\delta = \gamma$$

ou seja,

$$\Delta B_{k1}\delta + \Delta B_{k2}\delta = \gamma - B_k\delta .$$

Para que esta igualdade seja satisfeita, deve ocorrer $\Delta B_{k1}\delta = \gamma$ e $\Delta B_{k2}\delta = -B_k\delta$ e então temos

$$\Delta B_{k1} = \frac{\gamma\gamma^t}{\gamma^t\delta} \quad \text{e} \quad \Delta B_{k2} = \frac{-B_k\delta\delta^tB_k}{\delta^tB_k\delta}$$

d) Fórmula de Davidon-Fletcher-Powell (DFP).

Da mesma forma, para o caso (4) queremos que

$$H_{k+1} = H_k + \Delta H_{k1} + \Delta H_{k2}$$

onde obtemos

$$\Delta H_{k1} = \frac{\delta\delta^t}{\delta^t\gamma} \quad \text{e} \quad \Delta H_{k2} = \frac{-H_k\gamma\gamma^t H_k}{\gamma^t H_k\gamma}$$

2.2) Através destas fórmulas, podemos observar que:

2.2.1) Se B_k (H_k) é simétrica, então B_{k+1} (H_{k+1}) continua sendo simétrica. Portanto, se B_0 (H_0) é simétrica, então, para cada iteração k, B_k (H_k) também será simétrica e, dessa - forma, conseguimos poupar mais "memória da máquina".

2.2.2) Se B_0 (H_0) é positiva definida, então B_k (H_k) é positiva definida, para qualquer k; logo, B_k^{-1} também é definida positiva [2], e assim as direções geradas pelo método são todas direções de descida.

2.2.3) Dado um método que utiliza B_k na busca unidimensional, é possível encontrarmos um método que utiliza H_k nesta busca e que são equivalentes. Isto é, para que B_k seja uma boa aproximação de G, devemos ter $B_{k+1}\delta = \gamma$, ou seja, $B_{k+1}^{-1}\gamma = \delta$ e daí temos um método que utiliza $H_k = B_{k+1}^{-1}$ como aproximação para o hessiano da função em questão.

Também existem fórmulas de recorrência para os cálculos das matrizes H_k^{-1}, $k = 1,2,\ldots$, as quais nos permitem gerá-las com poucas operações (com exceção de H_0). Para B_k da DFP temos

$$H_{k+1}^{-1} = H_k^{-1} - \frac{\gamma\delta^t H_k^{-1}}{\delta^t\gamma} - \frac{H_k^{-1}\delta\gamma^t}{\delta^t\gamma} + \left(1 + \frac{\delta^t H_k \delta}{\delta^t\gamma}\right)\frac{\gamma\gamma^t}{\delta^t\gamma} \qquad (5)$$

E, o que nos garante este resultado é a

FÓRMULA DE SHERMAN-MORRISON: Seja A uma matriz invertível de ordem nxn, u e v $\in \mathbb{R}^n$. Então $A+uv^t$ é invertível se $1+v^tA^{-1}u \neq 0$ e, neste caso,

$$(A+uv^t)^{-1} = A^{-1} - [1/(1+v^tA^{-1}u)]A^{-1}uv^tA^{-1} \quad (*)$$

Para chegarmos ao resultado (5) devemos aplicar a fórmula de Sherman-Morrison duas vezes na matriz $H_{k+1} = H_k + \Delta H_{k1} + \Delta H_{k2}$ de DFP; ou seja, fazendo $\frac{\delta}{\delta^t\gamma} = u$, $\delta^t = v^t$ e $A = H_k$ obtemos

$$(H_k+\Delta H_{k1})^{-1} = \left(H_k + \frac{\delta\delta^t}{\delta^t\gamma}\right) = H_k^{-1} - \frac{H_k^{-1}\delta\delta^t H_k^{-1}}{(1+\delta^t\gamma)\delta^t\gamma}$$

e, repetindo o procedimento ao somarmos ΔH_{k2}, vamos encontrar

$$H_{k+1}^{-1} = \left(I - \frac{\gamma\delta^t}{\delta^t\gamma}\right) H_k^{-1} \left(I - \frac{\delta\gamma^t}{\delta^t\gamma}\right) + \frac{\gamma\gamma^t}{\delta^t\gamma} ,$$

donde vem o resultado desejado. (*)

Da mesma forma, podemos chegar a

$$B_{k+1}^{-1} = \left(I - \frac{\delta\gamma^t}{\delta^t\gamma}\right) B_k \left(I - \frac{\gamma\delta^t}{\delta^t\gamma}\right) + \frac{\delta\delta^t}{\delta^t\gamma} , \qquad (6)$$

onde $B_{k+1} = B_k + \Delta B_{k1} + \Delta B_{k2}$ segundo BFGS .

2.3) Vejamos agora, o teorema de Convergência para Quase-Newton:

(*) Todos esses resultados estão demonstrados em DENNIS, MORÉ [2].

TEOREMA: Seja $f : \mathbb{R}^n \to \mathbb{R}$ duas vezes diferenciável e convexa sobre $\mathbb{R}^n$ e assumimos que para um dado $x_0 \in \mathbb{R}^n$, o *conjunto das superfícies de nível* $L(x_0)$ é limitado, onde

$$L(x_0) = \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\}$$

Suponhamos que x_k é tal que $x_{k+1} = x_k - \lambda_k H_k \nabla f(x_k)$ e que λ_k e H_k são escolhidos por uma das seguintes maneiras:

(a) Numa "procura linear exata" e "DFP update" .

(b) $f(x_k + \lambda_k p_k) \leq f(x_k) + \alpha\lambda_k \langle \nabla f(x_k), p_k \rangle$, $\alpha \in (0, 1/2)$.

$\langle \nabla f(x_k + \lambda_k p_k), p_k \rangle \geq \beta \langle \nabla f(x_k), p_k \rangle$, $\beta \in (\alpha, 1)$

e "BFGS update" (tomamos $p_k = H_k \nabla f(x_k)$) .

Então, para alguma matriz $H_0 \in L(\mathbb{R}^n)$ simétrica positiva definida e $\varepsilon > 0$, existe $k > 0$ tal que $\|\nabla f(x_k)\| < \varepsilon$.

A parte (a) deste teorema foi demonstrado por Powell - (1971) [10], (1972). A parte (b), mais interessante, pois nos garante que p_k é uma boa direção de descida, foi provada, também por Powell, recentemente (1976) [9, 11] .

Um resultado muito interessante, principalmente em se tratando de cálculos computacionais, e que foi demonstrado por - Schuller [13] (1970), é o seguinte:

"Se $f : \mathbb{R}^n \to \mathbb{R}$, então a "ordem de convergência" (*) do Método de Newton é a *raiz positiva* t do polinômio $t^{n+1} - t^n - 1 = p(t)$."

Obviamente, da mesma forma que o método, o algorítmo para

(*) Ver Capítulo 3 - Parágrafo 3.1 .

"Quase-Newton" difere do algorítmo para Newton na sua "busca unidimensional", onde devemos ter

PASSO 5: Resolver $B_k z = -\nabla f(x_k)$, por (3), ou ainda conforme a escolha,

PASSO 5: Resolver $H_k^{-1} z = -\nabla f(x_k)$, por (4).

PASSO 6: $x_{k+1} = x_k + \lambda_k z$, onde calculamos λ_k de tal forma que encontramos o melhor valor da função para esta iteração.

3) O MÉTODO DOS GRADIENTES CONJUGADOS:

Assim como se explicita o título deste parágrafo, o método dos Gradientes Conjugados se baseia em encontrarmos uma direção conjugada d_k , que definiremos abaixo, pertencente a um subespaço gerado por vetores *mutuamente conjugados* (veremos a sua definição) tal que $x_{k+1} = x_k + \alpha_k d_k$, onde α_k é um escalar deste mesmo subespaço (como até agora, o "índice k" se refere à k-ésima iteração do algorítmo em questão, neste caso ainda não construído).

3.1) Seja $f : \mathbb{R}^n \to \mathbb{R}$ diferenciável, de classe c^3. Como já vimos, através do seu desenvolvimento de Taylor na vizinhança de um ponto x_0 (estamos chamando de "V" tal vizinhança), sabemos que para qualquer x pertencente a V, "f" se aproxima de uma função quadrática, ou seja, "f" é aproximável por uma função da forma

$$g(x) = \frac{1}{2} x^t Gx + b^t x + c$$

onde G é uma matriz simétrica de ordem n e $b \in \mathbb{R}^n$. Temos então o seguinte princípio heurístico que justifica a construção dos mé-

todos de minimização sem restrições:

"Se toda função diferenciável é aproximável, localmente, por uma função quadrática, então um bom método para minimizar funções quadráticas deverá ser um bom método para minimizar funções diferenciáveis em geral."

Vejamos como isto é feito no caso dos métodos de tipo - "gradientes conjugados":

DEFINIÇÃO DE DIREÇÕES CONJUGADAS: Seja $G \in \mathbb{R}^{nxn}$ simétrica definida positiva (isto é, $G > 0$) e d_1 e d_2 vetores do $\mathbb{R}^n$. Dizemos que d_1 e d_2 são conjugados com respeito a G (ou simplesmente "conjugados" ou "G-ortogonais") se e somente se $d_1^t G d_2 = 0$ (assim, dois vetores são ortogonais se são conjugados com respeito à identidade I) .

Os teoremas 1 e 2 abaixo, são muito importantes para a nossa teoria, pois fazem parte da sua base. Vejamos:

TEOREMA 1: Seja $\{d_1, d_2, \ldots, d_p\}$ um conjunto de vetores não nulos e conjugados, ou seja, $d_i^t G\, d_j = 0$, se $i \neq j$, para uma certa matriz $G > 0$. Então este conjunto é linearmente independente.

Este resultado é fácil de ser demonstrado, pois é consequência direta de ser $d_k^t G\, d_k \neq 0$, para $d_k \neq 0$.

TEOREMA 2: Seja $f : \mathbb{R}^n \to \mathbb{R}$ tal que $f(x) = \frac{1}{2} x^t Gx + b^t x + c$, G positiva definida, $x_0 \in \mathbb{R}^n$, $d \in \mathbb{R}^n$ com $d \neq \vec{0}$. Seja α^* tal que

$$f(x_0+\alpha^* d) = \min_{\alpha \in \mathbb{R}} \{f(x_0+\alpha d)\}$$

Então

$$\alpha^* = \frac{-[\nabla f(x_0)]^t d}{d^t G\, d}$$

Para provarmos este teorema, consideramos a função $\varphi : \mathbb{R} \to \mathbb{R}$ f.q. $\varphi(\alpha) = f(x_0+\alpha d)$, $\alpha \in \mathbb{R}$ e a ela aplicamos a regra da cadeira [6].

Com este resultado, podemos observar que se "d" é uma direção de descida, $\alpha^* > 0$; se é de subida, $\alpha^* < 0$; e se x_0 é o mínimo de "f", $\alpha^* = 0$. (*)

Como vimos, o Teorema 1 nos dá condições para conseguirmos uma *base* de direções conjugadas $\{d_i\}_{i=0}^{n-1}$. Chamamos então de $\mathcal{B}_k$ o subespaço gerado por $\{d_0,\ldots,d_{k-1}\}$ e daí podemos definir $x_0 + \mathcal{B}_k$ como sendo a *variedade* paralela a $\mathcal{B}_k$ que passa por x_0 (ou ainda, o "conjunto soma" de x_0 e $\mathcal{B}_k$), onde $x_0 \in \mathbb{R}^n$.

Pelo Teorema 2, $x_{k+1} = x_k+\alpha_k d_k$ com

$$\alpha_k = \frac{-g_k^t\, d_k}{d_k^t\, G\, d_k}$$

onde $g_k = \nabla f(x_k)$.

Sabemos que, sendo uma variedade linear, $x_0 + \mathcal{B}_k$ é um conjunto convexo do $\mathbb{R}^n$; logo, "f" restrita a $x_0 + \mathcal{B}_k$ tem certamente um único mínimo.

Se tomamos um ponto z pertencente à variedade $x_0 + \mathcal{B}_k$, o subespaço tangente a $x_0 + \mathcal{B}_k$ passando por z é justamente o subespaço $\mathcal{B}_k = [d_0,\ldots,d_{k+1}]$. Logo, a condição necessária e su-

---

(*) *Seja* $d \in \mathbb{R}^n$, $d \neq \vec{0}$. *Dizemos que d é uma direção de descida com relação a* $x_0 \in \mathbb{R}^n$, *quando* $\langle \nabla f(x_0), d\rangle < 0$. *Caso* $\langle \nabla f(x_0), d\rangle > 0$, *então d é uma direção de subida.*

ficiente para que z seja o mínimo de f em $x_0 + \mathcal{B}_k$ é que

$$\langle \nabla f(z), d_0 \rangle = 0, \ldots, \langle \nabla f(z), d_{k-1} \rangle = 0$$

ou seja, que $\nabla f(z) \perp \mathcal{B}_k$

Dessa forma, podemos enunciar e demonstrar o

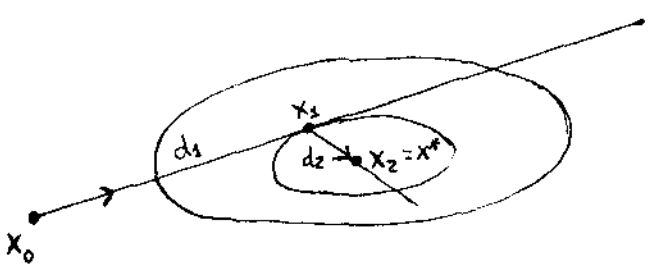
TEOREMA 3: Seja $f : \mathbb{R}^n \to \mathbb{R}$ diferenciável. Então, para $k \geq 0$, x_k é o mínimo de f restrita a $x_0 + \mathcal{B}_k$. Em particular, devemos ter que x_n é o mínimo global de f.

A prova deste teorema é simples e usa indução. Para $k = 1$, temos $\mathcal{B}_k = [d_1]$ e então o teorema anterior prova este resultado. Suponhamos que este teorema seja verdadeiro para um certo "k" e, usando o fato de que $d_k^t G d_i = 0$ e

$$g_{k+1} = g_k + (g_{k+1} - g_k) = g_k + \alpha_k G d_k$$

chegamos que $g_{k+1} \perp [d_0, d_1, \ldots, d_k]$. Logo, daqui podemos concluir que a afirmação feita acima é verdadeira.

Este resultado nos diz que, ao minimizarmos uma função quadrática por meio de direções conjugadas, encontramos, a cada passo, o mínimo da função restrita à variedade linear correspondente e que num número máximo de *n* "passos" obtemos o seu *mínimo global*. Isto nos é muito importante, posto que assegura a *convergência do algoritmo* para funções quadráticas.



Dessa maneira, podemos escrever os passos do algorítmo para o método dos gradientes conjugados da seguinte forma:

PASSO 1: x_0 : ponto inicial arbitrário

$d_0 = -g_0$.

PASSO 2: $x_{k+1} = x_k + \alpha_k d_k$, onde $\alpha_k = \dfrac{-g_k^t d_k}{d_k^t G d_k}$.

PASSO 3: Se $g_{k+1} = 0$, pare.

Senão, vá para PASSO 4.

PASSO 4: $d_{k+1} = -g_{k+1} + \gamma_k d_k$, onde $\gamma_k = \dfrac{g_{k+1}^G d_k}{d_k^t G d_k}$.

Vá para PASSO 2.

Outro resultado muito inportante é o

TEOREMA 4: Se o algorítmo dos Gradientes Conjugados para funções quadráticas não termina em x^k (i.e., não converge na iteração k-1), então:

(a) $[g_0,\ldots,g_k] = [g_0, Gg_0,\ldots,G^k g_0]$

(b) $[d_0,d_1,\ldots,d_k] = [g_0, Gg_0,\ldots,G^k g_0]$

(c) $d_k^t G d_i = 0$, para $i \leq k-1$.

As partes (a) e (b) são importantes teoricamente e servem para nos ajudar a demonstrar a parte (c), a qual nos é realmente importante, pois mostra que o algorítmo gera direções conjugadas e, portanto, serve para achar o mínimo de funções quadráticas num máximo de "n passos". As três partes são demonstradas indutivamente e podemos encontrar suas demonstrações no trabalho de David G.

Luenberger - 1973, [6].

3.2) GENERALIZAÇÃO PARA FUNCÕES NÃO-QUADRÁTICAS:

Podemos reformular o algorítmo de "Gradientes Conjugados" da seguinte maneira: fazemos

$$Gd_k = \frac{1}{\alpha_k} G(x_{k+1} - x_k) = \frac{1}{\alpha_k}\left[(Gx_{k+1}+b) - (Gx_k+b)\right] = \frac{1}{\alpha_k}(g_{k+1} - g_k)$$

e então obtemos o PASSO 4 na forma:

PASSO 4: $d_{k+1} = -g_{k+1} + \gamma_k d_k$, onde $\gamma_k = \dfrac{g_{k+1}^t (g_{k+1} - g_k)}{d_k^t (g_{k+1} - g_k)}$

(Fórmula de Sorenson-Wolfe)

Como podemos observar, o algorítmo, dessa forma, não conserva características de modelo quadrático; portanto, o mesmo pode ser aplicado para funções *não quadráticas*, esperando que os teoremas citados anteriormente também assegurem, de alguma forma, a sua convergência.

Existem outras fórmulas para γ_k, como as de Fletcher , Reeves, Polak-Rebiére, que são equivalentes à citada aqui, quando tratam de funções quadráticas, porém geram diferentes algorítmos para outras funções.

Na prática computacional, vamos utilizar a implementação de Shanno-Phua, 1978.

Algumas observações podem ser feitas aqui:

(a) Os erros de arredondamento têm uma influência grande nos métodos de gradientes conjugados. Por causa deles o algorítmo pode não convergir em "n passos" para as funções quadráticas, na

prática computacional. Em consequência, a cada n passos o algorítmo se "reinicia" pondo $d_n = -g_n$, $d_{2n} = -g_{2n}$, etc., até obter convergência.

(b) As direções d_k geradas por este método são, na maior parte das vezes, direções de descida. Por isso, pudemos substituir no método, o problema

$$f(x_k + \alpha_k d_k) = \min \{f(x_k + \alpha d_k), \quad \alpha \geq 0\}$$

(c) A importância dos métodos de gradientes conjugados é que usam pouca memória, só precisando guardar uns poucos vetores de n posições e *nenhuma matriz*. Isto faz com que os mesmos sejam eficientes para problemas grandes, embora para problemas pequenos hajam outros mais eficientes.

(d) Eles têm convergência quadrática para cada "n passos", isto é, existe uma constante $k > 0$ t.q. $\|x_{(k+1)n} - x^*\| \leq k\|x_k - x^*\|^2$ se x_0 está suficientemente próximo de x*. (Shanno, D.F.)

Trataremos agora, de alguns comentários a respeito dos três métodos acima descritos.

Enquanto o método de gradientes conjugados precisa, para ser implementado, de aproximadamente $7n + 2$ posições de memória, o método de Newton precisa de aproximadamente $\frac{n^2}{2} + \frac{11n}{2}$ posições e isto se deve à necessidade de que o último possui em armazenar a matriz hessiana. Numa iteração do método de gradientes conjugados são gastos, aproximadamente, $38n + 0(1)$ operações, além da busca unidimensional. No entanto, no método de Newton são gastas

$\frac{n^3}{6} + 1.5n^2 + 5.5n + 0(1)$, a busca unidimensional e o cálculo do hessiano. Dessa forma, podemos adiantar que, para n grande, o método de Newton pode ser impraticável (veremos isso mais adiante).

Quanto ao método Quase-Newton, este gasta, além da busca unidimensional, aproximadamente $2.5n^2 + 11.5n + 0(1)$ produtos e somas, a cada iteração, e precisa do mesmo número de posições de memória que o método de Newton.

CAPÍTULO II

FORMAS DE RESOLUÇÃO PARA O PROBLEMA DA NÃO-POSITIVIDADE DA MATRIZ HESSIANA PARA O MÉTODO DE NEWTON

Uma das propostas deste nosso trabalho, como vimos na introdução, é a de encontrarmos algorítmos para o método de Newton que sejam tão ou mais eficientes quanto o algorítmo já descrito, o qual utiliza direções de máxima descida quando o hessiano da função no ponto em questão, G(x), é não-positivo definido.

Ambos os procedimentos, aqui descritos, se utilizam da *procura de direções de curvatura negativa* para conseguir "direções de descida" no problema de minimização de funções, sendo que o primeiro, surgido de idéias de Gill e Murray, 1972, propõe uma aproximação da matriz hessiana a uma matriz positiva definida através de sua soma com uma matriz diagonal, apenas buscando uma direção de curvatura negativa no caso desta mesma matriz se apresentar *indefinida*; o segundo procedimento, também de Gill e Murray, transforma este problema num problema quadrático com restrições "canalizadas", isto é, limita a região em que o ponto da função em questão é um ponto crítico (no caso, ponto de sela ou de máximo) e faz uma aproximação local da função por uma função quadrática, buscando, daí, uma solução para o problema, agora restrito.

Vejamos então, quais são esses dois procedimentos.

2.1) MÉTODO DE NEWTON COM FATORIZAÇÃO DE CHOLESKI MODIFICADA:

2.1.1) Sabemos que se G é uma matriz positiva definida, então podemos escrevê-la na forma $G = LDL^t$, onde L é uma matriz triangu

lar inferior unitária e D uma matriz diagonal. Dessa forma, se chamarmos g_{ij}, ℓ_{ij} e d_j os ij-ésimos elementos e jj-ésimo elemento de $G^{(k)}$, $L^{(k)}$ e $D^{(k)}$, respectivamente, então temos que

$$(I) \quad d_j = g_{jj} - \sum_{r=1}^{j-1} \ell_{jr}^2 \, d_r$$

$$(II) \quad \ell_{ij} = (g_{ij} - \sum_{r=1}^{j-1} \ell_{jr} \, \ell_{ir} \, d_r) \; / \; d_j, \quad i = j+1,\ldots,n$$

Se denotarmos $c_{jr} = \ell_{jr} d_r$, então (I) e (II) ficam:

$$(I') \quad d_j = g_{jj} - \sum_{r=1}^{j-1} \ell_{jr} \, c_{jr}$$

$$(II') \quad c_{ij} = g_{ij} - \sum_{r=1}^{j-1} \ell_{jr} \, c_{ir}, \quad i = j+1,\ldots,n$$

Quando a matriz hessiana G não é positiva definida, devemos encontrar $d_j < 0$ para algum j (*). O que queremos, então, é aproximar cada $d_j < 0$ a um número positivo de tal forma que não tenhamos problemas de estabilidade numérica para os cálculos da nova fatorização; ou seja, temos, neste caso, que limitar o tamanho de cada d_j em questão. Quanto aos ℓ_{ij}, observamos pela equação (II) que, no caso dos mesmos se apresentarem muito grandes, podemos reduzi-los em módulo por um decréscimo dos elementos de $D^{(k)}$.

