Minimização de funções quadráticas com álgebra linear adaptativa e aplicações

Tese de doutorado em matemática aplicada

Francisco de Assis Magalhães Gomes NETO
José Mario Martínez - orientador

março de 1995

Ficha catalográfica elaborada pela
BIBLIOTECA DO IMECC DA UNICAMP

Gomes Neto, Francisco de Assis Magalhães

G585m Minimização de funções quadráticas com álgebra linear adaptativa e aplicações / Francisco de Assis Magalhães Gomes Neto. -- Campinas, [SP : s.n.], 1995.

Orientador : José Mario Martínez.

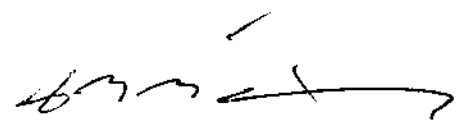
Tese (doutorado) - Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Ciência da Computação.

1. Otimização matemática. 2. Programação não linear 3. Algoritmos. I. Martínez, José Mario. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Ciência da Computação. III. Título.

Minimização de funções quadráticas com álgebra linear adaptativa e aplicações

Este exemplar corresponde à redação final da tese devidamente corrigida e defendida por Francisco de Assis Magalhães Gomes Neto e aprovada pela comissão julgadora.

Campinas, 31 de março de 1995,



Prof. Dr. José Mario Martínez
orientador

Dissertação apresentada ao Instituto de Matemática, Estatística e Ciência da Computação da Universidade de Campinas, como requisito parcial para a obtenção do título de doutor em matemática aplicada.

Tese defendida e aprovada em, 31 de Março de 1995

Pela Banca Examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). JOSÉ MÁRIO MARTÍNEZ PEREZ

Prof(a). Dr(a). MARIA CRISTINA MACIEL

Prof(a). Dr(a). MARCOS NEREU ARENALES

Prof(a). Dr(a). ANA FRIEDLANDER

Prof(a). Dr(a). RICARDO DAHAB

Prólogo

A programação quadrática não é propriamente um tema muito novo dentro da otimização. De fato, até em virtude de sua aparente simplicidade, temos inicialmente a impressão de que pouco ainda resta por ser dito sobre este assunto.

Nosso objetivo aqui é mostrar, todavia, que, pelo contrário, além de possuir um número incontável de aplicações ainda não exploradas, a programação quadrática pode ser usada com sucesso na construção de métodos competitivos destinados a resolver problemas de grande porte n ao lineares com restrições.

Para tanto, usaremos como premissa a idéia, quase consensual nos dias de hoje, de que para gerar algoritmos eficientes é preciso, em primeiro lugar, vencer o caráter combinatório associado à busca das restrições que estão ativas na solução ótima do problema, e, como um segundo objetivo, evitar que muito esforço seja dispendido à toa quando ainda estamos longe desta solução.

Com base nesses conceitos, no capítulo primeiro, apresentamos um novo algoritmo de minimização de funções quadráticas sujeitas a canalizações, que permite o uso de diferentes estratégias para a determinação de uma direção de descida nas faces do politopo definido pelas restrições. Analisamos também seus aspectos teóricos e de implementação computacional e avaliamos seu desempenho com base em algumas aplicações práticas interessantes.

O segundo capítulo é dedicado a análise de um método baseado na programação quadrática seqüencial, destinado à resolução de problemas não lineares com restrições, para o qual apresentamos os primeiros resultados numéricos.

Por fim, investigamos, em um último capítulo, a possibilidade de unir duas vertentes da programação não linear, a programação quadrática seqüencial e os métodos que

Sumário

usam o lagrangiano aumentado, com o objetivo de apresentar um algoritmo híbrido que visa, também, a solução de problemas com restrições.

CAPÍTULO 1

Minimização de quadráticas com canalizações

Reunimos sob o nome de *programação quadrática* o conjunto de problemas de otimização cujo objetivo é minimizar uma função quadrática satisfazendo restrições lineares. Dentre os problemas assim descritos, iremos nos dedicar neste capítulo apenas à resolução do mais elementar deles, ou seja, aquele no qual as únicas restrições são canalizações das variáveis. Mas tal simplicidade não diminui seu interesse ou importância, e tampouco implica ser algo absolutamente trivial a obtenção de sua solução.

De fato, para ressaltar a relevância da minimização de quadráticas em caixas (isto é, com canalizações) poderíamos nos deter em enumerar suas aplicações, que vão da mecânica dos sólidos (veja [28], [31]) à reconstrução de imagens (veja [17]), e que podem possuir de poucas a milhares de variáveis.

Mais do que isso, a utilidade dos algoritmos para este tipo de problema simples não se resume às suas aplicações diretas, pois, como veremos nos próximos capítulos, eles também podem ser empregados com sucesso na resolução de problemas mais complexos de programação não linear.

1.1 Descrição do problema

O *problema de minimização de uma função quadrática convexa com restrições de canalização* consiste em encontrar o vetor x que resolve

$$\begin{aligned} &\textit{minimizar} && q(x) = \tfrac{1}{2}x^T H x + b^T x \\ &\textit{sujeita a} && l \leq x \leq u \end{aligned} \tag{1.1}$$

onde o termo $\leq$ é usado componente a componente, a matriz H é semidefinida positiva, $x, l, u \in \mathbb{R}^n$ e $l < u$.

Dizemos que um ponto $\tilde{x}$ é *factível* se $l \leq \tilde{x} \leq u$. Se um ponto factível $\tilde{x}$ tem uma de suas componentes no limite, a canalização correspondente está *ativa* em $\tilde{x}$. Assim, por exemplo, se $\tilde{x}_i = l_i$ (ou $\tilde{x}_i = u_i$), chamamos de ativa em $\tilde{x}$ a restrição $x_i \geq l_i$ (ou $x_i \leq u_i$). Por outro lado, uma variável $\tilde{x}_i$ que não está em nenhum de seus limites (ou seja, $l_i < \tilde{x}_i < u_i$) é dita *livre*.

Tanto podemos considerar o problema 1.1 como um caso particular da programação quadrática no qual as restrições lineares foram reduzidas às canalizações das variáveis, como imaginar tratar-se de um problema de otimização em caixas que tem como objetivo minimizar uma função quadrática.

Entretanto, qualquer que seja a abordagem, devemos aproveitar algumas de suas peculiaridades para facilitar a resolução do problema. Por exemplo, o fato das restrições serem apenas canalizações de variáveis não só garante a existência de uma solução para 1.1, como evita a possibilidade de existir degeneração primal (pois nenhuma variável pode estar ao mesmo tempo em seus dois limites). Temos, assim, uma vantagem com relação à programação quadrática geral, que inclui tanto problemas infactíveis quanto degenerados.

Além disso, neste caso é absolutamente trivial a escolha de uma solução factível inicial (outra dificuldade da programação quadrática).

Também a convexidade da função objetivo deve ser considerada, pois implica que, encontrado um ponto estacionário para o problema, este será um minimizador global (vide a seção 1.2.2).

Outro detalhe importante, o fato de ser limitado o número de faces do politopo definido pelas restrições (ainda que este número cresça exponencialmente à medida em que aumentamos a quantidade de variáveis), permite-nos garantir, sob certas condições, ser possível identificar, em tempo finito, a face ótima do problema. Com isso, a determinação da solução passa a depender apenas do método usado para resolver um sistema linear restrito à face ótima, o que, como se sabe, consome uma quantidade de operações proporcional ao cubo do número de variáveis livres nesta face. Por conseqüência, podemos assegurar a existência de algoritmos capazes de resolver o problema 1.1 em um número finito de iterações.

Finalmente, cabe também salientar que, em virtude do tipo de restrição e de função objetivo que temos, a tarefa de encontrar uma direção de descida no subespaço definido pelas restrições ativas e a estimação dos multiplicadores de Lagrange são enormemente facilitadas.

Os algoritmos tradicionais para este tipo de problema são divididos de acordo com a estratégia adotada para trabalhar com as canalizações em:

- algoritmos que usam o *método das restrições ativas* (veja [22], [16]), nos quais a cada iteração k definimos o conjunto I de canalizações ativas no ponto x_k e tentamos encontrar uma direção de descida d_k interna à face associada a I (isto é, sem alterar o valor das variáveis que estão nos limites). Para tanto, ignoramos a existência das outras restrições. Em seguida, determinamos λ tal que $\bar{x} = x_k + \lambda d_k$ minimiza $q(x)$ na direção definida por d_k mantendo a factibilidade com relação às canalizações que não estavam ativas em x_k. Se uma restrição passou a ser ativa em $\bar{x}$, é preciso incluí-la em I, depois do que podemos passar a iteração seguinte. Por outro lado, no caso de termos encontrado o ponto de mínimo na face associada a I (ou seja, se $d_k = 0$) calculamos os multiplicadores de Lagrange correspondentes as restrições ativas e verificamos se seus sinais estão de acordo com as condições de otimalidade que iremos estabelecer na seção 1.2.2. No caso destes sinais estarem corretos, paramos o algoritmo (pois já obtivemos a solução do problema), senão escolhemos uma restrição ativa e a excluimos de I.

- algoritmos baseados no *método do gradiente projetado* (veja [4], [12], [31]) que tentam corrigir as inconveniências que se manifestam quando se usa a estratégia das restrições ativas para resolver problema grandes, quais sejam a lentidão com que se adiciona ou exclui restrições do conjunto I citado acima e a necessidade de minimizar a função objetivo dentro de uma face antes de tornar livre uma variável. Para tanto, os algoritmos mais modernos desta classe (como o descrito em [31]) empregam o método dos gradientes conjugados para encontrar direções de descida dentro de uma face e o método do gradiente projetado para alterar várias componentes de I antes mesmo do ponto de mínimo na face ter sido encontrado.

Recentemente, Friedlander e Martínez [17] propuseram um método que combina a estratégia das restrições ativas e o gradiente projetado. Nesta terceira linha de trabalho, uma face é explorada com o auxílio do método dos gradientes conjugados, mas sem que seja necessário encontrar nela o ponto de mínimo da função quadrática. A face é abandonada numa iteração k quando é possivel garantir que saindo dela conseguiremos reduzir a função objetivo a um valor menor do que poderíamos obter nos pontos contidos numa bola de determinado raio centrada no ponto x_k. Neste caso, usa-se a direção definida pelo gradiente projetado ou pelo gradiente "chopado" (cuja descrição será feita na próxima seção).

Foram os bons resultados apresentados por este último método que encorajaram a elaboração do algoritmo que passamos a descrever.

1.2 O algoritmo QuaCon

Reunindo resultados teóricos obtidos por Friedlander e Martínez [17] e novas propostas para a minimização dentro de uma face, com base em diferentes alternativas descritas por Friedlander, Martínez e Raydan em [20] e [19], elaboramos um algoritmo, que passou a ser denominado QuaCon, cujas características principais são;

- o uso do gradiente "chopado" nas mudanças de face;
- o abandono de uma face F com base na relação entre a norma da componente do vetor gradiente que é perpendicular a F e a componente tangente a F; e
- a utilização de diferentes métodos para minimização irrestrita de acordo com a dimensão da face na qual estamos trabalhando, incluindo a decomposição de Cholesky, o método dos gradientes conjugados e métodos de gradiente com retardo (apresentados em [19]).

1.2.1 Definições e notação

Por comodidade, e também por uma questão organizacional, vamos resumir abaixo as principais definições necessárias à descrição do algoritmo QuaCon, juntamente com alguns detalhes referentes à notação adotada no texto.

- A região factível do problema, Ω, é dada por:

$$\Omega = \{x \in I\!R^n \mid l \leq x \leq u\}. \tag{1.2}$$

- O vetor gradiente calculado em x é

$$g(x) \equiv \nabla q(x) \equiv Hx + b. \tag{1.3}$$

- Para representar corretamente as canalizações inferiores e superiores ativas na face em que estamos trabalhando, definimos I_l e I_u, subconjuntos de $\{1, 2, \ldots, n\}$ tais que $I_l \cap I_u = \emptyset$ (vide a descrição de F_I feita abaixo).

- O conjunto que identifica todas as restrições ativas é $I = I_l \cup I_u$.

- Descrevemos $F_I \subset \Omega$, a face aberta de Ω associada aos conjuntos I_l e I_u, por

$$F_I = \{x \in \Omega \mid x_i = l_i \text{ se } i \in I_l,\ x_i = u_i \text{ se } i \in I_u,\ l_i < x_i < u_i \text{ caso contrário }\}. \tag{1.4}$$

- Desta forma, a variedade afim correspondente a F_I é dada por

$$[F_I] = \{w \in I\!R^n \mid w_i = l_i \text{ se } i \in I_l,\ w_i = u_i \text{ se } i \in I_u\}. \tag{1.5}$$

- Definimos, também, $S(F_I)$, o subespaço paralelo a $[F_I]$, como

$$S(F_I) = \{w \in I\!R^n \mid w_i = 0 \text{ se } i \in I_l \cup I_u\}. \tag{1.6}$$

- O fecho de F_I é, então, $\bar{F}_I = \Omega \cap [F_I]$.

- A dimensão de $S(F_I)$ é
$$\dim F_I = n - |I|. \tag{1.7}$$

- Para cada ponto $w \in \Omega$, o *gradiente projetado* em w, $g_P(w)$, é um vetor tal que
$$g_P(w)_i = \begin{cases} 0, & \text{se } w_i = l_i \text{ e } g(w)_i > 0; \\ 0, & \text{se } w_i = u_i \text{ e } g(w)_i < 0; \\ g(w)_i, & \text{caso contrário .} \end{cases} \tag{1.8}$$

- Para um ponto $x \in F_I$ podemos dividir $g_P(x)$ em duas componentes, o *gradiente interno*, $g_I(x)$, e o *gradiente "chopado"*, $g_C(x)$, de modo que $g_P(x) = g_I(x) + g_C(x)$;

- O gradiente chopado, introduzido em [18], é definido por
$$g_C(x)_i = \begin{cases} 0, & \text{se } i \notin I_l \cup I_u; \\ g_P(x)_i, & \text{caso contrário .} \end{cases} \tag{1.9}$$

- Já o gradiente interno é dado por
$$g_I(x)_i = \begin{cases} 0, & \text{se } i \in I_l \cup I_u; \\ g_P(x)_i, & \text{caso contrário .} \end{cases} \tag{1.10}$$

1.2.2 Condições de otimalidade

As condições necessárias e suficientes para que um ponto x seja mínimo global de 1.1 são dadas por:

$$\begin{aligned} g(x) - \lambda_l - \lambda_u &= 0, \\ \lambda_l^T (x - l) &= 0, \\ \lambda_u^T (x - u) &= 0, \\ l \leq x &\leq u, \\ \lambda_l \geq 0,\ \lambda_u &\leq 0, \end{aligned} \tag{1.11}$$

onde λ_l e λ_u são vetores de multiplicadores de Lagrange associados às canalizações inferiores e superiores respectivamente. Estas condições podem ser confortavelmente resumidas na expressão:
$$g_P(x) = 0. \tag{1.12}$$

O lema abaixo, retirado de [17], mostra que se $\tilde{x}$ é um ponto estacionário do problema restrito a $\bar{F}_I$ podemos concluir que $\tilde{x}$ é um minimizador global de 1.1 ou que $g_C(\tilde{x})$ não é nulo. Neste último caso, $g_C(\tilde{x})$ fornece uma estimativa dos multiplicadores de Lagrange que não satisfazem 1.11 por estarem com o sinal trocado.

Lema 1.1 *Se $\bar{x} \in \bar{F}_I$ é tal que $q(\bar{x}) \leq q(x)$ para todo $x \in \bar{F}_I$, então é equivalente dizer que $q(\bar{x}) \leq q(x)$ para todo $x \in \Omega$, ou que $g_C(\bar{x}) = 0$.*

Demonstração. Veja [17]. □

1.2.3 Descrição do algoritmo

Para definir QuaCon admitimos a existência de um ponto inicial $x_0 \in \Omega$ e dos parâmetros $\theta \in (0,1)$ e $\delta > 0$ (além de outros parâmetros utilizados nos critérios de convergência, que serão enumerados posteriormente). Também iremos assumir que, em cada iteração k, o ponto x_k pertence a uma face F_I^k de Ω, de acordo com (1.4), e que a variável lógica *MudaFace* indica se decidimos ou não abandonar uma face.

Algoritmo 1.1 QuaCon

1. ENQUANTO não é obedecido algum critério de parada REPETIR
1.1. $MudaFace \leftarrow falso$;
1.2. SE $||g_C(x_k)||_2 > \theta ||g_P(x_k)||_2$,
1.2.1. $\bar{x} \leftarrow \arg\min\{q(y) \mid y = x_k - \lambda g_C(x_k),\ y \in \Omega\}$;
1.2.2. SE $q(\bar{x}) < q(x_k) - \delta ||g_I(x_k)||_2$,
1.2.2.1. $x_{k+1} \leftarrow \bar{x}$;
1.2.2.2. $MudaFace \leftarrow verdadeiro$;
1.3. SE (NÃO *MudaFace*),
1.3.1. Encontrar $\bar{x} \in [F_I^k]$ tal que $q(\bar{x}) < q(x_k)$;
1.3.2. SE $\bar{x} \in \bar{F}_I^k$,
1.3.2.1. $x_{k+1} \leftarrow \bar{x}$;
1.3.3. CASO CONTRÁRIO
1.3.3.1. Encontrar $x_{k+1} \in \bar{F}_I^k - F_I^k$ tal que $q(x_{k+1}) < q(x_k)$;
1.4. Atualizar o conjunto I para definir a nova face F_I^k.
1.5. $k \leftarrow k + 1$;

Em cada iteração do algoritmo 1.1 executamos, antes de mais nada, um teste (apresentado no item 1.2) que evita a minimização exata da função na face, induzindo um passo na direção do gradiente "chopado". Entretanto, este passo (1.2.1) pode ser recusado se não provocar um suficiente decréscimo da função objetivo (conforme 1.2.2). A inclusão deste último critério foi feita apenas para garantir a convergência do algoritmo quando há degeneração dual. Nos casos em que a degeneração não é problema usamos $\delta = 0$.

Permanecendo na mesma face, escolhemos uma das diferentes alternativas de passo disponíveis para o problema de minimização irrestrita em $[F_I]$ (item 1.3.1), as quais detalharemos em uma próxima seção. No caso de encontrarmos um ponto fora da região

factível, fazemos uma busca projetada para encontrar um novo ponto na fronteira de F_l (1.3.3.1).

Não é difícil garantir que Quacon está bem definido, bastando para isso que nos detenhamos um pouco mais nos itens 1.2.1, 1.3.1 e 1.3.3.1 do algoritmo. Em primeiro lugar, no caso de $g_C(x_k)$ ser suficientemente grande, recorremos ao lema 3.1 de [17] (que iremos nos abster de repetir aqui) no qual Friedlander e Martínez mostram ser possível executar um passo nesta direção. Por sua vez, a redução do valor da função quadrática quando nos movemos dentro da face (item 1.3.1) sempre é conseguida em virtude de serem convergentes os métodos de minimização irrestrita utilizados. Finalmente, a convexidade de $q(x)$ implica que a busca projetada irá gerar um ponto x_{k+1} tal que $q(x_{k+1}) < q(x)$, como veremos na seção 1.2.5.

1.2.4 Critérios de parada

Os critérios de parada adotados pelo algoritmo são três:

- *Critério absoluto de convergência.* Paramos quando a norma do gradiente projetado é muito pequena:

$$||g_P(x_k)||_2 \leq \epsilon_a.$$

- *Critério relativo de convergência.* Paramos quando o gradiente projetado é pequeno se comparado ao valor obtido na primeira iteração:

$$||g_P(x_k)||_2 \leq \epsilon_r ||g_P(x_0)||_2.$$

- *Critério de progresso insuficiente.* QuaCon também é interrompido se o progresso obtido numa iteração, medido em função do valor de $q(x)$, é relativamente pequeno:

$$q(x_{k-1}) - q(x_k) < \epsilon_p \max_{i=1,k-1} \{q(x_{i-1}) - q(x_i)\}.$$

Os valores de $\epsilon_a \geq 0$, $\epsilon_r \in [0, 1)$ e $\epsilon_p \in [0, 1)$ devem ser fornecidos como dados de entrada do programa.

1.2.5 A busca projetada

A busca projetada ao longo da direção d_k, partindo de x_k, sugerida no item 1.3.3.1 do algoritmo, é feita nos mesmos moldes de [21] e [31]. Definida a projeção de um ponto sobre a caixa como o vetor $P[x]$ cuja i-ésima componente é dada por

$$P[x]_i = \max\{l_i, \min\{x_i, u_i\}\} \tag{1.13}$$

queremos encontrar α tal que $\phi(\alpha) = q(P[x_k + \alpha d_k]) < q(x_k)$[1].

A função $\psi(\alpha) = P[x_k + \alpha d_k]$, embora contínua, é linear por partes. As mudanças de inclinação da função ocorrem sempre que alguma componente de x atinge um de seus limites. Os m pontos de mudança de direção correspondem a valores específicos de α que podem ser enumerados em ordem crescente

$$0 < \bar{\alpha}_1 < \bar{\alpha}_2 < \cdots < \bar{\alpha}_m$$

de modo que $\psi(\alpha)$ é linear no intervalo $[\bar{\alpha}_j, \bar{\alpha}_{j+1}]$.

Os valores de $\bar{\alpha}_1$ e $\bar{\alpha}_m$ são facilmente determinados por

$$\bar{\alpha}_1 = \max\{\alpha \geq 0 \mid x_k + \alpha d_k \in \Omega\}; \tag{1.14}$$

$$\bar{\alpha}_m = \min\{\alpha \geq 0 \mid (x_k)_i + \alpha(d_k)_i = l_i \text{ ou } (x_k)_i + \alpha(d_k)_i = u_i,\ i = 1, \ldots, n\} \tag{1.15}$$

Inicialmente, atribuimos a α o valor que minimiza $\phi(\alpha)$ supondo que o problema é irrestrito e envolve apenas as variáveis livres:

$$\alpha_0 = -\frac{\phi'(0)}{\phi''(0)} = -\frac{d_k^T(Hx_k + b)}{d_k^T H d_k}. \tag{1.16}$$

Obviamente, se este valor de α_0 é menor ou igual a $\bar{\alpha}_1$ não há problema em aceitar $x_{k+1} = x_k + \alpha_0 d_k$. Pensemos, então, no caso em que $\alpha_0 > \bar{\alpha}_1$, para o qual desenvolveu-se o algoritmo 1.2.

Algoritmo 1.2 Busca projetada

1. $\alpha \leftarrow \min\{\alpha_0, \bar{\alpha}_m\}$
2. ENQUANTO $(q(P[x_k + \alpha d_k]) > q(x_k))$ E $\alpha > \bar{\alpha}_1$ REPETIR
2.1. $\alpha_+ \leftarrow -\frac{\phi'(0)\alpha^2}{2(\phi(\alpha)-\phi(0)-\phi'(0)\alpha)}$;
2.2. SE $0.1\alpha \leq \alpha_+ \leq 0.9\alpha$,
2.2.1. $\alpha \leftarrow \alpha_+$;
2.3. SENÃO
2.3.1. $\alpha \leftarrow \alpha/2$;
3. SE $\alpha < \bar{\alpha}_1$,
3.1. $\alpha \leftarrow \bar{\alpha}_1$.

O critério de parada da busca projetada exige apenas que consigamos um ponto capaz de diminuir o valor da função objetivo com relação a x_k. Se este ponto ainda não foi

[1] Embora usemos subíndices tanto para representar as componentes de um vetor como a iteração na qual estamos trabalhando, não parece haver nisto motivo para confusão entre os dois significados. Mas para ter certeza de que a notação é clara, cabe citar que o índice k nesta fórmula representa a iteração.

encontrado, calculamos α_+ o minimizador da parábola que interpola $\phi(0) = q(x_k)$, $\phi'(0) = g(x_k)^T d_k$ e $\phi(\alpha) = q(P[x_k + \alpha d_k])$ (veja 2.1). O novo valor para α só é aceito se $\alpha_+ \in [0.1\alpha, 0.9\alpha]$. Caso contrário reduzimos α pela metade.

O ciclo também pára se α é menor que $\bar{\alpha}_1$, já que sabemos ser este valor aceitável. Para comprovar isso, basta observar que não há mesmo, nesse caso, nenhuma projeção. Se juntamos a isso o fato do minimizador de $q(x)$ na direção d_k não estar dentro da face, e lembramos que a função quadrática $q(x)$ é convexa, podemos garantir que

$$q(x_k + \bar{\alpha} d_k) < q(x_k). \tag{1.17}$$

Este resultado é, inclusive, usado para mostrar que a busca projetada está bem definida, posto que dividimos α pela metade a cada recusa de um novo ponto, o que certamente implica que iremos satisfazer um dos dois critérios de parada do algoritmo 1.2 em um número finito de iterações.

1.3 Alternativas para o cálculo da direção

O item 1.3.1 do algoritmo consiste em encontrar $\bar{x} \in [F_I^k]$ tal que $q(\bar{x}) < q(x_k)$. Uma vez que as restrições do problema são de canalização, podemos usar para isso algum método de minimização irrestrita de funções quadráticas convexas. As alternativas disponíveis em QuaCon incluem:

1. O uso da *decomposição de Cholesky*. A forma mais óbvia de encontrar uma direção de descida em $[F_I]$ consiste em determinar o minimizador da quadrática nesta variedade afim através da resolução por um método direto do sistema linear reduzido.

$$Z^T H Z \bar{x} = Z^T (b - H x_I) \tag{1.18}$$

onde, naturalmente, H e b foram extraídos do problema 1.1, $\bar{x}$ contém apenas $dim F_I$ elementos, x_I é um vetor tal que

$$(x_I)_i = \begin{cases} l_i & \text{se } (x_k)_i = l_i \\ u_i & \text{se } (x_k)_i = u_i \\ 0 & \text{caso contrário .} \end{cases} \tag{1.19}$$

e Z é uma matriz de dimensão $n \times \dim F_I$, obtida a partir da matriz identidade pela exclusão de suas colunas que correspondem às variáveis fixas na face F_I.

Uma vez que a matriz H é simétrica, optamos por decompô-la na forma LDL^T usando uma fatoração de Cholesky que leva em conta a esparsidade de $Z^T H Z$. Postergaremos para a seção 1.5 o detalhamento da implementação computacional deste método, juntamente com o procedimento adotado quando a matriz H é quase singular.

2. O *método dos gradientes conjugados.* Por convergir, ao menos teoricamente, em um número finito de passos, este é o método mais lembrado quando se trata de resolver de forma iterativa o sistema linear associado a uma face F_I. Neste caso, se estamos realizando a primeira iteração em F_I usamos

$$d_k = g_I(x_k), \tag{1.20}$$

e nas demais iterações

$$d_k = g_I(x_k) + \beta_k d_{k-1}, \tag{1.21}$$

onde

$$\beta_k = \frac{||g_I(x_k)||_2^2}{||g_I(x_{k-1})||_2^2}. \tag{1.22}$$

seguindo a descrição de Golub e Van Loan [23] para o clássico algoritmo de Hestenes e Stiefel [26].

3. *Métodos de gradiente com retardo.* Introduzida por Friedlander, Martínez e Raydan em [19], esta classe de métodos de minimização de quadráticas é composta de generalizações do método de máxima descida, do qual herdam a direção escolhida em cada iteração, diferindo tão somente na escolha do comprimento do passo.

Para melhor ilustrar este tipo de método, vamos fixar a notação usada tomando como base uma iteração — digamos a de índice k — do algoritmo de máxima descida, que pode ser descrita como:

$$x_{k+1} \leftarrow x_k + \lambda_k g_I(x_k), \tag{1.23}$$

onde

$$\lambda_k = -\frac{g_I(x_k)^T g_I(x_k)}{g_I(x_k)^T H g_I(x_k)}. \tag{1.24}$$

Barzilai e Borwein [1] propuseram uma modificação do método de máxima descida que consiste em tomar o valor de λ calculado na iteração anterior:

$$x_{k+1} \leftarrow x_k + \lambda_{k-1} g_I(x_k), \tag{1.25}$$

mantendo, é claro, o cálculo de λ_k. A escolha de λ_{-1} é arbitrária.

Friedlander, Martínez e Raydan [19] sugeriram outras variantes do método da máxima descida na forma:

$$x_{k+1} \leftarrow x_k + \lambda_{\nu(k)} g_I(x_k), \tag{1.26}$$

onde $\nu(k)$, o *retardo adotado na iteração* k, é um índice que pode ser escolhido de diferentes maneiras dentre os elementos de $\{\max(0, k-m), \ldots, k-1, k\}$. O valor de

m é conhecido como o *retardo máximo*, e é um parâmetro de entrada do algoritmo. Em virtude de não dispormos de $m+1$ valores para λ nas primeiras iterações, o índice da iteração correspondente ao retardo máximo é dado por $\bar{k} = \max\{0, k-m\}$. Como se observa, se adotamos $\nu(k) = k$ obtemos o método de máxima descida. Além desta alternativa, QuaCon ainda dispõe das seguintes formas de escolher $\nu(k)$:

(a) *retardo máximo*: $\nu(k) = \bar{k}$. O método de Barzilai e Borwein é um caso particular desta opção, e é obtido fazendo $m = 1$;

(b) *retardo aleatório*: $\nu(k)$ é um número aleatório entre $\bar{k}$ e k;

(c) *retardo cíclico*: $\nu(k) = k - (k \bmod (m+1))$;

(d) *retardo com o passo máximo*: $\nu(k) = \arg\max\{\lambda_{\bar{k}}, \ldots, \lambda_k\}$;

(e) *retardo com o passo mínimo*: $\nu(k) = \arg\min\{\lambda_{\bar{k}}, \ldots, \lambda_k\}$;

(f) *retardo mínimo-máximo*: $\nu(k) = \bar{k}$ se k é par, $\nu(k) = k$ se k é impar (de outra maneira: $\nu(k) = \max\{0, k - ((k+1) \bmod 2)m\}$);

(g) *retardo aleatório excetuando o método de máxima descida*: $\nu(k)$ é um número aleatório entre $\bar{k}$ e $k-1$;

Essas várias opções para a escolha do passo produzem algoritmos que, embora convergentes como o método de máxima descida, deste também diferem em outro ponto fundamental: a não monotonicidade. Enquanto no método do gradiente geramos a cada iteração um passo que reduz o valor da função objetivo, o mesmo não acontece obrigatoriamente quando escolhemos um comprimento de passo com retardo. Com isso, é preciso fazer algumas iterações até conseguir um ponto que forneça um decréscimo da função quadrática, como pede o item 1.3.1 de QuaCon, para então prosseguir fazendo a busca projetada. Desse e de outros pormenores voltaremos a falar mais tarde, na seção dedicada exclusivamente à implementação computacional.

A determinação do método mais indicado para a minimização de $q(x)$ em $[F_I]$ é difícil e depende, dentre outros fatores, da dimensão da face F_I, da disponibilidade de memória do computador no qual implantamos QuaCon, do padrão de esparsidade da matriz hessina H e do bom ou mal condicionamento desta matriz, devendo ser feita, portanto, com base nas particularidades de cada problema.

Em linhas gerais, como a resolução do sistema reduzido 1.18 fornece já na primeira iteração o minimizador da função quadrática em $[F_I]$, dever-se-ia optar por determinar desta forma o vetor d_k sempre que a dimensão da face e o padrão de esparsidade da matriz hessiana reduzida assim o permitissem, deixando o método dos gradientes conjugados para sistemas nos quais a matriz é um pouco maior ou mais densa, desde que bem condicionada. Os métodos com retardo serviriam para sistemas muito grandes, em virtude de serem muito econômicos no que diz respeito a memória computacional e ao tempo gasto em cada iteração.

Mas como não é possível definir de uma forma mais genérica qual alternativa QuaCon deveria empregar para a minimização dentro da face, estabelecemos uma regra

bastante elementar. Se dim F_I é menor que um valor limite fixado em função do problema que estamos resolvendo, usamos um método direto, caso contrário optamos por um único método iterativo que também deve ser arbitrado antecipadamente (de acordo com as características da matriz H).

1.4 Resultados Teóricos

Os algoritmos mais recentes para minimização de quadráticas convexas em caixas costumam apresentar resultados de convergência fortemente baseados no desempenho do método usado na minimização irrestrita dentro das faces do politopo definido pelas canalizações. Apenas para corroborar esta afirmação, lembramos ter sido a propriedade de terminação finita do método dos gradientes conjugados invocada por Moré e Toraldo [31] e Friedlander e Martínez [18] para garantir a convergência em um número finito de passos de seus próprios algoritmos.

Não seria, portanto, a QuaCon que ousaríamos dispensar tratamento outro daquele usado nestes já citados exemplos. E é exatamente por este motivo que iremos nos limitar a estudar a convergência do algoritmo, não lhe exigindo terminação finita, uma vez que esta propriedade não é observada pelos métodos de gradientes com retardo, empregados por QuaCon para a minimização nas faces (vide [19]).

Cabe ressaltar também, antes de prosseguir, que tudo aquilo que apresentaremos nesta seção nada mais é que uma reprodução dos resultados obtidos em [6].

Feita esta ressalva, comeÇemos por estabelecer um teorema que garante um decréscimo suficiente da função objetivo quando usamos o gradiente "chopado" como vetor direção.

