UM MÉTODO DE RESTRIÇÕES ATIVAS PARA MINIMIZAÇÃO EM CAIXAS

Marina Andretta

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do grau
de
Mestre em Ciência da Computação

Área de Concentração: **Ciência da Computação**
Orientador: **Prof. Dr. Ernesto G. Birgin**

São Paulo, Março de 2004

UM MÉTODO DE RESTRIÇÕES ATIVAS PARA MINIMIZAÇÃO EM CAIXAS

Este exemplar corresponde à redação final
da dissertação de mestrado devidamente
corrigida e defendida por
Marina Andretta
e aprovada pela comissão julgadora.

São Paulo, Março de 2004.

Banca examinadora:

- Prof. Dr. Ernesto G. Birgin (Orientador) - IME-USP
- Prof. Dr. Paulo José da Silva e Silva - IME-USP
- Prof. Dr. Roberto Andreani - IMECC-UNICAMP

Durante a elaboração deste trabalho, a autora recebeu apoio financeiro do CNPq.

Sumário

Introdução

O problema considerado neste trabalho consiste na minimização de uma função $f : \mathbb{R}^n \to \mathbb{R}$ suave com limitantes nas variáveis. O conjunto factível Ω é definido por $\Omega = \{x \in \mathbb{R}^n \mid l \leq x \leq u\}$. Algoritmos para minimização em caixas são usados como subalgoritmos para resolver os subproblemas que aparecem em vários métodos de Lagrangeano aumentado e métodos de penalização para otimização com restrições gerais.

Os métodos de restrições ativas para a minimização de funções com restrições de caixa baseiam-se no seguinte princípio: se conhecemos quais restrições são satisfeitas por igualdade na solução (ativas) e quais não (inativas) podemos fixar algumas variáveis nos seus limitantes e resolver um problema irrestrito nas outras variáveis (variáveis livres). Por isso, os métodos de restrições ativas tentam, a cada iteração, inferir quais restrições serão ativas na solução e resolver um problema localmente irrestrito nas variáveis livres.

Recentemente, foi introduzido um novo método de restrições ativas para a minimização de problemas com restrições de caixa [2]. O algoritmo combina um método irrestrito com Gradiente Espectral Projetado para eliminar restrições do conjunto das variáveis livres. O algoritmo irrestrito inclui uma busca linear que tenta acrescentar muitas restrições no conjunto de restrições ativas a cada iteração. Dentro das faces, utiliza-se um método do tipo Newton truncado que, desenhado para problemas de grande porte, aproxima os produtos hessiana-vetor por quocientes incrementais.

Experimentos computacionais mostraram que esta alternativa não é a mais adequada para problemas de pequeno e médio porte, com hessianas esparsas ou com gradientes computacionalmente custosos. No presente trabalho, estudaremos os métodos de restrições ativas, em particular o proposto em [2], e proporemos e implementaremos uma alternativa para o método utilizado na minimização dentro das faces.

O método alternativo escolhido para a minimização dentro das faces é o método de regiões de confiança combinado com o método do Gradiente Espectral Projetado. Para resolver os subproblemas que aparecem a cada iteração do método de regiões de confiança foi escolhido o método Moré-Sorensen para minimização de quadráticas em bolas descrito em [18]. O método Moré-Sorensen é interessante porque resultados numéricos mostram que métodos que aproximam o minimizador da quadrática na bola (como Dogleg ou a estratégia de somar uma matriz diagonal para contornar o caso de matrizes não positivas definidas) apresentam piores resultados do que os obtidos pela minimização exata (feita pelo método Moré-Sorensen), apesar deste último utilizar mais de uma fatoração de matriz (veja [10] e [1]).

O presente trabalho está organizado como segue. No capítulo 1 tratamos do método apresentado em [18] para a resolução do subproblema que aparece em cada iteração da técnica de regiões de confiança, ou seja, minimizar uma quadrática restrita a uma bola. Analisamos todo o desenvolvimento teórico do método, explicando as propriedades e características do problema a ser resolvido. Reproduzimos todos os lemas apresentados em [18] e enunciamos como novos lemas algumas afirmações de [18] que não nos pareceram triviais. No Apêndice, provamos todos lemas (tanto os de [18] como os por nós enunciados) passo a passo. Apresentamos o algoritmo para a resolução do problema. Por fim, apresentamos os resultados numéricos obtidos pela aplicação da nossa implementação deste algoritmo aos testes propostos em [18].

No capítulo 2 detalhamos a técnica de regiões de confiança e apresentamos um algoritmo para minimização de funções irrestritas que utiliza o método apresentado no capítulo 1 para resolver o subproblema que aparece em cada iteração (tal como sugerido em [18]). Apresentamos os resultados numéricos que nossa implementação deste algoritmo obteve quando aplicado às 18 funções de teste utilizadas em [18] e introduzidas em [17].

No capítulo 3 apresentamos o método do Gradiente Espectral Projetado. No capítulo 4 propomos um método híbrido para minimização nas faces, que mistura iterações dos métodos apresentados nos capítulos 2 e 3. Enunciamos e provamos o teorema de convergência deste método e apresentamos detalhes de nossa implementação.

No capítulo 5 apresentamos o método de restrições ativas para minimização em caixas, que chamamos de BETRA, baseado no método introduzido em [2] e que utiliza o método do Gradiente Espectral Projetado para sair das faces e o algoritmo proposto no capítulo 4 para trabalhar nas faces. Enunciamos e provamos o teorema de convergência de BETRA. Apresentamos, ainda, detalhes de nossa implementação de BETRA e os resultados numéricos de sua aplicação na resolução dos problemas da coleção CUTE (veja [6]). Comparamos seu desempenho com o desempenho de LANCELOT (um conhecido método para a resolução de problemas com restrições de caixa, veja [8]) e constatamos que BETRA mostra-se competitivo, obtendo melhores índices de eficiência e robustez quando comparado com LANCELOT. Apresentamos os resultados obtidos.

Por fim, apresentamos as conclusões e idéias para trabalhos futuros.

Capítulo 1

O método Moré-Sorensen para minimização de quadráticas em bolas

Métodos baseados em regiões de confiança para minimização de uma função f qualquer seguem o seguinte princípio: a cada iteração k, dado um ponto x^k e um raio Δ^k da região, minimiza-se uma aproximação quadrática da função f no ponto x^k restrita a esta região de confiança (ou, $\|x^k\| \leq \Delta^k$).

Ou seja, a cada iteração estamos interessados em resolver problemas do tipo

$$\begin{array}{ll} \text{minimizar} & \varphi(w) \equiv \frac{1}{2} w^T B w + g^T w \\ \text{sujeita a} & \|w\| \leq \Delta^k \end{array} , \tag{1.1}$$

onde Δ^k é um número positivo, $\|.\|$ é a norma euclidiana em $\mathbb{R}^n$, $g \in \mathbb{R}^n$ e $B \in \mathbb{R}^{n \times n}$ é uma matriz simétrica.

Existem vários métodos que fornecem uma solução aproximada para (1.1). No entanto, estamos interessados em soluções exatas, ou pelo menos muito próximas disso.

O método proposto em [18] para resolução deste problema tem justamente este objetivo: encontrar a solução "exata" para (1.1). Para isso, faz-se uso do fato de que a solução p^k de (1.1) é solução de um sistema do tipo $(B + \lambda I)p^k = -g$, com $\lambda \geq 0$, $\lambda(\Delta^k - \|p^k\|) = 0$ e $B + \lambda I$ semidefinida positiva. Assim, o método tenta, em linhas gerais, encontrar um valor de $\lambda \geq 0$ que faça $B + \lambda I$ ser definida positiva e resolve o sistema, encontrando p^k. Se $\lambda = 0$ e $\|p^k\| \leq \Delta^k$ já temos a solução de (1.1). Caso contrário, o valor de λ varia até que se encontre $\|p^k\| = \Delta^k$.

O método proposto em [18] tem alguns detalhes que serão vistos ao longo deste capítulo. Implementamos este método e fizemos os testes propostos em [18], verificando que os resultados obtidos pela nossa implementação foram equivalentes aos apresentados por este artigo. Isso é um indicador de que nossa implementação está correta.

Neste capítulo veremos as propriedades do problema (1.1) e as dificuldades que aparecem em alguns casos. Reproduzimos os lemas de [18] e transformamos em lemas algumas afirmações de [18] que não nos pareceram evidentes. Depois de analisadas as características da solução do problema (1.1), é apresentado o desenvolvimento do algoritmo para a resolução deste problema e sua versão final. Por fim, apresentamos os resultados obtidos pela nossa implementação deste algoritmo aplicados aos testes propostos em [18].

1.1 Fundamentação teórica

A primeira coisa a se saber a respeito do problema (1.1) são os seguintes lemas (todas as provas dos lemas apresentados neste capítulo estão no Apêndice):

Lema 1.1.1 *(Lema 2.1 de [18]) Se p é uma solução para (1.1) então p é uma solução para uma equação da forma*

$$(B + \lambda I)\, p = -g, \tag{1.2}$$

com $B + \lambda I$ semidefinida positiva, $\lambda \geq 0$ e $\lambda(\Delta - \|p\|) = 0$.

Prova : ver Apêndice.

Lema 1.1.2 *(Lema 2.3 de [18]) Sejam $\lambda \in \mathbb{R}$ e $p \in \mathbb{R}^n$ satisfazendo (1.2) com $B + \lambda I$ semidefinida positiva.*

(i) Se $\lambda = 0$ e $\|p\| \leq \Delta$ então p é solução de (1.1);

(ii) p é solução de $\min\{\varphi(w) : \|w\| \leq \|p\|\}$;

(iii) se $\lambda \geq 0$ e $\|p\| = \Delta$ então p é solução de (1.1).

Se $B + \lambda I$ é **definida** *positiva, p é único em (i), (ii) e (iii).*

Prova : ver Apêndice.

Note que, se B for **definida** positiva, B é inversível. Tomando-se $\lambda = 0$, temos que, se $\|B^{-1}g\| \leq \Delta$ então $p = -B^{-1}g$ é solução de (1.1) (pelo Lema 1.1.2-*(i)*). Em outras palavras, se o passo de Newton p está dentro da região delimitada por $D = \{w : \|w\| \leq \Delta\}$, podemos dar este passo e encontrar a solução de (1.1).

Mais ainda:

Lema 1.1.3 *O problema (1.1)* **não** *tem solução na borda de $D = \{w : \|w\| \leq \Delta\}$ se e somente se B é* **definida** *positiva e $\|B^{-1}g\| < \Delta$.*

Prova : ver Apêndice.

Ou seja, o Lema 1.1.3 nos diz que, se a solução de (1.1) não for o passo de Newton, então a solução deverá estar na fronteira de D.

Assim, quando B **não** é **definida** positiva ou o passo de Newton está fora da região D, o Lema 1.1.1 sugere que busquemos uma solução p de (1.2) com $\lambda \geq 0$ e $B + \lambda I$ semidefinida positiva. Como sabemos que, neste caso, a solução está na borda de D (ou seja, $\|p\| = \Delta$), vale que $\lambda(\Delta - \|p\|) = 0$. Além disso, o Lema 1.1.2-*(iii)* garante que p é a solução de (1.1).

Ora, se tivermos um $\lambda \geq 0$ que faz $B + \lambda I$ semidefinida positiva, é fácil ver que existe um λ^* maior ou igual a este λ que faz $B + \lambda^* I$ **definida** positiva.

Vamos definir a função de uma variável $p(\lambda)$ como

$$p(\lambda) = -(B + \lambda I)^{-1} g,$$

para $\lambda \neq -\lambda_j$, onde λ_j são os autovalores da matriz B.

Lembre-se de que estamos interessados na função $p(\lambda)$ apenas para valores de $\lambda > -\lambda$, ou seja, para valores de λ que façam $B + \lambda I$ **definida** positiva. Se, neste intervalo, encontrarmos um $\lambda^* \geq 0$ tal que $\|p(\lambda^*)\| = \Delta$ está claro que teremos uma solução de (1.1).

De fato esta solução sempre existe. Para verificar isso, olhemos mais de perto para a função $p(\lambda)$.

Lema 1.1.4 *Seja $p(\lambda) = -(B + \lambda I)^{-1} g$ com $B + \lambda I$ definida positiva.*
Tome a decomposição $B = Q \Lambda Q^T$, com $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ e $Q^T Q = I$, onde $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ são os autovalores de B (que existe porque B é simétrica).
Então, para $\lambda \neq -\lambda_j$,

$$p(\lambda) = -\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j,$$

onde q_j é a j-ésima coluna de Q.

Prova : ver Apêndice.

Pelo Lema 1.1.4 podemos ver que o quadrado da norma euclidiana função $p(\lambda)$ é dado por

$$\|p(\lambda)\|^2 = \left(\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j \right)^T \left(\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j \right).$$

Como $Q^T Q = I$, temos que $(q_i)^T q_i = 1$ para todo i e $(q_j)^T q_i = 0$ para todo $j \neq i$.

Assim,

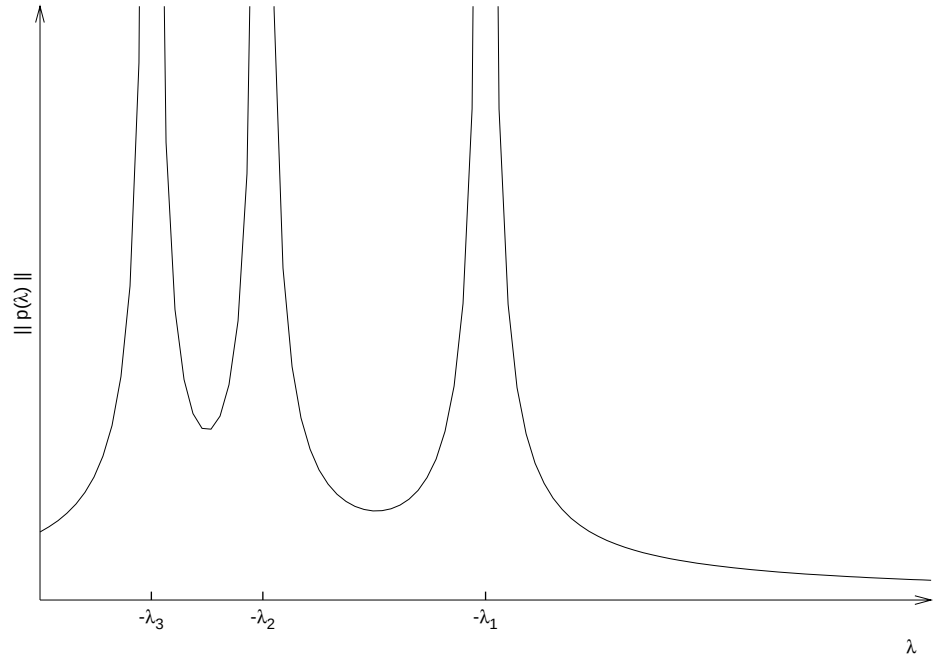

Figura 1.1: caso em que $q_1^T g \neq 0$

$$\|p(\lambda)\|^2 = \sum_{j=1}^{n} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2}. \tag{1.3}$$

Note que, como a norma é positiva e $\Delta > 0$, quando usamos relações do tipo $\|p(\lambda)\| = \Delta$, podemos também usar $\|p(\lambda)\|^2 = \Delta^2$.

Observe que

$$\lim_{\lambda \to \infty} \|p(\lambda)\|^2 = 0, \tag{1.4}$$

e, para $q_j^T g \neq 0$,

$$\lim_{\lambda \to -\lambda_j} \|p(\lambda)\|^2 = \infty. \tag{1.5}$$

Veja na figura 1.1 um desenho da função de $\|p(\lambda)\|$ para $q_j^T g \neq 0$.

Note que, se $\lambda > -\lambda_1$ temos que $\lambda + \lambda_j > 0$ para todo $j = 1, ..., n$. Portanto, no intervalo $(-\lambda_1, \infty)$ a função $\|p(\lambda)\|$ é contínua, decrescente e côncava.

Vamos definir a função

$$\phi_1(\lambda) = \|p(\lambda)\| - \Delta.$$

Agora podemos dividir o problema em 4 casos:

(i) Quando B é definida positiva e $\|B^{-1}q\| \leq \Delta$. Neste caso, como já vimos, a solução de (1.1) é dada por $p(0) = -B^{-1}g$;

(ii) Quando B é definida positiva mas $\|B^{-1}q\| > \Delta$. Neste caso, se usarmos $\lambda = 0$, temos $\|B^{-1}q\| > \Delta$. Então é preciso que se aumente o valor de λ para que a norma de $p(\lambda)$ diminua (veja a equação (1.3)).

Como $-\lambda_1 < 0$ (porque B é definida positiva), não precisamos nos preocupar com os pontos de descontinuidade de $\|p(\lambda)\|$, pois podemos procurar uma raiz λ^* de $\phi_1(\lambda)$ no intervalo $(0, \infty)$.

Pelo Lema 1.1.2-*(iii)*, a solução de (1.1) é dada por $p(\lambda^*) = -(B + \lambda^* I)^{-1}g$;

(iii) Quando B não é definida positiva e $q_1^T g \neq 0$. Neste caso, valem os limites (1.4) e (1.5). Então sabemos que existe um valor no intervalo $(-\lambda_1, \infty)$ em que $\|p(\lambda)\| = \Delta$ (pois neste intervalo a função é contínua e assume valores no intervalo $(0, \infty)$, ao qual pertence Δ).

Como B não é definida positiva, temos que $\lambda_1 \leq 0$. Assim, afirmamos que, além de existir uma raiz λ^* de $\phi_1(\lambda)$, $\lambda^* > -\lambda_1 \geq 0$.

Neste caso, a solução de (1.1) é dada por $p(\lambda^*) = -(B+\lambda^* I)^{-1}g$ (pelo Lema 1.1.2-*(iii)*);

(iv) Quando B não é definida positiva e $q_1^T g = 0$. Este é o **caso difícil**. Neste caso não existe o limite (1.5) (veja o desenho de uma função $\|p(\lambda)\|$ para $q_1^T g = 0$ na figura 1.2). Então não temos garantia de que a raiz de $\phi_1(\lambda)$ esteja no intervalo $(-\lambda_1, \infty)$.

Pelo Lema 1.1.1, temos que a solução de (1.1) é solução para uma equação do tipo (1.2) com $B + \lambda I$ semidefinida positiva.

Ora, para que $B + \lambda I$ seja semidefinida positiva, temos que $\lambda \in [-\lambda_1, \infty)$. Como não temos, necessariamente, a raiz λ^* de $\phi_1(\lambda)$ no intervalo $(-\lambda_1, \infty)$, esta raiz pode estar exatamente em $-\lambda_1$. Neste caso, para qualquer $\lambda > -\lambda_1$, $\|p(\lambda)\| < \Delta$.

Mas, com $\lambda^* = -\lambda_1$, temos que $B + \lambda^* I = B - \lambda_1 I$, que é singular. Ou seja, existe um $z \neq 0$ tal que

$$(B - \lambda_1 I)z = 0 \Rightarrow Bz - \lambda_1 z = 0 \Rightarrow Bz = \lambda_1 z,$$

ou seja, z é o autovetor de B associado ao autovalor λ_1.

Podemos, sem perda de generalidade, usar $z = \tau\hat{z}$, para $\|\hat{z}\| = 1$.

Note que, se p satisfaz a equação (1.2) então $p + \tau\hat{z}$ também a satisfaz.

Note que, neste caso, podemos escrever

$$p = \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j + \tau \hat{z},$$

para todo $\tau \in I\!R$.

Assim,

$$\|p\|^2 = \left(\sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j + \tau \hat{z} \right)^T \left(\sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j + \tau \hat{z} \right) =$$

$$\left\| \sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j \right\|^2 + \left(\sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j \right)^T (\tau \hat{z}) + \|\tau \hat{z}\|^2.$$

Como $\hat{z}$ é ortogonal a q_j, temos que

$$\left(\sum_{j:\lambda_j \neq \lambda_1} \frac{q_j^T g}{\lambda_j - \lambda_1} q_j \right)^T (\tau \hat{z}) = 0.$$

Ainda pela ortogonalidade de Q e por $\|\hat{z}\| = 1$, temos que

$$\|p\|^2 = \sum_{j:\lambda_j \neq \lambda_1} \frac{(q_j^T g)^2}{(\lambda_j - \lambda_1)^2} + \tau^2,$$

para todo $\tau \in I\!R$.

Desta forma, sempre é possível conseguir τ^* de tal forma que $\|p\| = \Delta$ para $\lambda = -\lambda_1$. Neste caso, o Lema 1.1.2-*(iii)* diz que a solução de (1.1) é p, ou seja, $p(-\lambda_1) + \tau^* \hat{z}$, para algum τ^*.

1.2 O algoritmo

1.2.1 Uma primeira versão do algoritmo

Nesta seção, analisamos mais alguns resultados teóricos apresentados em [18]. Reproduzimos as provas de alguns lemas (no Apêndice), detalhamos outras e provamos algumas afirmações que não nos pareceram evidentes.

Agora que conhecemos bem a estrutura do problema, podemos pensar em algoritmos para resolvê-lo.

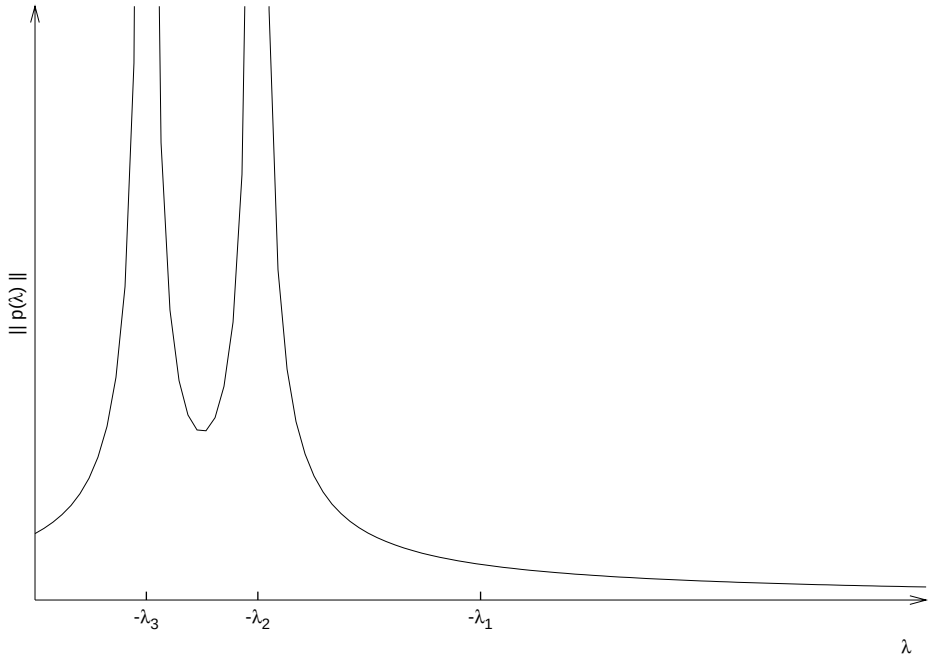

Figura 1.2: caso em que $q_1^T g = 0$

Na maioria dos casos, o Método de Newton para encontrar raízes de uma função pode ser aplicado em $\phi_1(\lambda)$. No entanto, se definirmos

$$\phi(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}$$

podemos obter convergência melhor do Método, porque no intervalo $(-\lambda_1, \infty)$ a função $\phi(\lambda)$ é quase linear.