Assim, como podemos verificar mais adiante, esta idéia se resume em obtermos a fatorização de uma matriz positiva definida $\bar{G}$

(*) Podemos verificar isto nos exemplos de § 2.2.

onde

$$\bar{G}^{(k)} = G^{(k)} + E^{(k)}$$

e $E^{(k)}$ uma matriz diagonal *a qual é zero quando* $G^{(k)}$ *for suficientemente positiva definida.*

2.1.2) Primeiramente, vamos ver como devemos proceder de forma que não haja problemas numéricos consequentes da modificação de G.

Suponhamos β uma constante tal que

$$\left|\ell_{rs}\, d_s^{1/2}\right| \leq \beta\,, \quad \begin{matrix} s = 1,\ldots,j-1 \\ r = 1,\ldots,n \end{matrix}$$

Definimos

$$\phi_j = g_{jj} - \sum_{r=1}^{j-1} \ell_{jr}\, c_{jr}$$

$$\bar{d}_j = \max\{\delta,\ |\phi_j|\}$$

onde δ é introduzido para evitar dificuldades numéricas na avaliação da direção $p^{(k)}$, quando $G^{(k)}$ é positiva definida mas "mal condicionada". (*) Podemos escolher $\delta = \max\{2^{-t}\,\|G^{(k)}\|,\ 2^{-t}\}$, onde 2^{-t} é a precisão da máquina.

Vemos que o nosso problema, agora, se resume em determinarmos um valor para β de tal forma que $\bar{d}_j$ vai ou não ser modificado se os elementos-coluna $\ell_{ij}\bar{d}_j^{1/2}$, $i = j+1,\ldots,n$, forem ou não limitados por β.

Antes de determinarmos tal valor para β observamos que se definimos $\theta = \max\{|c_{ij}|,\ i = j+1,\ldots,n\}$, então para $\theta^2 \leq \beta^2\bar{d}_j$

(*) *"Mal condicionada", aqui, se refere ao mal condicionamento do sistema relativo a esta matriz* $(G^{(k)})$.

os elementos $\ell_{ij}\,\bar{d}_j^{1/2}$ são limitados por β. Assim podemos escolher d_j tal que o maior elemento em módulo de $\ell_{ij}\,d_j^{1/2}$ é exatamente igual a β, ou seja,

$$d_j = \max \{\bar{d}_j,\ \theta^2|\beta^2\}$$

Desde que

$$d_j = \begin{cases} \delta\ , \text{ se } \delta \geq \max \{|\phi_j|,\ \theta^2|\beta^2\} \\ |\phi_j|\ , \text{ se } |\phi_j| \geq \max \{\theta^2|\beta^2,\ \delta\} \\ \theta^2|\beta^2, \text{ se } \theta^2|\beta^2 \geq \max \{\delta,\ |\phi_j|\} \end{cases}$$

então

$$\text{(III)} \qquad d_j = g_{jj} + E_j - \sum_{r=1}^{j-1} \ell_{jr}\, c_{jr}$$

onde

$$E_j = \begin{cases} \delta - \phi_j\ , \text{ se } \delta \geq \max \{|\phi_j|,\ \theta^2|\beta^2\} \\ |\phi_j| - \phi_j\ , \text{ se } |\phi_j| \geq \max \{\theta^2|\beta^2,\ \delta\} \\ \theta^2|\beta^2 - \phi_j, \text{ se } \theta^2|\beta^2 \geq \max \{\delta,\ |\phi_j|\} \end{cases}$$

Observando a equação (III) podemos dizer que, dessa forma, o elemento diagonal da matriz fatorizada é dado por $g_{jj} + E_j$ e que podemos obtê-lo, da mesma forma, utilizando a fatorização de Choleski da matriz

$$\bar{G}^{(k)} = G^{(k)} + E^{(k)}$$

onde $E^{(k)}$ é uma matriz diagonal com j-ésimo elemento E_j .

Portanto, podemos afirmar que para encontrarmos uma direção de descida devemos resolver o sistema

$$\left[G^{(k)} + E^{(k)}\right] p^{(k)} = -g^{(k)}$$

onde $g = \nabla f$.

Finalmente podemos citar o teorema que nos garante um valor para a constante β:

TEOREMA 1: Seja $G^{(k)}$ matriz simétrica com elementos limitados. O j-ésimo elemento-diagonal da matriz $E^{(k)}$ associado com a fatorização de Choleski de $G^{(k)}$ é limitado e satisfaz

$$|E_j| \leq (\xi_j/\beta + (j-1)\beta)^2 + 2(|g_{jj}| + (j-1)\beta^2) + \delta$$

onde $\xi_j = \max \{|g_{ij}|,\ i = j+1,\ldots,n\}$. (A prova deste teorema pode ser encontrada em [5].)

Deste resultado podemos concluir que

(a) O limite máximo possível para $\|E^{(k)}\|_\infty$ (*) é dado por

$$\|E^k\|_\infty \leq (\xi/\beta + (n-1)\beta)^2 + 2(\gamma + (n-1)\beta^2) + \delta$$

onde $\xi = \max \{|g_{ij}|,\ i \neq j\}$ e $\gamma = \max \{|g_{jj}|,\ j = 1,\ldots,n\}$.

(b) Se chamarmos de $\varphi(\beta)$ o limite acima definido, notamos que φ é uma função convexa e o seu mínimo ocorre quando $\beta^2 = \xi/\sqrt{n^2-1}$ e que β assim definido é ainda suficientemente grande para garantir a não modificação de G se esta for positiva definida. Na prática computacional é viável escolhermos $\beta^2 = \xi/n$.

Da fórmula (I) acima, temos que

$$d_j = g_{jj} - \sum_{r=1}^{j-1} \ell_{jr}^2\, d_r = g_{jj} - \sum_{r=1}^{j} \ell_{jr}^2\, d_r + \ell_{jj}\, d_j$$

(*) *Definimos* $\|E^k\|_\infty = \max\limits_{1 \leq i \leq n} \sum\limits_{j=1}^{n} |E_{ij}^k| = \max \{E_j^k,\ j = 1,\ldots,n\}$.

com $\ell_{jj} = 1$, donde tiramos $g_{jj} = \sum_{r=1}^{j} \ell_{jr}^2 d_r$ o que implica $\ell_{jr}^2 d_r \leq g_{jj}$ para cada j; daí, se tivermos

$$\beta^2 \geq \max_j \{|g_{jj}|,\ j = 1,\ldots,n\} = \gamma$$

teremos que $\ell_{jr}^2 d_r \leq \beta^2$ e então nenhuma modificação na matriz G será necessária.

Finalmente, para que β seja um valor "seguro", podemos escolher

$$\beta^2 = \max \{\gamma,\ \xi/n,\ 2^{-t}\}$$

onde 2^{-t} é a precisão da máquina e é aqui introduzida para o caso de $\|G^k\|_\infty = 0$.

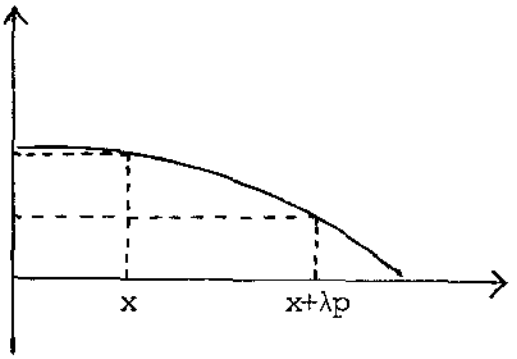
Novamente, devemos salientar aqui a importância da escolha de β para a estabilidade do método.

Outro problema, relacionado com a não-positividade da matriz hessiana, é do algorítmo estar num ponto de sela ou num ponto de máximo; neste caso não podemos encontrar uma direção de descida como solução do sistema $G^{(k)} p = -g$, pois certamente ela se anulará $(g = \vec{0})$.

Para este problema, procuramos uma direção de curvatura - negativa, que é

2.1.3) UMA ALTERNATIVA:

DEFINIÇÃO: Uma direção $p \in \mathbb{R}^n$ é uma *direção de curvatura negativa* se e só se $p^t G\, p < 0$, ou seja, na direção "p" a função em questão deve ser côncava; por exemplo:



OBSERVAÇÃO: Para os cálculos da direção p utilizaremos informações sobre as modificações necessárias na matriz hessiana com sua respectiva fatorização.

Vejamos agora alguns resultados importantes para este procedimento:

LEMA 1: Seja s um inteiro positivo tal que $\phi_s \leq \phi_j$, $j = 1,\ldots,n$. Se $G^{(k)}$ é indefinida, então $\phi_s \leq 0$.

LEMA 2: Seja $x^{(k)}$ tal que $\|g^{(k)}\| = 0$ e $G^{(k)}$ indefinida. Seja p a solução da equação $(L^{(k)})^t p = e_s$, onde $s \in z_+$ t. q. $\phi_s \leq \phi_j$, $j = 1,\ldots,n$. Então p é uma direção de curvatura negativa.

Podemos encontrar as provas desses resultados em [5] .

Dessa maneira, podemos agora recordar o PASSO 4 do algorítmo de Newton, descrito no capítulo I, e modificá-lo da seguinte forma:

PASSO 4: I - Se $G^{(k)}$ é não positiva definida e $\|g^{(k)}\| = 0$, vá para II. Senão, $G^{(k)} + E^{(k)} = L^{(k)} D^{(k)} L^{(k)t}$ e

vá para PASSO 5.

II - $L^{(k)t}y = e_j$; j t.q. $d_j^{(k)} - E_j^{(k)} \leq d_i^{(k)} - E_i^{(k)}$, e tomamos

$$p^{(k)} = \begin{cases} - \text{ sinal } (y^t g^{(k)}) y \text{ , se } \|g^{(k)}\|_2 \neq 0 & (*) \\ y \text{ , se } \|g^{(k)}\|_2 = 0 \end{cases}$$

O teorema 3, abaixo, nos garante a convergência de tal algorítmo.

Seja $F : \Gamma \subset \mathbb{R}^n \to \mathbb{R}$, Γ conjunto aberto. Denotaremos por $\bar{\Omega}(t)$ o conjunto fechado das superfícies de nível

$$\Omega(t) = \{x \mid x \in \Gamma , \ F(x) < t\}$$

e Co $[\bar{\Omega}]$ representa $\bar{\Omega}$ convexo .

TEOREMA 2: Seja $F : \Gamma \subset \mathbb{R}^n \to \mathbb{R}$ uma função continuamente diferenciável para todo $x \in \Gamma$ e seja $s = \{\hat{x};\ \hat{x} \in \Omega,\ g(\hat{x}) = 0\}$. Se x_0 é escolhido tal que

(i) $\bar{\Omega}[F^{(0)}]$ compacto

(ii) Co $[\Omega\{F^{(0)}\}] \subset \Gamma$

(iii) a matriz hessiana

G(x) é tal que $\|G(x)\| \leq \rho$ para todo $x \in \bar{\Omega}\{F^{(0)}\}$. Se $\varepsilon = 0$ no nosso algorítmo, então a sequência $\{x_k\}_{k=0}^{\infty}$ gerada pelo mesmo é tal que

---

(*) *Definimos* $\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2 \right)^{1/2}$ *onde* $x = (x_1, \ldots, x_n)$.

$$\lim_{k\to\infty} x_k = \hat{x} \ , \quad \text{onde} \quad \hat{x} \in s \ . \qquad (*)$$

TEOREMA 3: Considerando as mesmas condições do teorema anterior , seja $\epsilon > 0$ um escalar muito pequeno. Então a sequência de pontos $\{x_k(\epsilon)\}_{k=0}^{\infty}$ gerada pelo algorítmo é tal que

$$\lim_{\epsilon\to 0} \lim_{k\to\infty} \{x_k(\epsilon)\} = x^*$$

onde x* é um *mínimo local* de F. (*)

Vejamos agora o que vem a ser o segundo procedimento:

2.2) O MÉTODO DE NEWTON CANALIZADO - PROGRAMAÇÃO QUADRÁTICA:

2.2.1) Seja $p \in \mathbb{R}^n$ uma direção de curvatura negativa.

Como no nosso problema estamos interessados na busca do mínimo da função, então também consideramos que $\langle g,p\rangle < 0$, ou seja, p também deve ser uma *direção de descida.*

Seja $f : \mathbb{R}^n \to \mathbb{R}$ diferenciável e $(x_1,\ldots,x_n)^t \in \mathbb{R}^n$.

Dado o problema

$$\text{Min } f(x)$$
$$\text{s.a } x \in \mathbb{R}^n$$

em cada iteração k ,

$$f(x) \cong f(x_k) + \langle \nabla f(x_k), (x-x_k)\rangle +$$
$$+ \frac{1}{2} (x-x_k)^t G(x_k)(x-x_k) = \frac{1}{2} x^t Gx + b^t x + c$$

e podemos gerar, por algum critério heurístico, "limites" u_i e ℓ_i

---

(*) *Ver prova em* [5]. *(Estamos considerando ρ um escalar positivo.)*

tais que o problema de acharmos x_{k+1} se transforma em

$$\text{Min} \quad \frac{1}{2} x^t G x + b^t x + c$$

$$\text{s.a} \quad \ell_i \leq x_i \leq u_i$$

Se $u_i = +\infty$ e $\ell_i = -\infty$ e $G > 0$, então a solução x^{k+1} é simplesmente

$$x_{k+1} = x_k - \lambda_k [\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$$

Definiremos uma *variável livre* como sendo uma variável x_i tal que x_i *não* satisfaz as seguintes condições:

(a) $x_i = \ell_i$ e $g_i > 0$

(b) $x_i = u_i$ e $g_i < 0$

onde g_i é a componente do gradiente relativa à variável x_i.

As que satisfazem (a) ou (b) são ditas *variáveis fixas*.

Vamos agora explicitar a maneira com que calculamos a direção de curvatura negativa. Vejamos:

2.2.2) Seja t o número de variáveis fixas de um dado problema. Chamaremos de $z_{n \times (n-t)}$, onde n é o número de variáveis do problema, a matriz que define o subespaço das variáveis *não-fixas* - (das livres).

Então, para cada iteração k, definimos $z^{(k)}_{n \times (n-t)}$ e temos:

(a) $\bar{g}^k = (z^k)^t g^k$ como o gradiente de f(x) com respeito às variáveis livres - *gradiente projetado* no espaço das variáveis livres.

(b) $\bar{G}^k = (z^k)^t G z^k$ como o hessiano de f(x) com respeito às va-

riáveis livres - *hessiano projetado* no espaço das variáveis livres; logo, $\bar{G}^k$ tem ordem $(n-t) \times (n-t)$.

Chamamos de $L^k_{(n-t)x(n-t)}$ e $D^k_{(n-t)x(n-t)}$ as matrizes - com relação à fatorização de Choleski de $\bar{G}^k$, onde L^k é uma matriz triangular inferior unitária e D^k é diagonal.

A direção a ser calculada é "p". Então temos p tal que, a cada iteração k,

$$p^k = \pm \bar{z}^k y^k$$

onde $y^k / L^k y^k = e_{(n-t)}$, sendo $e_{(n-t)}$ o vetor que possui sua (n-t)-ésima componente igual a 1 e as outras nulas.

OBSERVAÇÃO: 1) $\bar{z}^k$ significa que não trabalharemos com todas as variáveis livres, na iteração k (veremos mais adiante).

2) Se p^k não for uma direção de descida, então $(-p^k)$ o será.

2.2.3) A cada iteração k, temos que classificar as variáveis livres e, a partir daí, fazer a fatorização de Choleski do hessiano com relação a estas variáveis a fim de escolhermos as que deverão ser "mexidas" (nem todas as variáveis livres devem entrar para os cálculos; falaremos disto mais adiante); ou seja, para $n = 4$ se

$$G = \begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{12} & g_{22} & g_{23} & g_{24} \\ g_{13} & g_{23} & g_{33} & g_{34} \\ g_{14} & g_{24} & g_{34} & g_{44} \end{bmatrix}$$

e se, na iteração k, x_2 for uma variável fixa, então

$$\bar{G}^k = \begin{bmatrix} g_{11} & g_{13} & g_{14} \\ g_{13} & g_{33} & g_{34} \\ g_{14} & g_{34} & g_{44} \end{bmatrix} = \begin{bmatrix} \bar{g}_{11} & \bar{g}_{12} & \bar{g}_{13} \\ \bar{g}_{12} & \bar{g}_{22} & \bar{g}_{23} \\ \bar{g}_{13} & \bar{g}_{23} & \bar{g}_{33} \end{bmatrix} = (L^k)\ D^k\ (L^k)^t$$

onde D^k é a matriz diagonal $\begin{bmatrix} d_1^k & & \\ & d_2^k & \\ & & d_3^k \end{bmatrix}$.

Se na fatorização de Choleski encontrarmos algum $d_i^k < 0$, então escolhemos as i's primeiras variáveis livres para trabalharmos na iteração; neste exemplo supomos que x_1, x_3 e x_4 são variáveis livres e, ainda supondo, se encontrarmos $d_2^k < o$, então devemos trabalhar apenas com x_1 e x_3 .

A idéia do algorítmo é a seguinte:

PASSO 1: Dado um ponto $x = (x_1,\ldots,x_n)^t$, classificar as variáveis: FIXAS - quando $x_i = \ell_i$ e $g_i > 0$ ou

$x_i = u_i$ e $g_i < 0$

LIVRES - as outras.

PASSO 2: Se todas as variáveis são fixas, parar; o ponto é ótimo.

PASSO 3: Considerar o hessiano em relação às variáveis livres. Se ele é semidefinido positivo, vá para PASSO 5.

PASSO 4: Conseguir uma direção de curvatura negativa no espaço das variáveis livres. Aqui convêm distinguir variáveis livres das "mexidas". Seguir esta direção, respeitando descida, até obter um novo ponto x e voltar para PASSO 1.

PASSO 5: Se as componentes do gradiente para as coordenadas das - variáveis livres são nulas, então o ponto é ótimo.Parar.

PASSO 6: Calcular a direção de Newton no espaço das variáveis livres. Obter um novo ponto. Voltar para PASSO 1.

2.2.4) Obviamente, a cada iteração, o número de variáveis livres pode diminuir ou aumentar, independente da direção que tomamos ser satisfatória ou não. Dessa maneira, teríamos, então, a cada iteração, que fazer novos cálculos para a fatorização de Choleski, mexendo com novas matrizes e, provavelmente, sobrecarregando a memória da máquina e aumentando o "custo computacional" do algorítmo. Para impedirmos que isto aconteça, podemos utilizar as idéias de Golub-Murray-Saunders, as quais permitem que façamos uso da fatorização de Choleski da matriz inicial, G, para obtermos as fatorizações de suas submatrizes, $\bar{G}^k$, a cada iteração k.

Para ilustrarmos tais idéias, vamos utilizar G de ordem 4, cujo ij-ésimo elemento é g_{ij}. Supomos G fatorizável por Choleski, isto é,

$$G = LDL^t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \begin{bmatrix} 1 & \ell_{21} & \ell_{31} & \ell_{41} \\ 0 & 1 & \ell_{32} & \ell_{42} \\ 0 & 0 & 1 & \ell_{43} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) Então, se tivermos, na iteração k, x_2 como variável fixa,

$$\bar{G}^k = \begin{bmatrix} g_{11} & g_{13} & g_{14} \\ g_{13} & g_{33} & g_{34} \\ g_{14} & g_{34} & g_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{31} & \ell_{32} & 1 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \begin{bmatrix} 1 & \ell_{31} & \ell_{41} \\ 0 & \ell_{32} & \ell_{42} \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix}$$

Podemos, daqui, escrever

$$\bar{G}^k = \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{31} & 0 & 1 & 0 \\ \ell_{41} & 0 & \ell_{43} & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \ell_{32} & 0 & 0 \\ 0 & \ell_{42} & 0 & 0 \end{bmatrix} \right) \left(\begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & d_2 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \right)$$

$$\left(\begin{bmatrix} 1 & \ell_{31} & \ell_{41} \\ 0 & 0 & 0 \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \ell_{32} & \ell_{42} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) = \left(\begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{31} & 0 & 1 & 0 \\ \ell_{41} & 0 & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} + \right.$$

$$\left. \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & d_2\ell_{32} & 0 & 0 \\ 0 & d_2\ell_{42} & 0 & 0 \end{bmatrix} \right) \left(\begin{bmatrix} 1 & \ell_{31} & \ell_{41} \\ 0 & 0 & 0 \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & \ell_{32} & \ell_{42} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) .$$

Logo

$$\bar{G}^k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \ell_{31} & 0 & 1 & 0 \\ \ell_{41} & 0 & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & d_3 & 0 \\ 0 & 0 & 0 & d_4 \end{bmatrix} \begin{bmatrix} 1 & \ell_{31} & \ell_{41} \\ 0 & 0 & 0 \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & d_2\ell_{32} & 0 \\ 0 & d_2\ell_{42} & 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & \ell_{32} & \ell_{42} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{31} & 1 & 0 \\ \ell_{41} & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_3 & 0 \\ 0 & 0 & d_4 \end{bmatrix} \begin{bmatrix} 1 & \ell_{31} & \ell_{41} \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix} +$$

$$+ \; d_2 \begin{bmatrix} 0 \\ \ell_{32} \\ \ell_{42} \end{bmatrix} \begin{bmatrix} 0, & \ell_{32}, & \ell_{42} \end{bmatrix} .$$

Portanto,

$$\bar{G}^k = (\bar{L}^k)\bar{D}^k(\bar{L}^k)^t + d_2 \, \ell^k(\ell^k)^t$$

onde $\ell^k = \begin{bmatrix} \ell_{12} \\ \ell_{32} \\ \ell_{42} \end{bmatrix}$.