Para tanto, não só será preciso supor a existência de um valor $L > 0$ tal que $||H||_2 \leq L$, bem como definir $\gamma = \min\{u_i - l_i \,|\, i \in I_l \cup I_U\}$ e lembrar que, dada a convexidade de $q(x)$, podemos escrever, para todo $x, z \in \mathbb{R}^n$,

$$0 \leq q(z) - q(x) - \nabla q(x)^T(z-x) = \frac{1}{2}(z-x)^T H(z-x) \leq \frac{L}{2}||z-x||_2^2. \tag{1.27}$$

Lema 1.2 *Se o valor de $\bar{x}$ é definido como no item 1.2.1 do algoritmo QuaCon , então*

$$q(x_k) - q(\bar{x}) \geq \min\{\frac{\theta\gamma}{2}||g_P(x_k)||_2, \frac{\theta}{2L}||g_P(x_k)||_2^2\}.$$

Demonstração. Uma vez atingido o item 1.2.1 do algoritmo, é possível garantir que $g_C(x_k) \neq 0$. Assim, $x_k - \lambda g_C(x_k) \in \Omega$ para todo $\lambda \in [0, \tilde{\lambda}]$, onde

$$\tilde{\lambda} = \frac{\gamma}{||g_C(x_k)||_2}. \tag{1.28}$$

Definida a função quadrática

$$q(x_k - \lambda g_C(x_k)) = q(x_k) - \lambda \nabla q(x_k)^T g_C(x_k) + \frac{\lambda^2}{2} g_C(x_k)^T H g_C(x_k), \tag{1.29}$$

o único minimizador de $q(x_k - \lambda g_C(x_k))$ no caso de $g_C(x_k)^T H g_C(x_k) \neq 0$ é dado por

$$\lambda^* = \frac{||g_C(x_k)||_2^2}{g_C(x_k)^T H g_C(x_k)}.$$

Consideremos, inicialmente, o caso em que $x_k - \lambda^* g_C(x_k)$ não é factível. Se isto ocorre, podemos escrever

$$\bar{x} = \arg\min\{q(y) \mid y = x_k - \bar{\lambda} g_C(x_k) \in \Omega\}$$

e também

$$q(x_k - \tilde{\lambda} g_C(x_k)) \geq q(x_k - \bar{\lambda} g_C(x_k)). \tag{1.30}$$

para algum $\tilde{\lambda}$ tal que $\tilde{\lambda} \leq \bar{\lambda} < \lambda^*$. Substituindo $\tilde{\lambda}$ dado por (1.28) em (1.29), obtemos

$$q(x_k) - q(x_k - \tilde{\lambda} g_C(x_k)) = \gamma ||g_C(x_k)||_2 - \frac{\gamma^2 g_C(x_k)^T H g_C(x_k)}{2||g_C(x_k)||_2^2}. \tag{1.31}$$

Usando novamente (1.28) para reescrever o último termo da equação acima e lembrando que $\lambda^* > \tilde{\lambda}$, transformamos (1.31) em

$$q(x_k) - q(x_k - \tilde{\lambda} g_C(x_k)) > \frac{\gamma}{2} ||g_C(x_k)||_2. \tag{1.32}$$

Combinando (1.32) e (1.30) com a desigualdade do passo 1.2 de QuaCon, obtemos, então,

$$q(x_k) - q(\bar{x}) > \frac{\gamma}{2} ||g_C(x_k)||_2 > \frac{\theta\gamma}{2} ||g_C(x_k)||_2. \tag{1.33}$$

Passemos agora a analisar a situação em que $x_k - \lambda^* g_C(x_k)$ é factível e é, portanto, o valor adotado para $\bar{x}$. Neste caso,

$$q(x_k) - q(\bar{x}) = \frac{(\lambda^*)^2}{2} g_C(x_k)^T H g_C(x_k) = \frac{||g_C(x_k)||_2^4}{2 g_C(x_k)^T H g_C(x_k)}. \tag{1.34}$$

E, com base nesta equação, em (1.27) e no item 1.2 de QuaCon, escrevemos diretamente

$$q(x_k) - q(\bar{x}) > \frac{1}{2L} ||g_C(x_k)||_2^2 > \frac{\theta}{2L} ||g_P(x_k)||_2^2. \tag{1.35}$$

Finalmente, no caso em que $g_C(x_k)^T H g_C(x_k) = 0$, usamos (1.31) para obter

$$q(x_k) - q(x_k - \tilde{\lambda} g_C(x_k)) = \gamma ||g_C(x_k)||_2$$

e, por ser $q(\bar{x}) < q(x_k - \tilde{\lambda} g_C(x_k))$, escrevemos, então,

$$q(x_k) - q(\bar{x}) > \gamma ||g_C(x_k)||_2 > \theta\gamma ||g_P(x_k)||_2. \ \square$$

Posto este lema, estamos prontos para demonstrar a convergência do algoritmo:

Teorema 1.1 *Se x^* é um ponto de acumulação de $\{x_k\}$, a seqüência infinita gerada por QuaCon, então x^* é um ponto estacionário do problema (1.1).*

Demonstração. A existência de pontos de acumulação para $\{x_k\}$ é assegurada pelo fato da seqüência permanecer em Ω, um conjunto fechado e limitado.

Provemos então que ao menos um destes pontos de acumulação é um ponto estacionário de (1.1). Para tanto iremos supor, por contradição, que existe $\epsilon > 0$ tal que

$$||g_P(x_k)|_2 > \epsilon \text{ para todo } k. \tag{1.36}$$

Consideremos primeiro a hipótese da condição estabelecida no passo 1.2 do algoritmo[2] ser atendida apenas em um número finito de passos. Neste caso, existe k_0 tal que para todo $k \geq k_0$ temos $x_k \in F_I$ para um conjunto I fixo. Com isso, a partir de $k = k_0$, os pontos da seqüência são gerados por um método de minimização irrestrita (passo 1.3.2.1) usado dentro de F_I. Se este método é convergente, então $g_I(x_k)$ tende a zero à medida que k vai para infinito. Sendo assim, como estamos admitindo ser verdadeira a equação (1.36), em algum passo $\bar{k} \geq k_0$ suficientemente grande teremos satisfeito a condição do item 1.2 de QuaCon, o que contradiz nossa hipótese.

Suponhamos agora que existe um conjunto infinito de índices $K_1 \subset I\!N$ tal que a condição do item 1.2 de QuaCon é satisfeita para todo $k \in K_1$. Seja k_j o j-ésimo índice em K_1. Com base no fato de $\{q(x_k)\}$ ser monotonicamente decrescente, escrevemos

$$q(x_{k_j}) - q(x_{k_1}) = \sum_{l=k_1}^{k_j-1} (q(x_{l+1}) - q(x_l)) \leq \sum_{l \in K_1, l=k_1}^{k_j-1} (q(x_{l+1}) - q(x_l))$$

Usando agora o lema 1.2, temos

$$q(x_{k_j}) - q(x_{k_1}) \leq \sum_{l \in K_1, l=k_1}^{k_j-1} - \min\{\frac{\theta\gamma}{2}||g_P(x_l)||_2, \frac{\theta}{2L}||g_P(x_l)||_2^2\}$$

e uma vez que $||g_P(x_l)||_2 > \epsilon$ para todo $l \in K_1$,

$$q(x_{k_j}) - q(x_{k_1}) < -j \ \min\{\frac{\theta\gamma}{2}\epsilon, \frac{\theta}{2L}\epsilon^2\}$$

donde concluimos que

$$\lim_{j \to \infty} q(x_{k_j}) = -\infty,$$

o que contradiz o fato de $q(x)$ ser limitada inferiormente em Ω.

Assim, ao menos um dos pontos de acumulação de $\{x_k\}$ é também um ponto estacionário de (1.1).

Chamemos este ponto de x^*. Como a seqüência $\{q(x_k)\}$ gerada por QuaCon é monótona decrescente, temos que $q(\bar{x}) = q(x^*)$ para todos os outros pontos de acumulação

[2] Ou seja, $||g_C(x_k)||_2 > \theta||g_P(x_k)||_2$, para $0 < \theta < 1$.

$\bar{x} \neq x^*$ de $\{x_k\}$. E sendo $q(x)$ uma função quadrática convexa e x^* um ponto estacionário, concluimos finalmente que $g_P(\bar{x}) = 0$. □

Uma das conseqüências imediatas do teorema acima é o

Corolário 1.1 *Se o problema (1.1) tem apenas uma solução x^*, então a seqüência $\{x_k\}$ gerada por QuaCon converge para x^*.*

Também podemos usar o teorema 1.1, juntamente com o corolário 3.6 de [7], para demostrar o seguinte resultado:

Corolário 1.2 *Se $\{x_k\}$ é uma seqüência gerada por QuaCon que converge para um ponto estacionário que não é dual degenerado x^* e se $x^* \in F_{I^*}$, então existe $\bar{k} \in \mathbb{N}$ tal que $x_k \in F_{I^*}$ para todo $k \geq \bar{k}$.*

A propriedade de identificação em um número finito de iterações da face ótima do problema de minimização no caso deste possuir uma solução ótima que não é dual degenerada, descrita no corolário acima, pode ser generalizada no

Teorema 1.2 *Se nenhum dos pontos de acumulação de uma seqüência $\{x_k\}$ gerada por QuaCon é dual degenerado, então existe $\bar{k}$ e uma face F_I tal que $x_k \in F_I$ para todo $k \geq \bar{k}$.*

Demonstração. Por contradição, admitamos que existe uma face F_J e um conjunto infinito $K_1 \subset \mathbb{N}$ tais que $x_k \in F_J$ e $x_{k+1} \notin F_J$ para $k \in K_1$. Então x_{k+1} é obtido pelo passo 1.2.2.1 de QuaCon[3] para todo $k \in K_1$, e assim uma das restrições que define F_J deixará de estar ativa em um subconjunto infinito $K_2 \subset K_1$. Sem perda de generalidade, suponhamos que esta restrição é $x_j = l_j$, de modo que $j \in I_l(x_k)$ e $j \notin I_l(x_{k+1})$ para $k \in K_2$. Assim,

$$-\frac{\partial q}{\partial x_j}(x_k) > 0. \tag{1.37}$$

Por outro lado, se x^* é um ponto de acumulação de $\{x_k\}_{k \in K_2}$, então $x_j^* = l_i$ e, usando (1.37), temos

$$-\frac{\partial q}{\partial x_j}(x^*) \geq 0.$$

Mas, pelo teorema (1.1), x^* é um ponto estacionário, de modo que $-\frac{\partial q}{\partial x_j}(x^*) = 0$, o que contradiz o fato de x^* não ser degenerado. □

Depois de apresentados tantos resultados teóricos baseados na hipótese de que o problema não seja dual degenerado, vejamos o que acontece nos casos em que este último fenômeno ocorre.

Podemos demonstrar não só que QuaCon converge mas que é capaz de identificar a face ótima de um problema do tipo (1.1) mesmo que este seja dual degenerado, desde que

[3] $x_{k+1} \leftarrow \arg\min\{q(y) \mid y = x_k - \lambda g_C(x_k),\ y \in \Omega\}$.

incluamos como condição adicional para abandonar uma face (no item 1.2.2 do algoritmo) a exigência de que o novo ponto x_{k+1}, isto é, o minimizador da função $q(x)$ na direção do gradiente "chopado", obedeça o seguinte critério de decréscimo suficiente:

$$q(x_{k+1}) < q(x_k) - \delta||g_I(x_k)||_2, \tag{1.38}$$

onde $\delta > 0$.

Este novo critério é usado no seguinte lema para mostrar que se saimos de uma face na iteração k, em nenhuma iteração posterior de QuaCon retornaremos a algum dos pontos da face inscritos numa bola com centro em x_k e raio δ.

Lema 1.3 *Sejam $x_k \in F_I$ e o ponto x_{k+1} gerado no passo 1.2.1 do algoritmo. Se x_{k+1} satisfaz a condição (1.38), então para todo $\bar{x} \in \bar{F}_I$ tal que $||\bar{x} - x_k||_2 \leq \delta$ vale a desigualdade $q(x_{k+1}) < q(\bar{x})$.*

Demonstração. De (1.27) temos, para todo $\bar{x} \in \bar{F}_I$,

$$q(\bar{x}) \geq q(x_k) + g_I(x_k)^T(\bar{x} - x_k).$$

Usando, então, a desigualdade de Cauchy-Schwarz e (1.38) podemos escrever, para todo $\bar{x} \in \bar{F}_I$ tal que $||\bar{x} - x_k||_2 \leq \delta$,

$$q(\bar{x}) \geq q(x_k) + \delta||g_I(x_k)||_2 > q(x_{k+1}). \ \square$$

Terminando esta seção, usamos este último resultado para estabelecer, finalmente, a convergência global de QuaCon mesmo na presença de degeneração dual:

Teorema 1.3 *Se $\{x_k\}$ é uma seqüência gerada por QuaCon, então $\{x_k\}$ converge para x^*, uma solução global do problema (1.1). Além disso, existe uma face F_I e um índice $k_0 > 0$ para os quais não somente $x_k \in F_I$ para todo $k \geq k_0$ como também $x^* \in \bar{F}_I$. No caso de x^* não ser dual degenerado, $x^* \in F_I$.*

Demonstração. Com base no lema 1.3 e no fato de ser finito o número de faces de Ω, é possível definir k_0 e F_I tais que $x \in F_I$ para todo $k \geq k_0$. Então, a partir de k_0, a seqüência $\{x_k\}$ é gerada por um método de minimização irrestrita que converge em $\bar{F}_I$. Usando o teorema (1.1), concluimos que $\{x_k\}$ converge para um ponto estacionário $x^* \in \bar{F}_I$, ou $x^* \in F_I$ no caso de x^* não ser dual degenerado, $\square$

De forma diferente do que acabamos de apresentar, Friedlander e Martínez [17], em artigo recente, impuseram um critério para mudança de face que envolvia o uso de um limitante para a norma da matriz H como forma de garantir a convergência de seu algoritmo quando submetido à resolução de problema degenerados. Entretanto, uma vez que este tipo de limitante é difícil de calcular na maioria das aplicações reais, optou-se por incluir em QuaCon um teste mais simples que dependesse de termos conhecidos como $q(x)$ e $g_I(x)$.

1.5 Aspectos Computacionais

Depois de termos comentado a importância teórica dos métodos de minimização irrestrita utilizados por QuaCon dentro de cada face, parece oportuno lembrar que deles depende não só a convergência mas também a eficiência do algoritmo. Isto porque enquanto em cada um dos demais passos de QuaCon efetuamos algo em torno de n operações aritméticas, onde n é o número de variáveis do problema, o cálculo da direção interna à face pode vir a consumir, em cada iteração, um valor cuja ordem de grandeza está próxima a n^2 ou n^3 operações. E como até mesmo o número de iterações realizadas pelo algoritmo está intimamente relacionado ao método usado neste passo, julgamos conveniente voltar ao assunto para tornar mais claro aquilo que foi comentado de forma propositalmente superficial até agora.

Dividiremos, então, esta seção em três para podermos analisar separadamente os aspectos mais relevantes de cada método de minimização.

1.5.1 A decomposição de Cholesky

Examinemos primeiro como minimizar uma função quadrática convexa

$$q(x) = \frac{1}{2}x^T Hx + b^T x, \tag{1.39}$$

resolvendo por um método direto o sistema linear

$$Hx = -b.$$

Para simplificar a notação, consideramos aqui que a matriz simétrica $H \in \mathbb{R}^{n\times n}$ e os vetores $b \in \mathbb{R}^n$ e $x \in \mathbb{R}^n$ correspondem ao problema quadrático já reduzido à face F (de dimensão n) na qual QuaCon está trabalhando, não contendo conseqüentemente os dados das variáveis fixas em F (vide 1.18).

Se decompomos a hessiana do problema na forma

$$H = LDL^T. \tag{1.40}$$

onde L é uma matriz triangular inferior unitária e D é diagonal, o sistema passa a ser escrito como $LDL^T x = -b$ e podemos encontrar facilmente sua solução por intermédio da seguinte seqüência de passos:

$$Ly = -b;\ \ Dz = y;\ \ L^T x = z.$$

Obviamente, a resolução destes três últimos sistemas é elementar, motivo pelo qual deteremos nossa atenção na *fatoração de Cholesky* (1.40) de H.

Os elementos da j-ésima coluna de L e de D podem ser obtidos de forma simples em função das colunas de índice menor através das expressões:

$$d_j \ \leftarrow \ h_{jj} - \sum_{k=1}^{j-1} d_k l_{jk}^2 \tag{1.41}$$

$$l_{ij} \leftarrow \frac{1}{d_j}(h_{ij} - \sum_{k=1}^{j-1} d_k l_{jk} l_{ik}), \quad j = i+1, \ldots, n.$$

Mas embora a decomposição de Cholesky seja simples de calcular, seu emprego em QuaCon possui alguns grandes inconvenientes, dentre os quais salientamos

- A obrigatoriedade de efetuarmos n iterações para determinar L e D, mesmo que tenhamos que abandonar a face logo após a determinação do minimizador da quadrática. Nas faces muito distantes da solução ótima do problema, a determinação precisa do minimizador da quadrática pode não ser necessária e tampouco conveniente.

- O alto consumo de memória computacional. Dependendo da dimensão e do padrão de esparsidade da hessiana do problema, o espaço necessário para armazenar a matriz L pode ser um fator limitante para a triangularização de H. E além de não ser possível, geralmente, prever de forma inexpensiva o tamanho da estrutura de dados que deve ser usada para armazenar L, existem mesmo casos em que H é bastante esparsa mas a matriz triangular oriunda da fatoração é densa.

Por outro lado, se o sistema é suficientemente pequeno ou se, como fruto da decomposição, obtemos L muito esparsa, a adoção de um método direto para resolver (1.39) torna-se extremamente atraente.

Levando em conta todas estas observações, a implementação computacional da decomposição de Cholesky foi feita de modo que:

1. Adotamos este tipo de método sempre que a dimensão do sistema for menor que um limite, *LimChol*, passado para QuaCon como parâmetro, desde que a memória do computador assim o permita.

2. O algoritmo considera densa uma matriz se suas colunas contêm menos elementos que um valor inteiro fixo *LimDen*, ou se a densidade da coluna (ou seja, o número de elementos não nulos dividido pela dimensão da matriz) é maior que outro parâmetro *FracDen*.

3. Se L é densa, guardamos seus elementos por colunas em um vetor U. A matriz diagonal, por sua vez, é armazenada em outro vetor *Diag*.

4. Salvo se um dos critérios descritos acima indicar que L deve ser considerada densa, aplicamos uma decomposição esparsa. Com isso, guardamos em U e $ColU$, novamente por colunas, o valor e o índice da linha correspondente a cada elemento não nulo de L. Fornecemos ainda outro vetor, $IniLU$, cuja componente $IniLU(k)$ indica a posição em U e $ColU$ do primeiro elemento da coluna k de L.

5. A estrutura de dados usada para definir L pode ser híbrida. Observa-se com o decorrer da decomposição de matrizes esparsas uma tendência de aumento da densidade das

colunas de L, pela introdução de elementos não nulos em posições que originalmente continham zeros, fenômeno que conhecemos como *preenchimento.* Se, com isso, em algum momento um dos critérios descritos no item 2 for atendido, a parte ainda não decomposta da matriz passa a ser tratada como densa.

6. Não se supõe disponível a estrutura de dados do fator L no início da decomposição. Dito de outra forma, em QuaCon não é feita uma fatoração simbólica, já que a dimensão de H pode variar muito entre as iterações.

7. A parte do algoritmo que depende do problema que estamos resolvendo fica confinada na subrotina PegaColHess que deve ser criada de forma a fornecer cada coluna da hessiana original de (1.1) usando dois vetores, um contendo os elementos não nulos da coluna e o outro os índices das linhas destes elementos. Já a eliminação dos elementos correspondentes às variáveis fixas na face F é feita numa rotina interna a QuaCon.

8. É possível promover a redução do número de elementos não nulos da matriz L criados durante a fatoração através da reordenação das colunas e linhas de H, o que é feito com base em um processo heurístico bastante conhecido, denominado *método do grau mínimo* (veja [15]). Obtém-se com isso, geralmente, uma redução do tempo computacional consumido pela decomposição, além da economia de memória.

   Entretanto, uma vez que a determinação desta permutação pode ser relativamente cara e como as variáveis fixas do problema nunca são as mesmas em iterações sucessivas, é difícil definir mecanismos para decidir quando é conveniente tentar reduzir o enchimento em L. Observa-se apenas que parece razoável tentar manter a mesma ordenação das variáveis por algumas iterações de QuaCon, desde que não haja uma mudança de face tal que a dimensão do subproblema aumente.

   Embora disponível em QuaCon, a subrotina de ordenação segundo o grau mínimo não foi utilizada em nenhum dos problemas testados. Em lugar disto preferimos dedicar um tempo maior à análise das características de cada problema e estabelecer uma ordem inicial conveniente para as variáveis.

9. Admite-se que a hessiana seja apenas semidefinida positiva. Nestes casos, usamos uma fatoração de Cholesky modificada, decompondo a matriz

$$\bar{H} = H + E$$

   onde E é diagonal e seus elementos, além de serem maiores ou iguais a zero, são tão pequenos quanto o necessário para tornar $\bar{H}$ definida positiva e mininamente bem condicionada. O procedimento é semelhante àqueles apresentados por Gill, Murray e Wright [22] e Dennis e Schnabel [14], diferindo apenas levemente na definição da matriz E.

Descrevemos simplificadamente no algoritmo 1.3 a decomposição, incluindo o tratamento de matrizes semidefinidas positivas, na forma de um algoritmo. Nele, o termo

nel_i que aparece em alguns passos representa o número de elementos não nulos da coluna i de L, e $ZrFat$, um número positivo pequeno, é o menor valor admitido em D.

Algoritmo 1.3 Decomposição de Cholesky

1. $\theta \leftarrow 0$; $\mu \leftarrow 1/[\max\{1, \sqrt{n^2-1}\}]$; $\beta \leftarrow ZrFat$;
2. $LimDen \leftarrow n - LimDen + 1$;
3. PARA $k \leftarrow 1, \ldots, n$ REPETIR
3.1. $diag_i \leftarrow h_{ki}$, $i = k, \ldots, n$;
3.2. $\beta \leftarrow \max\{\beta, |diag_k|\}$;
3.3. $\beta \leftarrow \max\{\beta, \mu|diag_i|\}$, $i = k+1, \ldots, n$;
3.4. $diag_k \leftarrow diag_k + \theta$;
3.5. PARA $i \leftarrow 1, \ldots, \min\{k-1, LimDen-1\}$ REPETIR
3.5.1. SE $(l_{ki} \neq 0)$, executar como uma operação esparsa
3.5.1.1. $diag_j \leftarrow diag_j - diag_i l_{k,i} l_{k,j}$, $j = k+1, \ldots, n$;
3.6. SE $(k < LimDen)$,
3.6.1. SE $(nel_k \geq FracDen(n-k+1))$,
3.6.1.1. $LimDen \leftarrow k$;
3.7. SE $(k \geq LimDen)$, executar como uma operação densa
3.7.1. PARA $i \leftarrow LimDen, \ldots, k-1$ REPETIR
3.7.1.1. $diag_j \leftarrow diag_j - diag_i l_{k,i} l_{k,j}$, $j = k+1, \ldots, n$;
3.8. SE $(diag_k < ZrFat)$,
3.8.1. $c \leftarrow 0$;
3.8.2. $c \leftarrow \max\{c, |diag_j|\}$, $j = k+1, \ldots, n$;
3.8.3. $\theta \leftarrow \max\{c^2/\beta, |diag_k|, ZrFat\} - diag_k$;
3.8.4. $diag_k \leftarrow diag_k + \theta$;
3.9. Passar os elementos de $diag$ para a coluna k de L;

1.5.2 O método dos gradientes conjugados

A implementação computacional do método dos gradientes conjugados não constitui dificuldade e decidimos, portanto, não nos deter demasiadamente em detalhá-la. Dos aspectos que mais podem nos interessar neste método iterativo, citamos apenas que

1. O custo computacional costuma ser pequeno se comparado ao da decomposição de Cholesky, tanto porque é necessário calcular apenas um passo do método a cada iteração de QuaCon (enquanto um método direto promove uma minimização exata na face em que estamos calculando o vetor direção, o que pode não ser necessário nem conveniente se esta face é grande e não contém a solução do problema), como pelo fato da parte mais cara de cada iteração estar relacionada ao produto da hessiana do problema por um vetor, o que, como se sabe, exige apenas um número de operações

proporcional à quantidade de elementos não nulos em H (geralmente pouco se comparado ao número operações necessárias à fatoração de uma matriz, inclusive porque matrizes esparsas costumam sofrer certo preenchimento durante sua decomposição).

2. A propriedade de terminação finita que na teoria o método possui é perdida na prática em virtude dos erros de arredondamento oriundos do uso de números com precisão finita pelo computador. Como veremos na seção de resultados numéricos, particularmente se H é mal condicionada, o número de iterações efetuadas dentro da mesma face por QuaCon pode ser muito grande.

3. A forma de evitar os efeitos negativos dos erros de arredondamento citados no item anterior costuma estar baseada no precondicionamento do problema. Isto inclui a solução por um método direto de um outro sistema cuja matriz M é relativamente fácil de decompor. O *precondicionador* (termo usado para designar M) freqüentemente não passa de uma cópia incompleta de H para a qual a decomposição de Cholesky é barata.

   Mas como pretendemos resolver problemas grandes, para os quais existe não apenas uma certa dificuldade em definir M corretamente, mas também uma limitação natural de memória computacional, não adotamos nenhum tipo de precondicionamento.

4. A única salvaguarda que adotamos para a prevenção de problemas relativos à convergência do método consiste em recomeçá-lo em alguns casos. Isto significa que voltamos a usar $-g_I(x_k)$ como direção no caso de já termos efetuado n iterações sucessivas de QuaCon sem mudar de face (veja abaixo o item 2 do algoritmo) ou quando o vetor direção d_k não forma um ângulo agudo com a direção de $-g_I(x_k)$ (item 3.3).

5. A interface entre o problema que resolvemos e QuaCon se resume, neste caso, à rotina MultHess, cuja função é multiplicar a matriz hessiana por um vetor.

No algoritmo 1.4 resumimos como obter uma nova aproximação da solução do problema de minimização de uma quadrática dentro de uma face F_I, a partir de um ponto x_k, através de um passo do método dos gradientes conjugados. A variável *iter* informa quantas iterações já fizemos dentro da mesma face em que estamos.

Algoritmo 1.4 *Determinação do passo pelo método dos gradientes conjugados*

1. $iter \leftarrow 0$
2. SE $(iter = 0)$ OU $(iter = n)$,
2.1. $\quad d_k \leftarrow -g_I(x_k)/||g_I(x_k)||_\infty$;
2.2. $\quad iter \leftarrow 1$;

3. SENÃO
3.1. $\beta \leftarrow ||g_I(x_k)||_2^2 / ||g_I(x_{k-1})||_2^2$;
3.2. $d_k \leftarrow -g_I(x_k)/||g_I(x_k)||_\infty + \beta d_{k-1}||g_I(x_{k-1})||_\infty / ||g_I(x_k)||_\infty$;
3.3. SE $(-d_k^T g_I(x_k)/(||g_I(x_k)||_2 ||d_k||_2) \leq 10^{-6})$,
3.3.1. $d_k \leftarrow -g_I(x_k)/||g_I(x_k)||_\infty$;
3.3.2. $iter \leftarrow 1$;
3.4. SENÃO
3.4.1. $iter \leftarrow iter + 1$;
4. $\lambda \leftarrow -d^T g_I(x_k)/d_k^T H d$;
5. $\bar{x} \leftarrow x_k + \lambda d_k$.

1.5.3 Os métodos de gradiente com retardo

Como já foi mencionado, o método dos gradientes conjugados costuma apresentar uma taxa de convergência teórica melhor que àquela obtida por métodos de gradiente com retardo. Por outro lado, por consumirem menos memória e tempo computacional em cada iteração, estes últimos parecem ser mais indicados quando precisamos resolver problemas muito grandes, para os quais uma economia de espaço correspondente a um vetor e uma redução da ordem de n operações aritméticas passam a ter importância.

Não foi, contudo, esse uso diferenciado dos métodos o motivo que nos fez adotá-los em QuaCon. De fato, estamos interessados em comparar o desempenho destas duas alternativas, de forma que permitimos o uso de qualquer uma delas para minimizar a função quadrática numa face cuja dimensão é relativamente grande (pois nas faces pequenas empregamos a decomposição de Cholesky).

E uma vez que os métodos de gradiente com retardo foram apresentados de forma bastante satisfatória na seção 1.3, iremos nos limitar aqui a expor um passo do algoritmo. Lembramos, então, que, devido à não monotonicidade do método, só consideramos calculada uma nova aproximação $\bar{x}$ para a solução do problema quadrático se $q(\bar{x})$ é menor que o melhor valor já encontrado para $q(x)$ (vide item 3 abaixo). Até que isso ocorra, permitimos que os pontos intermediários (x_+) gerados sejam infactíveis. Finalmente, após a determinação de $\bar{x}$, a busca projetada (item 1.3.3.1 de QuaCon) é feita apenas no caso de alguma de suas componentes estar fora dos limites.

Na descrição abaixo usamos q_{min} para representar o menor valor já encontrado para $q(x)$ e $ItTot$ para indicar o número total de iterações, incluindo aquelas em que $q(x)$ aumentou.

Algoritmo **1.5** *Determinação do passo por um método de gradiente com retardo*

1. $x_+ \leftarrow x_k$;
2. REPETIR
2.1. $d_+ \leftarrow -g_I(x_+)$;

2.2. $\lambda_{ItTot} \leftarrow d_+^T d_+ / d_+^T H d_+$;
2.3. Escolher $\bar{\lambda}$ usando uma das alternativas citadas na seção 1.3;
2.4. $x_+ \leftarrow x_+ + \bar{\lambda} d_+$;
2.5. ItTot $\leftarrow$ ItTot+1;
3. ENQUANTO $(q(x_+) \geq q_{min})$;
4. $d_k \leftarrow x_+ - x_k$;
5. $\bar{x} \leftarrow x_+$.

1.6 Aplicações

Das inúmeras possíveis aplicações da minimização de funções quadráticas sujeitas a canalizações selecionamos quatro tipos de problemas com características bastante diferentes, e já incluídos na bibliografia associada ao assunto, para auxiliar a análise do desempenho de QuaCon. O fator preponderante nesta escolha foi a possibilidade de adequar os problemas ao estudo do comportamento do algoritmo, não só no que diz respeito ao tratamento de matrizes com dimensões e padrões de esparsidade variados (assunto que nos interessa particularmente), mas também no que se refere, por exemplo, à convergência na presença de soluções dual degeneradas.

1.6.1 O problema do obstáculo

O objetivo do primeiro conjunto de problemas que vamos comentar é a obtenção da posição de equilíbrio de uma membrana elástica com tensão τ que, presa pela borda a uma curva Γ (no plano horizontal), foi submetida a uma força vertical f e obrigada a amoldar-se a um obstáculo inferior e a outro obstáculo superior, os quais iremos representar por intermédio de funções simples. A descrição completa deste problema e de seu significado físico encontra-se no livro de Ciarlet [8], referenciado em nossa bibliografia.

Como nos experimentos de Dembo e Tulowitzski [12], Moré e Toraldo [31] e Friedlander e Martínez [17], consideramos constante a força f, supomos que a curva Γ que define o domínio correponde a um retângulo e, usando elementos quadrados com lados iguais a h, aproximamos o problema através do método dos elementos finitos de modo a reescrevê-lo na forma

$$\begin{aligned} &\textit{minimizar} \quad && q(v) = \tfrac{1}{2} v^T H v - b^T v \\ &\textit{sujeita a} && l \leq v \leq u \end{aligned} \tag{1.42}$$

onde H é a matriz bloco diagonal oriunda da aproximacão por diferenças do operador laplaciano (ou seja, com elementos iguais a 4 na diagonal principal e a -1 na outras posições) e $b = [b_i]$ é tal que $b_i = fh^2$. Cada elemento v_i corresponde ao valor da função v (isto é, à posição vertical da membrana) no ponto de índice i do domínio, cujas coodenadas são x_i

e y_i. Os vetores l e u definem para cada ponto a altura dos obstáculos inferior e superior, respectivamente.

Três são os tipos de problemas desta classe que, a exemplo de Friedlander e Martínez [17], utilizamos em nossos experimentos. Em todo eles consideramos que a força f tem valor constante e igual a 1 e descrevemos os obstáculos através das funções $l(x, y)$ e $u(x, y)$, conforme a tabela 1.1.

Problema	$l(x, y)$	$u(x, y)$
A	$p_1[\sin(3.2x)\sin(3.3y)]^{p_2}$	2000
B	$[\sin(9.2x)\sin(9.3y)]^{p_1}$	$[\sin(9.2x)\sin(9.3y)]^{p_2} + 0.02$
C	$[16x(1-x)y(1-y)]^{p_1}$	$[16x(1-x)y(1-y)]^{p_2} + 0.01$

Tabela 1.1: obstáculos inferior e superior dos problemas

Observa-se que a definição de $l(x, y)$ e $u(x, y)$ foi feita propositalmente em função dos parâmetros p_1 e p_2. Atribuindo valores variados a estes termos, modificando a dimensão dos problemas e escolhendo pontos iniciais diferentes, com base nos vetores $\ell = [l_1, \ldots, l_n]^T$, $u = [u_1, \ldots, u_n]^T$ e $\underline{1} = [1, \ldots, 1]^T$, produzimos uma razoável gama de testes, que reunimos na tabela 1.2.

1.6.2 Problemas de reconstrução de imagens

Outra aplicação interessante para QuaCon tem origem na tomografia computadorizada (vide [25], [17]) e consiste em reconstruir uma imagem definida por uma função $v : [0, 1] \times [0, 1] \to [l, u]$ usando integrais de linha. Para tanto, dividimos cada aresta do quadrado $[0, 1] \times [0, 1]$ em nd intervalos de mesmo tamanho $h = \frac{1}{nd}$, de modo que o domínio da função contém $n = nd^2$ *pixels*, sobre os quais traçamos:

- nd raios horizontais igualmente espaçados, com alturas que vão de $h/2$ a $1 - h/2$;
- nd raios verticais também igualmente espaçados, com abcissas que vão de $h/2$ a $1 - h/2$;
- $2nd - 1$ raios com ângulo de 45^o em relação ao eixo-x, igualmente espaçados, começando pelo raio que liga os pontos $(0, 1 - h)$ e $(h, 1)$ e terminando no raio que liga $(1 - h, 0)$ a $(1, h)$;
- $2nd - 1$ raios com ângulo de 135^o em relação ao eixo-x, igualmente espaçados, começando pelo raio que liga os pontos $(0, h)$ e $(h, 0)$ e terminando no raio que liga $(1 - h, 1)$ a $(1, 1 - h)$;

o que totaliza $m = 6nd - 2$ raios.

#	Tipo	p_1	p_2	n	x_0
1	A	1	1	2601	ℓ
2	A	1	1	2601	$\underline{1}$
3	A	0.3	1	2601	ℓ
4	A	0.3	1	2601	$\underline{1}$
5	A	1	2	2601	ℓ
6	A	1	2	2601	$\underline{1}$
7	A	1	3	2601	ℓ
8	A	1	3	2601	$\underline{1}$
9	A	1	1	5041	ℓ
10	A	1	1	5041	$\underline{1}$
11	A	0.3	1	5041	ℓ
12	A	0.3	1	5041	$\underline{1}$
13	A	1	2	5041	ℓ
14	A	1	2	5041	$\underline{1}$
15	A	1	3	5041	ℓ
16	A	1	3	5041	$\underline{1}$
17	A	1	1	10000	ℓ
18	A	1	1	10000	$\underline{1}$
19	A	0.3	1	10000	ℓ
20	A	0.3	1	10000	$\underline{1}$
21	A	1	2	10000	ℓ
22	A	1	2	10000	$\underline{1}$
23	A	1	3	10000	ℓ
24	A	1	3	10000	$\underline{1}$
25	B	3	2	5041	u
26	B	3	2	5041	ℓ
27	B	3	2	5041	$(\ell + u)/2$
28	C	3	2	5041	u
29	C	3	2	5041	ℓ
30	C	3	2	5041	$(\ell + u)/2$

Tabela 1.2: problemas de obstáculo.

A imagem a ser reconstruída é obtida igualando-se a valores obtidos experimentalmente as integrais de linha discretizadas definidas pelas direções dos raios.

Obtém-se assim um sistema linear esparso $Ax = d$ que tem como incógnitas os valores da função v nos n *pixels* e onde cada elemento a_{ij} da matriz A corresponde ao comprimento da interseção entre o raio i e o pixel j. Já o termo d_i contém o valor fornecido pelo tomógrafo.

Para determinar a solução do sistema formulamos o seguinte problema de quadrados mínimos

$$\begin{array}{rl} \textit{minimizar} & \frac{1}{2}||Ax - d||_2^2 \\ \textit{sujeita a} & l \leq x \leq u \end{array} \tag{1.43}$$

cuja matriz hessiana, $H = A^T A$, além de esparsa, é singular, uma vez que o número de raios é menor que o número de *pixels* (supondo que $nd > 5$).

Os resultados numéricos que apresentaremos na seção 1.7 foram obtidos com base nos dados gerados a partir de três diferentes formulações da função v, as quais estão enumeradas abaixo. Em todos os problemas supomos que, para cada *pixel*, o valor da função que precisamos reconstruir é constante e equivale a $v(\bar{x}, \bar{y})$, onde $\bar{x}$ e $\bar{y}$ são as coordenadas do centro do *pixel*.

