Sistemas não lineares via região de confiança: o algoritmo de Levenberg-Marquardt*

John Lenon C. Gardenghi
Profa. Dra. Sandra Augusta Santos
Departamento de Matemática Aplicada - IMECC - UNICAMP
27 de fevereiro de 2011
Revisado em: 14 de janeiro de 2012

Resumo

Este trabalho consiste no estudo e na implementação computacional da estratégia de Levenberg-Marquardt abordada por Moré em [5] para a resolução de sistemas não lineares como um problema de otimização irrestrita. Tal estratégia é globalmente convergente e pode ser implementada de maneira robusta e eficiente. Nosso objetivo na preparação deste texto foi enriquecer a análise encaminhada por Moré, incluindo com detalhes as principais ideias da técnica de região de confiança. Além disso, procuramos organizar de forma concisa os conceitos necessários para a compreensão dos detalhes referentes à implementação numérica e computacional. Os experimentos efetuados validaram a implementação desenvolvida, que foi totalmente programada no CAS Maxima, sem recorrer a nenhuma função ou rotina pré-existente envolvendo a álgebra linear computacional do algoritmo.

Palavras-chave sistemas não lineares, Levenberg-Marquardt, região de confiança, quadrados mínimos, CAS Maxima.

Abstract

This work consists in the study and the computational implementation of the Levenberg-Marquardt algorithm, as proposed by Moré in [5], for the solution of nonlinear systems by means of an unconstrained optimization problem. Such a method is globally convergent and can be implemented in an efficient and robust way. Our goal in the preparation of this text was to follow the analysis presented by Moré, including further details and the main ideas of trust-region methods. Moreover, we aimed to organize the necessary concepts to follow the details concerning the numerical and computational implementation in a concise but thorough way. The numerical experiments validated the implementation and were totally coded using the CAS Maxima, not only the main program, but also the whole set of functions and routines involving the computational linear algebra of the algorithm.

Key words nonlinear systems, Levenberg-Marquardt, trust region, least squares, CAS Maxima.

*Este trabalho contou com apoio do CNPq (Processos 104680/2009-1, 303465/2007-7 e 304032/2010-7), FAPESP (Processo 2006/53768-0) e PRONEX-Otimização.

1 Introdução

O algoritmo proposto por Levenberg [3] e Marquardt [4] possui várias aplicações no campo da otimização. Originalmente, é uma proposta para a resolução de problemas de quadrados mínimos. Aliado às suas propriedades de convergência, o algoritmo de Levenberg-Marquardt oferece a possibilidade de uma implementação computacional robusta e eficiente, como sugerido por Moré em [5]. Neste trabalho, desenvolveremos uma breve discussão sobre métodos de região de confiança, e vamos implementar o algoritmo de Levenberg-Marquardt como um método de região de confiança, aplicando os resultados à resolução de problemas clássicos da literatura: solução de sistemas não lineares como problemas de otimização irrestritos, visando explorar a convergência global desse método.

2 O problema

Seja $F : \mathbb{R}^n \to \mathbb{R}^m$, o problema de quadrados mínimos é definido como:

$$\begin{aligned} \min \ \Psi(x) &= \frac{1}{2}F(x)^T F(x) = \frac{1}{2}\sum_{i=1}^{m} f_i(x)^2 \\ \text{s.a } x &\in \mathbb{R}^n, \end{aligned} \tag{1}$$

onde F é dita *função residual*, em geral não linear em x, e o vetor $F(x)$ é dito *resíduo* do problema em x. Assumiremos que F é uma função suave, ao menos com derivadas segundas contínuas.

O problema de quadrados mínimos está intimamente ligado a problemas clássicos de otimização. Por exemplo, se $m = n$, então estamos em um caso especial de resolução de sistemas não lineares. Para qualquer valor de m, estamos em um caso especial de minimização irrestrita.

Vamos estabelecer as derivadas da função residual, o que será muito útil para análise do problema adiante. A primeira derivada de $F(x)$ será expressa na matriz Jacobiana $J(x) \in \mathbb{R}^{m\times n}$ tal que

$$J(x)_{ij} = \frac{\partial f_i}{\partial x_j}(x).$$

Assim, pela regra da cadeia, a primeira derivada para a função objetivo $\Psi(x)$ em (1) será:

$$\nabla\Psi(x) = \sum_{i=1}^{m} f_i(x)\nabla f_i(x) = J(x)^T F(x). \tag{2}$$

Da mesma maneira, podemos pensar em sua derivada segunda, computando a matriz Hessiana:

$$\begin{aligned} \nabla^2\Psi(x) &= \sum_{i=1}^{m}(\nabla f_i(x)\nabla f_i(x)^T + f_i(x)\nabla^2 f_i(x)) \\ &= J(x)^T J(x) + S(x), \end{aligned} \tag{3}$$

onde $S(x) = \sum_{i=1}^{m} f_i(x)\nabla^2 f_i(x)$ contém as informações de segunda ordem da função objetivo.

Se pensarmos no método de Newton para resolver o problema (1), a cada iteração, teríamos um modelo quadrático baseado nas informações das derivadas definidas em (2) e (3) e o iterando de Newton aplicado a esse modelo em um ponto corrente x_c seria:

$$\begin{aligned} x_n &= x_c - \nabla^2\Psi(x_c)^{-1}\nabla\Psi(x_c) \\ &= x_c - (J(x_c)^T J(x_c) + S(x_c))^{-1} J(x_c)^T F(x_c). \end{aligned} \tag{4}$$

Obter as informações de segunda ordem para a função objetivo $\Psi(x)$ descritas na expressão $S(x)$ em (3) é muito caro, por isso o método de Newton se torna muito custoso se usado em problemas de grande porte, especialmente se $m \gg n$. É neste ponto que está a inspiração para o método de Levenberg-Marquardt: uma maneira barata e confiável para estimar essas informações de segunda ordem.

Por isso, nesta abordagem, vamos tomar o problema de quadrados mínimos para uma classe especial de sistemas não lineares: aqueles em que $m > n$, isto é, em que temos mais dados que variáveis no problema, ambiente típico do caso de quadrados mínimos em que deseja-se ajustar os parâmetros de um dado modelo a um conjunto de dados coletados. Portanto, trabalharemos com *sistemas de equações não lineares sobredeterminados*. Vamos começar apresentando ideias dos métodos de região de confiança, que nos permitirão avaliar como o método de Levenberg-Marquardt consiste em uma estratégia eficiente e barata para a resolução do problema em questão.

3 Métodos de região de confiança

O método de região de confiança é uma estratégia iterativa para solução de problemas de otimização com o objetivo de obter convergência global e aproveitar, ao máximo, as propriedades de convergência local dos métodos de Newton e Quase-Newton.

Para o caso de minimização irrestrita, a estratégia consiste na definição de uma região de raio Δ_c em torno do ponto corrente x_c e então encontrar o minimizador (aproximado) p^* de um modelo quadrático da função objetivo em tal região. O minimizador encontrado define então um passo para um novo ponto $x_n = x_c + p^*$ que objetiva o decrescimento da função a partir de x_c. Se isso realmente acontecer, então o modelo representou bem a função objetivo na região definida. Caso não haja o decrescimento da função objetivo no novo ponto x_n, isso significa que o modelo não representou bem a função na região dada; esta região é então reduzida para a próxima iteração. A região é definida de acordo com a boa representação do modelo com relação à função objetivo: seu tamanho varia de acordo com o descréscimo da função objetivo nas iterações anteriores.

Por isso, os métodos de região de confiança são comparados aos métodos de busca linear, visto que ambos visam gerar um passo que minimize dada função objetivo a partir de um modelo quadrático em torno de um ponto corrente. A principal diferença é que a estratégia de busca linear usa o modelo para encontrar a direção e, a partir desta direção, concentra seus esforços em encontrar um tamanho ótimo para o passo, enquanto a estratégia de região de confiança estabelece a direção e o tamanho do passo simultaneamente a partir da minimização do modelo quadrático sujeito à região de raio Δ_c; de fato, a direção e o passo mudam sempre que o tamanho da região é alterada. Para ilustrar o caso, ver a Figura 1, adaptada de [8, p. 66].

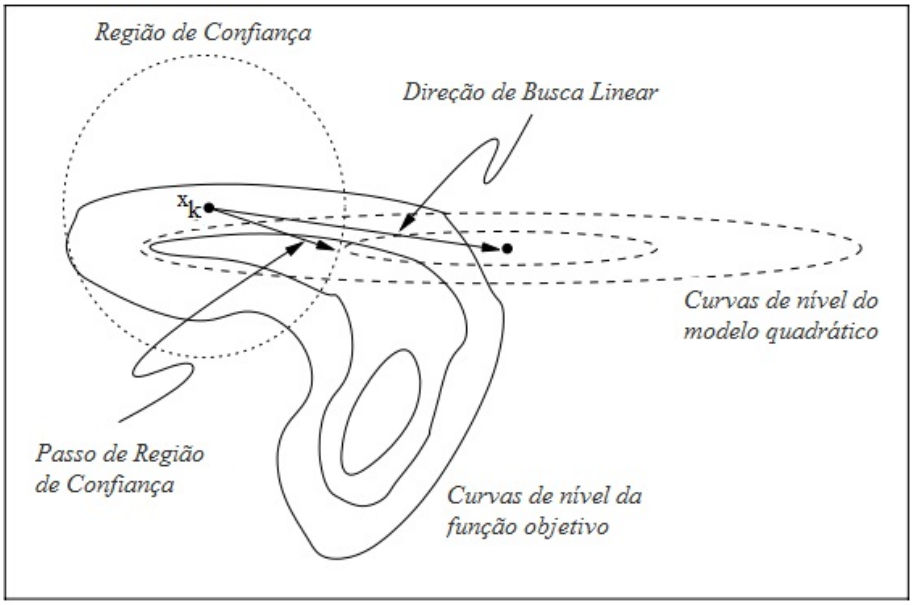


Figura 1: Métodos de busca linear e de região de confiança aplicados à mesma função.

3.1 Os subproblemas de região de confiança

Seja $f : \mathbb{R}^n \to \mathbb{R}$, $f \in C^2$. Dados o problema de minimização irrestrita

$$\begin{aligned} &\min\ f(x) \\ &\text{s.a } x \in \mathbb{R}^n, \end{aligned} \tag{5}$$

e um ponto corrente x_c, a cada iteração, define-se o modelo quadrático $q_c(p)$ em torno do ponto e resolve-se o subproblema:

$$\begin{aligned} \min\ q_c(p) \quad &= \quad f(x_c) + g_c^T p + \frac{1}{2} p^T H_c p \\ \text{s.a } \|p\| &\leq \Delta_c, \end{aligned} \tag{6}$$

onde Δ_c é o raio da região de confiança, $g_c \equiv \nabla f(x_c) \in \mathbb{R}^n$ é o vetor gradiente, $H_c \equiv \nabla^2 f(x_c) \in \mathbb{R}^{n \times n}$ é a matrix hessiana da função objetivo avaliada no ponto x_c e $\|.\|$ é a norma euclidiana. A solução global p^* do problema (6) é o minimizador de $q_c(p)$ na bola de raio Δ_c, e esta solução será o passo para atualizar $x_n = x_c + p^*$.

3.2 Caracterização da solução para o subproblema

No teorema abaixo, vamos caracterizar a solução do problema (6), com resultados adaptados de [1] e [8]. Essa caracterização é consequência da estrutura especial desse problema.

Teorema 1. *Seja $f(x)$ como definida em (5). Então, $p^* \in \mathbb{R}^n$ será solução global do problema (6) se e somente se p^* for viável e existir um escalar $\lambda^* \geq 0$ tal que:*

$$(H_c + \lambda^* I)p^* = -g_c, \tag{7a}$$

$$\lambda^*(\|p^*\| - \Delta_c) = 0, \tag{7b}$$

$$H_c + \lambda^* I \quad \textit{é semidefinida positiva.} \tag{7c}$$

Prova. Por uma pequena mudança de notação, seja $q(p) = q_c(p) - f(x_c)$. Como $f(x_c)$, para o problema (6), é uma constante, não há perda de generalidade.

Para começar, assumamos que existe $\lambda^* \geq 0$ tal que as condições (7) são satisfeitas. Nosso objetivo é mostrar que p^* é solução global de (6). A função:

$$\hat{q}(p) = g_c^T p + \frac{1}{2} p^T (H_c + \lambda^* I) p = q(p) + \frac{\lambda^*}{2} p^T p \tag{8}$$

é convexa, pois $\nabla^2 \hat{q}(p) = H_c + \lambda^* I$ é semidefinida positiva por (7c). Então, p^* é minimizador global de (8), visto que é ponto estacionário de $\hat{q}(p)$ por (7a). Com isso:

$$\hat{q}(p) \geq \hat{q}(p^*), \forall p \in \mathbb{R}^n. \tag{9}$$

Então, por (8) em (9):

$$\begin{aligned} q(p) + \frac{\lambda^*}{2} p^T p &\geq q(p^*) + \frac{\lambda^*}{2} (p^*)^T p^* \\ q(p) &\geq q(p^*) + \frac{\lambda^*}{2} ((p^*)^T p^* - p^T p). \end{aligned} \tag{10}$$

Por (7b), $\lambda^*(\|p^*\| - \Delta_c) = 0$, portanto $\lambda^*((p^*)^T p^* - \Delta_c^2) = 0$, ou seja, $\lambda^*(p^*)^T p = \lambda^* \Delta_c^2$. Daí vem que:

$$q(p) \geq q(p^*) + \frac{\lambda^*}{2} (\Delta_c^2 - p^T p). \tag{11}$$

Portanto, em (11), para todo p tal que $\|p\| \leq \Delta_c$, temos que:

$$q(p) \geq q(p^*),$$

mostrando que p^* é minimizador global para (6).

Agora, suponhamos que p^* é solução global para o problema (6). O objetivo agora é provar que existe $\lambda^* \geq 0$ tal que as condições (7) são satisfeitas. Para isso, vamos analisar dois casos:

(1) Se $\|p\| < \Delta_c$, então p^* é mínimizador irrestrito de $q(p)$. Logo, pelas condições necessárias para minimização sem restrições:

$$\begin{cases} \nabla q(p^*) = 0 & \rightarrow \quad H_c p^* + g_c = 0, \\ \nabla^2 q(p^*) = H_c \geq 0 & \rightarrow \quad \text{semidefinida positiva.} \end{cases}$$

Com isso, as condições (7) são satisfeitas para $\lambda^* = 0$.

(2) Se $\|p\| = \Delta_c$, então p^* é solução do problema de minimização com restrição de igualdade:

$$\begin{aligned} &\min \ q(p) \\ &\text{s.a } \|p\| = \Delta_c. \end{aligned} \tag{12}$$

Seja $r(p) = \frac{1}{2}(\|p\|^2 - \Delta_c^2) = 0$ uma expressão para a restrição do problema (12), que satisfaz (7b). Assim, pelas condições de otimalidade do problema, a Lagrangiana definida como

$$\mathscr{L}(p, \lambda) = q(p) + \lambda\, r(p) = q(p) + \frac{\lambda}{2}(p^T p - \Delta_c^2) \tag{13}$$

possui ponto estacionário em p^*, portanto:

$$\begin{aligned} \nabla_p \mathscr{L}(p^*, \lambda^*) = \quad & \nabla q(p^*) + \lambda^* \nabla r(p^*) = 0, \\ & H_c p^* + g_c + \lambda^* p^* = 0, \\ & (H_c + \lambda^* I) p^* = -g_c, \end{aligned} \tag{14}$$

o que nos dá (7a).