Da mesma forma, se considerarmos x_1 como variável fixa , na iteração k, teremos

$$\bar{G}^k = \begin{bmatrix} g_{22} & g_{23} & g_{24} \\ g_{23} & g_{33} & g_{34} \\ g_{24} & g_{34} & g_{44} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{32} & 1 & 0 \\ \ell_{42} & \ell_{43} & 1 \end{bmatrix} \begin{bmatrix} d_2 & 0 & 0 \\ 0 & d_3 & 0 \\ 0 & 0 & d_4 \end{bmatrix} \begin{bmatrix} 1 & \ell_{32} & \ell_{42} \\ 0 & 1 & \ell_{43} \\ 0 & 0 & 1 \end{bmatrix} +$$

$$+ \; d_1 \begin{bmatrix} \ell_{21} \\ \ell_{31} \\ \ell_{41} \end{bmatrix} \begin{bmatrix} \ell_{21}, & \ell_{31}, & \ell_{41} \end{bmatrix} = (\bar{L}^k)\bar{D}^k(\bar{L}^k)^t + d_1 \ell^k(\ell^k)^t$$

onde $\ell^k = \begin{bmatrix} \ell_{21} \\ \ell_{31} \\ \ell_{41} \end{bmatrix}$.

Como, para chegarmos a este resultado, utlizamos apenas propriedades elementares de matrizes, que sabemos válidas para - qualquer ordem, podemos concluir desde já que:

Se x_j é uma variável fixa na iteração k, então

$$\bar{G}^k = (\bar{L}^k)\bar{D}^k(\bar{L}^k)^t + d_j \ell^k (\ell^k)^t$$

onde $\ell^k = [\ell_{ij}]_{n-1}$, $i \neq j$.

NOTAS: 1) Utilizamos a notação $\bar{L}^k$ para a matriz L projetada no subespaço das variáveis livres da iteração k, ou seja , $\bar{L}^k = (z^k)L(z^k)^t$ e d_j é o j-ésimo elemento da diagonal da matriz D.

2) Se fizermos $\bar{L}^k p^k = \vec{\ell}^k$, então

$$\bar{G}^k = (\bar{L}^k)\ (\bar{D}^k + d_j\ p^k(p^k)^t)\ (\bar{L}^k)^t$$

Logo, só precisamos calcular a fatorização de Choleski da matriz $\bar{D}^k + d_j\ p^k(p^k)^t$.

3) Por hipótese, a fórmula acima só se verifica para o caso em que o número de variáveis livres na iteração k for *menor* que o número de variáveis livres na iteração k-1 .

(b) Para o caso em que o número de variáveis livres na iteração k é *maior* do que o número de variáveis livres na iteração k-1 (ou seja, se ganhamos uma variável livre), podemos utilizar o seguinte reciocínio:

Desta vez, temos que acrescentar uma linha e uma coluna na matriz $\bar{G}^{k-1}$, e então

$$\bar{G}^k = \left[\begin{array}{c|c} \bar{G}^{k-1} & b \\ \hline b^t & \alpha \end{array}\right]$$

onde $(b,\alpha)^t$ é o vetor-coluna de G relativo à nova variável livre.

Sabemos que $\bar{G}^{k-1} = (\bar{L}^{k-1})\bar{D}^{k-1}(\bar{L}^{k-1})^t$. Então,

$$\bar{G}^k = (\bar{L}^k)\ D^k\ (L^k)^t$$

com

$$\bar{L}^k = \left[\begin{array}{c|c} \bar{L}^{k-1} & 0 \\ \hline \ell^t & 1 \end{array}\right] \quad \text{e} \quad D^k = \left[\begin{array}{c|c} D^{k-1} & 0 \\ \hline 0 & d \end{array}\right] ,$$

onde ℓ^t e d são valores a serem calculados. Ou seja,

$$\left[\begin{array}{c|c} \bar{G}^{k-1} & b \\ \hline b^t & \alpha \end{array}\right] = \left[\begin{array}{c|c} L^{k-1} & 0 \\ \hline \ell^t & 1 \end{array}\right] \left[\begin{array}{c|c} D^{k-1} & 0 \\ \hline 0 & d \end{array}\right] \left[\begin{array}{c|c} (L^{k-1})^t & \ell \\ \hline 0 & 1 \end{array}\right]$$

donde concluímos que $b = L^{k-1}D^{k-1}\ell$ e $d = \alpha - \ell^t D^{k-1}\ell$.

NOTA: Até agora tratamos apenas de situações elementares, como o caso de perda ou ganho de *uma* variável livre a cada iteração. Para o caso de perdermos ou ganharmos mais de uma variável livre, deveremos proceder da mesma forma.

2.2.5) Uma maneira prática de visualizarmos o funcionamento desta modificação do Método de Newton é fazermos alguns exemplos simples.

EXEMPLO 1: Seja $F(x) = \frac{1}{2} x^t Gx + bx$

s.a $-1 \leq x_i \leq 1$, $i = 1, 2, 2$ onde

$$G = \begin{bmatrix} 1 & 3 & 2 \\ 3 & -1 & 1 \\ 2 & 1 & -2 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

Tomamos $x^{(0)} = [0, 0, 0]$. Então $g^{(0)} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$.

PASSO 1: $x_1^{(0)}$, $x_2^{(0)}$ e $x_3^{(0)}$ são livres.

PASSO 2: Não Pare.

PASSO 3: Façamos então a fatorização de Choleski da matriz G, isto é, $LDL^t = G$, donde tiramos $d_1 = 1$, $\ell_{12} = 3$, $\ell_{13} = 2$ e $d_2 = -10$ (!). Como $d_2 < 0$, isto significa que devemos trabalhar com apenas as duas primeiras variáveis livres, ou seja, $x_1^{(0)}$ e $x_2^{(0)}$. Assim, calculamos a direção de curvatura negativa p:

PASSO 4: $p^{(0)} = \pm \bar{z}^0 y$ onde $(\bar{z}^0)^t y = e_2$, isto é ,

$$\begin{bmatrix} 1 & 3 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \begin{matrix} y_2 = 1 \\ y_1 = -3 \end{matrix}$$

Então

$$p^{(0)} = \pm \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \end{bmatrix} = [-3, 1, 0]$$

onde

$$[-3, 1, 0] \begin{bmatrix} 1 & 3 & 2 \\ 3 & -1 & 1 \\ 2 & 1 & -2 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix} = [0, -4, -5] \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix} < 0$$

Logo, p é uma direção de curvatura negativa, onde temos também que $\langle g^{(0)}, p^{(0)} \rangle < 0$ e $p^{(0)}$ é uma direção de

descida.

Calculamos, então, o ponto $x^{(1)}$:

$$x^{(1)} = x^{(0)} + \alpha^{(0)} p^{(0)}$$

onde

$$\alpha^{(0)} \Big/ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \alpha^{(0)} \begin{bmatrix} -3 \\ 1 \\ 0 \end{bmatrix}$$

com $-1 \leq x_i \leq 1$, ou seja, calculamos $\alpha^{(0)}$ máximo tal que $x^{(1)}$ é factível. Para este cado, $\alpha^{(0)} = \frac{1}{3}$.

Logo

$$x^{(1)} = [-1, \tfrac{1}{3}, 0]^t \quad \text{e} \quad g^{(1)} = [1, -4/3, -4/3]^t$$

PASSO 1: Agora, $x_2^{(1)}$ e $x_3^{(1)}$ são variáveis livres.

PASSO 2: Não pare.

PASSO 3: $\bar{G}^{(1)} = \begin{bmatrix} -1 & 1 \\ 1 & -2 \end{bmatrix}$ é não positiva definida. Logo, procuramos novamente "p". Primeiramente, devemos saber quais variáveis livres deverão ser mexidas, i.e.,

$$\bar{G}^{(1)} = \begin{bmatrix} 1 & 0 \\ \ell_{12} & 1 \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & d_2 \end{bmatrix} \begin{bmatrix} 1 & \ell_{12} \\ 0 & 1 \end{bmatrix} \Rightarrow d_1 = -1 < 0 .$$

Portanto, vamos trabalhar com $x_2^{(1)}$.

PASSO 4: $p^{(1)} = \pm \bar{z}^{(1)} y$, onde $y \;/\; (\bar{L}^{(1)}) y = \bar{e}_i \rightarrow y = 1$. Logo

$$p^{(1)} = \pm \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \quad \text{e para} \quad p^{(1)} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

obtemos $p^{(0)t} G\, p^{(1)} < 0$ e $\langle g^{(1)}, p^{(1)} \rangle < 0$.

Portanto, $x^{(2)} = x^{(1)} + \alpha^{(1)} p^{(1)}$, onde $\alpha^{(1)}$ é máximo tal que $x^{(2)}$ é factível, i.e., $\alpha^{(1)} = 2/3$ e temos

$$x^{(2)} = (-1,\ 1,\ 0)^t \ .$$

PASSO 1: $g^{(2)} = Gx^{(2)} + b = [3,\ -2,\ 0]^t$. Logo, $x_3^{(2)}$ é "livre".

PASSO 2: Não pare.

PASSO 3: $\bar{G}^{(2)}$ é não positiva definida.

PASSO 4: $p^{(2)} = \pm \bar{z}^{(2)} y$ onde $y = 1$; logo $p^{(2)} = \pm \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$ com $p^{(2)t} G\, p^{(2)} < 0$. Assim

$$x^{(3)} = x^{(2)} + \alpha^{(2)} p^{(2)}$$

onde $\alpha^{(2)} = 1$. Portanto ,

$$x^{(3)} = [-1,1,1]^t \quad \text{e} \quad g^{(3)} = [5,-1,-2] \ .$$

Logo, $x^{(3)}$ é um ponto de ótimo, pois todas as variáveis $x_1^{(3)}$, $x_2^{(3)}$ e $x_3^{(3)}$ são fixas e satisfazem o critério de parada.

Um outro exemplo, talvez um pouco mais interessante,pois utiliza também a "direção de Newton", é o seguinte:

EXEMPLO 2: Seja $F(x) = \frac{1}{2} x^t G\, x + bx$

$$\text{s.a.} \quad -1 \leq x_i \leq 1$$

onde

$$G = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & -5 \end{bmatrix} \qquad e \qquad b = \begin{bmatrix} 5 \\ 3 \\ 1 \end{bmatrix}$$

Analogamente ao exemplo anterior, seguimos os PASSOS do algorítmo:

Tomamos $x^{(0)} = (0,0,0)$ e temos $g^{(0)} = (5,3,1)^t$.

PASSO 1: $x_1^{(0)}$, $x_2^{(0)}$ e $x_3^{(0)}$ são variáveis livres.

PASSO 2: Não pare.

PASSO 3: G é não positiva definida, onde para $G = LDL^t$ temos

$$d_1 = 1 , \quad \ell_{13} = 1$$

$$\ell_{12} = 2 \qquad d_2 = -2 < 0$$

Portanto, trabalhamos com apenas $x_1^{(0)}$ e $x_2^{(0)}$.

PASSO 4: $p^{(0)} = \pm\bar{z}^{(0)}y$, onde $(L^{(0)})^t y = e_2 \Rightarrow y_1 = 1$

$y_2 = -2$

$\therefore$ $p^{(0)} = (-1,2,0)^t$ é tal que $p^{(0)} G (p^0)^t < 0$ e $\langle g^{(0)}, p^{(0)} \rangle < 0$. Daí ,

$$x^{(1)} = x^{(0)} + \alpha^{(0)} p^{(0)}$$

onde $\alpha^{(0)} = \frac{1}{2}$. Logo

$$x^{(1)} = (-\tfrac{1}{2}, 1, 0)$$

PASSO 1: $g^{(1)} = Gx^{(1)} + b = (\frac{13}{2}, 4, \frac{3}{2})^t$. Logo, $x_1^{(1)}$ e $x_3^{(1)}$ - são livres.

PASSO 2: Não pare.

PASSO 3: $\bar{G}^{(1)}$ é não positiva definida, pois $\bar{G}^{(1)} = \begin{bmatrix} 1 & 1 \\ 1 & -5 \end{bmatrix}$

PASSO 4: $p^{(1)} = \pm\bar{z}^{(1)}$, onde

$$(L^{(1)})^t y = e_2 \Rightarrow \begin{bmatrix} 1 & 2 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \begin{matrix} y_1 = 1 \\ y_2 = -2 \end{matrix} \quad \text{e}$$

Logo ,

$p^{(1)} = (1,0,-2)$ é t.q. $\langle g^{(1)}, p^{(1)} \rangle < 0$ e $p^t G p < 0$.

Assim

$$x^{(2)} = x^{(1)} + \alpha^{(1)} p^{(1)} \quad \text{com} \quad \alpha_1^{(1)} = 1/2 .$$

$$x^{(2)} = (0, 1, -1)^t$$

PASSO 1: $g^{(2)} = Gx^{(2)} + b = (6,4,7)^t$, donde tiramos que $x_1^{(2)}$ e $x_2^{(2)}$ são variáveis livres.

PASSO 2: Não pare.

PASSO 3: $\bar{G}^{(2)} = \begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix}$ é positiva definida.

PASSO 5: Não pare.

PASSO 6: Calculamos uma direção de Newton, ou seja,

$$x^{(3)} = x^{(2)} - z^{(2)}$$

onde $\bar{G}^{(2)} z^{(2)} = \bar{g}^{(2)}$, isto é,

$$\begin{bmatrix} 1 & 2 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 6 \\ 4 \end{bmatrix} \Rightarrow \begin{matrix} z_1^{(2)} = -2 \\ z_2^{(2)} = 4 \end{matrix}$$

Logo

$$x^{(3)} = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix} - \begin{bmatrix} -2 \\ 4 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ -3 \\ -1 \end{bmatrix}$$

onde podemos observar que $x^{(3)}$, assim calculado, é *não factível*.

Usando a mesma direção $z^{(2)}$, calculamos, então, $\alpha^{(2)}$ máximo para que $x^{(3)}$ seja factível. Ou seja,

$$x^{(3)} = x^{(2)} - \alpha^{(2)} z^{(2)}$$

onde $\alpha^{(2)} = \frac{1}{2}$.

Dessa forma, obtemos $x^{(3)} = (1,-1,-1)^t$, que é factível.

PASSO 1: $g^{(3)} = Gx^{(3)}+b = (3,2,6)^t$. Temos $x_1^{(3)}$ variável livre e, portanto, temos que dar mais um *passo de Newton*, ou seja, $x^{(4)} = x^{(3)}-z^{(3)}$, onde

$$z^{(3)} \; / \; \bar{g}^{(4)} z^{(3)} = \bar{g}^{(4)} \quad \Rightarrow \quad z^{(3)} = 3 .$$

Novamente, $x^{(4)}$ é não factível e então somos obrigados a recalculá-lo, isto é, $x^{(4)} = x^{(3)}-\alpha^{(3)} z^{(3)}$, onde $\alpha^{(3)} = 2/3$ e então obtemos $x^{(4)} = (-1,-1,-1)^t$, com $g^{(4)} = (1,-2,4)^t$.

Logo, temos novamente uma outra iteração, onde $x_2^{(4)}$ é "livre" e $\bar{G}^{(4)}$ positiva definida.

Portanto, $x^{(5)} = x^{(4)}-z^{(4)}$ onde $z = -1$. Assim,

$$x^{(5)} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ -1 \end{bmatrix}$$

com $g^{(5)} = \begin{bmatrix} 3 \\ 0 \\ 5 \end{bmatrix}$.

Aqui notamos que $x_2^{(5)}$ é livre mas $g_2^{(5)} = 0$ e $\bar{G}^{(5)} > 0$. Logo, $x^{(5)} = [-1,0,-1]^t$ deve ser o ponto de ótimo para o nosso problema.

Para $f : \mathbb{R}^4 \to \mathbb{R}$ temos o

EXEMPLO 3: Seja $F(x) = \frac{1}{2} x^t G x + bx$, onde

$$\text{s.a} \quad -1 \leq x_i \leq 1$$

$$G = \begin{bmatrix} 1 & 1 & 1 & 3 \\ 1 & -1 & 0 & 2 \\ 1 & 0 & -2 & 1 \\ 3 & 2 & 1 & 1 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 1 \\ 1 \\ 0 \\ -1 \end{bmatrix}$$

Tomamos $x^{(0)} = (0,0,0,0)^t$ (como se pode observar, escolhemos sempre, como ponto inicial, o centro da região em que "F" está delimitada).

Apresentaremos agora, apenas os resultados obtidos a cada iteração:

ITERAÇÃO 1: $g^{(0)} = (1,1,0,-1)^t$.

Logo, todas as variáveis são livres e G é não positivo - definido; para $G = LDL^t$, obtemos $d_1 = 1$, $\ell_{12} = 1$, $\ell_{13} = 1$, $\ell_{14} = 3$ e $d_2 = -2 < 0$. Assim, trabalhamos somente com $x_1^{(0)}$ e $x_2^{(0)}$ e teremos

$$(L^{(0)})^t y = e_2 \quad \Rightarrow \quad \begin{matrix} y_1 = 1 \\ y_2 = -1 \end{matrix}$$

e $p^{(0)}=(1,-1,0,0)$ é tal que $\langle g^{(0)},p^{(0)}\rangle < 0$ e $p^{(0)^t} G p^{(0)} < 0$.

Assim

$$x^{(1)} = x^{(0)} + \alpha^{(0)} p^{(0)} , \quad \text{onde } \alpha^{(0)} = 1$$

$$\therefore\ x^{(1)} = (1,-1,0,0)^t$$

ITERAÇÃO 2: $g^{(1)} = (1,3,1,0)^t$ e daí, $x_2^{(1)}$ é "fixa" .

$\bar{G}^{(1)}$ é não positiva definida e trabalhamos com as variáveis livres $x_1^{(1)}$ e $x_3^{(1)}$, obtendo $p^{(1)} = (-1,0,1,0)$ satisfazendo as condições exigidas.

Assim ,

$$x^{(2)} = x^{(1)} + \alpha^{(1)} p^{(1)} , \quad \text{onde } \alpha^{(1)} = 1$$

$$\therefore\ x^{(2)} = (0,-1,1,0)^t .$$

ITERAÇÃO 3: $g^{(2)} = (1,2,-2,-2)$ e então $x_1^{(2)}$ e $x_4^{(2)}$ são variáveis livres; mas $\bar{G}^{(3)}$ é positiva definida.

Logo, procuramos uma direção de Newton, $z^{(2)}$, onde $\bar{G}^{(2)} z^{(2)} = -\bar{g}^{(2)}$; obtemos $z_1 = \frac{11}{12}$ e $z_2 = \frac{-3}{4}$, e então

$$x^{(3)} = x^{(2)} - z^{(2)} = (\tfrac{21}{24}, -1, 1, -5/8) .$$

ITERAÇÃO 4: $g^{(3)} = (0, 39/24, -21/12, 0)$. Logo, $x_1^{(3)}$ e $x_4^{(3)}$ são variáveis livres. Mas como $\bar{G}^{(3)} > 0$ e $g_1^{(3)} = g_4^{(3)} = 0$, temos que $x^{(3)}$ é um ponto ótimo para este problema.

NOTA: Para estes exemplos não utilizamos as idéias de Golub-Murray-Saunders, visto que as matrizes eram de dimensão pequena; mas existem subrotinas que implementam tais idéias e -

que devem ser utilizadas para problemas de ordem maior.

```
      SUBROUTINE MOCHO 1 (L,D,Z,ALFA,N,NL)
C     SEJA  A = LDL^T   e   B = A + ALFA*Z*Z^T
C     ESTA SUBROTINA CALCULA A FATORIZAÇÃO, CHOLESKI DE B
C     CONHECENDO A FATORIZAÇÃO DE A. É O ALGORÍTMO C1
C     DE GILL-GOLUB-MURRAY-SAUNDER.
C     NL É O NÚMERO DE LINHAS QUE ESTÁ DIMENSIONADA
C     A MATRIZ L.
      REAL L(NL,N), D(N), Z(N)
      DO 2  J = 1,N
      P = Z(J)
      DANT = D(J)
      D(J) = D(J) + ALFA*P*P
      IF (J.EQ.N) RETURN
      BETA = P*ALFA/D(J)
      ALFA = DANT*ALFA/D(J)
      DO 3  K = J+1, N
      Z(K) = Z(K) - P*L(K,J)
3     L(K,J) = L(K,J) + BETA*Z(K)
2     CONTINUE
      END

      SUBROUTINE MOCHO 2 (L,D,B,ALFA,N,NL)
C     PARA O CASO EM QUE O NÚMERO DE VARIÁVEIS LIVRES
C     NA ITERAÇÃO K É MAIOR DO QUE O NÚMERO DE VARIÁVEIS
C     LIVRES NA ITERAÇÃO K-1
```

```
      REAL L(NL,N), D(N), B(N)
      N1 = N+1
      L(N1,1) = B(1)
      IF(N.EQ.1) GO TO 1
      DO 2  I = 2,N
      AC = 0
      DO 3  J = 1,I-1
3     AC = AC+L(I,J)*L(N1,J)*D(J)
2     L(N1,I) = B(I)-AC
      D(N1) = 0
1     DO 4  I = 1,N
      AC = L(N1,I)
      L(N1,I) = AC/D(I)
4     D(N1) = D(N1)+AC*L(N1,I)*D(J)
      D(N1) = ALFA-D(N1)
      RETURN
      END

      SUBROUTINE MOCHO 3 (L,D,N,IND,NL,AUX)
C     ESTA SUBROTINA "AJUSTA" AS LINHAS E AS
C     COLUNAS DA MATRIZ A SER FATORIZADA.
      REAL L(NL,N), D(N), AUX(N)
      N1 = N-1
      IF (IND.EQ.N) RETURN
C     IND É O INDICADOR DA VARIÁVEL QUE ESTÁ
C     SENDO FIXADA.
      IF (IND.EQ.1) GO TO 1
```

```
      DO 2  I = 1,IND-1
2     AUX(I) = 0
1     DO 3  I = IND,N1
3     AUX(I) = L(I+1,IND)
      DO 4  I = IND,N1
      DO 4  J = 1,N
4     L(I,J) = L(I+1,J)
      DO 5  J = IND,N1
      DO 5  I = 1,N
5     L(I,J) = L(I,J+1)
      ALFA = D(IND)
      DO 6  I = IND,N1
6     D(I) = D(I+1)
      CALL MOCHO 1 (L,D,AUX,ALFA,N1,NL)
      RETURN
      END
```

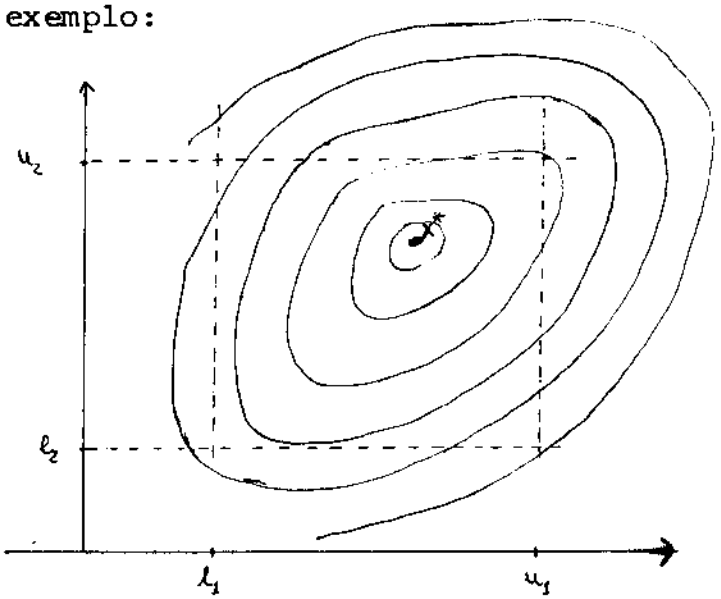
Devemos aqui fazer outras observações acerca do comportamento do algorítmo descrito neste parágrafo:

(a) Podemos notar, através dos exemplos anteriores, que se $d_j < 0$, então as j-ésimas primeiras variáveis livres encontradas determinam uma direção de curvatura negativa. É fácil verificar este resultado: seja $G = LDL^t$, onde $d_j < 0$. Então temos que p é tal que $L^t p = e_j$. Daí $p^t G\, p = p^t LDL^t p = e_j^t D\, e_j = d_j < 0$. Logo p é uma direção de curvatura negativa.