- Problema A:

$$v_A(x, y) = \begin{cases} 1, & \text{se } 0.25 \leq x \leq 0.75; \\ 0, & \text{caso contrário .} \end{cases}$$

- Problema B:

$$v_B(x, y) = (x^2 + y)/2.$$

- Problema C:

$$v_C(x, y) = \min\{1, v_A(x, y) + v_B(x, y)\}$$

Consideramos também que, em qualquer um dos casos, $l = 0$, $u = 1$ e $nd = 256$, donde $n = 65536$.

1.6.3 Projeções de pontos sobre politopos

O cálculo da projeção de um dado ponto $y \in \mathbb{R}^n$ sobre um politopo pode ser descrito como um problema de programação quadrática do tipo

$$\begin{array}{rl} \textit{minimizar} & \frac{1}{2}||w - y||_2^2 \\ \textit{sujeita a} & Aw \leq d. \end{array} \tag{1.44}$$

Para transformar (1.44) em um problema de minimização com canalizações formulamos o seu dual:

$$\begin{array}{rc} \textit{maximizar} & \frac{1}{2}||w-y||_2^2 + x^T(Aw-d) \\ \textit{sujeita a} & w - y + A^T x = 0 \\ & x \geq 0 \end{array} \tag{1.45}$$

e substituimos

$$w = y - A^T x \tag{1.46}$$

de modo a obter

$$\begin{array}{rc} \textit{maximizar} & -\frac{1}{2}||A^T x||_2^2 + x^T(Ay-d) \\ \textit{sujeita a} & x \geq 0 \end{array} \tag{1.47}$$

ou, de forma equivalente,

$$\begin{array}{rc} \textit{minimizar} & \frac{1}{2}x^T AA^T x - (Ay-d)^T x \\ \textit{sujeita a} & x \geq 0. \end{array} \tag{1.48}$$

Obviamente este problema tem a forma (1.1), onde $H = AA^T$ e $b = d - Ay$. A matriz H é semidefinida positiva, tornando-se definida positiva apenas quando A tem posto coluna completo (como aliás é o caso dos exemplos que apresentaremos abaixo). O vetor w, solução do problema original de projeção (1.44), pode ser obtido facilmente usando (1.46).

Para analisar o desempenho de QuaCon empregaremos dois tipos de testes na forma (1.44). No primeiro deles, denominado *problema A*, supomos que esteja disponível um vetor y que representa as ordenadas de uma função $y(z)$ calculadas em alguns pontos específicos $z_1, z_2, \ldots, z_n$. Procuramos, então, aproximar $y(z)$ por uma função crescente $w(z)$ usando o método dos quadrados mínimos, o que representamos por:

$$\begin{array}{rc} \textit{minimizar} & \frac{1}{2}||w-y||_2^2 \\ \textit{sujeita a} & w_i \leq w_{i+1}, \quad i = 1, \ldots, n-1. \end{array} \tag{1.49}$$

Por sua vez, o segundo problema, ou *problema B*, difere do primeiro apenas no fato de tentarmos aproximar $y(z)$ por uma função $w(z)$ convexa, em lugar de crescente. Queremos, neste caso,

$$\begin{array}{rc} \textit{minimizar} & \frac{1}{2}||w-y||_2^2 \\ \textit{sujeita a} & w_i \leq (w_{i-1} + w_{i+1})/2, \quad i = 2, \ldots, n-1 \end{array} \tag{1.50}$$

Quanto à escolha de $y(z)$, decidimos utilizar cinco funções em nossos experimentos numéricos:

1. $y(z) = z$;
2. $y(z) = \log(z + 0.01)$;
3. $y(z) = \sin(1.5z)$;
4. $y(z) = 1/(1 + 9e^{-6z})$;
5. $y(z) = 1.6z^2 - 0.7z + 0.1$.

Em cada caso supomos conhecidos os valores de $y(z)$ em uma centena de pontos $z_1, \ldots, z_{100}$ igualmente espaçados no intervalo $[0, 1]$. Com base nestes dados, geramos um vetor y cuja i-ésima componente é definida pela soma de $y(z_i)$ a um valor aleatório pequeno.

1.6.4 Problemas aleatórios

Além de aplicações reais, incluímos, em nosso conjunto de testes, problemas aleatórios criados artificialmente segundo sugestão formulada por Moré e Toraldo [30].

Cada um destes problemas, escritos na forma (1.1), possui uma matriz hessiana H definida por

$$H = YDY \tag{1.51}$$

e

$$Y = I - \frac{2}{\|y\|_2^2} yy^T. \tag{1.52}$$

Para obter esta matriz Y, geramos aleatoriamente (dentro do intervalo $[-1, 1]$) as componentes do vetor y. Já os elementos não nulos da matriz diagonal D são calculados segundo:

$$d_{ii} = 10^{\left(\frac{i-1}{n-1}\right) ncond}. \tag{1.53}$$

onde $ncond$ é um parâmetro cujo valor devemos escolher de acordo com o número de condição desejado para H, já que ncond = $\log(\text{cond}(H))$.

A solução do problema, x^*, é gerada de forma tal que suas componentes estão distribuídas aleatoriamente em $(-1, 1)$.

Definida a solução, podemos estipular aproximadamente a dimensão da face ótima do problema utilizando o parâmetro $na(x^*) \in [0, n]$. Uma variável x_i^* estará fixada em um de seus limites se $\mu_i n \leq na(x^*)$, onde μ_i é um número aleatório entre 0 e 1.

Os conjuntos I_l^* e I_u^* reúnem os índices das canalizações — inferiores e superiores, respectivamente — ativas em x^*. Com isso, os limites de cada componente x_i^* são dados por

$$l_i = \begin{cases} x_i^*, & \text{se } i \in I_l^* \\ -1, & \text{caso contrário .} \end{cases} \tag{1.54}$$

e

$$u_i = \begin{cases} x_i^*, & \text{se } i \in I_u^* \\ 1, & \text{caso contrário .} \end{cases} \tag{1.55}$$

O vetor b é calculado com o auxílio da fórmula $b = Hx^* - r$, onde

$$r_i = \begin{cases} 10^{-\mu \text{deg}}, & \text{se } i \in I_l^* \\ -10^{-\mu \text{deg}}, & \text{se } i \in I_u^* \\ 0, & \text{caso contrário .} \end{cases} \tag{1.56}$$

O parâmetro *deg* foi introduzido no expoente dos termos em (1.56) para que possamos gerar soluções mais ou menos próximas da degeneração dual.

Determinamos x_0, nosso ponto inicial de forma semelhante àquela descrita para x^*. A partir de $na(x_0)$, um valor aproximado da dimensão da face que contém x_0, obtemos os conjuntos I_l^0 e I_u^0, de modo que

$$(x_0)_i = \begin{cases} l_i, & \text{se } i \in I_l^0 \\ u_i, & \text{se } i \in I_u^0 \\ (l_i + u_i)/2, & \text{caso contrário .} \end{cases} \tag{1.57}$$

Problemas degenerados são criados a partir dos dados acima forçando alguns elementos do vetor r correspondentes a variáveis no limite a assumirem o valor zero.

Da mesma forma, admitimos autovalores nulos na matriz H se permitimos que alguns elementos da diagonal de D sejam iguais a zero.

Para as experiências que apresentaremos a seguir, geramos problemas com 1000 variáveis, a partir de diferentes valores fornecidos para os termos *ncond*, *deg*, $na(x^*)$ e $na(x_0)$ supra citados, conforme disposto na tabela 1.3.

Incluimos, também na tabela 1.4 alguns outros problemas, agora degenerados ou com hessiana semidefinida positiva, fazendo variar dois parâmetros, *fdeg* e *nsing* que indicam, respectivamente, qual fração do vetor r e da diagonal da matriz D será constituída de elementos nulos.

Problema	$na(x^*)$	$na(x_0)$	deg
1	100	100	1
2	100	100	12
3	100	500	1
4	100	500	12
5	100	900	1
6	100	900	12
7	500	100	1
8	500	100	12
9	500	500	1
10	500	500	12
11	500	900	1
12	500	900	12
13	900	100	1
14	900	100	12
15	900	500	1
16	900	500	12
17	900	900	1
18	900	900	12

Tabela 1.3: problemas aleatórios do tipo A.

Problema	$na(x^*)$	$na(x_0)$	deg	$fdeg$	$nsing$
1	900	100	0	0	238
2	900	100	0	384	238
3	900	100	0	650	238
4	900	100	0	0	740

Tabela 1.4: problemas aleatórios do tipo B.

1.7 Resultados numéricos

Para que QuaCon pudesse ser avaliado, elaboramos uma primeira versão do algoritmo na forma de uma subrotina em FORTRAN, a qual implantamos em estações de trabalho com ambiente UNIX.

Em linhas gerais, o programa segue à risca o disposto nas seções 2 e 5 acima, sendo exigido para a sua utilização apenas o fornecimento de duas outras subrotinas: uma que seja capaz de calcular o produto da hessiana do problema por um vetor, e outra (necessária apenas quando pretendemos usar a decomposição de Cholesky) que forneça os elementos de qualquer coluna de H.

Ao tentar resolver os problemas associados às aplicações que acabamos de mencionar, centramos nosso interesse não apenas na análise do desempenho geral do algoritmo, mas pricipalmente nos efeitos relacionados a variação de alguns de seus parâmetros como:

- A dimensão máxima da face na qual permitimos o uso da decomposição de Cholesky (dada pela variável *LimChol*);
- O método usado nas faces de dimensão maior que *LimChol*, isto é, se usamos gradientes conjugados ou algum tipo de gradiente com retardo;
- A constante θ utilizada para decidir se saimos ou não de uma face (item 1.2, pág. 8);
- A constante δ que define se um passo na direção do gradiente "chopado" será aceito, particularmente se estamos tratando de problemas dual degenerados (item 1.2.2 do algoritmo).

Dado o número muito grande de dados colhidos nos experimentos, para evitar que se torne maçante a leitura desta seção decidimos apresentar aqui apenas um resumo dos resultados que obtivemos, utilizando eventualmente alguma figura para reforçar uma ou outra conjectura. Isto não significa, entretanto, que nossa análise não esteja baseada em todas as informações fornecidas pelos inúmeros testes feitos com QuaCon, as quais incluímos como uma coleção de tabelas no apêndice A.

1.7.1 Empregando a decomposição de Cholesky

Vamos estudar, inicialmente, que efeito prático proprociona o uso da decomposição de Cholesky no cálculo do vetor direção.

Como o desempenho deste tipo de método de minimização está intimamente relacionado ao problema que estamos resolvendo, por depender da estrutura de dados da matriz hessiana correspondente à face na qual trabalhamos, vemo-nos forçados a tratar de cada tipo de aplicação em separado.

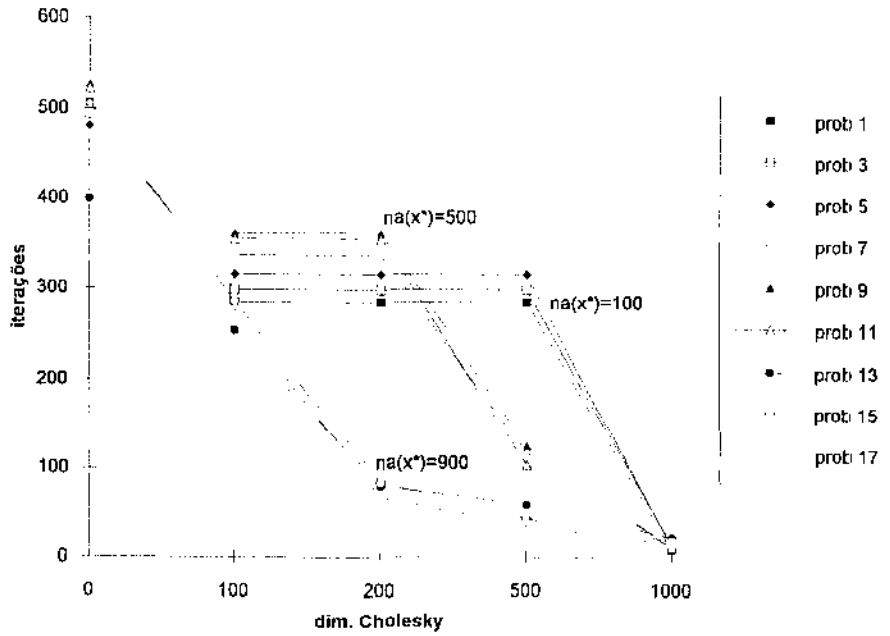


Figura 1.1: Problemas aleatórios do tipo A ($\theta = 0.1$).

1. *Problemas aleatórios*

Se a matriz H é totalmente densa, como é o caso dos problemas aleatórios que geramos, muito cuidado deve ser tomado na definição da dimensão máxima da face na qual utilizamos a decomposição de Cholesky.

Dizemos isto tendo em mente que cada vetor direção calculado por um método direto consome algo em torno de n^3 operações aritméticas, enquanto os métodos iterativos exigem apenas cerca de n^2 operações. Como se não bastasse, tal como se observa com relação ao trabalho computacional, a memória gasta no armazenamento da matriz triangular oriunda da fatoracão também cresce exponencialmente à medida em que aumentamos a dimensão da face. Desta forma, além de precisarmos exigir que o sistema linear seja pequeno, só conseguiremos alguma eficiência se a decomposição de Cholesky reduzir drasticamente o número de iterações necessárias à obtenção da solução global do problema.

Ainda assim, o fato dos problemas aleatórios serem totalmente controlávcis permitiu ao menos que pudéssemos utilizá-los para caracterizar de uma forma geral em quais circunstâncias parece ser interessante empregar um método direto. Para isso, vamos fixar nossa atenção na figura 1.1.

Como a face que contém x_0 parece estar longe de ser ótima, não percebemos em nenhum dos exemplos uma grande variação do comportamento de QuaCon quando alteramos $na(x_0)$ (o número de componentes de x_0 que estão em algum limite). Por outro lado, à medida em que reduzimos a dimensão da face que contém x^* (ou seja, que aumentamos $na(x^*)$), o desempenho do método direto melhora significativamente.

Ainda que a melhora obtida não seja suficiente para compensar, no nosso caso, o custo computacional criado, achamos que este comportamento pode ser generalizado para os problemas com matrizes mais esparsas, fazendo-nos crer que a utilidade da fatoração de Cholesky está intimamente ligada à dimensão da face ótima.

Também consideramos importante destacar aqui que, se a face ótima é pequena e se, portanto, nas últimas iterações teremos que executar sucessivas fatorações de matrizes até certo ponto semelhantes, poderíamos aumentar a eficiência dos métodos diretos se adotássemos alguma estratégia de atualização dos fatores L criados pela decomposição da hessiana reduzida, em lugar de recalculá-los sempre.

Embora não a tenhamos implementado, esta atualização poderia ser feita facilmente, por exemplo, com o emprego da conhecida fórmula de Sherman-Morrison, ou através da refatoração de uma parte pequena da matriz, economizando em muito o tempo gasto na decomposição, à custa de um pequeno consumo de memória.

2. *Problemas de reconstrução de imagens*

Mesmo quando a matriz hessiana não é completamente densa, existem aplicações, como as de reconstrução de imagens, em que o padrão de esparsidade de H não permite o uso freqüente da decomposição de Cholesky.

Percebemos isso facilmente a partir dos exemplos mencionados à seção 1.6.2, nos quais o domínio quadrado foi dividido em 256^2 *pixels*, e sobre estes traçamos 1534 raios. Deste modo, o problema quadrático resultante tem sua matriz hessiana dada por $H = A^T A$, onde A possui 1534 linhas, 256^2 colunas e mais de cem milhões de elementos, dos quais apenas 263144 são diferentes de zero. Já o número de elementos em H é da ordem de 4 bilhões, dos quais estimamos que cerca de 53 milhões não são nulos. Neste caso, antes mesmo de fazer a decomposição, o próprio cálculo de uma pequena parte de H pode ser uma operação cara em termos computacionais. Mas, como veremos, as dificuldades não param por aí.

Antes de prosseguir analisando o que aconteceria se fizéssemos a decomposição propriamente dida, para simplificar o trabalho, vamos supor que estamos lidando com um problema com apenas 16^2 pixels e 94 raios, capaz de reproduzir em menor escala o que acontece nos sistemas muito grandes. A matriz A seria, então, aquela representada pela figura 1.2[4]. Por sua vez, a matriz H teria um total de 65536 elementos, 3397 dos quais seriam diferentes de zero.

Imaginemos que é preciso resolver um problema no qual eliminamos de H várias linhas e colunhas de modo a obter $\bar{H}$ cuja dimensão é igual a 120, conforme mostra a figura 1.3.

Observa-se que, neste exemplo, embora menos de 30 porcento do elementos de $\bar{H}$ sejam diferentes de zero, a decomposição desta matriz produz um fator L ex-

[4]Nos gráficos, o termo *nz* expressa o número de elementos não nulos da matriz.

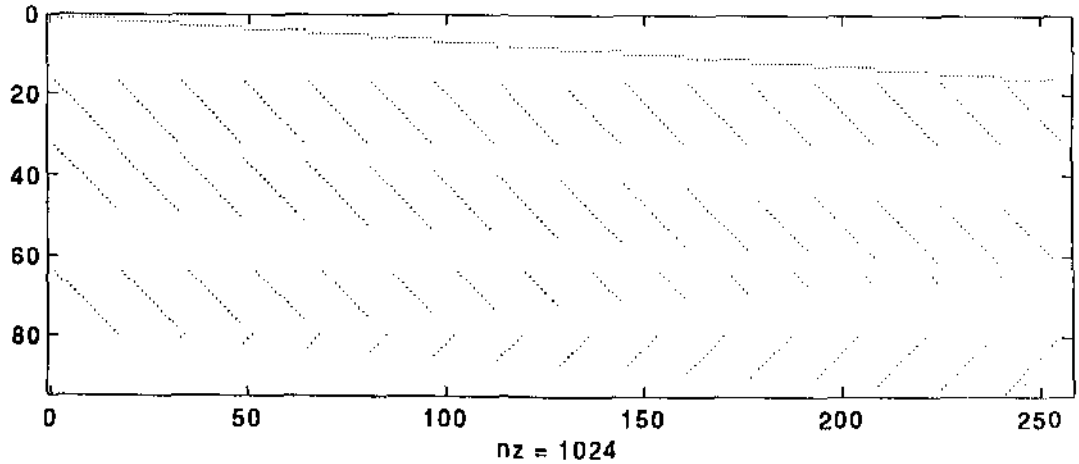


Figura 1.2: Matriz A.

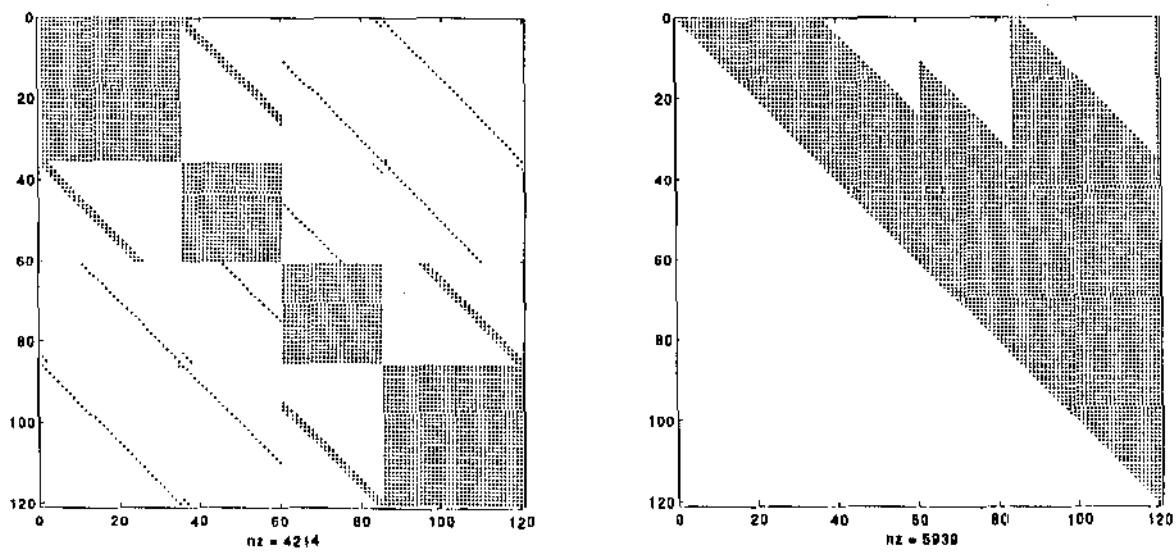


Figura 1.3: Matrizes $\bar{H}$ e L^T, onde $\bar{H} = LL^T$.

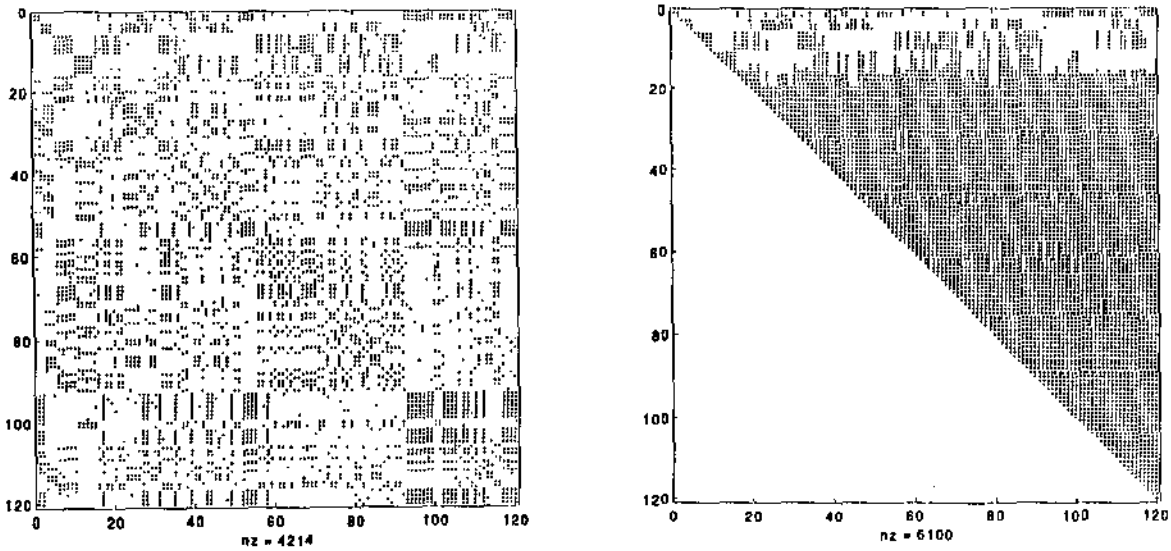


Figura 1.4: A mesma matriz $\bar{H}$ reordenada, e L^T, onde $\bar{H} = LL^T$.

tremamente denso, com 5939 elementos não nulos de um total de 7260, o que significa mais de oitenta porcento de preenchimento.

E mesmo escolhendo de forma totalmente diferente as colunas que compõem $\bar{H}$, obtivemos em outro exemplo semelhante uma matriz com 3397 elementos diferentes de zero, para qual o fator L apresentou 93 porcento de enchimento, mostrando que estes maus resultados não estão restritos a algumas faces do problema.

Poderíamos, então, nos perguntar se um reordenamento das linhas e colunas da hessiana não seria suficiente para fazer com que L fosse mais esparsa. A resposta, ao menos quando se trata dos problemas de reconstruções de imagens, seria um categórico não. E é fácil mostrar porque.

Primeiro vamos observar o que acontece se aplicamos uma permutação de linhas e colunas, segundo o algoritmo do grau mínimo[5], sobre a hessiana que define o problema quadrático geral (e que no nosso exemplo reduzido tem dimensão igual a 256). Neste caso, extraindo de H a mesma submatriz $\bar{H}$ que definimos acima, esta teria um padrão de esparsidade tal como o exposto na figura 1.4. Como se observa (na mesma figura), L tem 6100 elementos não nulos, número ainda maior que o que foi obtido sem as permutações.

Se, insatisfeitos com o efeito negativo demonstrado pelo reordenamento de uma matriz grande sobre a decomposição de uma submatriz sua, resolvermos utilizar o método do grau mínimo sobre $\bar{H}$, de forma a tentar minimizar o enchimento de

[5] Veja a seção 1.5.1.

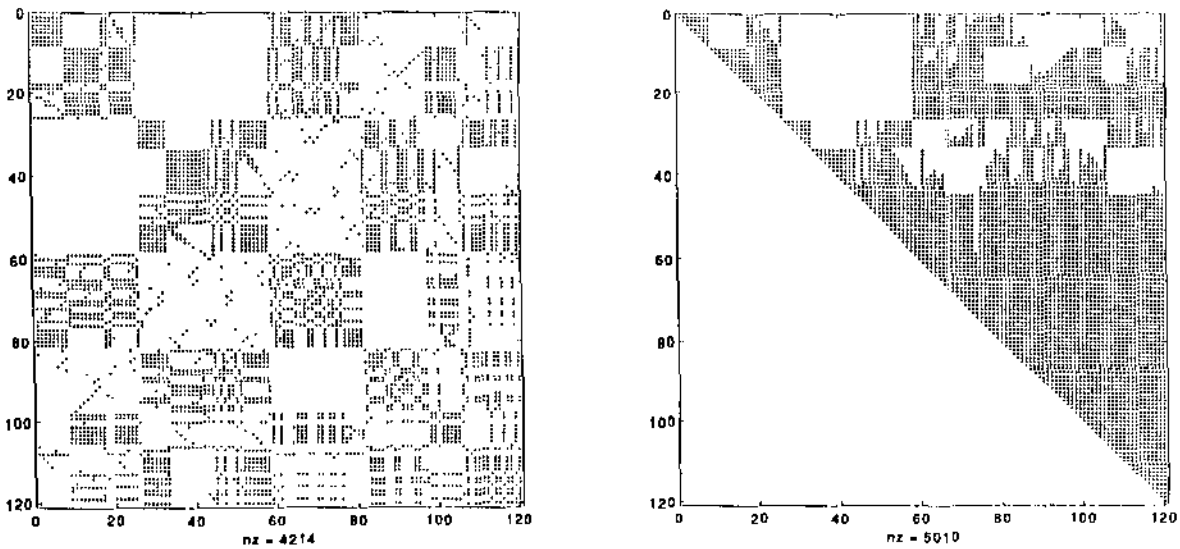


Figura 1.5: As matrizes $\bar{H}$ e L^T depois de um segundo reordenamento.

L diretamente, mesmo sabendo tratar-se de uma operação cara e que provavelmente só será aproveitada em uma iteração, ainda assim obteremos resultados pífios, como ilustra a figura 1.5.

Mais do que justificar o fato de não termos resultados numéricos dos problemas de reconstrução de imagens para apresentar, buscamos com esses comentários expor os cuidados que devem ser tomados para que seja possvel utilizar os métodos diretos de forma eficiente.

3. *Problemas de obstáculo*

À diferença do que vimos até agora, em um problema de obstáculo a matriz hessiana é bastante esparsa, possuindo apenas cinco diagonais com elementos não nulos. Dessas diagonais as mais distantes estão a $m = \sqrt{n}$ posições da diagonal principal da matriz, o que faz com que a decomposição de Cholesky produza matrizes triangulares L também com banda igual a m.

E mesmo que o gasto de memória ainda não seja comparável ao consumo insignificante de métodos como o dos gradientes conjugados, poderíamos, ao menos, conjecturar que se os sistemas lineares ralativamente são pequenos os métodos diretos são favorecidos.

Mas para nossa infelicidade, como bem ilustra a figura 1.6, nos problemas de obstáculo quase só percorremos faces grandes. Até mesmo quando partimos de um ponto inicial com muitas variáveis fixas em seus limites, um passo na direção do gradiente "chopado" já é suficiente para que aumentemos rapidamente a dimensão da

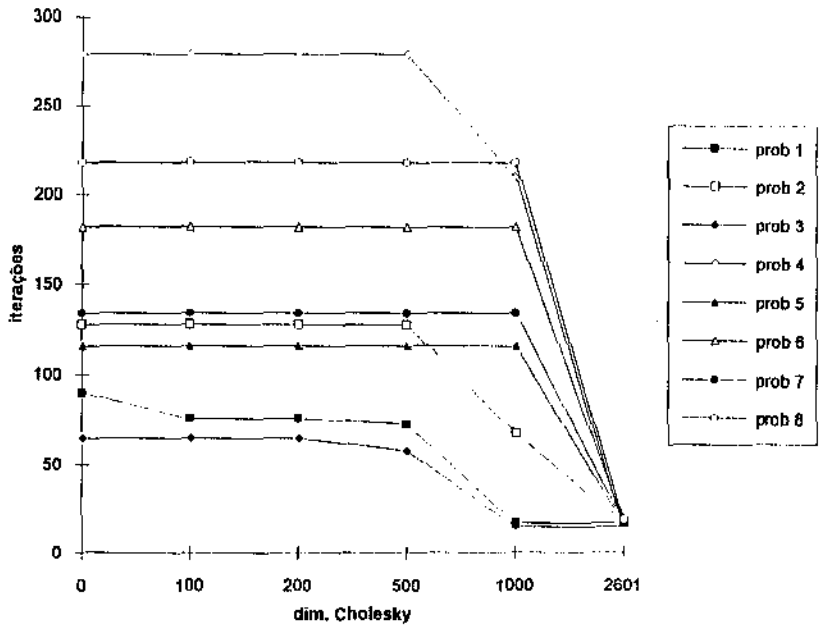


Figura 1.6: Problemas de obstáculo do tipo A ($n = 2601$, $\theta = 0.1$).

face, de forma que em quase todos os exemplos verificamos na prática que somente quando $DimChol$[6] é muito grande o número de iterações cai significativamente.

4. *Problemas de projeção*

Para encerrar esta seção, vamos tratar dos problemas de projeção, nos quais a hessiana tem banda pequena, sendo tridiagonal nos problemas do tipo A e pentadiagonal nos do tipo B.

Em ambos os casos, a decomposição de Cholesky não somente pode ser feita sem nenhum consumo adicional de memória, como tem um baixíssimo custo computacional em termos de operações aritméticas.

Por outro lado, as matrizes são muito mal condicionadas, fato que vai se agravando à medida em que aumentamos a dimensão do problema, e que tem como reflexo o péssimo desempenho dos métodos iterativos, conforme constatamos na figura 1.7 (cabendo inclusive menção à escala logarítmica nela empregada).

Nos problemas testados, que possuem não mais que cem variáveis, o número condição das matrizes chega a atingir 4×10^3 para os problemas do tipo A e 3×10^6 para os do tipo B. Aumentando a dimensão dos problemas de forma a fazer com que contenham 1000 variáveis, estes valores sobem para 5×10^5 e 4×10^{10} respectivamente, tornando imprevisível os resultados relativos aos métodos do gradiente e dos gradientes conjugados.

[6]variável que representa a dimensão máxima da face em que usamos a decomposição de Cholesky.

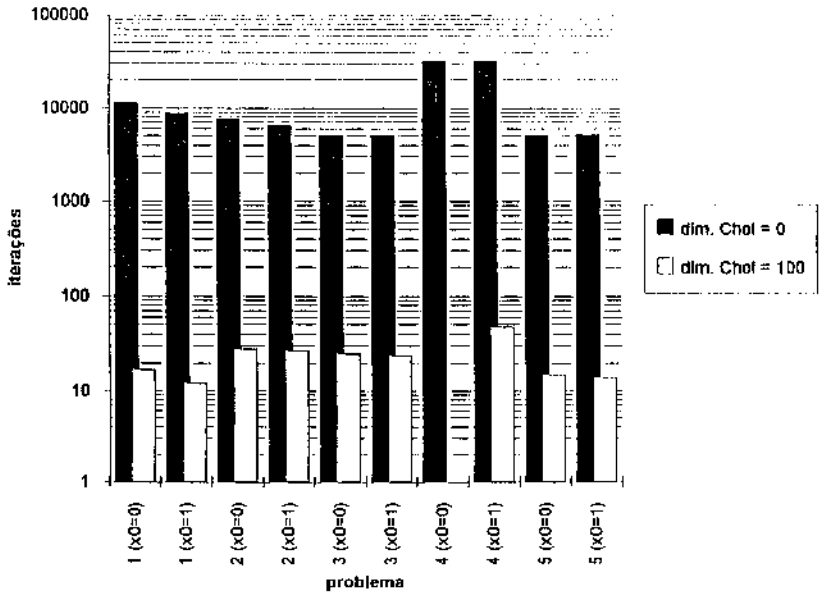


Figura 1.7: Problemas de projeção do tipo B (com duas alternativas para os pontos iniciais).

O mesmo fenômeno foi observado quando alteramos o número de condição da hessiana dos problemas aleatórios. Se nos exemplos já mencionados tínhamos $\log(ncond(H)) = 3$, fazendo subir este valor para 5 constatamos mais uma vez a dificuldade encontrada pelos método iterativos na resolução dos sistemas lineares, ficando também patente a importância de termos incluído um método direto dentre as alternativas de minimização irrestrita usadas por QuaCon.

Apenas a título de ilustração, apresentamos no apêndice A, alguns gráficos que contêm os resultados referentes a aproximação de funções através da projeção de pontos sobre politopos.

No que é possvel reunir todas estas informações para fazer uma análise global, parece interessante que um algoritmo para minimização de quadráticas inclua a possibilidade de que sistemas lineares pequenos sejam resolvidos da forma mais direta possvel. Contudo, para que esta alternativa seja completamente explorada, é imprescindvel que algum cuidado seja tomado para que a dimensão em que usamos um método direto e a estrutura de dados a ser adotada sejam escolhidas em função das caractersticas do problema que estamos tratando.

1.7.2 Investigando o desempenho dos métodos com retardo

Além de terem servido para que pudéssemos estudar o comportamento de QuaCon em função da dimensão das faces percorridas pelo algoritmo, nossas experiências numéricas também têm como objetivo estabelecer em que condições as variantes do método do gradiente são capazes de concorrer, em termos de eficiência, com o método dos gradientes conjugados e quais são as melhores alternativas para definir o retardo.