A cada passo do Método de Newton define-se

$$\lambda^{k+1} = \lambda^k - \frac{\phi(\lambda)}{\phi'(\lambda)}$$

Lema 1.2.1 *Seja* $\phi(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}$, *onde* $p(\lambda) = -(B + \lambda I)^{-1} g$ *com* B *simétrica e* $B + \lambda I$ *definida positiva.*

Então,

$$-\frac{\phi(\lambda)}{\phi'(\lambda)} = \left(\frac{\|p(\lambda)\|}{\|t(\lambda)\|}\right)^2 \left(\frac{\|p(\lambda)\| - \Delta}{\Delta}\right),$$

onde $t(\lambda)$ *é a solução de* $R^T t(\lambda) = p(\lambda)$ *e* R *é a matriz triangular superior tal que* $B + \lambda I = R^T R$.

Prova : ver Apêndice.

Com base no Lema 1.1.2, podemos escrever o algoritmo de Newton:

Algoritmo 1 *Dados* $\Delta > 0$ *e* $\lambda^0 \geq 0$ *com* $B + \lambda^0 I$ *definida positiva.*

Passo 1. $k = 0$

Passo 2. Fatore $B + \lambda^k I = R^T R$

Passo 3. Resolva $R^T R p_k = -g$ em p_k

Passo 4. Se o "critério de convergência foi satisfeito" então

Pare e devolva p_k como solução

Passo 5. Resolva $R^T t_k = p_k$ em t_k

Passo 6. $\lambda^{k+1} = \lambda^k + \left(\frac{\|p_k\|}{\|t_k\|}\right)^2 \left(\frac{\|p_k\| - \Delta}{\Delta}\right)$

Passo 7. $k = k + 1$

Passo 8. volte para o **Passo 2**

É claro que é necessário salvaguardar λ^k para que $B + \lambda^k I$ seja definida positiva a cada passo e para que tenhamos garantia de convergência do método.

Mas, antes de nos focar na salvaguarda de λ^k e nos critérios de convergência do algoritmo, concentremo-nos no **caso difícil**. Note em especial que, quando temos $g = 0$, o algoritmo está mal-definido.

Como visto anteriormente, para a resolução do **caso difícil** é necessário fornecer z que é o autovetor associado ao autovalor λ_1. Como é computacionalmente caro calcular o autovetor, calcularemos uma aproximação deste.

1.2.2 Cálculo da aproximação de z

Vejamos o lema que segue:

Lema 1.2.2 *(Lema 3.4 de [18]) Seja* $0 < \sigma < 1$ *dado e suponha que*

$$B + \lambda I = R^T R, \quad (B + \lambda I)p = -g, \quad \lambda \geq 0.$$

Seja $z \in I\!R^n$ *tal que*

$$\|p + z\| = \Delta, \quad \|Rz\|^2 \leq \sigma(\|Rp\|^2 + \lambda\Delta^2). \tag{1.6}$$

Então

$$-\varphi(p + z) \geq \frac{1}{2}(1 - \sigma)(\|Rp\|^2 + \lambda\Delta^2) \geq (1 - \sigma) \mid \varphi^* \mid,$$

onde φ^* é o valor ótimo de (1.1).

Prova : ver Apêndice.

Veja que, como conseqüência do Lema 1.2.2, temos

$$-\varphi(p+z) \geq (1-\sigma) \mid \varphi^* \mid \;\Rightarrow\; -\varphi(p+z) - \mid \varphi^* \mid \geq -\sigma \mid \varphi^* \mid \;\Rightarrow$$

$$\sigma \mid \varphi^* \mid \geq \; \varphi(p+z) + \mid \varphi^* \mid \geq \; \varphi(p+z) - \varphi^*.$$

Como $\varphi(p+z) - \varphi^* \geq 0$, temos que

$$\mid \varphi(p+z) - \varphi^* \mid \leq \; \sigma \mid \varphi^* \mid .$$

Ou seja, se vale (1.6), $p+z$ é uma solução próxima da solução ótima de (1.1). Mais ainda, quanto mais próximo de 0 está $\|R\hat{z}\|$, mais próximo $p+z$ está da solução ótima de (1.1).

O Lema 1.2.2 é conveniente no **caso difícil**, quando vale que $B + \lambda I = R^T R$ (definida positiva), $(B+\lambda I)p = -g$, $\lambda \geq 0$ e $\|p\| < \Delta$. Neste caso temos λ próximo de $-\lambda_1$.

Então estamos interessados num algoritmo que calcule um vetor $\hat{z}$, com $\|\hat{z}\| = 1$, de tal forma que, quando λ se aproxima de $-\lambda_1$, $\|R\hat{z}\|$ se aproxima de 0. Calculado $\hat{z}$, podemos calcular τ de maneira que $\|p + \tau\hat{z}\| = \Delta$.

Note que a solução ótima de (1.1) é dada por $p+z$, com $(B-\lambda_1)p = -g$ e z o autovetor de B associado a λ_1. Com $\hat{z}$ calculado como dito acima, temos que $\tau\hat{z}$ se aproxima do autovetor z.

Uma maneira de calcular $\hat{z}$ é usando uma técnica introduzida em [7] para estimar o menor valor singular de uma matriz triangular R. O Lema 1.2.3 mostra que esta técnica satisfaz os requisitos apresentados acima. Nesta técnica, um vetor e com componentes 1 ou -1 é escolhido de maneira que a solução w do sistema $R^T w = e$ seja grande. A idéia é escolher o sinal das componentes de e de maneira a obter máximo crescimento local de w durante a resolução do sistema. Depois, resolve-se o sistema $Rv = w$ para v. O vetor v é tal que, se $\hat{z}$ for definido como $\hat{z} = \frac{v}{\|v\|}$, então $\|R\hat{z}\|$ se aproxima do menor valor singular de R. Essa técnica é atrativa porque não é computacionalmente cara (gasta em torno de n^2 operações aritméticas) e, por isso, a implementamos.

Lema 1.2.3 *Seja $(B + \lambda I) = R^T R$ uma matriz definida positiva. Sejam $v \in I\!R^n$ o vetor calculado utilizando a técnica descrita em [7] e $\hat{z} = \frac{v}{\|v\|}$. Então $\|R\hat{z}\|$ se aproxima de 0 quando λ se aproxima de $-\lambda_1$, com λ_1 o menor autovalor de B.*

Prova : ver Apêndice.

Dado que estamos interessados no caso em que λ se aproxima de $-\lambda_1$, a matriz R é quase singular e isso poderia levar a dificuldades numéricas. O critério de convergência (1.8) (apresentado na seção 1.2.4) garante que o algoritmo pára antes que dificuldades numéricas apareçam.

Calculado $\hat{z}$, temos 2 possíveis escolhas para o valor de τ que satisfazem $\|p + \tau\hat{z}\| = \Delta$, a saber

$$\|p + \tau\hat{z}\| = \Delta \Rightarrow \|p + \tau\hat{z}\|^2 = \Delta^2 \Rightarrow \|p + \tau\hat{z}\|^2 - \Delta^2 = 0 \Rightarrow$$

$$\|p\|^2 + 2\tau(p^T\hat{z}) + \tau^2\|\hat{z}\|^2 - \Delta^2 = 0.$$

Como $\|\hat{z}\| = 1$, temos que

$$\tau^2 + 2\tau(p^T\hat{z}) + \|p\|^2 - \Delta^2 = 0 \Rightarrow$$

$$\tau = \frac{-2(p^T\hat{z}) \pm \sqrt{4((p^T\hat{z})^2 - (\|p\|^2 - \Delta^2))}}{2} \Rightarrow$$

$$\tau = -p^T\hat{z} \pm \sqrt{(p^T\hat{z})^2 - (\|p\|^2 - \Delta^2)}.$$

Pela equação (A.10), a escolha de τ com menor magnitude fornece um valor menor de φ.

Esta escolha, portanto, é

$$\tau = -p^T\hat{z} + \text{sgn}(p^T\hat{z})\sqrt{((p^T\hat{z})^2 - (\|p\|^2 - \Delta^2))}.$$

1.2.3 Salvaguarda de λ^k

Agora que temos todas as ferramentas para calcular $\hat{z}$ e τ, vamos voltar ao problema da salvaguarda de λ^k.

A salvaguarda de λ^k depende do fato de que $\phi(\lambda)$ é convexa e estritamente decrescente no intervalo $(-\lambda_1, \infty)$. Isso nos mostra que o Método de Newton, quando aplicado à função ϕ, partindo de $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) > 0$ gera uma seqüência monotonicamente crescente que converge para a solução $\phi(\lambda^*) = 0$. Além disso, se $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) < 0$, temos que ou

$\lambda^{k+1} \leq -\lambda_1$ ou $\phi(\lambda^{k+1}) > 0$.

Está claro que devemos manter os valores de λ^k no intervalo $(-\lambda_1, \infty)$, que é justamente onde está a solução λ^*.

Manteremos, então, 3 parâmetros: λ_L, λ_U e λ_S. O intervalo $[\lambda_L, \lambda_U]$ conterá a solução λ^* e será reduzido a cada iteração. λ_S será um limitante inferior para $-\lambda_1$.

Para a salvaguarda de λ^k temos os seguintes passos:

Passo 1. $\lambda^k = \max\{\lambda^k, \lambda_L\}$

Passo 2. $\lambda^k = \min\{\lambda^k, \lambda_U\}$

Passo 3. Se $\lambda^k \leq \lambda_S$ então $\lambda^k = \max\{0.001\lambda_U, \sqrt{\lambda_L \lambda_U}\}$

Claramente os **Passos 1** e **2** colocam λ^k no intervalo $[\lambda_L, \lambda_U]$. O **Passo 3** permite que se reduza o tamanho do intervalo de incerteza. Este tipo de atualização foi já usado em outras ocasiões (veja [11]) e os resultados foram satisfatórios. Note que o tamanho do intervalo só é reduzido quando $\lambda_S \in [\lambda_L, \lambda_U]$. Se o tamanho do intervalo de incerteza fica longe de 0, o **Passo 3** pode ser executado apenas um número finito de vezes. Isso ficará claro com as regras de atualização de λ_S, λ_L e λ_U. No final deste capítulo veremos a importância deste fato.

Veja agora como obter valores iniciais para λ_L, λ_U e λ_S e como atualizá-los.

1.2.3.1 Valores iniciais de λ_L, λ_U e λ_S

Encontrar λ_1 o menor autovalor da matriz simétrica B equivale a encontrar o mínimo de

$$\frac{x^T B x}{x^T x},$$

onde $x \in \mathbb{R}^n$ não é nulo (veja, por exemplo, [12]).

Note que, se escolhermos x como o vetor canônico e_i (componente i é igual a 1 e as restantes iguais a 0), temos que

$$\frac{e_i^T B e_i}{e_i^T e_i} = b_{ii} \geq \lambda_1, \text{ para } i = 1, ..., n,$$

onde b_{ii} é o i-ésimo elemento da diagonal de B.

Assim,

$$\min_{1 \leq i \leq n} \{b_{ii}\} \geq \lambda_1 \Rightarrow -\min_{1 \leq i \leq n} \{b_{ii}\} \leq -\lambda_1 \Rightarrow \max_{1 \leq i \leq n} \{-b_{ii}\} \leq -\lambda_1.$$

Portanto, podemos começar o algoritmo com

$$\lambda_S = \max_{1 \leq i \leq n} \{-b_{ii}\}.$$

Para escolhermos um valor inicial para λ_L e λ_U, veja o seguinte lema:

Lema 1.2.4 *Tome B, g e Δ do problema (1.1).*

Um limitante inferior λ_L para λ^* solução de $\phi(\lambda)$ é

$$\lambda_L = \frac{\|g\|}{\Delta} - \|B\|_1,$$

e um limitante superior λ_U é

$$\lambda_U = \frac{\|g\|}{\Delta} + \|B\|_1.$$

Prova : ver Apêndice.

Então já temos uma maneira de escolher os valores iniciais de λ_U e λ_S. Lembre-se que, como visto na seção anterior, a solução λ^* que procuramos, além de estar no intervalo $(-\lambda_1, \infty)$ é maior ou igual a 0.

Portanto, como valor inicial de λ_L temos

$$\lambda_L = \max\{0, \lambda_S, \frac{\|g\|}{\Delta} - \|B\|_1\}.$$

1.2.3.2 Atualização dos valores de λ_L, λ_U e λ_S

Vejamos primeiro como atualizar o valor de λ_S.

Note que, quando $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) < 0$, temos $\|p\| < \Delta$, e podemos calcular $\hat{z}$ e τ como apresentado na seção 1.2.2.

Veja agora o seguinte lema:

Lema 1.2.5 *Seja $B + \lambda I$ uma matriz definida positiva tal que $B + \lambda I = R^T R$. Para qualquer $\hat{z}$ tal que $\|\hat{z}\| = 1$ vale que*

$$\lambda - \|R\hat{z}\|^2 \leq -\lambda_1.$$

Prova : ver Apêndice.

Então, neste caso, podemos tomar λ_S como

$$\lambda_S = \max\{\lambda_S, \lambda^k - \|R\hat{z}\|^2\},$$

já que λ_S também era um limitante inferior para $-\lambda_1$.

No caso em que $\lambda^k \leq -\lambda_1$ temos $B + \lambda^k I$ indefinida. Desta forma, não é possível fazer a decomposição de Cholesky.

Porém, note que, durante a tentativa de decomposiçao, pode-se calcular $\delta \geq 0$ de forma que

$$B + \lambda^k I + \delta e_l e_l^T$$

seja singular e l é exatamente a posição da matriz $B + \lambda^k I$ onde a decomposição falha.

Ainda é possível determinar o vetor $u \in \mathbb{R}^l$ tal que

$$(B^l + \lambda^k I + \delta e_l e_l^T)u = 0,$$

com B^l a submatriz com as primeiras l linhas e colunas de B e $u_l = 1$.

Agora veja o lema a seguir:

Lema 1.2.6 *Seja $\delta \geq 0$ tal que $B + \lambda I + \delta e_l e_l^T$ é singular, com e_l a l-ésima coluna da matriz identidade. Seja $u \in \mathbb{R}^l$ tal que $(B^l + \lambda I + \delta e_l e_l^T)u = 0$, com $u_l = 1$.*
Então

$$\lambda + \frac{\delta}{\|u\|^2} \leq -\lambda_1.$$

Prova : ver Apêndice.

Então, neste caso, podemos tomar λ_S como sendo

$$\lambda_S = \max\left\{\lambda_S, \lambda^k + \frac{\delta}{\|u\|^2}\right\}.$$

A atualização de λ_L e λ_U é a esperada. Desta maneira, para atualizar os valores de λ_L, λ_U e λ_S basta seguir os seguintes passos:

Passo 1. Se $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) < 0$ então

$\lambda_U = \min\{\lambda_U, \lambda^k\}$

$\lambda_S = \max\{\lambda_S, \lambda^k - \|R\hat{z}\|^2\}$

Senão

$$\lambda_L = \max\{\lambda_L, \lambda^k\}$$

Passo 2. Se $\lambda^k \leq -\lambda_1$ então

compute δ e $\|u\|^2$

$$\lambda_S = \max\left\{\lambda_S, \lambda^k + \frac{\delta}{\|u\|^2}\right\}$$

Passo 3. $\lambda_L = \max\{\lambda_L, \lambda_S\}$

1.2.4 Critérios de convergência

Agora, para que o algoritmo fique completo, basta apenas definir os critérios de convergência. Queremos que ele pare com uma solução próxima do ótimo do problema (1.1).

Dados $\sigma_1, \sigma_2 \in (0,1)$ e $\lambda^k \geq 0$ tal que $B + \lambda^k I$ é definida positiva, calcula-se p usando o Método de Newton (Algoritmo 1). Se

$$| \Delta - \|p\| | \leq \sigma_1 \Delta, \text{ ou } \|p\| \leq \Delta, \lambda^k = 0 \tag{1.7}$$

então o algoritmo pára com $s = p$ como solução do problema (1.1).

No **caso difícil**, sempre que $\|p\| < \Delta$ (ou seja, sempre que $\phi(\lambda^k) < 0$), $\hat{z}$ é calculado. Neste caso, quando

$$\|R(\tau\hat{z})\|^2 \leq \sigma_1(2-\sigma_1)\max\{\sigma_2, \|Rp\|^2 + \lambda^k\Delta^2\} \tag{1.8}$$

o algoritmo pára com $s = p + \tau\hat{z}$ como solução do problema (1.1).

Quando tanto (1.7) como (1.8) são satisfeitos, devemos escolher entre p e $p + \tau\hat{z}$ o que fornece menor valor de φ.

Isso é fácil de verificar. Olhando para a equação (A.10) (da prova do Lema 1.2.2), é fácil ver que

$$\varphi(p + \tau\hat{z}) \leq \varphi(p) \Leftrightarrow \|R(\tau\hat{z})\|^2 \leq \lambda^k(\Delta^2 - \|p\|^2).$$

Veremos agora que, quando o algoritmo pára, se o critério (1.7) ou (1.8) for satisfeito, o algoritmo de fato termina com uma solução s próxima do ótimo para o problema (1.1), com s tal que

$$\varphi(s) - \varphi^* \leq \sigma_1(2-\sigma_1)\max\{| \varphi^* |, \sigma_2\}, \quad \|s\| \leq (1+\sigma_1)\Delta. \tag{1.9}$$

Primeiro analisemos o caso em que o critério (1.7) é satisfeito.

Lema 1.2.7 *(Lema 3.13 de [18]) Seja* $0 < \sigma < 1$ *dado e suponha que*

$$B + \lambda I = R^T R, \quad (B + \lambda I)p = -g, \quad \lambda \geq 0.$$

Se $\|p\| \geq (1 - \sigma)\Delta$ *então*

$$-\varphi(p) \geq \frac{1}{2}(1 - \sigma)^2(\|Rp\|^2 + \lambda\Delta^2) \geq (1 - \sigma)^2 \mid \varphi^* \mid,$$

onde φ^* *é o valor ótimo de (1.1).*

Prova : ver Apêndice.

Daí segue que, se o critério (1.7) for satisfeito então a solução fornecida pelo algoritmo satisfaz (1.9) (para verificar basta fazer algumas manipulações na inequação que resulta do Lema 1.2.7).

Agora vejamos o caso em que o (1.8) é satisfeito.

Se $\|Rp\|^2 + \lambda^k\Delta > \sigma_2$ então as hipóteses do Lema 1.2.2 valem para $\sigma = \sigma_1(2 - \sigma_1)$ e, portanto, (1.9) vale para $s = p + \tau\hat{z}$ (basta fazer as mesmas manipulações da inequação que resulta do Lema 1.2.2 que foram feitas para verificar o critério anterior).

Suponha agora que $\|Rp\|^2 + \lambda^k\Delta \leq \sigma_2$. Se $\varphi^* = \varphi(p + z^*)$ com $\|p + z^*\| \leq \Delta$ então a equação (A.10) (da prova do Lema 1.2.2) nos diz que

$$\mid \varphi^* \mid = -\varphi^* \leq \frac{1}{2}(\|Rp\|^2 + \lambda^k\Delta^2) \leq \frac{1}{2}\sigma_2.$$

Usando novamente (A.10), temos que

$$\varphi(p + \tau\hat{z}) = -\frac{1}{2}(\|Rp\|^2 + \lambda^k\Delta^2) + \frac{1}{2}\|R(\tau\hat{z})\|^2 \leq \varphi^* + \frac{1}{2}\sigma(2 - \sigma 1)\sigma_2.$$

Ou seja, quando o critério (1.8) é satisfeito, também vale (1.9).

1.2.5 Versão final do algoritmo

Agora que já dispomos de todos os elementos para a construção de um algoritmo para chegar à solução do problema (1.1), vamos escrevê-lo por completo.

Algoritmo 2 *Dados uma matriz simétrica* B*, um vetor* g*,* $\Delta > 0$*,* $\lambda^0 \geq 0$*,* $\sigma_1, \sigma_2 \in (0, 1)$

Passo 1. *Valores iniciais de* λ_L*,* λ_U*,* λ_S *e* k

$\lambda_S = \max_{1 \leq i \leq n}\{-b_{ii}\}$

$\lambda_L = \max\{0, \lambda_S, \frac{\|g\|}{\Delta} - \|B\|_1\}$.

$\lambda_U = \frac{\|g\|}{\Delta} + \|B\|_1$

$k = 0$

Passo 2. *Salvaguarda de* λ^k

$\lambda^k = \max\{\lambda^k, \lambda_L\}$

$\lambda^k = \min\{\lambda^k, \lambda_U\}$

Se $\lambda^k \leq \lambda_S$ então $\lambda^k = \max\{0.001\lambda_U, \sqrt{\lambda_L \lambda_U}\}$

Passo 3. *Tentativa de fatoração de* $B + \lambda^k I$

Use Cholesky para fatorar $B + \lambda^k I = R^T R$

Se a fatoração não foi possível então

compute δ e $\|u\|^2$ (como descrito na seção 1.2.3.2)

$\lambda_S = \max\left\{\lambda_S, \lambda^k + \frac{\delta}{\|u\|^2}\right\}$

vá para o **Passo 6**

Passo 4. Resolva $R^T R p_k = -g$ em p_k

Passo 5. *Cálculo da aproximação de* z

Se $\|p_k\| < \Delta$ então

calcule $\hat{z}$ usando a técnica de [7]

$\tau = -p_k^T \hat{z} + \text{sgn}(p_k^T \hat{z}) \sqrt{((p_k^T \hat{z})^2 - (\|p_k\|^2 - \Delta^2))}$

Passo 6. *Atualização dos valores de* λ_L, λ_U, λ_S

Se $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) < 0$ então

$\lambda_U = \min\{\lambda_U, \lambda^k\}$

$\lambda_S = \max\{\lambda_S, \lambda^k - \|R\hat{z}\|^2\}$

Senão

$\lambda_L = \max\{\lambda_L, \lambda^k\}$

$\lambda_L = \max\{\lambda_L, \lambda_S\}$

Passo 7. *Critérios de convergência*

Passo 7.1 Se $\|p_k\| \leq \Delta$ e $\lambda^k = 0$ então

Pare e devolva p_k como solução

Passo 7.2 Teste se $\mid \Delta - \|p_k\| \mid \leq \sigma_1 \Delta$

Passo 7.3 Teste se $\|R(\tau\hat{z})\|^2 \leq \sigma_1(2-\sigma_1)\max\{\sigma_2, \|Rp_k\|^2 + \lambda^k\Delta^2\}$

Passo 7.4 *Decisão do critério a ser aceito*

Se **apenas** o **Passo 7.2** foi satisfeito então

Pare e devolva p_k como solução

Se **apenas** o **Passo 7.3** foi satisfeito então

Pare e devolva $p_k + \tau\hat{z}$ como solução

Se tanto o **Passo 7.2** como o **Passo 7.3** foram satisfeitos então

Se $\|R(\tau\hat{z})\|^2 \leq \lambda^k(\Delta^2 - \|p_k\|^2)$ então

Pare e devolva $p_k + \tau\hat{z}$ como solução

Senão

Pare e devolva p_k como solução

Passo 8. *Atualização de λ^k e k*

Se foi possível a fatoração de Cholesky e $g \neq 0$ então

Resolva $R^T t_k = p_k$ em t_k

$\lambda^{k+1} = \lambda^k + \left(\frac{\|p_k\|}{\|t_k\|}\right)^2 \left(\frac{\|p_k\| - \Delta}{\Delta}\right)$

Senão

$\lambda^{k+1} = \lambda_S$

$k = k + 1$

Volte para o **Passo 2**

Depois de um número finito de iterações o Algoritmo 2 pára com $\lambda^k \in (-\lambda_1, \infty)$ e $\phi(\lambda^k) \geq 0$ ou com um intervalo de incerteza arbitrariamente pequeno. Vejamos o porquê.

