

Rodrigo Maciel Rosa

*Estudo e Implementação do Método Dogleg para
Programação Não Linear*

Florianópolis

2005

Rodrigo Maciel Rosa

*Estudo e Implementação do Método Dogleg para
Programação Não Linear*

Trabalho de Conclusão de Curso
apresentado ao Curso de Matemática
Habilitação Bacharelado
Departamento de Matemática
Centro de Ciências Físicas e Matemáticas

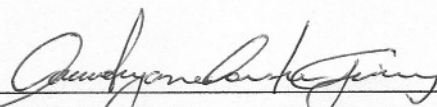
Orientador:
Clóvis Caesar Gonzaga

UNIVERSIDADE FEDERAL DE SANTA CATARINA

Florianópolis

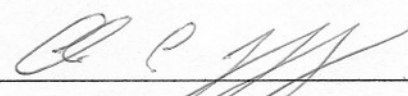
2005

Esta monografia foi julgada adequada como **TRABALHO DE CONCLUSÃO DE CURSO** no curso de Matemática – Habilitação Bacharelado e aprovada em sua forma final pela Banca Examinadora designada pela Portaria nº. 18/CCM/05.

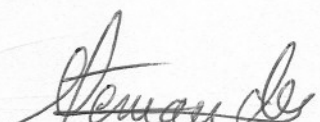


Prof.^a. Carmen Suzane Comitre Gimenez
Professora responsável pela disciplina

Banca examinadora:



Clóvis Caesar Gonzaga
Orientador



Márcio Rodolfo Fernandes



Mário César Zambaldi

Conteúdo

Introdução	p. 5
1 O Problema de Programação Não Linear Irrestrito	p. 6
2 Métodos de Otimização	p. 9
2.1 Método de Cauchy	p. 9
2.2 Método de Newton	p. 11
3 Método de Região de Confiança	p. 15
3.1 Introdução	p. 15
3.2 Um Exemplo	p. 17
3.3 O Algoritmo	p. 22
3.3.1 Análise do Algoritmo	p. 23
4 O Método Dogleg: um Subproblema de Região de Confiança	p. 24
4.1 O Ponto de Cauchy	p. 24
4.2 O Método	p. 26
4.3 O Algoritmo	p. 31
5 Testes Práticos	p. 32
5.1 Método de Barreiras	p. 32
5.1.1 Exemplos com a Função Barreira Logarítmica	p. 33
5.2 Função de Rosenbrock	p. 36
5.2.1 Comparação com Outros Métodos	p. 37

5.3	Outro Exemplo	p. 37
	Conclusão	p. 42
	Referências	p. 43
	Apêndice A - Programação em MATLAB	p. 44
A.1	Arquivo go.m	p. 44
A.2	Arquivo dados.m (exemplo)	p. 44
A.3	Arquivo trust.m	p. 45
A.4	Arquivo dogleg.m	p. 47
A.5	Arquivo figuras.m	p. 48
A.6	Arquivos de funções	p. 49
A.6.1	Arquivo pen.m	p. 49
A.6.2	Arquivo banana.m	p. 49
A.6.3	Arquivo exemplo.m	p. 50

Introdução

Neste trabalho, estaremos interessados em resolver problemas de programação não linear irrestritos. Os métodos de Cauchy e de Newton são métodos clássicos em otimização que buscam por minimizadores locais para estes problemas. O método de Cauchy é um método com convergência global porém muito lento. Por outro lado, o método de Newton é muito rápido quando partimos de um ponto próximo a um minimizador de f , no entanto, ele diverge facilmente. Como forma de unir o que há de melhor nestes dois métodos, surge o método de região de confiança, que busca resolver, em cada iteração, o problema de minimizar f em uma região “confiável”.

No presente trabalho, apresentaremos como um subproblema do método de região de confiança, o método dogleg, que busca uma solução aproximada do problema de minimizar um modelo quadrático da variação de f , em relação a um ponto dado, em uma certa região de confiança. O método dogleg trabalha bem com funções que tem a hessiana definida positiva; nos casos em que não temos isto, adotamos o passo de Cauchy.

No primeiro capítulo, apresentaremos o problema de programação não linear irrestrito e as condições necessárias e suficientes de otimalidade.

No segundo, apresentaremos o método de Newton e o método de Cauchy.

O terceiro capítulo trará o método de região de confiança. Uma seção destinar-se-á a apresentação do algoritmo para este método.