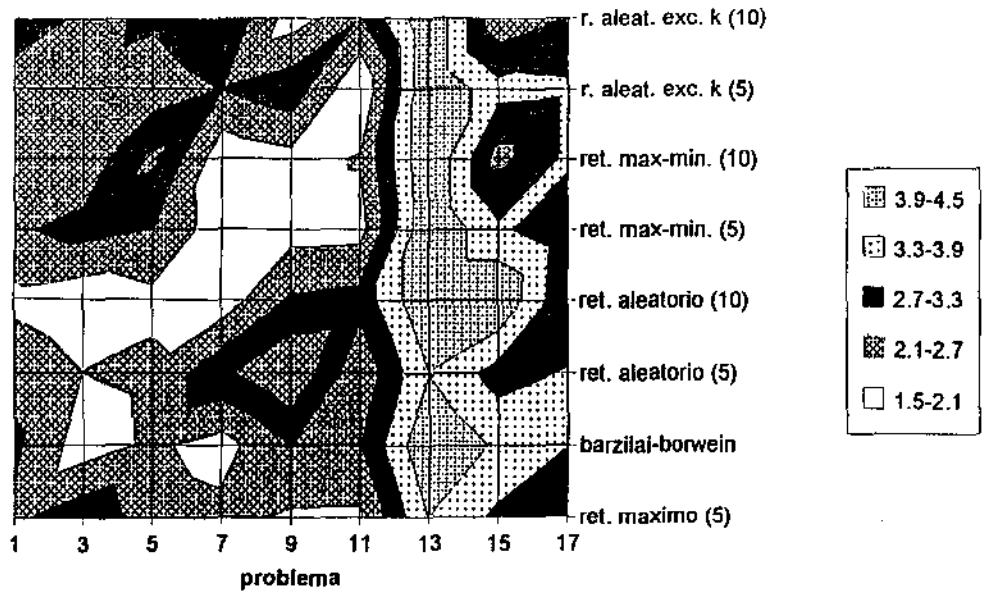


Figura 1.8: Problemas aleatórios do tipo A ($x_0 = \ell$)

Para permitir que este tipo de análise fosse feita, aplicamos QuaCon a um número razoavelmente grande de problemas, incluindo em cada um deles a possibilidade de escolhermos λ segundo:

- O retardo máximo com $m = 5$;
- O retardo máximo com $m = 1$ (a proposta de Barzilai e Borwein);
- Um retardo aleatório, com $m = 5$ e $m = 10$;
- O retardo mnimo-máximo, com $m = 5$ e $m = 10$;
- Um retardo aleatório excetuando o caso em que $\nu(k) = k$, com $m = 5$ e $m = 10$.

A partir dos dados obtidos experimentalmente, elaboramos gráficos do tipo mapa, nos quais as colunas estão relacionadas aos problemas, enquanto as linhas correspondem as escolhas diferentes de retardo. Em cada região, um determinado padrão representa o valor aproximado da fração it_{ret}/it_{gc}, onde it_{ret} indica quantas iterações foram necessárias para que o método com retardo convergisse, e it_{gc} representa o número de iterações gasto pelo método dos gradientes conjugados. Os intervalos correspondentes aos diversos padrões são fornecidos na legenda que aparece à direita da figura.

Observando, então, o gráficos coluna por coluna, podemos traçar um paralelo entre cada tipo de método com retardo. Da mesma forma, também não há dificuldade em identificar, a partir das cotas dos pontos de interseção entre as linhas horizontais e verticais, qual o comportamento dos métodos com relação aos gradientes conjugados.

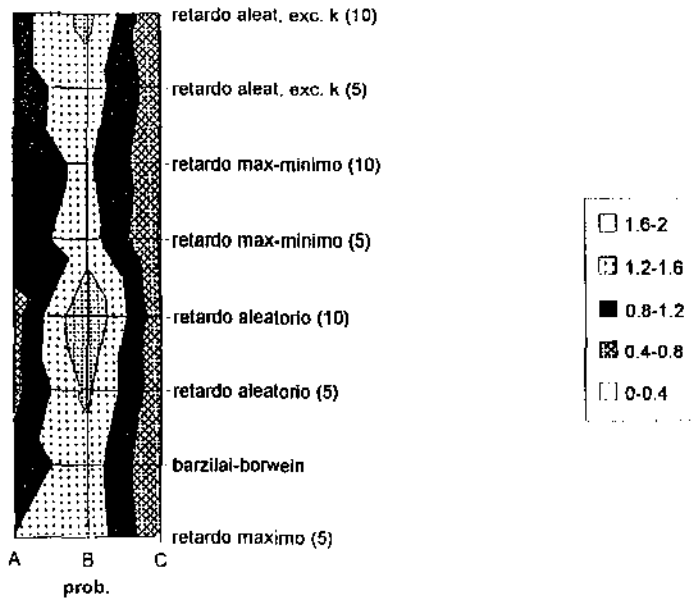


Figura 1.9: Problemas de reconstrução de imagens

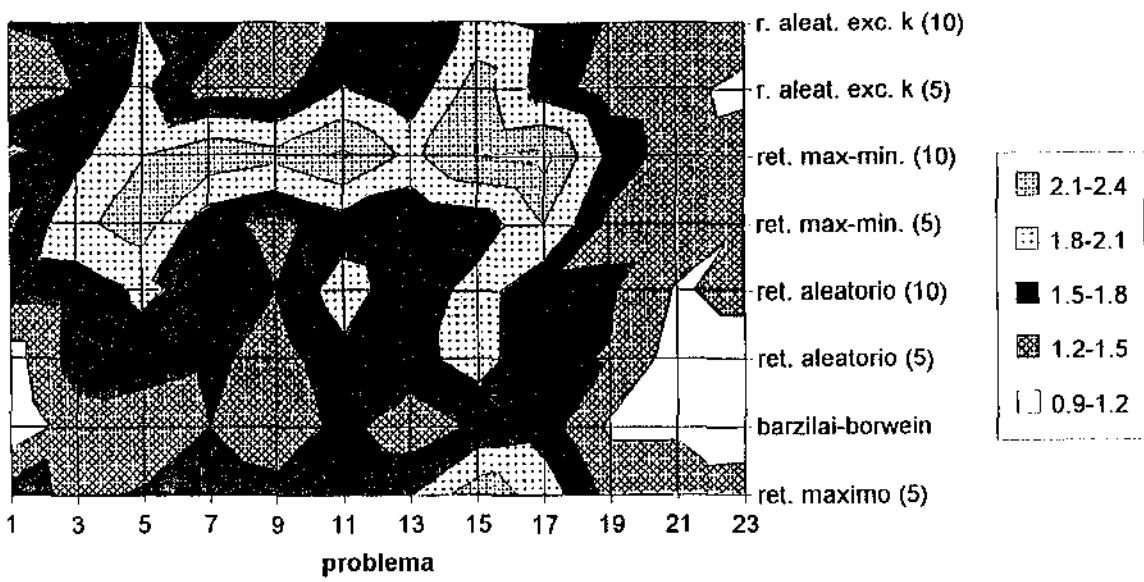


Figura 1.10: Problemas de obstáculo do tipo A ($x_0 = \ell$)

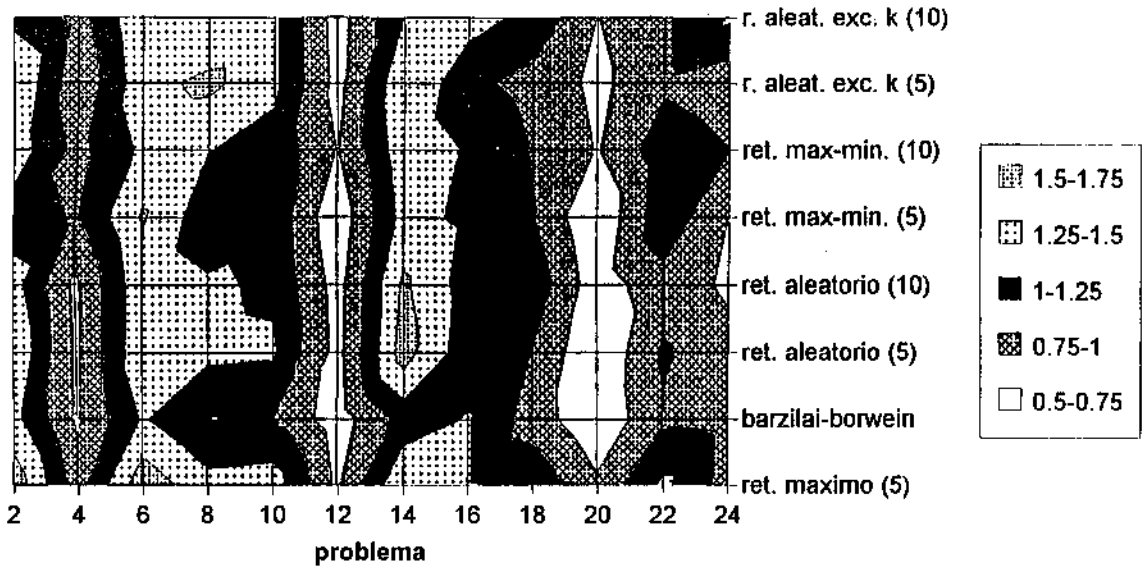


Figura 1.11: Problemas de obstáculo do tipo A ($x_0 = 1$)

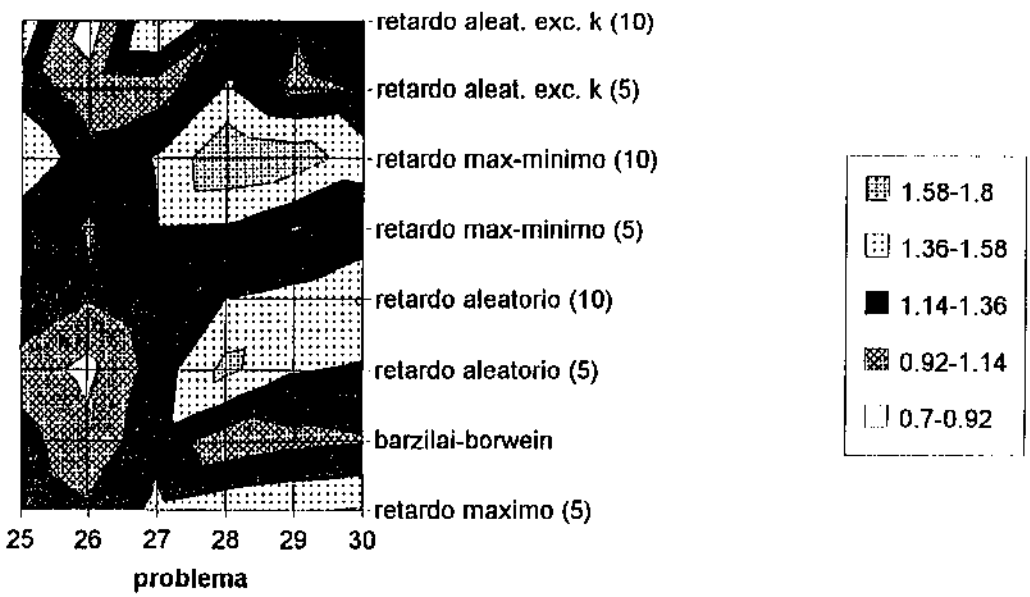


Figura 1.12: Problemas de obstáculo do tipo B e C

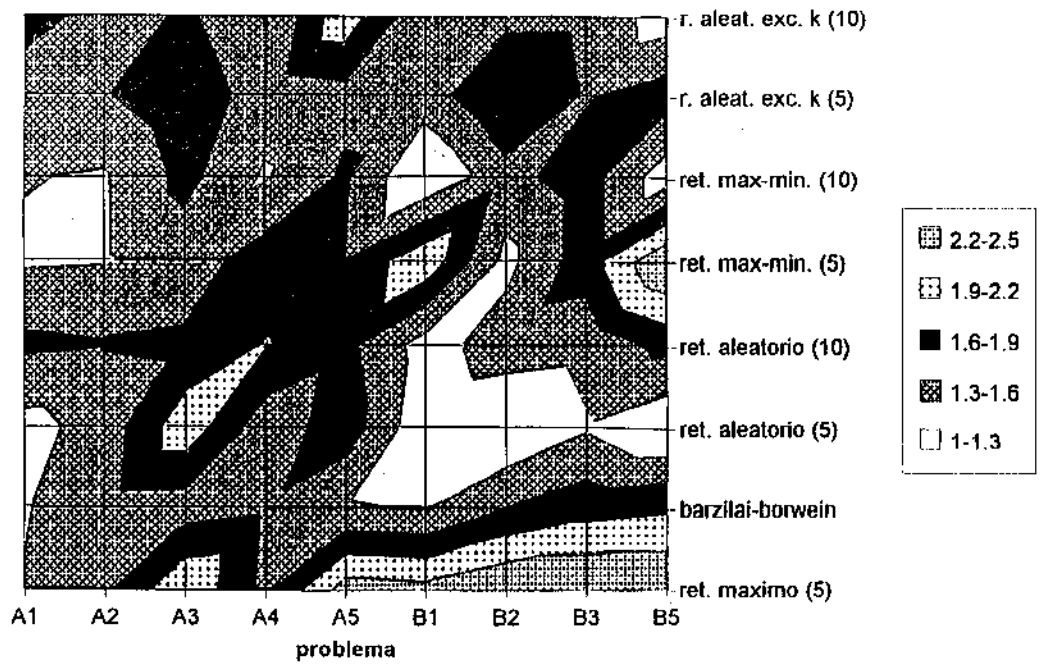


Figura 1.13: Problemas de projeção ($x_0 = 1$)

No entanto, a primeira constatação que fazemos a partir dessas figuras advém de seu número e não do conteúdo, pois parece óbvio que a necessidade de exibir muitos problemas ao mesmo tempo, mais que qualquer outra coisa, reflete a nossa dificuldade em definir com alguma segurança um comportamento médio para os diversos tipos de métodos com retardo.

Mas mesmo sendo grande a variação dos valores encontrados em alguns gráficos (os de número 1.8 e 1.13, por exemplo), podemos mencionar como conclusões preliminares que:

- Os melhores resultados comparativos foram obtidos para problemas grandes, como se constata particularmente a partir da figura 1.11, na qual quatro problemas de obstáculo são apresentados em versões que incluem 2601, 5041 e 10000 variáveis, de modo que os dados que aparecem mais à esquerda são relativos a problemas menores que os da direita.

- O aumento do valor de m, o retardo máximo permitido, em geral provoca uma deterioração do método correspondente, sendo raros os casos em que os resultados para $m = 10$ foram melhores que aqueles obtidos para $m = 5$, independentemente da dimensão do problema.

- O desempenho do método de Barzilai e Borwein parece ser, na média, bastante satisfatório, embora com algumas exceções (que incluem, por exemplo, os problemas de projeção).

- Levando em conta que o cálculo do vetor direção através do método de máxima descida é muito barato, podemos admitir, para efeito de comparação, que o número de iterações dos métodos com retardo seja ligeiramente maior que o valor obtido com o uso dos gradientes conjugados. Com isso, alternativas como aquela em que escolhemos o retardo de forma aleatória ou a que alterna o uso de um retardo máximo e um mínimo passam a ser interessantes.

- Para problemas como os de reconstução de imagens do tipo A e C, nos quais as mudanças de face são freqüentes, como também nos testes em que o parâmetro θ era pequeno, os resultados apresentados pelos métodos com retardo são muitas vezes melhores que os alcançados quando usamos gradientes conjugados, fazendo-nos crer que é preferível permenecer em uma face por algumas iterações, mesmo que passando por pontos nos quais o valor da função quadrática aumenta, em lugar de adotar sucessivas iterações do método de máxima descida[7].

Como comentário final, salientamos que embora em momento algum fosse do nosso interesse mostrar que os métodos com retardo são a melhor opção para o cálculo da direção em QuaCon, as observações que fizemos parecem suficientes para que não os descartemos como uma alternativa bastante interessante, principalmente quando os problemas são muito grandes ou se, como foi lembrado por Friedlander, Martínez e Raydan [20], pretendemos utilizar computadores paralelos.

1.7.3 Analisando as mudanças de face

No algoritmo 1.1, as mudanças de face são definidas em função de dois parâmetros. O primeiro deles, θ, é usado para decidir se a norma do gradiente "chopado" é grande o suficiente a ponto de justificar que este vetor seja usado como direção, enquanto o segundo, δ, permite que recusemos um passo na direção de $g_C(x_k)$ se o decréscimo da função quadrática não for suficiente.

A importância de θ está relacionada ao fato de ser com base em seu valor que definimos até que precisão pretendemos nos aproximar do minimizador de uma face. E como disso muito depende o desempenho de QuaCon, em uma larga maioria das experiências que executamos, fizemos este parâmetro assumir os valores 10^{-10}, 10^{-5}, 10^{-2}, 0.1 e 0.9, para que pudéssemos verificar qual deles proporciona os melhores resultados práticos (em termos do número de iterações).

O que constatamos como uma tendência comum a quase todos os testes foi que, se não exigimos a minimização exata nas faces do problema, ao menos parece não valer a pena sair delas com muita frequência, já que o número de iterações de QuaCon aumenta consideravelmente quando usamos valores muito pequenos de θ.

[7] Como acontece com o método dos gradientes conjugados quando mudamos de face.

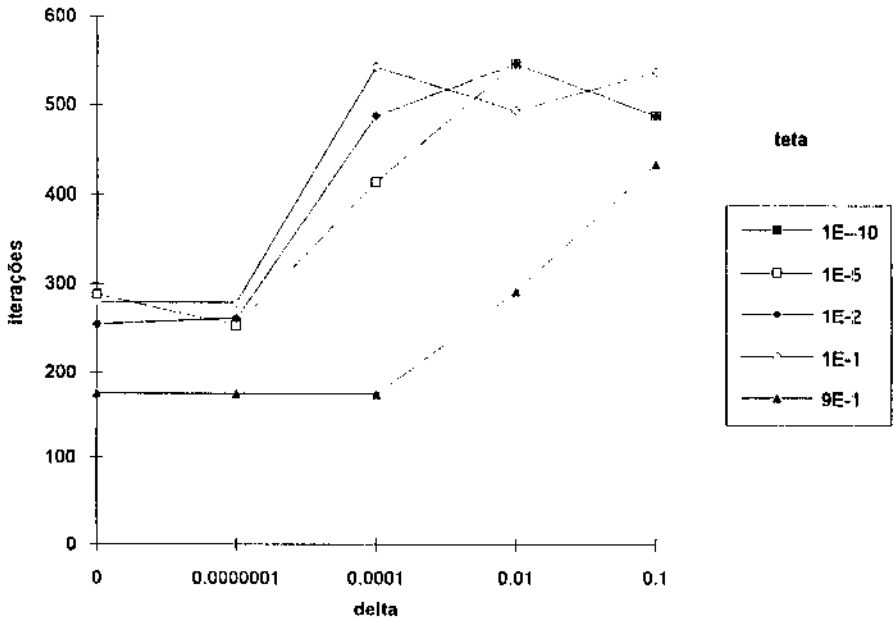


Figura 1.14: Problema de obstáculo do tipo A ($n = 2601$, $p_i = 0.3$, $x_0 = 1$).

Por outro lado, independentemente do método de minimização que utilizamos dentro da face, os melhores resultados foram sempre obtidos para $\theta = 0.1$ ou $\theta = 0.9$, com uma ligeira vantagem para o primeiro.

Já no que diz respeito a δ, sabendo que sua importância limita-se àqueles problemas em que há degeneração dual, utilizamos de nosso conjunto de testes os problemas que possuem essa caracterstica e variamos δ entre 0 e 0.5, para analisar a influência deste parâmetro sobre a convergência do algoritmo.

Observamos, então, que para obter bons resultados precisamos fazer δ assumir valores muito baixos, como mostra o gráfico 1.7.3, havendo mesmo certos casos em que o uso de valores altos para δ provocaram um aumento significativo do número de iterações.

Embora isso pareça contraditório com a sugestão de manter θ grande, acreditamos que, na prática, se por um lado o uso de valores altos para δ evita que abandonemos indevidamente a face ótima, por outro também pode exigir que um bom número de iteraccões sejam feitas desnecessariamente em torno de pontos degenerados pertencentes a muitas outras faces.

Assim, se fosse preciso recomendar algum valor deste parâmetro para uso geral, mesmo em problemas degenerados, certamente a melhor alternativa seria tentar usar inicialmente $\delta = 0$.

CAPÍTULO 2

Um algoritmo para minimização com restrições

Neste capítulo, bem como no próximo, iremos nos aventurar a tratar de problemas de uma das classes mais abrangentes da otimização: a programação não linear com restrições de igualdade e desigualdade.

Nossa atenção estará centrada na apresentação de um método que, além de ser absolutamente original, enquadra-se no que conhecemos como programação quadrática seqüencial, permitindo a manutenção da linha mestra da tese, e até mesmo o uso do algoritmo recém descrito para minimização em caixas.

2.1 Descrição do problema

O objetivo de um problema geral de otimização é a minimização de uma função não linear sujeita a quaisquer restrições também não lineares. o que por conveniência iremos escrever como

$$\begin{array}{rl} \textit{minimizar} & f(x) \\ \textit{sujeita a} & C(x) = 0 \\ & l \leq x \leq u \end{array} \tag{2.1}$$

onde o termo $\leq$ é usado componente a componente e $l < u$.

Embora esta não seja a representação mais freqüentemente utilizada, parece claro que a conversão de um problema no qual aparecem restrições de desigualdade para o formato

acima não constitui dificuldade alguma, exigindo apenas a introdução de algumas variáveis de folga e excesso. Por outro lado, nenhuma exigência adicional será feita sobre as componentes de (2.1), a não ser a suposição de que são contínuas as primeiras derivadas parciais da função $f : \mathbb{R}^n \to \mathbb{R}$ e das restrições $C : \mathbb{R}^n \to \mathbb{R}^m$.

Para comentar de forma breve e superficial o tipo de abordagem utilizada pelos diversos algoritmos já apresentados para otimização com restrições, iremos dividí-los em:

1. *Métodos do tipo gradiente reduzido* (veja [22], [29]).

   Retomando em parte a idéia dos métodos de restrições ativas para problemas com restrições lineares, em uma iteração típica de um algoritmo desta classe as variáveis são divididas em dois conjuntos. O primeiro, x_B, com m elementos, é composto das variáveis *básicas*, que devem satisfazer $l_B < x_B < u_B$. O outro, x_N, tem $n-m$ variáveis (supondo, obviamente, que $m < n$) ditas *não básicas*. Usando, então, as m restrições disponíveis, as componentes de x_B são isoladas de forma a tornar possível escrever o problema apenas em função das variáveis não básicas e de suas canalizações.

   Em seguida, adota-se um passo na direção definida pela minimização de uma aproximação de primeira ou segunda ordem da função objetivo do problema reduzido, mantido o cuidado de não tornar infactíveis as variáveis não básicas cujas canalizações estão ativas. Com a mudança dos valores das componentes de x_N, as variáveis básicas precisam ser recalculadas para manter $C(x) = 0$, o que é feito através da resolução de um sistema não linear pelo método de Newton.

2. *Métodos do tipo lagrangiano aumentado* (veja [3], [10]).

   Em cada iteração deste tipo de método é preciso resolver aproximadamente um subproblema que consiste em minimizar, com relação a x, o lagrangiano aumentado (2.8) definido pelo problema (2.1), desconsideradas as canalizações, que são tratadas separadamente[1]. A isto se segue uma atualização criteriosa do vetor de multiplicadores e do parâmetro de penalização (representados, respectivamente, por λ e θ em (2.8)).

3. *Métodos que usam programação quadrática seqüencial* (veja [16]).

   Também nestes métodos resolvemos um problema quadrático por iteração. O modo como isso se dá, entretanto, nos absteremos de comentar aqui para evitar redundância, já que na seção 2.2 apresentaremos detalhadamente um algoritmo desta natureza.

2.1.1 Definições e notação

Da mesma maneira que no primeiro capítulo, para facilitar uma eventual consulta posterior cabe-nos o dever de mencionar que:

[1]Como o subproblema definido a cada iteração é quadrático e possui apenas canalizações, o algoritmo QuaCon descrito no primeiro capítulo pode perfeitamente ser utilizado para sua solução, assunto ao qual retomaremos brevemente.

- Adotaremos o termo $||\cdot||$ para representar uma norma qualquer em $\mathbb{R}^n$;
- Ao longo do texto, iremos tratar de forma diferenciada as restrições de igualdade e as canalizações, de modo que definimos em separado o conjunto

$$\Omega = \{x \in \mathbb{R}^n \mid l \leq x \leq u\} \tag{2.2}$$

e denominamos *factível* um ponto $\bar{x}$ se $\bar{x} \in \Omega$ e $C(\bar{x}) = 0$.

- Derivando a função objetivo com relação às componentes de x, escrevemos o vetor *gradiente*:

$$g(x) = \nabla f(x). \tag{2.3}$$

- Definimos o *Jacobiano das restrições* é como uma matriz $m \times n$ cujas colunas são obtidas derivando $C(x)$ com relação a cada componente de x:

$$A(x) = \nabla C(x)^T. \tag{2.4}$$

- Esquecendo, por hora, as canalizações, definimos em função de $x \in \mathbb{R}^n$ e dos *multiplicadores* $\lambda \in \mathbb{R}^m$ o *lagrangiano* associado ao problema com restrições de igualdade:

$$\ell(x, \lambda) = f(x) + \lambda^T C(x), \tag{2.5}$$

cujas derivadas parciais com relação às componentes de x fornecem

$$\nabla_x \ell(x, \lambda) = g(x) + A(x)^T \lambda, \tag{2.6}$$

e que tem como hessiana

$$\nabla^2_{xx} \ell(x, \lambda) = \nabla^2 f(x) + \sum_{i=1}^{m} \lambda_i \nabla^2 C_i(x). \tag{2.7}$$

- Com base em um parâmetro de penalização $\theta \in [0, 1]$, usamos uma expressão bastante original para formular abaixo o que conhecemos como *lagrangiano aumentado*:

$$\mathcal{L}(x, \lambda, \theta) = \theta \ell(x, \lambda) + (1 - \theta)\varphi(x). \tag{2.8}$$

- Dada a função

$$\varphi(x) = \frac{1}{2}||C(x)||_2^2, \tag{2.9}$$

dizemos que um ponto $\bar{x} \in \Omega$ é *φ-estacionário* se é um ponto estacionário de $\varphi(x)$ sujeito a $x \in \Omega$.

- Derivando $\varphi(x)$ obtemos:

$$\nabla \varphi(x) = A(x)^T C(x). \tag{2.10}$$

- A aproximação de $C(x+s)$ obtida a partir da expansão em série de Taylor até o termo linear (em torno de x) denotamos por

$$A(x)s + C(x). \tag{2.11}$$

- Com base em (2.11) definimos a aproximação de $\varphi(x+s)$ como

$$M(x,s) = \frac{1}{2}||A(x)s + C(x)||_2^2, \tag{2.12}$$

de modo que

$$\nabla_s M(x,s) = A(x)^T(A(x)s + C(x)). \tag{2.13}$$

2.2 Usando programação quadrática seqüencial

A tentativa de aplicar o método de Newton para encontrar a solução do sistema não linear formado pelas condições de otimalidade definidas a partir do lagrangiano de problemas com restrições de igualdade deu origem a um conjunto de métodos ao qual nos referimos sob a epígrafe de programação quadrática seqüencial.

Naturalmente, o uso deste tipo de método foi estendido para os problemas que contém restrições não lineares, tais como aquele que formulamos em (2.1). Apresentaremos, em linhas gerais e apenas a título de ilustração, as idéias centrais de um algoritmo que figura nesta classe.

Tomando, então, como base a iteração que, como é de praxe, tem índice k, e supondo conhecido x_k, uma aproximação da solução de (2.1), além de uma estimativa dos multiplicadores de Lagrange, λ_k, tentamos obter um novo ponto $x_k + s_k$ resolvendo o modelo quadrático

$$\begin{array}{rl} \textit{minimizar} & q(s) = \dfrac{1}{2}s^T H_k s + \nabla f(x_k)^T s + f(x_k) \\ \textit{sujeita a} & A(x_k)s + C(x_k) = 0 \\ & l \leq x_k + s \leq u \end{array} \tag{2.14}$$

onde H_k é uma aproximação de $\nabla^2_{xx}\ell(x_k, \lambda_k)$, a hessiana do lagrangiano calculada em x_k, λ_k (veja (2.7)).

Resolvido este problema, resta-nos usar algum critério relativamente simples para atualizar as componentes de λ[2].

Infelizmente a convergência deste tipo de algoritmo fica bastante comprometida em pontos distantes da solução ótima, nos quais não se pode garantir que sejam válidas as condições suficientes de otimalidade de segunda ordem, isto é, que a matriz H seja positiva

[2]Este passo tem geralmente menos importância que o anterior e, como veremos, pouco iremos exigir dos valores obtidos para λ.

definida no cone gerado pelas direções factíveis obtidas a partir de x_k, desconsiderado o seu vértice (ou seja, a hipótese de s_k=0).

Para contornar este problema é preciso usar o que chamamos de *estratégia de globalização*. No algoritmo que ora apresentamos isto é feito restringindo em (2.14) a região na qual acreditamos ser válido o modelo quadrático. Introduzindo a exigência que encontremos s_k restrito a uma *região de confiança* definida pela limitação do tamanho deste vetor medido em alguma norma, obtemos o problema

$$\begin{array}{ll} \textit{minimizar} & Q(s) = \frac{1}{2}s^T H_k s + \nabla \ell(x_k)^T s + \ell(x_k) \\ \textit{sujeita a} & A(x_k)s + C(x_k) = 0 \\ & l \leq x_k + s \leq u \\ & ||s|| \leq \delta. \end{array} \tag{2.15}$$

Para medir se o modelo definido por (2.15) reflete razoavelmente o problema real na vizinhança de x_k e se s_k é um "bom" passo, ou seja, se é capaz de diminuir de uma forma que julgamos equilibrada tanto $f(x)$ quanto a infactibilidade (medida pela função $\varphi(x)$, por exemplo), lançamos mão de uma *função de mérito*. No nosso algoritmo é o lagrangiano aumentado, escrito na forma (2.8), que cumpre esse papel. O ponto $x_k + s_k$ só é aceito se a redução prevista pelo modelo para o lagrangiano aumentado for próxima ao decréscimo real obtida entre x_k e $x_k + s_k$. Caso contrário, o valor de δ é considerado alto e a região de confiança é reduzida.

O único inconveniente grave da formulação (2.15) para o problema quadrático é a possibilidade de que, com a restrição sobre a norma do passo, este problema não tenha solução. Uma das formas de contornar isso consiste em dividir s_k em duas componentes. A primeira delas, s_n, é definida numa direção segundo a qual seja reduzida a infactibilidade (ao menos para um valor de δ suficientemente pequeno). Já a segunda, s_t, que tem como função diminuir o lagrangiano, está contida no espaço tangente às restrições. Obtém-se, finalmente, o passo através de $s_k = s_n + s_t$.

2.2.1 Um novo algoritmo

Com base no que acaba de ser exposto, formulou-se, da forma mais abrangente possível, um algoritmo para o qual é possível demonstrar convergência global com base em hipóteses bastante flexíveis (veja [24]), generalizando e ampliando os resultados obtidos por Dennis, El-Alem e Maciel [13] para problemas que só contêm restrições de igualdade.

Nossa intenção é apresentar aqui este método, dispensando-lhe um tratamento cuidadoso como requer o seu ineditismo.

Vamos imaginar, então, que dispomos de uma aproximação $x_0 \in \Omega$ da solução do problema que estamos resolvendo, uma estimativa λ_0 dos multiplicadores de Lagrange, uma matriz H_0 simétrica e um raio δ_0 para a região de confiança. Iremos supor também a existência de um limitante L para a norma de $\Delta\lambda$ e de um outro número real positivo, δ_{min},

usado para impedir que δ seja muito pequeno. Nosso objetivo é obter a cada iteração novas aproximações aceitáveis para x e λ (eventualmente com a atualização de H, θ e δ). Para isso utilizamos o seguinte algoritmo.

Algoritmo **2.1** *Minimização com restrições*

1. *Atribuir valores iniciais:*
   ReduziDelta$\leftarrow$*falso*;
   $k \leftarrow 0$;
   $\theta_{-1} \leftarrow 1$;
2. ENQUANTO não é satisfeito algum critério de parada REPETIR
2.1. SE x_k é factível,
2.1.1. $s_n \leftarrow 0$
2.2. SENÃO
2.2.1. SE (NÃO *ReduziDelta*),
2.2.1.1. *Calcular uma direção de descida com relação às restrições:*
Determinar $d_n(x_k)$ tal que
$d_n^T \nabla \varphi(x_k) < 0$ e
$l \leq x_k + \alpha d_n \leq u$,
para $\alpha > 0$ suficientemente pequeno;
2.2.2. *Calcular o passo de controle do decréscimo da infactibilidade:*
Determinar $\bar{s}_n(x_k, \delta_k)$ que
$$\begin{aligned} \textit{minimiza} \quad & M(x_k, s) \\ \textit{sujeita a} \quad & l \leq x_k + s \leq u, \\ & \|s\| \leq 0.8\delta_k \text{ e} \\ & s = \alpha d_n; \end{aligned}$$
2.2.3. *Calcular o "passo normal":*
Determinar $s_n(x_k, \delta_k)$ tal que
$M(x_k, 0) - M(x_k, s_n) \geq 0.9[M(x_k, 0) - M(x_k, \bar{s}_n)]$,
$l \leq x_k + s_n \leq u$ e $\|s_n\| \leq 0.8\delta_k$;
2.3. SE (NÃO *ReduziDelta*) OU SE x_k é infactível,
2.3.1. *Calcular uma direção de descida com relação ao lagrangiano:*
Tentar determinar $d_t(H_k, x_k, \lambda_k, \delta_k)$ tal que
$d_t^T \nabla Q(H_k, x_k, \lambda_k, s_n) < 0$,
$A(x_k) d_t = 0$ e
$l \leq x_k + s_n + \beta d_t \leq u$ para algum $\beta > 0$;
2.4. SE não foi obtido sucesso no passo acima,
2.4.1. $\bar{s}_t \leftarrow 0$;
2.5. SENÃO

2.5.1. *Calcular o passo de controle do decréscimo do lagrangiano:*
Determinar $\bar{s}_t(H_k, x_k, \lambda_k, \delta_k)$ que
$$\begin{aligned} \textit{minimiza} \quad & Q(s_n + s) \\ \textit{sujeita a} \quad & l \leq x_k + s_n + s \leq u, \\ & ||s_n + s|| < \delta_k \text{ e} \\ & s = \beta d_t; \end{aligned}$$

2.6. *Calcular o "passo tangente":*
Determinar $s_t(H_k, x_k, \lambda_k, \delta_k)$ tal que
$Q(s_n) - Q(s_n + s_t) \geq 0.9[Q(s_n) - Q(s_n + \bar{s}_t)]$,
$A(x_k)s_t = 0$,
$l \leq x_k + s_n + s_t \leq u$ e
$||s_n + s_t|| \leq \delta_k$;

2.7. $s_k \leftarrow s_n + s_t$;

2.8. *Calcular a atualização dos multiplicadores de Lagrange:*
Determinar $\Delta\lambda(H_k, x_k, \lambda_k, \delta_k)$ tal que $||\Delta\lambda|| \leq L$;

2.9. *Calcular a redução prevista da função de mérito:*
$$\begin{aligned} P_{red}^{otm}(H_k, x_k, \lambda_k, s_k, \Delta\lambda) &= Q(H_k, x_k, \lambda_k, 0) - Q(H_k, x_k, \lambda_k, s_k) \\ &\quad -\Delta\lambda^T(A(x_k)s_k + C(x_k)); \\ P_{red}^{fct}(x_k, s_k) &= M(x_k, 0) - M(x_k, s_k); \\ P_{red}(H_k, x_k, \lambda_k, s_k, \Delta\lambda, \theta_k) &= \theta_k P_{red}^{otm}(H_k, x_k, \lambda_k, s_k, \Delta\lambda) \\ &\quad +(1 - \theta_k)P_{red}^{fct}(x_k, s_k); \end{aligned}$$

2.10. *Calcular a redução real da função de mérito:*
$A_{red}(x_k, \lambda_k, s_k, \Delta\lambda, \theta_k) = \mathcal{L}(x_k, \lambda_k, \theta_k) - \mathcal{L}(x_k + s_k, \lambda_k + \Delta\lambda, \theta_k)$.