O primeiro ponto a ser observado é que a terceira atribuição do **Passo 2** faz com que o intervalo de incerteza seja reduzido quando $\lambda_S \in [\lambda_L, \lambda_U]$. Quando este passo é executado, λ^k é "jogado" dentro do intervalo aberto (λ_L, λ_U) (quando $\lambda_L \neq \lambda_U$). No **Passo 6**, com a atualização dos valores de λ_L e λ_U, temos o intervalo de incerteza reduzido. Como λ_S é um limitante inferior para $-\lambda_1$, essa diminuição do tamanho do intervalo só pode acontecer enquanto $-\lambda_1$ ainda for candidato à raiz de $\phi(\lambda)$.

Se $-\lambda_1$ é a raiz de $\phi(\lambda)$, o intervalo será reduzido até que fique arbitrariamente pequeno, contendo $-\lambda_1$. Neste caso, λ^k ficará próximo de $-\lambda_1$ e, portanto, R será quase singular,

fazendo com que (1.8) seja satisfeito e o Algoritmo 2 pare.

Quando a raiz de $\phi(\lambda)$ não está em $-\lambda_1$, teremos, em um número finito de iterações, $-\lambda_1 < \lambda_L$. Ou seja, λ_S sairá do intervalo de incerteza e nunca mais voltará. Neste ponto, teremos $\lambda^k \in (-\lambda_1, \infty)$. Se $\phi(\lambda^k) < 0$, o Método de Newton garante que teremos $\phi(\lambda^k) \geq 0$. Ou seja, (1.7) será satisfeito e o Algoritmo 2 pára.

1.3 Resultados numéricos

Implementamos o Algoritmo 2 numa subrotina de FORTRAN 77, chamada MEQB. Para que esta subrotina seja testada, é preciso construir uma instância do problema (1.1). Para isso, é preciso fornecer a matriz simétrica B, o vetor g e o escalar Δ. Além disso, também é necessário que se estabeleça o valor inicial de λ.

Em [18] é sugerida uma maneira para se construir a matriz B, o vetor g e o valor inicial de λ de forma sistemática e conveniente. Utilizamos estes testes para verificar se a subrotina MEQB apresentava o mesmo comportamento da implementação mencionada em [18], o que nos levaria a acreditar que nossa implementação de fato estava correta. Explicamos a construção das instâncias para testes abaixo.

Para o valor inicial de λ, uma escolha razoável é

$$\lambda = \frac{\|g\|}{\Delta}.$$

Uma maneira para se construir a matriz B é, para alguma matriz ortogonal Q e uma matriz diagonal D, calcular $B = QDQ^T$. Assim, calculando $g = Q\hat{g}$, para algum $\hat{g}$, podemos gerar uma instância potencialmente pertencente ao **caso difícil** preenchendo com 0 a componente de $\hat{g}$ que corresponde ao menor elemento de D.

Obtemos a matriz ortogonal Q de forma que $Q = Q_1Q_2Q_3$, onde

$$Q_j = I - 2\frac{w_j w_j^T}{\|w_j\|^2}, j = 1, 2, 3,$$

onde as componentes de w_j são uniformemente distribuídas entre $(-1, 1)$.

Para se obter valores pseudo-aleatórios, uniformemente distribuídos, usou-se a função RAND apresentada em [21]. Dada uma semente, esta função gera um número pseudo-aleatório no intervalo $(0, 1)$ e muda o valor da semente. Assim, sucessivas chamadas à função RAND geram números pseudo-aleatórios, que podem ser reproduzidos se a semente usada inicialmente for mantida.

Agora, basta especificar D, $\hat{g}$ e Δ. A escolha desses parâmetros levam a diferentes tipos de problemas.

Em [18] são sugeridos 4 tipos de problemas: o caso geral, o **caso difícil**, o caso definido positivo e o caso de ponto de sela.

Para a obtenção de qualquer um dos tipos de problemas, tanto D como $\hat{g}$ têm, inicialmente, suas componentes uniformemente distribuídas entre $(-1, 1)$. Isso basta para se gerar uma instância do caso geral. Para se obter uma instância do **caso difícil**, atribui-se 0 à componente de $\hat{g}$ que correspondente à menor componente de D. Para o caso definido positivo, basta trocar $D = diag(d_1, d_2, ..., d_n)$ por $\bar{D} = diag(\mid d_1 \mid, \mid d_2 \mid, ..., \mid d_n \mid)$. Para o caso de ponto de sela, é atribuído 0 a todas as componentes de $\hat{g}$.

Para valores de Δ no intervalo $(0, 1)$, a escolha sugerida para o λ inicial é muito conveniente, deixando o teste mais fácil. Por isso, foram escolhidos, para os testes, valores de Δ uniformemente distribuídos em $(0, 100)$.

Nas tabelas 1.1, 1.2, 1.3 e 1.4 são apresentados os dados obtidos na execução dos testes para cada um dos casos. Em todos eles foram testadas instâncias de dimensão 10, 20, 40, 60, 80 e 100. Para cada um dos tamanhos foram usados, como valores de σ_1 e σ_2 (sempre utilizando $\sigma_1 = \sigma_2$), 10^{-1}, 10^{-3} e 10^{-5}. E, por fim, para cada um dos problemas obtidos (ou seja, dado uma das seis dimensões citadas e um dos três valores de σ_1 e σ_2), foram geradas 50 instâncias diferentes, com um valor de Δ no intervalo $(0, 100)$ para cada uma.

Em [18] não fica claro quais os valores de σ_1 e σ_2 utilizados para a obtenção dos resultados lá apresentados. Por isso testamos o comportamento de MEQB usando os três valores de σ_1 e σ_2 apresentados acima. Os resultados obtidos pelo MEQB para $\sigma_1 = \sigma_2 = 10^{-1}$ são equivalentes aos apresentados em [18], o que nos faz crer que este seja o valor de σ_1 utilizado em [18] e nos indica que a nossa implementação está correta.

Para a compreensão das tabelas apresentadas, lembre-se de que no Algoritmo 2 existem três critérios de convergência que podem ser satisfeitos, a saber:

- *critério 1:* $\|p\| \leq \Delta$ e $\lambda^k = 0$
- *critério 2:* $\mid \Delta - \|p\| \mid \leq \sigma_1 \Delta$
- *critério 3:* $\|R(\tau \hat{z})\|^2 \leq \sigma_1(2 - \sigma_1) \max\{\sigma_2, \|Rp_k\|^2 + \lambda^k \Delta^2\}$

Tabela 1.1: Comportamento de MEQB para o caso geral

$\sigma_1 = \sigma_2$	dimensão	número de iterações		número de vezes que é satisfeito		
		média	máximo	critério 1	critério 2	critério 3
10^{-1}	10	2.30	4	1	3	46
	20	2.32	5	0	4	46
	40	3.14	5	0	4	46
	60	3.20	5	0	5	45
	80	3.32	5	0	5	45
	100	3.42	5	0	8	42
10^{-3}	10	5.46	11	0	49	1
	20	6.12	11	0	44	6
	40	6.52	11	0	45	5
	60	7.10	11	0	47	3
	80	7.86	14	0	46	4
	100	7.20	12	0	45	5
10^{-5}	10	6.32	14	0	50	0
	20	6.70	13	0	50	0
	40	7.78	12	0	50	0
	60	7.78	14	0	50	0
	80	8.40	15	0	47	3
	100	8.20	12	0	50	0

Tabela 1.2: Comportamento de MEQB para o **caso difícil**

$\sigma_1 = \sigma_2$	dimensão	número de iterações		número de vezes que é satisfeito		
		média	máximo	critério 1	critério 2	critério 3
10^{-1}	10	2.44	5	1	0	49
	20	2.18	4	0	4	46
	40	2.90	4	0	3	47
	60	3.06	4	0	4	46
	80	3.22	5	0	5	45
	100	3.38	5	0	8	42
10^{-3}	10	7.12	10	0	9	41
	20	6.98	10	0	5	45
	40	6.42	10	0	12	38
	60	6.54	10	0	18	32
	80	7.14	10	0	19	31
	100	6.68	10	0	28	22
10^{-5}	10	13.16	18	0	3	47
	20	11.44	15	0	11	39
	40	11.88	15	0	11	39
	60	10.74	16	0	21	29
	80	10.54	16	0	23	27
	100	9.66	14	0	30	20

Tabela 1.3: Comportamento de MEQB para o caso de ponto de sela

$\sigma_1 = \sigma_2$	dimensão	número de iterações		número de vezes que é satisfeito		
		média	máximo	critério 1	critério 2	critério 3
10^{-1}	10	2.24	8	0	0	50
	20	2.08	4	0	0	50
	40	2.90	4	0	0	50
	60	3.10	4	0	0	50
	80	3.34	5	0	0	50
	100	3.50	5	0	0	50
10^{-3}	10	7.26	13	1	0	49
	20	6.84	10	0	0	50
	40	6.92	9	0	0	50
	60	6.70	10	0	0	50
	80	6.66	9	0	0	50
	100	6.66	10	0	0	50
10^{-5}	10	14.04	18	0	0	50
	20	13.28	16	0	0	50
	40	12.84	15	0	0	50
	60	12.74	16	0	0	50
	80	12.62	15	0	0	50
	100	11.94	16	0	0	50

Tabela 1.4: Comportamento de MEQB para o caso definido positivo

$\sigma_1 = \sigma_2$	dimensão	número de iterações		número de vezes que é satisfeito		
		média	máximo	critério 1	critério 2	critério 3
10^{-1}	10	2.08	3	42	7	1
	20	2.26	4	28	22	0
	40	2.30	4	28	20	2
	60	2.52	4	19	28	3
	80	2.58	4	18	29	3
	100	2.66	4	11	36	3
10^{-3}	10	2.38	4	38	12	0
	20	2.58	5	34	16	0
	40	3.08	5	19	31	0
	60	3.10	6	24	26	0
	80	3.66	5	14	36	0
	100	3.52	5	13	37	0
10^{-5}	10	2.40	5	41	9	0
	20	2.82	5	31	19	0
	40	3.02	5	30	20	0
	60	3.48	6	22	28	0
	80	4.02	6	12	38	0
	100	4.54	7	9	41	0

Note que, mesmo no caso geral, o *critério 3* de convergência é satisfeito em um grande número de vezes, principalmente quando os valores de σ_1 são maiores. Veja na tabela 1.1 que, para $\sigma_1 = 10^{-1}$, a maioria das instâncias termina com este critério. A medida que o valor de σ_1 decresce, aumenta o número de instâncias em que o *critério 2* é satisfeito.

No **caso difícil**, como já se esperava, para a maioria das instância, o Algoritmo 2 pára no *critério 3*. Mas, note que, mais uma vez, quanto menor o valor de σ_1, maior o número de instâncias que param no *critério 2*. No entanto, a "migração" de instâncias de um critério para o outro não é tão grande quanto no caso geral. Olhando para a tabela 1.2 pode-se observar que a diferença entre o número de instâncias que satisfazem o *critério 2* e o *critério 3*, quando se muda o valor de σ_1 de 10^{-3} para 10^{-5}, não varia muito.

O caso de ponto de sela é mais radical: apenas em 1 das instâncias o *critério 1* foi satisfeito. Isso porque, apenas neste caso, a matriz B gerada já era definida positiva e, como $g = 0$, o minimizador da quadrática é 0. Nas demais instâncias isso não aconteceu e, portanto, como o passo p obtido sempre é igual a 0, é necessário fazer uso do vetor z. E, neste caso, o critério a ser satisfeito deve ser o *3*.

No caso definido positivo é interessante notar que, ao contrário de todos os demais casos, independentemente do valor de σ_1, o número de iterações gastos até a convergência praticamente não varia. Além disso, este é o único caso em que a quantidade de instâncias que param pelo *critério 1* é significativo. Quando o valor de σ_1 é maior, algumas (poucas) instâncias satisfazem o *critério 3*. Mas, independente do valor de σ_1, o número de instâncias que por fim satisfazem os *critérios 1* e *2* se dividem: praticamente metade delas satisfaz um critério e a outra metade satisfaz o outro.

Em [18] é apresentado o comportamento da implementação do Algoritmo 2 feita pelos autores. O comportamento de MEQB é igual ao apresentado em [18]. Os relatos sobre a variação do número de iterações e do critério de convergência satisfeito para cada instância com relação à variação do valor de σ_1 é o mesmo observado em nossos experimentos.

Capítulo 2

Regiões de confiança

Como já mencionado anteriormente, estamos interessados na construção de um método para minimização de funções irrestritas baseado em regiões de confiança. Para isso, o algoritmo apresentado no capítulo 1 será usado para a resolução dos subproblemas que aparecem em cada iteração do método de regiões de confiança.

A técnica de regiões de confiança para a minimização de uma função irrestrita f é baseada na seguinte idéia: dado um ponto x^k, construímos uma aproximação quadrática de f. Estabelecemos uma região de confiança de raio Δ^k em torno de x^k para a qual acreditamos que a aproximação quadrática está próxima da função f. Então, encontramos o minimizador s^k desta aproximação nesta região. Depois de calculado s^k, verificamos se "andando" na direção s^k (ou seja, em $x^k + s^k$) obtemos decréscimo suficiente na função f. Se isso acontecer, passamos ao ponto $x^k + s^k$, aumentando ou não o raio Δ^k da região de confiança, e inicia-se uma nova iteração. Se o decréscimo obtido em f não for suficiente, diminui-se o raio Δ^k da região e calcula-se um novo s^k.

Neste capítulo apresentaremos detalhadamente a técnica de regiões de confiança. Mostraremos um algoritmo baseado nesta técnica para a minimização de funções irrestritas usando o método apresentado no capítulo 1 para a resolução do subproblema que aparece a cada iteração. Apresentamos os resultados numéricos da aplicação da nossa implementação deste algoritmo nas 18 funções de teste utilizadas em [18] e introduzidas em [17].

2.1 O algoritmo baseado em regiões de confiança

A aproximação quadrática de uma função f em C^2, no ponto x^k, é dada pela série de Taylor truncada de $f(x^k + w)$ de ordem 2. Ou seja,

$$f(x^k + w) \approx f(x^k) + \varphi_k(w),$$

onde

$$\varphi_k(w) = \frac{1}{2} w^T \nabla^2 f(x^k) w + \nabla f(x^k)^T w.$$

A técnica de regiões de confiança para a minimização de uma função irrestrita f é baseada na seguinte idéia: dado um ponto x^k, construímos uma aproximação quadrática $f(x^k) + \varphi_k(w)$ de f. Estabelecemos uma região de confiança de raio Δ^k em torno de x^k para a qual acreditamos que a aproximação quadrática está próxima da função f. Então, encontramos o minimizador s^k desta aproximação nesta região, ou seja, queremos s^k que resolve

$$\begin{array}{ll} \text{minimizar} & f(x^k) + \varphi_k(w) \\ \text{sujeita a} & \|w\| \leq \Delta^k. \end{array} \tag{2.1}$$

Calculado s^k, verificamos se "andando" na direção s^k (ou seja, em $x^k + s^k$) obtemos decréscimo suficiente na função f. Se isso acontecer, passamos ao ponto $x^k + s^k$, aumentando ou não o raio Δ^k da região de confiança, e inicia-se uma nova iteração. Se o decréscimo obtido em f não for suficiente, diminui-se o raio Δ^k da região e calcula-se um novo s^k.

O Algoritmo 3 mostra a forma geral de um método para minimização irrestrita baseado em regiões de confiança.

Algoritmo 3 *Dadas as constantes* $0 < \mu < \eta < 1$ *e* $0 < \gamma_1 < \gamma_2 < 1 < \gamma_3$. *Dados* $x^0 \in I\!R^n$ *e* $\Delta^0 > 0$.

Passo 1. $k = 0$

Passo 2. Calcule $\nabla f(x^k)$ e $\nabla^2 f(x^k)$

Passo 3. Se o "critério de convergência foi satisfeito" então

Pare e devolva x^k como solução

Passo 4. Calcule s^k uma solução aproximada de (2.1)

Passo 5. Calcule $\rho^k = \frac{f(x^k+s^k)-f(x^k)}{\varphi_k(s^k)}$

Passo 6. Se $\rho^k \leq \mu$ então

$\Delta^k = \Delta \in [\gamma_1\Delta^k, \gamma_2\Delta^k]$

Volte para o **Passo 4**

Passo 7. $x^{k+1} = x^k + s^k$

Passo 8. Se $\rho^k \leq \eta$ então

$\Delta^{k+1} \in [\gamma_2\Delta^k, \Delta^k]$

Senão

$\Delta^{k+1} \in [\Delta^k, \gamma_3\Delta^k]$

Passo 9. $k = k + 1$

Passo 10. Volte para o **Passo 2**

É claro que estamos interessados em utilizar o Algoritmo 2 para resolver o **Passo 4**. Vejamos agora que isto é possível.

Primeiro vejamos quais condições a solução aproximada s^k de (2.1) deve satisfazer para garantir que o Algoritmo 3 gere uma seqüência $\{x^k\}$ convergente a um ponto x^* com $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ semidefinida positiva.

Para isso, exige-se que exista $\beta \in (0,1)$ tal que

$$-\varphi(s^k) \geq \beta \max\{-\varphi(w) : w = \alpha \nabla f(x^k),\ \|w\| \leq \Delta^k\}, \quad \|s^k\| \leq \Delta^k.$$

Sob esta condição, provou-se em [23] que, se $\mu > 0$, toda seqüência $\{\nabla f(x^k)\}$ converge a 0. Com isso, podemos esperar que $\{x^k\}$ vá convergir a um ponto x^* com $\nabla f(x^*) = 0$.

Além disso, se temos $\sigma \in (0,1)$ tal que

$$\varphi_k(s^k) = \min\{\varphi_k(w) : \|w\| \leq \delta^k\},$$

com

$$\|s^k\| \leq \delta^k \in [(1-\sigma)\Delta^k, (1+\sigma)\Delta^k],$$

temos que $\nabla^2 f(x^*)$ é semidefinida positiva (veja [22]).

Como o critério de convergência (1.8) pode não ser consistente com essas condições, podemos usar a seguinte generalização

$$-\varphi_k(s^k) \geq \beta_1 \mid \varphi_k^* \mid \ \text{com } \|s^k\| \leq \beta_2 \Delta^k \tag{2.2}$$

com $\beta_1 > 0$ e $\beta_2 > 0$.

Por (1.9) temos que

$$\varphi_k(s^k) - \varphi_k^* \leq \sigma_1(2-\sigma_1) \max\{\mid \varphi_k^* \mid, \sigma_2\}, \quad \|s^k\| \leq (1+\sigma_1)\Delta^k.$$

Se $\varphi_k^* \neq 0$ e $\sigma_2 = 0$, temos que

$$\varphi_k(s^k) - \varphi_k^* \leq \sigma_1(2-\sigma_1) \mid \varphi_k^* \mid, \quad \|s^k\| \leq (1+\sigma_1)\Delta^k \Rightarrow$$

$$-\varphi_k(s^k) \geq -\varphi_k^* - \sigma_1(2-\sigma_1) \mid \varphi_k^* \mid, \quad \|s^k\| \leq (1+\sigma_1)\Delta^k \Rightarrow$$

$$-\varphi_k(s^k) \geq \mid \varphi_k^* \mid -\sigma_1(2-\sigma_1) \mid \varphi_k^* \mid, \quad \|s^k\| \leq (1+\sigma_1)\Delta^k \Rightarrow$$

$$-\varphi_k(s^k) \geq (1-\sigma_1)^2 \mid \varphi_k^* \mid, \quad \|s^k\| \leq (1+\sigma_1)\Delta^k.$$

Tomando-se $\beta_1 = (1-\sigma_1)^2$ e $\beta_2 = (1+\sigma_1)$, temos que o Algoritmo 2 satisfaz as condições requeridas.

Caso $\varphi^* = 0$, temos que $\nabla f(x^k) = 0$ e $\nabla^2 f(x^k)$ é semidefinida positiva, e o Algoritmo 2 pára com x^k como solução.

Portanto, podemos utilizar o Algoritmo 2, com $\sigma_2 = 0$, para resolver o **Passo 4** do Algoritmo 3.

O Algoritmo 4 mostra a versão final do método para minimização irrestrita baseado em regiões de confiança, com todos os detalhes não especificados pelo Algoritmo 3.

Algoritmo 4 *Dados* $\eta, \epsilon, \sigma \in (0,1)$, $x^0 \in I\!R^n$ *e* $\Delta^0 > \Delta_{min} > 0$

Passo 1. $k = 0$

Passo 2. $\lambda^0 = 0$

Passo 3. Calcule $\nabla f(x^k)$ e $\nabla^2 f(x^k)$

Passo 4. Se $\|\nabla f(x^k)\|_\infty \leq \epsilon$ então

Pare e devolva x^k como solução

Passo 5. Calcule s^k usando o Algoritmo 2, com os parâmetros:

$\nabla^2 f(x^k)$ (como a matriz simétrica),

$\nabla f(x^k)$ (como o vetor em $I\!R^n$),

Δ^k, λ^k, $\sigma_1 = \sigma$ e $\sigma_2 = 0$

Passo 6. Calcule $\rho^k = \frac{f(x^k+s^k)-f(x^k)}{\varphi^k(s^k)}$

Passo 7. Se $\rho^k \leq 0.25$ então

$$\Delta^k = \max\left\{\frac{\|s^k\|}{4}, \Delta_{min}\right\}$$

Passo 8. Se $\rho^k \geq 0.5$ e $\|s^k\| \approx \Delta^k$ então

$\Delta^k = 2\Delta^k$

Passo 9. Se $\rho^k < \eta$ então

Volte para o **Passo 5**

Passo 10. $\Delta^{k+1} = \max\{\Delta^k, \Delta_{min}\}$

Passo 11. $x^{k+1} = x^k + s^k$

Passo 12. Atualize λ^{k+1} com o valor de λ obtido como solução no **Passo 5**

Passo 13. $k = k + 1$

Passo 14. Volte para o **Passo 3**

Um detalhe importante é que, quando resolvemos um problema do tipo (2.1), podemos utilizar o valor de λ^k, resultante da aplicação do Algoritmo 2, como valor inicial de λ para a próxima chamada a este algoritmo. Isso porque o valor λ^k é um limitante inferior para a resolução do problema com os mesmos parâmetros, mas com um valor de Δ^k menor.