(b) Quanto à convergência deste algorítmo, vimos que, a cada itera

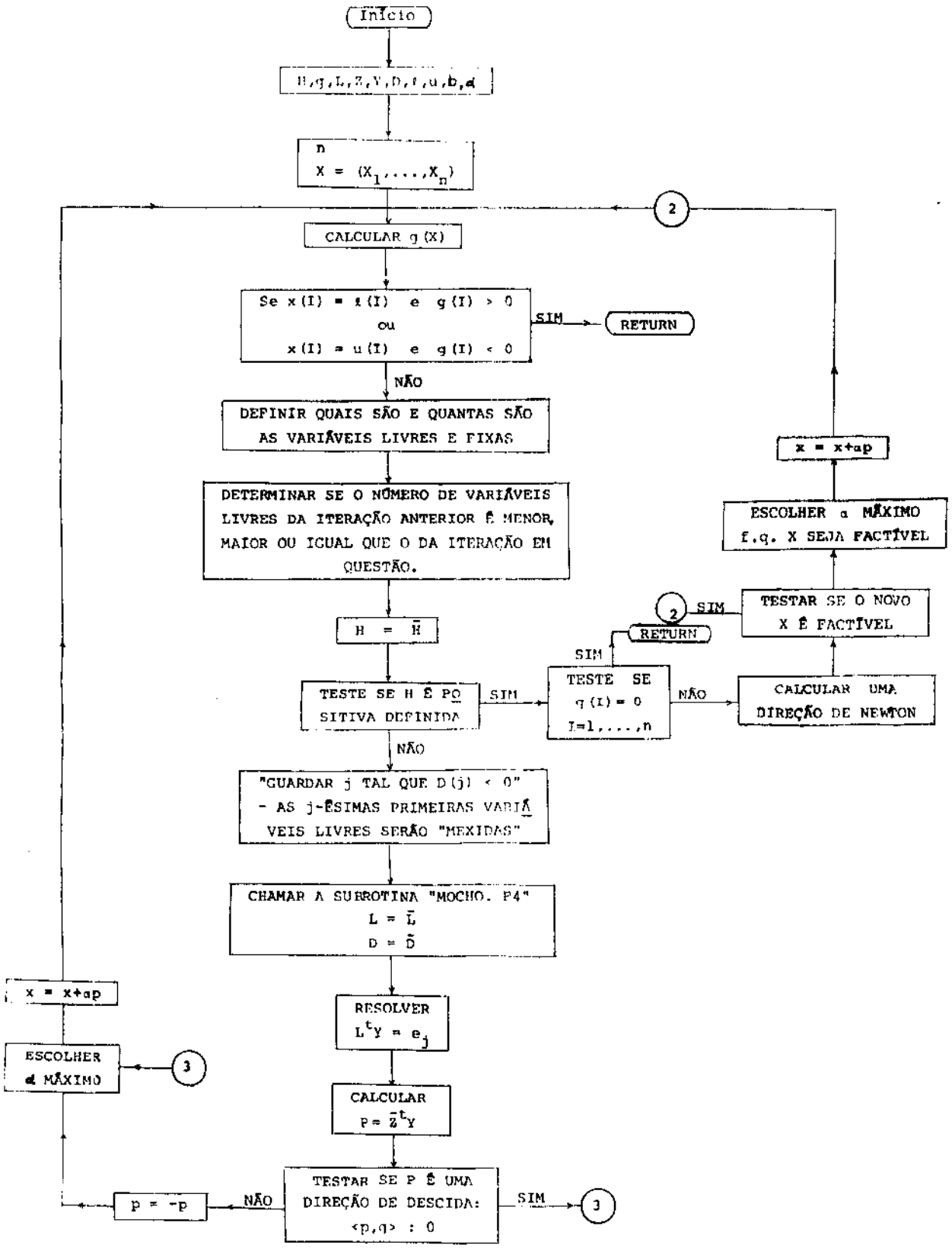
ção, o valor da função no ponto encontrado tende a "melhorar" e isto sabemos que se deve, principalmente, ao fato de que - também estamos tomando direções de descida, ou seja, escolhemos p tal que $\langle g^t, p \rangle < 0$. Dessa forma, podemos então esperar que, para n de tamanho razoável, o algorítmo convirja.

(c) Podemos notar nos exemplos acima que, em diversos casos, à medida em que íamos, a cada iteração, nos aproximando da solução do problema, o número de variáveis livres tendia a diminuir progressivamente, o que nos pode fazer pensar, erroneamente, que este fato está equivalentemente relacionado com "aproximação da solução". Podemos, numa iteração, ter o número de variáveis livres aumentado e obter um melhor valor para a função. Por exemplo:



Para termos uma idéia sequencial dos passos deste algorítmo construimos o

2.2.6) DIAGRAMA EM BLOCOS PARA PROGRAMAÇÃO QUADRÁTICA:


Início
n, q, L, Z, Y, D, ℓ, u, b, α
n
X = (X1, ..., Xn)
2
CALCULAR g(X)
Se x(I) = ℓ(I) e g(I) > 0
ou
x(I) = u(I) e g(I) < 0
SIM
RETURN
NÃO
DEFINIR QUAIS SÃO E QUANTAS SÃO AS VARIÁVEIS LIVRES E FIXAS
DETERMINAR SE O NÚMERO DE VARIÁVEIS LIVRES DA ITERAÇÃO ANTERIOR É MENOR, MAIOR OU IGUAL QUE O DA ITERAÇÃO EM QUESTÃO.
H = H̄
TESTE SE H É POSITIVA DEFINIDA
SIM
NÃO
"GUARDAR j TAL QUE D(j) < 0" - AS j-ÉSIMAS PRIMEIRAS VARIÁVEIS LIVRES SERÃO "MEXIDAS"
CHAMAR A SUBROTINA "MOCHO. P4"
L = L̄
D = D̄
RESOLVER L^t y = e_j
CALCULAR p = z̄^t y
TESTAR SE P É UMA DIREÇÃO DE DESCIDA: <p,g> : 0
NÃO
SIM
3
p = -p
ESCOLHER α MÁXIMO
3
x = x+αp
TESTE SE q(I) = 0 I=1,...,n
SIM
2
RETURN
NÃO
CALCULAR UMA DIREÇÃO DE NEWTON
TESTAR SE O NOVO X É FACTÍVEL
SIM
2
ESCOLHER α MÁXIMO f.q. X SEJA FACTÍVEL
x = x+αp

CAPÍTULO III

ESTIMATIVAS A PRIORI DA PERFORMANCE DE ALGORÍTMOS

A eficiência de um algorítmo se refere, basicamente, ao *número de iterações* que o mesmo gasta até chegar à solução ótima de um problema dado e ao *tempo* necessário para efetuar todos os cálculos a cada iteração (trabalho por iteração). A eficiência está, dessa maneira, diretamente relacionada ao "tempo computacional" necessário para que o algorítmo chegue à solução do problema.

Dessa forma, podemos escrever que o Trabalho Computacional Total (TC) de um algorítmo é, aproximadamente, o produto entre o Número de Iterações (NI) necessários para se chegar a uma solução do problema e o gasto computacional a cada iteração (TI), ou seja,

$$TC \overset{\sim}{} (NI) \times (TI)$$

Se estivermos comparando dois algorítmos A e B, então diremos que A é *mais eficiente* do que B se e somente se

$$(TC)_A < (TC)_B$$

isto é, A é "melhor" do que B se

$$\left[(NI) \times (TI)\right]_A < \left[(NI) \times (TI)\right]_B$$

Assim sendo, pode-se construir uma "Função de Decisão" para os algorítmos de Newton e Quase-Newton, isto é, uma inequação que possa nos dizer quando um será mais eficiente do que o outro; ou seja, queremos obter uma função tal que

$$\left[(NI) \times (TI)\right]_{NEWTON} < \left[(NI) \times (TI)\right]_{Q.N}$$

Para isso, temos que calcular (NI) e (TI) para cada um - dos casos. Comecemos por (NI):

3.1) Desenvolveremos agora, uma pequena teoria para o número de iterações para determinados algorítmos - no caso, Newton e Quase-Newton.

Consideremos a seguinte situação (o seguinte desenvolvimento é similar ao que conduz à noção de eficiência de Ostrovski [8]):

Dado um método de minimização, supomos que sua ordem de convergência é α; isto quer dizer que existe um número $R > 0$ tal que se x^* é solução, então

$$\|x_{k+1}-x^*\| \leq R\|x_k-x^*\|^{\alpha} \qquad (I)$$

Seja ε a distância que tomamos do ponto inicial, x_0, à solução x^* (ε um número pequeno). Se $\alpha > 1$ e se $\|x_0-x^*\| \leq \frac{\varepsilon}{R} < \varepsilon$, então, por (I) temos que

$$\|x_1-x^*\| \leq \varepsilon^{\alpha},\ \|x_2-x^*\| \leq \varepsilon^{\alpha^2},\ \ldots,\ \|x_p-x^*\| \leq \varepsilon^{\alpha^p}$$

Seja TOL a *distância máxima tolerada* entre a solução - achada, x_p, e a solução real do problema, x^*. Então o nosso primeiro problema é encontrarmos "p" tal que

$$\|x_p-x^*\| \leq TOL$$

ou seja, queremos encontrar $p \mid \varepsilon^{\alpha^p} \leq TOL$, ou ainda

$$\alpha^p \log \varepsilon < \log TOL \Rightarrow \alpha^p \geq \frac{\log TOL}{\log \varepsilon}$$

(ε deve pertencer ao intervalo (0,1)).

Se chamarmos $\frac{\log TOL}{\log \varepsilon} = k'$, então $\alpha^p \geq k'$, ou seja, $p \geq \frac{k}{\log \alpha}$ onde $k = \log k'$, k' dependendo de TOL e de ε.

Portanto, podemos concluir que: "se a ordem de convergência de um certo método é $\alpha > 1$, então a predição para o número de iterações (necessários) para obtermos a solução do problema com uma precisão de TOL a partir de uma precisão de ε é $\frac{k}{\log \alpha}$ ", visto que também estamos interessados no *menor* p possível.

A tabela abaixo mostra alguns resultados dessa teoria, com os seguintes dados: Se M_1 e M_2 forem dois métodos com ordem de convergência $\alpha_1 > 1$ e $\alpha_2 > 1$, respectivamente, então o quociente entre o número de iterações que M_1 precisa para convergir e o número de iterações que M_2 precisa é $\log \alpha_2 / \log \alpha_1$. No nosso caso particular, M_1 = Newton com $\alpha_1 = 2$ e M_2 = Quase-Newton com α_2 = raiz positiva de polinômio $t^{n+1}-t^n-1 = 0$ (Schuller), onde n é o número de variáveis da função em questão.

FUNÇÃO-TESTE	N	α_2	$\log \alpha_2 / \log \alpha_1$	Exp. Real
Rosenbrock	2	1.47	0.55	0.59
Vale-Helical de Fletcher-Powell	3	1.38	0.47	0.59
Quártica de Powell	4	1.33	0.41	0.52
Banana de Wood	4	1.33	0.41	0.44
$\sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^4$	4	1.33	0.41	0.37
	10	1.18	0.24	0.26

3.2) Trataremos, agora, de calcular o Trabalho por Iteração (TI) de cada um dos métodos (Newton e Quase-Newton).

Para o método de Newton, vimos que, a cada iteração, temos que calcular o gradiente e o hessiano da função tratada (portanto, estes cálculos dependem da função), resolver o sistema - $[\nabla^2 f]z = -\nabla f$ e mais o trabalho da busca unidimensional.

Chamamos de T_2 o "gasto computacional" para calcularmos o hessiano tendo calculado o gradiente e sabemos que o gasto de Newton para resolver $[\nabla^2 f]z = -\nabla f$ é da ordem de $\frac{n^3}{6} + 1.5n^2 + 5.5n$ produtos e que, além disso, a busca unidimensional consiste em aproximadamente 1.2 avaliações da "f" e "∇f" .

Se chamarmos de T_1 o "gasto computacional" para f e ∇f e de TN o trabalho de Newton *por iteração*, então podemos escrever

$$TN = T_2 + 1.2T_1 + \frac{n^3}{6} + 1.5n^2 + 5.5n$$

Para o método Quase-Newton, sabemos que é desnecessário o cálculo do hessiano, mas precisamos fazer o "updating" da matriz H_k (ou B_k) e calcularmos $H_{k+1}\nabla f$ (ou $B_{k+1}\nabla f$), o que nos dá um custo da ordem de $2.5n^2 + 11.5n$ produtos. Além disso, temos também que calcular f e ainda fazermos a busca unidimensional; ou seja, se chamarmos TQ o trabalho por iteração de Quase-Newton, então escrevemos:

$$TQ = 1.3T_1 + 2.5n^2 + 11.5n$$

OBSERVAÇÃO: Como podemos observar, o método de Newton possui uma pequena vantagem na busca unidimensional com relação ao método Quase-Newton, pois enquanto o primeiro necessita de 1.2 avaliações de f e ∇f, o segundo necessita de 1.3 para as mesmas avaliações.

Portanto, para que "Newton" seja melhor do que "Quase-Newton" devemos ter

$$\frac{1}{\log \alpha_1} \times TN \quad < \quad \frac{1}{\log \alpha_2} \times TQ$$

ou seja,

$$\frac{1}{\log\alpha_1}(T_2+1.2T_1+\frac{n^3}{6}+1.5n^2+5.5n) < \frac{1}{\log\alpha_2}(1.3T_1+2.5n^2+11.5n) \qquad (II)$$

Logo, podemos observar que a nossa "função de decisão" - *depende do número de variáveis* (n) da função que está sendo tratada, quando a solução inicial, x_0, está numa vizinhança próxima de x^* .

Para visualizarmos melhor esta idéia, seja o

EXEMPLO: Vamos resolver a desigualdade (II) para $n = 3$.

Sabemos ser $\alpha_1 = 2$ e $\alpha_2 = 1.38$ e calculamos

$$TN = T_2 + 1.2T_1 + 34.5 \quad \text{e} \quad TQ = 1.3T_1 + 57 .$$

Queremos então, saber quando Newton será melhor do que "Quase-Newton", isto é,

$$(II) \Rightarrow 1.45(T_2 + 1.2T_1 + 34.5) < 3.13(1.3T_1 + 57) .$$

Daqui podemos concluir que, para $n = 3$, Newton será mais eficiente que "Quase-Newton" quando

$$T_2 < 1.56T_1 + 88.5 \qquad (III)$$

Para

$$f(x) = \frac{1}{2} x^t \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 5 \end{bmatrix} x + x_1^4 + x_2^3 + x_1^2 x_2 + x_3^2$$

temos que

$$\nabla f(x) = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 5 \end{bmatrix} x + \begin{bmatrix} 4x_1^3 + 2x_1x_2 \\ 3x_2^2 + x_1^2 \\ 2x_3 \end{bmatrix}$$

e

$$\nabla^2 f(x) = \begin{bmatrix} 3 & 0 & 1 \\ 0 & 4 & 1 \\ 1 & 1 & 5 \end{bmatrix} + \begin{bmatrix} 12x_1^2 + 2x_2 & 2x_1 & 0 \\ 2x_1 & 6x_2 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Logo, $T_1 = 30$ produtos e $T_2 = 6$ produtos, de onde podemos concluir que, para esta função, "Newton" é mais eficiente do que "Quase-Newton".

Observamos, na equação (III), que a "Função de Decisão" procurada, para cada n, possui a forma $T_2 < M_n T_1 + B_n$, onde M_n e B_n são os coeficientes a serem calculados.

Para isso, chamamos $ALN = \frac{1}{\log \alpha_1}$ e $UMS = \frac{1}{\log \alpha_2}$, $CN = n^3/6 + 1.5n^2 + 5.5n$ e $C.Q.N = 2.5n^2 + 11.5n$ e, então, da equação (III) tiramos que

$$ALN(T_2 + 1.2T_1 + CN) < UMS(1.3T_1 + C.Q.N)$$

de onde podemos concluir que

$$T_2 < \frac{(1.3*UMS - 1.2*ALN)}{ALN} T_1 + \frac{UMS*CQN - ALN*CN}{ALN}$$

Portanto, M_n e B_n são tais que

$$M_n = (1.3*UMS - 1.2*ALN) / ALN$$

e

$$B_n = (UMS*CQN - ALN*CN) / ALN$$

Logo, se quisermos calcular M_n e B_n para qualquer n, te-

remos:

```
C       PROGRAMA PARA O CÃLCULO DOS COEFICIENTES DA FUNÇÃO DE
C       DECISÃO "NEWTON - QUASE-NEWTON".
        ALN = 1./ALOG(2.)
        DO 1  N = 2,1ØØ
C       CÃLCULO DA RAIZ POSITIVA DE "F" PARA OBTENÇÃO DO GRAU
C       DE CONVERGÊNCIA DE QUASE-NEWTON
        T = 1.
3       F = T**(N+1)-T**N-1
        IF (ABS(F).LE.1.E-6) GO TO 2
        FL = (N+1)*T**N - N*T**(N-1)
        TN = T-F/FL
        T = TN
        GO TO 3
2       ALT = ALOG(T)
        UMS = 1./ALT
        QUOC = ALT*ALN
C       CÃLCULO DOS COEFICIENTES
        CQN = 2.5*N*N + 11.5*N
        CN = (N**3)/6 + 1.5*N*N + 5.5*N
        MN = (1.3*UMS - 1.2*ALN)/ALN
        BN = (UMS*CQN - ALN*CN)/ALN
        WRITE (7,1Ø)N,T,ALT,UMS,QUOC,BN,MN
1Ø      FORMAT (1X,I4,6E12.6.1/)
        STOP
        END
```

O resultado deste programa é a *TABELA NEWQUA* (ver Apêndice) e, se a observarmos, podemos notar que a "Função de Decisão" - para Newton - Quase-Newton se comporta da seguinte maneira:

- para $n = 2,\ldots,95$, podemos verificar que M_n e B_n são sempre números positivos, com valores muito grandes e crescendo à medida em que n cresce. Logo, podemos desde já afirmar que, *para que "Quase-Newton" seja mais eficiente do que Newton*, o gasto computacional para o cálculo do hessiano de dada função deverá ser extremamente maior do que o gasto computacional para se calcular essa mesma função e o seu gradiente, e isto dependerá da função com que estivermos trabalhando.

3.3) Para conferirmos estes resultados, escolhemos as funções-teste (trigonométricas de Fletcher-Powell)

$$f(x) = \sum_i (E_i - \sum_j A_{ij} \operatorname{sen} x_j + B_{ij} \cos x_j)^2$$

onde A_{ij} e B_{ij} são números aleatórios entre -100 e 100. Como o problema é de minimização, então, para uma solução x_j^* ,

$$E_i = \sum_j A_{ij} \operatorname{sen} x_j^* + B_{ij} \cos x_j^*$$

onde $x_j^* \in (-\pi, \pi)$.