2.11. SE $P_{red}(H_k, x_k, \lambda_k, s_k, \Delta, \theta_{k-1}) \geq 0.5[M(x, 0) - M(x, s_k)]$,

2.11.1. $\theta_k \leftarrow \theta_{k-1}$;

2.12. SENÃO

2.12.1. *Calcular o parâmetro de penalização:*
Determinar $\theta_k(H_k, x_k, \lambda_k, s_k, \Delta\lambda, \delta_k) \in [0, 1]$ tal que
$P_{red}(H_k, x_k, \lambda_k, s_k, \Delta, \theta_k) \geq 0.5[M(x, 0) - M(x, s_k)]$;

2.13. SE $A_{red}(x_k, \lambda_k, s_k, \Delta\lambda, \theta_k) \geq 0.1P_{red}(H_k, x_k, \lambda_k, s_k, \Delta\lambda, \theta_k)$,

2.13.1. *Aceitar o passo:*
$x_{k+1} \leftarrow x_k + s_k$;
$\lambda_{k+1} \leftarrow \lambda_k + \Delta\lambda$;
Eventualmente, atualizar H e δ;
ReduziDelta$\leftarrow$*falso*;

2.14. SENÃO

2.14.1. *Manter o ponto e reduzir a região de confiança:*
$x_{k+1} \leftarrow x_k$;
$\lambda_{k+1} \leftarrow \lambda + k$;
Escolher δ_{k+1} tal que $0.1\delta_k \leq \delta_{k+1} \leq 0.9\delta_k$;
ReduziDelta$\leftarrow$*verdadeiro*;

2.15 $k \leftarrow k + 1$;

Na descrição feita acima, permitimos que em alguns itens a clareza fosse sacrificada para evitar que o excesso de informação tornasse mais difícil a leitura do algoritmo. A ocasião parece, assim, oportuna para comentarmos que

- Os motivos de parada aos quais o teste do item 2 faz referência incluem, além do caso em que x é uma solução de (2.1), a possibilidade de encontrarmos um ponto φ-estacionário infactível e um ponto factível mas não regular.

- O termo "passo normal", foi cunhado em virtude de ser a direção normal ao núcleo do jacobiano das restrições a escolha natural para d_n no item 2.2.3 acima (veja [13]). Entretanto, isto pode não ter ficado claro à primeira vista já que fizemos exigências bem mais brandas quanto ao referido passo. De forma análoga, mais suaves também são as condições estipuladas para a determinação do "passo tangente" (o passo dado no subespaço tangente às restrições) e do parâmetro de penalização do lagrangiano aumentado. Até mesmo a norma usada na regiões de confiança é arbitrária, o que torna bastante genérico o algoritmo apresentado.

- A variável lógica *ReduziDelta* é usada apenas para evitar que alguns passos sejam repetidos desnecessariamente quando um ponto é recusado e iniciamos nova iteração sem atualizar os valores de x e λ.

Por outro lado, deixamos de citar, também, algumas hipóteses que precisam ser estabelecidas, por exemplo, sobre a seqüência $\{\theta_k\}$ e o vetor s_n, para que tenhamos como assegurar a convergência do algoritmo. Antes de tratar deste assunto, porém, terminaremos esta seção provando ser possível garantir que o algoritmo assim exposto é capaz de, em um número finito de iterações, definir um par de vetores $(s, \Delta\lambda)$ aceito segundo o critério estabelecido no item 2.13. Em outras palavas, mostraremos que, à medida em que é reduzida a região de confiança, o modelo quadrático se aproxima do problema real.

Teorema 2.1 *Se f e C têm primeiras derivadas parciais contínuas e se x_k não atendeu a um dos critérios de parada acima mencionados, então o algoritmo 2.1 encontra, em um número finito de passos, um novo ponto que satisfaz o teste imposto no item 2.13.*

Demonstração. Eliminando, antes de mais nada, os subíndices que identificam a iteração de forma a simplificar a notação, vamos provar este teorema em duas etapas. Na primeira delas suporemos que x não é factível (ou melhor, que não é φ-estacionário). Neste caso, determinada $d_n \neq 0$ (item 2.2.1.1), definimos, para todo $\delta > 0$,

$$\alpha(\delta) = \max\{\alpha > 0 \mid x + \alpha d_n \in \Omega, \text{ e } ||\alpha d_n|| \leq 0.8\delta\},$$

$$v(\delta) = x + \alpha(\delta) d_n, \text{ e}$$

$$\kappa = -\frac{1}{2} d_n^T \nabla\varphi(x)/||d_n|| = -\frac{1}{2} d_n^T \nabla M(x,0)/||d_n||.$$

Obviamente, $\kappa > 0$. Dado que $M(x,s)$ é uma função quadrática, existe $\bar{\delta} > 0$ tal que, para todo $\delta \in (0,\bar{\delta})$,

$$M(x,0) - M(x,\alpha(\delta)d_n) \geq -\frac{1}{2}\alpha(\delta)d_n^T\nabla\varphi(x) = \kappa\alpha(\delta)||d_n|| = 0.8\kappa\delta.$$

Supondo que, na pior de hipóteses, obtivemos $\bar{s}_n = \alpha(\delta)d_n$ no item 2.2.2 do algoritmo, a escolha do passo normal no item 2.2.3 permite que escrevamos

$$M(x,0) - M(x,s_n) \geq 0.72\kappa\delta.$$

E como $A(x)s_t = 0$ (passo 2.6), obtemos

$$M(x,0) - M(x,s) \geq 0.72\kappa\delta.$$

Definido, então, θ segundo o item 2.9.1 ou 2.10.1, é possível garantir que

$$P_{red}(H,x,\lambda,s,\Delta\lambda,\theta) \geq 0.36\kappa\delta. \tag{2.16}$$

Por outro lado, dada a diferenciabilidade de f e C e o fato de $||\Delta\lambda(\delta)||$ ser limitada, usamos as definições de A_{red} e P_{red} para obter

$$\lim_{\delta\to 0}\frac{|A_{red}(\delta) - P_{red}(\delta)|}{\delta} = 0 \tag{2.17}$$

Assim, com base em (2.16) e (2.17) podemos finalmente concluir que

$$\lim_{\delta\to 0}\left|\frac{A_{red}(\delta)}{P_{red}(\delta)} - 1\right| = 0 \tag{2.18}$$

Passando à segunda etapa, analisemos o que acontece quando x é factível. Neste caso, $s_n = 0$ e existe $d_t \neq 0$ que satizfaz os critérios definidos no passo 2.3.1. Desta forma, à semelhança do que foi feito quando o ponto era infactível, escrevemos, para todo $\delta > 0$,

$$\beta(\delta) = \max\{\beta > 0 \mid x + \beta d_t \in \Omega, \text{ e } ||\beta d_t|| \leq \delta\},$$

$$v(\delta) = x + \beta(\delta)d_t, \text{ e}$$

$$\kappa = -\frac{1}{2}d_n^T\nabla\ell(x,\lambda)/||d_t|| = -\frac{1}{2}d_n^T\nabla Q(0)/||d_n||.$$

Observando que $\kappa > 0$, considerando que $Q(s) = Q(H,x,\lambda,s)$ é quadrática, e atendendo ao prescrito nos passos 2.5.1 e 2.6, existe $\bar{\delta} > 0$ tal que para todo $\delta \in (0,\bar{\delta})$ temos

$$Q(0) - Q(s) \geq 0.9\kappa\delta.$$

Como o fato de x ser factível implica $A(x)s = 0$ e $C(x) = 0$, obtemos $M(x,0) = M(x,s) = 0$. Com isso a condição do item 2.11 do algoritmo será sempre satisfeita, de modo que nunca recalcularemos θ nas iterações em que δ é reduzido. Assim, para $\delta \in (0,\bar{\delta})$,

$$P_{red}(H,x,\lambda,s,\Delta\lambda,\theta) = \bar{\theta}[Q(0) - Q(s)] \geq 0.9\bar{\theta}\kappa\delta. \tag{2.19}$$

onde $\bar{\theta}$ é o valor de θ calculado na última iteração em que obtivemos x infactível. Usando agora (2.17) e (2.19) verificamos (2.18) facilmente. □

2.3 Aspectos práticos do algoritmo

Reduzimos ao máximo o detalhamento do algoritmo 2.1 na certeza de que quanto mais abrangente este fosse, tanto maior seria o número de alternativas eficientes para implementá-lo. Entretanto, como já dissemos, para estabelecer resultados teóricos de convergência será necessário restringir em parte a liberdade de escolha que existe em alguns de seus passos.

Assim, com o intuito de tornar clara uma possível formulação prática do algoritmo, apresentamos abaixo maneiras simples e até certo ponto intuitivas de definir aqueles itens que ficaram em aberto.

1. *Definindo a direção d_n:*

   Particularmente no que diz respeito a d_n, a forma mais elementar de, conforme exige o item 2.2.1.1, garantir que esta direção, além de factível, faça um ângulo agudo com $-\nabla\varphi(x_k)$ consiste em escolher

   $$d_n(x) = P(x - \nabla\varphi(x)) - x \tag{2.20}$$

   onde $P(y)$ é a projeção ortogonal de y em Ω.

2. *Calculando o passo normal:*

   Neste caso, uma sugestão seria obter s_n como solução de:

   $$\begin{array}{rl} \textit{minimizar} & \frac{1}{2}||s||_2^2 \\ \textit{sujeita a} & A(x_k)s + C(x_k) = 0 \\ & l \leq x_k + s \leq u \\ & ||s||_\infty \leq 0.8\delta. \end{array} \tag{2.21}$$

   Observamos facilmente que, se s_n resolve este problema quadrático, então certamente satisfaz o critério descrito no item 2.2.3 do algoritmo, tornando inclusive desnecessário o cálculo de d_n e $\bar{s}_n$ (segundo os itens 2.2.1.1 e 2.2.2, respectivamente).

   Quando tratarmos da implementação computacional deste passo, veremos como adaptar o algoritmo QuaCon, apresentado no capítulo anterior, para a determinação de s_n.

3. *Definindo a direção d_t:*

   Uma escolha natural para a direção de descida do lagrangiano, definida no espaço tangente às restrições (como pede o item 2.3.1), é dada por

   $$d_t(x) = P_x(-\nabla Q(s_n)) \tag{2.22}$$

   onde $P_x(y)$ é a projeção ortogonal de y em $\{y \in N(A(x)) \mid l \leq x + s_n + y \leq u\}$[3].

[3]Naturalmente, N(A(x)) representa aqui o núcleo de A(x).

4. *Calculando o passo tangente*

Embora a determinação de s_t segundo o que foi disposto no item 2.6 seja suficiente para demonstrar a convergência do algoritmo, aproveitamos o momento para comentar também como é possível encontrar $s_k = s_n + s_t$ resolvendo um problema quadrático, e evitando o cálculo de d_t e $\bar{s}_t$ (no item 2.5.1). Para tanto, é suficiente que definamos o passo tangente como a solução de:

$$\begin{array}{rl} \textit{minimizar} & \frac{1}{2}s^T H_k s + g(x_k)^T s \\ \textit{sujeita a} & A(x_k)s = A(x_k)s_n \\ & l \leq x_k + s \leq u \\ & ||s||_\infty \leq \delta \end{array} \tag{2.23}$$

onde H_k é uma aproximação da hessiana do lagrangiano. Novamente aqui, a resolução de 2.23 pode ser feita recorrendo ao uso de QuaCon.

5. *Atualizando o termo penalizador da função de mérito*

Um dos aspectos mais interessantes do algoritmo reside no fato da escolha de θ_k não exigir que este seja monotonamente decrescente, como é comumente feito com parâmetros equivalentes em métodos do mesmo gênero.

Contudo, para assegurar que o algoritmo seja capaz de gerar uma seqüência $\{x_k\}$ com pontos de acumulação φ-estacionários precisaremos estipular como condição, na p'roxima seção, que $\{\theta_k\}$ seja convergente. Com o objetivo de satisfazer antecipadamente esta exigência[4] e ao mesmo tempo permitir que não haja um decréscimo monótono de θ, definimos no passo 2.12.1

$$\theta_k^{min} = \min\{1, \theta_0, \ldots, \theta_k - 1\};$$

$$\begin{array}{rcl} \theta_k^{sup} & = & \sup\{\theta \in [0,1] \mid P_{red}(H_k, x_k, \lambda_k, s_k, \Delta\lambda, \theta) \geq 0.5 P_{red}^{fct}(x_k, s_k)\}. \\ & = & 0.5 P_{red}^{fct}(x_k, s_k)/[P_{red}^{fct}(x_k, s_k) - P_{red}^{otm}(H_k, x_k, \lambda_k, s_k, \Delta\lambda)] \end{array}$$

e finalmente

$$\theta_k \leftarrow \min\{\theta_k^{sup}, (1 - \nu/k)\theta_k^{min}\}, \tag{2.24}$$

onde ν é um número escolhido de acordo com o grau de não monotonicidade desejado.

2.4 Resultados teóricos

Mencionamos até agora em quais situações o algoritmo termina e provamos que sempre é possível gerar em um número finito de iterações uma nova aproximação x_k aceitável.

[4] A demonstração da convergência de $\{\theta_k\}$ encontra-se em [24].

Vamos, então, extrair de [24] os resultados que garantem a convergência de toda seqüência infinita de passos gerada pelo algoritmo para um ponto estacionário do problema 2.1.

Comecemos por demonstrar que todo ponto de acumulação de $\{x_k\}$ é φ-estacionário. Para tanto será preciso estabelecer as seguintes hipóteses sobre o problema e o algoritmo:

Hipótese 1 *As seqüências infinitas x_k, λ_k e H_k estão contidas em subconjuntos compactos de $\mathbb{R}^n$, $\mathbb{R}^m$ e $\mathbb{R}^{n\times n}$, respectivamente.*

Hipótese 2 *Para todo $x, y \in \Omega$ temos*

$$||\nabla f(x) - \nabla f(y)||_2 \leq O(|x-y|)$$

e

$$||\nabla C(x) - \nabla C(y)||_2 \leq O(|x-y|).$$

Hipótese 3 *Se $x \in \Omega$ não é φ-estacionário, então d_n é contínua em x.*

Estas condições são pouco restritivas se consideramos o tipo de problema que queremos resolver e a generalidade com que definimos o algoritmo. Alem disso, não é difícil provar que a determinação de d_n segundo (2.20) satisfaz a hipótese 3 acima (vide [24]). Feitas estas ressalvas, pode-se formular o seguinte teorema.

Teorema 2.2 *Se são atendidas as hipóteses supra citadas,e se $\{\theta_k\}$ é convergente, então é φ-estacionário todo ponto de acumulação de $\{x_k\}$, isto é, da seqüência gerada pelo algoritmo 2.1.*

Demonstração. Definimos, primeiro, $\bar{\theta}$ tal que

$$\lim_{k\to\infty} \theta_k = \bar{\theta}$$

Supomos, estão, por absurdo que x^* é um ponto de acumulação de $\{x_k\}$ mas não é φ-estacionário. Do lema 4.3 de [24] temos que, sob as hipóteses já mencionadas, se x^* não é φ-estacionário e se K_1 é um conjunto infinito de índices tal que $\lim_{k\in K_1} x_k = x^*$, então existe $c > 0$ para o qual

$$A_{red}(x_k, \lambda_k, s_k, \Delta\lambda_k, \theta_k) \geq c.$$

Usando este resultado temos, para todo $k \in \mathbb{N}$,

$$\begin{aligned}
\mathcal{L}(x_{k+1}, \lambda_{k+1}, \bar{\theta}) &= \bar{\theta}\ell(x_{k+1}, \lambda_{k+1}) + (1-\bar{\theta})\varphi(x_{k+1}) \\
&= \theta_k\ell(x_{k+1}, \lambda_{k+1}) + (1-\theta_k)\varphi(x_{k+1}) \\
&\quad +(\bar{\theta}-\theta_k)[\ell(x_{k+1}, \lambda_{k+1}) - \varphi(x_{k+1})] \\
&\leq \theta_k\ell(x_k, \lambda_k) + (1-\theta_k)\varphi(x_k) - c \\
&\quad +(\bar{\theta}-\theta_k)[\ell(x_{k+1}, \lambda_{k+1}) - \varphi(x_{k+1})] \\
&= \bar{\theta}\ell(x_k, \lambda_k) + (1-\bar{\theta})\varphi(x_k) + (\theta_k - \bar{\theta})[\ell(x_k, \lambda_k) - \varphi(x_k)] - c \\
&\quad +(\bar{\theta}-\theta_k)[\ell(x_{k+1}, \lambda_{k+1}) - \varphi(x_{k+1})]
\end{aligned}$$

Considerando o fato de $\{\theta_k\}$ ser convergente e ℓ e φ serem limitadas, a seqüência $\bar{\theta}\ell(x_k, \lambda_k) + (1 - \bar{\theta})\varphi(x_k)$ tende a $-\infty$, o que contradiz a hipótese de que ℓ e φ são limitadas. Assim, x^* é um ponto estacionário. □

Posto que os pontos de acumulação de $\{x_k\}$ são φ-estacionários, resta provar que o algoritmo encontra um ponto crítico de (2.1). Para tanto, iremos mais uma vez supor que o problema e o algoritmo atendem a algumas condições adicionais, quais sejam:

Hipótese 4 *Se x é φ-estacionário, então x é factível.*

Hipótese 5 *Todos os pontos de acumulação de $\{x_k\}$ são regulares, o que significa que são linearmente independentes os gradientes das restrições ativas nestes pontos.*

Hipótese 6 $||s_n(x,\delta)|| \leq O(||C(x)||)$, *ou seja, existe um número finito $\kappa > 0$ tal que* $||s_n(x,\delta)|| \leq \kappa||C(x)||$.

Hipótese 7 *s_n é tal que*

$$[M(x,0) - M(x,s_n)] \geq \nu||C(x)||\delta$$

sempre que $\alpha\delta \leq ||C(x)|| \leq \beta$, onde ν, β e α são constantes positivas.

Hipótese 8 *Se x não é φ-estacionário, então é possível definir $\epsilon_1, c_1 > 0$ tais que, para* $||C(x)|| \leq \epsilon_1$, $\delta > 0$,

$$-\nabla Q(s_n(x,\delta))^T d_t(H,x,\lambda,\delta) \geq c_1$$

e $||d_t(H,x,\lambda,\delta)||$, além de ser limitada, não assume o valor zero.

E mais uma vez constatamos que as formas propostas na seção anterior para o cálculo de s_n e s_t são suficientes para garantir que as três últimas hipóteses acima são satisfeitas.

Com base nessas premissas, demonstra-se em [24] que se, nenhum ponto de acumulação de uma seqüência $\{x_k\}$ é um ponto estacionário de (2.1), então obrigatoriamente

$$\lim_{k\to\infty} \theta_k = \bar{\theta} = 0.$$

Por outro lado, também se prova que, mesmo nas iterações em que é preciso reduzir θ, é possível garantir que δ/θ é uniformemente limitado. Como conseqüência da contradição exprimida pelas duas afirmações acima, obtém-se o teorema abaixo:

Teorema 2.3 *Seja $\{x_k\}$ a seqüência gerada pelo algoritmo 2.1. Supondo válidas todas as condições acima estabelecidas, existe um ponto limite de $\{x_k\}$ que é um ponto estacionário de (2.1).*

Demonstração. Veja os detalhes em [24].

2.5 Aspectos computacionais

Estabelecidas todas as condições necessárias para que o algoritmo, além de bem definido, tivesse sua convergência garantida, resta-nos descrever como implantá-lo em um computador.

Assim, apresentamos abaixo, em detalhes, tanto os passos que julgamos mais importantes, como os dados do problema que é preciso fornecer.

1. *determinando o passo normal.*

   Como mencionamos na seção 2.3, uma das formas de encontrar s_n consiste em resolver o problema quadrático (2.21). Em virtude deste problema conter restrições de igualdade, além das canalizações, e de pretendermos utilizar o algoritmo QuaCon, fomos obrigados a embutir na função objetivo um termo de penalização, obtendo assim:

$$\begin{array}{ll} \textit{minimizar} & \frac{\mu_1}{2}\|s\|_2^2 + \frac{1}{2}\|A(x_k)s + c(x_k)\|_2^2 \\ \textit{sujeita a} & l - x_k \leq s \leq u - x_k \\ & \|s\|_\infty \leq 0.8\delta. \end{array} \tag{2.25}$$

   Reescrevendo o problema acima, temos

$$\begin{array}{ll} \textit{minimizar} & \frac{1}{2}s^T[\mu_1 I + A(x_k)^T A(x_k)]s + C(x_k)^T A(x_k)s \\ \textit{sujeita a} & l - x_k \leq s \leq u - x_k \\ & \|s\|_\infty \leq 0.8\delta. \end{array} \tag{2.26}$$

   Não há dificuldade em aplicar QuaCon a (2.26), utilizando $s_n = 0$ como ponto inicial (factível). Mas cabe aqui ressaltar que a escolha de μ_1 é um detalhe complicado desta formulação, devendo ser feita cuidadosamente com base nos dados do problema que pretendemos resolver. Na verdade, em muitos de nossos experimentos observamos que variando μ_1 alterávamos razoavelmente o comportamento do algoritmo.

2. *determinando o passo tangente.*

   Também a determinação de $s_k = s_n + s_t$ (segundo (2.23)) exige a solução de um problema que, por conter restrições de igualdade, precisa ser transformado, através da introdução de um termo penalizado na função objetivo, de modo que possamos aplicar QuaCon. Neste caso escrevemos

$$\begin{array}{ll} \textit{minimizar} & \mu_2[\frac{1}{2}s^T H_k s + g(x_k)^T s] + \frac{1}{2}\|A(x_k)[s - s_n]\|_2^2 \\ \textit{sujeita a} & l - x_k \leq s \leq u - x_k \\ & \|s\|_\infty \leq \delta \end{array} \tag{2.27}$$

ou, de outra forma,

$$\begin{array}{ll} \textit{minimizar} & \frac{1}{2}s^T[\mu_2 H_k + A(x_k)^T A(x_k)]s + [\mu_2 g(x_k)^T - s_n^T A(x_k)^T A(x_k)]s \\ \textit{sujeita a} & l - x_k \leq s \leq u - x_k \\ & ||s||_\infty \leq \delta. \end{array} \tag{2.28}$$

Para este problema, um ponto inicial razoável e factível seria $s_k = s_n$. Além disso, μ_2 deve ser escolhido de forma ainda mais cautelosa que μ_1, na tentativa de manter um equilíbrio entre os objetivos de manter $A(x_k)s_k + C(x_k) = 0$ e reduzir o lagrangiano.

3. *escolhendo H.*

Diversas vezes ao longo do texto fizemos referência à matriz H_k, uma aproximação da hessiana do lagrangiano do problema sem canalizações. Nesta primeira versão do algoritmo, como não estamos demasiadamente preocupados com o esforço dispendido no cálculo de tal matriz, e como dispomos de todas as informações necessárias para definir $\nabla^2_{xx}\ell(x_k, \lambda_k)$, resolvemos adotar como H_k esta matriz.

Quanto à possibilidade de termos que fatorar H_k (para resolver alguns sistemas lineares necessários à determinação de d_t), que pode não ser definida positiva, nada temos com que nos preocupar pois QuaCon possui uma rotina de decomposição de Cholesky modificada, particularmente indicada para estes casos.

4. *definindo os critérios de parada*

A interrupção do algoritmo é determinada sempre que:

(a) O passo for muito pequeno:

$$||x_{k+1} - x_k||_\infty \leq \epsilon_x \max\{1, ||x_{k+1}||_\infty\}; \tag{2.29}$$

(b) A redução da infactibilidade e da função objetivo for pouco significativa:

$$P^{fct}_{red}(x_k, s_k) < \epsilon_f \text{ e } P^{otm}_{red}(H_k, x_k, \lambda_k, s_k, \Delta\lambda) < \epsilon_o. \tag{2.30}$$

5. *atualizando os multiplicadores de Lagrange.*

No passo 2.8, fazemos uma atualização de primeira ordem das estimativas dos multiplicadores de Lagrange diretamente pela emprego da fórmula:

$$\lambda_{k+1} \leftarrow \frac{1}{\mu_2} A(x_k)[s_k - s_n] \tag{2.31}$$

6. *atualizando o do raio da região de confiança*

Toda vez que o passo é aceito, dobramos (em 2.13.1) o raio da região de confiança. Por outro lado, reduzimos (em 2.14.1) δ pela metade se o passo não for aceito. Em qualquer caso, não aceitamos δ menor que 10^{-4}.

7. *Calculando o termo penalizador da função de mérito.*

   No passo 2.12.1, o valor de θ depende de ν, um parâmetro utilizado para permitir diferentes graus de não monotonicidade. Durante os nossos testes, experimentamos diversos valores para ν, na tentativa de verificar qual influência este termo tem sobre o desempenho do método.

8. *detalhando a estrutura de dados*

   Mais que uma aproximação inicial da solução do problema ou dos multiplicadores de Lagrange, para que possamos aplicar o algoritmo é necessário fornecer subrotinas que calculem $f(x)$, $C(x)$, $\nabla f(x)$, $\nabla C(x)$, $\nabla^2 f(x)$ e $\nabla^2 C_i(x), para i = 1, \ldots, m$.

   Na tentativa de fazer com que os dados sejam utilizados de forma eficiente por QuaCon, estabelecemos algumas regras para a criação destas rotinas:

   - $g(x)$ e $C(x)$ devem ser fornecidos em vetores densos;
   - $\nabla C(x)$ deve ser fornecida como um conjunto de n vetores densos ou esparsos, cada um deles contendo a derivada de todas as restrições de igualdade com relação a uma das componentes de x.
   - $\nabla^2 f(x)$ também deve ser fornecida como um conjunto de n vetores densos ou esparsos, compostos das segundas derivadas de f com relação a cada componente de x. Nesta versão do algoritmo, não utilizamos uma aproximação por diferenças finitas para as hessianas.
   - Cada matriz $\nabla^2 C_i(x)$ deve ser fornecida da mesma forma que a hessiana da função objetivo.

   No caso de desejarmos resolver um problema relativamente grande sem usar a decomposição de Cholesky como método de minimização irrestrita em QuaCon, também permitimos que, em lugar do cálculo de $\nabla C(x)$, $\nabla^2 f(x)$ e cada uma das matrizes $\nabla^2 C_i(x)$, sejam criadas apenas rotinas de multiplicação destas matrizes por um vetor qualquer.

2.6 Aplicações e resultados numéricos

Para testar este novo algoritmo, decidimos aplicá-lo à resolução de um subconjunto dos problemas compilados por Hock e Schittkowski em [27].

A descrição completa dos problemas que selecionamos, incluindo o ponto inicial sugerido para cada um deles, encontra-se no apêndice B. Mas apresentamos na tabela 2.1 um resumo das informações que julgamos mais relevantes, indicando, pela ordem, o índice

problema		número de			tipo de função	
aqui	em [27]	var.	restr.	can.	f. objetivo	restrições
1	38	4	0	8	polinomial	-
2	41	4	1	8	polinomial	linear
3	45	5	0	10	polinomial	-
4	53	5	3	10	quadrática	linear
5	54	6	1	12	não polinomial	linear
6	55	6	6	8	não polinomial	linear
7	60	3	1	6	polinomial	polinomial
8	62	3	1	6	não polinomial	linear
9	63	3	2	3	quadrática	quadrática
10	74	6	5	12	polinomial	não polinomial
11	75	6	5	12	polinomial	não polinomial
12	80	5	3	10	não polinomial	polinomial
13	81	5	3	10	não polinomial	polinomial
14	99	7	2	14	não polinomial	não polinomial
15	107	9	6	8	polinomial	não polinomial
16	110	10	0	20	polinomial	-
17	111	10	3	20	não polinomial	não polinomial
18	112	10	3	20	não polinomial	linear
19	113	18	8	8	quadrática	quadrática
20	119	16	8	32	polinomial	linear

Tabela 2.1: Problemas de Hock e Schittkowski

de cada problema, o índice usado em [27], o número de variáveis, restrições de igualdade e canalizações[5], o tipo de função objetivo e o tipo de função usado nas restrições.

A opção por problemas pequenos mas com características variadas revela nossa intenção de fazer apenas uma primeira análise do desempenho do algoritmo quando submetido a testes práticos, sem a preocupação de compará-lo a outros métodos conhecidos.

O algoritmo 2.1 foi implementado (em FORTRAN) conforme as especificações contidas nas seções 2.2.1, 2.3 e 2.5, feitas as seguintes ressalvas:

- Para definir os critérios de parada segundo (2.29) e (2.30), adotamos $\epsilon_x = 10^{-6}$, $\epsilon_f = 10^{-8}$ e $\epsilon_o = 10^{-8}$, salvo no problema 5, em que usamos $\epsilon_x = 10^{-7}$ e $\epsilon_f = 10^{-1}$, e no problema 14, onde empregamos $\epsilon_x = 10^{-10}$ e $\epsilon_f = 10^{-1}$.

- Escolhemos um valor inicial suficientemente grande para δ, com base na magnitude de x_0 e da solução do problema a ser resolvido.

- Na prática, em lugar de usarmos o teste mencionado no item 2.13 do algoritmo, aceitamos um passo sempre que

$$A_{red}(x_k, \lambda_k, s_k, \Delta\lambda, \theta_k) \geq 0.0001 P_{red}(H_k, x_k, \lambda_k, s_k, \Delta\lambda, \theta_k).$$

- Para a minimização dos subproblemas quadráticos definidos em (2.26) e (2.28), necessários à determinação de s_n e s_t, adotamos como critério de parada

$$||g_P(s_k)||_2 < 0.1||g_P(s_0)||_2.$$

- Utilizamos $\mu_1 = 10^{-6}$, exceto no problema 14, para o qual $\mu_1 = 10^{-1}$.

A tabela 2.2 contém os resultados que obtivemos para cada problema, incluindo o número total de iterações nas quais o passo foi aceito, a quantidade de passos recusados e o total de iterações de QuaCon, um número que indica qual critério de parada foi atendido de acordo com (2.29) e (2.30), o valor de μ_2 utilizado e o valor de θ obtido na última iteração.

Em todos os exemplos o algoritmo foi capaz de encontrar uma solução e, em geral, o número de iterações foi pequeno se comparado à dimensão do problema.

Obtivemos os resultados mencionados usando $\nu = 0$ em (2.24), ou seja, exigindo um decréscimo monótono do parâmetro de penalização do lagrangiano aumentado.

Para analisar a influência causada sobre o desempenho do programa por uma estratégia não monótona para a escolha de θ, experimentamos resolver os mesmos problemas atribuindo a ν os valores 10^2, 10^4 e 10^6. Mas apenas nos testes 6, 13 e 14 notamos alguma diferença, conforme mostra a tabela 2.3.

Atribuímos a pouca influência causada pela variação de ν em nossos exemplos ao fato estarmos usando a própria hessiana do lagrangiano, além de boas estimativas para os

[5]Para indicar o número de canalizações consideramos que cada variável pode estar sujeita a um limite inferior e outro superior, podendo ser sendo contada, neste caso, duas vezes.

prob.	n. de iterações			crit.	parâmetros	
	total	recus.	QuaCon	par.	μ_2	θ_{final}
1	31	0	39	1	-	-
2	5	0	12	2	10^{-4}	1.0
3	2	0	2	1	-	-
4	2	0	4	2	10^{-6}	0.13
5	433	0	877	2	10^{3}	1.0
6	2	1	9	1	10^{-5}	0.28
7	7	0	14	1	10^{-6}	0.5
8	5	2	15	1	10^{-10}	10^{-11}
9	5	1	18	2	10^{-6}	0.64
10	9	0	21	1	10^{-6}	0.49
11	148	0	297	1	10^{-10}	10^{-5}
12	7	0	17	2	10^{-6}	0.49
13	20	14	117	2	10^{-4}	0.037
14	12	24	337	1	10^{-15}	10^{-11}
15	6	2	27	1	10^{-10}	10^{-5}
16	4	1	5	1	-	-
17	52	2	390	1	10^{-10}	0.017
18	12	0	33	1	10^{-6}	1.0
19	5	0	15	1	10^{-6}	0.5
20	7	1	35	1	10^{-6}	10^{-6}

Tabela 2.2: Resultados numéricos

prob.	iterações	ν			
		0	10^2	10^4	10^6
6	total	2	3	3	3
	recus.	1	0	0	0
	QuaCon	9	9	9	9
13	total	20	17	72	85
	recus.	14	7	37	38
	QuaCon	117	76	390	466
14	total	12	4	4	4
	recus.	24	20	20	20
	QuaCon	337	219	219	219

Tabela 2.3: Resultados para diversos valores de ν

multiplicadores de Lagrange. Com isso, obtivemos normalmente em cada iteração uma boa redução da infactibilidade e da função objetivo (nos nossos exemplos quase nunca recusamos iterações), de modo que os valores de θ mantêm-se altos, como se constata na tabela 2.6.

Embora seja necessário efetuar muitos experimentos mais para permitir que cheguemos a uma conclusão segura, acreditamos que, nos casos em que contamos com estimativas piores de H e λ, a adoção de uma estratégia não monótona para determinar de θ_k forneça resultados mais significativos.

Por fim, não se pode deixar de mencionar novamente aqui a sensibilidade do algoritmo a variações dos parâmetros μ_1 e μ_2, assunto que também deve ser objeto de uma análise mais demorada.

CAPÍTULO 3

Outro algoritmo para minimização com restrições

Terminando nosso trabalho, voltamos neste capítulo à programação não linear com restrições para tratar de um algoritmo que emprega diretamente, e não apenas como função de mérito, o lagrangiano aumentado.

Se aqui a minimização de quadráticas ainda aparece, isto acontece de forma um tanto sutil, não pela aplicação direta de QuaCon, mas através da combinação de um novo tipo de método com aquele baseado na programacão quadrática seqüencial que acabamos de analisar.

3.1 Usando o lagrangiano aumentado

Consideremos, mais uma vez, o problema de minimização de uma função não linear sujeita a restrições de igualdade e canalizações das variáveis descrito por[1]:

$$\begin{array}{rl} \textit{minimizar} & f(x) \\ \textit{sujeita a} & C(x) = 0 \\ & l \leq x \leq u \end{array} \tag{3.1}$$

onde, como já sabemos, o termo $\leq$ é usado componente a componente e $l < u$.

[1]Reproduzimos aqui a formulação feita no capítulo anterior para tornar menos aborrecida a leitura do texto.

Excluindo de (3.1) as canalizações, o lagrangiano aumentado do problema resultante pode ser definido como a função

$$\mathcal{L}(x, \lambda, \mu) = f(x) + \lambda^T C(x) + \frac{1}{2\mu}||C(x)||_2^2, \tag{3.2}$$

onde o vetor λ contém uma estimativa dos multiplicadores de Lagrange e μ é um parâmetro de penalização[2].

Com a atencão voltada às aplicações de grande porte, Conn, Gould e Toint [10] sugeriram, em artigo recente, um método que consiste em resolver a cada iteração um problema na forma

$$\begin{aligned} &\textit{minimizar} && \mathcal{L}(x, \lambda, \mu) \\ &\textit{sujeita a} && l \leq x \leq u. \end{aligned} \tag{3.3}$$

Nosso objetivo é construir um algoritmo eficiente com base nesta proposta, recorrendo para isso a tudo o que já foi apresentado nos capítulos anteriores.

3.1.1 Definições e notação

Como sempre, à explanação do algoritmo precederemos uma definição da notação adotada, ainda que tenhamos, com isso, que repetir alguns termos já apresentados.