Como p^* é solução global para (12), então $q(\bar{p}) \geq q(p^*)$ para todo $\bar{p}$ tal que $\bar{p}^T\bar{p} = (p^*)^T p^* = \Delta_c^2$, podemos escrever:

$$q(\bar{p}) \geq q(p^*) + \frac{\lambda^*}{2}((p^*)^T p^* - \bar{p}^T\bar{p}). \tag{15}$$

Sabendo que $(H_c + \lambda^* I)p^* = -g_c$, podemos desenvolver a expressão (15), o que nos dá:

$$\frac{1}{2}(\bar{p} - p^*)^T (H_c + \lambda^* I)(\bar{p} - p^*) \geq 0. \tag{16}$$

Vamos analisar a expressão (16). Pelas condições necessárias de segunda ordem para o problema (12) temos que a matriz hessiana da Lagrangiana definida como:

$$\nabla_p^2 \mathscr{L}(p^*, \lambda^*) = \nabla^2 q(p^*) + \lambda^* \nabla^2 r(p^*) = H_c + \lambda^* I \tag{17}$$

será semidefinida positiva no núcleo de $\nabla r(p^*) = p^*$, ou seja, $z^T(H_c + \lambda^* I)z \geq 0$ para todo $z \in \mathbb{R}^n$ tal que $z^T p^* = 0$. Sabendo disto, para mostrar (7c), precisamos provar que $v^T(H_c + \lambda^* I)v \geq 0$ para todo $v \in \mathbb{R}^n$ tal que $v^T p^* \neq 0$. Então, consideremos a reta $p^* + \xi v$, onde $\xi \in \mathbb{R}$. Como $v^T p^* \neq 0$, essa reta irá interceptar a restrição do problema $\|p\| = \Delta_c$ para dois valores de ξ, a saber, $\xi = 0$, para o qual $p = p^*$ e para $\xi = \bar{\xi} \neq 0$, onde $p = \bar{p}$. À vista disso, $\bar{p} - p^* = \bar{\xi} v$, o que, em (16) nos dá:

$$\frac{1}{2}(\bar{\xi})^2 v^T (H_c + \lambda^* I)v \geq 0, \tag{18}$$

o que basta para provar (7c). ■

O Teorema 1 estabelece uma característica muito interessante do subproblema de regiões de confiança: as condições de otimalidade para este problema de minimização com uma única restrição de desigualdade são necessárias e suficientes, e o minimizador do problema é global. Isso quer dizer que basta que as condições necessárias sejam satisfeitas para que cheguemos a uma solução global do problema, o que não acontece em problemas convencionais de minimização.

A solução p^* do problema (6) enquadra-se em duas possíveis instâncias, definidas pelo comportamento da função objetivo do problema (5) e de suas derivadas e pelo tamanho do raio:

1. Solução interna, isto é, $\|p^*\| < \Delta_c$;

2. Solução na fronteira, isto é, $\|p^*\| = \Delta_c$.

No corolário a seguir, inspirado no desenvolvimento de [7, p.13], vamos mostrar em quais circunstâncias temos uma solução interna e, na sequência, discutir a melhor maneira de encontrar a solução quando esta se situa na fronteira.

Corolário 1. *O problema (6) não possui solução global p^* na fronteira se, e somente se, a matriz H_c é positiva definida e $\|H_c^{-1}g_c\| < \Delta_c$.*

Prova. Se H_c é positiva definida e $\|H_c^{-1}g_c\| < \Delta_c$, pelo Teorema 1, segue que $\lambda^* = 0$ e a direção de Newton $p^* = -H_c^{-1}g_c$ é a solução global de (6), interior à bola de confiança.

Agora, suponhamos que não existe p^*, solução global para (6), tal que $\|p^*\| = \Delta_c$, então por (7b) segue que $\lambda^* = 0$ e por (7a) temos que $H_c p^* = -g_c$ com H_c semidefinida positiva por (7c). Supondo H_c singular, então $H_c z = 0$ para algum $z \in \mathbb{R}^n$, $z \neq 0$. Da mesma maneira, para $\alpha \in \mathbb{R}, \alpha \neq 0$, temos

$$H_c(\alpha z) = 0. \tag{19}$$

Logo, $H_c(p^* + \alpha z) = -g_c$ e então $\alpha z^T H_c(p^* + \alpha z) = -\alpha z^T g_c = 0$, portanto:

$$\alpha g_c^T z = 0. \tag{20}$$

Tomemos agora:

$$\begin{aligned} q(p^* + \alpha z) &= f_c + g_c^T(p^* + \alpha z) + \frac{1}{2}(p^* + \alpha z)^T H_c(p^* + \alpha z) \\ &= f_c + g_c^T p^* + \alpha g_c^T z + \frac{1}{2}(p^*)^T H_c p^* + (p^*)H_c(\alpha z) + \alpha^2 z^T H_c z \\ &\overset{(19)-(20)}{=} q(p^*). \end{aligned} \tag{21}$$

Com isso, conclui-se que $p^* + \alpha z$ também será solução de (6). Se tormarmos α tal que $\|p^* + \alpha z\| = \Delta_c$, então temos uma solução na fronteira, o que contradiz a hipótese. Portanto, H_c deve ser positiva definida e $p^* = -H_c^{-1}g_c$ com $\|p^*\| < \Delta_c$. ■

Pelo Corolário 1 vemos que a solução no interior da região de confiança ocorre apenas na situação particular em que o modelo quadrático é estritamente convexo e o raio da região de confiança é suficientemente grande.

Portanto, excluindo-se o caso particular explicitado no Corolário 1, nosso problema se resume em encontrar um parâmetro $\lambda^* \geq 0$, $\lambda^* \in (-\mu_1, +\infty)$, onde μ_1 é o menor autovalor da matriz H_c, tal que, dada a função:

$$\delta(\lambda) \overset{\text{def}}{=} \|p(\lambda)\| = \|(H_c + \lambda I)^{-1}g_c\|, \tag{22}$$

este parâmetro satisfaça a equação:

$$\delta(\lambda^*) = \Delta_c. \tag{23}$$

Em decorrência do Corolário 1, a equação (23) terá a solução desejada sempre que:

- H_c for positiva definida mas $\|H_c^{-1}g_c\| \geq \Delta_c$, ou
- H_c não for positiva definida.

Vamos então analisar a melhor maneira de resolver o problema (23). Como $f \in C^2$, então a matriz $H_c \in \mathbb{R}^{n\times n}$ é simétrica, e portanto possui uma base ortonormal de autovetores e n autovalores associados, de modo que sua decomposição espectral é dada por:

$$H_c = VDV^T,$$

onde $D \in \mathbb{R}^{n\times n}$ e uma matriz diagonal composta por autovalores

$$\mu_1 \leq \mu_2 \leq \cdots \leq \mu_n$$

da matriz H_c e $V \in \mathbb{R}^{n\times n}$ é uma matriz ortonormal cujas colunas são os autovetores

$$v_1, v_2, \ldots, v_n$$

associados aos autovalores da matriz H_c. Então, podemos escrever:

$$\delta(\lambda)^2 = \|V(D+\lambda I)^{-1}V^T g_c\|^2 = \sum_{j=1}^{n}\left(\frac{v_j^T g_c}{\mu_j+\lambda}\right)^2. \tag{24}$$

A função $\delta(\lambda)$ é contínua no intervalo $(-\mu_1, +\infty)$, mas pode não estar bem definida para $\lambda = \mu_k$, $k = 1, \ldots, n$, pois, para cada k, se $v_k^T g_c \neq 0$, temos:

$$\lim_{\lambda\to-\mu_k}\delta(\lambda) = +\infty. \tag{25}$$

O limite expresso em (25) mostra que a função $\delta(\lambda)$ pode possuir polos nos autovalores da matriz H_c.

Tomemos agora a função $\delta(\lambda)$ no intervalo de interesse $(-\mu_1, +\infty)$, no qual é contínua e diferenciável. De fato, sua derivada é dada por:

$$\delta'(\lambda) = \frac{p'(\lambda)^T p(\lambda)}{\|p(\lambda)\|} = -\frac{p(\lambda)^T(H_c+\lambda I)^{-1}p(\lambda)}{\|p(\lambda)\|}. \tag{26}$$

Observe que $\delta'(\lambda) < 0$ sempre que $g_c \neq 0$. Com isso, se $\lambda_a > \lambda_b$, então $\|p(\lambda_a)\| < \|p(\lambda_b)\|$. Logo, como

$$\lim_{\lambda\to+\infty}\delta(\lambda) = 0,$$

a função $\delta(\lambda)$ é monótona e descrescente no intervalo $(-\mu_1, +\infty)$ (veja Figuras 2, 5, 8, 11 e 14) e por isso o problema (23) possui solução única neste intervalo, expressa em (7a).

Além disso, para qualquer $\lambda \geq 0$ em $(-\mu_1, +\infty)$, a direção $p(\lambda)$ é de descida para o problema (5) a partir do ponto corrente x_c, pois nesse caso $H_c + \lambda I$ é positiva definida e então

$$\nabla f(x_c)^T p(\lambda) = -g_c(H_c+\lambda I)^{-1}g_c < 0.$$

Vamos agora definir a função:

$$\phi(\lambda) \stackrel{\text{def}}{=} \frac{1}{\|p(\lambda)\|}. \tag{27}$$

Como recíproca de $\delta(\lambda)$ definida em (22), a função $\phi(\lambda)$ não possui polos finitos e possui zeros nos valores negativos dos autovalores da matriz H_c (veja Figuras 3, 6, 9, 12 e 15). Por conseguinte, ao invés de resolver o problema (23), vamos resolver a *equação secular*:

$$\phi(\lambda) = \frac{1}{\Delta_c}. \tag{28}$$

Com isso, concluímos que a melhor maneira de encontrar o parâmetro λ^* é avaliando a situação particular em que $\lambda^* = 0$, dada no Corolário 1, ou resolvendo a equação secular (28). O parâmetro λ^* nos dará uma solução p^* que será o tamanho do passo para o problema (6).

Para ilustrar como a solução p^* pode se comportar a cada iteração, vamos analisar cinco casos possíveis, baseados em duas possíveis propriedades para a matriz hessiana H_c: ser positiva definida ou não ser definida positiva.

Se H_c é positiva definida, caso $\delta(0) = \|p(0)\| < \Delta_c$ então a solução de (6) estará no interior da região de confiança, conforme ponto a nas Figuras 2 e 3 e ponto A na Figura 4. Se $\delta(0) = \|p(0)\| \geq \Delta_c$ então a solução de (6) estará na fronteira da região de confiança, conforme ponto b nas Figuras 2 e 3 e ponto B na Figura 4.

Quando H_c não é positiva definida, podem ocorrer duas possibilidades para seu menor autovalor: $\mu_1 \equiv 0$, ou $\mu_1 < 0$.

No primeiro caso, um autovetor de H_c associado ao autovalor nulo será uma direção de curvatura nula para H_c, isto é, $H_c v_1 = 0$. Caso a componente do vetor g_c no subespaço associado ao menor autovalor de H_c seja nula, isto é, se $g_c^T v_1 = 0$, então $-\mu_1$ não é polo da função (24). Se, além disso, o raio de confiança for maior que $\delta(0)$, então ocorre o chamado *hard case*: o problema (6) possui infinitas soluções, e uma solução na fronteira da bola pode ser obtida por meio da direção de curvatura nula v_1. Este caso pode ser acompanhado pelas Figuras 5, 6 e 7. Ainda que a matriz H_c seja semidefinida positiva e singular, quando zero é um polo da função secular então a solução global de (6) situa-se na fronteira da região de confiança. Isto pode ser visto nas Figuras 8, 9, 10.

No segundo caso, H_c é uma matriz indefinida e o autovetor v_1 é uma direção de curvatura negativa, de tal forma que a solução do problema (6) situa-se obrigatoriamente na fronteira da região de confiança, conforme ilustram as Figuras 11, 12 e 13. Vale dizer que no caso indefinido também pode acontecer o *hard case*, basta que $g_c^T v_1 = 0$ (pois então $-\mu_1$ não é polo da função secular) e que o raio de confiança seja suficientemente grande (maior que $\delta(-\mu_1)$). Esta situação pode ser visualizada nas Figuras 14, 15 e 16.

O desenvolvimento feito até aqui nos dá uma ideia de como resolver o problema (5) com métodos de região de confiança. O parâmetro λ^*, multiplicador de Lagrange do problema (6), quando distinto de zero, pode ser visto como um "regularizador" ou "modificador" do modelo para que este se torne convexo, caso não o seja, e que o minimizador esteja na fronteira da região de confiança. Se o parâmetro for suficientemente grande para tornar o modelo estritamente convexo, o minimizador será único.

O algoritmo para resolver o problema (28) será explicitado na próxima seção, em que vamos introduzir a ideia de Levenberg-Marquardt para resolver o problema (1).

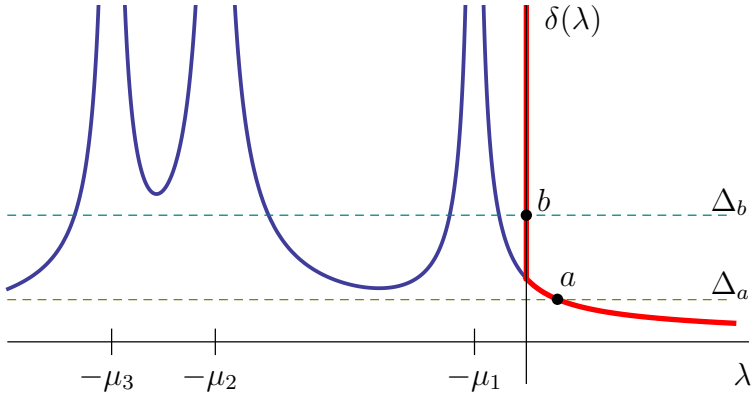


Figura 2: Função secular no caso estritamente convexo. Os valores $-\mu_j, j = 1, 2, 3$, são pólos dessa função. Os pontos $a \equiv (\lambda_a, \delta(\lambda_a)) = (\lambda_a, \Delta_a)$ e $b \equiv (0, \Delta_b)$, que correspondem às soluções dos problemas delimitados pelas bolas de raios Δ_a e Δ_b, ilustram, respectivamente, um caso típico de solução na fronteira e outro, no interior da região de confiança. A curva em destaque define a curva solução da equação (23) para cada raio de confiança.

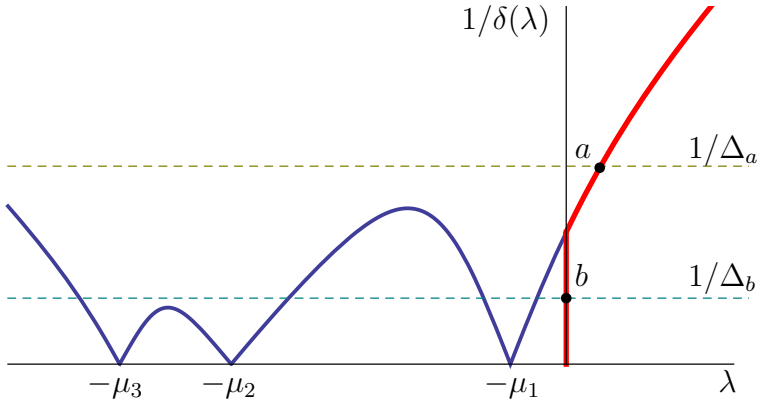


Figura 3: Recíproca da função secular no caso estritamente convexo. Os valores $-\mu_j$, $j = 1, 2, 3$, são zeros dessa função. Os pontos demarcados são $a \equiv (\lambda_a, 1/\delta(\lambda_a)) = (\lambda_a, 1/\Delta_a)$ e $b \equiv (0, 1/\Delta_b)$. Note que para $\lambda \geq -\mu_1$, a função $1/\delta(\lambda)$ é quase linear, o que favorece o desempenho do Método de Newton. A curva em destaque define a curva solução da equação (28) para cada raio de confiança.