Se fizermos

$$f_i(x) = E_i - \sum_j A_{ij} \operatorname{sen} x_j + B_{ij} \cos x_j$$

então

$$f(x) = \sum_i f_i(x)^2 = \|FF(x)\|^2$$

e teremos

$$\nabla f(x) = 2F'(x)FF(x)$$

onde

$$FF(x) = \begin{bmatrix} E_1 - \sum_j A_{1j} \operatorname{sen} x_j + B_{1j} \cos x_j \\ \vdots \\ E_n - \sum_j A_{nj} \operatorname{sen} x_j + B_{nj} \cos x_j \end{bmatrix}$$

e

$$F'(x) = \begin{bmatrix} -A_{11}\cos x_1 + B_{11}\operatorname{sen} x_1 & \cdots & -A_{1n}\cos x_n + B_{1n}\operatorname{sen} x_n \\ \vdots & & \vdots \\ -A_{1n}\cos x_n + B_{1n}\operatorname{sen} x_n & \cdots & -A_{nn}\cos x_n + B_{nn}\operatorname{sen} x_n \end{bmatrix} .$$

Portanto,

$$\left(F'(x)\right)_{ij} = -A_{ij} \cos x_j + B_{ij} \operatorname{sen} x_j , \quad \begin{matrix} i = 1,\ldots,n \\ j = 1,\ldots,n \end{matrix}$$

Ainda ,

$$\nabla^2 f(x) = 2(F'(x)^t F'(x) + \sum_j f_i(x) \nabla^2 f_i(x))$$

onde

$$\nabla f_k(x) = \begin{bmatrix} -A_{k1} \cos x_1 + B_{k1} \operatorname{sen} x_1 \\ \vdots \\ -A_{kn} \cos x_n + B_{kn} \operatorname{sen} x_n \end{bmatrix}$$

donde vem que

$$\nabla^2 f_k(x) = \begin{bmatrix} A_{k1} \operatorname{sen} x_1 + B_{k1} \cos x_1 & 0 & \cdots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & A_{kn} \operatorname{sen} x_n + B_{kn} \cos x_n \end{bmatrix}$$

A partir daqui, então, podemos construir as subrotinas

para os cálculos dos hessianos e gradientes destas funções. Chamamos de CALHES a subrotina para o cálculo do hessiano e de CALCFG a subrotina para o cálculo da função e seu gradiente:

```
      SUBROUTINE CALCFG (N,X,F,G)
      DIMENSION X(1), G(1)
      COMMON A (5Ø,5Ø), B(5Ø,5Ø), E(5Ø)
      COMMON /C1/ S(5Ø), C(5Ø), FL(5Ø,5Ø), FF(5Ø)
      DO 1  I = 1,N
      S(I) = SIN (X(I))
1     C(I) = COS (X(I))
      DO 2  J = 1,N
      DO 2  I = 1,N
2     FL(I,J) = -A(I,J)*C(J)+B(I,J)*S(J)
      FF(I) = Ø
      DO 3  I = 1,N
      DO 4  J = 1,N
4     FF(I) = FF(I)+A(I,J)*S(J)+B(I,J)*C(J)
3     FF(I) = E(I)-FF(I)
      DO 5  I = 1,N
      G(I) = Ø
      DO 6  J = 1,N
6     G(I) = G(I)+FL(J,I)*FF(J)
5     G(I) = 2.*G(I)
      F = Ø
      DO 7  I = 1,N
7     F = F+FF(I)**2
      RETURN
      END
```

n senos e n cossenos

2 n^2 produtos

2 n^2 produtos

n^2+n produtos

n produtos

```
      SUBROUTINE CALHES (N,X,H)
      DIMENSION A(5Ø,5Ø), B(5Ø,5Ø), E(5Ø)
      COMMON A(5Ø,5Ø), B(5Ø,5Ø), E(5Ø)
      COMMON /C1/ S(5Ø), C(5Ø), FL(5Ø,5Ø), FF(5Ø)
      DO 1  I = 1,N
      DO 1  J = 1,N
      HH(I,J) = Ø
      DO 1  K = 1,N
1     HH(I,J) = HH(I,J)+FL(K,I)*FL(K,J)
      DO 2  I = 1,N
      DO 2  J = 1,N
2     HH(J,J) = HH(J,J)+FF(I)*(A(I,J)*S(J)+B(I,J)*C(J))
      K = 1
      DO 3  J = 1,N
      DO 3  I = J,N
      H(K) = HH(I,J)*2
3     K = K+1
      RETURN
      END
```

n^3 produtos

n^3 produtos

$\frac{n^2}{2} + \frac{n}{2}$ produtos

Considerando reduzidos os gastos computacionais com somas e produtos para a ordem de 1 nano-segundo e com seno e cosseno para a ordem de 15 nano-segundo , chegamos que, para estas funções, o "custo" da subrotina CALCFG é de $5n^2 + 32n$ nano-segundos e da CALHES é de $n^3 + 3.5n^2 + \frac{n}{2}$ nano-segundos.

Dessa forma podemos escrever

$$T_2 = n^3 + 3.5n^2 + \frac{n}{2}$$

e

$$T_1 = 5n^2 + 32n$$

e notamos que T_2 "cresce" muito mais rapidamente do que T_1 à medida em que n cresce. Para visualizarmos isto, podemos utilizar a tabela NEWQUA (ver Apêndice) e efetuarmos alguns cálculos, como seguem:

para $n = 2$, $M_n = 1.15737$ e $B_n = 41.8408$,

$$T_2 = 23 \quad e \quad T_1 = 84 \Rightarrow T_2 < M_n T_1 + B_n .$$

para $n = 30$, $M_n = 9.78968$ e $B_n = 15922.1$,

$$T_2 = 30165 \quad e \quad T_1 = 5460 \Rightarrow T_2 < M_n T_1 + B_n .$$

para $n = 94$, $M_n = 24.2798$ e $B_n = 301947$,

$$T_2 = 861557 \quad e \quad T_1 = 47188 \Rightarrow T_2 < M_n T_1 + B_n .$$

Mas, M_n e B_n também crescem à medida em que n cresce e então, como já foi dito anteriormente, deveremos ter n bastante grande para que $T_2 > M_n T_1 + B_n$.

OBSERVAÇÃO: Quando este problema é tratado com o Método de Newton Discreto, temos que o cálculo para o hessiano de uma função de "N" variáveis é aproximadamente o produto de N pelo cálculo do gradiente da mesma função, i.e., $T_2 = NT_1$.

Em particular, para as funções-teste tratadas acima, podemos utilizar a tabela NEWQUA e verificar que:

para $N = 2$, $T_2 = 2 \times 84 = 168$ e

$$M_n T_1 + B_n \tilde{=} 84 + 42 = 126$$

Logo, $T_2 > M_n T_1 + B_n$. Da mesma forma,

para $N = 30$, $T_2 = 30 \times 546 = 193.800$ e

$$M_n T_1 + B_n \cong 75.982 .$$

Portanto, $T_2 > M_n T_1 + B_n$.

E notamos que, para N grande, a diferença $(T_2 - (M_n T_1 + B_n))$ também cresce. Disto podemos concluir que o método Quase-Newton, para essas mesmas funções-teste, é melhor do que o Método de Newton Discreto.

Através da equação (II), podemos encontrar uma estimativa para quando o método de Newton Discreto será melhor do que o método do Quase-Newton, ou seja, para $T_2 = nT_1$, temos que

$$\frac{1}{\log \alpha_1}[nT_1 + 1.2T_1 + CN] < \frac{1}{\log \alpha_2}[1.3T_1 + C.Q.N]$$

Para o caso particular em que $n = 2$, temos $C.Q.N = 33$, $C.N \cong 18.33$ e $\frac{1}{\log \alpha_2} \cong 2.60$ e então

$$1.44[3.2T_1 + 18.33] < 2.6[1.3T_1 + 33]$$

ou seja,

$$T_1 < \frac{2.6 \times 33 - 1.44 * 18.33}{1.44 \times 3.2 - 2.6 * 1.3} \cong 48.38$$

Portanto, para $T_1 \leq 48$ podemos esperar que "Newton Discreto" seja melhor do que "Quase-Newton".

No próximo capítulo, mostraremos os testes computacionais feitos com os algorítmos dos métodos aqui já estudados e, com isso, analisaremos com mais detalhes o comportamento do Método de Newton com hessiano discretizado.

CAPÍTULO IV

TESTES

Como foi dito na Introdução deste trabalho, um dos nossos objetivos, aqui, é o de encontrarmos um algorítmo para o método de Newton que seja eficiente e compará-lo com os demais algorítmos dos métodos já descritos, Quase-Newton e Gradientes Conjugados.

Dessa forma, nos propomos, neste capítulo, a fazer comparações entre os algorítmos tratados no Capítulo I para verificarmos o real comportamento de cada um deles. Juntamente com esses, analisaremos o comportamento do algorítmo do método de Newton com fatorização de Choleski modificada (que chamaremos de "Newton Modificado") para, daí, podermos tirar resultados sobre as possíveis vantagens, ou desvantagens, das modificações propostas (ver Cap. 2, § 2.1).

Para isso, fizemos vários testes que veremos logo a seguir e que nos serão úteis, também, para verificarmos a validade da tabela construída no capítulo 3 acerca da "eficiência" dos algorítmos Newton e Quase-Newton.

4.1) Assim, utilizando os conceitos de eficiência descritos no capítulo anterior, vamos testar e comparar os seguintes programas:

4.1.1) Para o algorítmo de Newton, utilizaremos três programas diferentes que são: SUBROUTINE MFØ1A, inserida no arquivo MFØ1A da biblioteca do Laboratório de Matemática Aplicada (LABMA), IMECC-

UNICAMP, SUBROUTINE NEWMIN e a SUBROUTINE EØ4EBF, que se refere ao algorítmo de "Newton Modificado" e que pode ser encontrada no Pacote da "NAG" - LABMA .

Os dois primeiros programas diferem, basicamente, em suas "buscas unidimensionais", sendo que o programa da "NEWMIN" *aceita um novo valor para a função* simplesmente *quando este é menor do que o valor antigo* desta mesma função, tendo já, daí, início a uma nova iteração, na busca de um melhor ponto. Já o programa da "MFØ1A" é mais exigente na aceitação do novo valor da função, pois exige que

$$f(x_n) < f(x) + 0.0001 \langle d, \nabla f(x) \rangle$$

e

$$\langle \nabla f(x_n), d \rangle \geq 0.9 \langle \nabla f(x), d \rangle \ .$$

Ainda com referência a Newton, vamos também testar, para cada uma das subrotinas acima, o algorítmo contendo o hessiano discretizado, a fim de podermos avaliar o seu comportamento e em quais ocasiões será válido empregá-lo. Para testarmos "Newton Discreto" usamos um programa que calcula cada elemento da matriz hessiana discretizada por "diferenças finitas" (Cap. I - § 1.4) .

4.1.2) Para o algorítmo Quase-Newton, utilizamos os programas das SUBROUTINE MFØ1A e SUBROUTINE VA13A, tendo o primeiro a versão de Shanno-Phua e o segundo de Powell é o algorítmo (B.F.G.S.).

A SUBROUTINE VA13A foi retirada do PACOTE DE HARWELL.

O motivo pelo qual também estamos testando vários programas para o Método Quase-Newton se insere no objetivo deste traba-

lho, visto que estamos interessados em encontrar programas eficientes, compará-los entre si, para daí tirarmos conclusões sobre a eficiência dos algorítmos aqui tratados, dadas as "circunstâncias" ressaltadas.

Obviamente, existe um ponto fundamental no qual estes programas diferem e este ponto está traduzido no cálculo da "matriz aproximada" do hessiano da função em questão, empregada na busca unidimensional do algorítmo (Cap. I, § 2.1). Mais explicitamente, o programa "MFØ1A" armazena essa matriz através do cálculo direto da fatorização de Choleski da forma "H_k", mas para o programa da "VA13A", os valores armazenados para essa mesma matriz - são relativos à fatorização de Choleski da forma "B_k^{-1}" .

Segundo os autores da "VA13A", esta maneira de armazenar valores causa menos riscos de arredondamentos, podendo-se, assim, haver também uma melhor garantia da "matriz aproximada" ser positiva definida.

4.1.3) Para o algorítmo dos Gradientes Conjugados, testamos apenas o programa da SUBROUTINE MFØ1A . (*)

4.2) Quanto às funções-teste utilizadas, tentamos escolhê-las de forma que as mesmas nos trouxesse as informações necessárias;ou seja, algumas delas são testadas para valores diferentes de "N" - (número de variáveis) de tal forma que possamos observar possíveis influências da variação de "N" na eficiência de cada algorítmo.

(*) *O ARQUIVO MFØ1A contém programas dos três métodos aqui estudados.*

A tabela abaixo descreve tais funções com os seus respectivos pontos-solução:

FUNÇÃO	EXPRESSÃO ANALÍTICA	NÚMERO VARIÁVEIS	SOLUÇÃO
ROSENBROCK	$f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2$	2	$x^* = (1,1)$ $f(x^*) = 0$
BANANA DE WOOD	$f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1(x_2-1)^2 + x_4-1)^2) + 19.8(x_2-1)(x_4-1)$	4	$x^* = (1,1,1,1)$ $f(x^*) = 0$
QUÁRTICA POWELL	$f(x) = (x_1+10x_2)^2 + 5(x_2-x_4)^2 + (x_2-2x_3)^4 + 10(x_1-x_4)^4$	4	$x^* = (0,0,0,0)$ $f(x^*) = 0$
BOX	$f(x) = \sum_{i=1}^{10} (\exp(-x_1 t_i) - \exp(-x_2 t_i)] - [\exp(-t_i) - \exp(-10t_i)])^2$ onde $t = (0.1, 0.2, \ldots, 1)$	2	$x^* = (1,1)$ $f(x^*) = 0$
CRAGG--LEVY	$f(x) = (e_1^x - x_2)^4 + 100(x_2-x_3)^6 + \tan^4(x_3-x_4) + x_1^8 + (x_4-1)^2$	4	$x^* = (0,1,1,(1\pm n\pi)$ $f(x^*) = 0$
VALE HELICAL	$f(x) = 100([x_3 - 10\theta(x_1,x_2)]^2 + [(x_1^2+x_2^2)^{1/2} - 1]^2) + x_3^2$ onde $\theta(x_1,x_2) = \begin{cases} \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1}, & x_1 > 0 \\ \frac{1}{2} + \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1}, & x_1 < 0 \end{cases}$	3	$x^* = (1,0,0)$ $f(x^*) = 0$
QUADR. TRIGONOMÉTRICA	$f(x) = \frac{1}{2}(x-2)^t G(x-2) + \sum_{r=1}^{n} a_i \sin^2(x_i - z_i)$ onde $G = LL^t$, L_{ij}, a_i, z_i são aleatórios em $[-1,1]$	v	$x^* = z$ $f(x^*) = 0$
ELBA	$f(x) = \frac{1}{2}x^t G x$, onde $G_{ij} = \frac{1}{i+j-1}$ se $i \neq j$ $G_{ii} = n$, $i = 1,\ldots,n$	v	$x^* = \vec{0}$ $f(x^*) = 0$

4.3) Conforme o Capítulo 3, a "teoria da eficiência" se baseia no fato do ponto inicial, x_0, dado no problema, estar bastante próximo do ponto solução, donde tiramos uma previsão para o número de iterações necessárias para certos algorítmos, com ordem de convergência conhecida, convergirem.

Dessa forma, testamos os vários algorítmos com vários pontos iniciais, cujas distâncias com relação ao ponto-solução da fun

ção em questão são distintas. Com isso, tentamos conseguir um intervalo de valores confiáveis para esta distância de tal maneira que este intervalo nos diga quando podemos esperar, realmente, que Newton possa ser mais eficiente que Quase-Newton e/ou vice-versa.

Resumindo, sabemos que tanto o número de variávies das funções a serem testadas quanto a distância, por nós estabelecida, entre o ponto inicial e a solução de cada função devem exercer influência sobre o trabalho de cada algorítmo e, consequentemente, sobre sua eficiência.

Assim, para cada função-teste considerada, adotamos o seguinte critério para a escolha do ponto inicial:

Sejam SOL(I) , I = 1,...,n, a solução da função e DIST $\in$ {Ø.ØØ1, Ø.01, 0.1, 1., 10., 100.} . Então,

$$x(I) = SOL(I) + (RAN(1.)-Ø.5)*2*DIST, \quad I = 1,...,n ,$$

onde RAN(.) é a subrotina que gera pontos aleatórios entre Ø e 1.

Portanto, à medida em que atribuímos valores maiores para DIST, obtemos pontos iniciais, x(I), I = 1,...,n, cujas distâncias ao ponto-solução, SOL(I), I = 1,...,n, são cada vez maiores.

Para cada "DIST", procuramos gerar dez pontos iniciais diferentes a fim de termos condições razoáveis para avaliar os vários casos distintos. As tabelas apresentadas abaixo são formas simplificadas que adotamos para darmos idéia dos resultados que obtivemos nos testes realizados.

Quanto aos critérios-de-parada dos algorítmos, sabemos estarem eles baseados na aproximação da norma do gradiente da função a zero. Algumas diferenças em termos de programação devem existir

entre os mesmos, mas isto será desprezado no nosso problema. No entanto, algumas observações com relação à aproximação do algorítmo à solução da função serão feitas no final deste capítulo.

Antes de passarmos a analisar as várias tabelas, devemos acrescentar aqui que apesar de sabermos que o tempo computacional (CPU), gasto por cada algorítmo até a sua convergência, está diretamente relacionado com o número de iterações e o seu trabalho por iteração, não podemos considerá-lo o fator único na determinação da "eficiência" dos algorítmos aqui tratados, devido ao fato do sistema utilizado para nossas experiências apresentar "cargas de trabalho" frequentemente oscilatórias, acarretando, dessa forma , problemas de precisão na comparação dos vários "tempos" conseguidos.

4.4) Para cada experiência realizada, apresentamos três dados, o primeiro referente ao número de iterações, o segundo ao número de avaliações da função e o terceiro ao tempo computacional gastos, respectivamente, por cada algorítmo até a sua convergência.

Através dos dois primeiros dados podemos verificar as estimativas "1.2" e "1.3" que se referem ao número de avaliações de função por iteração gastos pelos algorítmos de Newton e Quase-Newton, respectivamente. Com relação ao primeiro dado, faremos, no final do capítulo, comparações entre a "razão entre os números de iterações necessárias para "Newton" e Quase-Newton" convergirem", descrita no Capítulo III, e os dados reais.

Utilizaremos, para cada tabela abaixo, as seguintes notações para os casos de não-convergência dos algorítmos:

NFLAG = 1: Excedeu o número máximo de avaliações da função.

NFLAG = 2: A busca inidimensional falhou ao melhorar o valor da - função.

NFLAG = 3: O vetor-direção não é uma direção de descida.

TABELA 1

1) Para os algorítmos de Newton, podemos observar que, para DIST menor do que dez, o programa "MFØ1A" converge mais rapidamente do que o programa "NEWMIN", apresentando um número menor de iterações e avaliações da função, o que nos surpreende pelo fato dele ser mais exigente na escolha do valor da função. Mas, para valores maiores de DIST, "NEWMIN" nos parece ser mais eficiente do que "MFØ1A".

2) Já para o algorítmo Quase-Newton, notamos que o programa MFØ1A é melhor do que o programa VA13A para valores pequenos de "DIST"; para DIST igual a cem, este programa excede várias vezes o número máximo de avaliações da função dado no programa principal (MXFUN = 5ØØ*N), apesar de, para os casos de convergência, - apresentar um tempo computacional melhor do que o tempo do programa VA13A .

3) Sobre o algorítmo do método de Gradientes Conjugados, podemos notar a sua superioridade com relação ao algorítmo do método Quase-Newton para quaisquer valores de DIST, ainda que o mesmo necessite de um número maior de avaliações da função por iteração. Com relação ao algorítmo de Newton, observamos alguns ca-

TABELA 1

FUNÇÃO-TESTE: ROSENBROCK

NEWMIN	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
3, 4, 0.015	3, 4, 0.038	3, 4, 0.018	3, 4, 0.015	9, 24, 0.030	7, 10, 0.061	6, 12, 0.030	
3, 11, 0.023	2, 10, 0.015	2, 4, 0.012	2, 4, 0.042	8, 19, 0.057	9, 18, 0.077	6, 12, 0.047	0.01
4, 5 0.031	3, 4, 0.048	3, 4, 0.026	3, 4, 0.043	12, 34, 0.075	8, 11, 0.054	5, 12, 0.037	
4, 5, 0.030	4, 5, 0.035	3, 4, 0.032	4, 5, 0.023	7, 16, 0.029	10, 13, 0.067	9, 14, 0.030	
5, 8, 0.031	4, 7, 0.030	4, 6, 0.025	4, 6, 0.017	6, 15, 0.026	12, 15, 0.075	9, 13, 0.030	0.1
4, 10, 0.038	4, 10, 0.029	4, 6, 0.011	5, 7, 0.030	8, 18, 0.047	10, 11, 0.059	14, 16, 0.041	
14, 19, 0.047	13, 18, 0.060	13, 19, 0.038	13, 19, 0.062	20, 48, 0.077	22, 29, 0.115	20, 26, 0.061	
13, 16, 0.031	13, 17, 0.058	13, 19, 0.047	13, 18, 0.052	6, 16, 0.031	15, 21, 0.086	16, 18, 0.047	1.
10, 13, 0.015	9, 12, 0.052	9, 12, 0.033	9, 12, 0.042	11, 27, 0.046	17, 22, 0.089	13, 19, 0.047	
48, 66, 0.104	NFLAG = 1	48, 69, 0.126	NFLAG = 1	28, 70, 0.128	55, 75, 0.223	51, 65, 0.138	
15, 19, 0.043	16, 21, 0.066	15, 21, 0.049	16, 23, 0.069	18, 46, 0.078	25, 32, 0.110	31, 38, 0.085	10.
11, 14, 0.053	12, 22, 0.053	9, 12, 0.039	12, 17, 0.062	20, 47, 0.061	17, 23, 0.088	37, 51, 0.092	
102, 142, 0.195	596, 611, 1.854	104, 157, 0.297	161, 281, 0.654	22, 56, 0.083	50, 65, 0.199	57, 75, 0.133	
124, 170, 0.240	NFLAG = 1	126, 192, 0.351	NFLAG = 1	37, 93, 0.147	83, 116, 0.320	82, 117, 0.217	100.
166, 228, 0.377	NFLAG = 1	170, 257, 0.408	NFLAG = 1	61, 154, 0.223	98, 132, 0.537	96, 135, 0.264	

sos de sua superioridade já para DIST pequeno; para os valores maiores de DIST, podemos afirmar que Gradientes Conjugados é mais eficiente que os demais métodos aqui referidos.

4) Para esta função-teste, a superioridade do algorítmo do método de Newton com relação ao algorítmo do método Quase-Newton é indiscutível.

5) Quanto ao algorítmo de Newton com hessiano discretizado, torna-se evidente a sua ineficiência quando DIST é grande, e este fato pode ser explicado teoricamente, pois sabemos que a aproximação local de uma função diferenciável à sua forma linear é tão boa quanto menor for a vizinhança tomada com relação ao ponto em questão, ou seja,

$$\text{(III)} \qquad f''(x) \approx \frac{f'(x+h) - f'(x)}{h}$$

quando $h \to 0$, e para evitar o cancelamento $[f(x+h)-f'(x)]$ quando x é grande, h também deve ser grande, e, portanto a aproximação por diferenças será ruim.

Para DIST pequeno, este algorítmo é ainda mais caro do que Newton Analítico e isto se deve ao cálculo do quociente das diferenças (III) necessário para cada elemento da matriz (Cap. I , § 3) .

TABELA 2

1) As observações 1, 4 e 5 da tabela anterior são válidas, também, para esta função.