- Adotamos o termo $||\cdot||$ para representar uma norma qualquer em $\mathbb{R}^n$;
- As canalizações são representadas pelo conjunto Ω, definido por:

$$\Omega = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}. \tag{3.4}$$

- O lagrangiano associado ao problema sem canalizações é:

$$\ell(x, \lambda) = f(x) + \lambda^T C(x) \tag{3.5}$$

que derivado com relação a x fornece

$$\nabla \ell(x, \lambda) = \nabla f(x) + \nabla C(x)^T \lambda. \tag{3.6}$$

- As derivadas da função $\mathcal{L}(x, \lambda, \mu)$ com relação às componentes de x são definidas pelo vetor

$$\nabla \mathcal{L}(x, \lambda, \mu) = \nabla f(x) + \nabla C(x)^T [\lambda + \frac{1}{\mu} C(x)]. \tag{3.7}$$

[2] Apesar de equivalente a (3.2), a definição do lagrangiano aumentado feita em (2.8) não será adotada aqui.

- Definindo como estimativa (de primeira ordem) dos multiplicadores de Lagrange associados às restrições de igualdade de (3.1) o termo

$$\bar{\lambda}(x,\lambda,\mu) = \lambda + \frac{1}{\mu}C(x) \tag{3.8}$$

obtemos

$$\nabla\mathcal{L}(x,\lambda,\mu) = \nabla\ell(x,\bar{\lambda}). \tag{3.9}$$

- Para todo ponto $x \in \Omega$, definimos também o vetor $P(x, \nabla\mathcal{L}(x,\lambda))$ como

$$(P(x,\nabla\mathcal{L}(x,\lambda)))_i = \begin{cases} x_i - l_i, & \text{se } x_i - l_i \leq (\nabla\mathcal{L}(x,\lambda))_i; \\ u_i - x_i, & \text{se } u_i - x_i \leq -(\nabla\mathcal{L}(x,\lambda))_i; \\ (\nabla\mathcal{L}(x,\lambda))_i, & \text{caso contrário .} \end{cases} \tag{3.10}$$

3.2 Descrição do algoritmo

Supondo conhecidas uma aproximação inicial do problema (3.1), x_0, uma estimativa dos multiplicadores de Lagrange, λ_0, além dos parâmetros ϵ_0, η_0, ϵ^*, η^* e μ_0, cuja utilidade tornar-se-á clara abaixo, formulou-se, em [10], o seguinte algoritmo[3]:

Algoritmo 3.1 *Minimização com restrições usando o lagrangiano aumentado.*

1. $k \leftarrow 0$;
2. ENQUANTO $||P(x_k, \nabla\mathcal{L}(x_k,\lambda_k))|| \geq \epsilon^*$ OU $||C(x_k)|| \geq \eta^*$ REPETIR
2.1. *Resolver o problema quadrático definido pelo lagrangiano aumentado e pelas canalizações:*
Encontrar $x_k \in \Omega$ tal que
$||P(x_k, \nabla\mathcal{L}(x_k,\lambda_k))|| \leq \epsilon_k$;
2.2. SE $||C(x_k)||_2 \leq \eta_k$,
2.2.1. *Aceitar o passo:*
$\lambda_{k+1} \leftarrow \lambda_k + C(x_k)/\mu$;
$\mu_{k+1} \leftarrow \mu_k$;
$\alpha \leftarrow \min\{\mu_{k+1}, \gamma\}$;
$\epsilon_{k+1} \leftarrow \alpha\epsilon_k$;
$\eta_{k+1} \leftarrow 0.9\eta_k$;
2.3. SENÃO

[3] Como sempre, para reduzir a notação e facilitar o entendimento, na descrição do algoritmo representamos com números alguns termos que são tidos normalmente como parâmetros.

2.3.1. *Recusar o passo:*
$\lambda_{k+1} \leftarrow \lambda_k$;
$\mu_{k+1} \leftarrow \tau \mu_k$;
$\alpha \leftarrow \min\{\mu_{k+1}, \gamma\}$;
$\epsilon_{k+1} \leftarrow \alpha \epsilon_0$;
$\eta_{k+1} \leftarrow 0.1 \eta_0$;

2.4. $k \leftarrow k + 1$.

Embora tenha uma estrutura muito simples, o algoritmo exige, no item 2.1, a resolução aproximada de um problema não linear sujeito a canalizações, o que, como sabemos, encerra grande complexidade. Como a forma pela qual é executado este passo está relacionada mais à eficiência do método que aos resultados teóricos de convergência, voltaremos a tratar deste assunto na seção dedicada aos aspectos computacionais.

Por sua vez, a atualização dos valores estimados dos multiplicadores de Lagrange adotada no passo 2.2.1, tendo por base a fórmula (3.8), não é a única alternativa sugerida por Conn, Gould e Toint, que apresentam em [10] um outro algoritmo no qual λ pode ser calculado de forma bem mais genérica.

3.3 Resultados teóricos

Supondo que $f(x)$ e $C(x)$ têm segundas derivadas contínuas, que a seqüência $\{x_k\}$ está contida em um subconjunto compacto de $\mathbb{R}^n$ e que os pontos de acumulação de $\{x_k\}$ são regulares[4], Conn, Gould e Toint [10] demonstraram que o algoritmo 3.2 satizfaz o seguinte teorema:

Teorema 3.1 *Seja x^* um ponto de acumulação de $\{x_k\}$ gerada pelo algoritmo 3.2, seja também λ^* o vetor de multiplicadores definido pela estimativa de quadrados mínimos[5] para $x = x^*$, e K o conjunto de índices de uma subseqüência infinita de $\{x_k\}$ que converge para x^*. Se são satisfeitas as hipóteses do parágrafo acima, então:*

- *existem constantes $\alpha > 0$, $\beta > 0$ e um inteiro k_0 tais que, para todo $k > k_0$, $k \in K$,*

$$||\lambda_k - \lambda^*|| \leq \alpha ||x_k - x^*||,$$

e

$$||C(x_k)|| \leq \mu_k [\beta \epsilon_k + ||\lambda_k - \lambda^*|| + \alpha ||x_k - x^*||];$$

- *se $C(x^*) = 0$, então x^* é um ponto estacionário de (3.1) e λ^* é o vetor de multiplicadores de Lagrange correspondente. Além disso, $\{\lambda_k\}$ converge para λ^* e $\nabla \mathcal{L}(x_k, \lambda_k)$ converge para $\nabla \ell(x^*, \lambda^*)$.*

[4] Veja hipótese 6 na seção 2.4.

[5] Obtida através da minimização de $||\nabla \bar{C}(x^*)^T \lambda - \nabla f(x^*)||_2^2$, onde $\bar{C}(x^*)$ contém as restrições ativas em x^*.

3.4 Aspectos computacionais

Como já foi dito, o desempenho do algoritmo está subordinado quase que exclusivamente à forma pela qual é implementado o passo 2.1, descrito acima com certa liberalidade.

Para encontrar uma solução aproximada do problema de minimização do lagrangiano aumentado sujeito a canalizações, Conn, Gould e Toint sugerem o uso de um método de minimização em caixas introduzido por eles mesmos recentemente (vide [9], [11]), e já bastante conhecido.

Nada impede, porém, o emprego direto, neste passo, do método de programação quadrática seqüencial que formulamos no capítulo anterior, que pode ser adaptado para ser incluído como subrotina do algoritmo global. Para tanto, a única alteração que precisamos fazer diz respeito à compatibilização dos critérios de parada dos dois métodos.

Assim, para não termos que calcular a norma de $P(x_k, \nabla\mathcal{L}(x_k, \lambda_k))$, adotamos na minimização do lagrangiano aumentado o critério de parada definido no algoritmo 2.1. Em função disso, precisamos modificar também o passo 2 do algoritmo 3.1, aceitando como solução do problema 3.1 um ponto x_k que satisfaz:

$$||x_k - x_{k-1}||_2 \leq \epsilon^* ||x_k||_2$$

e

$$||C(x_k)||_2 \leq \eta^*.$$

Estas alterações não nos obrigam, felizmente, a definir novas fórmulas para o cálculo de ϵ_k e η_k, de modo que podemos manter inalterados os demais itens do algoritmo 3.1.

3.5 Experimentos numéricos

Ainda que este novo método seja mais indicado para problemas grandes, decidimos avaliá-lo preliminarmente utilizando o conjunto de testes apresentado na figura 2.1 (excluindo os problemas 1, 3 e 16 por não possuirem restrições de igualdade).

A tabela 3.1 exibe os resultados alcançados, fornecendo, pela ordem, o índice do problema, o número total de iterações do algoritmo 3.1, o número de iterações em que aumentamos o penalizador do lagrangiano aumentado, a quantidade de iterações aceitas e recusadas ao utilizarmos como subrotina o algoritmo 2.1 e o número de iterações de QuaCon. Indicamos entre parênteses os valores análogos obtidos no capítulo 2, para facilitar uma comparação dos métodos.

No programa em FORTRAN que elaboramos, atribuimos ao parâmetro ϵ^* o mesmo valor adotado para o parâmetro ϵ_x definido em (2.29). Além deste, outros valores utilizados que merecem menção são: $\eta^* = 10^{-6}$, $\epsilon_0 = 1$, $\eta_0 = 1$ e $\mu_0 = 0.1$.

Os resultados obtidos mostram que esta nova proposta é interessante, tendo alcançado a solução em todos os testes feitos, e geralmente em um número razoável de iterações. Na maioria dos exemplos, entretanto, o desempenho foi inferior ao obtido pelo

prob.	número de iterações				
	algoritmo 3.1		algoritmo 2.1		QuaCon
	total	recus.	total	recus.	
2	4	0	5(5)	0(0)	5(12)
4	5	1	10(2)	0(0)	5(4)
5	8	0	440(433)	0(0)	464(877)
6	2	0	5(2)	0(1)	8(9)
7	6	0	15(7)	3(0)	24(14)
8	7	3	22(5)	2(2)	29(15)
9	5	0	20(5)	9(1)	35(18)
10	3	1	32(9)	21(0)	61(21)
11	8	4	245(148)	214(0)	472(297)
12	5	0	21(7)	12(0)	33(17)
13	5	0	13(20)	2(14)	16(117)
14	12	7	1012(12)	1167(24)	4655(337)
15	7	3	23(6)	2(2)	37(27)
17	6	1	1005(52)	996(2)	3446(390)
18	4	1	18(12)	0(0)	22(33)
19	7	2	29(5)	0(0)	33(15)
20	5	2	17(7)	2(1)	23 (35)

Tabela 3.1: Resultados numéricos

algoritmo 2.1, particularmente se observamos o número de iterações de QuaCon exigidas pelos dois métodos, ainda que a quantidade de testes seja insuficiente para que possamos extrair conclusões definitivas.

Além disso, o grande número de iterações efetuadas pelo algoritmo 3.1 em alguns poucos casos sugere a necessidade de estudarmos novos critérios para a redução do parâmetro de penalização incluído na função objetivo.

Mas o que mais nos impressiona é o desperdício que há em utilizar o algoritmo 2.1 para resolver um problema que contém apenas canalizações, em lugar de aproveitá-lo totalmente. Para comentar isso seremos forçados a recorrer a uma última seção ...

3.6 Mesclando os algoritmos

Tendo à mão duas maneiras diferentes de minimizar funções não lineares com restrições, não foi possível resistir à enorme tentação de entremeá-las ainda mais e criar, com isso, um método híbrido que divide as restrições de igualdade e atribui a cada algoritmo o encargo de tornar factível uma parte delas.

Para observar como isso é feito precisamos, em primeiro lugar, reescrever o problema 3.1 na forma

$$\begin{aligned} &\textit{minimizar} && f(x) \\ &\textit{sujeita a} && C(x) = \begin{bmatrix} C_1(x) \\ C_2(x) \end{bmatrix} = 0 \\ & && l \leq x \leq u. \end{aligned} \tag{3.11}$$

Nesta nova formulação, supomos que apenas o conjunto de restrições $C_1(x)$ será incluído no lagrangiano aumentado, de modo que em cada iteração temos que

$$\begin{aligned} &\textit{minimizar} && \bar{\mathcal{L}}(x, \bar{\lambda}, \mu) \\ &\textit{sujeita a} && C_2(x) = 0 \\ & && l \leq x \leq u. \end{aligned} \tag{3.12}$$

onde $\bar{\lambda}$ contém os multiplicadores associados a $C_1(x)$, e

$$\bar{\mathcal{L}}(x, \bar{\lambda}, \mu) = f(x) + \bar{\lambda}^T C_1(x) + \frac{1}{2\mu} ||C_1(x)||_2^2. \tag{3.13}$$

Mesmo contendo as restrições $C_2(x)$, (3.12) pode ser resolvido no item 2.1 do método se usarmos nesse passo o algoritmo 2.1. Esta alteração simples não exige sequer que modifiquemos o programa descrito acima, mas apenas que redefinamos alguns dados de entrada.

Dentre todos os nossos problemas, selecionando para estes últimos testes os únicos que continham dois tipos diferentes de restrições:

prob.	restrições usadas em $C_1(x)$	número de iterações				
		algoritmo 3.1		algoritmo 2.1		QuaCon
		total	recus.	total	recus.	
9	quadrática	7	3	21	1	47
	linear	7	3	19	6	67
10	lineares	6	3	20	10	37
	polinomiais	10	5	18	0	41
11	lineares	6	3	13	22	95
	polinomiais	10	5	21	2	56
19	lineares	9	4	27	1	60
	quadráticas	7	3	11	0	27

Tabela 3.2: Misturando os métodos

- O problema 9, com uma restrição quadrática e uma linear;
- O problemas 10 e 11, que possuem duas restrições lineares e 3 polinomiais;
- O problema 19, composto de três restrições lineares e cinco quadráticas.

Como existem duas formas de dividir as restrições em dois conjuntos, resolvemos cada problema ora fazendo $C_1(x)$ conter as restrições lineares, ora as demais. A tabela 3.2 resume as informações colhidas nessas experiências.

Observando estes resultados, e comparando-os aos valores das tabelas 2.2 e 3.1, mais uma vez somos levados a crer que ambas as alternativas são promissoras e devem ser melhor exploradas. E mesmo sendo muito pequeno o número e o tamanho dos testes feitos, tudo indica que a divisão das restrições entre os dois algoritmos melhora o desempenho do método que usa o lagrangiano aumentado.

Os resutados também sugerem ser melhor embutir as restrições mais complicadas (não lineares) na função objetivo (em lugar das restrições lineares).

Além disso, cabe ressaltar que como o algoritmo 2.1 pouco mais foi que formulado, do seu desenvolvimento ainda podemos esperar obter indiretamente algum progresso aqui.

Conclusões

" ... como quereis vós que não me encha
de confusão o antigo legislador, chamado Vulgo,
quando ele vir que no cabo de tantos anos,
como há que durmo no silêncio do esquecimento,
me saio agora, tendo já tão grande carga
de anos às costas, com uma legenda
seca como as palhas, falta de invenção,
minguada de estilo, pobre de conceitos,
e alheia a toda erudição e doutrina..."
Miguel de Cervantes
in: Dom Quixote de La Mancha

Chegando a este ponto do trabalho em que o damos por concluído, longe ainda estamos de imaginar que nada mais haja por fazer. Pelo contrário, julgamos sim que a utilidade deste relato de nossas experiências reside justamente no fato de termos dado um primeiro passo.

Empregamos um algoritmo novo para minimização de quadráticas em caixas na resolução de aplicações práticas e o resultado foi muitíssimo promissor, como também demonstraram grande potencial os algoritmos apresentados para problemas não lineares com restrições.

Mas dos resultados obtidos não podemos extrair mais que essa promessa vaga, pois muito ainda resta por detalhar, definir, alterar até que venhamos a ter finalmente como

assegurar aos métodos propostos eficiência ou competitividade.

Cumpre-nos, assim, antes de bater o ponto final, apenas o dever de utilizar parte do que aprendemos para sugerir temas de futuras linhas de pesquisa nesta área, incluindo:

- A comparação de todos os algoritmos com programas do mesmo gênero já bem estabelecidos, visando principalmente aplicá-los a problemas de grande porte;
- A elaboração e avaliação de uma rotina de atualização da decomposição de cholesky em QuaCon;
- Uma análise mais ampla do desempenho dos métodos do tipo gradiente com retardo, particularmente no que diz respeito à programação em paralelo;
- A discussão de alternativas para a resolução dos problemas de programação quadrática com restrições contidos no algoritmo 2.1, sem que seja usado, como fizemos, um método baseado em penalização;
- A definição de novas propostas para o cálculo de alguns passos no algoritmo 2.1, atendendo melhor ao espírito menos restritivo do método e reduzindo o custo computacional de cada iteração.
- A especialização de um dos dois métodos mesclados no capítulo 3 para o tratamento exclusivo de algum tipo particular de restrição (as lineares, por exemplo), visando uma melhoria do desempenho global do algoritmo.

APÊNDICE A

Experimentos do primeiro capítulo

Reunimos neste apêndice todos os resultados dos experimentos realizados com o algoritmo QuaCon. Os dados aparecem agrupados em tabelas segundo o tipo de problema a que se referem.

Inicialmente, investigamos qual o comportamento de QuaCon quando alteramos a dimensão da face em que utilizamos a decomposição de Cholesky para a minimização unidimensional.

Dedicamo-nos, em seguida ao estudo dos desempenho dos diversos métodos de gradiente com retardo. Nas tabelas correspondentes, os resultados são fornecidos em função do parâmetro θ (vide item 1.2 à página 8), que define a freqüência com que mudamos de face.

Passamos, então, à analise dos efeitos decorrentes do uso de vários valores para o termo δ (vide item 1.2.2 à página 8), que tem por finalidade assegurar a convergência do algoritmo quando resolvemos problemas dual degenerados.

Finalmente, apresentamos alguns gráficos que ilustram o resultado da aplicação do programa aos problemas de projeção de pontos sobre politopos.

Muitos dos termos que aparecem nas tabelas, como θ, $na(x^*)$, $na(x_0)$, deg e δ, já foram mencionados e supomos conhecidos seus significados. Além disso, em alguns casos, usamos expressões como $x_0 = 0$ ou $x_0 = U$, por exemplo, para indicar que todas as componentes da aproximação inicial utilizada são iguais a zero ou seus limites superiores.

Nas tabelas A17, A18, A19, A20, A29 e A30, os valores fornecidos representam o número total de iterações, o número de iterações com decréscimo da função e o índice da iteração na qual foi encontrada a face ótima.

Teta	Dim.	Problema							
	Chol	1	2	3	4	5	6	7	8
10^{-10}	0	127	250	115	288	202	259	229	240
	100	127	250	115	288	202	259	229	240
	200	127	250	115	288	202	259	229	240
	500	88	250	80	235	202	259	229	240
	1000	21	134	17	165	202	259	229	240
	2601	21	21	17	18	19	20	22	23
10^{-5}	0	127	250	117	288	202	259	229	240
	100	127	250	117	288	202	259	229	240
	200	127	250	117	288	202	259	229	240
	500	88	250	80	235	202	259	229	240
	1000	21	134	17	165	202	259	229	240
	2601	21	21	17	18	19	20	22	23
0.01	0	88	212	78	254	153	211	152	229
	100	88	212	78	254	153	211	152	229
	200	88	212	78	254	153	211	152	229
	500	89	212	76	234	153	211	152	229
	1000	20	122	17	170	153	211	152	229
	2601	20	19	17	18	19	20	20	21
0.1	0	90	128	65	279	116	182	134	218
	100	76	128	65	279	116	182	134	218
	200	76	128	65	279	116	182	134	218
	500	73	128	58	279	116	182	134	218
	1000	17	68	15	210	116	182	134	218
	2601	17	17	15	15	17	18	18	19
0.9	0	75	146	64	175	116	217	137	274
	100	75	146	64	175	116	217	137	274
	200	73	146	63	175	116	217	137	274
	500	65	146	55	176	116	217	137	274
	1000	21	90	18	93	116	217	137	274
	2601	21	20	18	18	18	19	26	27

Tabela A.1: Problemas de obstáculo A (n=2601)

Teta	Dim.	Problema							
	Chol	9	10	11	12	13	14	15	16
10^{-10}	0	200	314	211	410	330	413	401	406
	100	200	314	211	410	330	413	401	406
	200	200	314	211	410	330	413	401	406
	500	200	314	211	410	330	413	401	406
	1000	153	269	150	410	330	413	401	406
	2000	32	116	25	249	330	413	401	406
	5041	32	27	25	24	24	25	31	32
10^{-5}	0	200	313	211	437	330	413	401	406
	100	200	313	211	437	330	413	401	406
	200	200	313	211	437	330	413	401	406
	500	200	313	211	437	330	413	401	406
	1000	153	269	150	369	330	413	401	406
	2000	32	116	25	257	330	413	401	406
	5041	32	27	25	24	24	25	31	32
0.01	0	148	328	123	364	262	356	286	392
	100	148	328	123	364	262	356	286	392
	200	148	328	123	364	262	356	286	392
	500	148	328	123	364	262	356	286	392
	1000	133	328	110	364	262	356	286	392
	2000	30	206	21	253	262	356	286	392
	5041	30	24	21	22	22	23	28	29
0.1	0	119	215	75	499	168	279	165	362
	100	119	215	75	499	168	279	165	362
	200	119	215	75	499	168	279	165	362
	500	116	215	75	499	168	279	165	362
	1000	106	209	76	499	168	279	165	362
	2000	24	135	20	359	168	279	165	362
	5041	24	22	20	19	21	22	26	27
0.9	0	120	171	96	478	166	335	219	388
	100	120	171	96	478	166	335	219	388
	200	119	171	96	478	166	335	219	388
	500	114	176	94	478	166	335	219	388
	1000	98	160	72	478	166	335	219	388
	2000	30	89	25	281	166	335	219	388
	5041	30	26	25	23	22	23	34	35

Tabela A.2: Problemas de obstáculo A (n=5041)

Teta	Dim.	Problema							
	Chol	17	18	19	20	21	22	23	24
10^{-10}	0	428	1226	408	1851	813	765	742	850
	100	428	1226	408	1851	813	765	742	850
	200	428	1226	408	1851	813	765	742	850
	500	428	1226	408	1851	813	765	742	850
	1000	428	1226	408	1851	813	765	742	850
	2000	255	1226	259	1851	813	765	742	850
	5000	41	659	32	1390	813	765	742	850
	10000	41	35	32	28	32	29	37	24
10^{-5}	0	420	1224	406	1847	812	765	741	850
	100	420	1224	406	1847	812	765	741	850
	200	420	1224	406	1847	812	765	741	850
	500	420	1224	406	1847	812	765	741	850
	1000	420	1224	406	1847	812	765	741	850
	2000	254	1224	258	1847	812	765	741	850
	5000	41	658	32	1388	812	765	741	850
	10000	41	35	32	28	32	29	37	24
0.01	0	220	676	181	1046	430	487	499	604
	100	220	676	181	1046	430	487	499	604
	200	220	676	181	1046	430	487	499	604
	500	220	676	181	1046	430	487	499	604
	1000	218	676	181	1046	430	487	499	604
	2000	221	676	187	1046	430	487	499	604
	5000	38	347	27	672	430	487	499	604
	10000	38	33	27	24	28	26	34	21
0.1	0	154	351	146	732	355	552	404	680
	100	154	351	146	732	355	552	404	680
	200	154	351	146	732	355	552	404	680
	500	154	351	146	732	355	552	404	680
	1000	188	351	133	708	355	552	404	680
	2000	163	351	155	675	355	552	404	680
	5000	33	156	27	597	355	552	404	680
	10000	33	27	27	22	26	23	35	18
0.9	0	171	275	158	584	230	472	314	649
	100	171	275	158	584	230	472	314	649
	200	171	275	158	584	230	472	314	649
	500	162	275	153	584	230	472	314	649
	1000	165	273	146	584	230	472	314	649
	2000	150	248	114	584	230	472	314	649
	5000	40	147	36	287	230	472	314	649
	10000	40	35	36	28	28	25	48	18

Tabela A.3: Problemas de obstáculo A (n=10000)

Teta	Dim.	Problema B			Problema C		
	Chol	25	26	27	28	29	30
10^{-10}	0	234	454	490	227	276	187
	100	234	454	490	227	276	187
	200	234	454	490	227	276	187
	500	234	454	490	227	276	187
	1000	234	454	490	227	276	187
	2000	234	454	490	227	276	187
	5041	24	28	29	22	27	21
10^{-5}	0	261	452	490	227	276	187
	100	261	452	490	227	276	187
	200	261	452	490	227	276	187
	500	261	452	490	227	276	187
	1000	261	452	490	227	276	187
	2000	261	452	490	227	276	187
	5041	24	28	29	22	27	21
0.01	0	170	281	289	154	204	184
	100	170	281	289	154	204	184
	200	170	281	289	154	204	184
	500	170	281	289	154	204	184
	1000	170	281	289	154	204	184
	2000	170	281	289	154	204	184
	5041	21	24	26	21	25	20
0.1	0	143	190	222	142	181	184
	100	143	190	222	142	181	184
	200	143	190	222	142	181	184
	500	143	190	222	142	181	184
	1000	143	190	222	142	181	184
	2000	143	190	222	142	181	184
	5041	18	22	23	18	22	17
0.9	0	113	166	223	128	191	195
	100	113	166	223	128	191	195
	200	113	166	223	128	191	195
	500	113	166	223	128	191	195
	1000	113	166	223	128	191	195
	2000	113	163	223	128	191	195
	5041	18	28	25	23	29	22

Tabela A.4: Problemas de obstáculo B e C (n=5041)

Teta	Dim.	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
	Chol	deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	0	620	722	543	630	551	769
	100	355	563	318	626	363	803
	200	355	563	318	626	363	803
	500	355	563	318	626	363	803
	1000	9	21	22	14	15	24
10^{-5}	0	604	639	543	630	551	769
	100	355	618	318	672	363	908
	200	355	618	318	672	363	908
	500	355	618	318	672	363	908
	1000	9	21	22	14	15	24
0.01	0	483	551	482	536	479	652
	100	299	485	299	686	328	604
	200	299	485	299	686	328	604
	500	299	485	299	686	328	604
	1000	8	19	14	12	14	24
0.1	0	506	510	494	520	481	548
	100	285	453	298	498	315	454
	200	285	453	298	498	315	454
	500	285	453	298	498	315	454
	1000	8	18	14	12	11	21
0.9	0	457	508	472	496	521	538
	100	283	329	307	309	316	306
	200	283	329	307	309	316	306
	500	283	329	307	309	316	306
	1000	11	9	8	7	10	12

Tabela A.5: Problemas aleatórios ($na(x^*) = 100$)

Teta	Dim.	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
	Chol	deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	0	547	*	580	*	569	*
	100	375	1479	408	3046	396	4218
	200	375	1479	408	3046	396	4218
	500	159	1341	223	3046	119	4218
	1000	20	33	14	48	20	44
10^{-5}	0	547	1878	598	*	569	*
	100	375	1359	428	7173	396	4180
	200	375	1359	428	7173	396	4180
	500	159	1359	215	7173	119	4180
	1000	20	33	14	47	20	44
0.01	0	552	1239	542	1014	537	1253
	100	382	661	372	663	370	651
	200	382	661	372	663	370	651
	500	142	661	140	663	141	651
	1000	18	38	12	37	18	33
0.1	0	505	834	527	851	522	883
	100	336	609	361	507	355	547
	200	336	609	361	507	355	547
	500	118	609	126	450	104	547
	1000	11	29	10	30	11	33
0.9	0	461	556	477	591	498	539
	100	294	377	310	385	329	371
	200	294	377	310	385	329	371
	500	69	205	94	299	101	259
	1000	7	8	13	17	15	20

* excedido o limite de iterações

Tabela A.6: Problemas aleatórios ($na(x^*) = 500$)

Teta	Dim.	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
	Chol	deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	0	478	1790	471	*	523	*
	100	328	1211	325	3589	372	9458
	200	172	1110	172	3326	154	9458
	500	104	406	49	441	61	4882
	1000	20	38	19	55	19	51
10^{-5}	0	478	1711	465	*	523	*
	100	328	1029	323	2755	372	9155
	200	172	1029	170	2755	154	9155
	500	104	403	47	428	61	4792
	1000	20	38	19	55	19	51
0.01	0	403	963	451	1371	454	1035
	100	259	554	304	807	309	585
	200	82	554	147	763	97	585
	500	61	190	40	257	36	159
	1000	24	56	16	50	17	41
0.1	0	399	844	430	976	422	983
	100	254	452	286	498	276	625
	200	80	337	84	384	68	419
	500	60	95	44	98	35	123
	1000	22	39	19	43	16	34
0.9	0	380	635	393	569	435	529
	100	237	429	250	419	300	397
	200	54	182	67	141	80	159
	500	37	34	37	37	33	42
	1000	10	16	11	15	18	16

* excedido o limite de iterações

Tabela A.7: Problemas aleatórios ($na(x^*) = 900$)

$na(x^*)$	$na(x_0)$	dimensão em que usamos Cholesky				
		0	100	200	500	1000
100	100	3770	3770	3770	3770	23
	500	4014	4014	4014	4014	26
	900	4699	4699	4699	4699	19
500	100	9943	9943	9943	3846	38
	500	8901	8901	8901	3676	47
	900	9597	9597	9597	3086	69
900	100	6777	6777	1724	208	42
	500	7296	7296	1439	236	117
	900	6809	6809	1042	267	54

Tabela A.8: Problemas aleatórios (cond=10^5, $\theta = 0.1$, deg=1)

Prob.	x_0	Dim.Chol.	
		0	100
1	0	40	7
	1	92	8
2	0	57	9
	1	84	10
3	0	51	10
	1	110	11
4	0	81	9
	1	130	10
5	0	54	7
	1	128	8

Tabela A.9: Problemas de projeção A

Prob.	x_0	Dim.Chol.	
		0	100
1	0	8717	12
	1	11342	17
2	0	6512	27
	1	7517	28
3	0	5021	24
	1	5053	25
4	0	*	47
	1	*	1
5	0	5132	14
	1	4906	15

*excedidas 32000 iterações

Tabela A.10: Problemas de projeção B

Teta	Método	Problema							
		1	2	3	4	5	6	7	8
10^{-10}	gradientes conjugados	127	250	115	288	202	259	229	240
	retardo maximo (5)	175	257	102	297	274	297	275	395
	retardo maximo (10)	248	433	155	367	423	391	550	490
	barzilai-borwein	177	199	128	203	240	245	330	260
	retardo aleatorio (5)	208	212	170	357	346	313	277	325
	retardo aleatorio (10)	202	283	178	323	312	292	292	329
	retardo max-minimo (5)	148	185	102	224	273	261	214	293
	retardo max-minimo (10)	166	217	113	338	244	358	222	298
	retardo aleat. exc. k (5)	164	244	134	285	303	301	228	317
	retardo aleat. exc. k (10)	167	224	124	364	306	327	248	336
10^{-5}	gradientes conjugados	127	250	117	288	202	259	229	240
	retardo maximo (5)	175	257	102	297	274	297	275	395
	retardo maximo (10)	248	433	155	367	423	374	550	490
	barzilai-borwein	177	199	128	203	240	245	330	260
	retardo aleatorio (5)	201	212	170	357	333	339	277	325
	retardo aleatorio (10)	190	283	178	323	353	292	292	329
	retardo max-minimo (5)	148	185	102	296	273	276	214	293
	retardo max-minimo (10)	166	217	113	358	244	358	222	298
	retardo aleat. exc. k (5)	164	244	134	285	303	301	228	317
	retardo aleat. exc. k (10)	167	224	124	364	306	327	248	374
0.01	gradientes conjugados	88	212	78	254	153	211	152	229
	retardo maximo (5)	119	209	115	264	205	316	223	308
	retardo maximo (10)	117	353	134	363	370	362	511	605
	barzilai-borwein	129	162	115	204	252	253	173	239
	retardo aleatorio (5)	154	189	132	238	254	285	201	255
	retardo aleatorio (10)	157	282	135	283	284	285	298	321
	retardo max-minimo (5)	119	160	93	235	200	296	210	248
	retardo max-minimo (10)	137	235	94	215	203	224	230	311
	retardo aleat. exc. k (5)	120	207	101	202	256	264	214	293
	retardo aleat. exc. k (10)	109	188	103	242	227	295	202	272

Tabela A.11: Problemas de obstáculo A (n=2601)

Teta	Método	Problema							
		1	2	3	4	5	6	7	8
0.1	gradientes conjugados	90	128	65	279	116	182	134	218
	retardo maximo (5)	150	212	91	249	180	302	233	293
	retardo maximo (10)	219	372	153	261	434	543	559	494
	barzilai-borwein	88	170	88	201	143	232	201	216
	retardo aleatorio (5)	96	189	107	199	199	262	211	285
	retardo aleatorio (10)	145	174	106	196	213	260	231	281
	retardo max-minimo (5)	129	144	131	273	262	278	214	237
	retardo max-minimo (10)	165	178	111	257	242	238	310	274
	retardo aleat. exc. k (5)	112	191	98	227	216	257	184	341
	retardo aleat. exc. k (10)	137	151	106	257	209	267	227	277
0.9	gradientes conjugados	75	146	64	175	116	217	137	274
	retardo maximo (5)	165	218	122	257	275	292	290	322
	retardo maximo (10)	223	590	185	359	415	519	453	530
	barzilai-borwein	118	178	104	207	185	206	235	270
	retardo aleatorio (5)	155	224	130	265	204	355	232	412
	retardo aleatorio (10)	164	229	169	300	269	277	256	392
	retardo max-minimo (5)	146	219	100	251	244	290	268	352
	retardo max-minimo (10)	148	204	138	233	245	276	251	290
	retardo aleat. exc. k (5)	145	197	111	234	230	243	202	313
	retardo aleat. exc. k (10)	183	247	129	209	260	389	257	300

Tabela A.12: Problemas de obstáculo A (n=2601)

Teta	Método	Problema							
		9	10	11	12	13	14	15	16
10^{-10}	gradientes conjugados	200	314	211	410	330	413	401	406
	retardo maximo (5)	273	447	300	416	398	405	491	451
	retardo maximo (10)	319	401	294	774	501	1103	838	1004
	barzilai-borwein	294	355	294	590	434	416	425	410
	retardo aleatorio (5)	245	429	341	907	620	517	549	423
	retardo aleatorio (10)	329	453	338	373	607	507	565	409
	retardo max-minimo (5)	242	331	303	272	431	449	529	417
	retardo max-minimo (10)	251	351	290	397	429	471	471	505
	retardo aleat. exc. k (5)	241	315	255	616	466	529	586	551
	retardo aleat. exc. k (10)	284	265	279	380	505	550	473	536
10^{-5}	gradientes conjugados	200	313	211	437	330	413	401	406
	retardo maximo (5)	284	393	300	416	398	405	491	451
	retardo maximo (10)	318	401	294	687	501	1103	838	1004
	barzilai-borwein	274	355	294	590	434	417	415	417
	retardo aleatorio (5)	345	429	341	907	627	482	549	423
	retardo aleatorio (10)	329	453	338	373	604	507	565	409
	retardo max-minimo (5)	245	325	303	272	431	449	529	417
	retardo max-minimo (10)	256	456	290	397	429	471	471	505
	retardo aleat. exc. k (5)	241	338	255	577	466	529	586	551
	retardo aleat. exc. k (10)	262	265	279	380	505	550	473	536
0.01	gradientes conjugados	148	328	123	364	262	356	286	392
	retardo maximo (5)	192	400	169	330	350	393	320	483
	retardo maximo (10)	211	595	249	734	536	1246	406	1237
	barzilai-borwein	195	268	156	301	290	348	293	341
	retardo aleatorio (5)	204	268	180	320	393	554	385	415
	retardo aleatorio (10)	202	296	194	291	347	363	376	394
	retardo max-minimo (5)	159	277	158	329	306	306	290	358
	retardo max-minimo (10)	196	302	154	318	382	394	360	442
	retardo aleat. exc. k (5)	187	230	151	351	284	425	362	438
	retardo aleat. exc. k (10)	173	276	172	308	274	477	354	428