No quarto capítulo faremos um estudo do método dogleg onde, também, mostraremos o algoritmo.

No quinto, e último, capítulo exibiremos os resultados obtidos através das implementações do método de região de confiança e do método dogleg no software MATLAB. Os arquivos utilizados para tais implementações serão mostrados no apêndice A.

1 O Problema de Programação Não Linear Irrestrito

A programação não linear irrestrita diferenciável estuda problemas da forma:

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} f(x) \quad (1.1)$$

em que a função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ é continuamente diferenciável.

Estaremos interessados em obter dois tipos de solução para estes problemas:

- (i) $x^* \in \mathbb{R}^n$ tal que $f(x^*) \leq f(x), \forall x \in \mathbb{R}^n$;
- (ii) $x^* \in \mathbb{R}^n$ tal que $f(x^*) \leq f(x), \forall x \in V$, em que $V \subset \mathbb{R}^n$ é uma vizinhança de x^* .

A solução obtida em (i) é chamada *solução ótima* de (1.1) ou *minimizador global* de (1.1), e a obtida em (ii) é chamada *ótimo local* de (1.1) ou *minimizador local* de (1.1).

Teorema 1 (Condições Necessárias de Otimalidade de 1ª Ordem) *Se $x^* \in \mathbb{R}^n$ é um minimizador local de f e $f \in C^1$ em uma vizinhança aberta de x^* , então $\nabla f(x^*) = 0$.*

Prova Suponha que x^* é minimizador local de f . Seja $d \in \mathbb{R}^n$ uma direção arbitrária, $d \neq 0$. Temos, pela fórmula de Taylor, que para $\lambda \in \mathbb{R}$

$$f(x^* + \lambda d) = f(x^*) + \lambda \nabla f(x^*)^T d + o(x^*, \lambda)$$

$$\text{com } \lim_{\lambda \rightarrow 0} \frac{o(x^*, \lambda)}{\lambda} = 0.$$

Daí, para $\lambda > 0$,

$$\nabla f(x^*)^T d = \frac{f(x^* + \lambda d) - f(x^*)}{\lambda} - \frac{o(x^*, \lambda)}{\lambda}. \quad (1.2)$$

Como x^* é minimizador local de f , então $f(x^* + \lambda d) - f(x^*) \geq 0$ para λ suficientemente pequeno.

De (1.2), podemos escrever:

$$\begin{aligned}\lim_{\lambda \rightarrow 0^+} \nabla f(x^*)^T d &= \lim_{\lambda \rightarrow 0^+} \frac{f(x^* + \lambda d) - f(x^*)}{\lambda} - \lim_{\lambda \rightarrow 0^+} \frac{o(x^*, \lambda)}{\lambda} \\ \nabla f(x^*)^T d &= \lim_{\lambda \rightarrow 0^+} \frac{f(x^* + \lambda d) - f(x^*)}{\lambda} \geq 0\end{aligned}$$

Concluimos que

$$\nabla f(x^*)^T d \geq 0, \quad \forall d \in \mathbb{R}^n \quad (1.3)$$

Repetindo o argumento para $p = -d$, concluimos que $\nabla f(x^*)^T p \leq 0$, e assim, $\nabla f(x^*)^T d = 0, \forall d \in \mathbb{R}^n$.

Disso, temos que

$$\nabla f(x^*)^T e_i = 0, \text{ ou equivalentemente, } [\nabla f(x^*)]_i = 0, \forall i \in 1, 2, \dots, n$$

onde $e_i, i = 1, \dots, n$ são os vetores de base canônica de \mathbb{R}^n .

Portanto, $\nabla f(x^*) = 0$. □

Teorema 2 (Condições Necessárias de Otimalidade de 2ª Ordem) *Se x^* é um minimizador local de f e $\nabla^2 f$ é contínua em uma vizinhança aberta de x^* , então $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ é semi-definida positiva.*

Prova Suponha que x^* é minimizador local de f . Seja $d \in \mathbb{R}^n$ uma direção arbitrária, $d \neq 0$. Temos, pela fórmula de Taylor, que para $\lambda \in \mathbb{R}, \lambda > 0$

$$f(x^* + \lambda d) = f(x^*) + \lambda \nabla f(x^*)^T d + \frac{\lambda^2}{2} d^T \nabla^2 f(x^*) d + o_2(\lambda)$$

com $\lim_{\lambda \rightarrow 0^+} \frac{o_2(\lambda)}{\lambda^2} = 0$.