TABELA 2

FUNÇÃO-TESTE: BANANA DE WOOD

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
3, 4, 0.039	3, 4, 0.060	2, 3, 0.031	2, 3, 0.035	10, 22, 0.081	10, 15, 0.138	18, 20, 0.114	
4, 5, 0.041	3, 4, 0.041	3, 4, 0.034	3, 4, 0.044	10, 22, 0.069	11, 29, 0.126	18, 20, 0.101	0.01
3, 4, 0.023	3, 4, 0.030	3, 4, 0.017	3, 4, 0.031	16, 34, 0.094	9, 14, 0.102	18, 20, 0.086	
6, 10, 0.040	5, 9, 0.056	5, 7, 0.048	5, 7, 0.057	11, 24, 0.075	13, 17, 0.128	20, 22, 0.091	
6, 9, 0.042	5, 8, 0.069	5, 7, 0.031	5, 7, 0.054	16, 34, 0.101	16, 19, 0.179	10, 16, 0.071	0.1
6, 9, 0.048	5, 8, 0.058	5, 7, 0.037	5, 7, 0.052	12, 26, 0.085	13, 17, 0.140	19, 21, 0.115	
10, 15, 0.058	9, 14, 0.111	11, 17, 0.054	11, 16, 0.109	25, 51, 0.175	22, 26, 0.172	25, 26, 0.153	
9, 10, 0.069	8, 9, 0.098	8, 9, 0.074	8, 9, 0.084	24, 49, 0.149	20, 26, 0.178	21, 28, 0.136	1.
8, 9, 0.045	8, 9, 0.076	8, 9, 0.046	8, 9, 0.077	20, 41, 0.123	19, 24, 0.148	28, 29, 0.138	
29, 35, 0.133	25, 31, 0.214	28, 34, 0.145	25, 31, 0.214	71, 148, 0.370	62, 68, 0.468	99, 120, 0.513	
18, 21, 0.076	47, 182, 0.490	18, 19, 0.097	19, 22, 0.178	31, 65, 0.204	44, 52, 0.328	89, 110, 0.445	10.
21, 23, 0.107	NFLAG = 1	20, 22, 0.152	NFLAG = 1	66, 145, 0.368	46, 57, 0.440	101, 117, 0.491	
26, 28, 0.138	NFLAG = 1	26, 28, 0.117	115, 148, 1.034	78, 173, 0.456	88, 99, 0.603	97, 111, 0.478	
37, 43, 0.183	NFLAG = 1	37, 44, 0.233	NFLAG = 1	71, 146, 0.429	75, 84, 0.537	100, 120, 0.518	100.
37, 41, 0.199	NFLAG = 1	38, 43, 0.210	NFLAG = 1	54, 111, 0.292	94, 115, 0.631	108, 130, 0.513	

2) Para o método Quase-Newton, podemos observar uma "melhora" do programa VA13A sobre MFØ1A, não deixando de lado o fato de, ainda, o programa MFØ1A apresentar um tempo computacional (CPU) inferior ao do programa VA13A .

3) Quanto ao algorítmo do método dos Gradientes Conjugados, notamos, aqui, uma pequena diferença no seu comportamento com relação à tabela anterior, ou seja, ao mesmo tempo em que apresenta uma maior eficiência diante do algorítmo do método Quase-Newton, apresenta, também, uma maior inferioridade com relação a "Newton" e isto deve ser observado para quaisquer valores de DIST.

TABELA 3

1) Para esta função notamos que os dois programas de Newton, NEWMIN e MFØ1A, convergem num mesmo número de iterações e avaliações de função, distinguindo-se apenas no tempo de CPU, cujas diferenças são pequenas.

2) Para "Quase-Newton", podemos observar uma maior eficiência do programa MFØ1A sobre o programa VA13A, cujo trabalho computacional é extremamente grande.

3) As observações 3, 4 e 5 da tabela anterior são válidas, também, para esta função.

TABELA 4

1) Nesta tabela, podemos observar que o programa MFØ1A, de Newton, é *pouco* mais eficiente do que o programa NEWMIN, estando esta

TABELA 3

FUNÇÃO-TESTE: QUÁRTICA DE POWELL

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
1, 2, 0.026	1, 2, 0.024	1, 2, 0.016	1, 2, 0.030	3, 7, 0.031	49, 64, 0.387	4, 7, 0.030	
1, 2, 0.015	1, 2, 0.030	1, 2, 0.022	1, 2, 0.023	3, 7, 0.031	27, 33, 0.141	8, 10, 0.045	0.01
1, 2, 0.022	1, 2, 0.027	1, 2, 0.031	1, 2, 0.028	3, 7, 0.029	42, 57, 0.339	7, 9, 0.038	
6, 7, 0.041	6, 7, 0.066	6, 7, 0.029	6, 7, 0.070	25, 52, 0.120	57, 61, 0.409	29, 34, 0.151	
5, 6, 0.041	5, 6, 0.061	5, 6, 0.042	5, 6, 0.045	22, 45, 0.124	68, 82, 0.476	23, 25, 0.123	0.1
6, 7, 0.030	6, 7, 0.046	6, 7, 0.031	6, 7, 0.061	27, 55, 0.138	70, 88, 0.497	36, 39, 0.164	
11, 12, 0.061	11, 12, 0.107	11, 12, 0.060	11, 12, 0.102	25, 52, 0.130	73, 87, 0.497	18, 20, 0.091	
12, 13, 0.061	12, 13, 0.107	13, 14, 0.075	13, 14, 0.130	47, 100, 0.239	78, 88, 0.515	36, 37, 0.153	1.
12, 13, 0.068	12, 13, 0.099	12, 13, 0.060	12, 13, 0.107	23, 49, 0.127	78, 91, 0.563	37, 40, 0.177	
17, 18, 0.081	20, 21, 0.199	18, 19, 0.092	23, 24, 0.274	41, 87, 0.225	72, 84, 0.517	59, 65, 0.297	
14, 15, 0.070	14, 15, 0.154	14, 15, 0.077	14, 15, 0.132	37, 77, 0.207	75, 83, 0.509	52, 55, 0.272	10.
17, 18, 0.092	18, 19, 0.172	17, 18, 0.081	22, 24, 0.220	21, 43, 0.117	75, 93, 0.520	52, 60, 0.248	
22, 23, 0.107	111, 112, 0.780	22, 23, 0.116	38, 51, 0.316	29, 66, 0.153	83, 101, 0.577	87, 101, 0.391	
19, 20, 0.102	97, 102, 1.107	19, 20, 0.096	58, 69, 0.730	29, 63, 0.171	73, 85, 0.527	66, 78, 0.330	100.
23, 24, 0.107	99, 106, 0.725	23, 24, 0.107	44, 49, 0.356	29, 59, 0.154	99, 121, 0.701	81, 90, 0.426	

TABELA 4

FUNCÃO-TESTE: DE BOX

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
2, 3, 0.229	2, 3, 0.246	1, 2, 0.139	2, 3, 0.242	6, 17, 0.667	6, 9, 0.369	2, 4, 0.170	
1, 2, 0.137	2, 3, 0.242	1, 2, 0.136	1, 2, 0.152	6, 18, 0.697	6, 8, 0.337	2, 5, 0.194	0.01
2, 3, 0.225	2, 3, 0.244	1, 2, 0.137	2, 3, 0.261	7, 20, 0.789	7, 12, 0.552	2, 5, 0.209	
2, 3, 0.218	2, 3, 0.233	2, 3, 0.224	2, 3, 0.264	3, 7, 0.310	12, 13, 0.654	11, 13, 0.557	
2, 3, 0.215	2, 3, 0.250	2, 3, 0.212	2, 3, 0.247	4, 10, 0.398	13, 15, 0.715	12, 14, 0.576	0.1
2, 3, 0.228	2, 3, 0.258	2, 3, 0.223	2, 3, 0.243	2, 5, 0.226	8, 9, 0.393	3, 5, 0.211	
4, 5, 0.387	4, 7, 0.538	4, 5, 0.391	4, 7, 0.561	4, 9, 0.387	8, 11, 0.460	14, 15, 0.618	
4, 5, 0.385	NFLAG = 1	4, 5, 0.398	4, 10, 0.659	8, 17, 0.723	8, 11, 0.491	5, 8, 0.333	1.
4, 5, 0.424	5, 6, 0.545	3, 4, 0.301	4, 5, 0.452	4, 9, 0.391	13, 14, 0.709	7, 11, 0.473	
14, 16, 1.332	14, 17, 1.519	14, 16, 1.288	13, 16, 1.439	15, 32, 1.265	41, 54, 2.303	NFLAG = 1	
9, 10, 0.849	9, 10, 0.957	9, 10, 0.807	7, 9, 0.792	9, 21, 0.885	16, 22, 0.920	13, 16, 0.656	10.
27, 61, 3.919	15, 20, 1.767	34, 58, 3.970	14, 17, 1.529	13, 32, 1.315	25, 38, 1.597	31, 32, 1.328	
NFLAG = 2	NFLAG = 2	NFLAG = 3	NFLAG = 3	NFLAG = 3	NFLAG = 1	NFLAG = 1	
NFLAG = 1	NFLAG = 1	NFLAG = 3	NFLAG = 3	NFLAG = 3	NFLAG = 1	NFLAG = 3	100.
NFLAG = 2	NFLAG = 2	NFLAG = 3	NFLAG = 3	NFLAG = 3	NFLAG = 1	NFLAG = 3	

superioridade concentrada mais na sua rapidez computacional - (tempo). Mas, para DIST grande, ambos demonstraram bastante fragilidade na convergência.

2) Para "Quase-Newton", observamos aqui uma maior eficiência do programa MFØ1A com relação a "VA13A", sendo que, para DIST grande, nenhum dos dois conseguiram convergência.

3) Para DIST igual a 0.1 a 1. notamos que o algorítmo dos Gradientes Conjugados é mais eficiente do que o algorítmo Quase-Newton, mas para pontos iniciais longe da solução ele apresenta fragilidade, como os demais algorítmos.

4) As observações 4 e 5 da tabela anterior são aqui válidas.

TABELA 5

1) Para DIST igual a 0.1 e 1., "NEWMIN" é melhor do que "MFØ1A" . Para valores maiores de DIST, ambos são ineficientes, pois não conseguem atingir o ponto-solução da função.

2) Quanto ao método Quase-Newton, o programa MFØ1A é muito mais eficiente do que "VA13A" para valores pequenos de DIST. Para DIST igual a um, alguma comparação sobre eficiência seria precoce, pois há casos em que "VA13A" se apresenta muito melhor que "MFØ1A" e vice-versa.

3) Nesta tabela, novamente podemos observar a superioridade do método de Gradientes Conjugados sobre os métodos de Newton e Quase-Newton, com exceções, obviamente, dos casos em que DIST é

TABELA 5

FUNÇÃO-TESTE: CRAGG-LEVY

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MPØ1A"	NEW. MPØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MPØ1A"	DIST
							0.01 (*)
1, 2, 0.032	1, 5, 0.048	1, 2, 0.042	2, 9, 0.067	2, 7, 0.031	48, 61, 0.443	2, 6, 0.031	
2, 8, 0.047	2, 8, 0.058	3, 15, 0.077	3, 15, 0.101	2, 7, 0.040	38, 46, 0.403	2, 6, 0.033	0.1
1, 5, 0.106	1, 5, 0.048	2, 9, 0.046	2, 9, 0.063	2, 7, 0.042	23, 28, 0.248	2, 6, 0.044	
7, 8, 0.090	8, 28, 0.179	7, 8, 0.108	9, 33, 0.218	7, 15, 0.077	59, 72, 0.554	22, 24, 0.152	
10, 11, 0.124	9, 26, 0.212	10, 11, 0.122	7, 24, 0.168	6, 13, 0.059	35, 45, 0.416	13, 15, 0.121	1.
12, 13, 0.143	16, 65, 0.391	12, 13, 0.154	13, 60, 0.329	8, 18, 0.099	42, 52, 0.433	47, 49, 0.312	
NFLAG = 1	NFLAG = 1	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	
43, 45, 0.458	NFLAG = 1	38, 42, 0.415	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	10.
NFLAG = 2	NFLAG = 2	NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	NFLAG = 1	
NFLAG = 2	NFLAC = 2	NFLAG = 2	NFLAG = 2	NFLAG = 3	NFLAG = 1	NFLAG = 3	
NFLAG = 2	NFLAG = 2	NFLAG = 2	NFLAG = 2	NFLAG = 3	NFLAG = 1	NFLAG = 3	100.
NFLAG = 1	NFLAG = 1	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	

(*) Não consideramos DIST = 0.01, nesta Tabela, porque determinamos que o ponto-solução achado tivesse apenas uma aproximação de 10^{-2} .

NÃO CONVERGIU: Não tivemos informação da causa da não-convergência.

muito grande, pois aí este método também deixa de convergir.

4) Podemos notar uma superioridade de "Newton" em relação a "Quase-Newton para os casos de DIST = 0.1 e DIST = 1.

5) Quanto a "Newton Discreto", confirmamos a sua eficiência para pontos iniciais próximos à solução, apresentando poucas diferenças em relação a "Newton Analítico".

TABELA 6

1) Para DIST pequeno, podemos dizer que os programas NEWMIN e MFØ1A empatam; para DIST = 1. notamos que o número de avaliações da função no programa NEWMIN cresce bastante, acarretando-lhe uma eficiência menor. Quando DIST é igual a dez, também há um "empate" e para DIST igual a cem vemos que "MFØ1A" é melhor que "NEWMIN", visto que converge com um trabalho computacional bem menor.

2) Para o método Quase-Newton observamos que, para todos os valores de DIST, o programa VA13A converge num número de iterações e avaliações de função menor do que o programa MFØ1A, mas o seu tempo de C.P.U., mesmo assim, ainda é maior.

3) "Gradientes Conjugados" mostra-se algumas vezes superior a "Quase-Newton", mas é seguramente superior que "Newton" para "DIST" maior ou igual a dez.

4) Da mesma forma, "Quase-Newton" também é superior a "Newton" para valores de DIST iguais a dez e cem.

TABELA 6

FUNÇÃO-TESTE: VALE HELICAL

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
3, 4, 0.109	3, 4, 0.063	3, 4, 0.146	3, 4, 0.065	11, 27, 0.139	9, 12, 0.115	13, 18, 0.091	
3, 4, 0.043	3, 4, 0.054	3, 4, 0.037	3, 4, 0.051	13, 29, 0.118	9, 12, 0.116	13, 18, 0.092	0.01
3, 4, 0.029	3, 4, 0.056	3, 4, 0.044	3, 4, 0.035	9, 19, 0.075	13, 15, 0.136	19, 22, 0.131	
5, 9, 0.050	5, 9, 0.076	4, 7, 0.046	4, 7, 0.057	18, 40, 0.148	18, 19, 0.241	18, 22, 0.116	
5, 9, 0.046	5, 9, 0.076	5, 8, 0.043	5, 8, 0.065	8, 19, 0.077	9, 14, 0.093	20, 23, 0.119	0.1
3, 4, 0.028	3, 4, 0.055	4, 5, 0.069	4, 5, 0.066	10, 24, 0.096	9, 13, 0.110	16, 20, 0.099	
9, 34, 0.114	9, 34, 0.129	11, 25, 0.100	11, 21, 0.135	22, 49, 0.165	19, 24, 0.290	25, 30, 0.113	
9, 27, 0.121	9, 26, 0.132	10, 21, 0.089	13, 29, 0.192	21, 49, 0.184	20, 26, 0.209	23, 27, 0.173	1.
11, 26, 0.111	12, 27, 0.162	10, 16, 0.094	10, 16, 0.133	17, 39, 0.138	20, 29, 0.200	24, 29, 0.175	
33, 103, 0.308	35, 125, 0.584	31, 75, 0.291	26, 53, 0.335	29, 64, 0.241	46, 70, 0.447	42, 65, 0.265	
52, 128, 0.415	26, 61, 0.326	38, 74, 0.269	27, 49, 0.325	29, 67, 0.246	26, 43, 0.255	35, 46, 0.230	10.
21, 42, 0.153	26, 27, 0.316	23, 39, 0.169	25, 51, 0.270	29, 69, 0.238	22, 37, 0.245	41, 57, 0.266	
230, 366, 1.251	90, 133, 0.892	165, 268, 1.251	45, 99, 0.545	34, 76, 0.293	43, 69, 0.450	52, 81, 0.380	
136, 196, 0.997	17, 38, 0.216	27, 54, 0.211	14, 23, 0.150	28, 70, 0.235	45, 65, 0.434	33, 51, 0.208	100.
171, 226, 0.827	64, 98, 0.668	107, 185, 0.718	30, 67, 0.381	41, 95, 0.369	43, 60, 0.388	NFLAG = 1	

5) Excepcionalmente, notamos que, para esta função, o algorítmo de Newton Discreto é mais eficiente do que o algorítmo de Newton Analítico, para DIST igual a cem (sic...) .

TABELA 7

1) Para valores de DIST menores do que um, os dois programas do método de Newton convergem com poucas diferenças em seus trabalhos computacionais (número de iterações, número de avaliações e tempo). Para DIST igual a um, "MFØ1A" mostra-se um pouco mais eficiente, mas, para maiores valores de DIST, ambos os programas falham na convergência, sendo que "MFØ1A", na maioria dos casos, atinge o limite de avaliações de função, ao passo que "NEWMIN" falha na sua busca unidimensional.

2) Apesar do programa VA13A convergir em menas iterações do que MFØ1A, o seu número de avaliações de função é bem maior do que o número de avaliações de "MFØ1A" e, consequentemente, o seu tempo computacional também passa a ser maior. Mas, para valores de DIST maiores do que um, "VA13A" e "MFØ1A" deixam de atingir o ponto de mínimo da função.

3) "Gradientes Conjugados" mostra-se superior a "Quase-Newton" e menos eficiente do que "Newton" para valores de DIST menores ou iguais a um. Para DIST grande, ele também deixa de atingir o valor mínimo da função.

4) Obviamente, através das observações anteriores chegamos que o método de Newton, para esta função, é superior ao método Quase

TABELA 7

FUNÇÃO-TESTE: QUADRÁTICA TRIGONOMÉTRICA (n=1Ø)

NEWTON "NEWMIN"	NEWMIN DISCRETO	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
1, 2, 0.159	1, 2, 0.179	1, 2, 0.135	1, 2, 0.152	9, 19, 0.358	12, 22, 0.628	13, 15, 0.596	
1, 2, 0.062	1, 2, 0.214	1, 2, 0.108	1, 2, 0.192	9, 19, 0.489	12, 22, 0.658	11, 13, 0.439	0.01
1, 2, 0.061	1, 2, 0.155	1, 2, 0.062	2, 3, 0.251	9, 19, 0.328	13, 25, 0.668	12, 14, 0.367	
2, 3, 0.169	2, 3, 0.312	2, 3, 0.110	2, 3, 0.275	11, 23, 0.431	14, 24, 0.662	14, 16, 0.504	
2, 3, 0.124	2, 3, 0.287	2, 3, 0.106	2, 3, 0.261	11, 23, 0.407	14, 24, 0.770	16, 18, 0.559	0.1
2, 3, 0.109	2, 3, 0.315	2, 3, 0.106	2, 3, 0.439	11, 23, 0.461	14, 24, 0.942	15, 17, 0.637	
5, 6, 0.254	5, 6, 0.822	5, 6, 0.312	5, 6, 0.684	13, 27, 0.447	18, 27, 0.834	17, 18, 0.567	
5, 7, 0.240	6, 8, 0.799	5, 7, 0.227	6, 8, 0.774	14, 29, 0.509	15, 26, 0.772	18, 19, 0.591	1.
7, 10, 0.331	5, 8, 0.691	5, 8, 0.236	7, 11, 0.934	14, 29, 0.574	15, 25, 0.734	17, 18, 0.631	
NFLAG = 2	NFLAG = 2	NFLAG = 1	NFLAG = 2	NFLAG = 2	NÃO CONVERGIU	NFLAG = 1	
NFLAG = 2	NFLAG = 2	NFLAG = 2	NFLAG = 2	NFLAG = 1	NÃO CONVERGIU	NÃO CONVERGIU	10.
NFLAG = 2	NFLAG = 2	NFLAG = 1	NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	
NFLAG = 2	NFLAG = 2	NFLAG = 1	NFLAG = 2	NFLAG = 1	NÃO CONVERGIU	NFLAG = 1	
NFLAG = 2	NFLAG = 2	NFLAG = 1	NFLAG = 2	NFLAG = 1	NÃO CONVERGIU	NFLAG = 1	100.
NFLAG = 2	NFLAG = 2	NFLAG = 1	NFLAG = 2	NFLAG = 1	NÃO CONVERGIU	NÃO CONVERGIU	

NÃO CONVERGIU: Para este caso, não tivemos informação da causa da não-convergência.

-Newton, quando DIST assume valores pequenos.

5) Novamente, podemos notar através desta tabela que o método de Newton Discreto, apesar de convergir num número igual de iterações e avaliações de função, ao método de Newton, ainda é mais "caro" do que o mesmo.

TABELA 8

1) Observamos maiores diferenças entre os programas do algorítmo de Newton, quando DIST assume o valor *um*, pois, aí, "NEWMIN" é superior a "MFØ1A". Para valores menores de DIST, notamos que estes dois programas se distinguem apenas em centésimos de segundos de C.P.U.

2) Nesta tabela, o comportamento dos programas do algorítmo Quase-Newton se invertem diante da tabela anterior, visto que "VA13A" é mais eficiente que "MFØ1A", para quaisquer valores de DIST onde os mesmos convergem.

3) As observações 3, 4 e 5 da tabela anterior também podem ser aqui incluídas.

TABELA 9

1) Para a função-teste Elba (n = 3Ø), os programas do algorítmo de Newton se comportam semelhantemente ao caso anterior (Quadr. Trigonométrica (n = 20)), excluindo o fato dos mesmos convergirem para quaisquer valores de DIST.

TABELA 8

FUNÇÃO-TESTE: QUADRÁTICA TRIGONOMÉTRICA (n=2ø)

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØLA"	NEW. MFØLA "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØLA"	DIST
1, 2, 0.332	1, 2, 0.833	1, 2, 0.293	1, 2, 0.821	22, 45, 2.143	26, 38, 3.615	33, 35, 3.738	
1, 2, 0.301	1, 2, 0.859	1, 2, 0.225	1, 2, 0.833	20, 41, 1.950	26, 38, 3.677	32, 34, 3.610	0.01
1, 2, 0.223	2, 3, 1.661	1, 2, 0.301	2, 3, 1.677	22, 45, 2.197	24, 36, 3.400	37, 39, 4.238	
2, 3, 0.415	2, 3, 1.631	2, 3, 0.391	2, 3, 1.611	23, 47, 2.318	25, 34, 3.412	39, 41, 4.416	
2, 3, 0.426	2, 3, 1.582	2, 3, 0.404	2, 3, 1.681	27, 55, 2.699	25, 40, 3.538	37, 39, 4.178	0.1
2, 3, 0.406	2, 3, 1.593	2, 3, 0.389	2, 3, 1.593	27, 55, 2.652	26, 36, 3.500	39, 41, 4.390	
8, 10, 1.389	8, 10, 6.216	6, 8, 1.139	6, 8, 4.915	39, 79, 4.449	26, 34, 3.655	47, 48, 5.234	
4, 6, 1.048	5, 6, 4.081	6, 7, 1.136	6, 7, 4.869	30, 61, 3.002	27, 39, 3.949	43, 44, 4.851	1.
6, 7, 1.081(*)	6, 7, 4.960(*)	11, 16, 1.952(*)	11, 15, 9.024(*)	34, 69, 3.461	26, 42, 3.791	43, 44, 5.080	
NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	N. CONVERGIU	NFLAG = 1	
NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	N. CONVERGIU	NFLAH = 1	10.
158,677,45.578(*)	148,624,138.01(*)	62,112,11.927(*)	52, 92,42.744(*)	NFLAG = 1(*)	N.CONVERGIU(*)	NFLAG = 1 (*)	
NFLAG = 2	NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	N. CONVERGIU	NFLAG = 1	
NFLAG = 2	NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 2	N. CONVERGIU	NFLAG = 1	100.
NFLAG = 2	NFLAG = 2	NÃO CONVERGIU	NÃO CONVERGIU	NFLAG = 1	N. CONVERGIU	NFLAG = 1	

(*) Aproximação ao ponto-solução foi ruim.