Tabela A.13: Problemas de obstáculo A (n=5041)

Teta	Método	Problema							
		9	10	11	12	13	14	15	16
0.1	gradientes conjugados	119	215	75	499	168	279	165	362
	retardo maximo (5)	194	298	135	358	300	415	380	459
	retardo maximo (10)	246	688	303	615	1090	1047	787	1593
	barzilai-borwein	149	215	120	308	220	306	255	460
	retardo aleatorio (5)	162	276	128	335	286	451	319	393
	retardo aleatorio (10)	179	260	147	326	264	430	320	413
	retardo max-minimo (5)	169	251	122	287	253	376	262	432
	retardo max-minimo (10)	261	251	184	371	341	396	400	444
	retardo aleat. exc. k (5)	161	279	136	334	262	413	374	386
	retardo aleat. exc. k (10)	158	277	116	324	259	349	310	491
0.9	gradientes conjugados	120	171	96	478	166	335	219	388
	retardo maximo (5)	292	303	202	374	341	443	462	553
	retardo maximo (10)	422	518	252	1067	483	1231	794	1056
	barzilai-borwein	173	211	138	289	249	265	310	386
	retardo aleatorio (5)	228	282	171	316	302	399	368	410
	retardo aleatorio (10)	230	270	198	345	334	441	448	501
	retardo max-minimo (5)	198	304	143	313	350	411	370	485
	retardo max-minimo (10)	264	338	192		356	386	379	464
	retardo aleat. exc. k (5)	235	296	177		346	411	416	393
	retardo aleat. exc. k (10)	247	301	195		319	448	384	468

Tabela A.14: Problemas de obstáculo A (n=5041)

Teta	Método	Problema							
		17	18	19	20	21	22	23	24
10^{-10}	gradientes conjugados	428	1226	408	1851	813	765	742	850
	retardo maximo (5)	430	625	457	711	726	1006	815	798
	retardo maximo (10)	620	1195	479	637	874	1231	1603	1566
	barzilai-borwein	502	572	470	672	785	935	891	577
	retardo aleatorio (5)	642	754	550	1004	931	713	1098	889
	retardo aleatorio (10)	707	467	539	673	923	672	1056	1002
	retardo max-minimo (5)	440	455	450	786	813	726	861	694
	retardo max-minimo (10)	460	505	456	870	749	809	849	573
	retardo aleat. exc. k (5)	445	599	464	925	760	726	803	665
	retardo aleat. exc. k (10)	453	603	473	846	775	831	960	848
10^{-5}	gradientes conjugados	420	1224	406	1847	812	765	741	850
	retardo maximo (5)	467	581	455	694	757	963	917	798
	retardo maximo (10)	480	1064	477	819	1009	1683	1491	1566
	barzilai-borwein	454	572	490	672	804	950	938	577
	retardo aleatorio (5)	616	718	535	814	901	667	1068	798
	retardo aleatorio (10)	596	467	537	663	872	696	1134	992
	retardo max-minimo (5)	435	455	448	802	786	702	906	600
	retardo max-minimo (10)	424	508	454	830	745	718	825	678
	retardo aleat. exc. k (5)	459	640	462	961	771	726	841	665
	retardo aleat. exc. k (10)	432	621	478	727	742	883	841	848
0.01	gradientes conjugados	220	676	181	1046	430	487	499	604
	retardo maximo (5)	326	512	197	527	370	607	490	619
	retardo maximo (10)	302	1216	204	1279	926	1681	962	1471
	barzilai-borwein	276	410	275	530	330	530	438	533
	retardo aleatorio (5)	297	369	249	479	551	635	501	603
	retardo aleatorio (10)	305	491	296	505	474	535	580	621
	retardo max-minimo (5)	290	391	224	487	512	633	585	672
	retardo max-minimo (10)	370	438	325	468	490	578	684	564
	retardo aleat. exc. k (5)	325	483	199	534	399	512	491	751
	retardo aleat. exc. k (10)	297	511	206	584	546	729	448	619

Tabela A.15: Problemas de obstáculo A (n=10000)

Teta	Método	Problema							
		17	18	19	20	21	22	23	24
0.1	gradientes conjugados	154	351	146	732	355	552	404	680
	retardo maximo (5)	302	422	205	584	512	723	594	607
	retardo maximo (10)	684	753	536	1049	1165	1928	1516	1918
	barzilai-borwein	254	313	168	398	407	534	367	633
	retardo aleatorio (5)	236	348	217	384	374	570	429	578
	retardo aleatorio (10)	235	405	225	450	416	502	518	479
	retardo max-minimo (5)	323	356	210	385	512	646	489	509
	retardo max-minimo (10)	386	346	248	539	482	636	592	681
	retardo aleat. exc. k (5)	253	323	201	504	464	526	444	577
	retardo aleat. exc. k (10)	266	411	216	548	446	531	571	878
0.9	gradientes conjugados	171	275	158	584	230	472	314	649
	retardo maximo (5)	436	580	356	887	596	779	661	704
	retardo maximo (10)	618	757	473	1430	911	1621	1234	-
	barzilai-borwein	288	378	231	400	373	419	530	572
	retardo aleatorio (5)	315	380	268	555	450	528	609	721
	retardo aleatorio (10)	374	400	270	498	464	537	645	794
	retardo max-minimo (5)	348	426	250	558	594	580	747	841
	retardo max-minimo (10)	399	421	340	579	587	577	651	697
	retardo aleat. exc. k (5)	319	347	255	504	507	464	480	609
	retardo aleat. exc. k (10)	332	408	267	553	490	692	620	816

Tabela A.16: Problemas de obstáculo A (n=10000)

teta	método	$x_0 = (L+U)/2$			$x_0 = U$			$x_0 = L$		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
10^{-10}	grad. conjugados	234	234	225	454	454	437	490	490	478
	ret. maximo (5)	211	181	172	512	475	462	482	412	401
	ret. maximo (10)	315	228	187	542	452	429	830	499	455
	barzilai-borwein	241	211	179	611	583	541	482	452	425
	ret. aleatorio (5)	321	305	290	558	540	517	500	455	435
	ret. aleatorio (10)	357	343	330	612	578	561	511	448	426
	ret. max-min (5)	219	208	174	533	514	484	510	502	486
	ret. max-min (10)	260	253	197	533	524	499	502	490	438
	r. aleat. exc. k (5)	218	178	161	474	448	425	481	451	438
	r. aleat. exc. k (10)	226	203	174	493	442	419	501	433	416
10^{-5}	grad. conjugados	261	261	240	452	452	435	490	490	478
	ret. maximo (5)	285	199	172	525	464	456	482	412	401
	ret. maximo (10)	492	216	183	657	490	464	830	499	455
	barzilai-borwein	242	215	184	536	517	497	482	452	425
	ret. aleatorio (5)	357	307	284	573	530	512	500	455	435
	ret. aleatorio (10)	305	289	285	546	535	501	511	448	426
	ret. max-min (5)	247	230	193	561	534	508	510	502	486
	ret. max-min (10)	273	267	201	560	551	510	502	490	438
	r. aleat. exc. k (5)	201	174	146	501	452	430	481	451	438
	r. aleat. exc. k (10)	289	210	205	485	460	430	501	433	416
0.01	grad. conjugados	170	170	154	281	281	276	289	289	283
	ret. maximo (5)	210	147	122	283	214	204	239	178	174
	ret. maximo (10)		172	148	326	201	198	448	233	211
	barzilai-borwein	157	135	121	306	273	258	238	207	194
	ret. aleatorio (5)	193	161	134	307	264	241	316	259	254
	ret. aleatorio (10)	230	168	130	378	325	298	332	260	242
	ret. max-min (5)	229	211	164	273	249	214	233	201	177
	ret. max-min (10)	280	261	226	225	211	172	238	219	187
	r. aleat. exc. k (5)	216	157	141	264	231	211	250	203	185
	r. aleat. exc. k (10)	173	129	116	293	232	203	317	206	187

Tabela A.17: Problemas de obstáculo B (n=5041)

teta	método	$x_0 = (L+U)/2$			$x_0 = U$			$x_0 = L$		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
0.1	grad. conjugados	143	143	124	190	190	172	222	222	212
	ret. maximo (5)	201	89	69	243	127	122	260	111	98
	ret. maximo (10)		142	106	543	167	130	521	105	84
	barzilai-borwein	182	135	93	233	195	177	220	170	123
	ret. aleatorio (5)	176	123	98	198	175	139	191	144	132
	ret. aleatorio (10)	189	151	116	252	204	193	256	203	190
	ret. max-min (5)	193	159	126	229	207	191	250	218	157
	ret. max-min (10)	197	136	110	296	239	199	270	215	153
	r. aleat. exc. k (5)	149	116	113	240	137	105	225	150	137
	r. aleat. exc. k (10)	233	109	94	292	156	125	175	108	91
0.9	grad. conjugados	113	113	112	166	166	139	223	223	165
	ret. maximo (5)	229	96	86	348	145	128	325	163	148
	ret. maximo (10)	392	93	71	646	200	166	591	199	168
	barzilai-borwein	143	101	75	221	163	133	227	169	168
	ret. aleatorio (5)	205	120	119	269	207	206	276	199	197
	ret. aleatorio (10)	200	138	137	268	201	170	326	230	218
	ret. max-min (5)	174	121	90	263	202	201	288	244	243
	ret. max-min (10)	213	164	121	341	278	277	306	264	246
	r. aleat. exc. k (5)	197	96	79	250	170	159	216	144	107
	r. aleat. exc. k (10)	191	119	103	361	207	177	308	196	191

Tabela A.18: Problemas de obstáculo B (n=5041)

teta	método	$x_0 = (L+U)/2$			$x_0 = U$			$x_0 = L$		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
10^{-10}	grad. conjugados	227	227	208	276	276	254	187	187	167
	ret. maximo (5)	257	167	153	391	338	317	316	191	168
	ret. maximo (10)	360	176	138	654	356	300	355	185	162
	barzilai-borwein	216	185	166	444	392	361	241	216	209
	ret. aleatorio (5)	220	200	168	393	375	334	294	272	246
	ret. aleatorio (10)	263	210	188	417	356	320	331	280	254
	ret. max-min (5)	209	195	158	351	340	297	292	269	220
	ret. max-min (10)	204	191	129	352	341	296	282	272	244
	r. aleat. exc. k (5)	210	139	108	398	358	344	235	189	165
	r. aleat. exc. k (10)	232	140	126	390	327	312	303	229	203
10^{-5}	grad. conjugados	227	227	208	276	276	254	187	187	167
	ret. maximo (5)	257	167	153	391	338	317	316	191	168
	ret. maximo (10)	483	170	138	654	356	300	355	185	162
	barzilai-borwein	216	185	166	444	392	361	241	216	209
	ret. aleatorio (5)	220	200	168	393	375	334	294	272	246
	ret. aleatorio (10)	263	210	188	417	356	320	331	280	254
	ret. max-min (5)	209	195	158	351	340	297	292	269	220
	ret. max-min (10)	204	191	129	352	341	296	282	272	244
	r. aleat. exc. k (5)	210	139	108	398	358	344	235	189	165
	r. aleat. exc. k (10)	232	140	126	390	327	312	303	229	203
0.01	grad. conjugados	154	154	141	204	204	171	184	184	170
	ret. maximo (5)	228	113	97	275	180	150	232	161	156
	ret. maximo (10)	357	156	132	354	194	169	866	211	177
	barzilai-borwein	226	184	128	270	232	201	205	171	135
	ret. aleatorio (5)	223	174	154	308	289	278	248	204	180
	ret. aleatorio (10)	270	238	223	321	253	226	284	203	163
	ret. max-min (5)	183	167	125	273	233	184	271	253	220
	ret. max-min (10)	192	181	149	268	247	191	270	260	238
	r. aleat. exc. k (5)	229	144	125	227	167	148	251	202	188
	r. aleat. exc. k (10)	210	146	115	287	208	187	248	194	168

Tabela A.19: Problemas de obstáculo C (n=5041)

teta	método	$x_0 = (L+U)/2$			$x_0 = U$			$x_0 = L$		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
0.1	grad. conjugados	142	142	112	181	181	169	184	184	167
	ret. maximo (5)	228	120	103	272	148	131	284	113	93
	ret. maximo (10)	528	141	91	632	161	141	381	147	106
	barzilai-borwein	160	118	94	183	142	105	200	152	101
	ret. aleatorio (5)	188	150	115	298	222	187	253	186	146
	ret. aleatorio (10)	227	179	162	246	171	141	261	202	189
	ret. max-min (5)	169	149	111	240	204	152	208	183	144
	ret. max-min (10)	206	160	141	323	233	206	315	245	200
	r. aleat. exc. k (5)	164	101	86	250	138	104	200	133	108
	r. aleat. exc. k (10)	193	117	98	215	139	119	215	157	147
0.9	grad. conjugados	128	128	108	191	191	169	195	195	155
	ret. maximo (5)	249	86	70	324	149	129	313	172	150
	ret. maximo (10)	430	104	54	604	204	181	576	196	174
	barzilai-borwein	154	112	89	253	187	154	213	159	108
	ret. aleatorio (5)	198	140	120	311	268	236	251	157	145
	ret. aleatorio (10)	232	145	110	325	257	234	293	176	155
	ret. max-min (5)	214	151	114	288	219	195	241	196	163
	ret. max-min (10)	224	184	156	345	276	251	343	307	223
	r. aleat. exc. k (5)	213	115	104	316	188	162	237	154	123
	r. aleat. exc. k (10)	209	88	69	428	212	202	283	201	164

Tabela A.20: Problemas de obstáculo C (n=5041)

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	gradientes conjugados	620	722	543	630	551	769
	retardo maximo (5)	830	1282	1352	1560	1758	1496
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1499	1337	1285	1426	1283	1142
	retardo aleatorio (5)	1024	933	1197	1016	1168	1481
	retardo aleatorio (10)	-	1650	1253	1329	1270	1191
	retardo max-minimo (5)	1519	1431	1225	1305	1385	1078
	retardo max-minimo (10)	1134	1662	1056	1885	1346	1374
	retardo aleat. exc. k (5)	1075	1231	907	1534	1105	1278
	retardo aleat. exc. k (10)	1940	1347	1308	1356	1235	1521
10^{-5}	gradientes conjugados	604	639	543	630	551	769
	retardo maximo (5)	830	1282	1352	1560	1758	1496
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1499	1337	1285	1426	1283	1142
	retardo aleatorio (5)	1024	1426	1197	1016	1168	1150
	retardo aleatorio (10)	-	1377	1253	1329	1270	1502
	retardo max-minimo (5)	1519	1431	1225	1305	1385	1202
	retardo max-minimo (10)	1134	1606	1056	1416	1346	-
	retardo aleat. exc. k (5)	1075	1231	907	1534	1105	1278
	retardo aleat. exc. k (10)	1940	1347	1308	1295	1235	1521
0.01	gradientes conjugados	483	551	482	536	479	652
	retardo maximo (5)	1280	1074	1194	1400	1345	1434
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1566	1358	1386	844	1260	1300
	retardo aleatorio (5)	1684	1372	850	1466	1635	1855
	retardo aleatorio (10)	1764	1622	1416	1544	1144	1171
	retardo max-minimo (5)	1757	1286	1241	1538	1786	1675
	retardo max-minimo (10)	1156	1694	832	1492	1144	1155
	retardo aleat. exc. k (5)	1376	1550	1091	1106	1423	869
	retardo aleat. exc. k (10)	1474	1242	1076	1645	1807	1400

- excedidas 2000 iterações

Tabela A.21: Problemas aleatórios $[na(x^*) = 100]$

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
0.1	gradientes conjugados	506	510	494	520	481	548
	retardo maximo (5)	1293	1302	1538	1481	1155	1129
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1430	1457	890	1690	1060	1139
	retardo aleatorio (5)	1250	1226	1040	1306	1074	1207
	retardo aleatorio (10)	988	1303	852	972	942	1426
	retardo max-minimo (5)	1325	1491	1437	1362	1300	1168
	retardo max-minimo (10)	1238	811	1110	1423	1649	1155
	retardo aleat. exc. k (5)	1118	1216	1260	1273	1135	1225
	retardo aleat. exc. k (10)	1684	929	1248	1350	1350	983
0.9	gradientes conjugados	457	508	472	496	521	538
	retardo maximo (5)	1269	1231	1213	1323	1650	1117
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1145	1378	1020	955	1434	1084
	retardo aleatorio (5)	1163	1104	1312	1554	1175	1588
	retardo aleatorio (10)	1631	942	1408	1662	1108	1323
	retardo max-minimo (5)	1058	984	1812	1553	1058	1735
	retardo max-minimo (10)	1189	1051	1358	1143	1534	1050
	retardo aleat. exc. k (5)	1425	1520	1048	1656	1372	1171
	retardo aleat. exc. k (10)	1012	1369	1576	1355	1598	1032

- excedidas 2000 iterações

Tabela A.22: Problemas aleatórios $[na(x^*) = 100]$

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	gradientes conjugados	547	-	580	-	569	-
	retardo maximo (5)	1108	-	1508	-	1573	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1026	-	1806	-	1485	-
	retardo aleatorio (5)	1076	-	1239	-	1482	-
	retardo aleatorio (10)	1201	-	1466	-	1253	-
	retardo max-minimo (5)	1118	-	1699	-	1356	-
	retardo max-minimo (10)	1136	-	1086	-	1277	-
	retardo aleat. exc. k (5)	1195	-	1364	-	1977	-
	retardo aleat. exc. k (10)	975	-	1291	-	1694	-
10^{-5}	gradientes conjugados	547	1878	598	-	569	-
	retardo maximo (5)	1017	-	1508	-	1573	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1026	-	1806	-	1485	-
	retardo aleatorio (5)	817	-	1130	-	1257	-
	retardo aleatorio (10)	1201	-	1370	-	1169	-
	retardo max-minimo (5)	1118	-	1339	-	1356	-
	retardo max-minimo (10)	1136	-	1086	-	1277	-
	retardo aleat. exc. k (5)	1195	-	1364	-	1977	-
	retardo aleat. exc. k (10)	975	-	1291	-	1694	-
0.01	gradientes conjugados	552	1239	542	1014	537	1253
	retardo maximo (5)	1139	-	1686	-	945	-
	retardo maximo (10)	-	-	-	-	2037	-
	barzilai-borwein	902	-	1660	-	1320	-
	retardo aleatorio (5)	1320	1842	1165	-	834	-
	retardo aleatorio (10)	989	-	1059	-	1284	-
	retardo max-minimo (5)	1044	-	1563	-	1494	-
	retardo max-minimo (10)	1022	-	1436	-	1353	-
	retardo aleat. exc. k (5)	1528	-	1259	-	1460	-
	retardo aleat. exc. k (10)	1186	-	809	-	1376	-

- excedidas 2000 iterações

Tabela A.23: Problemas aleatórios $[na(x^*) = 500]$

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
0.1	gradientes conjugados	505	834	527	851	522	883
	retardo maximo (5)	1143	-	1069	-	1049	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	945	-	1429	-	1397	1413
	retardo aleatorio (5)	1599	-	2033	-	1174	1169
	retardo aleatorio (10)	873	-	1474	-	1515	1770
	retardo max-minimo (5)	879	-	987	-	987	1412
	retardo max-minimo (10)	866	-	1009	-	1120	1709
	retardo aleat. exc. k (5)	1377	-	1628	-	814	-
	retardo aleat. exc. k (10)	1661	-	961	1916	1398	-
0.9	gradientes conjugados	461	556	477	591	498	539
	retardo maximo (5)	1926	1601	1481	-	1663	1098
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	996	1253	1369	1792	1148	1386
	retardo aleatorio (5)	1067	1061	982	1782	888	1321
	retardo aleatorio (10)	1237	1285	1408	1492	838	1686
	retardo max-minimo (5)	1628	1015	1143	1777	1328	-
	retardo max-minimo (10)	1085	1809	1296	-	1418	1427
	retardo aleat. exc. k (5)	1107	1287	1744	1640	1193	1129
	retardo aleat. exc. k (10)	1268	1171	1282	1622	808	839

- excedidas 2000 iterações

Tabela A.24: Problemas aleatórios $[na(x^*) = 500]$

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
10^{-10}	gradientes conugados	478	1790	471	-	523	-
	retardo maximo (5)	-	-	1092	-	858	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	-	-	1156	-	849	-
	retardo aleatorio (5)	1653	-	1752	-	1500	-
	retardo aleatorio (10)	1573	-	1671	-	1222	-
	retardo max-minimo (5)	1305	-	1351	-	1245	-
	retardo max-minimo (10)	1759	-	1077	-	1325	-
	retardo aleat. exc. k (5)	1948	-	1433	-	1131	-
	retardo aleat. exc. k (10)	1127	-	1571	-	1649	-
10^{-5}	gradientes conugados	478	1711	465	-	523	-
	retardo maximo (5)	1632	-	1651	-	1171	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	955	-	992	-	849	-
	retardo aleatorio (5)	1653	-	1160	-	1500	-
	retardo aleatorio (10)	1573	-	1253	-	1222	-
	retardo max-minimo (5)	-	-	1203	-	1382	-
	retardo max-minimo (10)	1868	-	1322	-	1057	-
	retardo aleat. exc. k (5)	1457	-	1065	-	1131	-
	retardo aleat. exc. k (10)	-	-	1570	-	1649	-
0.01	gradientes conugados	403	963	451	1371	454	1035
	retardo maximo (5)	1585	-	882	-	1468	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1210	-	767	-	1138	-
	retardo aleatorio (5)	1694	-	1142	-	1311	-
	retardo aleatorio (10)	1631	-	1638	-	899	-
	retardo max-minimo (5)	1380	-	1125	-	1230	-
	retardo max-minimo (10)	1817	-	997	-	1597	-
	retardo aleat. exc. k (5)	1324	-	1439	-	1345	-
	retardo aleat. exc. k (10)	1704	-	1637	-	1627	-

- excedidas 2000 iterações

Tabela A.25: Problemas aleatórios $[na(x^*) = 900]$

teta	método	$na(x_0) = 100$		$na(x_0) = 500$		$na(x_0) = 900$	
		deg=1	deg=12	deg=1	deg=12	deg=1	deg=12
0.1	gradientes conugados	399	844	430	976	422	983
	retardo maximo (5)	1557	-	1365	-	1258	-
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	-	1778	1629	-	1399	-
	retardo aleatorio (5)	1549	-	1310	-	1414	-
	retardo aleatorio (10)	-	-	-	-	1168	1948
	retardo max-minimo (5)	1921	-	1467	-	1212	-
	retardo max-minimo (10)	1928	-	1084	-	1438	-
	retardo aleat. exc. k (5)	1904	-	1502	-	1438	-
	retardo aleat. exc. k (10)	1883	-	969	-	1135	-
0.9	gradientes conugados	380	635	393	569	435	529
	retardo maximo (5)	1473	-	1548	-	-	1479
	retardo maximo (10)	-	-	-	-	-	-
	barzilai-borwein	1217	940	1076	1087	959	1746
	retardo aleatorio (5)	1015	-	1353	1228	1371	1523
	retardo aleatorio (10)	1188	-	1225	1309	1144	1445
	retardo max-minimo (5)	1645	-	1435	1552	1189	1991
	retardo max-minimo (10)	1492	1484	1133	1240	1138	1582
	retardo aleat. exc. k (5)	-	1623	1052	1594	995	1490
	retardo aleat. exc. k (10)	1480	-	1236	1019	1453	1695

- excedidas 2000 iterações

Tabela A.26: Problemas aleatórios $[na(x^*) = 900]$

teta	método	Problema			
		1	2	3	4
10^{-10}	gradientes conjugados	381	-	1437	170
	retardo maximo (5)	979	-	-	1375
	retardo maximo (10)	-	-	-	-
	barzilai-borwein	1350	-	-	898
	retardo aleatorio (5)	1369	-	-	942
	retardo aleatorio (10)	1269	-	-	1385
	retardo max-minimo (5)	1579	-	-	1287
	retardo max-minimo (10)	1272	-	-	1016
	retardo aleat. exc. k (5)	1405	-	-	669
	retardo aleat. exc. k (10)	1767	-	-	830
10^{-5}	gradientes conjugados	381	-	1498	170
	retardo maximo (5)	979	-	-	1375
	retardo maximo (10)	-	-	-	-
	barzilai-borwein	1350	-	-	898
	retardo aleatorio (5)	1369	-	-	942
	retardo aleatorio (10)	1269	-	-	1385
	retardo max-minimo (5)	1123	-	-	1287
	retardo max-minimo (10)	1272	-	1967	1016
	retardo aleat. exc. k (5)	1405	-	-	669
	retardo aleat. exc. k (10)	1767	-	-	830
0.01	gradientes conjugados	385	1011	1049	175
	retardo maximo (5)	1493	-	-	998
	retardo maximo (10)	-	-	-	-
	barzilai-borwein	1300	-	-	600
	retardo aleatorio (5)	1331	-	-	882
	retardo aleatorio (10)	1354	-	-	1211
	retardo max-minimo (5)	-	-	-	1220
	retardo max-minimo (10)	738	-	-	799
	retardo aleat. exc. k (5)	1174	-	-	834
	retardo aleat. exc. k (10)	1009	-	-	636

- excedidas 2000 iterações

Tabela A.27: Problemas aleatórios B

teta	método	Problema			
		1	2	3	4
0.1	gradientes conjugados	389	945	941	124
	retardo maximo (5)	1339	-	-	1303
	retardo maximo (10)	-	-	-	-
	barzilai-borwein	1313	1228	1812	962
	retardo aleatorio (5)	794	-	-	1334
	retardo aleatorio (10)	1133	1640	-	877
	retardo max-minimo (5)	1687	1348	1769	1200
	retardo max-minimo (10)	1062	1389	1466	1438
	retardo aleat. exc. k (5)	991	1778	-	637
	retardo aleat. exc. k (10)	1437	-	1522	918
0.9	gradientes conjugados	381	491	478	149
	retardo maximo (5)	1427	1292	1145	1535
	retardo maximo (10)	-	-	-	-
	barzilai-borwein	1523	1364	1363	772
	retardo aleatorio (5)	1582	1315	1592	748
	retardo aleatorio (10)	1464	1228	1955	666
	retardo max-minimo (5)	1105	1496	1758	1223
	retardo max-minimo (10)	1438	1561	1339	910
	retardo aleat. exc. k (5)	1165	1500	1449	1179
	retardo aleat. exc. k (10)	1740	1189	1296	1608

- excedidas 2000 iterações

Tabela A.28: Problemas aleatórios B

teta	método	Problema A			Problema B			Problema C		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
10^{-10}	grad. conjugados	70	70	68	22	22	17	222	222	220
	ret. maximo (5)	52	52	51	28	15	0	222	222	220
	ret. maximo (10)	61	61	60	62	23	0	222	222	220
	barzilai-borwein	62	62	61	25	21	17	222	222	220
	ret. aleatorio (5)	70	70	68	35	32	22	222	222	220
	ret. aleatorio (10)	70	70	68	32	30	25	222	222	220
	ret. max-min (5)	63	63	61	26	24	0	222	222	220
	ret. max-min (10)	63	63	61	24	21	0	222	222	220
	r. aleat. exc. k (5)	57	57	56	27	19	0	222	222	220
	r. aleat. exc. k (10)	57	57	56	33	19	0	222	222	220
10^{-5}	grad. conjugados	70	70	68	22	22	17	222	222	220
	ret. maximo (5)	52	52	51	28	15	0	222	222	220
	ret. maximo (10)	61	61	60	62	23	0	222	222	220
	barzilai-borwein	55	55	54	25	21	17	222	222	220
	ret. aleatorio (5)	70	70	68	35	32	22	222	222	220
	ret. aleatorio (10)	70	70	68	32	30	25	222	222	220
	ret. max-min (5)	63	63	61	26	24	0	222	222	220
	ret. max-min (10)	63	63	61	24	21	0	222	222	220
	r. aleat. exc. k (5)	57	57	56	27	19	0	222	222	220
	r. aleat. exc. k (10)	57	57	56	33	19	0	222	222	220
0.01	grad. conjugados	68	68	67	22	22	20	144	144	137
	ret. maximo (5)	50	50	47	28	15	0	78	60	54
	ret. maximo (10)	53	53	51	62	23	0	141	63	59
	barzilai-borwein	62	62	60	27	20	0	70	66	64
	ret. aleatorio (5)	69	69	67	35	32	22	83	82	79
	ret. aleatorio (10)	69	69	67	32	30	25	95	87	85
	ret. max-min (5)	51	51	49	26	24	0	60	58	52
	ret. max-min (10)	51	51	49	24	21	0	60	58	52
	r. aleat. exc. k (5)	54	54	52	27	19	0	62	58	56
	r. aleat. exc. k (10)	54	54	52	33	19	0	69	64	62

Tabela A.29: Problemas de reconstrução de imagens

teta	método	Problema A			Problema B			Problema C		
		iterações		face	iterações		face	iterações		face
		total	decr.	ótima	total	decr.	ótima	total	decr.	ótima
0.1	grad. conjugados	47	47	44	19	19	0	130	130	123
	ret. maximo (5)	57	33	30	28	15	0	60	38	25
	ret. maximo (10)	46	46	35	62	23	0	91	37	33
	barzilai-borwein	45	43	36	27	20	0	49	38	33
	ret. aleatorio (5)	33	33	32	32	23	0	69	42	40
	ret. aleatorio (10)	31	31	26	38	27	0	64	53	51
	ret. max-min (5)	48	46	45	26	24	0	46	42	36
	ret. max-min (10)	48	46	45	24	21	0	63	42	39
	r. aleat. exc. k (5)	47	43	41	27	19	0	75	53	51
	r. aleat. exc. k (10)	47	43	41	33	19	0	56	52	50
0.9	grad. conjugados	35	35	0	19	19	0	147	147	0
	ret. maximo (5)	32	25	0	28	15	0	79	25	0
	ret. maximo (10)	49	29	0	62	23	0	168	48	45
	barzilai-borwein	23	18	0	27	20	0	35	25	0
	ret. aleatorio (5)	23	20	0	32	23	0	42	28	0
	ret. aleatorio (10)	26	24	0	38	27	0	53	31	0
	ret. max-min (5)	21	18	0	26	24	0	50	40	0
	ret. max-min (10)	28	27	0	24	21	0	42	38	0
	r. aleat. exc. k (5)	26	21	0	27	19	0	54	28	0
	r. aleat. exc. k (10)	31	22	0	33	19	0	61	32	0

Tabela A.30: Problemas de reconstrução de imagens

método	Problema									
	1		2		3		4		5	
	x_0		x_0		x_0		x_0		x_0	
	0	1	0	1	0	1	0	1	0	1
gradientes conjugados	40	92	57	84	51	110	81	130	54	128
retardo maximo (5)	91	121	101	129	96	238	186	197	108	302
retardo maximo (10)	94	141	165	336	104	346	266	967	133	601
barzilai-borwein	60	118	105	124	75	161	160	202	82	169
retardo aleatorio (5)	66	110	88	120	86	230	138	184	102	226
retardo aleatorio (10)	69	153	103	136	91	181	159	249	96	216
retardo max-minimo (5)	50	116	94	107	83	159	143	227	92	205
retardo max-minimo (10)	70	121	110	107	80	186	158	165	111	211
retardo aleat. exc. k (5)	62	122	91	133	87	201	136	186	84	193
retardo aleat. exc. k (10)	78	168	89	119	93	182	146	166	68	267

Tabela A.31: Problemas de projeção A ($\theta = 0.1$).

método	Problema									
	1		2		3		4		5	
	x_0		x_0		x_0		x_0		x_0	
	0	1	0	1	0	1	0	1	0	1
grad. conjugados	11342	8717	7517	6512	5053	5021	*	*	4906	5132
ret. maximo (5)	25691	19903	24379	23032	19350	13385	*	*	8225	17425
ret. maximo (10)	*	*	*	*	*	*	*	*	*	*
barzilai-borwein	13174	11125	5659	10456	8734	8959	*	*	10179	9529
ret. aleatorio (5)	7746	9353	8414	6403	5186	6378	*	*	6842	5294
ret. aleatorio (10)	14655	10195	7279	9524	8530	7037	*	*	7071	8582
ret. max-min (5)	15957	18523	11816	7902	8062	8784	*	*	13686	13343
ret. max-min (10)	8936	8686	6703	9806	5239	8745	*	*	4430	5842
r. aleat. exc. k (5)	11322	12748	12705	12243	5490	7915	*	*	8734	8933
r. alcat. exc. k (10)	17545	11598	10748	9955	11013	7706	*	*	10080	5964

*excedidas 32000 iterações

Tabela A.32: Problemas de projeção B ($\theta = 0.1$).

teta	delta	n		
		2061	5041	10000
1E-10	0.E+00	288	410	1851
	1.E-07	253	422	1200
	1.E-04	415	1059	1916
	1.E-02	548	958	1809
	1.E-01	490	992	1512
1E-5	0.E+00	288	437	1847
	1.E-07	253	480	1200
	1.E-04	415	1059	1916
	1.E-02	548	958	1809
	1.E-01	490	992	1512
1E-2	0.E+00	254	364	1046
	1.E-07	261	380	1448
	1.E-04	489	985	1818
	1.E-02	548	958	1809
	1.E-01	490	992	1512
1E-1	0.E+00	279	499	732
	1.E-07	279	502	732
	1.E-04	544	808	1295
	1.E-02	495	*	1782
	1.E-01	540	*	*
9E-1	0.E+00	175	478	584
	1.E-07	175	478	584
	1.E-04	175	481	637
	1.E-02	292	776	*
	1.E-01	436	*	*

*não obtivemos progresso suficiente

Tabela A.33: Problemas de obstáculo A (p1=0.3, p2=1)

teta	delta	problema			
		1	2	3	4
1E–10	1.E-07	392	2876	3278	167
	1.E-04	506	*	4420	166
	1.E-02	466	*	*	1024
	1.E-01	1081	*	*	345
	5.E-01	1370	4741	*	*
1E-5	1.E-07	392	2913	3184	167
	1.E-04	506	*	4420	166
	1.E-02	466	*	*	1024
	1.E-01	1081	*	*	345
	5.E-01	1370	4741	*	*
1E-2	1.E-07	385	1536	2150	175
	1.E-04	382	1232	4534	133
	1.E-02	638	4990	*	708
	1.E-01	1259	3461	*	345
	5.E-01	779	2903	*	904
1E-1	1.E-07	389	652	794	124
	1.E-04	404	1685	984	124
	1.E-02	767	1104	2235	342
	1.E-01	1134	2539	1870	345
	5.E-01	779	2578	2163	628
9E-1	1.E-07	381	437	455	149
	1.E-04	381	437	455	149
	1.E-02	381	458	484	149
	1.E-01	614	861	680	303
	5.E-01	1248	1238	635	571

*excedidas 5000 iterações

Tabela A.34: Problemas alcatórios B

teta	delta	problema		
		A	B	C
1E−10	1.E-07	70	22	217
	1.E-04	64	21	163
	1.E-02	37	20	137
	1.E-01	46	20	149
	5.E-01	35	20	149
1E-5	1.E-07	70	22	217
	1.E-04	64	21	163
	1.E-02	37	20	137
	1.E-01	46	20	149
	5.E-01	35	20	149
1E-2	1.E-07	68	22	144
	1.E-04	64	21	141
	1.E-02	37	20	137
	1.E-01	46	20	149
	5.E-01	35	20	149
1E-1	1.E-07	47	19	130
	1.E-04	47	19	130
	1.E-02	37	19	123
	1.E-01	46	19	149
	5.E-01	35	19	149
9E-1	1.E-07	35	19	147
	1.E-04	35	19	147
	1.E-02	35	19	147
	1.E-01	35	19	147
	5.E-01	35	19	147

Tabela A.35: Problemas de reconstrução de imagens

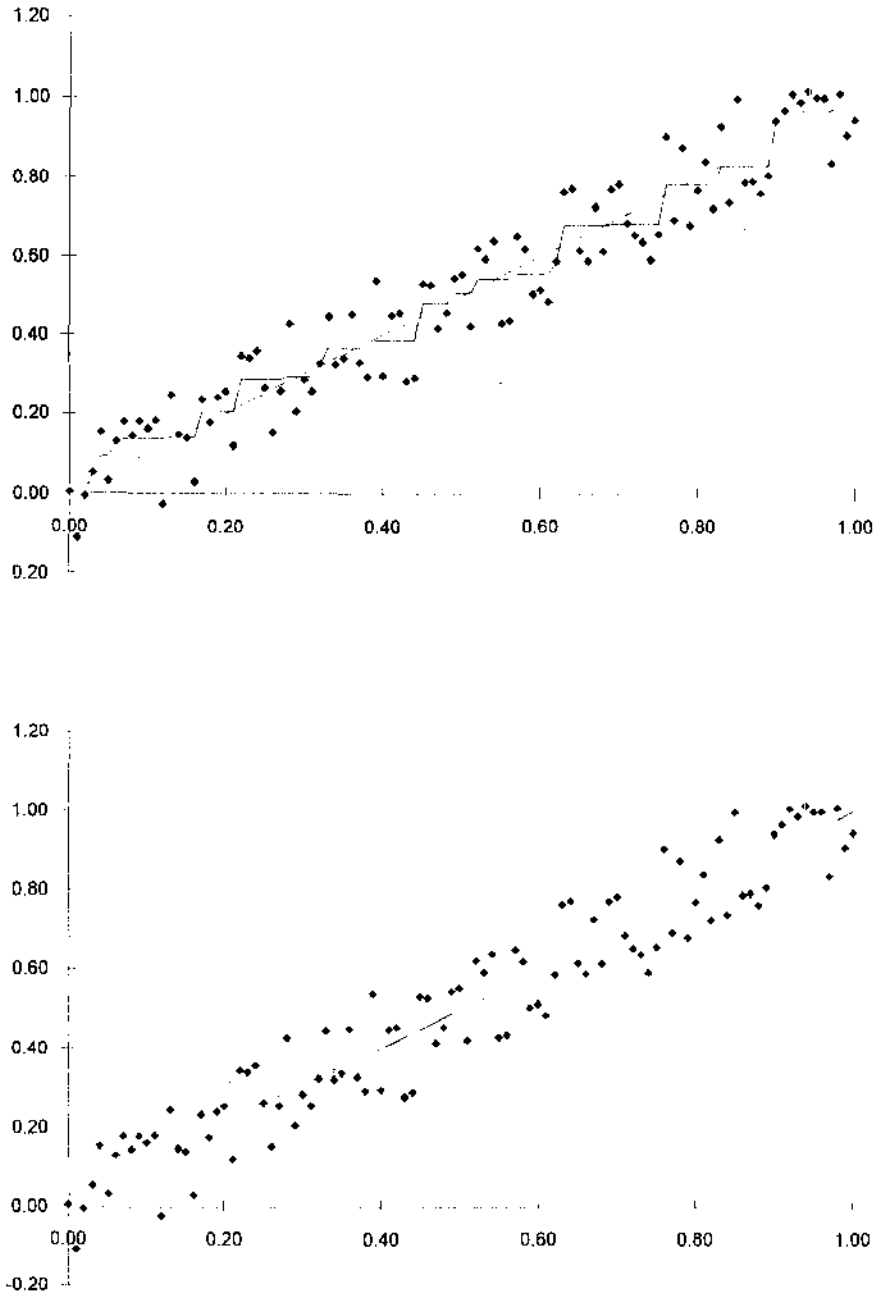


Figura A.1: Aproximando $y(x) = x + \epsilon$ por uma função crescente e por uma função convexa.