Pelo teorema 1, $\nabla f(x^*) = 0$.

Obtemos:

$$\begin{aligned}\frac{d^T \nabla^2 f(x^*) d}{2} &= \frac{f(x^* + \lambda d) - f(x^*)}{\lambda^2} - \frac{\lambda \nabla f(x^*)^T d}{\lambda^2} - \frac{o_2(\lambda)}{\lambda^2} \\ &= \frac{f(x^* + \lambda d) - f(x^*)}{\lambda^2} - \frac{o_2(\lambda)}{\lambda^2}\end{aligned} \quad (1.4)$$

Para λ suficientemente pequeno,

$$f(x^* + \lambda d) - f(x^*) \geq 0$$

pois x^* é minimizador local de f .

De (1.4), podemos escrever:

$$\begin{aligned} \lim_{\lambda \rightarrow 0^+} \frac{d^T \nabla^2 f(x^*) d}{2} &= \lim_{\lambda \rightarrow 0^+} \frac{f(x^* + \lambda d) - f(x^*)}{\lambda^2} - \lim_{\lambda \rightarrow 0^+} \frac{o_2(\lambda)}{\lambda^2} \\ &= \lim_{\lambda \rightarrow 0^+} \frac{f(x^* + \lambda d) - f(x^*)}{\lambda^2} \\ &\geq 0 \end{aligned}$$

Concluimos que $d^T \nabla^2 f(x^*) d \geq 0, \forall d \in \mathbb{R}^n$, o que é a definição de matriz semi-definida positiva. \square

Teorema 3 (Condições Suficientes de Otimalidade de 2ª Ordem) *Suponha que $x^* \in \mathbb{R}^n$, $\nabla^2 f$ é contínua em uma vizinhança aberta de x^* , $\nabla f(x^*) = 0$ e $\nabla^2 f(x^*)$ é definida positiva. Então x^* é um minimizador local de f .*

Prova Como $\nabla^2 f$ é contínua e definida positiva em x^* , existe um raio $r > 0$, tal que $\nabla^2 f(x)$ permanece definida positiva para todo x na bola aberta $\mathcal{D} = \{z; \|z - x^*\| < r\}$.

Seja $p \in \mathbb{R}^n$ tal que $0 < \|p\| < r$. Então $x^* + p \in \mathcal{D}$ e, assim, utilizando o teorema de Taylor, teremos:

$$\begin{aligned} f(x^* + p) &= f(x^*) + p^T \nabla f(x^*) + \frac{1}{2} p^T \nabla^2 f(x^* + tp) p \\ &= f(x^*) + \frac{1}{2} p^T \nabla^2 f(x^* + tp) p \end{aligned}$$

onde $t \in (0, 1)$.

Visto que $x^* + tp \in \mathcal{D}$, nós temos que $p^T \nabla^2 f(x^* + tp) p > 0$ e, portanto, $f(x^* + p) > f(x^*)$.

\square

2 Métodos de Otimização

Este capítulo apresenta dois métodos muito importantes na área de otimização. O objetivo destes métodos é resolver o problema (1.1).

2.1 Método de Cauchy

Suponha que $g \in \mathbb{R}^n$, $g \neq 0$, e que queiramos resolver o seguinte problema:

$$\underset{h \in \mathbb{R}^n}{\text{minimizar}} \quad g^T h \quad \text{s.a.} \quad \|h\| = 1. \quad (2.1)$$

Resolvemos este problema utilizando a desigualdade de Cauchy-Schwarz que nos diz que

$$g^T h \geq -\|g\| \|h\|$$

com a igualdade valendo quando g e h são colineares em sentidos opostos. Isto ocorre para

$$h = -\frac{g}{\|g\|}.$$

Assim, $h = -g/\|g\|$ é solução de (2.1).