NÃO CONVERGIU: Não tivemos informação da causa da não-convergência.

TABELA 9

FUNÇÃO-TESTE: ELBA (n = 30)

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MF01A"	NEW. MF01A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MF01A"	DIST
1, 2, 0.628	1, 2, 2.245	1, 2, 0.608	2, 3, 4.374	3, 7, 0.497	10, 17, 2.821	3, 5, 0.611	
1, 2, 0.565	1, 2, 2.160	1, 2, 0.530	2, 3, 4.516	3, 7, 0.504	10, 15, 2.679	3, 5, 0.625	0.01
1, 2, 0.580	1, 2, 2.275	1, 2, 0.564	2, 3, 4.327	3, 7, 0.516	10, 17, 2.904	3, 5, 0.598	
1, 2, 0.527	2, 3, 4.323	1, 2, 0.540	2, 3, 4.313	5, 10, 0.741	13, 19, 3.626	4, 6, 0.850	
1, 2, 0.547	2, 3, 4.261	3, 2, 0.548	2, 3, 4.486	3, 7, 0.528	10, 17, 2.792	3, 5, 0.627	0.1
1, 2, 0.553	2, 3, 4.271	1, 2, 0.516	2, 3, 4.335	4, 9, 0.671	10, 17, 2.921	4, 6, 0.867	
1, 2, 0.529	2, 3, 4.323	1, 2, 0.510	2, 3, 4.926	4, 8, 0.838	10, 16, 2.989	4, 5, 0.951	
1, 2, 0.636	2, 3, 4.988	1, 2, 0.608	2, 3, 5.195	5, 10, 0.881	10, 17, 3.208	4, 5, 0.974	1.
1, 2, 0.592	2, 3, 4.740	1, 2, 0.636	2, 3, 4.886	5, 11, 0.892	10, 17, 2.978	4, 5, 0.811	
1, 2, 0.564	2, 3, 4.751	2, 3, 1.098	2, 3, 4.883	4, 9, 0.699	10, 17, 3.124	5, 7, 1.135	
1, 2, 0.539	2, 3, 4.834	2, 3, 1.046	2, 3, 4.682	4, 9, 0.709	8, 14, 2.443	5, 7, 1.134	10.
1, 2, 0.552	2, 3, 4.544	2, 3, 1.024	2, 3, 4.413	5, 12, 0.895	10, 17, 3.019	5, 7, 1.077	
2, 3, 1.016	3, 4, 6.297	2, 3, 0.986	3, 4, 6.439	5, 11, 0.819	10, 17, 2.853	5, 8, 1.137	
2, 3, 1.076	2, 3, 4.293	2, 3, 0.999	3, 4, 6.200	5, 10, 0.715	9, 14, 2.380	5, 7, 1.012	100.
2, 3, 0.987	2, 3, 4.454	2, 3, 1.010	2, 3, 4.505	5, 10, 0.722	10, 17, 2.742	5, 7, 1.037	

2) Neste caso, o programa MFØ1A, para "Quase-Newton", mostra-se mais eficiente do que o programa VA13A, para todos os valores de DIST.

3) Aqui, o algorítmo de Gradientes Conjugados é "mais barato" do que o algorítmo "Quase-Newton". Com relação ao algorítmo de Newton, o mesmo observamos para valores de DIST iguais a um e maiores que dez, o que nos ajuda a confirmar a "robustez" de "Gradientes Conjugados" nos casos em que o ponto inicial dado se encontra distante do ponto-solução.

4) Para todos os valores de DIST, o algorítmo de Newton é superior ao algorítmo Quase-Newton.

5) Para esta função, "Newton Discreto" torna-se muito "mais caro" do que "Newton Analítico", pois o seu tempo computacional é bastante grande.

TABELA 10

1) Para o algorítmo de Newton, notamos aqui que ambos os seus programas se mostram superior ou inferior, se comparados, em vários casos e isto se relaciona somente com o tempo computacional dos mesmos.

2) Quanto a "Quase-Newton", a mesma observação anterior pode ser considerada.

3) Já "Gradientes Conjugados" se mostra superior a ambos os algorítmos, Newton e Quase-Newton, para quaisquer valores de DIST.

TABELA 10

FUNÇÃO-TESTE: ELBA (n = 6Ø)

NEWTON "NEWMIN"	NEWMIN "DISCRETO"	NEWTON "MFØ1A"	NEW. MFØ1A "DISCRETO"	GRADIENTES CONJUGADOS	Q.N. "VA13A"	Q.N. "MFØ1A"	DIST
1, 2, 3.036	2, 3, 31.602	1, 2, 3.068	2, 3, 29.995	3, 7, 1.680	9, 15, 8.834	3, 5, 2.197	
1, 2, 2.937	2, 3, 31.136	1, 2, 2.958	2, 3, 31.924	3, 7, 1.640	9, 15, 8.887	3, 5, 2.169	0.01
1, 2, 2.943	2, 3, 30.778	1, 2, 2.966	2, 3, 30.884	3, 7, 1.689	9, 15, 8.724	3, 5, 2.151	
1, 2, 3.149	2, 3, 31.642	1, 2, 2.993	2, 3, 31.292	3, 7, 1.682	9, 15, 8.621	3, 5, 2.210	
1, 2, 3.002	2, 3, 30.305	1, 2, 2.922	2, 3, 30.484	3, 7, 1.759	8, 14, 7.952	3, 5, 2.182	0.1
1, 2, 2.923	2, 3, 30.474	1, 2, 2.928	2, 3, 30.718	3, 7, 1.675	9, 15, 8.869	3, 5, 2.206	
1, 2, 3.341	2, 3, 31.277	1, 2, 3.224	2, 3, 31.026	5, 10, 2.423	11, 17,10.009	4, 5, 2.826	
1, 2, 3.021	2, 3, 30.853	1, 2, 2.987	2, 3, 31.612	5, 10, 2.794	6, 11, 6.013	4, 5, 2.907	1.
1, 2, 3.074	2, 3, 31.135	1, 2, 2.956	2, 3, 31.503	4, 9, 2.256	9, 15, 8.690	4, 5, 2.897	
2, 3, 5.638	2, 3, 30.362	2, 3, 5.643	2, 3, 30.447	4, 9, 2.133	9, 15, 8.649	4, 6, 3.010	
2, 3, 5.569	2, 3, 30.941	2, 3, 5.474	2, 3, 30.955	4, 8, 1.947	9, 15, 8.701	4, 6, 3.049	10.
2, 3, 5.622	2, 3, 31.028	2, 3, 5.593	2, 3, 31.480	4, 8, 1.988	7, 13, 7.277	4, 6, 3.236	
2, 3, 6.097	3, 4, 46.742	2, 3, 5.941	3, 4, 46.896	5, 10, 2.426	9, 15, 8.991	5, 7, 3.891	
2, 3, 5.773	3, 4, 46.649	2, 3, 5.717	3, 4, 46.429	5, 12, 2.956	8, 14, 7.894	5, 8, 4.079	100.
2, 3, 5.623	2, 3, 29.859	2, 3, 5.565	3, 4, 45.156	5, 12, 2.790	9, 14, 8.543	5, 8, 4.075	

4) As observações 4 e 5 da tabela anterior são aqui válidas.

Quanto à aproximação da solução "achada" à solução real, dada pelos algorítmos aqui tratados, afirmamos que, quando convergem, todos os algorítmos aqui referidos apontam para um ponto com boa aproximação da solução, não existindo diferenças grandes entre os mesmos de modo tal que implique em alguma alteração ou invalidação das tabelas observadas acima.

Devemos fazer, aqui, apenas uma ressalva para os casos em que a aproximação foi ruim (Tabela 8), visto que, para um deles, podemos observar a superioridade de "Gradientes Conjugados" e "Quase-Newton" sobre "Newton"; mas, sendo esta amostra muito pequena (apenas um caso entre muitos considerados), pode ser desprezada.

Deixamos de incluir, nas tabelas anteriores, os dados referentes às experiências realizadas com o algorítmo de "Newton Modificado", isto porque tais dados foram extraídos de experiências realizadas em condições diferentes das que nos referimos até agora; ou seja, os pontos iniciais gerados foram calculados de forma diferente, mas, o que é importante, obedecendo o parâmetro "DIST"; isto é, consideramos, agora,

$$X(I) = SOL(I) + P(I)*DIST/PNOR$$

onde $PNOR = \sqrt{\Sigma P(I)^2}$, $P(I) = RAN(1.)*2-1$, e $RAN(.)$ é a subrotina do sistema, geradora de números aleatórios entre (0,1) .

Em segundo lugar, o número máximo de avaliações determinado para estas experiências é $MXFUN = 50*N$, o que pode nos expli-

car, em alguns casos (como veremos), a não convergência deste algorítmo diante da convergência do algorítmo de Newton, pois, anteriormente, MXFUN = 500*N.

Mesmo assim, diante destas diferenças, ainda podemos observar e comparar os comportamentos destes dois algorítmos através das tabelas anteriores e das tabelas 11 e 12, as quais descrevem, da mesma forma, o número de iterações, número de avaliações de função e tempo computacional gastos pelo algorítmo de "Newton Mofificado" para obter convergência.

Dessa maneira, podemos verificar que:

TABELA 11

1) Para a função-teste Rosenbrock, ambos os algorítmos, Newton e Newton Modificado, assumem comportamentos semelhantes, com exceção a DIST igual a dez, onde "Newton Modificado" é mais custoso, e DIST igual a cem, onde o mesmo deixa de convergir, apesar de já termos salientado as diferenças entre os valores do parâmetro MXFUN, utilizado nas duas experiências.

2) Já para a função-teste Banana-de-Wood, podemos notar comportamentos bastante semelhantes entre estes dois algorítmos, para todos os valores de DIST.

3) Nas experiências com a função Quártica-de-Powell, observamos que o algorítmo de Newton converge mais rapidamente do que o algorítmo de Newton Modificado, e isto se deve tanto ao número de iterações quanto ao número de avaliações da função e tempo computacional. Porém, não devemos esquecer, aqui, que os pontos -

TABELA 11

MÉTODO DE NEWTON "MODIFICADO"

F	MÉTODO	DIST = 0.01	DIST = 0.1	DIST = 1.	DIST = 10.	DIST = 100.
ROSENBROCK	NEWTON ANALÍTICO	4, 5, 0.040	4, 5, 0.049	14, 18, 0.068	39, 53, 0.228	NFLAG = 1
		4, 5, 0.034	4, 5, 0.032	14, 18, 0.071	45, 62, 0.212	NFLAG = 1
		2, 3, 0.031	5, 6, 0.037	14, 18, 0.066	36, 49, 0.213	NFLAG = 1
	NEWTON DISCRETO	4, 5, 0.025	5, 6, 0.052	15, 18, 0.080	40, 53, 0.245	NFLAG = 1
		4, 5, 0.036	4, 5, 0.045	14, 18, 0.081	48, 66, 0.253	NFLAG = 1
		3, 4, 0.032	5, 6, 0.048	14, 18, 0.091	37, 48, 0.195	NFLAG = 1
BANANA DE WOOD	NEWTON ANALÍTICO	4, 5, 0.042	7, 8, 0.080	9, 11, 0.104	21, 25, 0.215	27, 28, 0.321
		2, 3, 0.016	7, 8, 0.065	11, 17, 0.125	27, 35, 0.234	20, 21, 0.197
		3, 4, 0.032	4, 5, 0.044	7, 8, 0.082	17, 19, 0.163	38, 45, 0.423
	NEWTON DISCRETO	4, 5, 0.074	7, 8, 0.152	9, 11, 0.146	22, 26, 0.306	81, 83, 1.302
		2, 3, 0.042	7, 8, 0.109	11, 15, 0.184	26, 30, 0.375	34, 35, 0.459
		3, 4, 0.029	4, 5, 0.070	7, 8, 0.093	18, 20, 0.228	75, 81, 1.180
QUART. POWELL	NEWTON ANALÍTICO	NFLAG = 1	11, 12, 0.092	16, 17, 0.161	23, 24, 0.198	27, 28, 0.219
		5, 6, 0.057	8, 9, 0.077	17, 18, 0.137	23, 24, 0.183	29, 30, 0.238
		6, 7, 0.050	11, 12, 0.092	17, 18, 0.133	23, 24, 0.182	29, 30, 0.235
	NEWTON DISCRETO	2, 3, 0.031	11, 12, 0.134	16, 17, 0.194	23, 24, 0.256	34, 42, 0.425
		6, 7, 0.071	8, 9, 0.095	17, 18, 0.192	23, 24, 0.276	40, 44, 0.473
		6, 7, 0.072	11, 12, 0.134	17, 18, 0.200	24, 25, 0.284	38, 44, 0.457
BOX	NEWTON ANALÍTICO	2, 3, 0.207	2, 3, 0.208	6, 7, 0.504	NFLAG = 1	NFLAG = 2
		2, 3, 0.186	2, 3, 0.199	5, 6, 0.432	26, 30, 2.396	NFLAG = 1
		1, 2, 0.116	2, 3, 0.215	4, 5, 0.348	8, 9, 0.091	NFLAG = 1
	NEWTON DISCRETO	2, 3, 0.215	3, 4, 0.306	4, 5, 0.412	NFLAG = 2	NFLAG = 1
		2, 3, 0.222	3, 4, 0.318	4, 5, 0.408	NFLAG = 1	NFLAG = 1
		2, 3, 0.241	3, 4, 0.328	4, 5, 0.421	7, 8, 0.690	NFLAG = 1
CRAGG LEVY	NEWTON ANALÍTICO		11, 12, 0.178	19, 20, 0.274	33, 38, 0.486	NFLAG = 1
			6, 7, 0.105	20, 21, 0.283	NFLAG = 2	NFLAG = 1
			10, 11, 0.154	17, 19, 0.243	NFLAG = 1	NFLAG = 1
	NEWTON DISCRETO		11, 12, 0.223	18, 19, 0.385	NFLAG = 2	NFLAG = 1
			6, 7, 0.131	20, 21, 0.433	NFLAG = 2	NFLAG = 1
			10, 11, 0.191	17, 19, 0.320	NFLAG = 2	NFLAG = 1

iniciais considerados para estes dois algorítmos são diferentes, apesar de estarem a uma mesma distância do ponto-solução.

4) Nas experiências com a Função de Box, também observamos semelhanças no comportamento destes algorítmos, salvo quando DIST é igual a cem, pois "Newton Modificado" deixa de convergir por falhar na busca unidimensional, ao melhorar o valor da função, e "Newton" por falhar na sua "direção de descida".

5) Para a função-teste de Cragg-Levy, notamos diferenças maiores entre os dois algorítmos, pois "Newton Modificado" nos parece ser mais "caro" do que "Newton", para os casos de convergência. Para os casos de não-convergência, a mesma observação do item 4 pode aqui ser feita.

TABELA 12

1) Para a função-teste Vale Helical, notamos diferenças entre os algorítmos de Newton e Newton Modificado, quando DIST é grande, ou seja, para DIST igual a dez, "Newton Modificado" apresenta um maior número de avaliações da função *por iteração*, mas, para DIST igual a cem, este mesmo algorítmo se mostra mais eficiente do que o algorítmo de Newton.

2) Já para a função-teste Quadrática Trigonométrica (n =1Ø), o algorítmo de "Newton Modificado" e o algorítmo de Newton "empatam" quando DIST é pequeno. Para DIST igual a um, "Newton Modificado" é pouco mais eficiente que Newton e, para DIST igual a dez, o primeiro converge várias vezes, o que o último não conse

TABELA 12

MÉTODO DE NEWTON "MODIFICADO"

F	MÉTODO	DIST = 0.01	DIST = 0.1	DIST = 1.	DIST = 10.	DIST = 100.
VALE HELICAL	NEWTON ANALÍTICO	3, 4, 0.056	8, 16, 0.091	9, 12, 0.080	45, 125, 0.374	37, 94, 0.346
		3, 4, 0.054	5, 11, 0.056	11, 18, 0.110	11, 20, 0.106	36, 82, 0.318
		3, 4, 0.046	5, 6, 0.061	5, 6, 0.060	26, 66, 0.234	
	NEWTON DISCRETO	3, 4, 0.058	8, 20, 0.110	12, 23, 0.171	49, 140, 0.606	21, 41, 0.251
		3, 4, 0.053	6, 12, 0.078	10, 18, 0.131	35, 86, 0.396	17, 31, 0.196
		3, 4, 0.051	4, 5, 0.054	5, 6, 0.075	22, 50, 0.249	
QUAD TRIG N=10	NEWTON ANALÍTICO	1, 2, 0.079	2, 3, 0.113	3, 4, 0.173	NFLAG = 2	NFLAG = 2
		1, 2, 0.098	2, 3, 0.131	3, 4, 0.167	10, 15, 0.582	NFLAG = 2
		1, 2, 0.066	2, 3, 0.118	5, 6, 0.259	19, 30, 1.098	NFLAG = 2
	NEWTON DISCRETO	1, 2, 0.154	2, 3, 0.241	3, 4, 0.359	NFLAG = 2	NFLAG = 2
		1, 2, 0.150	2, 3, 0.232	3, 4, 0.361	8, 12, 0.939	NFLAG = 2
		1, 2, 0.136	2, 3, 0.219	5, 6, 0.559	NFLAG = 2	NFLAG = 2
QUAD TRIG N=20	NEWTON ANALÍTICO	1, 2, 0.301	2, 3, 0.466	3, 4, 0.662	NFLAG = 2	NFLAG = 2
		1, 2, 0.341	2, 3, 0.466	3, 4, 0.660	NFLAG = 2	NFLAG = 2
		1, 2, 0.302	2, 3, 0.463	3, 4, 0.672	NFLAG = 2	NFLAG = 2
	NEWTON DISCRETO	1, 2, 0.836	2, 3, 1.251	3, 4, 1.755	NFLAG = 2	NFLAG = 2
		1, 2, 0.811	2, 3, 1.238	3, 4, 1.750	NFLAG = 2	NFLAG = 2
		1, 2, 0.885	2, 3, 1.217	3, 4, 1.787	NFLAG = 2	NFLAG = 2
ELBA N=30	NEWTON ANALÍTICO	1, 2, 0.968	1, 2, 0.954	1, 2, 0.964	1, 2, 0.964	1, 2, 0.965
		1, 2, 0.976	1, 2, 0.958	1, 2, 0.966	1, 2, 0.975	1, 2, 0.973
		1, 2, 0.989	1, 2, 0.956	1, 2, 0.966	1, 2, 0.978	1, 2, 0.966
	NEWTON DISCRETO	1, 2, 5.551	1, 2, 5.559	2, 3, 10.938	2, 3, 11.087	2, 3, 10.884
		1, 2, 5.552	1, 2, 5.559	2, 3, 10.911	2, 3, 10.906	2, 3, 10.910
		1, 2, 5.560	1, 2, 5.530	2, 3, 10.914	2, 3, 10.903	2, 3, 10.907
ELBA N=60	NEWTON ANALÍTICO	1, 2, 4.723	1, 2, 4.719	1, 2, 4.693	1, 2, 4.711	1, 2, 4.697
		1, 2, 4.706	1, 2, 4.716	1, 2, 4.770	1, 2, 4.690	1, 2, 4.739
		1, 2, 4.734	1, 2, 4.698	1, 2, 4.710	1, 2, 4.693	1, 2, 4.700
	NEWTON DISCRETO	1, 2, 42.909	1, 2, 42.947	2, 3, 85.225	2, 3, 84.723	2, 3, 85.016
		1, 2, 42.741	1, 2, 43.071	2, 3, 84.939	2, 3, 85.371	2, 3, 85.642
		1, 2, 42.720	1, 2, 42.677	2, 3, 84.614	2, 3, 84.920	2, 3, 84.975

gue fazer.

3) Para a mesma função anterior, mas com vinte variáveis, notamos que "Newton Modificado" é melhor do que "Newton" quando DIST é igual a um. Para outros valores de DIST, os mesmos apresentam - um comportamento semelhante.

4) Para a função Elba, notamos comportamentos idênticos de ambos os algorítmos, tanto quando o número de variáveis desta função é igual a trinta, como quando o número de variáveis é igual a sessenta; ou seja, o algorítmo de Newton Modificado é superior ao algorítmo de Newton para DIST igual a cem, e, podemos notar, também, que o seu tempo computacional se mantém estável à medida em que o ponto inicial se distancia do ponto-solução, o que não acontece com o algorítmo de Newton, pois este apresenta número maior de iterações e avaliações da função, quando DIST é grande.

4.6) Falemos, agora, do "quociente previsto" entre o número de iterações de Newton e o número de iterações de Quase-Newton, necessários para convergirem, e do "quociente real", conseguido através das tabelas descritas na secção anterior.

Lembrando que α_2 é a ordem de convergência do algorítmo Quase-Newton e α_1 a ordem de convergência do algorítmo de Newton, e denotando $\frac{\log \alpha_2}{\log \alpha_1} = QUOC$, construímos a tabela 13.

Lembramos aqui, também, que α_2 e QUOC já se encontram calculados na Tabela NEWQUA (Apêndice) .