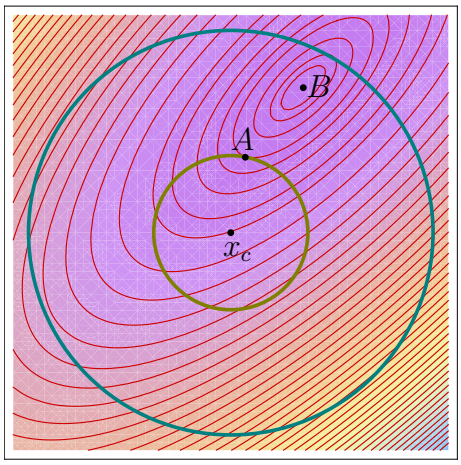


Figura 4: Curvas de nível de uma quadrática estritamente convexa e fronteiras das regiões de confiança centradas em x_c, de raios Δ_a e Δ_b ($\Delta_a < \Delta_b$), respectivamente, às quais correspondem as soluções $A \equiv x_c + p(\lambda_a)$ e $B \equiv x_c + p(0)$, conforme Figuras 2 e 3.

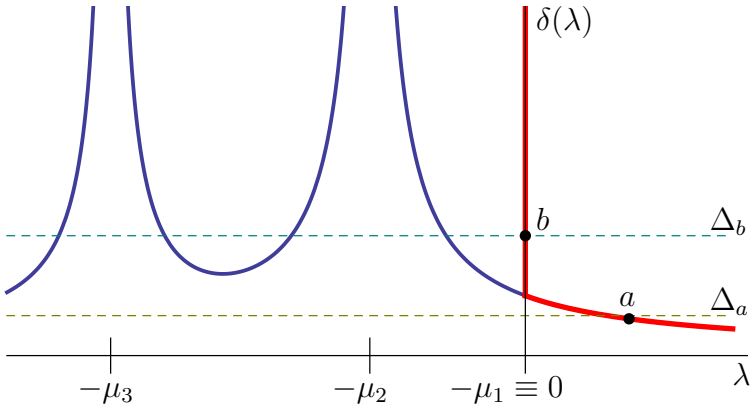


Figura 5: Função secular no caso convexo e singular, com possível *hard case*.

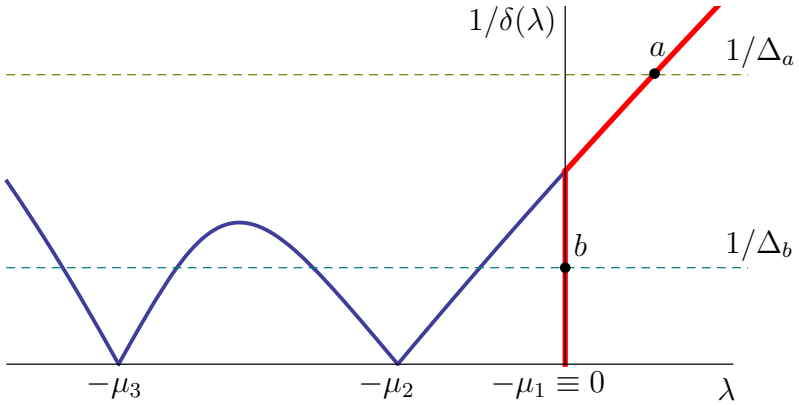


Figura 6: Recíproca da função secular no caso convexo e singular, com possível *hard case*.

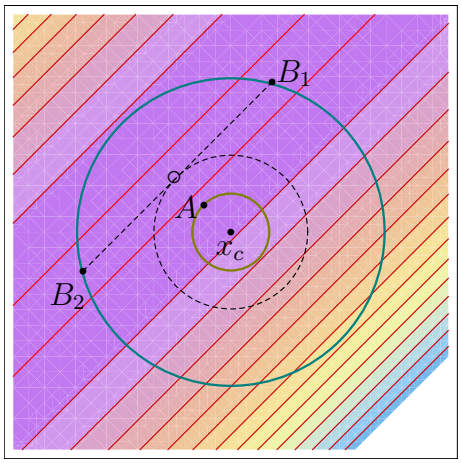


Figura 7: Curvas de nível de uma quadrática convexa e singular. A circunferência tracejada delimita a região limítrofe, de raio $\delta(0)$, em cujo exterior ocorre o *hard case*. Os pontos B_1 e B_2 correspondem a duas soluções na fronteira da região de raio Δ_b, com mesmo valor da quadrática q_c. Estas soluções podem ser obtidas a partir da solução na região limítrofe ($\circ$), somada a um múltiplo conveniente do autovetor v_1, conforme indica o segmento de reta tracejado.

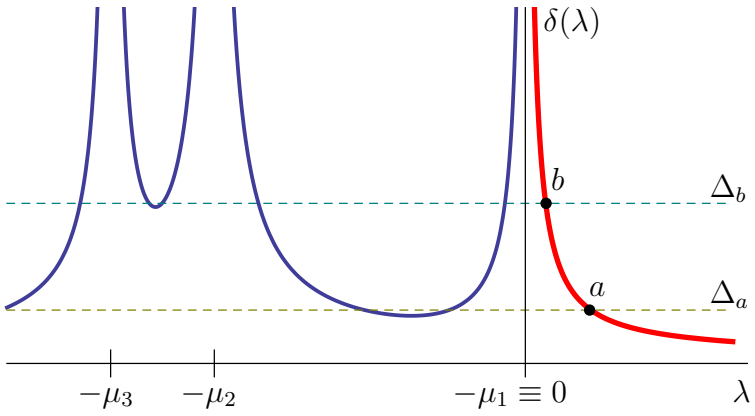


Figura 8: Função secular no caso convexo com hessiana singular.

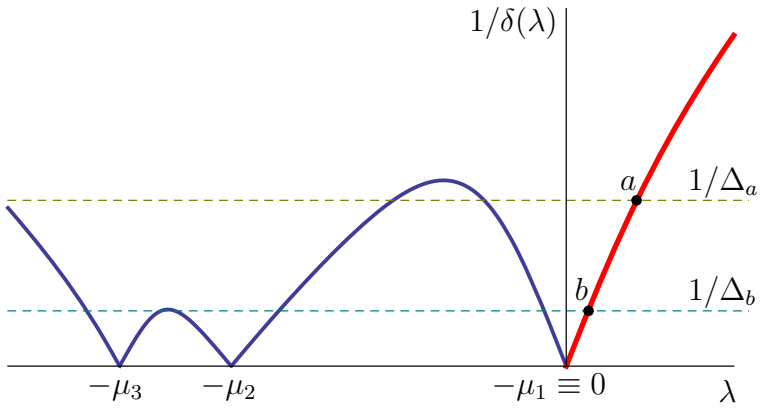


Figura 9: Recíproca da função secular no caso convexo com hessiana singular.

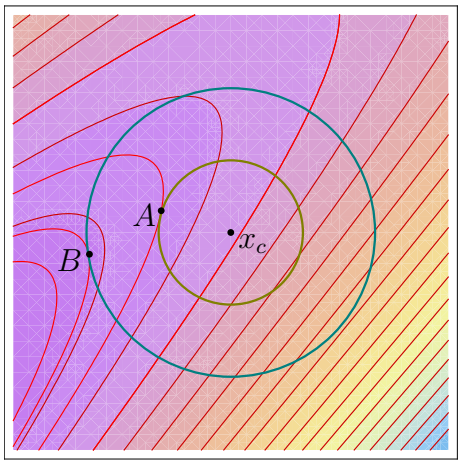


Figura 10: Curvas de nível de uma quadrática convexa com hessiana singular.

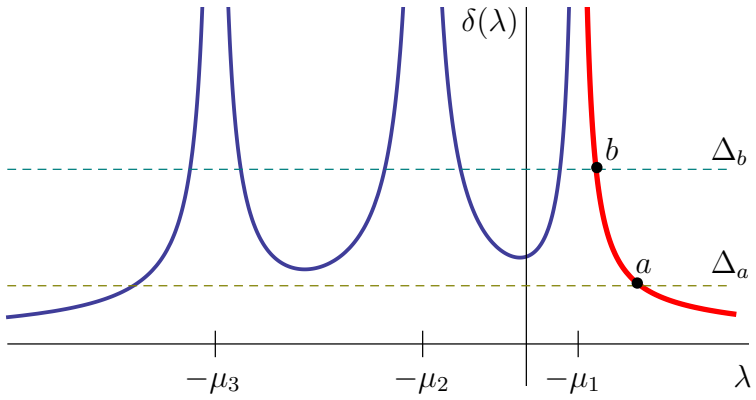


Figura 11: Função secular no caso indefinido.

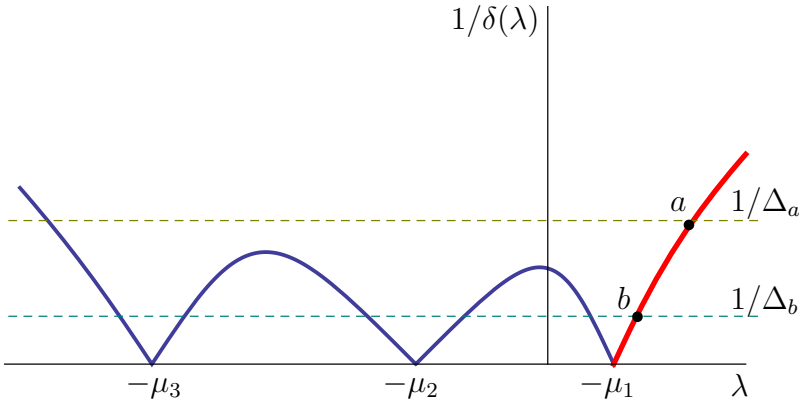


Figura 12: Recíproca da função secular no caso indefinido.

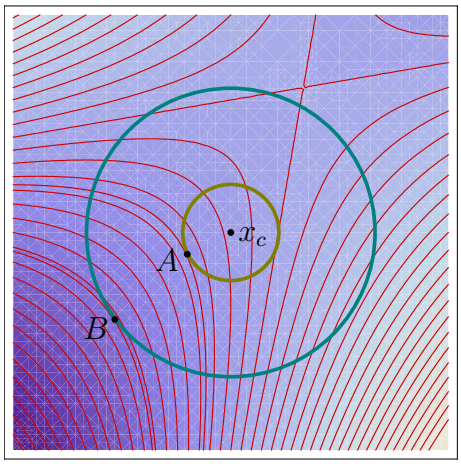


Figura 13: Curvas de nível de uma quadrática indefinida.

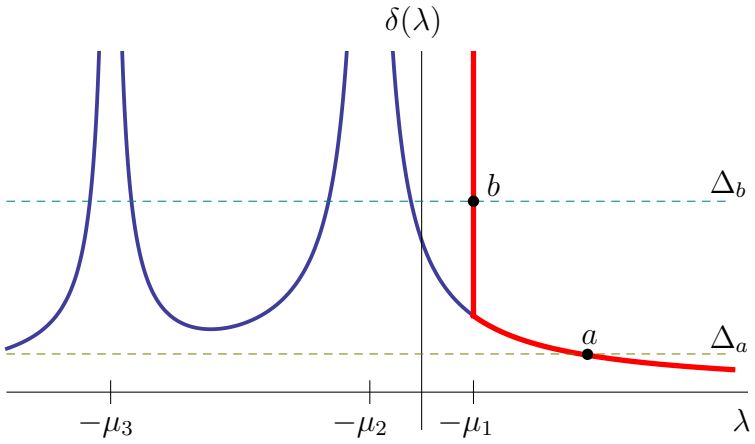


Figura 14: Função secular no caso indefinido, com possível *hard case*.

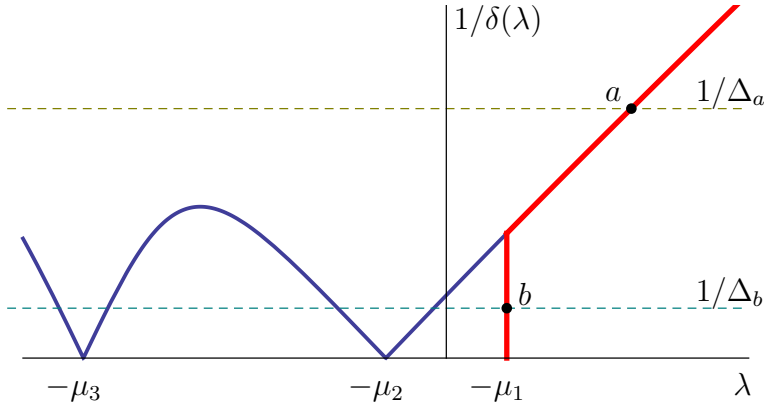


Figura 15: Recíproca da função secular no caso indefinido, com possível *hard case*.

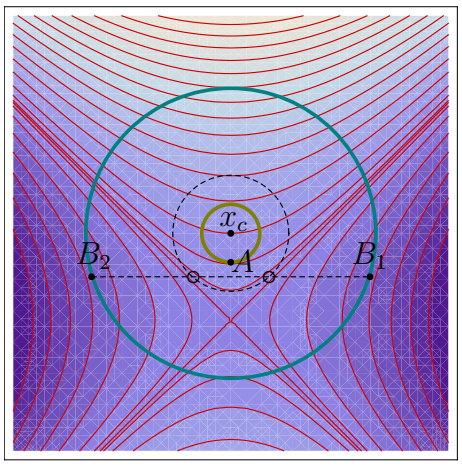


Figura 16: Curvas de nível de uma quadrática indefinida. A circunferência tracejada delimita a região limítrofe, de raio $\delta(-\mu_1)$, em cujo exterior ocorre o *hard case*. Os pontos B_1 e B_2 correspondem a duas soluções na fronteira da região de raio Δ_b, com mesmo valor da quadrática q_c. Estas soluções podem ser obtidas a partir das soluções na região limítrofe ($\circ$), somadas a um múltiplo conveniente do autovetor v_1, associando a μ_1, o menor autovalor da H_c, conforme indica o segmento de reta tracejado.

4 O algoritmo de Levenberg-Marquardt

O breve desenvolvimento do tema de regiões de confiança feito até agora objetiva uma base para podermos descrever e analisar o método de Levenberg-Marquardt (LM) para a resolução do problema (1). De fato, esse método é tido como o antecessor do método de região de confiança para o problema geral de otimização irrestrita.

O algoritmo de LM trabalha com o modelo linear da função residual. Então, dada a linearização de $F(x)$ em torno de um ponto dado $x_c \in \mathbb{R}^n$:

$$M_c(x_c + p) = F(x_c) + J(x_c)p,$$

com uma pequena mudança de notação, temos:

$$m_c = J_c p + F_c, \tag{29}$$

onde $F_c \in \mathbb{R}^m$ e $J_c \in \mathbb{R}^{m\times n}$, com $m > n$. Nosso objetivo será resolver, a cada iteração, o caso linear do problema de quadrados mínimos:

$$\begin{array}{l} \min \frac{1}{2}\|J_c p + F_c\|^2 \\ \text{s.a } \|p\| \leq \Delta_c, \end{array} \tag{30}$$

onde $\Delta_c > 0$, que nos dá o subproblema de região de confiança.

A solução para o problema (30) é consequência direta do Teorema 1, e é apresentada no corolário a seguir.