Em [18] temos a prova de que, usando $\eta \in (0, 0.25)$ no Algoritmo 4, temos $\lim_{k\to\infty} \|\nabla f(x^k)\| = 0$. Mais ainda, se $C = \{x : f(x) \leq f(x^0)\}$ for compacto então há duas alternativas: ou o Algoritmo 4 termina num ponto $x^k \in C$ que satisfaz as condições necessárias de segunda ordem para que x^k seja o minimizador de f (ou seja, $\nabla f(x^k) = 0$ e $\nabla^2 f(x^k)$ é semidefinida positiva); ou a seqüência $\{x^k\}$ gerada pelo Algoritmo 4 tem um ponto limite $x^* \in C$ para o qual valem as condições necessárias de segunda ordem.

Em [18] também se afirma que o grau de convergência do Algoritmo 4 é Q-superlinear.

2.2 Resultados numéricos

Implementamos o Algoritmo 4 numa subrotina em FORTRAN 77 chamada MEQBRC. Para verificar o comportamento desta subrotina, foram utilizadas as 18 funções sugeridas em [17] para teste de métodos de minimização irrestrita. São elas:

1. *Helical valley function*
2. *Biggs EXP6 function*
3. *Gaussian function*
4. *Powell badly scaled function*
5. *Box three-dimensional function*
6. *Variably dimensioned function*
7. *Watson function*
8. *Penalty function I*
9. *Penalty function II*
10. *Brown badly scaled function*
11. *Brown and Dennis function*

12. *Gulf research and development function*
13. *Trigonometric function*
14. *Extended Rosenbrock function*
15. *Extended Powell singular function*
16. *Beale function*
17. *Wood function*
18. *Chebyquad function*

Na tabela 2.1 pode ser verificado o comportamento da subrotina quando aplicada a cada uma das funções acima citadas, com os parâmetros $\eta = 10^{-1}$, $\epsilon = 10^{-4}$, $\sigma = 0.1$ e $\Delta_{min} = 10^{-3}$.

A cada iteração são feitas exatamente uma avaliação de gradiente e uma avaliação de hessiana. Portanto, tanto o número de avaliações de gradiente quanto o número de avaliações de hessiana é igual ao número de iterações.

Tabela 2.1: Comportamento de MEQBRC para funções gerais

função	número de iterações	número de avaliações de função	mínimo	
			obtido	esperado
1	9	20	4.2668906 x 10^{-17}	0
2	4331	9626	1.9781722 x 10^{-04}	0
3	1	2	1.1292712 x 10^{-08}	1.12793...x 10^{-08}
4	414	834	2.8396414 x 10^{-08}	0
5	18	36	4.2190293 x 10^{-09}	0
6	106	364	2.3891072 x 10^{-08}	0
7	1343	3650	7.7074693 x 10^{-03}	2.28767... x 10^{-03}
8	10	20	2.4089543 x 10^{-05}	2.24997... x 10^{-05}
9	9	28	9.5186103 x 10^{-06}	9.37629... x 10^{-06}
10	35	74	1.3509536 x 10^{-27}	0
11	36	80	8.5822202 x 10^{+04}	8.58222... x 10^{+04}
12	34	70	5.5319044 x 10^{-13}	0
13	19	56	2.3674882 x 10^{-09}	0
14	23	50	1.9317734 x 10^{-16}	0
15	1931	10226	5.8417645 x 10^{-06}	0
16	6	14	2.9592252 x 10^{-13}	0
17	44	96	7.4953421 x 10^{-19}	0
18	19	54	5.0992633 x 10^{-10}	0

Note que MEQBRC obteve excelentes resultados. O mínimo encontrado é muitíssimo próximo do esperado. Além de o número de iterações (e, conseqüentemente, de avaliações de gradiente e hessianas) e de avaliações de funções ser, na maioria dos casos, muito pequeno.

Capítulo 3

O método do Gradiente Espectral Projetado

No método de restrições ativas proposto em [2], cada ponto candidato a solução obtido pelo algoritmo pertence a uma face da caixa (restrições do problema) e, a cada iteração, precisamos fazer uma minimização da função objetivo numa dessas faces. Para isso, não utilizaremos o método de regiões de confiança puro, apresentado no capítulo 2. Utilizaremos, sim, um algoritmo que mistura a técnica de regiões de confiança e o método do Gradiente Espectral Projetado (veja [3], [4] e [5]).

Além disso, quando o método de restrições ativas decide mudar de face, ou seja, decide mudar o conjunto de restrições ativas corrente, usamos o método do Gradiente Espectral Projetado (chamado de SPG) para definir qual será a nova face a ser trabalhada. Em outras palavras, usamos o método do Gradiente Espectral Projetado para escolher quais restrições devem entrar no conjunto das restrições ativas e quais devem sair deste conjunto. O funcionamento do método de restrições ativas e o papel do SPG neste método ficarão mais claros nos capítulos 4 e 5.

O SPG foi escolhido por ter fácil implementação e ser eficiente. Isso porque apenas informação de primeira ordem é necessária para a execução deste método. Outro fator importante é que, pelo fato de se basear em projeções, o SPG tende a acrescentar um grande número de restrições ao conjunto de restrições ativas. Isso é muito interessante, pois, quanto maior o número de restrições ativas, menor a dimensão do subproblema a ser resolvido na face.

Na seção 3.1 apresentamos o método do Gradiente Espectral Projetado, utilizado para minimizar funções gerais restritas a uma caixa, proposto em [4]. As provas de convergência do método podem ser encontradas em [4]. Na seção 3.2 apresentamos os detalhes de nossa implementação deste método.

3.1 O algoritmo

O método do Gradiente Espectral Projetado é usado para resolver o seguinte problema:

$$\begin{array}{ll} \text{minimizar} & f(x) \\ \text{sujeita a} & x \in A, \end{array}$$

onde f é uma função em C^1, $x \in \mathbb{R}^n$ e A é um conjunto convexo fechado.

Estamos interessados no caso em que A é uma caixa, ou seja, $l \leq x \leq u$, com $l, u, x \in \mathbb{R}^n$.

A idéia do algoritmo é a seguinte: na iteração k, calculamos uma direção de descida d^k e fazemos uma busca linear para que $x^k + t_k d^k$ satisfaça a condição de Armijo. Quando isso acontece, temos o novo ponto $x^{k+1} = x^k + t_k d^k \in A$. A direção d^k é calculada com base na projeção ortogonal de $-\lambda_k^{\text{SPG}} \nabla f(x^k)$ no conjunto A, onde λ_k^{SPG} (chamado de *passo espectral*) é um escalar calculado a partir de informações fornecidas pelo ponto atual x^k e pelo ponto x^{k-1} da iteração anterior.

Antes de apresentar uma típica iteração do algoritmo, vejamos algumas notações: dado um ponto $x \in \mathbb{R}^n$, denotamos por $P_A(x)$ a projeção ortogonal do ponto x no conjunto A; dado um conjunto A, denotamos por A^0 o conjunto de pontos interiores de A.

Iteração do SPG *Dados* $x^k \in A^0$, $0 < \lambda_{min}^{SPG} < \lambda_{max}^{SPG} < \infty$, $\alpha \in \left(0, \frac{1}{2}\right)$. *Supondo que* $\nabla f(x^k) \neq 0$.

Passo 1. Calcule $s_k = x^k - x^{k-1}$

Passo 2. Calcule $y_k = \nabla f(x^k) - \nabla f(x^{k-1})$

Passo 3. *Cálculo do* λ_k^{SPG}

Se ${s_k}^T y_k \leq 0$ então

$\lambda_k^{\text{SPG}} = \lambda_{max}^{\text{SPG}}$

Senão

$\lambda_k^{\text{SPG}} = \min\left\{\lambda_{max}^{\text{SPG}}, \max\left\{\lambda_{min}^{\text{SPG}}, \frac{{s_k}^T s_k}{{s_k}^T y_k}\right\}\right\}$

Passo 4. Calcule $d^k = P_A(x^k - \lambda_k^{\text{SPG}} \nabla f(x^k)) - x^k$

Passo 5. $t_k = 1$

Passo 6. Se $f(x^k + t_k d^k) \leq f(x^k) + \alpha t_k \nabla f(x^k)^T d^k$ então

Termine a iteração com $y^k = x^k + t_k d^k$

Senão

Escolha $t_{novo} \in [0.1 t_k, 0.5 t_k]$

Faça $t_k = t_{novo}$

Repita o **Passo 6**

O valor obtido para λ_k^{SPG} possui alguma informação de segunda ordem da função objetivo (veja [4]). Na verdade, temos que

$$\left(\frac{1}{\lambda_k^{\mathrm{SPG}}}\right) I = \begin{array}{ll} \mathrm{argmin}_D & \|Ds_k - y_k\| \\ \text{sujeita a} & D = \gamma I, \end{array}$$

onde I é a matriz identidade. Ou seja, $\left(\frac{1}{\lambda_k^{\mathrm{SPG}}}\right) I$ é a matriz diagonal que melhor aproxima a equação secante (equação esta que é uma aproximação da hessiana da função objetivo).

Quanto à convergência do algoritmo do Gradiente Espectral Projetado, temos que todo ponto de acumulação da seqüência $\{x^k\}$ gerada por ele é um ponto estacionário restrito. Para a prova, veja [4].

3.2 Implementação

Implementamos o algoritmo SPG em FORTRAN 77. Para implementar este método, foi necessário escolher uma maneira para se calcular o passo espectral inicial λ_0^{SPG}, já que este não pode ser calculado da mesma maneira que os demais passos espectrais. Além disso, para que todos os passos estejam perfeitamente definidos, precisamos definir uma maneira para escolher t_{novo} no **Passo 3** da Iteração do SPG. Nas seções abaixo, apresentamos a maneira utilizada em nossa implementação para se calcular λ_0^{SPG} e t_{novo}.

3.2.1 Cálculo de $\lambda_0^{\mathbf{SPG}}$

Uma maneira de calcular λ_0^{SPG} foi proposta em [3]. A idéia é calcular um $\bar{x}$ dando um passo muito pequeno a partir de x^0 na direção de $-\nabla f(x^0)$. Neste caso, esperamos que a função objetivo decresça e satisfaça a condição de Armijo. Por isso utilizamos, para calcular λ_0^{SPG}, os mesmos passos usados para calcular λ_k^{SPG}, considerando $\bar{x}$ como x^k e x^0 como x^{k-1}. Abaixo estão descritos todos os passos necessários para o cálculo de λ_0^{SPG} (dados $\epsilon_{rel}, \epsilon_{abs} > 0$):

Passo 1. Calcule $t_{peq} = \max\{\epsilon_{rel}\|x^0\|_\infty, \epsilon_{abs}\}$

Passo 2. Calcule $\bar{x} = x^0 - t_{peq}\nabla f(x^0)$

Passo 3. Calcule $\bar{s} = \bar{x} - x^0$

Passo 4. Calcule $\bar{y} = \nabla f(\bar{x}) - \nabla f(x^0)$

Passo 5. *Cálculo do* λ_0^{SPG}

Se $\bar{s}^T\bar{y} \leq 0$ então

$\lambda_0^{\mathrm{SPG}} = \lambda_{max}^{\mathrm{SPG}}$

Senão

$$\lambda_0^{\text{SPG}} = \min\left\{\lambda_{max}^{\text{SPG}}, \max\left\{\lambda_{min}^{\text{SPG}}, \frac{\bar{s}^T\bar{s}}{\bar{s}^T\bar{y}}\right\}\right\}$$

3.2.2 Escolha de t_{novo}

Uma forma simples é calcular $t_{novo} = \frac{t_k}{2}$. Uma forma mais interessante é utilizar a interpolação quadrática uni-dimensional $q(w)$ tal que $q(0) = f(x^k)$, $q(t_k) = f(x^k + t_k d^k)$ e $\nabla q(0) = \nabla f(x^k)^T d^k$. Para obter t_{novo}, calculamos o mínimo de $q(w)$ e fazemos a salvaguarda (lembre-se que estamos interressados em $t_{novo} \in [0.1t_k, 0.5t_k]$).

Primeiro, vejamos como calcular o mínimo de $q(w)$. Observe que

$$q(w) = \frac{1}{2}aw^2 + bw + c \tag{3.1}$$

$$q^{'}(w) = aw + b. \tag{3.2}$$

Por (3.1), temos que $q(0) = c$. Como $q(0) = f(x^k)$, temos

$$c = f(x^k).$$

Por (3.2), temos $q^{'}(0) = b$. Como $q^{'}(0) = \nabla f(x^k)^T d^k$, temos

$$b = \nabla f(x^k)^T d^k.$$

Ainda por (3.1), temos $q(t_k) = \frac{1}{2}at_k^2 + bt_k + c$. Substituindo b e c pelos valores obtidos, temos

$$q(t_k) = \frac{1}{2}at_k^2 + t\nabla f(x^k)^T d^k + f(x^k) = f(x^k + t_k d^k) \Rightarrow$$

$$a = 2\frac{f(x^k + t_k d^k) - f(x^k) - t_k \nabla f(x^k)^T d^k}{t_k^2}.$$

Como estamos interessados no mínimo de $q(w)$, queremos $q^{'}(t_{novo}) = 0$. Ou seja,

$$q^{'}(t_{novo}) = at_{novo} + b = 0 \Rightarrow t_{novo} = -\frac{b}{a} \Rightarrow$$

$$t_{novo} = -\frac{\nabla f(x^k)^T d^k}{2\frac{f(x^k + t_k d^k) - f(x^k) - t_k \nabla f(x^k)^T d^k}{t_k^2}} \Rightarrow$$

$$t_{novo} = -\frac{t_k^2 \nabla f(x^k)^T d^k}{2(f(x^k + t_k d^k) - f(x^k) - t_k \nabla f(x^k)^T d^k)}.$$

A seguir estão os passos utilizados para a escolha de t_{novo} utilizando a interpolação quadrática e salvaguarda:

Passo 1. *Para t_k pequeno não vale a pena calcular a interpolação quadrática*

Se $t_k \leq 0.1$ então

$t_{novo} = \frac{t_k}{2}$

Passo 2. Calcule $t_{tent} = -\frac{t_k^2 \nabla f(x^k)^T d^k}{2(f(x^k + t_k d^k) - f(x^k) - t_k \nabla f(x^k)^T d^k)}$

Passo 3. *Salvaguarda de t_{novo}*

Se $(t_{tent} \geq 0.1)$ e $(t_{tent} \leq 0.5 t_k)$ então

$t_{novo} = t_{tent}$

Senão

$t_{novo} = \frac{t_k}{2}$

Um ponto interessante a se observar na salvaguarda do t_{novo} é que não verificamos, como classicamente se faz, se $t_{novo} \in [0.1t_k, 0.5t_k]$, mas sim que $t_{novo} \in [0.1, 0.5t_k]$ (veja [5]). Isso pode ser interpretado da seguinte maneira: quando a interpolação rejeita 90% do intervalo original de busca pela solução ($[0, 1]$), julgamos que esta interpolação não é confiável e preferimos a opção $t_{novo} = \frac{t_k}{2}$. Este tipo de salvaguarda se mostrou mais eficiente do que a maneira clássica.

Capítulo 4

Um método híbrido para minimização nas faces

O método de restrições ativas usa o esquema descrito em [2]. A caixa será dividida em "faces abertas". Em cada cada face aberta, um algoritmo interno é usado para modificar somente as variáveis livres. O algoritmo interno pode ser descrito como um método de ponto interior (irrestrito) que pode atingir a borda do conjunto convexo no qual a função objetivo está definida.

O algoritmo interno utilizado neste trabalho é baseado no método de regiões de confiança apresentado no capítulo 2. A dificuldade de implementar a estratégia de regiões de confiança usando norma euclidiana na presença de restrições de caixa está no fato de a intersecção de caixas e bolas não ser um conjunto simples. Por isso, encontrar o minimizador da aproximação quadrática em tal conjunto não é direta e a técnica de intersecção de algoritmos de regiões de confiança com restrições (veja [13] e [14]) não é fácil de aplicar.

Geralmente se diz que, quando temos problemas com restrições de caixa, regiões de confiança baseadas em norma infinito são melhores, porque elas apresentam a mesma geometria do domínio do problema (intersecção de caixas e caixas também são caixas). Entretanto, encontrar minimizadores globais de uma quadrática (não necessariamente convexa) não é tão simples quanto encontrar minimizadores de quadráticas arbitrárias em bolas (veja [11], [16] e [18]).

Portanto, regiões de confiança euclidianas comuns, cujos raios, em princípio, independem das restrições de caixa, merecem consideração. No entanto, as dificuldades técnicas devem ser eliminadas.

Neste capítulo propomos um algoritmo híbrido para minimização nas faces. Neste algoritmo usamos, basicamente, a técnica de regiões de confiança apresentada no capítulo 2, modificada para lidar com o caso em que o minimizador da quadrática está fora da face. Quando estamos num ponto muito próximo da borda da face, usamos o método SPG, pois, neste caso, pode não valer a pena resolver o subproblema de regiões de confiança.

Apresentamos ainda o teorema (e prova) de convergência do algoritmo descrito na seção

4.1 e mostramos os detalhes de nossa implementação.

4.1 O algoritmo

Seja $A \subset I\!R^m$ um conjunto fechado, limitado e convexo com interior não-vazio. Suponha que

$$\bar{f} : I\!R^m \to I\!R$$

tenha derivadas terceiras contínuas num conjunto aberto que contém A. Usaremos a seguinte notação:

$$\bar{B}(x, \delta) = \{z \in I\!R^m | \|z - x\| \leq \delta\}.$$

Lembrando que o subproblema de regiões de confiança, neste caso, pode ser escrito como

$$\begin{array}{ll} \text{minimizar} & \bar{f}(x^k) + \varphi_k(x) \\ \text{sujeita a} & \|x\| \leq \Delta, \end{array}$$

com

$$\varphi_k(x) = \frac{1}{2} x^T \nabla^2 \bar{f}(x^k) x + \nabla \bar{f}(x^k)^T x.$$

Uma típica iteração do Algoritmo Interno é apresentada a seguir:

Iteração do Algoritmo Interno *Dados* $x^k \in A^0$*;* $\Delta_{min} > 0$*;* $\Delta \geq \Delta_{min}$

Passo 1. *Cálculo da distância até a borda*

Calcule $\Delta_{borda} = \max\{\bar{\Delta} \geq 0 \mid \bar{B}(x^k, \bar{\Delta}) \subset A\}$

Passo 2. *Se está perto da borda, usa SPG*

Se $\Delta_{borda} < 2\Delta_{min}$ então

Se $\nabla \bar{f}(x^k) = 0$ então

Termine a execução do Algoritmo Interno com x^k declarando:
"Ponto estacionário de primeira ordem perto da borda"

Senão

Faça uma iteração do SPG para calcular d^k e y^k de forma que $y^k = x^k + t_k d^k$ (usando $\bar{f}$ e $\nabla \bar{f}$ no lugar de f e ∇f)

Vá para o **Passo 6**

Passo 3. *Se está longe da borda, resolve o subproblema de regiões de confiança*

Passo 3.1.

Calcule $\nabla^2 \bar{f}(x^k)$

Passo 3.2.

Calcule $x^{tent} = x^k + d^k$ com d^k solução do subproblema de regiões de confiança

Se $\varphi_k(d^k) = 0$ então

Termine a execução do Algoritmo Interno com x^k declarando:
"Ponto estacionário de segunda ordem"

Se $x^{tent} \notin A^0$ então

Calcule $t_{max} = \max\{t \in [0,1] \mid [x^k, x^k + td^k] \subset A\}$

Se $\bar{f}(x^k + t_{max}d^k) < \bar{f}(x^k)$ então

Defina $y^k = x^k + t_{max}d^k$

Vá para o **Passo 6**

Senão

Escolha $\Delta \in [\Delta_{min}, \Delta_{borda})$

Volte para o **Passo 3.2**

Passo 4. *Aceitação ou rejeição da solução do subproblema de regiões de confiança*

Calcule $\rho^k = \frac{\bar{f}(x^{tent}) - \bar{f}(x^k)}{\varphi_k(x^{tent})}$

Se $\rho^k \geq 0.1$ então

Defina $y^k = x^{tent}$

Senão

Escolha $\Delta \in [0.1\|d^k\|, 0.9\|d^k\|]$

Volte para o **Passo 3.2**

Passo 5. *Atualização do raio da região de confiança*

Escolha $\Delta \geq \Delta_{min}$

Passo 6. *Possível melhora adicional*

Escolha $x^{k+1} \in A$ tal que $\bar{f}(x^{k+1}) \leq \bar{f}(y^k)$
(Observe que $x^{k+1} = y^k$ é uma possível escolha)

Se x^{k+1} pertence à borda de A então

Termine a execução do Algoritmo Interno com x^{k+1} declarando:

"Iterando na borda"

No **Passo 6**, a possível escolha de $x^{k+1} \neq y^k$ tem, essencialmente, propósitos teóricos. Veremos a importância desta hipótese na prova do teorema de convergência na próxima seção. Entretanto, a liberdade permitida para a escolha de x^{k+1} também pode ser usada para melhorar a iteração utilizando técnicas de extrapolação. Na seção 4.3 comentaremos o uso desta técnica.

Note que não é possível utilizar o método puro de regiões de confiança tal como apresentado do capítulo 2 para fazer a minimização nas faces. Perto da borda da face, algo diferente deve ser feito para se obter convergência. A estratégia utilizada pelo Algoritmo Interno é uma das possíveis estratégias que leva a propriedades de convergência adequadas. Para entender porque a estratégia de regiões de confiança sozinha não funciona, considere o problema

$$\begin{array}{ll} \text{minimizar} & x_1 + x_2 \\ \text{sujeita a} & -1 \leq x_1 \leq 100,\ 0 \leq x_2 \leq 100. \end{array}$$

A solução deste problema é $(-1, 0)$. Suponha que $x^0 = (1, 1)$ e que $x^{tent} = x^k + d^k$ com d^k solução do subproblema de regiões de confiança definida pela distância de x^k até a borda da face. O teste $\rho^k \geq 0.1$ é satisfeito por x^{tent} para todo k, então podemos definir $x^{k+1} = x^{tent}$. No entanto, todos os iterandos estão na reta $x_1 = x_2$, a seqüência é infinita e converge para $(0, 0)$, que não é um ponto interior e $\nabla \bar{f}(0, 0) = (1, 1) \neq (0, 0)$.