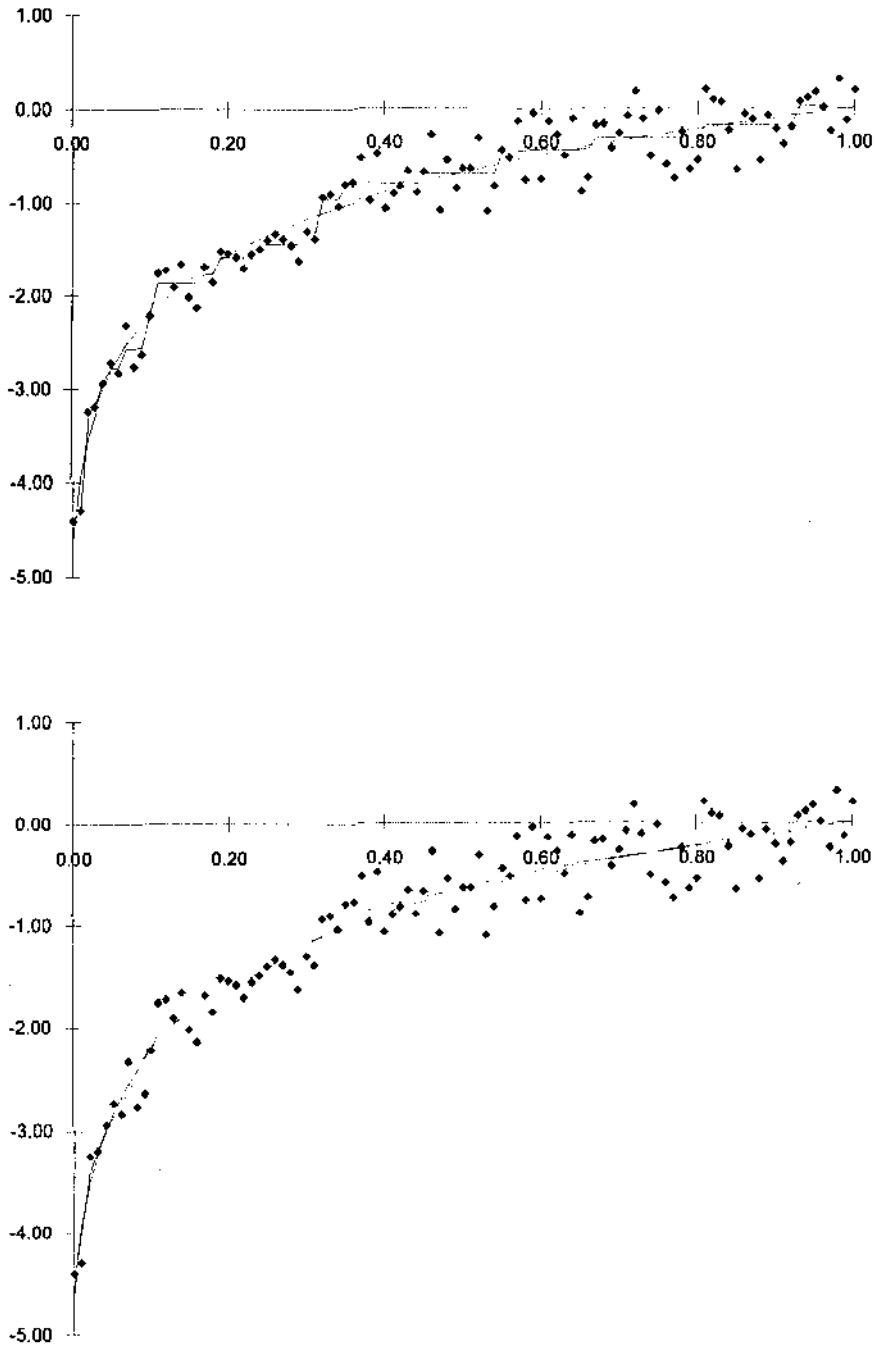


Figura A.2: Aproximando $y(x) = log(x) + \epsilon$ por uma função crescente e por uma função côncava.

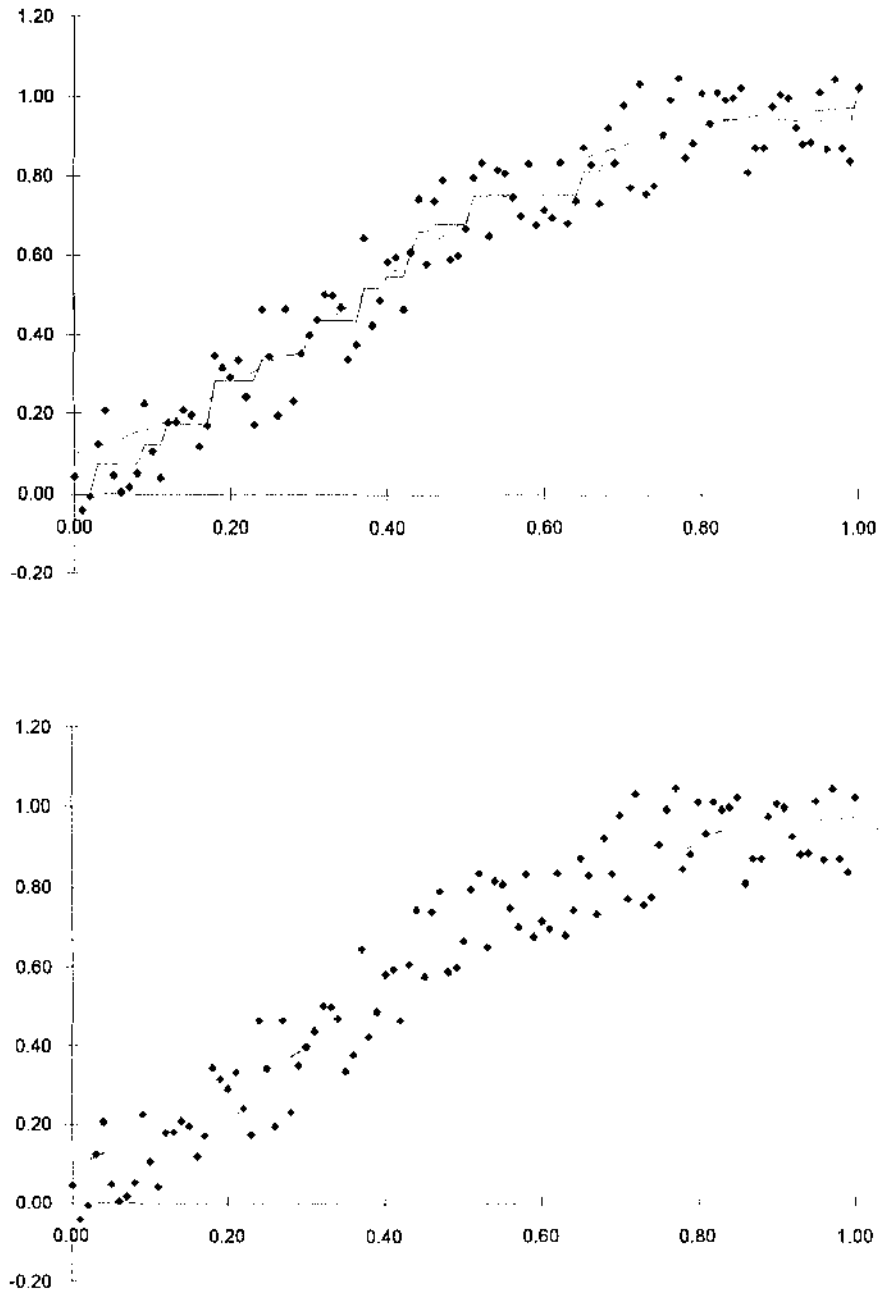


Figura A.3: Aproximando $y(x) = 1/(1 + 9e^{-6x}) + \epsilon$ por uma função crescente e por uma função convexa.

Problemas de Hock e Schittkowski

Apresentamos aqui os problemas extraídos do livro de Hock e Schittkowski [27] e usados nos captulos 2 e 3.

PROBLEMA 1 (38)

Função objetivo:

$$\begin{aligned} f(x) &= 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2 \\ &\quad +10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1) \end{aligned}$$

Canalizações:

$$-10 \leq \ x_i \ \leq 10, \ i = 1, \ldots, 4$$

Ponto inicial:

$$x_0 \ = \ (-3, -1, -3, -1)$$

PROBLEMA 2 (41)

Função objetivo:

$$f(x) = 2 - x_1 x_2 x_3$$

Restrição:

$$x_1 + 2x_2 + 2x_3 - x_4 = 0$$

Canalizações:

$$\begin{aligned} 0 \leq x_i &\leq 1, \ i = 1, 2, 3 \\ 0 \leq x_4 &\leq 2 \end{aligned}$$

Ponto inicial:

$$x_0 = (2, 2, 2, 2)$$

Problema 3 (45)

Função objetivo:

$$f(x) = 2 - \frac{1}{120}x_1x_2x_3x_4x_5$$

Canalizações:

$$0 \leq x_i \leq i,\ i = 1, \ldots, 5$$

Ponto inicial:

$$x_0 = (2, 2, 2, 2)$$

PROBLEMA 4 (53)

Função objetivo:

$$f(x) = (x_1 - x_2)^2 + (x_2 + x_3 - 2)^2 + (x_4 - 1)^2 + (x_5 - 1)^2$$

Restrições:

$$\begin{aligned} x_1 + 3x_2 &= 0 \\ x_3 + x_4 - 2x_5 &= 0 \\ x_2 - x_5 &= 0 \end{aligned}$$

Canalizações:

$$-10 \leq x_i \leq 10,\ i = 1, \ldots, 5$$

Ponto inicial:

$$x_0 = (2, 2, 2, 2, 2)$$

PROBLEMA 5 (54)

Função objetivo:

$$\begin{aligned} f(x) &= -e^{-h(x)/2} \\ h(x) &= [(x_1 - 10^4)^2/6.4\,10^7 + (x_1 - 10^4)(x_2 - 1)/2\,10^4 + (x_2 - 1)^2]/0.96 \\ &\quad +(x_3 - 2\,10^6)^2/4.9\,10^{13} + (x_4 - 10^2/2.5\,10^3 \\ &\quad +(x_5 - 0.001)^2/0.0025 + (x_6 - 1\,10^8)^2/2.5\,10^{17} \end{aligned}$$

Restrição:

$$x_1 + 4000x_2 - 17600 = 0$$

Canalizações:

$$\begin{aligned} 0 \leq\ & x_1 \leq 2000 \\ -10 \leq\ & x_2 \leq 10 \\ 0 \leq\ & x_3 \leq 10^7 \\ 0 \leq\ & x_4 \leq 20 \\ -1 \leq\ & x_5 \leq 1 \\ 0 \leq\ & x_6 \leq 2\,10^8 \end{aligned}$$

Ponto inicial:

$$x_0 = (6000, 1.5, 4\,10^6, 2, 0.003, 5\,10^7)$$

PROBLEMA 6 (55)

Função objetivo:

$$f(x) = x_1 + 2x_2 + 4x_5 + e^{x_1 x_4}$$

Restrições:

$$\begin{aligned} x_1 + 2x_2 + 5x_5 - 6 &= 0 \\ x_1 + x_2 + x_3 - 3 &= 0 \\ x_4 + x_5 + x_6 - 2 &= 0 \\ x_1 + x_4 - 1 &= 0 \\ x_2 + x_5 - 2 &= 0 \\ x_3 + x_6 - 2 &= 0 \end{aligned}$$

Canalizações:

$$\begin{aligned} 0 \leq\ & x_i \\ & x_1 \leq 1 \\ & x_4 \leq 1 \end{aligned}$$

Ponto inicial:

$$x_0 = (1, 2, 0, 0, 0, 2)$$

PROBLEMA 7 (60)

Função objetivo:

$$f(x) = (x_1 - 1)^2 + (x_1 - x_2)^2 + (x_2 - x_3)^4$$

Restrição:

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

Canalizações:

$$-10 \leq x_i \leq 10, \; i = 1, 2, 3$$

Ponto inicial:

$$x_0 = (2, 2, 2)$$

PROBLEMA 8 (62)

Função objetivo:

$$\begin{aligned} f(x) &= -32.174\{255\log[(x_1+x_2+x_3+0.03)/(0.09x_1+x_2+x_3+0.03)] \\ &\quad +280\log[(x_2+x_3+0.03)/(0.07x_2+x_3+0.03)] \\ &\quad +290\log[(x_3+0.03)/(0.13x_3+0.03)]\} \end{aligned}$$

Restrição:

$$x_1+x_2+x_3-1 = 0$$

Canalização:

$$0 \leq x_i,\ i=1,2,3$$

Ponto inicial:

$$x_0 = (0.7, 0.2, 0.1)$$

PROBLEMA 9 (63)

Função objetivo:

$$f9x) \;=\; 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$$

Restrições:

$$\begin{aligned} 8x_1 + 14x_2 + 7x_3 - 56 &= 0 \\ x_1^2 + x_2^2 + x_3^2 - 25 &= 0 \end{aligned}$$

Canalizações:

$$0 \leq \; x_i, \; i = 1, 2, 3$$

Ponto inicial:

$$x_0 \;=\; (2, 2, 2)$$

PROBLEMA 10 (74)

Função objetivo:

$$f(x) = 3x_1 + 10^{-6}x_1^3 + 2x_2 + \frac{2}{3}10^{-6}x_2^3$$

Restrições:

$$\begin{aligned} x_4 - x_3 - x_5 + 0.55 &= 0 \\ x_3 - x_4 - x_6 + 0.55 &= 0 \\ 1000[\sin(-x_3 - 0.25) + \sin(-x_4 - 0.25)] + 894.8 - x_1 &= 0 \\ 1000[\sin(x_3 - 0.25) + \sin(x_3 - x_4 - 0.25)] + 894.8 - x_2 &= 0 \\ 1000[\sin(x_4 - 0.25) + \sin(x_4 - x_3 - 0.25)] + 1294.8 &= 0 \end{aligned}$$

Canalizações:

$$\begin{aligned} 0 \leq \ & x_i \ \leq 1200, \ i = 1, 2 \\ -0.55 \leq \ & x_i \ \leq 0.55, \ i = 3, 4 \end{aligned}$$

Ponto inicial:

$$x_0 = (0, 0, 0, 0, 0.55, 0.55)$$

PROBLEMA 11 (75)

Função objetivo:

$$f(x) = 3x_1 + 10^{-6}x_1^3 + 2x_2 + \frac{2}{3}10^{-6}x_2^3$$

Restrições:

$$\begin{aligned}
x_4 - x_3 - x_5 + 0.48 &= 0 \\
x_3 - x_4 - x_6 + 0.48 &= 0 \\
1000[\sin(-x_3 - 0.25) + \sin(-x_4 - 0.25)] + 894.8 - x_1 &= 0 \\
1000[\sin(x_3 - 0.25) + \sin(x_3 - x_4 - 0.25)] + 894.8 - x_2 &= 0 \\
1000[\sin(x_4 - 0.25) + \sin(x_4 - x_3 - 0.25)] + 1294.8 &= 0
\end{aligned}$$

Canalizações:

$$\begin{aligned}
0 \leq \ & x_i \ \leq 1200, \ i = 1, 2 \\
-0.55 \leq \ & x_i \ \leq 0.55, \ i = 3, 4
\end{aligned}$$

Ponto inicial:

$$x_0 = (0, 0, 0, 0, 0.48, 0.48)$$

PROBLEMA 12 (80)

Função objetivo:

$$f(x) = e^{x_1 x_2 x_3 x_4 x_5}$$

Restrições:

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 &= 0 \\ x_2 x_3 - 5 x_4 x_5 &= 0 \\ x_1^3 + x_2^3 + 1 &= 0 \end{aligned}$$

Canalizações:

$$\begin{aligned} -2.3 \leq x_i \leq 2.3, \ i = 1, 2 \\ -3.2 \leq x_i \leq 3.2, \ i = 3, 4, 5 \end{aligned}$$

Ponto inicial:

$$x_0 = (-2, 2, 2, -1, -1)$$

Problema 13 (81)

Função objetivo:

$$f(x) = e^{x_1 x_2 x_3 x_4 x_5} - 0.5(x_1^3 + x_2^3 + 1)^2$$

Restrições:

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 &= 0 \\ x_2 x_3 - 5 x_4 x_5 &= 0 \\ x_1^3 + x_2^3 + 1 &= 0 \end{aligned}$$

Canalizações:

$$\begin{aligned} -2.3 \leq x_i &\leq 2.3, \ i = 1, 2 \\ -3.2 \leq x_i &\leq 3.2, \ i = 3, 4, 5 \end{aligned}$$

Ponto inicial:

$$x_0 = (-2, 2, 2, -1, -1)$$

PROBLEMA 14 (99)

Função objetivo:

$$\begin{aligned} f(x) &= -r_8(x)^2 \\ r_1(x) &= 0 \\ r_i(x) &= a_i(t_i - t_{i-1})\cos(x_{i-1}) + r_{i-1}(x),\ i = 2, \ldots, 8 \end{aligned}$$

Restrições:

$$\begin{aligned} q_8(x) - 10^5 &= 0 \\ s_8(x) - 10^3 &= 0 \\ q_1(x) &= 0 \\ s_1(x) &= 0 \\ q_i(x) &= 0.5(t_i - t_{i-1})^2[a_i \sin(x_{i-1}) - b] \\ &\quad +(t_i - t_{i-1})s_{i-1}(x) + q_{i-1}(x),\ i = 2, \ldots, 8 \\ s_i(x) &= (t_i - t_{i-1})[a_i \sin(x_{i-1}) - b] \\ &\quad +s_{i-1}(x),\ i = 2, \ldots, 8 \\ a^T &= [0, 50, 50, 75, 75, 75, 100, 100] \\ t^T &= [0, 25, 50, 100, 150, 200, 290, 380] \end{aligned}$$

Canalizações:

$$0 \leq \ x_i \ \leq 1.58,\ i = 1, \ldots, 7$$

Ponto inicial:

$$x_0 \ = \ (0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$$

Problema 15 (107)

Função objetivo:

$$f(x) = 3000x_1 + 1000x_1^3 + 2000x_2 + 666.667x_2^3$$

Restrições:

$$\begin{aligned}
0.4 - x_1 + 2cx_5^2 - x_5x_6(dy_1 + cy_2) - x_5x_7(dy_3 + cy_4) &= 0 \\
0.4 - x_2 + 2cx_6^2 - x_5x_6(dy_1 - cy_2) + x_6x_7(dy_5 - cy_6) &= 0 \\
0.8 + 2cx_7^2 + x_5x_7(dy_3 - cy_4) - x_6x_7(dy_5 + cy_6) &= 0 \\
0.2 - x_3 + 2dx_5^2 + x_5x_6(cy_1 - dy_2) + x_5x_7(cy_3 - dy_4) &= 0 \\
0.2 - x_4 + 2dx_6^2 - x_5x_6(cy_1 + dy_2) - x_6x_7(cy_5 + dy_6) &= 0 \\
-0.337 + 2dx_7^2 - x_5x_7(cy_3 + dy_4) + x_6x_7(cy_5 - dy_6) &= 0
\end{aligned}$$

$$\begin{aligned}
y_1 = \sin(x_8), \quad y_3 = \sin(x_9), \quad y_5 = \sin(x_8 - x_9) \\
y_2 = \cos(x_8), \quad y_3 = \cos(x_9), \quad y_5 = \cos(x_8 - x_9)
\end{aligned}$$

$$c = (48.4/50.176)\sin(0.25), \qquad d = (48.4/50.176)\cos(0.25)$$

Canalizações:

$$\begin{aligned}
0 \leq \quad & x_i \quad i = 1, 2 \\
0.90909 \leq \quad & x_i \quad 1.0909, \; i = 5, 6, 7
\end{aligned}$$

Ponto inicial:

$$x_0 = (0.8, 0.8, 0.2, 0.2, 1.0454, 1.0454, 0, 0)$$

PROBLEMA 16 (110)

Função objetivo:

$$f(x) = \sum_{i=1}^{10}\{[\log(x_i - 2)]^2 + [\log(10 - x_i)]^2\} - (\prod_{i=1}^{10} x_i)^{0.2}$$

Canalizações:

$$2.001 \leq x_i \leq 9.999,\ i = 1, \ldots, 10$$

Ponto inicial:

$$x_0 = (9, 9, 9, 9, 9, 9, 9, 9, 9, 9)$$

PROBLEMA 17 (111)

Função objetivo:

$$\begin{aligned} f(x) &= \sum_{j=1}^{10} e^{x_j}[c_j + x_j - \log(\sum_{k=1}^{10} e^{x_k})] \\ c^T &= [-6.089, -17.164, -34.054, -5.914, -24.721, \ldots \\ &\quad -14.986, -24.100, -10.708, -26.662, -22.179] \end{aligned}$$

Restrições:

$$\begin{aligned} e^{x_1} + 2e^{x_2} + 2e^{x_3} + e^{x_6} + e^{x_{10}} - 2 &= 0 \\ e^{x_4} + 2e^{x_5} + e^{x_6} + e^{x_7} - 1 &= 0 \\ e^{x_3} + e^{x_7} + e^{x_8} + 2e^{x_9} + e^{x_{10}} - 1 &= 0 \end{aligned}$$

Canalizações:

$$-100 \leq \quad x_i \quad \leq 100, \; i = 1, \ldots, 10$$

Ponto inicial:

$$x_0 = (-2.3, -2.3, -2.3, -2.3, -2.3, -2.3, -2.3, -2.3, -2.3, -2.3)$$

PROBLEMA 18 (112)

Função objetivo:

$$\begin{aligned} f(x) &= \sum_{j=1}^{10} x_j\{c_j - \log[x_j/(\sum_{k=1}^{10} x_k)]\} \\ c^T &= [-6.089, -17.164, -34.054, -5.914, -24.721, \ldots \\ &\quad -14.986, -24.100, -10.708, -26.662, -22.179] \end{aligned}$$

Restrições:

$$\begin{aligned} x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 &= 0 \\ x_4 + 2x_5 + x_6 + x_7 - 1 &= 0 \\ x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 &= 0 \end{aligned}$$

Canalizações:

$$10^{-6} \leq \quad x_i, \quad i = 1, \ldots, 10$$

Ponto inicial:

$$x_0 = (0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1)$$

PROBLEMA 19 (113)

Função objetivo:

$$\begin{aligned} f(x) &= x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 \\ &\quad +4(x_4 - 5)^2 + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 \\ &\quad +7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_10 - 7)^2 + 45 \end{aligned}$$

Restrições:

$$\begin{aligned} 105 - 4x_1 - 5x_2 + 3x_7 - 9x_8 - x_{11} &= 0 \\ -10x_1 + 8x_2 + 17x_7 - 2x_8 - x_{12} &= 0 \\ 8x_1 - 2x_2 - 5x_9 + 2x_{10} - x_{13} + 12 &= 0 \\ -3(x_1 - 2)^2 - 4(x_2 - 3)^2 - 2x_3^2 + 7x_4 - x_{14} + 120 &= 0 \\ -5x_1^2 - 8x_2 - (x_3 - 6)^2 + 2x_4 - x_{15} + 40 &= 0 \\ -0.5(x_1 - 8)^2 - 2(x_2 - 4)^2 - 3x_5^2 + x_6 - x_{16} + 30 &= 0 \\ -x_1^2 - 2(x_2 - 2)^2 + 2x_1x_2 - 14x_5 + 6x_6 - x_{17} &= 0 \\ 3x_1 - 6x_2 - 12(x_9 - 8)^2 + 7x_{10} - x_{18} &= 0 \end{aligned}$$

Ponto inicial:

$$x_0 = (2, 3, 5, 5, 1, 2, 7, 3, 6, 10, 76, 117, 12, 105, 5, 9, 4, 10)$$

PROBLEMA 20 (119)

Função objetivo:

$$f(x) = \sum_{i=1}^{16}\sum_{j=1}^{16} a_{ij}(x_i^2 + x_i + 1)(x_j^2 + x_j + 1)$$

Restrições:

$$\sum_{j=1}^{16} b_{ij}x_j - c_i = 0,\ i = 1, \ldots, 8$$

Canalizações:

$$0 \leq x_i \leq 5,\ i = 1, \ldots, 16$$

Ponto inicial:

$$x_0 = (10, \ldots, 10)$$

Os elementos a_{ij}, b_{ij} e c_i são fornecidos na próxima página.

$$A = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}$$

$$B^T = \begin{bmatrix}
0.22 & -1.46 & 1.29 & -1.10 & 0.00 & 0.00 & 1.12 & 0.00 \\
0.20 & 0.00 & -0.89 & -1.06 & 0.00 & -1.72 & 0.00 & 0.45 \\
0.19 & -1.30 & 0.00 & 0.95 & 0.00 & -0.33 & 0.00 & 0.26 \\
0.25 & 1.82 & 0.00 & -0.54 & -1.43 & 0.00 & 0.31 & -1.10 \\
0.15 & -1.15 & -1.16 & 0.00 & 1.51 & 1.62 & 0.00 & 0.58 \\
0.11 & 0.00 & -0.96 & -1.78 & 0.59 & 1.24 & 0.00 & 0.00 \\
0.12 & 0.80 & 0.00 & -0.41 & -0.33 & 0.21 & 1.12 & -1.03 \\
0.13 & 0.00 & -0.49 & 0.00 & -0.43 & -0.26 & 0.00 & 0.10 \\
1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.36 & 0.00 \\
0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00
\end{bmatrix}$$

$$c^T = [2.5, 1.1, -3.1, -3.5, 1.3, 2.1, 2.3, -1.5]$$

Bibliografia

[1] BARZILAI, J. & BORWEIN, J. Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8(1):141–8, 1988.

[2] BAZARAA, M.; SHERALI, H.; C., SHETTY. *Nonlineat programming: theory and algorithms.* 2 ed. New York, John Wiley, 1993.

[3] BERTSEKAS, D. *Constrined optimization and Lagrange multiplier methods.* New York, Academic Press, 1982.

[4] BERTSEKAS, D. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221–46, 1982.

[5] BETTS, J. & FRANK, P. A sparse nonlinear optimization algorithm. *Journal of Optimization Theory and Applications*, 82(3):519–41, 1994.

[6] BIELSCHOWSKY, R.; FRIEDLANDER, A.; GOMES, F.; MARTÍNEZ, J.; RAYDAN, M. An adaptative algorithm for bound constrained convex quadratic minimization. 1995. Vers o preliminar.

[7] BURKE, J. & MORÉ, J. On the identification of active constraints. *SIAM Journal on Numerical Analysis*, 25(5):1197-211, 1988.

[8] CIARLET, P. *The finite element method for elliptic problems.* Amsterdam, North Holland, 1978.

[9] CONN, A.; GOULD, N.; TOINT, P. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis*, 25(2):433–60, 1988.

[10] CONN, A.; GOULD, N.; TOINT, P. A globally convergent augmented lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–72, 1991.

[11] CONN, A.; GOULD, N.; TOINT, P. Testing a class of methods for solving minimization problems with simple bounds on the variables. *Mathematics of Computation*, 50(2):399–430, 1988.

[12] DEMBO, R. & TULOWITZKI, U. *On the minimization of quadratic functions subject to box constraints.* Working Paper B–71, School of Organization and Management, Yale Univ., New Haven, 1983.

[13] DENNIS, J.; EL-ALEM, M.; MACIEL, M. A global convergence theory for general trust-region-based algorithms for equality constrained optimization. Por aparecer em *SIAM Journal on Optimization.*

[14] DENNIS, J. & SCHNABEL, R. *Numerical methods for unconstrained optimization and nonlinear equations.* Englewood Cliffs, Prentice-Hall, 1983.

[15] DUFF, I.; ERISMAN, A.; REID, J. *Direct methods for sparse matrices.* Oxford, Clarendon, 1986.

[16] FLETCHER, R. *Pratical methods of optimization.* 2 ed. New York, John Wiley, 1987.

[17] FRIEDLANDER, A. & MARTÍNEZ, J. On the maximization of a concave quadratic funtion with box constraints. *SIAM Journal on Optimization*, 4(1):177–92, 1994.

[18] FRIEDLANDER, A. & MARTÍNEZ, J. On the numerical solution of bound constrained optimization problems. *RAIRO Operations Research*, 23(4):319–41, 1989.

[19] FRIEDLANDER, A.; MARTÍNEZ, J.; RAYDAN, M. Gradient method with retards and generalizations. 1994. Submetido a *SIAM Journal on Numerical Analysis.*

[20] FRIEDLANDER, A.; MARTÍNEZ, J.; RAYDAN, M. A new method for large-scale box constrained convex quadratic minimization problems. 1994. Por aparecer em *Optimization Methods and Software.*

[21] FRIEDLANDER, A.; MARTÍNEZ, J.; SANTOS, S. A new trust region algorithm for bound constrained minimization. *Applied Mathematics and Optimization*, 30(3):235–66, 1994.

[22] GILL, P.; MURRAY, W.; WRIGHT, M. *Practical optimization.* London, Academic Press, 1981.

[23] GOLUB, G. & VAN LOAN, C. *Matrix computations.* London, North Oxford Univ., 1986.

[24] GOMES, F.; MACIEL, M.; MARTÍNEZ, J. Successive quadratic programming for minimization with equality and inequality constraints using trust regions, augmented lagrangians and nonmonotone penalty parameters. 1995. Vers o preliminar.

[25] HERMAN, G. *Image reconstruction from projections: the fundamentals of computerized tomography.* New York, Academic Press, 1980.

[26] HESTENES M. amd STIEFEL , E. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards B*, 49:409–36, 1952.

[27] HOCK, W. & SCHITTKOWSKI, K. Test examples for nonlinear programming codes. *Lecture Notes in Economics and Mathematical Systems*, 187, 1981.

[28] LOTSTEDT, P. Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. *SIAM Journal on Scientific and Statistical Computing*, 5(2):370–93, 1984.

[29] LUENBERGER, D. *Linear and nonlinear programming.* 2 ed. Reading, Addison-Wesley, 1984.

[30] MORÉ, J. & TORALDO, G. Algorithms for bound constrained quadratic programming problems. *Numerische Mathematik*, 55(4):377–400, 1989.

[31] MORÉ, J. & TORALDO, G. On the solution of large quadratic programming problems with bound constraints. *SIAM Journal on Optimization*, 1(1):93–113, 1991.

[32] MORÉ, J. & WRIGHT, S. *Optimization software guide.* Philadelphia, SIAM, 1993.

[33] RAYDAN, M. On the Barzilai and Borwein choice of steplength for the gradient method. *IMA Journal of Numerical Analysis*, 13(3):321–6, 1993.

Índice