Corolário 2. *O vetor $p^* \in \mathbb{R}^n$ é solução para o subproblema de região de confiança (30) se e somente se p^* é viável e existir $\lambda^* \geq 0$ tal que*

$$(J_c^T J_c + \lambda^* I)p^* = -J_c^T F_c \tag{31a}$$

$$\lambda^*(\|p^*\| - \Delta_c) = 0. \tag{31b}$$

Prova. Segue do Teorema 1, visto que a condição (7c) é cumprida intrinsecamente dado que $J_c^T J_c$ é semidefinida positiva e $\lambda^* \geq 0$. As condições (31a) e (31b) seguem de (7a) e (7b), respectivamente. ■

A ligação deste problema com o método de região de confiança está na estratégia para encontrar λ. Na proposta original de Levenberg e Marquardt, o valor de λ era ajustado diretamente por um esquema heurístico; com a estratégia de região de confiança, ajustaremos os valores de λ encontrando zeros da equação secular apresentada em (28), definindo, neste caso, para $\lambda > 0$,

$$p(\lambda) = -(J_c^T J_c + \lambda I)^{-1} J_c^T F_c. \tag{32}$$

Definido o problema e a solução básica, vamos agora desenvolver o Algoritmo 1 como uma proposta para a solução de (30), de forma a compor uma implementação robusta, eficiente e confiável, baseada na proposta de Moré em [5] e usando elementos das demais referências bibliográficas.

Algoritmo 1: Método de Levenberg-Marquardt – Básico

Dados: $x_c \in \mathbb{R}^n$ o ponto corrente, $F_c \in \mathbb{R}^m$ e $J_c \in \mathbb{R}^{m\times n}$ a função residual e a matriz jacobiana avaliadas em x_c, Δ_c o raio da região de confiança atual, $\phi(\lambda)$ definida em (28) e $\Psi(x)$ como definida em (1), faça:

1 Encontre $\lambda^* \geq 0$ e o passo $p^* \in \mathbb{R}^n$ tais que $\phi(\lambda^*) = \dfrac{1}{\Delta_c}$ e avalie $F(x_c + p^*)$;
2 **se** $\Psi(x_c + p^*) < \Psi(x_c)$ **então** defina $x_n = x_c + p^*$ e avalie a jacobiana no novo ponto J_n;
3 **senão** mantenha os valores de x_c e J_c;
4 defina o novo raio Δ_n.

4.1 Caracterização da solução do problema de quadrados mínimos

A solução do problema de quadrados mínimos, expressa em (32), estará bem definida se a matriz $J_c^T J_c + \lambda I$ for positiva definida, para todo $\lambda \in \mathbb{R}$ assumidos no processo iterativo para resolução do problema. Uma maneira para garantir essa condição é calcular a fatoração de *Cholesky* para esta matriz, que existirá somente se ela for positiva definida.

Entretanto, a condição especial do problema (30) nos garante que $J_c^T J_c$ é, ao menos, positiva semidefinida e que, para todo $\lambda > 0$, $J_c^T J_c + \lambda I$ é positiva definida. Como $\lambda = 0$ define um caso especial em que a solução será interna (ver Seção 3.2), então a expressão (32) é valida ao longo da resolução do problema.

A existência de solução para o problema (30) é garantida pelo Corolário 2. A equação (31a) pode ser vista como um sistema de equações normais para o caso linear do problema de quadrados mínimos:

$$\begin{pmatrix} J_c \\ \sqrt{\lambda} I \end{pmatrix} p \cong - \begin{pmatrix} F_c \\ 0 \end{pmatrix}. \tag{33}$$

Essa equivalência entre as soluções de (31a) e de (33) é muito importante do ponto de vista computacional, pois nos permite resolver o problema sem avaliar o produto $J_c^T J_c$, reduzindo pela metade o número de operações. Por conseguinte, temos em mãos um primeiro recurso interessante: calcular a decomposição QR da matriz dos coeficientes em (33):

$$Q_\lambda^T \begin{pmatrix} J_c \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R_\lambda \\ 0 \end{pmatrix}, \tag{34}$$

onde $Q_\lambda \in \mathbb{R}^{(m+n)\times(m+n)}$, ortogonal e $R_\lambda \in \mathbb{R}^{n\times n}$, triangular superior, satisfazem $R_\lambda^T R_\lambda = J_c^T J_c + \lambda I$.

Computacionalmente, a melhor maneira para decompor (34) é combinar os métodos de Householder e Givens.

O primeiro passo consiste em usar tranformações de Householder – veja o Apêndice A, Algoritmo 5 – para calcular a seguinte fatoração QR:

$$J_c = Q \begin{pmatrix} R_c \\ 0 \end{pmatrix}. \tag{35}$$

Isso implica:

$$\begin{pmatrix} Q^T & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} J_c \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R_c \\ 0 \\ \sqrt{\lambda} I \end{pmatrix}. \tag{36}$$

A matriz resultante em (36) possui uma estrutura muito característica. A matriz R_c é triangular superior, enquanto a matriz $\sqrt{\lambda} I$ é diagonal. Para que esta matriz fique triangular superior, basta eliminar os n termos não nulos da matriz $\sqrt{\lambda} I$ com uma sequência de $n(n+1)/2$ rotações de Givens, aplicadas em cada linha i de $\sqrt{\lambda} I$, simultaneamente a cada linha $i+j$ de R_c, para $j = 0, \ldots, (n-i)$, começando da linha n (veja o Apêndice B, Algoritmo 6).

Expressando todas as rotações de Givens em uma matriz $G_\lambda \in \mathbb{R}^{(m+n)\times(m+n)}$, ortogonal, por meio dos respectivos produtos, temos que:

$$G_\lambda^T \begin{pmatrix} R_c \\ 0 \\ \sqrt{\lambda} I \end{pmatrix} = \begin{pmatrix} R_\lambda \\ 0 \\ 0 \end{pmatrix}. \tag{37}$$

O método adotado em (37) possui a grande vantagem de que, para cada alteração no valor de λ, não é necessário recalcular a decomposição QR em (36), mas basta calcular a matriz G_λ e fazer a correção em (37) para obter novamente o fator R_λ tal que $R_\lambda^T R_\lambda = J_c^T J_c + \lambda I$. A matriz R_λ será o fator de *Cholesky* caso todos os termos da sua diagonal sejam positivos. Este fator será utilizado posteriormente no cálculo da raiz da equação secular, que nos dará elementos suficientes para atualizar o Algoritmo 1 para uma versão computacionalmente mais robusta e eficiente.

4.2 O método de Newton e a raiz da equação secular

O método de Newton aplicado ao problema (28) gera uma sequência λ_k tal que:

$$\lambda_{k+1} = \lambda_k - \frac{\phi(\lambda_k)}{\phi'(\lambda_k)}. \tag{38}$$

Para reduzir o custo no cálculo da derivada da função $\phi(\lambda)$, podemos caracterizar a iteração de Newton como segue no seguinte lema:

Lema 1. *Considere o problema (28) de encontrar a raiz da equação secular. Seja R_λ a matriz triangular calculada em (37), $R_\lambda^T R_\lambda p_k = -J_c^T F_c$ e tal que $R_\lambda^T q_k = p_k$ e seja Δ_c o raio da região considerada. A sequência λ_k gerada pelo método de Newton definida em (38) é equivalente a:*

$$\lambda_{k+1} = \lambda_k + \left(\frac{\|p_k\|}{\|q_k\|}\right)^2 \left(\frac{\|p_k\| - \Delta_c}{\Delta_c}\right). \tag{39}$$

Prova. A derivada primeira da função $\phi(\lambda_k)$ é dada por:

$$\begin{aligned} \phi'(\lambda_k) &= \frac{d}{d\lambda}\left(\frac{1}{\|p(\lambda_k)\|}\right) \\ &= \frac{d}{d\lambda}(\|p(\lambda_k)\|^2)^{-1/2} \\ &= -\frac{1}{2}(\|p(\lambda_k)\|^2)^{-3/2}\frac{d}{d\lambda}(\|p(\lambda_k)\|^2). \end{aligned} \tag{40}$$

Por sua vez:

$$\frac{d}{d\lambda}(\|p(\lambda_k)\|^2) = 2p(\lambda_k)^T p'(\lambda_k). \tag{41}$$

Com isso, por (41) em (40), temos que:

$$\phi'(\lambda_k) = -\frac{p(\lambda_k)^T p'(\lambda_k)}{\|p(\lambda_k)\|^3}. \tag{42}$$

Seja $p(\lambda_k) = p_k$. Então, temos que:

$$\begin{aligned} \frac{\phi(\lambda_k)}{\phi'(\lambda_k)} &= -\frac{\dfrac{1}{\|p_k\|} - \dfrac{1}{\Delta_c}}{\dfrac{p_k^T p_k'}{\|p_k\|^3}} \\ &= -\left(\frac{\|p_k\|^2}{p_k^T p_k'}\right)\left(\frac{\Delta_c - \|p_k\|}{\Delta_c}\right). \end{aligned} \tag{43}$$

Analisando o termo $p_k^T p_k'$, temos que:

$$\begin{aligned} p_k^T p_k' &= -F_c^T J_c (J_c^T J_c + \lambda_k I)^{-3} J_c^T F_c \\ &= -p_k^T (J_c^T J_c + \lambda_k I)^{-1} p_k \\ &= -p_k^T (R_\lambda^T R_\lambda)^{-1} p_k \\ &= -p_k^T R_\lambda^{-1} R_\lambda^{-T} p_k \\ &= -q_k^T q_k \\ &= -\|q_k\|^2. \end{aligned} \tag{44}$$

Logo, por (44) em (43), vem que:

$$\frac{\phi(\lambda_k)}{\phi'(\lambda_k)} = -\left(\frac{\|p_k\|}{\|q_k\|}\right)^2 \left(\frac{\|p_k\| - \Delta_c}{\Delta_c}\right)$$

que, em (38) nos dá (39). ■

Vamos analisar este método iterativo frente à estrutura da própria equação secular. Como dito anteriormente, na Seção 3.2, a equação secular (28) possui zeros e não possui polos finitos, os zeros acontecem, neste caso, nos valores negativos dos autovalores da $J_c^T J_c$. Entretanto, nosso objetivo é encontrar valores positivos para λ. Portanto, para que o método de Newton nos seja uma ferramenta confiável, é necessário estabelecer salvaguardas para evitar que o método convirja para uma solução indesejável da equação secular.

4.2.1 Salvaguarda para o método de Newton

Para isso, vamos seguir o proposto em [5] e estabelecer uma cota inferior e outra superior para os valores que λ pode assumir no processo iterativo, impondo que a cada iteração k,

$\lambda_k \in [\ell_k, u_k]$, tal que ℓ_0 é uma estimativa para o menor autovalor da $J_c^T J_c$ e u_0 estabelece um limite superior para λ. Como $J_c^T J_c$ é, ao menos, semidefinida positiva,

$$\ell_0 = 0. \tag{45}$$

Para estimar o limitante inicial u_0, vamos usar a aproximação de *Rayleigh* em que, para qualquer matriz $B \in \mathbb{R}^{p\times p}$ com autovalores definidos por $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_p$, temos:

$$\mu_1 \leq \frac{u^T B u}{u^T u} \leq \mu_p \tag{46}$$

para todo $u \in \mathbb{R}^p$

Aplicando esta desigualdade à nossa matriz $J_c^T J_c + \lambda I$, sendo $\mu_1 \leq \mu_2 \leq \ldots \leq \mu_n$ autovalores, bem definidos, da matriz $J_c^T J_c$, simétrica, e $p \in \mathbb{R}^n$, temos que:

$$(\mu_1 + \lambda)^2 \leq \frac{p^T (J_c^T J_c + \lambda I)^T (J_c^T J_c + \lambda I) p}{p^T p} \leq (\mu_n + \lambda)^2. \tag{47}$$

Tomando p como definido em (32) tal que $\|p\|^2 = \Delta_c^2$, então (47) nos dá:

$$(\mu_1 + \lambda)^2 \leq \frac{F_c^T J_c J_c^T F_c}{\Delta_c^2} \leq (\mu_n + \lambda)^2.$$

Como nossa estimativa para o menor autovalor é zero por (45), vem que:

$$\lambda \leq \frac{\|J_c^T F_c\|}{\Delta_c} = u_0, \tag{48}$$

o que nos dá a cota superior para os valores que λ pode assumir.

Essas cotas serão atualizadas a cada iteração da seguinte maneira:

Algoritmo 2: Atualização das cotas ℓ e u

A cada iteração k, começando de $k = 0$, dados as cotas iniciais definidas em (45) e (48) e a função $\delta(\lambda)$ como definida em (23), faça:

se $\delta(\lambda_k) < 0$ **então**
  $\ell_{k+1} = \ell_k$;
  $u_{k+1} = \lambda_k$;
senão
  $\ell_{k+1} = \max\left\{\ell_k, \lambda_k - \dfrac{\delta(\lambda_k)}{\delta'(\lambda_k)}\right\}$;
  $u_{k+1} = u_k$;
fim

Cabe aqui uma breve discussão sobre as atualizações propostas no Algoritmo 2. Repare que no algoritmo usamos a função $\delta(\lambda)$ ao invés de usar sua recíproca $\phi(\lambda)$ definida em (27). Embora ela não seja ideal para ser usada com o método de Newton para encontrar zero de

funções, neste caso, sua convexidade no intervalo de trabalho é uma característica que pode ser bem aproveitada frente à quase linearidade da função $\phi(\lambda)$ dada em (27). Em primeira instância, nós usamos essa função no Passo 1 pois sabe-se que, quando negativa, o valor de λ corrente está acima do valor λ^* procurado (ver Figuras 2 e 8). Em outra instância, usamos essa função para estabelecer a cota inferior caso a condição do Passo 1 não seja satisfeita. No Passo 4, definimos a nova cota inferior como sendo o valor máximo entre a cota atual e o próximo iterando de Newton aplicado à função $\delta(\lambda)$. O valor de $\delta(\lambda_k)$ pode ser avaliado na iteração e o valor de sua derivada $\delta'(\lambda_k)$ é conhecido, pois:

$$\begin{aligned} \delta'(\lambda) &= \frac{p(\lambda)^T p'(\lambda)}{\|p(\lambda)\|} \\ &\overset{(44)}{=} -\frac{\|q\|^2}{\|p\|}. \end{aligned}$$

Esse valor aplicado a iteração de Newton nos dá a seguinte expressão:

$$\ell_{k+1} = \max\left\{\ell_k, \lambda_k + \frac{\|q\|^2}{\|p\|}\right\}. \tag{49}$$

Definidas as cotas, a cada iteração verifica-se se $\lambda_k \notin [\ell_k, u_k]$, caso isso aconteça, λ_k será propriamente inserido dentro do intervalo; essa estratégia será apresentada adiante.

4.2.2 O algoritmo para o problema da equação secular

O método para encontrar o zero da equação secular exige uma estimativa inicial λ_0 para gerar a sequência λ_k. Como a cota inferior em (45) é $\ell_0 = 0$, vamos definir $\lambda_0 = 0$ para que, caso haja uma solução interior do subproblema como apresentada na Seção 3.2, ela seja apontada logo no início da execução, ou já seja descartada a possibilidade de sua ocorrência.

Baseados nos elementos discutidos até agora, apresentamos o Algoritmo 3 para resolver o problema de encontrar o zero da equação secular definida em (28).

Para que o Algoritmo 3 se torne útil na prática, nos resta discutir sobre critérios de parada no Passo 9; é o que faremos na próxima seção.