Seja, agora, uma função $f : \mathbb{R}^n \rightarrow \mathbb{R}$ com f diferenciável em $\bar{x} \in \mathbb{R}^n$. Considere o modelo linear de f , $m_{\text{linear}} : \mathbb{R}^n \rightarrow \mathbb{R}$ em que $m_{\text{linear}}(h) = f(\bar{x}) + h^T \nabla f(\bar{x})$. Utilizando, novamente, a desigualdade de Cauchy-Schwarz, temos que

$$\bar{h} = -\frac{\nabla f(\bar{x})}{\|\nabla f(\bar{x})\|} \quad (2.2)$$

é solução do problema

$$\underset{h \in \mathbb{R}^n}{\text{minimizar}} \quad m_{\text{linear}}(h) \quad \text{s.a.} \quad \|h\| = 1.$$

O resultado obtido em (2.2) é conhecido como *direção de máximo declive* e está ilustrado na figura 1. Na figura, as linhas retas referem-se as curvas de nível do modelo linear de uma função,

cujas curvas de nível estão representadas pelas linhas tracejadas. Os valores, em destaque, revelam o sentido em que o modelo linear cresce, que é o mesmo sentido do $\nabla f(\bar{x})$, apresentado na figura pela seta mais grossa. A seta mais fina, com sentido oposto ao gradiente e norma 1, é a direção de máximo declive.

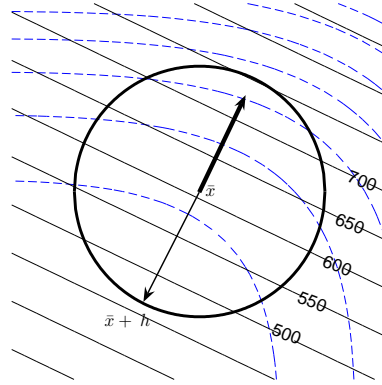


Figura 1: A direção de máximo declive, representada pela seta mais fina.

Agora que já conhecemos a direção de máximo declive podemos apresentar o método de Cauchy. O método de Cauchy é definido pelo algoritmo iterativo

$$x_{k+1} = x_k + \lambda_k h_k$$

em que h_k é a direção de máximo declive a partir de x_k , $-\nabla f(x_k)$, e λ_k é um escalar positivo. Existem métodos específicos para se calcular o valor de λ_k em cada iteração. Como este não é nosso objetivo, vamos nos limitar a dizer que um bom valor pra λ_k é

$$\lambda_k \in \arg \min_{\lambda \geq 0} \phi_k(\lambda), \quad \phi_k(\lambda) = f(x_k - \lambda \nabla f(x_k))$$

se existir minimizador.

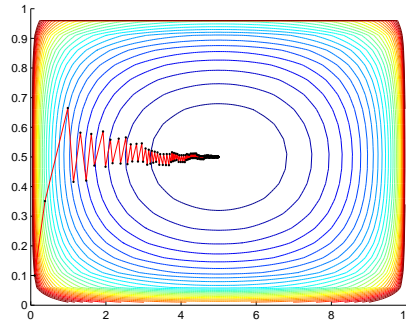
Estudos mais detalhados sobre o tamanho do passo λ_k podem ser obtidos em [1, p. 36–43].

Em outras palavras, este método, a partir de um x_k dado, procura ao longo da direção h_k um mínimo sobre esta reta, dado por x_{k+1} .

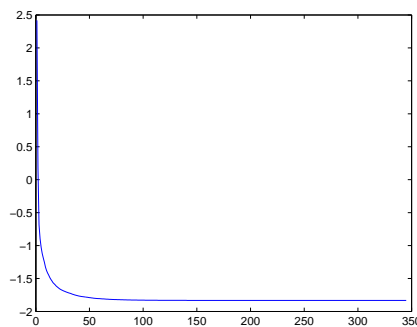
O método de Cauchy tem uma boa convergência global, ou seja, independe do ponto inicial para resolver (1.1), mas, por outro lado, ele é muito lento e apresenta uma convergência local ruim (linear).

A figura 2(a) apresenta as curvas de nível de uma função f e exemplifica os avanços do método de Cauchy para obtenção do minimizador desta f . Neste exemplo, foram necessários

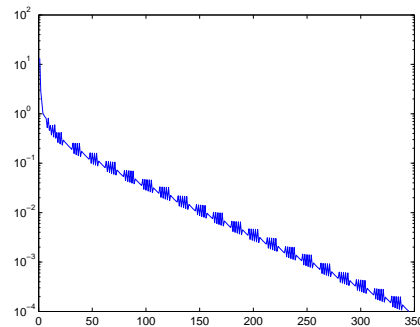
345 iterações para se obter uma precisão de 10^{-4} , tornando evidente como este é um método vagaroso. A figura 2(b) mostra como f decresce com as iteradas do método de Cauchy, confirmando a convergência global do método. Em 2(c), apresentamos a norma do gradiente de f em função das iteradas.