4.2 Teorema de convergência

Teorema 4.2.1 *Na execução do Algoritmo Interno, ocorre uma e somente uma das seguintes possibilidades:*

1. *o algoritmo termina em um número finito de iterações num ponto x^k tal que a distância de x^k à borda é menor que $2\Delta_{min}$ e $\nabla \bar{f}(x^k) = 0$;*
2. *o Algoritmo Interno termina em um número finito de iterações num ponto estacionário de segunda ordem x^k tal que sua distância $\Delta_{borda} \geq 2\Delta_{min}$;*
3. *o Algoritmo Interno termina depois de um número finito de iterações num ponto x^{k+1} na borda de A e*
$$\bar{f}(x^{k+1}) < \bar{f}(x^k) < ... < \bar{f}(x^0);$$
4. *o Algoritmo Interno gera um seqüência infinita de iterandos. Neste caso: (a) todo ponto limite x^* da seqüência gerada satisfaz $g_I(x^*) = 0$ (veja a seção 5.1 para a definição de $g_I(x^k)$); (b) se um ponto limite x^* é tal que sua distância até a borda é maior que $2\Delta_{min}$ então $\nabla^2 \bar{f}(x^*)$ é semidefinida positiva.*

Prova As três primeiras possibilidades são consideradas na definição do algoritmo, já que elas correspondem aos três critérios de parada. Suponha, então, que uma seqüência infinita

seja gerada e que x^* é um ponto limite desta seqüência. Então, existe um conjunto infinito $K_1 \subset I\!N$ tal que

$$\lim_{k \in K_1} x^k = x^*.$$

Considere as duas possibilidades:

(a) Para infinitas iterações $k \in K_2 \subset K_1$, x^{k+1} é computado pelo **Passo 2**.

(b) Para infinitas iterações $k \in K_3 \subset K_1$, x^{k+1} é computado pelo **Passo 3**.

No caso (a), os argumentos usados na prova de convergência do método do Gradiente Espectral Projetado (veja [4]) mostram que $g_I(x^*) = 0$. Na verdade, se supomos que $\nabla \bar{f}(x^k) \neq 0$ e que t_k fica longe de 0, chegamos à contradição $\bar{f}(x^k) \to -\infty$. Por outro lado, se o ínfimo de $\{t_k\}$ é 0, temos uma seqüência auxiliar $t_k^{'}$ que tende a 0 e tal que o teste feito no **Passo 3** do SPG é rejeitado e então $t_k = t_k^{'}$. Isso implica, usando argumentos clássicos, que $g_I(x^*) = 0$.

No caso (b), depois de renomeação, a seqüência pode ser interpretada como uma seqüência de iterandos de um método de regiões de confiança (com o raio de confiança inicial Δ maior ou igual a $\Delta_{min} > 0$), onde cada iterando é modificado com uma maior redução da função objetivo. Então, argumentos clássicos da literatura sobre regiões de confiança (veja, por exemplo, [13] e [14]) mostram que $\nabla \bar{f}(x^*) = 0$ (o que implica que $g_I(x^*) = 0$) e $\nabla^2 \bar{f}(x^*)$ é semidefinida positiva.

Os argumentos acima mostram que qualquer ponto limite x^* satisfaz $g_I(x^*) = 0$. Mais ainda, se a distância de x^* até a borda é maior que $2\Delta_{min}$ então, para $k \in K_1$ suficientemente grande, x^{k+1} é computado pelo **Passo 3**. Portanto, o caso (b) se aplica e, então, a hessiana é semidefinida positiva. ■

4.3 Implementação

Implementamos o Algoritmo Interno em FORTRAN 77. Para que a implementação fosse possível, foi necessário definir como calcular alguns passos do Algoritmo Interno que foram deixados em aberto, a saber, como resolver o subproblema de regiões de confiança e como fazer as atualizações do raio da região de confiança. Além disso, nossa implementação utiliza a técnica de extrapolação apresentada em [2].

Para resolver subproblema de regiões de confiança do **Passo 3.2**, utilizamos nossa implementação do algoritmo apresentado em [18], o MEQB (veja capítulo 1), com os seguintes parâmetros: $\nabla^2 \bar{f}(x^k)$ como a matriz simétrica B, $\nabla \bar{f}(x^k)$ como o vetor g, $\sigma_1 = \sigma$, $\sigma_2 = 0$ e $\lambda^0 = 0$.

Nas seções a seguir apresentamos como foram feitas todas as atualizações do raio Δ da região de confiança e mostramos a técnica de extrapolação usada em nossa implementação.

4.3.1 Atualizações de Δ

No **Passo 3.2**, quando temos de escolher $\Delta \in [\Delta_{min}, \Delta_{bound})$, definimos

$$\Delta = \max\left\{\Delta_{min}, \Delta_{min} + 0.9\left(\left(\frac{\Delta_{bound}}{1+\sigma}\right) - \Delta_{min}\right)\right\}. \tag{4.1}$$

Isso porque, utilizando MEQB para encontrar o minimizador do modelo quadrático com raio Δ, podemos ter o ponto x^{tent} a uma distância σ para fora da região de confiança. Utilizando a atualização (4.1) garantimos que, ao repetir o **Passo 3.2**, teremos o ponto x^{tent} no interior da face.

No **Passo 4**, quando não obtemos decréscimo suficiente (ou seja, $\rho^k < 0.1$), precisamos escolher $\Delta \in [0.1\|d^k\|, 0.9\|d^k\|$. Neste caso, definimos

$$\Delta = 0.25\|d^k\|.$$

No **Passo 5**, atualizamos o raio da região de confiança baseado na relação entre a redução obtida na função objetivo e a redução predita por seu modelo quadrático (ρ^k) da seguinte maneira (veja [10]):

$$\bar{\Delta} = \begin{cases} 0.25\|d^k\|, & \text{if } \rho^k \leq 0.25, \\ 2\Delta, & \text{if } \rho^k \geq 0.5 \text{ e } \mid \|d^k\| - \Delta \mid \leq 10^{-5}, \\ \Delta, & \text{caso contrário,} \end{cases}$$

e

$$\Delta = \max\{\Delta_{min}, \bar{\Delta}\}.$$

4.3.2 Extrapolação

Como dito anteriormente, utilizamos em nossa implementação do Algoritmo Interno a técnica de extrapolação apresentada em [2]. Veja como funciona esta técnica:

Extrapolação *Dados $x^k \in A$, d^k direção de descida, $\mu \in (0, 1]$ e $N > 1$*

Passo 1. *Decisão de extrapolar ou não*

Se $(d^k)^T \nabla \bar{f}(x^k + d^k) \geq 0.5(d^k)^T \nabla \bar{f}(x^k)$ então

Termine a Extrapolação e devolva x^k

Passo 2. Calcule $\mu_{max} = \max\{t \geq 0 \mid [x^k, x^k + td^k] \subset A\}$

Passo 3. *Cálculo do novo escalar que multiplica d^k*

Se $(\mu < \mu_{max})$ e $(N\mu > \mu_{max})$ então

$\mu_{tent} = \mu_{max}$

Senão

$\mu_{tent} = N\mu$

Passo 4. *Pára extrapolação quando ponto novo está muito perto do anterior*

Se $(\mu \geq \mu_{max})$ e $(\|P_A(x^k + \mu_{tent}d^k) - P_A(x^k + \mu d^k)\|_\infty <$
$\max\{\epsilon_{abs}, \epsilon_{rel}\|P_A(x^k + \mu d^k)\|_\infty\})$ então

$x^{k+1} = P_A(x^k + \mu d^k)$

Termine a Extrapolação e devolva x^{k+1}

Passo 5. *Pára extrapolação quando consegue decréscimo simples da função*

Se $\bar{f}(P(x^k + \mu_{tent}d^k)) \geq \bar{f}(P(x^k + \mu d^k))$ então

$x^{k+1} = P_A(x^k + \mu d^k)$

Termine a Extrapolação e devolva x^{k+1}

Senão

$\mu = \mu_{tent}$

Volte para o **Passo 3**

O parâmetro μ se refere ao escalar usado para calcular $y^k = x^k + \mu d^k$ no Algoritmo Interno. Note que podemos chegar ao **Passo 6** do Algoritmo Interno, onde é feita a Extrapolação, saindo dos **Passos 2**, **3** ou **5**. Assim, quando chegamos à Extrapolação vindos do **Passo 2**, temos que $\mu = t_k$; quando chegamos pelo **Passo 3**, temos que $\mu = t_{max}$; e, quando chegamos pelo **Passo 5**, temos $\mu = 1$.

Em nossa implementação acrescentamos 3 parâmetros que permitem ao usuário escolher se a Extrapolação será ou não tentada a partir de cada um dos passos do Algoritmo Interno que levam a ela. Quando não se deseja tentar a Extrapolação, toma-se $x^{k+1} = y^k$ no **Passo 6** do Algoritmo Interno.

Capítulo 5

Um método de restrições ativas para minimização em caixas

Como já foi dito anteriormente, os métodos de restrições ativas para a minimização de funções com restrições de caixa seguem o seguinte princípio: se conhecemos quais restrições são satisfeitas por igualdade na solução (ativas) e quais não (inativas) podemos fixar algumas variáveis nos seus limitantes e resolver um problema irrestrito nas outras variáveis (variáveis livres). Por isso, os métodos de restrições ativas tentam, a cada iteração, inferir quais restrições serão ativas na solução e resolver um problema localmente irrestrito nas variáveis livres.

Recentemente, foi introduzido um novo método de restrições ativas para a minimização de problemas com restrições de caixa (veja [2]). Neste método, divide-se o conjunto factível Ω em faces abertas disjuntas. De acordo com um teste, decide-se se o algoritmo deve continuar a trabalhar na face corrente ou deve mudar de face. Utiliza-se um algoritmo interno para minimizar a função nas faces e o método do Gradiente Espectral Projetado para sair das faces. Neste capítulo apresentamos um método baseado naquele apresentado em [2], que utiliza o algoritmo proposto no capítulo 4 para trabalhar nas faces.

Na seção 5.1 apresentamos o método de restrições ativas para minimização em caixas. Na seção 5.2 apresentamos o teorema de convergência deste método. Na seção 5.3 apresentamos os detalhes de nossa implementação do método apresentado na seção 5.1 e, na seção 5.4 apresentamos os resultados numéricos obtidos por nossa implementação, bem como a comparação do desempenho de nosso método e de LANCELOT.

5.1 O algoritmo

Nesta seção,

$$\Omega = \{x \in I\!R^n \mid \ell \leq x \leq u\}$$

e $f : I\!R^n \to I\!R$ tem derivadas contínuas de terceira ordem num conjunto aberto que contém Ω. Denotamos $g = \nabla f$. Como em [2], dividimos o conjunto factível Ω em faces abertas disjuntas.

Para todo $I \subset \{1, 2, \ldots, n, n+1, n+2, \ldots 2n\}$ definimos

$$F_I = \{x \in \Omega \mid x_i = \ell_i \text{ se } i \in I,\ x_i = u_i \text{ se } n+i \in I, \ell_i < x_i < u_i \text{ caso contrário}\}.$$

A caixa Ω é a união das faces abertas. As variáveis x_i tais que nem i nem $n+i$ pertencem a I são chamadas *livres* em F_I. O Algoritmo Interno definido no capítulo 4 modifica somente as variáveis livres. Portanto, o conjunto convexo A será identificado com os diferentes fechos das faces abertas. Definimos V_I o menor espaço afim que contém F_I e S_I o subespaço linear paralelo a V_I. Definimos o gradiente projetado da seguinte maneira:

$$g_P(x) = P_\Omega(x - g(x)) - x$$

e, para todo $x \in F_I$, definimos o gradiente interno como

$$g_I(x) = P_{S_I}[g_P(x)].$$

Observe que $g_I(x) = 0$ quando a face F_I é um vértice, pois, neste caso, $S_I = \{0\}$.

Como em [2], o algoritmo para minimização com restrições de caixa irá combinar iterações do Algoritmo Interno com iterações de SPG. Chamaremos de iterações do "SPG interno" às que iterações do SPG que forem feitas a partir do **Passo 2** do Algoritmo Interno, e de iterações do "SPG global" àquelas que forem feitas a partir do **Passo 4** do BETRA.

A idéia do método de restrições ativas é: dividimos o conjunto factível em faces abertas e, a cada iteração k, estamos num ponto x^k pertencente a uma dessas faces. No início de cada iteração, verificamos qual a relação entre a norma do gradiente interno e a norma do gradiente projetado. Quando a norma do gradiente projetado é "muito maior" que a norma do gradiente interno, achamos que é interessante mudar de face e fazemos uma iteração do Gradiente Espectral Projetado (como mostrado no capítulo 3) para definir o novo ponto x^{k+1}, pertencente a uma nova face aberta. Quando a norma do gradiente projetado não é "muito maior", decidimos continuar na mesma face e utilizamos um iteração do Algoritmo Interno apresentado no capítulo 4 para obter um ponto x^{k+1} onde a função tem um valor menor do que em x^k.

A seguir definimos o algoritmo de restrições ativas para minimização em caixas, chamado de BETRA (box-Euclidian-trust-region algorithm):

BETRA *Dados* $x^0 \in \Omega$ *e* $\eta \in (0, 1)$

Passo 1. $k = 0$

Passo 2. Seja F_I a face aberta à qual x^k pertence.

Calcule $g_P(x^k)$ e $g_I(x^k)$

Passo 3. *Critério de convergência*

Se F_I é um vértice e $g_P(x^k) = 0$ então

Pare e devolva x^k como solução

Passo 4. *Decisão de permanecer ou não na mesma face*

Se

$$\|g_I(x^k)\| \geq \eta \|g_P(x^k)\| \tag{5.1}$$

Suponha, sem perda de generalidade, que as variáveis livres em F_I são $x_1, ..., x_m$ e que as variáveis restantes estão fixas em $\bar{x}_{m+1}, ..., \bar{x}_n$ ($\bar{x}_i \in \{l_i, u_i\}$, para $m+1 \leq i \leq n$). Defina $\bar{f}(x_1, ..., x_m) = f(x_1, ..., x_m, \bar{x}_{m+1}, ..., \bar{x}_n)$, para todo $x_1, ..., x_m \in I\!R$ e $A = \{x \in I\!R^m \mid l_i \leq x_i \leq u_i, \forall i = 1, ..., m\}$.

Use uma iteração do Algoritmo Interno para calcular x^{k+1}

Se o Algoritmo Interno terminar sua execução com x^k declarando *"Ponto estacionário de primeira ordem perto da borda"* ou *"Ponto estacionário de segunda ordem"* então
Pare e devolva x^k como solução,
declarando o mesmo que o Algoritmo Interno

Senão

Faça uma iteração do SPG com $A = \Omega$ e tome $x^{k+1} = y^k$

Passo 5. $k = k + 1$

Passo 6. Volte para o **Passo 2**

Note que, se a face corrente não é um vértice, o algoritmo BETRA não termina, necessariamente, quando, no início da iteração, $g_P(x^k) = 0$. Neste caso, vale (5.1) e uma iteração do Algoritmo Interno é feita. Se x^k não está perto da borda, um iteração de regiões de confiança é feita com o objetivo de encontrar um ponto estacionário de segunda ordem.

5.2 Teorema de convergência

Teorema 5.2.1 *Vale uma das seguintes afirmações:*

1. *A execução do BETRA termina numa iteração onde o Algoritmo Interno termina com $g_P(x^k) = 0$. Neste caso, se $x^k \in F_I$ e a distância entre x^k e a borda de F_I é maior que $2\Delta_{min}$, a matriz $\nabla^2 f(x^k)$ é semidefinida positiva quando restrita a F_I;*

2. *BETRA gera uma seqüência infinita de iterandos e pelo menos um ponto limite dessa seqüência é um ponto estacionário de primeira ordem.*

No segundo caso, se todos os pontos limite são não-degenerados, todos os iterandos pertencerão, em algum momento, à mesma face aberta F_I. Neste caso, se a distância de um ponto limite x^* à borda de F_I for maior que $2\Delta_{min}$, então $\nabla^2 f(x^*)$ é semidefinida positiva quando restrita a F_I. Mais ainda, se esta matriz reduzida for definida positiva, a seqüência converge quadraticamente a x^*.

Prova Note que, pela definição de BETRA, a terminação finita ocorre somente quando x^k é um vértice no qual $g_P(x^k) = 0$ ou quando o Algoritmo Interno termina com as declarações *"Ponto estacionário de primeira ordem perto da borda"* ou *"Ponto estacionário de segunda ordem"*. Em ambos os casos, temos que as derivadas correspondentes às variáveis livres são nulas, então $g_I(x^k) = 0$. Então, por (5.1), temos que $g_P(x^k) = 0$. Pelo Teorema 4.2.1, a hessiana reduzida é semidefinida positiva se a distância de x^k até a borda da face à qual este ponto pertence é maior que $2\Delta_{min}$.

Falta agora considerar o caso em que BETRA gera infinitos iterandos. Consideremos duas possibilidades:

(a) Existe um conjunto infinito de índices $K_1 \subset I\!N$ tal que x^{k+1} é computado por uma iteração do SPG (global) para todo $k \in K_1$.

(b) Para todo $k \in I\!N$ suficientemente grande, x^{k+1} é computado pelo Algoritmo Interno.

Se vale (a), provamos, como em [2], que todo ponto limite de $\{x^k\}_{k\in K_1}$ é estacionário de primeira ordem restrito.

Se vale (b), existe $k_0 \in I\!N$ tal que x^{k+1} é computado pelo Algoritmo Interno para todo $k \geq k_0$. Como o número de faces é finito e restrições ativas são abandonadas apenas por iterações do SPG (global), existe uma face F_I tal que $x^k \in F_I$ para todo k suficientemente grande. Portanto, a tese do Teorema 4.2.1 se aplica e, conseqüentemente, $\lim_{k\to\infty} g_I(x^k) = 0$. Então, por (5.1), $\lim_{k\to\infty} g_P(x^k) = 0$. Isso implica que todos os pontos limites satisfazem $g_P(x^*) = 0$. Ainda pelo Teorema 4.2.1, se a distância de um ponto limite até a borda for maior que $2\Delta_{min}$, a hessiana reduzida é semidefinida positiva. Neste caso, se a hessiana for definida positiva, a convergência quadrática segue como na teoria clássica do método newtoniano irrestrito de regiões de confiança (veja [22]). O fato de que no caso de não-degenerescência todos os iterandos pertencem, em algum momento, à mesma face segue como no Teorema 3.4 de [1]. ■

5.3 Implementação

Implementamos o algoritmo BETRA em FORTRAN 77 para comparar seu comportamento com LANCELOT, outro método para minimização de funções com restrições de caixa (veja [8]).

Para calcular o raio inicial da região de confiança, recebemos um parâmetro $\Delta_{inicial}$ e calculamos Δ da seguinte forma:

$$\Delta = \max\{\Delta_{min}, \Delta_{inicial} \max\{1, \|x^0\|\}\}.$$

BETRA possui 4 critérios de parada:

1. *Vértice estacionário de primeira ordem* (**Passo 3** do BETRA):
   A face corrente é um vértice e $\|g_P(x^k)\|_\infty \leq \varepsilon$;
2. *Ponto estacionário de primeira ordem perto da borda* (**Passo 2** do Algoritmo Interno):
   Valem as desigualdades $\Delta_{borda} < 2\Delta_{min}$ e $\|\bar{g}(x^k)\|_\infty \leq \varepsilon$;
3. *Ponto estacionário de segunda ordem* (**Passo 3** do Algoritmo Interno):
   d^k é solução do subproblema de regiões de confiança, $|\varphi_k(d^k)| < \varepsilon$ e $\|\bar{g}(x^k)\|_\infty < \varepsilon$;
4. *Ponto estacionário de primeira ordem* (**Passo 4** do BETRA):
   O teste (5.1) não vale (a face corrente deve ser abandonada) e $\|g_P(x^k)\|_\infty < \varepsilon$.

Nos experimentos utilizamos $\epsilon = 10^{-5}$ para MEQB; $\alpha = 10^{-4}$, $\lambda_{min}^{\text{SPG}} = 10^{-10}$, $\lambda_{max}^{\text{SPG}} = 10^{10}$, $\epsilon_{rel} = 10^{-7}$ e $\epsilon_{abs} = 10^{-10}$ para SPG; $\varepsilon = 10^{-5}$ para o critério de parada (para o teste (5.1) utilizamos norma infinito).

Para decidir os parâmetros restantes (η, $\Delta_{inicial}$, Δ_{min}, σ e N) testamos todas as combinações de $\eta \in \{0.1, 0.9\}$, $\Delta_{inicial} \in \{0.1, 1, 10, 100\}$, $\Delta_{min} \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, $\sigma \in \{0.1, 0.2\}$, e $N \in \{2, 4\}$. A combinação que forneceu os melhores resultados foi $\eta = 0.1$, $\Delta_{inicial} = 100$, $\Delta_{min} = 10^{-4}$, $\sigma = 0.2$, e $N = 4$. Extrapolação só é permitida quando o **Passo 6** do Algoritmo Interno é atingida pelos **Passos 3** e **5**. Levamos em consideração a robustez do BETRA para a escolha desses parâmetros.

5.4 Resultados numéricos

Para comparar o desempenho do BETRA e do LANCELOT, especificamos que este deveria usar segunda derivada exata e a opção "full-matrix-preconditioned-cg-solver-used" fosse ativada (para forçar que o LANCELOT decomponha a hessiana, como faz o BETRA). Para fazer tal comparação, utilizamos os problemas da coleção CUTE (Constrained and Unconstrained Testing Environment, veja [6]) com restrições de caixa e dimensões menores ou iguais a 120, totalizando 140 problemas. Como os tempos de execução de ambos os métodos para os 140 problemas selecionados são, em geral, muito pequenos, executamos cada par método/problema várias vezes, a fim de evitar erros na medição do tempo.

Todos os experimentos foram feitos utilizando o compilador g77 - GNU project Fortran Compiler (v0.5.24), com opção de compilação -O3 para otimizar o código. Usando um Pentium I com 133 MHZ, 64 Mb de memória RAM, com sistema operacional Linux.

Nas tabelas 5.1, 5.2 e 5.3 apresentamos o desempenho de BETRA e LANCELOT em todo o conjunto de problemas. Nas tabelas, temos nas colunas:
Problema (n): nome do problema e o número de variáveis;
naf: número de avaliações de função;
nag: número de avaliações de gradiente;
nah: número de avaliações de hessiana;
nch: número de fatorações de Cholesky;
ncg: número de iterações de Gradientes Conjugados;
Tempo: tempo de CPU em segundos;
$f(x^*)$: valor da função na melhor solução encontrada.

O número de avaliações de hessiana feitas pelo LANCELOT é **nag** - 1. As linhas duplas dividem os problemas de acordo com o tipo da função objetivo. As funções objetivo dos problemas do primeiro grupo são quadráticas, as do segundo grupo são soma de quadrados, e as do terceiro grupo são não-lineares.

Em média, 12.48% das avaliações de função foram feitas pelo SPG global, 3.87% foram feitas pelo SPG interno e 83.65% foram feitas por iterações de regiões de confiança do Algoritmo Interno. Dos 140 problemas testados, 33 problemas (23.57%) satisfizeram o critério *1*, 5 problemas (3.57%) satisfizeram o critério *2*, 98 (70.71%) satisfizeram o critério *3*, e nenhum satisfez o critério *4*. Os 3 problemas restantes pararam porque BETRA atingiu o número máximo de iterações permitidas (10000). Estes problemas foram: PALMER5A que parou com $\|g_P(x^k)\|_\infty$ = 1.8088E+00; PALMER7A, que parou com $\|g_P(x^k)\|_\infty$ = 4.0094E−01; e SINEALI (20) que parou com $\|g_P(x^k)\|_\infty$ = 2.6944E−03.