4.2.3 Critérios de parada para o algoritmo

A discussão de critérios de parada está intimamente ligada com a resolução do problema. E ainda, é necessário nas condições de parada estabelecer um limitante para a execução do algoritmo caso ele não convirja para a solução. Em nosso caso, vamos estabelecer cinco critérios de parada, que são:

1. *O algoritmo convergiu para a solução do problema, dada certa tolerância τ.*

   Neste caso, avaliamos se a função secular já está suficientemente próxima de zero, isto é, se:

   $$| \phi(\lambda_k) | < \tau.$$

Algoritmo 3: Algoritmo para encontrar a raiz da equação secular

Dados $x_c \in \mathbb{R}^n$ o ponto corrente, $F_c \in \mathbb{R}^m$ e $J_c \in \mathbb{R}^{m\times n}$ a função residual e a matriz jacobiana avaliadas em x_c, $\lambda_0 = 0$ a estimativa inicial para a raiz da equação secular, Δ_c o raio da região de confiança corrente e uma tolerância τ para convergência do algoritmo à solução, faça:

1 $\ell_0 \leftarrow 0$;

2 $u_0 \leftarrow \dfrac{\|J_c^T F_c\|}{\Delta_c}$;

3 $J_c \leftarrow Q_c \begin{pmatrix} R_c \\ 0 \end{pmatrix}$; `// Usando reflexões de Householder, ver Apêndice A`

4 **para** $k = 0, 1, \ldots$ **faça**

5 Defina a matriz $\begin{pmatrix} R_c \\ 0 \\ \sqrt{\lambda_k} I \end{pmatrix}$;

6 $\begin{pmatrix} R_c \\ 0 \\ \sqrt{\lambda_k} I \end{pmatrix} = G_\lambda \begin{pmatrix} R_\lambda \\ 0 \\ 0 \end{pmatrix}$; `// Usando rotações de Givens, ver Apêndice B`

7 Resolva $R_\lambda^T R_\lambda p_k = -J_c^T F_c$;

8 Resolva $R_\lambda^T q_k = p_k$;

9 *Analisar os critérios de parada do algoritmo*;

10 *Atualizar as cotas ℓ e u cf. Algoritmo 2*;

11 Obtenha λ_{k+1} de (39);

12 **se** $\lambda_{k+1} \notin [\ell_k, u_k]$ **então** $\lambda_{k+1} \leftarrow \max\{\sqrt{\ell_k * u_k}, \ell_k + 0.1 * (u_k - \ell_k)\}$;

fim

2. *Houve pequena variação em λ.*

   Caso a variação em λ seja muito pequena, dada certa tolerância τ, interrompe-se a execução do algoritmo e assume-se λ_k como solução. Neste caso, avaliamos

   $$| \lambda_{k+1} - \lambda_k | < \tau.$$

3. *Houve pequena variação em $\delta(\lambda)$.*

   Caso a variação em $\delta(\lambda)$ seja muito pequena, menor que certa tolerância τ, interrompe-se a execução do algoritmo e assume-se como solução λ_k. Em outras palavras,

   $$| \|p_{k+1}\| - \|p_k\| | < \tau.$$

4. *Convergência interior do problema*

   Avaliamos o caso de λ_k ser zero e p_k correspondente ser viável (ver Seção 3.2):

   $$\|p_k\| = \|p(0)\| \leq \Delta_c, \lambda_k = 0.$$

5. *Quantidade de iterações do processo*

   Em nosso caso, vamos estabelecer um limite de `ITOL` iterações, definido pelo usuário, para que o algoritmo convirja à solução, senão decretamos falha na convergência do algoritmo e interrompemos a execução:

$$k \leq \texttt{ITOL}.$$

4.3 Atualização do raio da região de confiança

Nesta seção, vamos apresentar os fundamentos para atualizar o raio Δ_c da região de confiança. Sabe-se que para resolver o problema (1), não trabalhamos diretamente com a função objetivo $\Psi(x)$, mas sim com a aproximação linear definida em (29) para a função residual $F(x)$ em torno de um ponto x_c. Em nosso caso, esse é um problema quadrático. A cada iteração, resolvemos um problema restrito a uma bola de raio Δ_c. Então, o descréscimo que obtemos é no modelo linear, e não diretamente na função residual; sua redução da está ligada à representação do modelo na região de raio Δ_c. A atualização deste raio está justamente neste ponto: caso obtenha-se um descréscimo satisfatório na função residual, temos que o modelo linear representou bem a função na região definida, então aumenta-se o valor de Δ_c para que o modelo seja melhor aproveitado. Caso contrário, diminui-se o tamanho do raio, para que possamos ter uma boa representação da função residual e obtenhamos um descréscimo aceitável.

Para avaliar a relação entre o descréscimo da função objetivo e o descréscimo do modelo definido no ponto, vamos seguir [5] e avaliar a razão entre a *redução atual* (redução da função objetivo) e a *redução predita* (redução do modelo), dada por:

$$\rho(p) = \frac{\|F(x_c)\|^2 - \|F(x_c + p)\|^2}{\|F(x_c)\|^2 - \|F(x_c) + J(x_c)p\|^2}, \tag{50}$$

onde p é o passo ótimo obtido na solução do problema (30).

Analisando o quociente (50), temos que o numerador representa a redução da função objetivo partindo do ponto corrente x_c para o ponto candidato $x_n = x_c + p$ e o denominador representa essa redução a partir do modelo. No melhor dos casos, se $F(x)$ fosse linear, então $\rho(p) = 1$ para todo p, e teríamos que o modelo seria uma ótima representação para a função, pois seria a própria função. Já no pior dos casos, se $\|F(x+p)\| > \|F(x)\|$, então $\rho(p) < 0$ e, ao invés de obter um descréscimo na função objetivo, obteríamos um aumento, o que significa que o modelo foi uma péssima representação para a função na região de confiança definida.

Com isso, podemos definir algumas condições para manipular o raio Δ_c:

1. Se $\|F(x+p)\| > \|F(x)\|$, então a houve um aumento na função objetivo e vamos tomar $\rho(p) = 0$ e diminuir o raio Δ_c.

2. Se $\rho(p) \leq 1/4$, então o modelo não representou muito bem a função na região definida, então devemos reduzir o tamanho do raio Δ_c.

3. Se $\rho(p) \geq 3/4$, então temos que o modelo representou muito bem a função na região definida; por isso, podemos aumentar o tamanho de Δ_c.

4. Se $1/4 < \rho(p) < 3/4$, tivemos um nível de redução razoável, então mantém-se o valor de Δ_c.

O problema agora resume-se em encontrar a melhor maneira de calcular o quociente em (50), que nos dirá o que fazer com o valor de Δ_c e, ainda, qual a melhor maneira de aumentar ou diminuir o valor de Δ_c. É o que vamos discutir nas seções subsequentes.

4.3.1 Computando o quociente de redução relativa

Nesta seção vamos seguir as ideias propostas por Moré em [5], apenas detalhando-as. Nosso objetivo aqui é evitar ou amenizar o máximo possível a ocorrência de *overflows* e *underflows* na computação do quociente (50). Por isso, precisamos manipular o denominador desta relação, pois este pode ser o responsável em causar resultados indesejados. De (31a), temos que:

$$J_c^T J_c p = -J_c^T F_c - \lambda p \tag{51a}$$
$$-F_c^T J_c = \lambda p^T + p^T J_c^T J_c. \tag{51b}$$

Para simplificar a notação, vamos reescrever (50):

$$\rho(p) = \frac{\|F_c\|^2 - \|F_n\|^2}{\|F_c\|^2 - \|F_c + J_c p\|^2}.$$

Disso vem que:

$$\begin{aligned} \|F_c\|^2 - \|F_c + J_c p\|^2 &= F_c^T F_c - F_c^T F_c - F_c^T J_c p - p^T J_c^T F_c - p^T J_c^T J_c p \\ &\overset{(51a)}{=} -F_c^T J_c p - p^T J_c^T F_c + p^T(-J_c^T F_c - \lambda p) \\ &\overset{(51b)}{=} \lambda p^T p + p^T J_c^T J_c p + \lambda p^T p \\ &= \|J_c p\|^2 + 2\lambda \|p\|^2. \end{aligned} \tag{52}$$

A relação (52) possui duas consequências importantes:

1. Podemos reescrever o quociente (50), dividindo todos os termos por $\|F_c\|^2$, obtendo a seguinte relação:

$$\rho(p) = \frac{1 - \left(\frac{\|F_n\|}{\|F_c\|}\right)^2}{\left(\frac{\|J_c p\|}{\|F_c\|}\right)^2 + 2\lambda \left(\frac{\|p\|}{\|F_c\|}\right)^2}. \tag{53}$$

2. Manipulando a relação (52), temos que:

$$\|F_c\|^2 \geq \|F_c\|^2 - \|F_c + J_c p\|^2 = \|J_c p\|^2 + 2\lambda \|p\|^2 \geq \|J_c p\|^2$$
$$\boxed{\|J_c p\| \leq \|F_c\|} \tag{54a}$$
$$\|F_c\|^2 \geq \|F_c\|^2 - \|F_c + J_c p\|^2 = \|J_c p\|^2 + 2\lambda \|p\|^2 \geq \|p\|^2$$
$$\boxed{\|p\| \leq \|F_c\|} \tag{54b}$$

Disto, podemos concluir:

1. O denominador de (50) será sempre não-negativo.
2. As relações (54) nos permitem afirmar que a computação do denonimador não irá resultar em *overflow*.
3. O numerador jamais gerará *overflows* a não ser que $\|F_n\| \gg \|F_c\|$. Neste caso, não é necessário computar o fator ρ, pois ele visa avaliar a redução da função objetivo com relação ao modelo. Caso isso aconteça, assumiremos que $\rho(p) = 0$.

Tendo a melhor maneira de computar a redução relativa, vamos agora discutir como atualizar o tamanho do raio da região de confiança.

4.3.2 Atualizando o raio da região de confiança

A atualização do raio será feita em função do fator ρ, como discutido na Seção 4.3. A atualização do raio será feita em três instâncias: ou seu valor será mantido, ou aumentado ou diminuído. Aumentar o tamanho do raio é relativamente simples, basta multiplicá-lo por uma constante. Em nosso caso, vamos fazer:

$$\Delta_n = 2\|p^*\|. \tag{55}$$

Esta maneira de aumentar o valor de Δ_c é a maneira mais equilibrada, pois:

1. Se $\|p^*\| = \Delta_c$, então estamos dobrando o tamanho de Δ_c.
2. Se $\|p^*\| < \Delta_c,\ \lambda^* = 0$, então não é necessário nem aconselhável dobrar o tamanho do raio, pois o minimizador do modelo foi um ponto interior; neste caso estamos permitindo, numa próxima iteração, que dobre o tamanho do passo tomado.

Nos resta o problema de diminuir o raio. Para isso, vamos primeiro definir a função:

$$\gamma(\vartheta) \overset{\text{def}}{=} \frac{1}{2}\|F(x_c + \vartheta p^*)\|^2. \tag{56}$$

Desta função, conhecemos:

1. $\gamma(0) = \frac{1}{2}\|F_c\|^2$;
2. $\gamma'(0) = (p^*)^T J_c^T F_c$;
3. $\gamma(1) = \frac{1}{2}\|F_n\|^2$.

Com isso, definimos a parábola interpolante:

$$m_q(\vartheta) = [\gamma(1) - \gamma(0) - \gamma'(0)]\vartheta^2 + \gamma'(0)\vartheta + \gamma(0) \tag{57}$$

e seu minimizador ϑ^* será usado tal que:

$$\Delta_n = \vartheta^* \Delta_c. \tag{58}$$

Se $\vartheta^* \notin [\frac{1}{10}, \frac{1}{2}]$, então assumiremos o extremo do intervalo mais próximo do valor de ϑ^*. Como a função (57) é suave e convexa, seu minimizador é tal que $m'_q(\vartheta^*) = 0$. Então:

$$\begin{aligned} \vartheta^* &= -\frac{\gamma'(0)}{2[\gamma(1) - \gamma(0) - \gamma'(0)]} \\ &= -\frac{(p^*)^T J_c^T F_c}{\|F_n\|^2 - \|F_c\|^2 - 2(p^*)^T J_c^T F_c}. \end{aligned} \tag{59}$$

Divindo todos os termos em (59) por $\|F_c\|^2$, temos:

$$\vartheta^* = -\frac{\dfrac{(p^*)^T J_c^T F_c}{\|F_c\|^2}}{\left(\dfrac{\|F_n\|}{\|F_c\|}\right)^2 - 1 - 2\dfrac{(p^*)^T J_c^T F_c}{\|F_c\|^2}}. \tag{60}$$

Usando as ideias e conclusões apresentadas na Seção 4.3.1, vamos analisar (60) de forma a evitar a ocorrência de *overflows*. Pré-multiplicando (31a) avaliado em p^* por $(p^*)^T$, temos que:

$$-(p^*)^T(J_c^T J_c + \lambda^* I)p^* = (p^*)^T J_c^T F_c. \tag{61}$$

Então:

$$\begin{aligned} \frac{(p^*)^T J_c^T F_c}{\|F_c\|^2} &= -\left[\frac{(p^*)^T J_c^T J_c p^*}{\|F_c\|^2} + \lambda^* \frac{(p^*)^T p^*}{\|F_c\|^2}\right] \\ &= -\left[\left(\frac{\|J_c p^*\|}{\|F_c\|}\right)^2 + \lambda^* \left(\frac{\|p^*\|}{\|F_c\|}\right)^2\right] = \alpha. \end{aligned} \tag{62}$$

Podemos usar as relações (54) para concluir que não teremos nenhum *overflow* em (62). Então podemos reescrever a relação (60), multiplicando-a por $-\frac{1}{2}$ e substituindo α de (62):

$$\vartheta^* = \frac{\dfrac{1}{2}\alpha}{\alpha + \dfrac{1}{2}\left[1 - \left(\dfrac{\|F_n\|}{\|F_c\|}\right)^2\right]}. \tag{63}$$

A expressão (63) nos dá uma ferramenta confiável para encontrar o valor de ϑ^* e aplicá-lo na redução do raio Δ_c como definido em (58).

Com isso, encerramos a coleta de elementos para compor um algoritmo eficiente e robusto para resolução do problema (1) e estamos prontos para construí-lo; é o que faremos na próxima parte do texto.

4.4 A construção do algoritmo LMA

Nesta seção vamos apresentar a estratégia de Levenberg-Marquardt no Algoritmo 4, com todas as propriedades apresentadas neste trabalho.

Algoritmo 4: Método de Levenberg-Marquardt

Dados: $x_0 \in \mathbb{R}^n$ o ponto inicial, $F_0 \in \mathbb{R}^m$ e $J_0 \in \mathbb{R}^{m\times n}$ a função residual e a matriz jacobiana avaliadas em x_0, Δ_0 o raio da região de confiança inicial definido em (64), $\phi(\lambda)$ definida em (28) e $\Psi(x)$ como definida em (1), faça:

para $k = 0, 1, \ldots$ **faça**
- Encontre p_k e λ_c usando o Algoritmo 3;
- Avalie o ponto candidato $x_{k+1} = x_k + p_k$;
- Avalie o valor da função F_{k+1} no ponto candidato;
- Avalie a redução relativa ρ_k por (53);
- **se** $\rho_k > 0.001$ **então**
  - Aceite o ponto candidato x_{k+1};
  - Avalie a Jacobiana J_{k+1};
  - Avalie os critérios de parada;
- **senão**
  - Descarte o ponto candidato;
  - Assuma $x_{k+1} = x_k$, $F_{k+1} = F_k$ e $J_{k+1} = J_k$;
- **fim**
- **se** $\rho_k \leq 1/4$ **então**
  - Encontre ϑ^* por (63);
  - **se** $\vartheta^* < 1/10$ **então** $\vartheta^* = 1/10$;
  - **se** $\vartheta^* > 1/2$ **então** $\vartheta^* = 1/2$;
  - $\Delta_{k+1} = \vartheta^* \Delta_k$;
- **fim**
- **se** $\rho_k \geq 3/4$ *ou* $\lambda_c = 0$ **então** $\Delta_{k+1} = 2\|p_k\|$;

fim

Como estimativa inicial para o raio da região de confiança, vamos assumir:

$$\Delta_0 = \frac{1}{10}\|J_0^T F_0\|, \tag{64}$$

visto que esta opção nos chamou a atenção por ser numericamente razoável e apresentar eficiência satisfatória para problemas bem escalados.