TABELA 13

FUNÇÃO	N	α_2	QUOC	EXPERIÊNCIA				
				DIST=0.01	DIST=0.1	DIST=1.	DIST=10.	DIST=100.
ROSENBROCK	2	1.47	0.55	0.47	0.34	0.71	0.60	1.70
BANANA DE WOOD	4	1.33	0.41	0.15	0.31	0.36	0.23	0.33
QUART. POWELL	4	1.33	0.41	0.16	0.19	0.39	0.30	0.27
BOX	2	1.47	0.55	0.50	0.23	0.42	0.93	-
CRAGG-LEVY	4	1.33	0.41	-	1.00	0.35	-	-
VALE HELICAL	3	1.38	0.47	0.20	0.29	0.43	0.78	2.25
QUADR. TRIGONOM.	10	1.19	0.25	0.08	0.13	0.28	-	-
QUADR. TRIGONOM.	20	1.12	0.16	0.03	0.05	0.17	-	-
ELBA	30	1.09	0.12	0.33	0.27	0.25	0.40	0.40
ELBA	60	1.05	0.07	0.33	0.33	0.25	0.50	0.40

O que podemos observar na Tabela 13 é que, com exceção à função Rosenbrock, o "quociente" obtido através de nossas experiências se aproxima de "QUOC" à medida em que DIST se aproxima de um, o que pode ser explicado pelo fato do algorítmo Quase-Newton se "recuperar" bastante diante do algorítmo de Newton, para pontos-iniciais mais distantes da solução. E este fato está bastante relacionado com a aproximação da matriz "B_k", obtida a cada iteração k, a qual não é satisfatória nas primeiras iterações. (Esses dados foram extraídos de cálculos da *média aritmética* dos números de iterações dos algorítmos "MFØlA", para cada valor de DIST.)

Vamos considerar estes mesmos algorítmos para verificarmos a real validade das estimativas "1.2" e "1.3" utilizadas na construção da "Função de Decisão", do Capítulo 3, e que se referem ao número de avaliações-de-função por iteração gastas pelos mesmos.

Como podemos observar, para algumas funções-teste obtive

TABELA 14

FUNÇÃO	N	ESTIMATIVA PARA NEWTON					MÉDIA	ESTIMATIVA PARA QUASE-NEWTON					MÉDIA
		DIST=0.01	DIST=0.1	DIST=1.	DIST=10.	DIST=100.		DIST=0.01	DIST=0.1	DIST=1.	DIST=10.	DIST=100.	
ROSENBROOK	2	1.5	1.45	1.42	1.41	1.5	1.44	2.1	1.3	1.1	1.3	1.4	1.44
BANANA DE WOOD	4	1.3	1.4	1.3	1.1	1.1	1.24	1.1	1.2	2.4	1.2	1.2	1.42
QUÁRTICA POWELL	4	2.	1.3	1.1	1.1	1.	1.3	1.3	1.1	1.1	1.1	1.1	1.14
BOX	2	2.	2.2	1.2	1.4	-	1.7	2.3	1.2	1.3	1.1	-	1.4
CRAGG-LEVY	4	-	4.3	1.1	-	-	2.7	-	3.	1.	-	-	2.
VALE HELICAL	3	1.7	1.5	1.2	2.	1.7	1.6	1.3	1.2	1.2	1.4	1.5	1.3
QUADR. TRIGONOM.	10	2.	2.2	1.4	-	-	1.8	1.16	1.13	1.3	-	-	1.2
QUADR. TRIGONOM.	20	2.	2.2	1.3	-	-	1.8	1.05	1.05	1.02	-	-	1.04
MÉDIAS		1.78	2.00	1.25	1.40	1.3	1.70	1.47	1.4	1.3	1.22	1.37	1.37

mos o número de avaliações-de-função do algorítmo de Newton bastante alterado; mas acreditamos que esta alteração não é suficientemente significativa a ponto de invalidar a tabela NEWQUA.

CONCLUSÕES

Diante das nossas observações feitas em cada tabela deste trabalho, podemos concluir que:

1) A modificação apresentada na busca unidimensional do programa do algorítmo de Newton, "MFØ1A", diante da busca unidimensional da "NEWMIN", não nos traz qualquer vantagem com relação a convergência deste algorítmo. Longe disso, muitas vezes se mostrou inferior ao programa NEWMIN.

2) Quanto ao algorítmo Quase-Newton, não nos parece ser a forma da "VA13A" a mais adequada para se armazenar a matriz aproximada - do hessiano, visto que este programa, na maioria dos casos, se mostra muito mais "caro" do que o programa MFØ1A. Mesmo quando o número de iterações e o número de avaliações de função da "VA13A" era pequeno, ainda apresentava um tempo computacional - maior do que o tempo de "MFØ1A", como observamos na Tabela 6.

3) Com relação a Gradientes Conjugados, podemos dizer que este - apresenta superioridade com relação aos algorítmos de Newton e Quase-Newton na maioria dos casos em que o ponto-inicial estava a uma distância grande do ponto-solução. Em alguns casos (Tabelas 7, 8, 9, 10) ele se mostrou superior a Quase-Newton para todos os valores de DIST.

4) Comparando "Newton" e "Quase-Newton", o primeiro é superior ao segundo em quase todas as experiências realizadas, exceto para pontos-iniciais longe da solução, pois neste caso o algorítmo

de Newton se mostrou ineficiente, tanto pelo seu "custo computacional" quanto pela sua convergência (Tabela 8).

5) Tratando do algorítmo de Newton com hessiano discretizado, as mesmas observações feitas anteriormente nos ajudam a concluir que ainda é válido utilizarmos este algorítmo para os casos em que a função a ser minimizada não possui derivada segunda. Basta "chutarmos bem" um ponto-inicial.

6) O algorítmo de Newton Modificado se comportou semelhantemente ao algorítmo de Newton, chegando a convergir em ocasiões onde "Newton" não convergiu (Tabela 7) .

7) Quanto à "função-de-decisão" construída no Capítulo 3, podemos dizer que ela pode nos dar informações, mais ou menos seguras , de quando devemos usar o algorítmo de Newton ou o algorítmo Quase-Newton, dada a função em questão. E afirmamos isso diante dos resultados da Tabela 13, onde comprovamos que a "predição QUOC" é bastante razoável para pontos iniciais *não muito distantes da solução.*

Pode surgir, aqui, uma dúvida de como conseguirmos tal "distância" que seja "perfeita" para fazermos tal predição. Em geral, o que sempre acontece é que procuramos um ponto-inicial que seja o melhor possível e, nesta situação, jogamos com a probabilidade de também podermos errar!

CONSIDERAÇÕES:

Com a teoria descrita no Capítulo 3 acerca da previsão do número de iterações necessárias para certos algorítmos convergirem,

no nosso caso "Newton" e "Quase-Newton", podemos avaliar, a priori, o *custo computacional* de cada um desses algorítmos diante das funções requeridas (como fizemos no exemplo do Capítulo 3).

Vimos que, para várias funções-teste, o algorítmo de Newton é mais eficiente do que o algorítmo Quase-Newton quando o ponto inicial dado está próximo ao ponto-solução.

O fato dessas funções possuirem poucas variáveis (com exceção da "Elba", n=60, mas que é quadrática) nos faz ficar "devendo" experiências com funções "maiores" para que pudéssemos observar melhor o comportamento desses algorítmos; porém, sobrecarregaríamos a memória do "PDP10" utilizado para essas experiências...!

Quanto ao algorítmo dos Gradientes Conjugados, ele realmente se apresenta mais "robusto" em relação aos algorítmos "Newton" e "Quase-Newton", para pontos-iniciais distantes da solução . Para "n" grande, ele também apresenta um custo computacional bem inferior aos demais, o que já era esperado por nós, conforme as observações do Capítulo 1.

Talvez esperássemos que o algorítmo de Newton Modificado fosse mais eficiente do que o algorítmo de Newton para pontos onde o hessiano é indefinido, o que supomos não ter acontecido, pois nos faltou analisar o hessiano em cada ponto achado, a cada iteração.

Porém, temos que o algorítmo de Newton apresentou um comportamento muito bom diante das funções-teste utilizadas, o que já nos basta para também concluirmos sobre o bom comportamento de - "Newton Modificado".

PROPOSTA:

O nosso interesse principal, neste trabalho, foi o de encontrarmos um algorítmo tão bom quanto ou melhor do que o algorítmo de Newton, para pontos em que o hessiano da função fosse indefinido. Fizemos, então, o estudo de uma outra possível implementação do algorítmo de Newton, que é a *programação quadrática local* da função e vimos, através de exemplos, a possibilidade de obtermos - um outro algorítmo que possa tratar com eficiência o problema da "indefinição" do hessiano.

Deixamos de programar tal algorítmo visto que o nosso tempo era escasso e programá-lo requer bastante trabalho; mas, esperamos que tais estudos possam ser levados adiante.

Um estudo mais minucioso deste algorítmo poderia ser feito em cima do fato de, a cada iteração, *tomarmos todas as variáveis livres para obtermos a nova direção de curvatura negativa*.Nos exemplos mostrados, pode ser verificado que tal escolha leva a uma convergência mais rápida do algorítmo; mas, para funções com um número maior de variáveis, talvez possam surgir problemas de estabilidade numérica. Quais, teríamos que verificar.

Interessante seria verificar em quais circunstâncias tais problemas podem ocorrer e daí tirar proveito das circunstâncias em que o uso de todas as variáveis livres pode ser feito.

A P Ê N D I C E

TABELA NEWQUA

N	T	ALT	UMS	QUOC	BN	MN
2	.146557E+01	.382245E+00	.261612E+01	.551463E+00	.418408E+02	.115737E+01
3	.138029E+01	.322285E+00	.310285E+01	.464958E+00	.885916E+02	.159595E+01
4	.132472E+01	.281200E+00	.355619E+01	.405685E+00	.155987E+03	.200445E+01
5	.128520E+01	.250914E+00	.398544E+01	.361992E+00	.246499E+03	.239124E+01
6	.125542E+01	.227472E+00	.439614E+01	.328173E+00	.361500E+03	.276132E+01
7	.123205E+01	.208693E+00	.479195E+01	.301066E+00	.505270E+03	.311799E+01
8	.121315E+01	.193220E+00	.517545E+01	.278758E+00	.679011E+03	.346355E+01
9	.119749E+01	.180229E+00	.554850E+01	.260015E+00	.884854E+03	.379971E+01
10	.118429E+01	.169132E+00	.591265E+01	.244006E+00	.112487E+04	.412774E+01
11	.117295E+01	.159523E+00	.626870E+01	.230142E+00	.140106E+04	.444868E+01
12	.116312E+01	.151106E+00	.661788E+01	.218000E+00	.171441E+04	.476331E+01
13	.115449E+01	.143682E+00	.696079E+01	.207260E+00	.206882E+04	.507231E+01
14	.114685E+01	.137023E+00	.729807E+01	.197682E+00	.246517E+04	.537623E+01
15	.114003E+01	.131058E+00	.763021E+01	.189077E+00	.290531E+04	.567551E+01
16	.113390E+01	.125665E+00	.795765E+01	.181297E+00	.339104E+04	.597057E+01
17	.112836E+01	.120762E+00	.828077E+01	.174222E+00	.392413E+04	.626173E+01
18	.112331E+01	.116280E+00	.859990E+01	.167757E+00	.450533E+04	.654930E+01
19	.111870E+01	.112166E+00	.891532E+01	.161822E+00	.513836E+04	.683352E+01
20	.111446E+01	.108374E+00	.922727E+01	.156351E+00	.582391E+04	.711462E+01
21	.111056E+01	.104866E+00	.953599E+01	.151289E+00	.656363E+04	.739280E+01
22	.110695E+01	.101609E+00	.984168E+01	.146590E+00	.735919E+04	.766825E+01
23	.110360E+01	.985756E-01	.101445E+02	.142215E+00	.821220E+04	.794112E+01
24	.110048E+01	.957431E-01	.104446E+02	.138128E+00	.912326E+04	.821156E+01
25	.109756E+01	.930909E-01	.107422E+02	.134302E+00	.100959E+05	.847969E+01
26	.109483E+01	.906016E-01	.110373E+02	.130710E+00	.111308E+05	.874565E+01
27	.109227E+01	.882598E-01	.113302E+02	.127333E+00	.122295E+05	.900953E+01
28	.108986E+01	.860523E-01	.116208E+02	.124147E+00	.133934E+05	.927143E+01

29	.108759E+01	.839673E-01	.119094E+02	.121139E+00	.146241E+05	.108875E+02
30	.108545E+01	.819943E-01	.121960E+02	.118293E+00	.159221E+05	.111804E+02
31	.108342E+01	.801242E-01	.124806E+02	.115595E+00	.172909E+05	.114764E+02
32	.108150E+01	.783486E-01	.127635E+02	.113033E+00	.187309E+05	.117704E+02
33	.107968E+01	.766604E-01	.130445E+02	.110598E+00	.202436E+05	.120627E+02
34	.107794E+01	.750529E-01	.133239E+02	.108279E+00	.218305E+05	.123532E+02
35	.107629E+01	.735203E-01	.136017E+02	.106067E+00	.234929E+05	.126420E+02
36	.107472E+01	.720570E-01	.138779E+02	.103956E+00	.252314E+05	.129291E+02
37	.107321E+01	.706584E-01	.141526E+02	.101939E+00	.270492E+05	.132147E+02
38	.107178E+01	.693200E-01	.144258E+02	.100009E+00	.289469E+05	.134988E+02
39	.107041E+01	.680379E-01	.146977E+02	.981580E-01	.309257E+05	.137815E+02
40	.106909E+01	.668084E-01	.149682E+02	.963842E-01	.329872E+05	.140627E+02
41	.106783E+01	.656282E-01	.152374E+02	.946815E-01	.351325E+05	.143426E+02
42	.106662E+01	.644943E-01	.155053E+02	.930455E-01	.373621E+05	.146211E+02
43	.106546E+01	.634038E-01	.157719E+02	.914723E-01	.396794E+05	.148984E+02
44	.106434E+01	.623542E-01	.160374E+02	.899581E-01	.420847E+05	.151744E+02
45	.106326E+01	.613432E-01	.163017E+02	.884995E-01	.445792E+05	.154492E+02
46	.106223E+01	.603685E-01	.165649E+02	.870934E-01	.471644E+05	.157229E+02
47	.106123E+01	.594282E-01	.168270E+02	.857368E-01	.498415E+05	.159954E+02
48	.106027E+01	.585204E-01	.170881E+02	.844271E-01	.526108E+05	.162668E+02
49	.105934E+01	.576433E-01	.173481E+02	.831617E-01	.554756E+05	.165371E+02
50	.105844E+01	.567954E-01	.176071E+02	.819384E-01	.584363E+05	.168064E+02
51	.105757E+01	.559751E-01	.178651E+02	.807550E-01	.614940E+05	.170747E+02
52	.105673E+01	.551811E-01	.181221E+02	.796095E-01	.646501E+05	.173420E+02
53	.105592E+01	.544121E-01	.183783E+02	.785000E-01	.679059E+05	.176083E+02
54	.105513E+01	.536668E-01	.186335E+02	.774248E-01	.712615E+05	.178736E+02
55	.105437E+01	.529442E-01	.188878E+02	.763823E-01	.747203E+05	.181381E+02
56	.105363E+01	.522431E-01	.191413E+02	.753708E-01	.782824E+05	.184016E+02
57	.105292E+01	.515626E-01	.193939E+02	.743891E-01	.819492E+05	.186642E+02
58	.105222E+01	.509017E-01	.196457E+02	.734357E-01	.857218E+05	.189260E+02
59	.105154E+01	.502596E-01	.198967E+02	.725093E-01	.896014E+05	.191870E+02
60	.105089E+01	.496355E-01	.201469E+02	.716089E-01	.935884E+05	.194471E+02

61	.105025E+01	.490285E-01	.203963E+02	.707332E-01	.976859E+05	.197065E+02
62	.104963E+01	.484390E-01	.206450E+02	.698812E-01	.101894E+06	.199650E+02
63	.104903E+01	.478632E-01	.208929E+02	.690520E-01	.106214E+06	.202228E+02
64	.104844E+01	.473035E-01	.211401E+02	.682446E-01	.110647E+06	.204798E+02
65	.104787E+01	.467583E-01	.213866E+02	.674580E-01	.115195E+06	.207360E+02
66	.104731E+01	.462271E-01	.216323E+02	.666916E-01	.119857E+06	.209916E+02
67	.104677E+01	.457092E-01	.218774E+02	.659444E-01	.124636E+06	.212464E+02
68	.104624E+01	.452041E-01	.221219E+02	.652158E-01	.129534E+06	.215006E+02
69	.104573E+01	.447114E-01	.223656E+02	.645050E-01	.134550E+06	.217540E+02
70	.104522E+01	.442306E-01	.226088E+02	.638113E-01	.139687E+06	.220068E+02
71	.104473E+01	.437612E-01	.228513E+02	.631341E-01	.144945E+06	.222589E+02
72	.104425E+01	.433029E-01	.230931E+02	.624729E-01	.150324E+06	.225104E+02
73	.104379E+01	.428551E-01	.233344E+02	.618269E-01	.155828E+06	.227613E+02
74	.104333E+01	.424177E-01	.235751E+02	.611956E-01	.161457E+06	.230115E+02
75	.104288E+01	.419901E-01	.238152E+02	.605783E-01	.167211E+06	.232611E+02
76	.104245E+01	.415720E-01	.240547E+02	.599747E-01	.173093E+06	.235101E+02
77	.104202E+01	.411631E-01	.242936E+02	.593838E-01	.179102E+06	.237585E+02
78	.104161E+01	.407632E-01	.245320E+02	.588088E-01	.185241E+06	.240064E+02
79	.104120E+01	.403718E-01	.247698E+02	.582442E-01	.191510E+06	.242536E+02
80	.104080E+01	.399887E-01	.250071E+02	.576915E-01	.197911E+06	.245003E+02
81	.104041E+01	.396137E-01	.252438E+02	.571505E-01	.204445E+06	.247465E+02
82	.104003E+01	.392464E-01	.254800E+02	.566206E-01	.211112E+06	.249921E+02
83	.103965E+01	.388867E-01	.257157E+02	.561017E-01	.217914E+06	.252372E+02
84	.103929E+01	.385343E-01	.259509E+02	.555932E-01	.224851E+06	.254817E+02
85	.103893E+01	.381889E-01	.261856E+02	.550949E-01	.231926E+06	.257257E+02
86	.103858E+01	.378504E-01	.264198E+02	.546065E-01	.239140E+06	.259692E+02
87	.103823E+01	.375184E-01	.266536E+02	.541277E-01	.246492E+06	.262123E+02
88	.103789E+01	.371930E-01	.268868E+02	.536581E-01	.253985E+06	.264548E+02
89	.103756E+01	.368737E-01	.271196E+02	.531976E-01	.261619E+06	.266968E+02
90	.103724E+01	.365606E-01	.273519E+02	.527457E-01	.269395E+06	.269383E+02
91	.103692E+01	.362533E-01	.275837E+02	.523024E-01	.277315E+06	.271794E+02
92	.103661E+01	.359516E-01	.278151E+02	.518673E-01	.285380E+06	.274200E+02

93	.103630E+01	.356556E-01	.280461E+02	.514401E-01	.293590E+06	.276601E+02
94	.103600E+01	.353649E-01	.282766E+02	.510208E-01	.301947E+06	.278998E+02
95	.103570E+01	.350795E-01	.285067E+02	.506090E-01	.310452E+06	.281390E+02
96	.103541E+01	.347991E-01	.287364E+02	.502045E-01	.319105E+06	.283778E+02
97	.103513E+01	.345237E-01	.289656E+02	.498072E-01	.327909E+06	.286161E+02

REFERÊNCIAS BIBLIOGRÁFICAS

[1] BUNCH, J.R.; PARLETT, B.N.: *Direct Methods for Solving Symmetric Indefinite Systems of Linear Equations*, SIAM Journal on Numerical Analysis, Vol. 8, nº 4.

[2] DENNIS JR., J.E.; MORÉ, J.J.: *Quasi-Newton Methods, Motivation and Theory*, SIAM Review, Vol. 19, nº 1 (1977).

[3] GILL, P.E.; GOLUB, G.H.; MURRAY, W.; SAUNDERS: *Methods for Modifying Matriz Factorization*, Math. Comp., nº 28, pg. 505-535 (1974).

[4] GILL, P.E.; MURRAY, W.: *Minimization Subject to Bounds on the Variables*, National Phisical Laboratory, Division of Num. Analysis and Computing, Dept. of Industry, 1976.

[5] GILL, P.E.; MURRAY, W.: *Newton Type Methods for Unconstrained and Linearly-Constrained Optimization*, Math. Programming, Vol. 7, nº 4 (1974).

[6] LUENBERGER, D.G.: *Introduction to Linear and Nonlinear Programming*, parte 2, Addilson Wesley, 1973.

[7] MORÉ, J.J.; SORENSEN, D.C.: *On the Use of Directions of Negative Curvature in a Modified Newton Method*, Argonne National Laboratory, Applied Math. Division, 1977.

[8] ORTEGA, J.M.; RHEINBOLDT, W.C.: *Iterative Solution of Nonlinear Equations in Several Variables*, New York - London, Academic Press, 1970.

[9] POWELL, M.J.D.: *Convergence Properties of a Class of Minimization Algorithms*, Nonlinear Programming, nº 2 (Proc. Special Interest Group on Math. Progr. Sympos., Univ. - Wisconsin, Madison, Wis., 1974), pg. 1-27, Academic Press, N.Y., 1974.

[10] POWELL, M.J.D.: *On the Convergence of the Variable Metric Algorithm*, J. Inst. Math. Applied, nº 7, pg. 21-36, 1971.

[11] POWELL, M.J.D.: *Some Global Convergence Properties of a Variable Metric Algorith for Minimization without exact Line Searches*, Nonlinear Programming, SIAM-AMS, Proceedings, Vol. 9, An. Math. Society, Providence, R.I. (1976).

[12] PSHENICHNY, B.N.; DANILIN, YU. M.: *Numerical Methods in Extremal Problems*, Mir Publishers, Moscow, 1978.

[13] SCHULLER, G.: *On the Order of Convergence of Certain Quasi-Newton Methods*, Num. Math., nº 23, 1974, pg. 181-192.

[14] SHANNO, D.F.: *On the Convergence of a New Conjugate Gradient Methods*, SIAM Journal Num. Analysis, 1978.

[15] SHANNO, D.F.: *Quadratic Termination of Conjugate Gradient Methods*, Proceedings Conference of Extremal Math., por aparecer.

[16] SHANNO, D.F.; PHUA, K.H.: *A Variable Method Subroutine for Unconstrained Nonlinear Minimization*, MIS Thechnical Report, nº 28, Management Information Systems, Univ. of Arizona, dec. 78.

[17] SHANNO, D.F.; PHUA, K.H.: *Algorithm 500. Minimization of Multivariate Functions*, TOMS, nº 2, pg. 87-94, (1976).

[18] SHANNO, D.F.; PHUA, K.H. E OUTROS:*Conjugate Gradient Methods with Inexact Searches*, Math. of Oper. Research, nº 3 , pg. 244-256 (1978).

[19] SHANNO, D.F.; PHUA, K.H.: *Matrix Conditioning and Nonlinear Otimization*, Math. Programming, nº 14, pg. 149-160 (1978).