LANCELOT possui apenas um critério de parada com sucesso: a norma infinito do gradiente projetado deve menor que 10^{-5}. Em 137 dos 140 problemas testados, este critério foi satisfeito. Nos 3 problemas restantes, LANCELOT parou porque atingiu o número máximo de avaliações de função permitidas (10000). Estes problemas foram: PALMER5A que parou com $\|g_P(x^k)\|_\infty$ = 1.5959E−01; PALMER7E que parou com $\|g_P(x^k)\|_\infty$ = 5.9967E+00; e SINEALI (20) que parou com $\|g_P(x^k)\|_\infty$ = 7.9770E−04. É importante ressaltar que, apesar de BETRA possuir maior número de critérios de parada, os critérios diferentes daquele adotado por LANCELOT são mais fortes. Assim, BETRA não pode parar "antes" de LANCELOT e prejudicar a comparação entre ambos.

Para visualizar melhor a comparação do desempenho dos dois métodos, utilizamos o tempo de CPU gasto pelos métodos para resolver os problemas como fator de comparação e construímos um gráfico do perfil de desempenho (veja [9]). Neste gráfico, se um ponto da curva correspondente ao método M vale I no eixo das abscissas e J no eixo das ordenadas, significa que o método M é o I-ésimo método mais rápido em $(J \times 100)\%$ dos problemas. Os pontos mais significativos do gráfico são o mais à esquerda e o mais à direita. À esquerda medimos a *eficiência* dos métodos (verificamos a porcentagem de problemas para os quais cada método

	BETRA						LANCELOT				
Problema (n)	naf	nag	nah	nch	Tempo	$f(x^*)$	naf	nag	ncg	Tempo	$f(x^*)$
BQP1VAR (1)	2	2	1	1	0.000	0.0000E+00	3	3	0	0.004	0.0000E+00
HS3 (2)	3	4	3	4	0.001	7.8886E−36	8	8	0	0.009	9.5614E−37
HS3MOD (2)	3	4	3	5	0.001	7.8886E−31	4	4	3	0.008	4.1591E−32
OSLBQP (8)	2	2	2	2	0.001	6.2500E+00	3	3	0	0.008	6.2500E+00
SIM2BQP (2)	2	3	0	0	0.000	0.0000E+00	3	3	0	0.005	0.0000E+00
SIMBQP (2)	5	5	4	4	0.001	2.4074E−34	2	2	0	0.005	1.1132E−30
HATFLDA (4)	21	14	13	16	0.005	1.8367E−14	28	28	22	0.037	2.7495E−14
HATFLDB (4)	18	11	10	13	0.004	5.5728E−03	25	25	20	0.033	5.5728E−03
HS1 (2)	29	27	25	39	0.005	4.3945E−15	33	28	23	0.033	1.2242E−12
HS2 (2)	6	6	6	6	0.001	4.9412E+00	7	7	2	0.010	4.9412E+00
HS25 (3)	1	1	1	1	0.013	3.2835E+01	1	1	0	0.013	3.2835E+01
PALMER1 (4)	34	38	28	67	0.069	1.1755E+04	26	22	14	0.080	1.1755E+04
PALMER1A (6)	87	99	80	136	0.255	8.9883E−02	78	68	61	0.308	8.9883E−02
PALMER1B (4)	79	100	71	116	0.165	3.4473E+00	44	38	25	0.119	3.4473E+00
PALMER1E (8)	102	105	80	132	0.441	8.3523E−04	5	5	5	0.034	3.1103E+00
PALMER2 (4)	21	23	16	34	0.033	3.6511E+03	28	23	12	0.068	3.6511E+03
PALMER2A (6)	50	55	47	84	0.109	1.7110E−02	100	86	94	0.316	1.7110E−02
PALMER2B (4)	58	71	51	95	0.081	6.2327E−01	25	22	15	0.060	6.2327E−01
PALMER2E (8)	81	89	76	116	0.282	2.0650E−04	315	261	320	1.357	2.0650E−04
PALMER3A (6)	95	116	89	143	0.209	2.0431E−02	89	76	86	0.286	2.0431E−02
PALMER3B (4)	47	63	41	67	0.068	4.2276E+00	28	23	22	0.067	4.2276E+00
PALMER3E (8)	53	58	48	92	0.182	5.0741E−05	57	47	57	0.251	5.0741E−05
PALMER4 (4)	24	22	16	33	0.030	2.2854E+03	47	38	22	0.117	2.2854E+03
PALMER4A (6)	42	41	38	82	0.085	4.0606E−02	67	56	65	0.211	4.0606E−02
PALMER4B (4)	55	72	50	84	0.086	6.8351E+00	26	22	18	0.063	6.8351E+00
PALMER4E (8)	50	51	48	80	0.171	1.4800E−04	40	32	40	0.176	1.4800E−04
PALMER5A (8)	10029	15633	10000	10474	20.280	4.0560E−02	10001	8402	9975	27.250	3.1022E−02
PALMER5B (9)	646	872	627	886	1.326	9.7524E−03	123	101	114	0.377	9.7524E−03
PALMER5D (8)	3	3	3	5	0.002	8.7339E+01	2	2	1	0.010	8.7339E+01
PALMER5E (8)	1437	1989	1428	1511	3.015	2.2937E−02	8763	7212	8762	24.970	2.0716E−02
PALMER6A (6)	124	143	117	188	0.171	5.5949E−02	141	120	134	0.317	5.5949E−02
PALMER6E (8)	82	101	80	111	0.185	2.2395E−04	61	50	62	0.189	2.2395E−04
PALMER7A (6)	10008	15505	10000	10116	15.220	1.0345E+01	4007	3569	4008	8.910	1.0335E+01
PALMER7E (8)	6	7	6	17	0.013	1.0154E+01	10001	8065	9930	29.120	6.5607E+00
PALMER8A (6)	38	40	36	68	0.049	7.4010E−02	52	45	55	0.119	7.4010E−02
PALMER8E (8)	27	37	26	52	0.058	6.3393E−03	41	34	41	0.124	6.3393E−03
PSPDOC (4)	8	6	6	11	0.002	2.4142E+00	10	10	9	0.015	2.4142E+00
WEEDS (3)	23	8	5	10	0.007	9.2054E+03	25	22	27	0.050	2.5873E+00
YFIT (3)	60	63	50	93	0.086	6.6697E−13	51	42	50	0.112	6.6698E−13
ALLINIT (4)	12	11	10	13	0.007	1.6706E+01	12	10	6	0.017	1.6706E+01
CAMEL6 (2)	37	30	30	49	0.010	-1.0316E+00	19	11	12	0.020	-1.0316E+00
HART6 (6)	15	16	13	22	0.013	-3.3229E+00	9	7	7	0.019	-3.3229E+00
HIMMELP1 (2)	9	7	6	8	0.002	-6.2054E+01	16	15	5	0.014	-6.2054E+01
HS38 (4)	55	54	49	91	0.023	1.9971E−20	54	46	56	0.069	5.2378E−20
HS4 (2)	2	3	1	2	0.000	2.6667E+00	2	2	0	0.004	2.6667E+00
HS45 (5)	2	3	1	13	0.001	1.0000E+00	9	9	0	0.009	1.0000E+00
HS5 (2)	10	9	8	16	0.002	-1.9132E+00	6	6	3	0.009	-1.9132E+00
LOGROS (2)	73	59	54	127	0.018	0.0000E+00	56	45	35	0.054	0.0000E+00
MAXLIKA (8)	56	56	35	61	4.017	1.1363E+03	9	8	19	0.659	1.1493E+03
MDHOLE (2)	39	43	37	65	0.010	4.8148E−33	61	51	52	0.058	3.0080E−37
S368 (8)	6	9	6	8	0.026	-9.3750E−01	6	6	3	0.029	-9.3750E−01

Tabela 5.1: Desempenho de BETRA e LANCELOT nos problemas com menos de 10 variáveis.

	BETRA						LANCELOT				
Problema (n)	naf	nag	nah	nch	Tempo	$f(x^*)$	naf	nag	ncg	Tempo	$f(x^*)$
BIGGSB1 (25)	42	28	15	15	0.026	1.5000E−02	15	15	14	0.041	1.5000E−02
BQPGABIM (50)	60	46	0	0	0.053	-3.7903E−05	3	3	4	0.042	-3.7903E−05
BQPGASIM (50)	56	45	0	0	0.054	-5.5198E−05	3	3	3	0.040	-5.5198E−05
CHENHARK (10)	8	8	5	5	0.004	-2.0000E+00	2	2	5	0.011	-2.0000E+00
CVXBQP1 (10)	3	3	1	3	0.001	2.4750E+00	2	2	0	0.009	2.4750E+00
HARKERP2 (10)	9	11	9	9	0.011	-5.0000E−01	3	3	1	0.011	-5.0000E−01
JNLBRNG1 (16)	2	2	2	2	0.002	-2.2474E−01	2	2	2	0.012	-2.2474E−01
JNLBRNG2 (16)	2	2	2	2	0.002	-4.7640E+00	2	2	0	0.011	-4.7640E+00
JNLBRNGA (16)	3	4	2	2	0.002	-5.0967E−01	5	5	0	0.013	-5.0967E−01
JNLBRNGB (16)	3	4	2	2	0.002	-1.8551E+01	5	5	1	0.013	-1.8551E+01
NCVXBQP1 (10)	2	3	1	1	0.001	-2.2050E+04	3	3	0	0.010	-2.2050E+04
NCVXBQP2 (10)	2	3	1	1	0.001	-1.4382E+04	3	3	0	0.010	-1.4382E+04
NCVXBQP3 (10)	3	4	1	1	0.001	-1.1958E+04	3	3	0	0.010	-1.1958E+04
NOBNDTOR (16)	2	2	2	2	0.001	-5.4321E−01	3	3	0	0.011	-5.4321E−01
OBSTCLAE (16)	2	3	1	1	0.001	7.5366E−01	3	3	0	0.010	7.5366E−01
OBSTCLAL (16)	1	1	0	0	0.000	7.5366E−01	1	1	0	0.008	7.5366E−01
OBSTCLBL (16)	2	3	0	0	0.000	-8.1108E−03	2	2	0	0.009	-8.1108E−03
OBSTCLBM (16)	2	3	1	1	0.001	-8.1108E−03	2	2	0	0.010	-8.1108E−03
OBSTCLBU (16)	1	1	0	0	0.000	-8.1108E−03	1	1	0	0.009	-8.1108E−03
PENTDI (50)	3	4	2	2	0.007	-7.5000E−01	2	2	0	0.023	-7.5000E−01
QUDLIN (12)	2	3	0	0	0.000	-7.2000E+03	2	2	0	0.011	-7.2000E+03
TORSION1 (16)	1	1	0	0	0.000	-5.1852E−01	1	1	0	0.009	-5.1852E−01
TORSION2 (16)	2	2	1	1	0.001	-5.1852E−01	3	3	0	0.010	-5.1852E−01
TORSION3 (16)	1	1	0	0	0.000	-1.2593E+00	1	1	0	0.008	-1.2593E+00
TORSION4 (16)	2	2	1	1	0.001	-1.2593E+00	3	3	0	0.010	-1.2593E+00
TORSION5 (16)	1	1	0	0	0.000	-2.7407E+00	1	1	0	0.008	-2.7407E+00
TORSION6 (16)	2	2	1	1	0.001	-2.7407E+00	2	2	0	0.009	-2.7407E+00
TORSIONA (16)	3	4	2	2	0.003	-3.0864E−01	2	2	0	0.011	-3.0864E−01
TORSIONB (16)	2	2	2	2	0.003	-3.0864E−01	3	3	0	0.012	-3.0864E−01
TORSIONC (16)	1	1	0	0	0.000	-1.0370E+00	1	1	0	0.008	-1.0370E+00
TORSIOND (16)	2	2	1	1	0.002	-1.0370E+00	3	3	0	0.011	-1.0370E+00
TORSIONE (16)	1	1	0	0	0.000	-2.5185E+00	1	1	0	0.009	-2.5185E+00
TORSIONF (16)	2	2	1	1	0.002	-2.5185E+00	2	2	0	0.010	-2.5185E+00
CHEBYQAD (50)	27	24	22	103	11.300	5.3863E−03	106	87	108	43.310	5.3863E−03
DECONVB (61)	12006	7809	22	52	29.894	5.3548E−08	20	16	128	0.852	5.6636E−03
HATFLDC (25)	5	5	5	5	0.013	8.2639E−14	5	5	3	0.026	7.7700E−19
HS110 (10)	7	6	6	8	0.009	-4.5778E+01	5	5	0	0.016	-4.5778E+01
LINVERSE (19)	6	6	6	9	0.017	7.0000E+00	10	8	16	0.055	7.0000E+00
NONSCOMP (50)	39	23	7	7	0.078	1.1916E−11	9	9	8	0.065	1.1212E−16
QR3DLS (40)	23	21	20	43	0.324	5.5383E−16	29	24	28	0.477	1.2807E−13
EXPLIN (12)	11	12	9	14	0.006	-6.8500E+03	13	11	13	0.024	-6.8500E+03
EXPLIN2 (12)	13	14	9	16	0.007	-7.0925E+03	14	13	16	0.027	-7.0925E+03
EXPQUAD (12)	11	14	10	45	0.019	-4.2011E+03	13	11	20	0.033	-4.2011E+03
HADAMALS (16)	17	20	16	22	0.045	0.0000E+00	8	8	37	0.046	0.0000E+00
MCCORMCK (10)	7	7	7	11	0.008	-9.5980E+00	5	5	4	0.016	-9.5980E+00
PROBPENL (10)	34	25	19	62	0.030	-3.1787E+05	5	3	13	0.020	1.5235E−05
QRTQUAD (12)	35	33	20	30	0.020	-3.6077E+03	71	56	64	0.117	-3.6077E+03
SINEALI (20)	10221	10626	10000	11627	23.720	-1.9010E+03	10001	8887	9994	26.070	-1.9010E+03

Tabela 5.2: Desempenho de BETRA e LANCELOT nos problemas com mais de 10 e menos de 100 variáveis.

	BETRA						LANCELOT				
Problema (n)	naf	nag	nah	nch	Tempo	$f(x^*)$	naf	nag	ncg	Tempo	$f(x^*)$
BIGGSB1 (100)	284	103	52	52	0.950	1.5000E−02	52	52	50	0.323	1.5000E−02
CHENHARK (100)	3	4	3	6	0.145	-2.0000E+00	25	25	61	0.279	-2.0000E+00
CVXBQP1 (100)	3	3	1	3	0.104	2.2725E+02	2	2	0	0.055	2.2725E+02
HARKERP2 (100)	7	9	7	7	12.570	-5.0000E−01	2	2	2	0.223	-5.0000E−01
JNLBRNG1 (100)	5	4	3	3	0.050	-1.7896E−01	2	2	1	0.056	-1.7896E−01
JNLBRNG2 (100)	4	4	3	3	0.046	-3.9528E+00	3	3	2	0.066	-3.9528E+00
JNLBRNGA (100)	6	6	3	3	0.044	-3.6116E−01	3	3	2	0.061	-3.6116E−01
JNLBRNGB (100)	3	4	2	2	0.026	-7.2552E+00	4	4	3	0.069	-7.2552E+00
NCVXBQP1 (100)	2	3	1	1	0.045	-1.9956E+06	2	2	0	0.057	-1.9956E+06
NCVXBQP2 (100)	6	7	3	6	0.058	-1.3330E+06	3	3	4	0.061	-1.3330E+06
NCVXBQP3 (100)	7	9	5	6	0.066	-6.7085E+05	4	4	4	0.064	-6.7085E+05
NOBNDTOR (100)	8	6	4	4	0.060	-5.5211E−01	3	3	2	0.067	-5.5211E−01
OBSTCLAE (100)	9	9	5	5	0.079	1.3979E+00	3	3	29	0.138	1.3979E+00
OBSTCLAL (100)	9	8	4	4	0.060	1.3979E+00	4	4	3	0.071	1.3979E+00
OBSTCLBL (100)	7	10	5	5	0.056	2.8750E+00	3	3	6	0.067	2.8750E+00
OBSTCLBM (100)	3	4	3	3	0.043	2.8750E+00	2	2	3	0.060	2.8750E+00
OBSTCLBU (100)	4	6	3	3	0.036	2.8750E+00	2	2	1	0.053	2.8750E+00
TORSION1 (100)	5	6	3	3	0.040	-4.9234E−01	3	3	2	0.059	-4.9234E−01
TORSION2 (100)	9	7	4	4	0.061	-4.9234E−01	4	4	2	0.073	-4.9234E−01
TORSION3 (100)	3	4	2	2	0.022	-1.2705E+00	2	2	1	0.051	-1.2705E+00
TORSION4 (100)	7	5	3	3	0.046	-1.2705E+00	4	4	2	0.066	-1.2705E+00
TORSION5 (100)	1	1	0	0	0.002	-2.8971E+00	1	1	0	0.044	-2.8971E+00
TORSION6 (100)	2	3	1	1	0.025	-2.8971E+00	3	3	0	0.056	-2.8971E+00
TORSIONA (100)	5	6	3	3	0.050	-4.0570E−01	3	3	2	0.066	-4.0570E−01
TORSIONB (100)	5	5	5	5	0.095	-4.0570E−01	5	5	6	0.114	-4.0570E−01
TORSIONC (100)	3	4	2	2	0.027	-1.1766E+00	2	2	1	0.054	-1.1766E+00
TORSIOND (100)	7	5	3	3	0.054	-1.1766E+00	4	4	3	0.076	-1.1766E+00
TORSIONE (100)	1	1	0	0	0.002	-2.7984E+00	1	1	0	0.045	-2.7984E+00
TORSIONF (100)	2	3	1	1	0.029	-2.7984E+00	3	3	0	0.059	-2.7984E+00
HS110 (100)	2	3	1	2	0.109	-9.9800E+19	2	2	0	0.054	-9.9800E+19
NONSCOMP (100)	47	25	7	7	0.337	8.0101E−12	9	9	8	0.130	2.6015E−16
BDEXP (100)	13	12	12	16	0.696	1.3497E−05	11	11	10	0.192	3.9646E−05
EXPLIN (120)	29	47	25	31	0.107	-7.2376E+05	14	14	61	0.113	-7.2324E+05
EXPLIN2 (120)	17	19	13	61	0.056	-7.2446E+05	12	12	30	0.093	-7.2446E+05
EXPQUAD (120)	22	24	17	32	1.642	-3.6260E+06	18	15	49	0.354	-3.6260E+06
HADAMALS (100)	29	43	26	49	2.904	2.5316E+01	14	14	370	2.610	2.5316E+01
MCCORMCK (100)	8	7	7	12	0.472	-9.1788E+01	7	6	5	0.117	-9.1788E+01
PROBPENL (100)	4	5	4	13	0.673	-4.9571E−06	91	52	268	4.923	-2.8726E+05
QRTQUAD (120)	56	70	42	66	3.783	-3.6246E+06	205	168	195	2.469	-3.6242E+06
S368 (100)	7	10	7	10	6.630	-1.4869E+02	9	7	9	7.007	-1.3369E+02
SINEALI (100)	11	12	10	24	0.890	-9.9010E+03	13	9	6	0.138	-9.9010E+03

Tabela 5.3: Desempenho de BETRA e LANCELOT nos problemas com mais de 100 variáveis.

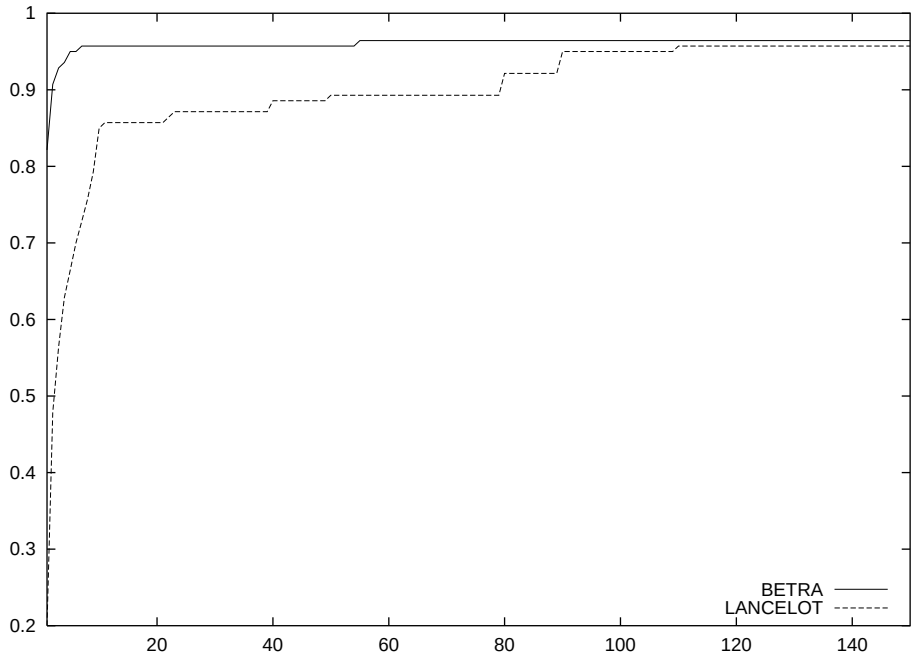

Figura 5.1: Curva do perfil de desempenho comparando BETRA e LANCELOT para todos os problemas

é o mais rápido dentre todos os comparados) e à esquerda medimos a *robustez* (verificamos a porcentagem de problemas que cada método foi capaz de resolver).

Para a construção do gráfico 5.1, levou-se em conta não somente o tempo de CPU gasto por cada método, mas também a solução encontrada por cada um deles. Quando tanto BETRA como LANCELOT encontram o mesmo valor de função, comparamos o tempo gasto por cada um para encontrá-la. Quando encontram soluções diferentes, considera-se que aquele que encontrou solução pior (maior valor de função) levou tempo "infinito" para chegar à solução. Definimos que dois valores de função f_1 e f_2 são iguais se e somente se

$$| f_1 - f_2 | \leq \max\{10^{-10}, 10^{-6} \min\{| f_1 |, | f_2 |\}\}.$$

No gráfico 5.2, utilizando a mesma definição de igualdade de soluções apresentada acima, comparamos os tempos gastos pelos dois métodos apenas nos problemas nos quais ambos obtiveram a mesma solução e ambos pararam sua execução com sucesso (perfazendo um total de 127 problemas).

No gráfico 5.1 podemos observar que BETRA é mais rápido que LANCELOT em 82.14% dos problemas, enquanto LANCELOT foi mais rápido que BETRA em 19.28% dos problemas. Pode-se verificar também que BETRA solucionou 96.43% dos problemas enquanto LANCELOT solucionou 95.71%. Considerando apenas os problemas presentes no gráfico 5.2, BETRA é mais rápido que LANCELOT em 85.04% dos casos e LANCELOT é mais rápido que BETRA em 16.53%.