Para finalizar o algoritmo, vamos analisar suas condições de parada no Passo 9.

4.4.1 Critérios de parada para o algoritmo

Os critérios de parada aqui visam detectar se o método de Levenberg-Marquardt convergiu para a solução do problema (1). Neste algoritmo, vamos utilizar cinco condições de parada:

1. *O algoritmo convergiu para a solução do problema, dada certa tolerância* `FTOL`.

Neste caso, vamos avaliar se o algoritmo convergiu para a solução do problema original, $\|F(x)\| = 0$, condição válida apenas para problemas zero-residuais:

$$\|F(x_k)\| \leq \texttt{FTOL}.$$

2. *A condição de otimalidade de primeira ordem para o problema de minimização irrestrita é satisfeita, dada certa tolerância* `GRADTOL`.

   Para esta condição, analisamos se o gradiente relativo da função (como sugerido em [2, p. 160]) no ponto está próximo de zero (condição de otimalidade de primeira ordem para minimização irrestrita):

$$\left| \frac{x_k^T (J(x_k)^T F(x_k))}{\max\{\|F(x_k)\|, 1\}} \right| \leq \texttt{GRADTOL}.$$

3. *Houve pequena variação no valor de $F(x)$, dada certa tolerância* `VARTOL`.

   Caso a variação no valor de $\|F(x)\|$ seja muito pequena, dada certa tolerância `VARTOL`, interrompe-se a execução do algoritmo e assume-se como solução x_k:

$$\|F(x_{k+1}) - F(x_k)\| \leq \texttt{VARTOL}.$$

4. *Houve pequena variação no tamanho do passo x_k, dada certa tolerância* `VARTOL`.

   Caso a variação no valor de $\|x\|$ seja muito pequena, dada certa tolerância `VARTOL`, interrompemos a execução do algoritmo e assumimos como solução o iterando atual:

$$\|x_{k+1} - x_k\| \leq \texttt{VARTOL}.$$

5. *Número de iterações do algoritmo.*

   O usuário estabelece um número `ITOL` limite de iterações para que o algoritmo convirja à solução, senão decretamos falha na convergência do algoritmo e interrompemos a execução, retornando o último iterando obtido, x_k:

$$k \leq \texttt{ITOL}.$$

5 Experimentos numéricos

Baseados no conteúdo desenvolvido neste trabalho, implementamos um programa para o *CAS Maxima* e rodamos alguns problemas em um microcomputador Intel Dual-Core T3400 2.16 GHz, 2 GB de memória RAM, sistema operacional Windows 7 32-bits e a versão 5.22.1 do CAS Maxima.

O *Maxima* é um Sistema Algébrico Computacional (CAS) implementado em *Lisp*. Seu nome é oriundo do nome original do projeto, MACSYMA, desenvolvido no *Massachusets Institute of Technology* (MIT) nos anos de 1968 a 1982 como parte do projeto MAC. MACSYMA é um acrônimo para MAC's SYmbolic MAnipulation System. O próprio sufixo MAC também é

um acrônimo, geralmente citado como MAn and Computer (Homem e Computador) ou Machine Aided Cognition (Conhecimento por Computador). Em 1998, o código fonte do projeto MACSYMA foi lançado sob a licença GNU, e no ano 2000 foi dado continuidade ao projeto na comunidade *SourceForge*, sob o nome Maxima, que até hoje o mantém e desenvolve.

O programa desenvolvido neste projeto possui três funções que são o coração da álgebra linear numérica da estratégia e concentram em si as informações mais importantes para a análise dos experimentos dessa seção. São elas:

- Uma função para calcular a fatoração QR de uma matriz por reflexões de Householder (ver Apêndice A); chamaremos essa função de `HHSTEP`,
- Uma função para fatorar a matriz em (36) por rotações de Givens; chamaremos essa função de `GIVENS`,
- Uma função para calcular as rotações de Givens propriamente ditas sobre um vetor (ver Apêndice B); chamaremos essa função de `GIVSTEP`.

Para todos os testes, tomamos por referência os problemas propostos por Moré, Garbow e Hillstrom [6].

Os testes foram realizados em 4 partes:

I. Selecionar alguns problemas e, partindo do ponto inicial proposto, rodar o problema no algoritmo desenvolvido.

II. Estabelecer uma caixa (em torno da solução), sortear k pontos dentro desta caixa e executar o algoritmo tomando como ponto inicial o sorteado.

III. Comparar o resultado da execução de alguns problemas com os resultados em [6] para o caso de quadrados mínimos.

IV. Comparar os resultados de alguns problemas com os resultados obtidos usando o algoritmo `lsqnonlin` do MATLAB.

Para os testes, vamos considerar os resultados apresentados na Tabela 1.

Como critérios de parada, adotamos aqueles discutidos na Seção 4.4.1, com `FTOL` = `GRADTOL` = 10^{-6}, `VARTOL` = 10^{-12} e um limitante para o número de iterações `ITOL` = 200.

5.1 Parte I

Nesta parte, analisamos a convergência do algoritmo à solução do problema para alguns problemas propostos por Moré, Garbow e Hillstrom em [6]; a numeração da função é a mesma usada nesse artigo e, para cada função apresentamos os resultados **CP**, $\|F_*\|$, **IT**, **AF**, **AJ**, **AH**, **AG1**, **AG2** e **TE** definidos na Tabela 1.

Os problemas testados são apresentados na Tabela 2, em que criamos siglas mnemônicas para facilitar a referência ao nome do problema, acompanhadas de suas respectivas dimensões m e n, bem como dos pontos iniciais utilizados. Os resultados desta parte são apresentados na Tabela 3.

Sigla	Descrição
AF	Avaliações da função efetuadas.
AG1	Chamadas feitas à função `GIVENS`.
AG2	Chamadas feitas à função `GIVSTEP`.
AH	Chamadas feitas à função `HHSTEP`.
AJ	Avaliações da jacobiana efetuadas.
CP	O critério de parada, um número conforme apresentado na Seção 4.4.1.
$\|F_*\|$	A norma euclidiana da função residual F avaliada na solução.
IT	A quantidade total de iterações do processo.
ITM	A quantidade média de iterações do processo.
TE	Tempo total de execução (em segundos).
TM	Tempo médio de execução (em segundos).

Tabela 1: Descrição dos itens considerados nos testes.

Função	Sigla	n	m	Ponto Inicial
1	ROS	2	2	$(-1.2, 1)$
4	BBS	2	3	$(1, 1)$
5	BEA	2	3	$(1, 1)$
13	POWS	4	4	$(3, -1, 0, 1)$
14	WOOD	4	6	$(-3, -1, -3, -1)$
16	BDF	4	20	$(25, 5, -5, -1)$
20	WAT	12	31	$(0, 0, \ldots, 0)$
34	LP1Z	6	40	$(1, 1, \ldots, 1)$

Tabela 2: Problemas testados na primeira parte.

Função	CP	$\|F_*\|$	IT	AF	AJ	AH	AG1	AG2	TE
ROS	1	0	18	19	14	18	33	99	0.16
BBS	1	0.26556 E-12	13	14	12	26	18	54	0.11
BEA	1	0.62509 E-11	5	6	6	10	51	133	0.16
POWS	2	0.19361 E-03	8	9	9	24	0	0	0.06
WOOD	1	0.17266 E-08	67	68	63	268	46	460	0.91
BDF	2	0.29295 E 03	40	41	24	160	87	870	1.80
WAT	2	0.38213 E-04	4	5	5	48	2	156	0.75
LP1Z	2	0.33361 E 01	2	3	3	12	5	105	0.31

Tabela 3: Resultados dos testes da primeira parte.

Frente aos resultados, podemos ver que para todos os problemas testados tivemos a convergência a uma solução em tempo satisfatório. Além disso, podemos destacar algumas características interessantes:

1. Em todas as funções, os critérios de parada foram completamente seguros: ou pelo zero

da função, para aqueles problemas zero-residuais, isto é, aqueles em que o valor da função avaliada na solução for menor que `FTOL`, ou pelo zero do gradiente, satisfazendo a condição necessária de primeira ordem para minimização sem restrições, para aqueles problemas que possuem resíduo não nulo.

2. Em algumas funções, o número de avaliações da jacobiana foi menor que o número de avaliações da função. Isso porque nem todos os pontos obtidos no processo foram satisfatórios. A jacobiana somente é avaliada para pontos candidatos $x_n = x_c + p^*$ tal que $F(x_n) < F(x_c)$, isto é, para pontos candidatos que são aceitos por provocar um decréscimo na função residual do problema.

3. Na Seção 4.1, discutimos a quantidade de rotações de Givens necessárias para ajustar a matriz em (37): $n(n+1)/2$ rotações, onde n é a dimensão do domínio da função residual $F(x)$ em (1). Em todos os resultados, reparamos que essa asserção foi satisfeita: para cada chamada da função `GIVENS` expressa em `AG1`, tivemos exatamente $n(n+1)/2$ chamadas feitas à função `GIVSTEP`, expressas em `AG2`.

4. Para cada iteração é feita uma decomposição QR da matriz jacobiana por reflexões de Householder. Pelos resultados, percebe-se que são feitas $n * \texttt{IT}$ chamadas à função `HHSTEP`, onde n é a dimensão do domínio da função residual $F(x)$ em (1). Isso significa que a reflexão foi feita para cada coluna da matriz jacobiana. Isso somente não acontece se a jacobiana for uma matriz quadrada, como podemos ver no caso dos problemas `ROS` e `POWS`: a reflexão feita na última coluna é sobre um elemento escalar e não sobre um vetor, com isso não há a necessidade de fazer a reflexão nem de chamar a função `GIVSTEP` para o último elemento, por isso temos $(n-1) * \texttt{IT}$ chamadas a esta função.

5. No problema `POWS`, não houve nenhuma chamada à função `GIVENS` e, portanto, nenhuma à função `GIVSTEP`. Isso acontece porque, em todas as iterações, a solução do subproblema de região de confiança foi um ponto interior do problema e, portanto, $\lambda^* = 0$ (ver Seção 3.2). Como $\lambda = 0$, a matriz em (37) já está sob a forma triangular, não havendo a necessidade de efetuar as rotações de Givens sobre a matriz em questão.

5.2 Parte II

Nesta parte, nosso objetivo é analisar como o algoritmo se comporta na resolução de alguns problemas a partir de outros pontos iniciais (em uma vizinhança da solução). Para isso, vamos tomar algumas das funções testadas na Seção 5.1, estabelecer uma caixa em torno da solução do problema, sortear k pontos nesta caixa e verificar a convergência média do algoritmo à solução. As denominações das funções serão iguais às da Tabela 2. Apresentamos na Tabela 4 os problemas considerados nesta seção, as dimensões, os valores para os limitantes da caixa que contém o ponto inicial e a quantidade de pontos considerada; os resultados, por sua vez, são apresentados na Tabela 5. Para cada resultado, consideramos **CP**, $\|F_*\|$, **IT**, **ITM**, **AF**, **AJ**, **TM** e **TE** definidos na Tabela 1.

Para efetuar o sorteio, usou-se uma função pseudorandômica do Maxima alimentada por uma semente. A semente escolhida para os testes foi 3820.

Destes testes podemos tirar algumas conclusões:

Função	n	m	Caixa para o ponto inicial	Qtde. k de pontos
ROS	2	2	$[-3,5] \times [-3,5]$	200
POWS	4	4	$[-3,3] \times [-3,3] \times [-3,3] \times [-3,3]$	200
WOOD	4	6	$[-1,3] \times [-1,3] \times [-1,3] \times [-1,3]$	100

Tabela 4: Problemas testados na segunda parte.

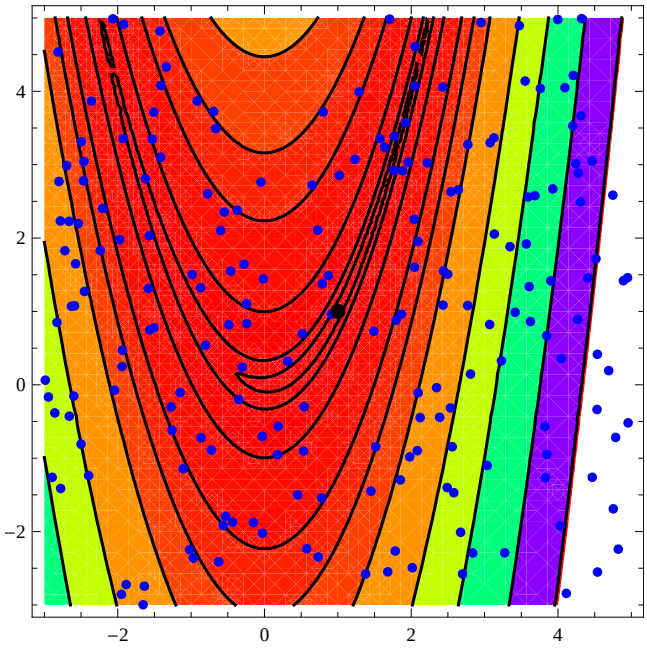


Figura 17: Ilustraçao do problema ROS considerado: mapa com as curvas de nível da função, pontos iniciais gerados e destaque para a solução do problema.

Função	CP		$\|F_*\|$	IT	ITM	AF	AJ	TM	TE
	C	NC							
ROS	1	200	1.2552 E-13	1547	7,735	1747	1284	0.038	7.73
POWS	2	200	3.7610 E-04	1565	7,825	1765	1765	0.040	8.10
WOOD	1	100	6.0755 E-08	919	9,19	1019	867	0.099	9.98

Tabela 5: Resultado dos testes da segunda parte.

1. O algoritmo convergiu para a solução de todos os problemas testados. Isso significa que para qualquer ponto próximo da solução o algoritmo é capaz de convergir à solução.

2. Os critérios de paradas foram homogêneos, isto é, coerentes com a norma euclidiana (média) da função objetivo avaliada nas soluções: para os problemas zero-residuais, em todas as iterações o algoritmo parou pelo critério 1, enquanto que para problemas que apresentam uma pequena norma do resíduo, o algoritmo parou pelo segundo critério (veja Seção 4.4.1).

3. Em um contexto comparativo com os testes da primeira parte, podemos observar que o tempo médio de execução foi melhor para todos os problemas, assim como a quantidade de avaliação das funções e da jacobiana; esse é um resultado esperado visto que todos os pontos testados estão próximos da solução.

5.3 Parte III

Nesta parte, nosso objetivo é comparar os nossos resultados com os resultados obtidos por Moré, Garbow e Hillstrom [6]. Para cada resultado, consideramos **CP**, $\|F_*\|$, **AF** e **AJ** definidos na Tabela 1.

Função	Sigla	n	m	Ponto Inicial
1	ROS	2	2	$(-1.2, 1)$
2	FRF	2	2	$(0.5, -2)$
6	JSF	2	10	$(0.3, 0.4)$
8	BARD	3	15	$(1, 1, 1)$
10	MEY	3	16	$(0.02, 4000, 250)$
12	BTD	3	10	$(0, 10, 20)$
13	POWS	4	4	$(3, -1, 0, 1)$
15	KOF	4	11	$(0.25, 0.39, 0.415, 0.39)$
16	BDF	4	20	$(25, 5, -5, -1)$
17	OS1	5	33	$(0.5, 1.5, -1, 0.01, 0.02)$
19	OS2	11	65	$(1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$
20	WAT	12	31	$(0, 0, \ldots, 0)$
27	BAL	10	10	$(0.5, 0.5, \ldots, 0.5)$
32	LPC	5	50	$(1, 1, \ldots, 1)$
33	LP1	5	50	$(1, 1, \ldots, 1)$
34	LP1Z	5	50	$(1, 1, \ldots, 1)$

Tabela 6: Problemas testados na terceira parte.

Nas Tabelas 6 e 7 apresentamos, respectivamente, os problemas propostos testados e o quadro comparativo dos resultados. Disso podemos discorrer:

1. Todas as funções testadas com o algoritmo `LMA` portaram-se bem na prática, convergindo a resultados semelhantes, quando não iguais, aos das rotinas `NLSQ1` e `NLSQ2`.

2. Todos os testes tiveram uma parada satisfatória, isto é, ou pela norma da função (para problemas zero residuais) ou pela norma do gradiente, condição de otimalidade de primeira ordem para minimização irrestrita. Isto aconteceu principalmente pela escolha da tolerância para norma da função e do gradiente, que foi 10^{-6}. Caso essa tolerância seja menor, algumas funções podem parar por falta de progresso, isto é, variação mínima dos pontos ou norma da função entre duas iterações (dada pela tolerância `VARTOL` $= 10^{-12}$). A tolerância, portanto, foi definida experimentalmente, e se comportou muito bem na maioria dos problemas.

Função	CP	LMA			NLSQ1			NLSQ2		
		$\|F_*\|$	AF	AJ	$\|F_*\|$	AF	AJ	$\|F_*\|$	AF	AJ
ROS	1	0.0	19	14	0.0	18	14	0.0	18	14
FRF	2	0.69988 E 01	41	28	0.69988 E 01	17	10	0.69988 E 01	17	15
JSF	2	0.11151 E 02	20	9	0.11151 E 02	25	14	0.11151 E 02	17	9
BARD	2	0.90635 E-01	6	6	0.90635 E-01	7	6	0.90635 E-01	7	6
MEY	2	0.93779 E 01	144	124	0.93779 E 01	136	120	0.93779 E 01	174	133
BTD	1	0.62754 E-10	13	12	0.72111 E-16	8	7	0.18041 E-15	7	6
POWS	2	0.19361 E-03	9	9	0.95234 E-35	68	62	0.72126 E-12	23	22
KOF	2	0.17536 E-01	13	11	0.17535 E-01	23	21	0.17535 E-01	18	15
BDF	2	0.29295 E 03	41	24	0.29295 E 03	315	282	0.29295 E 03	377	325
OS1	2	0.73924 E-02	19	16	0.73924 E-02	19	16	0.73924 E-02	167	117
OS2	2	0.20034 E 00	15	14	0.20034 E 00	18	14	0.20034 E 00	15	13
WAT	2	0.38213 E-04	5	5	0.21731 E-04	10	9	0.21731 E-04	7	6
BAL	1	0.19146 E-09	15	11	0.89874 E-15	17	15	0.16064 E-12	15	9
LPC	2	0.67082 E 01	5	5	0.67082 E 01	3	2	0.67082 E 01	3	2
LP1	2	0.34826 E 01	3	3	0.34826 E 01	3	2	0.34826 E 01	11	10
LP1Z	2	0.36917 E 01	2	2	0.36917 E 01	3	2	0.36917 E 01	13	12

Tabela 7: Quadro comparativo com os resultados das rotinas NLSQ1 e NLSQ2 de [6].

3. A função `POWS` apresentou um resultado com diferença considerável aos resultados de `NLSQ1` e `NLSQ2`. De fato, até entre as rotinas houve uma diferença significativa. O resultado de `NLSQ1` permite afirmar que o problema é zero residual, enquanto os resultados de `NLSQ2` e `LMA` apenas permitem afirmar que o problema possui resíduo de norma pequena. Comparando, ainda, nosso resultado ao resultado da rotina `NLSQ1`, a diferença é dada pelos valores considerados como tolerâncias. A parada aconteceu pela norma do gradiente da função no ponto, que é da ordem de 10^{-6}. Se diminuirmos essa tolerância, conseguimos alcançar no máximo um resíduo com norma de 10^{-5}, tendo como critério de parada a variação dos pontos, da ordem de 10^{-12}.

4. A função `WAT` apresentou resíduo com norma ligeiramente maior que os obtidos pelas rotinas `NLSQ1` e `NLSQ2`. Isso deve-se à tolerância do gradiente. Se baixarmos esse valor, obtemos ao final um resíduo com norma semelhante, mas com avaliações de função e jacobiana semelhantes.

5.4 Parte IV

Nesta parte, nosso objetivo é comparar nossos resultados com os resultados obtidos com a execução do algoritmo `lsqnonlin` do MATLAB. Os testes foram rodados usando o MATLAB 7.0.1.24704 (R14). Novamente, selecionamos algumas funções a serem testadas, conforme Tabela 8. Os resultados são apresentados na Tabela 9. Como critérios de comparação, consideraremos apenas **CP**, $\|F_*\|$e **AF** definidos na Tabela 1.

Nesta seção, em especial, vamos considerar também o critério de parada dos resultados comparados. Nossos critérios de parada foram apresentados na Seção 4.4.1. Usando a mesma convenção dos nossos e procurando compatibilizar as tolerâncias da maneira mais coerente e comparável possível, denotamos os critérios de parada do algoritmo `lsqnonlin` por:

2 O algoritmo convergiu para um ponto estacionário do problema.
Neste item, avalia-se a derivada direcional ao longo da direção p e a norma infinito do gradiente da função objetivo considerada (comparando ao nosso problema (30), a função objetivo considerada em `lsqnonlin` é a mesma, mas sem o fator $1/2$ em sua expressão). Então:

$$2(J_c^T F_c)^T p < \text{TolFun e } \|2(J_c^T F_c)\|_\infty < 10(\text{TolFun} + \text{TolX}),$$

onde TolFun = TolX = 10^{-6}.

4 O tamanho do passo não é significativo, numa tolerância de 10^{-6}.
Este critério corresponde ao nosso quarto critério exceto pela norma infinito:

$$\|p_k\|_\infty < 10^{-6}.$$

Esse critério é menos rigoroso que o nosso, pois em nosso caso `VARTOL` = 10^{-12}.

5 O número máximo de iterações foi atingido, no caso, 200 iterações, ou o número máximo de avaliação de função foi atingido, no caso, $100n$.

Função	Sigla	n	m	Ponto Inicial
1	ROS	2	2	$(-1.2, 1)$
2	FRF	2	2	$(0.5, -2)$
4	BBS	2	3	$(1, 1)$
5	BEA	2	3	$(1, 1)$
6	JSF	2	10	$(0.3, 0.4)$
8	BARD	3	15	$(1, 1, 1)$
10	MEY	3	16	$(0.02, 4000, 250)$
12	BTD	3	10	$(0, 10, 20)$
13	POWS	4	4	$(3, -1, 0, 1)$
14	WOOD	4	6	$(-3, -1, -3, -1)$
15	KOF	4	11	$(0.25, 0.39, 0.415, 0.39)$
16	BDF	4	20	$(25, 5, -5, -1)$
17	OS1	5	33	$(0.5, 1.5, -1, 0.01, 0.02)$
19	OS2	11	65	$(1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$
32	LPC	5	50	$(1, 1, \ldots, 1)$
33	LP1	5	50	$(1, 1, \ldots, 1)$
34	LP1Z	5	50	$(1, 1, \ldots, 1)$

Tabela 8: Problemas testados na quarta parte.

Desses testes, vale a pena destacar:

1. O algoritmo `LSQNONLIN` trabalha em dois modos: o de *otimização de grande porte*, onde é usado um método de região de confiança baseado no método de Newton, e o de *otimização de médio porte*, onde é usada a estratégia de Levenberg-Marquardt com busca linear. A nossa escolha foi pela otimização de médio porte, para que pudéssemos comparar a estratégia de Levenberg-Marquardt.

Função	LMA			LSQNONLIN		
	CP	$\|F_*\|$	AF	CP	$\|F_*\|$	AF
ROS	2	0.0	19	2	0.28247 E-08	85
FRF	2	0.69988 E 01	41	2	0.69989 E 01	68
BBS	2	0.26556 E-12	14	4	0.44409 E-15	186
BEA	2	0.62509 E-11	06	2	0.79365 E-08	49
JSF	2	0.11151 E 02	20	4	0.11151 E 02	25
BARD	2	0.90635 E-01	06	2	0.90635 E-01	51
MEY	2	0.93779 E 01	144	5	0.25156 E 03	302
BTD	2	0.62754 E-10	13	2	0.27364 E-06	42
POWS	2	0.19361 E-03	09	2	0.10241 E-05	86
WOOD	2	0.17266 E-08	68	2	0.45356 E-09	158
KOF	2	0.17536 E-01	13	2	0.17536 E-01	26
BDF	2	0.29295 E 03	41	4	0.29295 E 03	155
OS1	2	0.73924 E-02	19	2	0.73924 E-02	89
OS2	2	0.20034 E 00	15	2	0.20034 E 00	67
LPC	2	0.67082 E 01	05	2	0.67082 E 01	17
LP1	2	0.34826 E 01	03	4	0.34826 E 01	10
LP1Z	2	0.36917 E 01	02	4	0.36917 E 01	10

Tabela 9: Resultados dos testes da quarta parte.

2. Os critérios de parada foram definidos de maneira equivalente. Repare que nesses testes suprimimos nosso critério de parada 1 (veja Seção 4.4.1) pois o algoritmo `lsqnonlin` não implementa este critério. Mas tal critério é bom na prática, visto que para problemas que possuem norma do resíduo pequena este critério pode agir primeiro que a condição de otimalidade de primeira ordem (critério 2), caso as tolerâncias sejam diferentes, e fazer com que o algoritmo declare convergência mais rapidamente. O critério 1 apareceria para as funções `ROS`, `BBS`, `BEA` e `WOOD` (veja Parte I, Seção 5.1) e `BTD` (veja Parte III, Seção 5.3).

3. Algumas funções (`BBS`, `BEA`, `BTD`, `POWS` e `WOOD`) apresentaram diferença no valor da norma do resíduo. Ora `LMA` teve menor valor, ora `LSQNONLIN`. Mas essas diferenças não foram maiores que 10^3, e todas aconteceram em problemas com pequenos resíduos.

4. O algoritmo `LSQNONLIN` não convergiu para uma solução da função `MEY`, enquanto o nosso algoritmo obteve convergência em tempo satisfatório para uma solução do problema, que é a solução dada em [6].

5. Pelas comparações podemos concluir que nosso algoritmo está validado e que a implementação usando regiões de confiança é muito mais barata que usar o método de Newton, bem robusta do ponto de vista do custo-benefício e se comporta melhor na prática que a implementação do MATLAB usando Levenberg-Marquardt com busca linear.

6 Conclusões

Os problemas de quadrados mínimos expressos em (1) possuem várias aplicações no campo da otimização, e existem muitas propostas de resolução para este problema. Sem dúvidas, o método mais clássico e eficiente, que serve como base para todos os demais estudos, é o método de Newton. Entretanto esse método possui a desvantagem de ser muito custoso por exigir a avaliação das informações de segunda ordem da função objetivo expostas em (3).

A estratégia de Levenberg-Marquardt é uma proposta atraente para a resolução do problema (1) justamente por ser uma opção barata se comparada ao método de Newton, pois substitui o cálculo das informações de segunda ordem por um "regularizador" ou "modificador" λ ligado intimamente ao problema. Entretanto, por estimar tais informações, este método pode ser localmente lento se o problema for extremamente não linear ou se a função $F(x)$ em (1) possuir norma muito grande na solução. Não obstante, muitas implementações desta estratégia portam-se muito bem na prática, e o algoritmo pode obter até convergência q-quadrática à solução sob algumas hipóteses (veja, por exemplo, Teorema 10.2.6 em [2, p. 228]).

A abordagem do método de Levenberg-Marquardt do ponto de vista das técnicas de região de confiança foi proposta por Moré em [5], e foi nossa base para este trabalho. A ligação com tais técnicas está na maneira de calcular o modificador λ supracitado, o que agrega robustez e eficiência ao algoritmo, pois aproveita ao máximo a estrutura do problema em questão.

A ilustração do problema (1) como um sistema de equações não lineares nos serviu como parâmetro para pensar e desenvolver os conceitos apresentados neste texto, enriquecendo assim a análise encaminhada por Moré. Na prática, o algoritmo implementado comportou-se de maneira satisfatória, obtendo convergência à solução dos problemas propostos que consta na literatura.

Apêndice

A Transformações de Householder

Para executar as transformações de Householder, usamos elementos de [9, cap. 4]. Vamos começar com a definição de transformação de Householder:

Definição 1. *Uma transformação de Householder, também conhecida como Transformação Elementar, é uma matriz da forma:*

$$H = I - \kappa u u^T, \tag{65}$$

onde $\kappa\|u\|^2 = 2$. *Em nosso caso, vamos assumir* $\kappa = 1$ *e* $\|u\| = \sqrt{2}$.

Note que H é simétrica e que

$$\begin{aligned} H^T H = (I - uu^T)(I - uu^T) &= I - uu^T - uu^T + uu^T uu^T \\ &= I - 2uu^T + 2uu^T = I, \end{aligned}$$

portanto, a transformação de Householder é uma transformação ortogonal.

Aplicada a um vetor $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, o objetivo da transformação é gerar um vetor $\hat{x} = (\|x\|, 0, \ldots, 0) \in \mathbb{R}^n$; portanto, as transformações de Householder em nosso caso serão usadas para fatorar uma matriz X. E, ainda, o produto de todas as matrizes de Householder Q_H é uma matriz simétrica e ortogonal, portanto a resultante de uma sequência de transformações elementares no dá a fatoração QR da matriz X.

Uma transformação de Householder de ordem n pode ser armazenada em um vetor $u \in \mathbb{R}^n$ e, sua aplicação em uma matriz X se daria da seguinte forma:

$$HX = (I - uu^T)X = X - u(u^T X).$$

Daí:

1. $v^T = u^T X$;
2. $X = X - uv^T$.

Portanto, neste caso, vamos armazenar apenas o vetor u.

Dada uma matriz $X \in \mathbb{R}^{m\times n}$, o procedimento para calcular a transformação segue de duas instâncias:

1. Se $m \leq n$, o algoritmo percorre todas as linhas i da matriz, começando da primeira linha; para cada linha i a transformação elementar é gerada a partir do vetor $\gamma = (x_{i,i}, x_{i,i+1}, \ldots, x_{i,n})$, será de ordem m, aplicada à matriz X. O procedimento repete-se m vezes, e o resultado será o seguinte:

   $$X = Q_H \left(R_{m,m}\ R_{m,m+1}\right),$$

   onde $Q_H \in \mathbb{R}^{m\times m}$ é a resultante das transformações elementares e $R_{m,m} \in \mathbb{R}^{m\times m}$ é triangular superior.

2. Se $m > n$, o algoritmo percorre todas as colunas j da matriz, começando da primeira coluna; para cada coluna j a transformação elementar é gerada a partir do vetor $\gamma = (x_{j,j}, x_{j+1,j}, \ldots, x_{m,j})$ e será de ordem m aplicada à matriz X. O procedimento repete-se n vezes, e o resultado será o seguinte:

$$X = Q_H \begin{pmatrix} R_{n,n} \\ 0 \end{pmatrix},$$

onde $Q_H \in \mathbb{R}^{m\times m}$ é a resultante das transformações elementares e $R_{n,n} \in \mathbb{R}^{n\times n}$ é triangular superior.