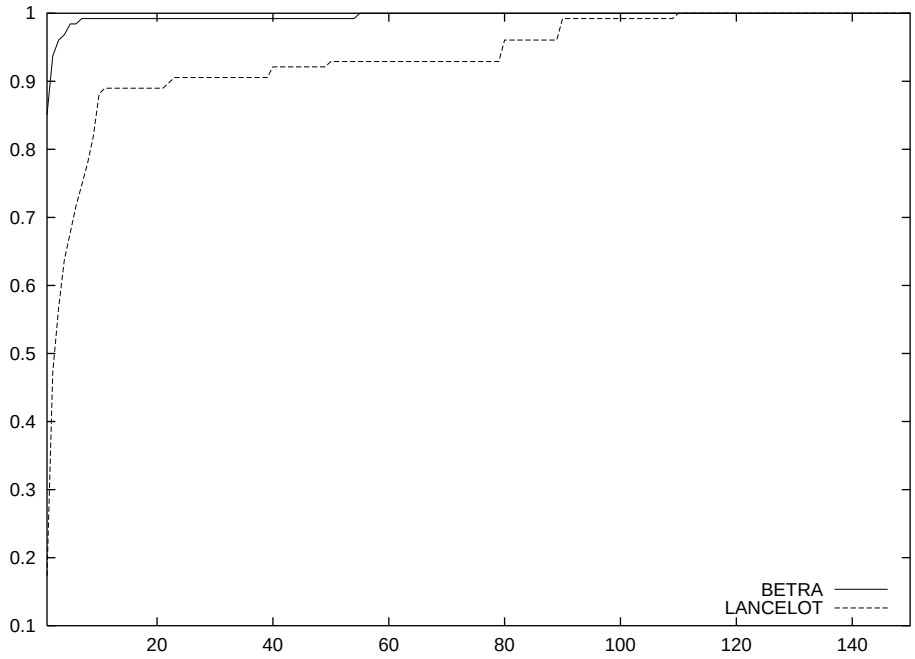

Figura 5.2: Curva do perfil de desempenho comparando BETRA e LANCELOT somente para os problemas nos quais ambos pararam com sucesso e com a mesma solução

É importante notar que, quando se aumenta o valor de η, o número de iterações do SPG global tende a aumentar, e, quando se aumenta o valor de Δ_{min}, o número de iterações do SPG interno também tende a aumentar. Apesar disso, quando aumentamos o valor desses 2 parâmetros, apenas para um pequeno número de problemas o BETRA deixa de encontrar como solução um ponto que satisfaz condições necessárias de segunda ordem. Como exemplo, testamos os 140 problemas mantendo os parâmetros escolhidos para os outros experimentos e modificando apenas os valores de η e Δ_{min}. Para $\eta = 0.01$ e $\Delta_{min} = 10^{-5}$, 33 problemas pararam pelo critério *1*, 5 pelo critério *2*, 97 pelo critério *3*, nenhum pelo critério *4* e 5 atingiram o número máximo de iterações. Para $\eta = 0.99$ e $\Delta_{min} = 10^{-3}$, o número de problemas que pararam satisfazendo os critérios *1* e *2* se manteve, o número de problemas que pararam pelo critério *3* passou para 96, o número de problemas que pararam pelo critério *4* passou para 2 e o número de problemas que atingiram o número máximo de iterações passou para 4. Portanto, apesar de não ser fácil a escolha destes dois parâmetros, a robustez do BETRA não depende fortemente deles.

Conclusões e trabalho futuro

Apresentamos um método prático de restrições ativas para resolver problemas de minimização suave com restrições de caixa, onde o algoritmo interno é um método de regiões de confiança euclidiano clássico e as iterações para deixar as faces são do tipo Gradiente Espectral Projetado. Passos de Gradiente Espectral Projetado também são usados quando algumas restrições inativas são "quase" ativas. Provamos convergência a pontos estacionários de primeira ordem e, sob hipóteses adequadas, a pontos estacionários de segunda ordem. No último caso, os resultados de ordem de convergência de métodos de regiões de confiança de segunda ordem são mantidos.

Comparamos o método com um algoritmo conhecido que usa hessianas verdadeiras e precondicionador completo (ou seja, decomposições de Cholesky) para resolver problemas com restrições de caixa. Os resultados são promissores.

Regiões de confiança euclidianas e elipsoidais têm a vantagem, sobre regiões de confiança com caixas, de minimizadores *globais* dos subproblemas correspondentes poderem ser obtidos utilizando algoritmos estáveis e eficientes. Por outro lado, bolas e elipsóides não apresentam a mesma forma do domínio do problema, como acontece com regiões de confiança com caixas. Portanto, a vantagem de cada uma das abordagens na prática é controversa e, provavelmente, a decisão correta depende fortemente do problema.

Neste trabalho optamos por regiões de confiança euclidianas, mas também observamos que não há razões para manter os iterandos interiores, com tanto que tenhamos procedimentos eficientes para deixar as faces e para extrapolação, ou seja, para ganhar e abandonar restrições. Aparentemente os resultados numéricos confirmam que esta abordagem é válida para minimização geral com restrições em caixa.

A continuação deste trabalho segue em duas direções: primeiro, considerar problemas de grande porte usando "solvers" de regiões de confiança de grande porte [20]. Segundo, estender a estratégia apresentada neste trabalho para politopos em geral.

Apêndice A

Provas dos lemas do capítulo 1

Lema 1.1.1 *Se p é uma solução para (1.1) então p é uma solução para uma equação da forma*

$$(B + \lambda I)\, p = -g \tag{1.2}$$

com $B + \lambda I$ semidefinida positiva, $\lambda \geq 0$ e $\lambda(\Delta - \|p\|) = 0$.

Prova Seja p uma solução de (1.1).

Como o problema (1.1) é um problema de minimização com restrições, temos que p é solução de (1.1) se e somente se p satisfaz as condições necessárias de primeira ordem, ou seja, as condições de *Karush-Kuhn-Tucker*, a saber,

$$\nabla_w \mathcal{L}(p, \lambda) = 0 \tag{A.1}$$
$$c(p) \geq 0 \tag{A.2}$$
$$\lambda \geq 0 \tag{A.3}$$
$$c(p)\lambda = 0 \tag{A.4}$$

onde $c(w) = \Delta - \|w\|$ e $\mathcal{L}(w, \lambda) = \varphi(w) - \lambda c(w)$.

Note que

$$\nabla_w \mathcal{L}(w, \lambda) = \nabla_w \varphi(w) - \lambda \nabla_w c(w).$$

Como $\nabla_w \varphi(w) = Bw + g$ e $\nabla_w c(w) = -2w$, temos que

$$\nabla_w \mathcal{L}(w, \lambda) = Bw + g + 2\lambda w.$$

Assim, por (A.1), temos

$$\nabla_w \mathcal{L}(p, \lambda) = Bp + g + 2\lambda p = 0.$$

Ou seja,

$$Bp + 2\lambda p = -g \Rightarrow (B + 2\lambda I)p = -g.$$

Por (A.4) temos que $c(p)\lambda = 0$. Ou seja,

$$(\Delta - \|p\|)\lambda = 0 \Rightarrow 2\lambda(\Delta - \|p\|) = 0.$$

Por (A.3) temos que $\lambda \geq 0$. Assim, temos que $2\lambda \geq 0$.

Trocando 2λ por λ temos que p satisfaz (1.2) com $\lambda \geq 0$ e $\lambda\,(\Delta - \|p\|) = 0$.

Agora basta provar que $B + \lambda I$ é semidefinida positiva.

Suponha que $p \neq 0$. Como p é solução de (1.1), temos que p é solução de

$$\begin{array}{ll} \text{minimizar} & \varphi(w) \\ \text{sujeita a} & \|w\| \leq \|p\|. \end{array} \tag{A.5}$$

Ou seja, para qualquer w com $\|w\| = \|p\|$, temos que

$$\varphi(w) \geq \varphi(p) \Rightarrow$$

$$\frac{1}{2}w^T Bw + g^T w \geq \frac{1}{2}p^T Bp + g^T p.$$

Como $g = -(B + \lambda I)p$, temos a seguinte desigualdade

$$\frac{1}{2}w^T Bw - ((B + \lambda I)p)^T w \geq \frac{1}{2}p^T Bp - ((B + \lambda I)p)^T p \Rightarrow$$

$$\frac{1}{2}w^T Bw - p^T(B + \lambda I)^T w \geq \frac{1}{2}p^T Bp - p^T(B + \lambda I)^T p.$$

Como B é simétrica, $B + \lambda I$ também é simétrica, ou seja, $(B + \lambda I)^T = (B + \lambda I)$. Assim, temos

$$\frac{1}{2}w^T Bw - p^T(B + \lambda I)w \geq \frac{1}{2}p^T Bp - p^T(B + \lambda I)p \Rightarrow$$

$$\frac{1}{2}w^T Bw - p^T Bw - \lambda p^T w \geq \frac{1}{2}p^T Bp - p^T Bp - \lambda p^T p = -\frac{1}{2}p^T Bp - \lambda p^T p \Rightarrow$$

$$\frac{1}{2}w^TBw - p^TBw + \frac{1}{2}p^TBp \geq -\lambda p^Tp + \lambda p^Tw \Rightarrow$$

$$\frac{1}{2}(w-p)^TB(w-p) \geq -\lambda p^Tp + \lambda p^Tw.$$

Mas $\frac{1}{2}(w-p)^TB(w-p) = \frac{1}{2}(w-p)^T(B+\lambda I)(w-p) - \frac{\lambda}{2}(w-p)^T(w-p)$.

$$\frac{1}{2}(w-p)^T(B+\lambda I)(w-p) - \frac{\lambda}{2}(w-p)^T(w-p) \geq -\lambda p^Tp + \lambda p^Tw \Rightarrow$$

$$\frac{1}{2}(w-p)^T(B+\lambda I)(w-p) \geq -\lambda p^Tp + \lambda p^Tw + \frac{\lambda}{2}(w-p)^T(w-p) \Rightarrow$$

$$\frac{1}{2}(w-p)^T(B+\lambda I)(w-p) \geq -\lambda p^Tp + \lambda p^Tw + \frac{\lambda}{2}w^Tw - \lambda p^Tw + \frac{\lambda}{2}p^Tp \Rightarrow$$

$$\frac{1}{2}(w-p)^T(B+\lambda I)(w-p) \geq \frac{\lambda}{2}(w^Tw - p^Tp) = \frac{\lambda}{2}(\|w\|^2 - \|p\|^2).$$

Como $\|w\| = \|p\|$, temos que

$$\frac{1}{2}(w-p)^T(B+\lambda I)(w-p) \geq 0.$$

Assim, $B + \lambda I$ é semidefinida positiva quando $p \neq 0$.

Se $p = 0$, como $g = -(B+\lambda I)p$, temos que $g = 0$.

Desta forma, como p é solução de (1.1), temos que p também é solução de

$$\begin{array}{ll} \text{minimizar} & \frac{1}{2}w^TBw \\ \text{sujeita a} & \|w\| \leq \Delta \end{array} \tag{A.6}$$

Mas (A.6) só tem solução se B é semidefinida positiva. Como $\lambda \geq 0$, $B + \lambda I$ também é semidefinida positiva. ■

Lema 1.1.2 *Sejam $\lambda \in \mathbb{R}$ e $p \in \mathbb{R}^n$ satisfazendo (1.2) com $B + \lambda I$ semidefinida positiva.*

(i) Se $\lambda = 0$ e $\|p\| \leq \Delta$ então p é solução de (1.1);

(ii) p é solução de $\min\{\varphi(w) : \|w\| \leq \|p\|\}$;

(iii) se $\lambda \geq 0$ e $\|p\| = \Delta$ então p é solução de (1.1).

Se $B + \lambda I$ é **definida** *positiva, p é único em (i), (ii) e (iii).*

Prova Se $(B + \lambda I)p = -g$ e $B + \lambda I$ é semidefinida positiva, p é o mínimo de uma quadrática que tem como hessiana $B + \lambda I$ e $(B + \lambda I)w + g$ como gradiente.

Ou seja, p é o minimizador da quadrática

$$\frac{1}{2}w^T(B + \lambda I)w + g^T w.$$

Neste caso, para qualquer $w \in I\!R^n$ vale que

$$\frac{1}{2}w^T(B + \lambda I)w + g^T w \geq \frac{1}{2}p^T(B + \lambda I)p + g^T p \Rightarrow$$

$$\frac{1}{2}w^T Bw + \frac{\lambda}{2}w^T w + g^T w \geq \frac{1}{2}p^T Bp + \frac{\lambda}{2}p^T p + g^T p.$$

Como

$$\frac{1}{2}w^T Bw + g^T w = \varphi(w) \quad \text{e} \quad \frac{1}{2}p^T Bp + g^T p = \varphi(p),$$

temos que

$$\varphi(w) + \frac{\lambda}{2}w^T w \geq \varphi(p) + \frac{\lambda}{2}p^T p \Rightarrow$$

$$\varphi(w) \geq \varphi(p) + \frac{\lambda}{2}p^T p - \frac{\lambda}{2}w^T w \Rightarrow$$

$$\varphi(w) \geq \varphi(p) + \frac{\lambda}{2}(p^T p - w^T w). \tag{A.7}$$

Agora, com base na inequação (A.7), podem-se provar os itens *(i)*, *(ii)* e *(iii)* do Lema 1.1.2.

(i) Se $\lambda = 0$ então, por (A.7), temos que $\varphi(w) \geq \varphi(p)$, ou seja, p é minimizador irrestrito de $\varphi(w)$. Como $\|p\| \leq \Delta$, temos que p é solução de (1.1).

(ii) Se considerarmos apenas os vetores $w \in I\!R^n$ tais que $\|w\| \leq \|p\|$, temos que $\|p\|^2 - \|w\|^2 \geq 0$. Por (A.7),

$$\varphi(w) \geq \varphi(p) + \frac{\lambda}{2}(p^T p - w^T w) \geq \varphi(p).$$

Portanto, p é solução de $\min\{\varphi(w) : \|w\| \leq \|p\|\}$;

(iii) se $\|p\| = \Delta$ então, pelo item *(ii)*, temos que p é solução de (1.1).

É fácil ver que se $B + \lambda I$ é **definida** positiva, então as soluções para *(i)*, *(ii)* e *(iii)* são únicas, pois na inequação (A.7) tem-se "$<$" no lugar de "$\leq$". ■

Lema 1.1.3 *O problema (1.1)* **não** *tem solução na borda de* $\{w : \|w\| \leq \Delta\}$ *se e somente se* B *é* **definida** *positiva e* $\|B^{-1}g\| < \Delta$.

Prova

(i) *Se* B *é* **definida** *positiva e* $\|B^{-1}g\| < \Delta$ *então o problema (1.1)* **não** *tem solução na borda de* $\{w : \|w\| \leq \Delta\}$.

Se B é **definida** positiva, $\varphi(w)$ é convexa e seu único ponto estacionário é um minimizador, dado por $p = -B^{-1}g$. Se $\|p\| < \Delta$ então p é o minimizador de $\varphi(w)$ sujeita a $\|w\| \leq \Delta$. Ou seja, p é solução do problema (1.1) que não está na borda de $\{w : \|w\| \leq \Delta\}$.

(ii) *Se o problema (1.1)* **não** *tem solução na borda de* $\{w : \|w\| \leq \Delta\}$ *então* B *é* **definida** *positiva e* $\|B^{-1}g\| < \Delta$.

Seja p uma solução do problema (1.1).

Como, por hipótese, (1.1) não tem solução na borda de $\{w : \|w\| \leq \Delta\}$, temos que $\|p\| < \Delta$.

Como vale o Lema 1.1.1, temos que

$$\lambda(\Delta - \|p\|) = 0 \Rightarrow \lambda = 0.$$

Além disso, temos que $B + \lambda I$ é semidefinida positiva. Como $\lambda = 0$, na verdade B é semidefinida positiva.

Agora basta provar que B é não-singular. Desta forma temos que B é **definida** positiva e, portanto, inversível. Daí, temos que, como $Bp = -g$, $p = -B^{-1}g$ que, por hipótese, tem norma menor do que Δ.

Suponha, por absurdo, que B seja singular. Neste caso, existe algum z com $\|p+z\| = \Delta$ tal que $Bz = 0$.

Mas como $\lambda = 0$ e $B+\lambda I$ é semidefinida positiva, pelo Lema 1.1.2-*(iii)* temos que $(p+z)$ é solução de (1.1) na borda, o que contradiz a hipótese de que (1.1) não tem solução na borda.

Portanto, B é não-singular. Ou seja, B é **definida** positiva. ■

Lema 1.1.4 *Seja $p(\lambda) = -(B + \lambda I)^{-1}g$ com $B + \lambda I$ definida positiva.*

Tome a decomposição $B = Q\Lambda Q^T$, com $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$ e $Q^TQ = I$, onde $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ são os autovalores de B (que existe porque B é simétrica).

Então, para $\lambda \neq -\lambda_j$,

$$p(\lambda) = -\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j,$$

onde q_j é a j-ésima coluna de Q.

Prova Como $B = Q\Lambda Q^T$, temos que

$$B + \lambda I = Q\Lambda Q^T + \lambda I.$$

Além disso, $Q^TQ = I$. Assim,

$$B + \lambda I = Q\Lambda Q^T + \lambda Q^TQ = Q\Lambda Q^T + \lambda QQ^T = Q\Lambda Q^T + Q(\lambda I)Q^T = Q(\Lambda + \lambda I)Q^T.$$

Então temos que

$$p(\lambda) = -(Q(\Lambda + \lambda I)Q^T)^{-1}g = -Q^{-T}(\Lambda + \lambda I)^{-1}Q^{-1}g = -Q(\Lambda + \lambda I)^{-1}Q^T g.$$

Note que

$$(\Lambda + \lambda I)^{-1} = \begin{bmatrix} \frac{1}{\lambda_1+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda_n+\lambda} \end{bmatrix}.$$

Assim,

$$p(\lambda) = -Q \begin{bmatrix} \frac{1}{\lambda_1+\lambda} & 0 & \cdots & 0 \\ 0 & \frac{1}{\lambda_2+\lambda} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\lambda_n+\lambda} \end{bmatrix} Q^T g =$$

$$p(\lambda) = -\left[\begin{array}{c|c|c|c} \frac{1}{\lambda_1+\lambda} q_1 & \frac{1}{\lambda_2+\lambda} q_2 & \cdots & \frac{1}{\lambda_n+\lambda} q_n \end{array} \right] Q^T g,$$

onde q_j é a j-ésima coluna de Q.

$$p(\lambda) = -\left[\begin{array}{c|c|c|c} \frac{1}{\lambda_1+\lambda}q_1 & \frac{1}{\lambda_2+\lambda}q_2 & \cdots & \frac{1}{\lambda_n+\lambda}q_n \end{array} \right] \left[\begin{array}{c} q_1^T g \\ q_2^T g \\ \vdots \\ q_n^T g \end{array} \right] = -\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j,$$

para $\lambda \neq -\lambda_j$. ■

Lema 1.2.1 *Seja* $\phi(\lambda) = \frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}$, *onde* $p(\lambda) = -(B + \lambda I)^{-1} g$ *com* B *simétrica e* $B + \lambda I$ *definida positiva.*

Então,

$$-\frac{\phi(\lambda)}{\phi'(\lambda)} = \left(\frac{\|p(\lambda)\|}{\|t(\lambda)\|}\right)^2 \left(\frac{\|p(\lambda)\| - \Delta}{\Delta}\right),$$

onde $t(\lambda)$ *é a solução de* $R^T t(\lambda) = p(\lambda)$ *e* R *é a matriz triangular superior tal que* $B + \lambda I = R^T R$.

Prova Primeiramente note que, se $B + \lambda I$ é definida positiva, necessariamente temos que $\lambda \neq -\lambda_j$, para $j = 1, ..., n$.

Assim, pelo Lema 1.1.4, temos que $p(\lambda) = -\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j$.

Note que

$$\|p(\lambda)\|^2 = \left(\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j\right)^T \left(\sum_{j=1}^{n} \frac{q_j^T g}{\lambda_j + \lambda} q_j\right).$$

Como $Q^T Q = I$, temos que $(q_i)^T q_i = 1$ para todo i e $(q_j)^T q_i = 0$ para todo $j \neq i$.

Assim,

$$\|p(\lambda)\|^2 = \sum_{j=1}^{n} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2}.$$

É fácil verificar que

$$\left(\|p(\lambda)\|^2\right)' = -2\sum_{j=1}^{n} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^3}. \tag{A.8}$$

Vejamos agora como podemos escrever $\phi'(\lambda)$.

$$\phi'(\lambda) = -\left(\frac{1}{\|p(\lambda)\|}\right)' = -\left((\|p(\lambda)\|^2)^{-\frac{1}{2}}\right)' = \frac{1}{2}(\|p(\lambda)\|^2)^{-\frac{3}{2}}(\|p(\lambda)\|^2)' = \frac{1}{2}\|p(\lambda)\|^{-3}(\|p(\lambda)\|^2)'.$$

Usando a equação (A.8), obtemos

$$\phi^{'}(\lambda) = \frac{1}{2}\|p(\lambda)\|^{-3}\left(-2\sum_{j=1}^{n}\frac{(q_j^T g)^2}{(\lambda_j+\lambda)^3}\right) = -\|p(\lambda)\|^{-3}\left(\sum_{j=1}^{n}\frac{(q_j^T g)^2}{(\lambda_j+\lambda)^3}\right). \qquad \text{(A.9)}$$

Observe ainda que

$$\|t(\lambda)\|^2 = \|R^{-T}p(\lambda)\|^2 = (R^{-T}p(\lambda))^T R^{-T}p(\lambda) = p(\lambda)^T R^{-1}R^{-T}p(\lambda) =$$

$$p(\lambda)^T(R^T R)^{-1}p(\lambda) = p(\lambda)^T(B+\lambda I)^{-1}p(\lambda).$$

Como $p(\lambda) = -\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}q_j$ (pelo Lema 1.1.4), temos que

$$\|t(\lambda)\|^2 = \left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}q_j\right)^T(B+\lambda I)^{-1}\left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}q_j\right).$$

Lembrando que $(B+\lambda I)^{-1} = Q(\Lambda+\lambda I)^{-1}Q^T$ (veja prova do Lema 1.1.4), temos que

$$\|t(\lambda)\|^2 = \left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}q_j\right)^T Q(\Lambda+\lambda I)^{-1}Q^T\left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}q_j\right).$$

Como Q é ortonormal, temos que

$$\|t(\lambda)\|^2 = \left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}e_j^T\right)(\Lambda+\lambda I)^{-1}\left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}e_j\right) =$$

$$\left(-\sum_{j=1}^{n}\frac{q_j^T g}{(\lambda_j+\lambda)^2}e_j^T\right)\left(-\sum_{j=1}^{n}\frac{q_j^T g}{\lambda_j+\lambda}e_j\right) = \sum_{j=1}^{n}\frac{(q_j^T g)^2}{(\lambda_j+\lambda)^3}.$$

Substituindo em (A.9), temos que

$$\phi^{'}(\lambda) = -\|p(\lambda)\|^{-3}\|t(\lambda)\|^2 = -\frac{\|t(\lambda)\|^2}{\|p(\lambda)\|^3}.$$

Finalmente, de posse destes resultados, analisemos $-\frac{\phi(\lambda)}{\phi^{'}(\lambda)}$.

$$-\frac{\phi(\lambda)}{\phi'(\lambda)} = -\frac{\left(\frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}\right)}{-\left(\frac{\|t(\lambda)\|^2}{\|p(\lambda)\|^3}\right)} = \left(\frac{1}{\Delta} - \frac{1}{\|p(\lambda)\|}\right)\left(\frac{\|p(\lambda)\|^3}{\|t(\lambda)\|^2}\right) =$$

$$\left(\frac{\|p(\lambda)\| - \Delta}{\Delta\|p(\lambda)\|}\right)\left(\frac{\|p(\lambda)\|^3}{\|t(\lambda)\|^2}\right) = \left(\frac{\|p(\lambda)\|}{\|t(\lambda)\|}\right)^2\left(\frac{\|p(\lambda)\| - \Delta}{\Delta}\right).$$

■

Lema 1.2.2 *Seja* $0 < \sigma < 1$ *dado e suponha que*

$$B + \lambda I = R^T R, \quad (B + \lambda I)p = -g, \quad \lambda \geq 0.$$

Seja $z \in I\!R^n$ *tal que*

$$\|p+z\| = \Delta, \quad \|Rz\|^2 \leq \sigma(\|Rp\|^2 + \lambda\Delta^2). \tag{1.6}$$

Então

$$-\varphi(p+z) \geq \frac{1}{2}(1-\sigma)(\|Rp\|^2 + \lambda\Delta^2) \geq (1-\sigma) \mid \varphi^* \mid,$$

onde φ^* *é o valor ótimo de (1.1).*

Prova Primeiramente calculemos $\varphi(p+z)$:

$$\varphi(p+z) = \frac{1}{2}(p+z)^T B(p+z) + g^T(p+z) =$$

$$\frac{1}{2}p^T Bp + z^T Bp + \frac{1}{2}z^T Bz + g^T p + g^T z.$$

Como, por hipótese, $g = -(B+\lambda I)p = -R^T Rp$, temos

$$\varphi(p+z) = \frac{1}{2}p^T Bp + z^T Bp + \frac{1}{2}z^T Bz + (-R^T Rp)^T p + (-R^T Rp)^T z.$$

Além disso, para quaisquer $x, y \in I\!R^n$ vale que $x^T By = x^T(B+\lambda I)y - \lambda x^T y = x^T(R^T R)y - \lambda x^T y$. Então,

$$\varphi(p+z) = \frac{1}{2}p^T(R^T R)p - \frac{\lambda}{2}p^T p + z^T(R^T R)p - \lambda z^T p + \frac{1}{2}z^T(R^T R)z - \frac{\lambda}{2}z^T z - p^T R^T Rp - p^T R^T Rz =$$

$$\frac{1}{2}(Rp)^T Rp - \frac{\lambda}{2}p^T p + (Rz)^T Rp - \lambda z^T p + \frac{1}{2}(Rz)^T Rz - \frac{\lambda}{2}z^T z - (Rp)^T Rp - (Rz)^T Rp =$$

$$\frac{1}{2}\|Rp\|^2 - \frac{\lambda}{2}\|p\|^2 + (Rz)^T Rp - \lambda z^T p + \frac{1}{2}\|Rz\|^2 - \frac{\lambda}{2}\|z\|^2 - \|Rp\|^2 - (Rz)^T Rp.$$

Como $\|p+z\|^2 = \|p\|^2 + 2p^T z + \|z\|^2$, concluímos que

$$\varphi(p+z) = -\frac{1}{2}(\|Rp\|^2 + \lambda\|p+z\|^2) + \frac{1}{2}\|Rz\|^2. \tag{A.10}$$

Para qualquer z que satisfaça (1.6), temos

$$-\varphi(p+z) = \frac{1}{2}(\|Rp\|^2 + \lambda\|p+z\|^2) - \frac{1}{2}\|Rz\|^2 \geq$$

$$\frac{1}{2}(\|Rp\|^2 + \lambda\|p+z\|^2) - \frac{1}{2}\sigma(\|Rp\|^2 + \lambda\Delta^2).$$

Como z satisfaz (1.6), temos que $\|p+z\| = \Delta$. Assim,

$$-\varphi(p+z) \geq \frac{1}{2}(\|Rp\|^2 + \lambda\Delta^2) - \frac{1}{2}\sigma(\|Rp\|^2 + \lambda\Delta^2) = \frac{1}{2}(1-\sigma)(\|Rp\|^2 + \lambda\Delta^2).$$

Ainda temos que, se $\varphi^* = \varphi(p+z^*)$, onde $\|p+z^*\| \leq \Delta$,

$$-\varphi(p+z^*) = \frac{1}{2}(\|Rp\|^2 + \lambda\|p+z^*\|^2) - \frac{1}{2}\|Rz^*\|^2 \leq$$

$$\frac{1}{2}(\|Rp\|^2 + \lambda\|p+z^*\|^2) \leq \frac{1}{2}(\|Rp\|^2 + \lambda\Delta^2).$$

Note que $p = 0$ sempre é solução de (1.1). Então, certamente $\varphi(p+z^*) \leq 0$, ou seja, $\varphi^* = |\ \varphi^*\ |$. Assim, temos as duas desigualdades que gostaríamos de provar. ■

Lema 1.2.3 *Seja $(B+\lambda I) = R^T R$ uma matriz definida positiva. Sejam $v \in \mathbb{R}^n$ o vetor calculado utilizando a técnica descrita em [7] e $\hat{z} = \frac{v}{\|v\|}$. Então $\|R\hat{z}\|$ se aproxima de 0 quando λ se aproxima de $-\lambda_1$, com λ_1 o menor autovalor de B.*

Prova Como mencionado na seção 1.2.2, a técnica descrita em [7] constrói um vetor e com componentes 1 ou -1 de forma que a solução w do sistema $R^T w = e$ seja grande. Então resolve-se o sistema $Rv = w$ para v e calculamos

$$\hat{z} = \frac{v}{\|v\|}.$$

Lembre-se que R é o fator de Cholesky de $B + \lambda I$ e, portanto, não é singular.

Para quaisquer vetores v e w tais que $R^T w = e$ e $Rv = w$ temos que

$$\|w\|^2 = \|R^{-T}e\|^2 = e^T R^{-1} R^{-T} e = e^T v \leq \|e\|\|v\|.$$

Como as componentes de e são todas iguais a 1 ou -1, $\|e\| = \sqrt{n}$. Assim,

$$\|w\|^2 \leq \sqrt{n}\|v\|.$$

Note que, como $\hat{z} = \frac{v}{\|v\|}$, temos

$$\|R\hat{z}\| = \frac{\|Rv\|}{\|v\|} = \frac{\|w\|}{\|v\|} \leq \frac{\sqrt{n}}{\|w\|}. \tag{A.11}$$

Vejamos agora como se comporta $\|w\|$. Em [7], quando vamos calcular a componente w_k, já temos calculados $w_1, ..., w_{k-1}$. Então definimos

$$p_i^{(k-1)} = \sum_{l=1}^{k-1} r_{li} w_l, \quad k \leq i \leq n,$$

onde r_{ij} é o elemento na posição (i, j) de R. Temos então duas escolhas para w_k, que são

$$w_k^+ = \frac{1 - p_k^{(k-1)}}{r_{kk}} \text{ e } w_k^- = \frac{-1 - p_k^{(k-1)}}{r_{kk}}.$$

Se

$$| \omega_k^+ | + \sum_{i=k+1}^{n} | p_i^{(k-1)} + r_{ki}\omega_k^+ | \ \geq \ | \omega_k^- | + \sum_{i=k+1}^{n} | p_i^{(k-1)} + r_{ki}\omega_k^- |,$$

então w_k é escolhido como w_k^+. Senão, $w_k = w_k^-$.

Veja que, se $p_k^{(k-1)} \leq 0$, temos que

$$| w_k^+ | \ \geq \ | w_k^- | \ \text{ e } \ | w_k^+ | \ \geq \frac{1}{| r_{kk} |}.$$

Se $p_k^{(k-1)} \geq 0$, temos que

$$| w_k^- | \ \geq \ | w_k^+ | \ \text{ e } \ | w_k^- | \ \geq \frac{1}{| r_{kk} |}.$$

Portanto,

$$\max\{| w_k^+ |, | w_k^- |\} \geq \frac{1}{| r_{kk} |},$$

e daí segue que

$$\frac{1}{| r_{kk} |} \leq \ | w_k | + \sum_{i=k+1}^{n} | p_i^{(k-1)} + r_{ki} w_k | \ \leq (1+r)\|w\| \Rightarrow$$

$$\frac{1}{\|w\|} \leq (1+r) | r_{kk} |, \quad 1 \leq k \leq n,$$

onde r é tal que

$$\sum_{j=1}^{n} \sum_{i=1}^{j} | r_{ij} | \leq r.$$

Voltando à inequação (A.11), temos que

$$\|R\hat{z}\| \leq \sqrt{n}(1+r) \min\{| r_{kk} |: 1 \leq k \leq n\}. \tag{A.12}$$

Ora, quando λ se aproxima de $-\lambda_1$, algum elemento da diagonal de R se aproxima de 0 e a equação (A.12) nos diz que $\|R\hat{z}\|$ também se aproxima de 0, como gostaríamos. ■

Lema 1.2.4 *Tome B, g e Δ do problema (1.1).*

Um limitante inferior λ_L para λ^* solução de $\phi(\lambda)$ é

$$\lambda_L = \frac{\|g\|}{\Delta} - \|B\|_1,$$

e um limitante superior λ_U é

$$\lambda_U = \frac{\|g\|}{\Delta} + \|B\|_1.$$

Prova Por (1.3) temos que

$$\|p(\lambda)\|^2 = \sum_{j=1}^{n} \frac{(q_j^T g)^2}{(\lambda_j + \lambda)^2}.$$

Como $\lambda_1 \leq \lambda_j \leq \lambda_n$, para todo j, temos que

$$\frac{1}{(\lambda_n + \lambda)^2} \sum_{j=1}^{n} (q_j^T g)^2 \leq \|p(\lambda)\|^2 \leq \frac{1}{(\lambda_1 + \lambda)^2} \sum_{j=1}^{n} (q_j^T g)^2.$$

Note agora que

$$\sum_{j=1}^{n} (q_j^T g)^2 = \sum_{j=1}^{n} [Q^T g]_j^2 = \|Q^T g\|^2.$$

Como Q é ortonormal,

$$\|Q^T g\|^2 = \|g\|^2.$$

Assim, temos que

$$\frac{\|g\|^2}{(\lambda_n + \lambda)^2} \leq \|p(\lambda)\|^2 \leq \frac{\|g\|^2}{(\lambda_1 + \lambda)^2}.$$

Como $\lambda^* > -\lambda_1$ temos que $\lambda^* + \lambda_j > 0$ para todo j. Daí segue que

$$\frac{\|g\|}{\lambda_n + \lambda^*} \leq \|p(\lambda^*)\| \leq \frac{\|g\|}{\lambda_1 + \lambda^*}. \tag{A.13}$$

Para continuar, vejamos algumas propriedades de $\|B\|_1$ (veja, por exemplo, [12]).

Por definição,

$$\|B\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^{n} \mid b_{ij} \mid$$

e

$$\|B\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^{n} \mid b_{ij} \mid .$$

Existe uma propriedade que diz que $\|B\|_2 \leq \sqrt{\|B\|_1 \|B\|_\infty}$.

Como B é simétrica, temos que

$$\|B\|_1 = \|B\|_\infty \Rightarrow \|B\|_2 \leq \sqrt{\|B\|_1^2} \Rightarrow \|B\|_2 \leq \|B\|_1.$$

Uma outra propriedade de normas de matrizes é $\|B\|_2 = \max\{\mid \lambda_1 \mid, \mid \lambda_n \mid\}$, quando B é simétrica.

Daí temos que

$$\|B\|_2 = \max\{\mid \lambda_1 \mid, \mid \lambda_n \mid\} \leq \|B\|_1 \Rightarrow$$

$$\mid \lambda_1 \mid \leq \|B\|_1 \Rightarrow -\|B\|_1 \leq \lambda_1 \leq \|B\|_1$$

e

$$\mid \lambda_n \mid \leq \|B\|_1 \Rightarrow -\|B\|_1 \leq \lambda_n \leq \|B\|_1.$$

Voltando à inequação à esquerda de (A.13), temos que

$$\frac{\|g\|}{\lambda_n + \lambda^*} \leq \|p(\lambda^*)\| \Rightarrow \frac{\|g\|}{\|p(\lambda^*)\|} \leq \lambda_n + \lambda^* \Rightarrow \frac{\|g\|}{\|p(\lambda^*)\|} - \lambda_n \leq \lambda^*.$$

Como $\lambda_n \leq \|B\|_1$ temos

$$\lambda^* \geq \frac{\|g\|}{\|p(\lambda^*)\|} - \lambda_n \geq \frac{\|g\|}{\|p(\lambda^*)\|} - \|B\|_1.$$

Como $\|p(\lambda^*)\| = \Delta$, temos

$$\lambda^* \geq \frac{\|g\|}{\Delta} - \|B\|_1 = \lambda_L.$$

Agora, olhando a inequação à direita de (A.13) e observando que $\lambda_1 \geq -\|B\|_1$, temos

$$\lambda^* \leq \frac{\|g\|}{\|p(\lambda^*)\|} - \lambda_1 \leq \frac{\|g\|}{\Delta} + \|B\|_1 = \lambda_U.$$

■

Lema 1.2.5 *Seja $B + \lambda I$ uma matriz definida positiva tal que $B + \lambda I = R^T R$. Para qualquer $\hat{z}$ tal que $\|\hat{z}\| = 1$ vale que*

$$\lambda - \|R\hat{z}\|^2 \leq -\lambda_1.$$

Prova Observe que

$$\|R\hat{z}\|^2 = (R\hat{z})^T(R\hat{z}) = \hat{z}^T R^T R\hat{z} = \hat{z}^T(B + \lambda I)\hat{z} =$$

$$\hat{z}^T B\hat{z} + \hat{z}^T(\lambda I)\hat{z} = \hat{z}^T B\hat{z} + \lambda\hat{z}^T\hat{z} = \hat{z}^T B\hat{z} + \lambda\|\hat{z}\|^2.$$

Como $\|\hat{z}\| = 1$, temos que

$$\|R\hat{z}\|^2 = \hat{z}^T B\hat{z} + \lambda \Rightarrow \lambda - \|R\hat{z}\|^2 = -\hat{z}^T B\hat{z}.$$

Tome a decomposição $B = Q\Lambda Q^T$, com $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_n)$, $\lambda_1 \leq \lambda_2 \leq ... \leq \lambda_n$ e $Q^T Q = I$. Temos então que

$$\lambda - \|R\hat{z}\|^2 = -\hat{z}^T Q\Lambda Q^T\hat{z} = -\hat{z}^T Q \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} Q^T\hat{z}.$$

Como Q é ortogonal, temos

$$\lambda - \|R\hat{z}\|^2 = -\lambda_1\hat{z}_1^2 - \lambda_2\hat{z}_2^2 - ... - \lambda_n\hat{z}_n^2,$$

onde $\hat{z}_i$ é a i-ésima componente de $\hat{z}$.

Como $\lambda_1 \leq \lambda_j$, para todo j, temos

$$\lambda - \|R\hat{z}\|^2 \leq -\lambda_1(\hat{z}_1^2 + \hat{z}_2^2 + ... + \hat{z}_n^2) = -\lambda_1\|\hat{z}\|.$$

Mas $\|\hat{z}\| = 1$. Portanto,

$$\lambda - \|R\hat{z}\|^2 \leq -\lambda_1.$$

■

Lema 1.2.6 *Seja $\delta \geq 0$ tal que $B + \lambda I + \delta e_l e_l^T$ é singular, com e_l a l-ésima coluna da matriz identidade. Seja $u \in I\!R^l$ tal que $(B^l + \lambda I + \delta e_l e_l^T)u = 0$, com $u_l = 1$.*

Então

$$\lambda + \frac{\delta}{\|u\|^2} \leq -\lambda_1.$$

Prova Vamos calcular o quociente de Rayleigh $r(u)$ de B^l (veja, por exemplo, [12]):

$$r(u) = \frac{u^T B^l u}{u^T u}.$$

Como

$$(B^l + \lambda I + \delta e_l e_l^T)u = 0 \Rightarrow B^l u = -\lambda u - \delta e_l e_l^T u.$$

Assim,

$$r(u) = \frac{u^T(-\lambda u - \delta e_l e_l^T u)}{u^T u} = \frac{-\lambda u^T u - \delta u^T e_l e_l^T u}{u^T u} = -\lambda - \delta \frac{(e_l^T u)^T (e_l^T u)}{u^T u} \Rightarrow$$

$$r(u) = -\lambda - \delta \frac{\|e_l^T u\|^2}{\|u\|^2}.$$

Mas, como $u_l = 1$, temos $\|e_l^T u\|^2 = 1$. Ou seja,

$$r(u) = -\lambda - \frac{\delta}{\|u\|^2}.$$

Como $r(u) \geq \bar{\lambda_1}$, onde $\bar{\lambda_1}$ é o menor autovalor de B^l, para todo u, temos que

$$-\lambda - \frac{\delta}{\|u\|^2} \geq \bar{\lambda_1} \Rightarrow \lambda + \frac{\delta}{\|u\|^2} \leq -\bar{\lambda_1}.$$

Note que

$$\lambda_1 = \min_{x \in I\!R^n} \frac{x^T B x}{x^T x} \leq \min_{\substack{x \in I\!R^n \\ x_i = 0, i \geq l+1}} \frac{x^T B x}{x^T x} = \min_{y \in I\!R^l} \frac{y^T B^l y}{y^T y} = \bar{\lambda_1}.$$

Logo,

$$\lambda + \frac{\delta}{\|u\|^2} \leq -\bar{\lambda_1} \leq \lambda_1,$$

como gostaríamos. ■

Lema 1.2.7 *Seja* $0 < \sigma < 1$ *dado e suponha que*

$$B + \lambda I = R^T R, \quad (B + \lambda I)p = -g, \quad \lambda \geq 0.$$

Se $\|p\| \geq (1 - \sigma)\Delta$ *então*

$$-\varphi(p) \geq \frac{1}{2}(1 - \sigma)^2(\|Rp\|^2 + \lambda\Delta^2) \geq (1 - \sigma)^2 \mid \varphi^* \mid,$$

onde φ^* *é o valor ótimo de (1.1).*

Prova Como a equação (A.10) vale para qualquer z, tome $z = 0$

$$\varphi(p) = -\frac{1}{2}(\|Rp\|^2 + \lambda\|p\|^2) \Rightarrow -\varphi(p) = \frac{1}{2}(\|Rp\|^2 + \lambda\|p\|^2) \geq$$

$$\frac{1}{2}((1 - \sigma)^2\|Rp\|^2 + \lambda\|p\|^2).$$

Por hipótese, temos que $\|p\| \geq (1 - \sigma)\Delta$. Assim, temos que

$$-\varphi(p) \geq \frac{1}{2}((1 - \sigma)^2\|Rp\|^2 + \lambda((1 - \sigma)\Delta)^2) = \frac{1}{2}(1 - \sigma)^2(\|Rp\|^2 + \lambda\Delta^2).$$

Pelo mesmo raciocínio feito na prova do Lema 1.2.2, temos que

$$\mid \varphi^* \mid \leq \frac{1}{2}(\|Rp\|^2 + \lambda\Delta^2) \Rightarrow (1 - \sigma)^2 \mid \varphi^* \mid \leq \frac{1}{2}(1 - \sigma)^2(\|Rp\|^2 + \lambda\Delta^2).$$

Daí temos as duas desigualdades nas quais estamos interessados. ■

Referências Bibliográficas

[1] E. G. Birgin e J. M. Martínez. A box-constrained optimization algorithm with negative curvature directions and spectral projected gradients. *Computing [Suppl]* 15, pp. 49–60, 2001.

[2] E. G. Birgin e J. M. Martínez. Large-scale active-set box-constrained optimization method with spectral projected gradients. *Computational Optimization and Applications* 23, pp. 101–125, 2002.

[3] E. G. Birgin, J. M. Martínez e M. Raydan. Inexact spectral projected gradient methods on convex sets. *IMA Journal of Numerical Analysis* 23, pp. 539–559, 2003.

[4] E. G. Birgin, J. M. Martínez e M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization* 10, pp. 1196–1211, 2000.

[5] E. G. Birgin, J. M. Martínez e M. Raydan. Algorithm 813: SPG - software for convex-constrained optimization. *ACM Transactions on Mathematical Software* 27, pp. 340–349, 2001.

[6] I. Bongartz, A. R. Conn, N. I. M. Gould e Ph. L. Toint. CUTE: constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software* 21, pp. 123–160, 1995.

[7] A. K. Cline, C. B. Moler, G. W. Stewart e J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis* 16, pp. 368–375, 1979.

[8] A. R. Conn, N. I. M. Gould e Ph. L. Toint. A globally convergent Augmented Lagrangian algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Numerical Analysis* 28, pp. 545–572, 1991.

[9] E. D. Dolan e J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming* 91, pp. 201–213, 2002.

[10] R. Fletcher. *Practical Methods of Optimization*, Wiley, 1997.

[11] D. M. Gay. Computing optimal locally constrained steps. *SIAM Journal on Scientific and Statistical Computing* 2, pp. 186–197, 1981.

[12] G. H. Golub e C. F. van Loan. *Matrix Computations*, Johns Hopkins, 1996.

[13] J. M. Martínez e S. A. Santos. A trust-region strategy for minimization on arbitrary domains. *Mathematical Programming* 68, pp. 267–302, 1995.

[14] J. M. Martínez e S. A. Santos. Convergence results on an algorithm for norm constrained regularization and related problems. *RAIRO Operations Research* 31, pp. 269–294, 1997.

[15] J. M. Martínez e S. A. Santos. *Métodos Computacionais de Otimização*, XX Colóquio Brasileiro de Matemática, IMPA, 1995.

[16] J. J. Moré. Recent developments in algorithms and software for trust-region methods. Em: A. Bachem, M. Grötschel e B. Korte, eds., *Mathematical Programming Bonn 1982 - The State of Art*, Springer-Verlag, 1983.

[17] J. J. Moré, B. S. Garbow e K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software* 7, pp. 17–41, 1981.

[18] J. J. Moré e D. C. Sorensen. Computing a trust region step. *SIAM Journal on Scientific and Statistical Computing* 4, pp. 553–572, 1983.

[19] J. Nocedal e S. J. Wright. *Numerical Optimization*, Springer, 1999.

[20] M. Rojas, S. A. Santos e D. C. Sorensen. A new matrix-free algorithm for the large-scale trust-region subproblem. *SIAM Journal on Optimization* 11, pp. 611–646, 2000.

[21] L. Schrage. A more portable Fortran random number generator. *ACM Transactions on Mathematical Software* 5, pp. 132–138, 1979.

[22] D. C. Sorensen. Newton's method with a model trust-region modification. *SIAM Journal on Numerical Analisis* 19, pp. 409–426, 1982.

[23] S. W. Thomas. Sequential estimation techniques for quasi-Newton algorithms, Tese de Ph. D., Cornell University, Ithaca, NY, 1975.