Para gerar a transformação elementar, propriamente o vetor gerador u, aplicamos o seguinte algoritmo, extraído de [9, p. 257]:

Algoritmo 5: Geração da Transformação de Householder

Dado um vetor x ao qual será aplicada a transformação, faça:

HouseholderStep(x);

$u = x$;
$\nu = \|u\|$;
se $\nu = 0$ **então** $u[1] = \sqrt{2}$;
senão
    $u = x/\nu$;
    se $u[1] \geq 0$ **então**
        $u[1] = u[1] + 1$;
        $\nu = -\nu$;
    senão
        $u[1] = u[1] - 1$;
    fim
    $u = u/\sqrt{|\, u[1]\, |}$;
fim
retorna u.

B Transformações de Givens

O intuito do uso das Transformações de Givens é o mesmo das Transformações de Householder abordadas no Apêndice A. Entretanto, muitas vezes, a matriz a ser fatorada possui uma estrutura especial de forma que o método de Householder não seria o método mais eficiente a ser aplicado. Nesta seção, vamos usar elementos de [9, cap. 4].

Primeiro, vamos definir o que são Transformações de Givens.

Definição 2. *Transformações de Givens são matrizes da forma:*

$$G = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \tag{66}$$

onde $c^2 + s^2 = 1$.

A matriz G apresentada na Definição 2 é uma matriz ortogonal que representa uma rotação em um plano conveniente, daí a nomenclatura 'Rotações de Givens'. A relação $c^2 + s^2 = 1$ apresentada sugere a existência um ângulo θ, único, tal que:

$$c = \cos\theta \text{ e } s = \operatorname{sen}\theta.$$

Portanto, a aplicação sobre um vetor (a, b) qualquer corresponde à rotação, no sentido horário, do vetor (a, b) de um ângulo θ.

Entretanto, em nosso caso, as rotações de Givens nos serão úteis para inserir zeros em uma matriz. Portanto, ao aplicar a rotação sobre um vetor (a, b) qualquer, nosso objetivo é obter o seguinte resultado:

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} r \\ 0 \end{pmatrix}. \tag{67}$$

Logo, não é necessário nos preocuparmos com o ângulo θ neste caso especial, mas sim na resolução do seguinte sistema:

$$\begin{cases} ca + sb &= r \\ cb - sa &= 0 \end{cases},$$

que nos dá:

$$c = \frac{a}{r}, s = \frac{b}{r} \; e \; r = \sqrt{a^2 + b^2}. \tag{68}$$

Por consequência, o vetor resultante da transformação (67) será:

$$\begin{pmatrix} \sqrt{a^2 + b^2} \\ 0 \end{pmatrix}.$$

De forma geral, a transformação de Givens pode ser vista como uma matriz da forma:

$$G = \begin{pmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & c & \cdots & s & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & -s & \cdots & c & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{pmatrix}, \tag{69}$$

dita *rotação no plano* $(\imath, \jmath)$, onde G nada mais é que uma matriz identidade de ordem m com as seguintes substituições:

$$\begin{aligned} g_{\imath,\imath} &= c \quad g_{\imath,\jmath} = s \\ g_{\jmath,\imath} &= -s \quad g_{\jmath,\jmath} = c. \end{aligned}$$

Essa matriz pode então ser construída sobre qualquer vetor $(a, \ldots, b) \in \mathbb{R}^p$, contido em uma matriz qualquer $X \in \mathbb{R}^{m \times n}$ e, da mesma forma que em (68), o vetor resultante será $(\sqrt{a^2 + b^2}, \ldots, 0)$ tal que somente a última coordenada se anula, e sua aplicação pode ser feita na matriz $X = GX$.

Algoritmo 6: Geração da transformação de Givens

Dados o vetor $v = (a, \ldots, b) \in \mathbb{R}^p$, extraído da matriz B em (70), faça:

$a = v[1]$;
$b = v[p]$;
$\upsilon =\mid a \mid + \mid b \mid$;
se $\upsilon = 0$ **então**
    $c = 1$;
    $s = 0$;
senão
    se $a = 0$ **então**
        $c = 0$;
        $s = 1$;
    senão
        $r = \upsilon * \sqrt{(a/\upsilon)^2 + (b/\upsilon)^2}$;
        $c = a/r$;
        $s = b/r$;
    fim
fim
retorna $[c, s]$;

Neste trabalho, as rotações de Givens nos serão uma ferramenta eficiente para produzir a relação (37). A matriz

$$B = \begin{pmatrix} R_c \\ 0 \\ \sqrt{\lambda} I \end{pmatrix} \tag{70}$$

possui uma estrutura muito característica, como já apontado na Seção 4.1. A ideia é, por meio de $n(n+1)/2$ rotações de Givens, eliminar todos os termos da diagonal da submatriz $D_\lambda = \sqrt{\lambda} I \in \mathbb{R}^{n \times n}$ e, para cada elemento[1] da diagonal $d_{\imath,\imath}$ ao qual a rotação é aplicada, eliminar os termos gerados $d_{\imath,\jmath}$, para $\jmath = \imath + 1, \ldots, n$.

A cada iteração, o vetor $(a, 0, \ldots, 0, b)$ gerado será tal que $a = r_{\jmath,\jmath}$ e $b = d_{\imath,\jmath}$, e a sua estrutura especial deve-se justamente porque este vetor é tal que todas as coordenadas entre os elementos a e b são nulas, por isso basta zerar o termo b.

No Algoritmo 6, adaptado de [9, p. 272], os fatores c e s são gerados a partir dos valores a e b do vetor dado; a matriz da transformação será de ordem m da mesma forma da matriz em (69). Essa ordem m vem da ordem da matriz $B \in \mathbb{R}^{m \times n}$ em (70).

O termo υ no Algoritmo 6 é usado para evitar *overflows* e fazer com que *underflows*, caso aconteçam, se manifestem da maneira mais inofensiva possível. Para mais detalhes, ver [9, p. 139].

[1] Aqui, assuma que $\imath$ corresponde a linhas da matriz e $\jmath$ a colunas da matriz.

C Resultados computacionais detalhados

Neste apêndice, apresentaremos os resultados computacionais detalhados dos testes realizados na segunda parte (Seção 5.2). Nosso objetivo aqui é mostrar como o algoritmo se comportou do ponto de vista computacional, isto é, quanto cada função consumiu do tempo total de execução, destacar quais funções são o "coração" do algoritmo e visualizar o algoritmo como uma composição de algumas rotinas. Começaremos dando uma breve explicação de cada rotina implementada no Maxima, depois apresentaremos os tempos de execução detalhados de cada um dos testes.

Na implementação computacional da estratégia estudada neste trabalho, foram escritas as seguintes rotinas:

- `LMA` – esta é a função principal. É responsável por controlar o descréscimo na função objetivo, o tamanho do raio da região de confiança e a convergência do algoritmo à solução do problema (análise dos critérios de parada). Seus parâmetros de entrada são essecialmente a função residual $F(x)$ em (1) e o ponto inicial x_0. Os retornos são o critério de parada, a solução encontrada x^* e a norma do vetor $F(x^*)$.

- `EXACTTR` – esta rotina é responsável por encontrar o zero da equação secular (28). É invocada a cada iteração da rotina principal, isto é, para cada ponto x_c do processo. A partir da função e da jacobiana avaliadas no ponto corrente, de uma estimativa inicial λ_0 para a solução, o tamanho do raio da região de confiança Δ_c e a tolerância para o zero da função secular, a rotina devolve λ^*, solução do problema, e p^*, o passo calculado em função de λ^*.

- `TDOWNSYS` – é uma rotina para resolução de sistemas triangulares inferiores. Dados a matriz de coeficientes (triangular inferior) do sistema e o vetor dos termos independentes, a rotina retorna a solução do sistema em um vetor ou um vetor vazio para sistemas impossíveis.

- `TUPSYS` – é a mesma coisa que a rotina `TDOWNSYS`, entretanto para sistemas triangulares superiores.

- `HOUSEHOLDER` – função responsável em calcular a fatoração QR de uma dada matriz $M \in \mathbb{R}^{m\times n}$. É dividida em três casos: `HOUSEHOLDER1`, `HOUSEHOLDER2` e `HOUSEHOLDER3` para os casos em que $m < n$, $m > n$ e $m = n$, respectivamente.

- `HHSTEP` – é a rotina em que encontra-se o vetor u para formar a matriz de Householder H. É a implementação do Algoritmo 5 do Apêndice A.

- `GIVENS` – é a função responsável por corrigir a matriz em (37) por Rotações de Givens.

- `GIVENSSTEP` – é a rotina responsável por gerar os elementos c e s da matriz de Givens. É a implementação do Algoritmo 6 do Apêndice B.

Apresentamos os resultados na Tabela 10. Também ilustramos os tempos na Figura 18. Vale destacar os seguintes pontos:

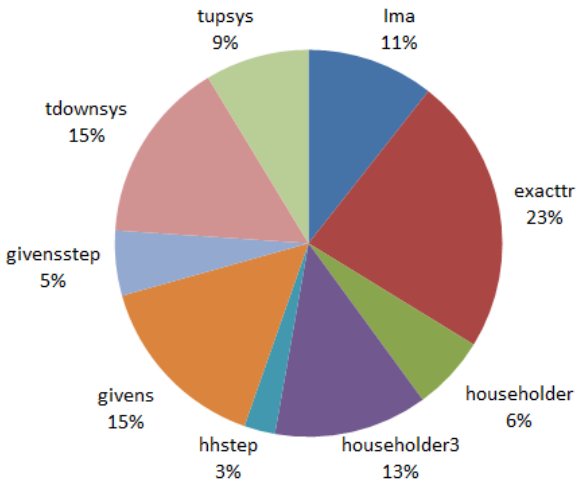
- Os experimentos foram executados em um microcomputador como descrito na Seção 5, portanto, para efeito de reprodutibilidade dos experimentos, os tempos registrados podem variar conforme o computador em que os problemas forem executados, a versão do Maxima utilizada ou ainda até mesmo a ociosidade do processador no momento do teste.

- Podemos reparar que a rotina que consumiu mais tempo de execução foram as transformações de Givens. Mas repare também que a porção de tempo consumido pelas transformações de Householder não foi muito menor. Na Seção 4.1 defendemos a vantagem das rotações de Givens frente ao recálculo da fatoração QR da matriz $J_c^T J_c + \lambda I$. Os resultados confirmam esta asserção: se recalculássemos a fatoração QR a cada iteração da rotina `EXACTTR` em que o valor de λ varia, com certeza o tempo total consumido seria muito maior.

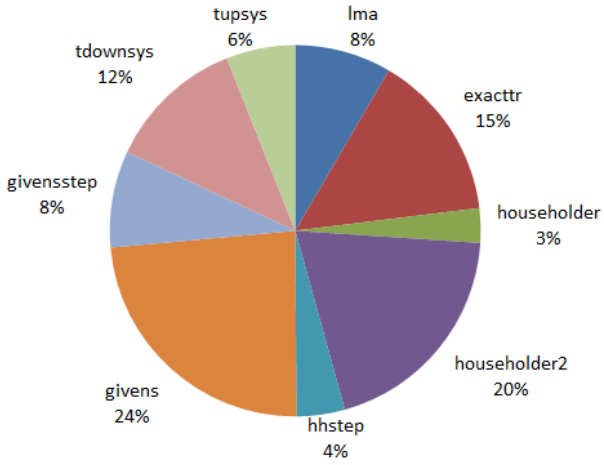
- A Função de Powell não exigiu nenhuma rotina para executar Transformações de Givens. Veja os resultados obtidos na Parte I, Seção 5.1 para mais detalhes.

Função	Rosenbrock		Wood		Powell	
Rotina	Chamadas	Tempos	Chamadas	Tempos	Chamadas	Tempos
lma	200	0.82 seg.	100	0.84 seg.	200	1.46 seg.
exacttr	1547	1.79 seg.	919	1.46 seg.	1565	1.07 seg.
householder	1547	0.48 seg.	919	0.30 seg.	1565	0.55 seg.
householder2			919	1.96 seg.		
householder3	1547	0.99 seg.			1565	2.53 seg.
hhstep	1547	0.20 seg.	3676	0.42 seg.	4695	0.73 seg.
givens	2080	1.18 seg.	979	2.36 seg.		
givenstep	6240	0.42 seg.	9790	0.84 seg.		
tdownsys	7254	1.18 seg.	3796	1.20 seg.	3130	1.17 seg.
tupsys	3627	0.67 seg.	1898	0.60 seg.	1565	0.59 seg.
AF	1747		1019		1765	
AJ	1284		867		1765	
TOTAL	7.73 seg.		9.98 seg.		8.10 seg.	
	CP: 1 - 200 vezes		CP: 1 - 100 vezes		CP: 2 - 200 vezes	
	Sol. Média: 0.12552 E-12		Sol. Média: 0.60755 E-7		Sol. Média: 0.37610 E-3	

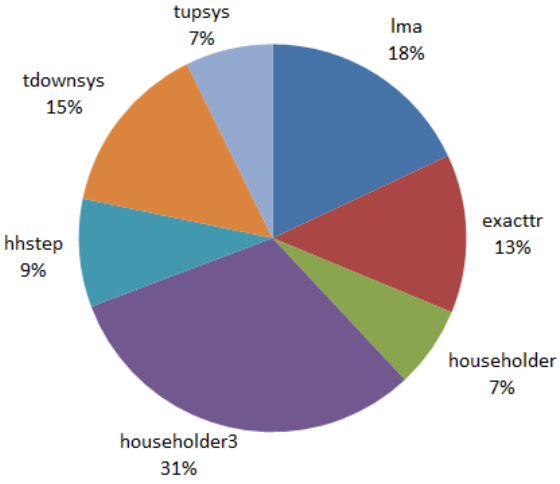
Tabela 10: Tempos de execução das funções testadas na segunda parte.



(a) Rosenbrock.



(b) Wood.



(c) Powell.

Figura 18: Gráficos dos tempos de execução relativos nos testes da segunda parte.

Referências

[1] A.R. Conn, N.I.M. Gould & P.L. Toint, *Trust-Region methods*, MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[2] J.E. Dennis Jr. & R.B. Schnabel, *Numerical methods for unconstrained optimization and nonlinear equations*, SIAM, Philadelphia, 1996.

[3] K. Levenberg, "A method for the solution of certain non-linear problems in least squares", *The Quarterly of Applied Mathematics 2*, 1944, pp. 164-168.

[4] D.W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters", *SIAM Journal on Applied Mathematics 11*, 1963, pp. 431-441.

[5] J.J. Moré, "The Levenberg-Marquardt algorithm: implementation and theory", em: G.A. Watson, *Lecture Notes in Mathematics 630: Numerical Analysis*, Springer-Verlag, New York, 1978, pp. 105-116.

[6] J.J. Moré, B.S. Garbow & K.E. Hillstrom, "Testing unconstrained optimization software", *ACM Transactions on Mathematical Software*, Volume 7, pp. 17-41, 1981.

[7] J.J. Moré & D.C. Sorensen, *Newton's Method*, Technical Report ANL-82-8, Argonne National Laboratory, Argonne, Illinois, 1982, 41p.

[8] J. Nocedal & S.J. Wright, *Numerical optimization*, Springer-Verlag, New York, 1999.

[9] G.W. Stewart, *Matrix algorithms, Volume I: Basic decompositions*, SIAM, Philadelphia, 1998.