(a) Curvas de nível da função



(b) Valores da função



(c) Norma do gradiente da função

Figura 2: O método de Cauchy.

2.2 Método de Newton

No método de Cauchy, trabalhamos com o modelo linear da função. Mas, supondo que a nossa função seja duas vezes diferenciável, é natural pensarmos em modelo quadrático da variação da função em torno do ponto \bar{x} , ou seja, $m_q : \mathbb{R}^n \rightarrow \mathbb{R}$ definida por

$$m_q(h) = \nabla f(\bar{x})^T h + \frac{1}{2} h^T \nabla^2 f(\bar{x}) h. \quad (2.3)$$

Quando $\nabla^2 f(\bar{x})$ for definida positiva, teremos que o minimizador \bar{h} de m_q será a única solução de $\nabla m_q(h) = 0$. Disto, temos que

$$0 = \nabla m_q(\bar{h}) = \nabla f(\bar{x}) + \nabla^2 f(\bar{x}) \bar{h}.$$

E, assim,

$$\bar{h} = -(\nabla^2 f(\bar{x}))^{-1} \nabla f(\bar{x}).$$

Com base nestas informações, o método de Newton gera uma seqüência de iteradas da forma

$$x_{k+1} = x_k - (\nabla^2 f(x_k))^{-1} \nabla f(x_k). \quad (2.4)$$

É claro que, para resolvermos (2.4), não calculamos a inversa daquela matriz, mas sim a equação linear

$$\nabla^2 f(x_k) s = -\nabla f(x_k).$$

E, daí, $x_{k+1} = x_k + s$.

Segundo [6], o valor de $(\nabla^2 f(x_k))^{-1}$ é interpretado como uma ‘correção’ na direção oposta ao gradiente da função, o que acelera o processo iterativo.

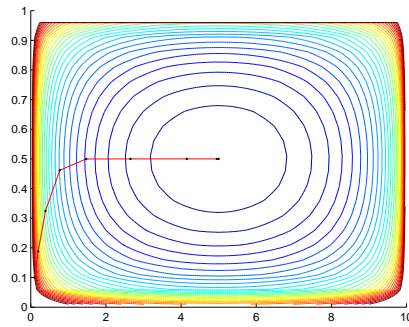
Observe que para chegarmos em (2.4), precisamos trabalhar com a hessiana definida positiva, que é uma exigência do método de Newton.

Podemos verificar na figura 3(a), que apresenta a mesma função que a figura 2(a), como o método de Newton tem um melhor desempenho do que o método de Cauchy; enquanto foram necessárias 345 iterações para o primeiro, o outro solucionou o problema em apenas 9. Isto ocorre pois o método de Newton tem uma excelente convergência local (quadrática) e o ponto inicial dado está próximo a um minimizador da função. Este é um método rápido e de 2ª ordem porém, diverge facilmente.

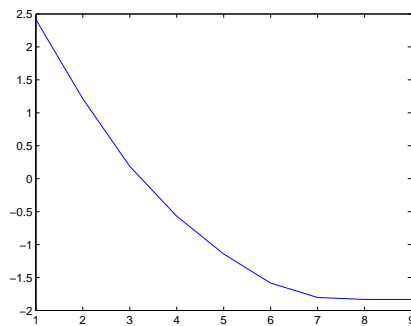
A figura 3(b) mostra como f decresce com as iteradas do método de Newton. Em 3(c), podemos ver a norma do gradiente de f em função das iteradas.

Teorema 4 *Suponha que f é duas vezes diferenciável e que a hessiana $\nabla^2 f(x)$ é Lipschitz contínua em uma vizinhança de uma solução x^* em que as condições suficientes (teorema 3) são satisfeitas. Considere a iteração dada em (2.4). Então:*

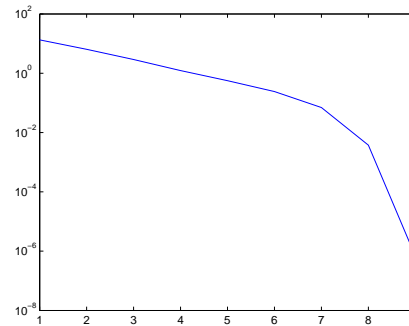
1. *se o ponto x_0 está suficientemente perto de x^* , a seqüência de iteradas converge para x^* ;*
2. *a taxa de convergência de $\{x_k\}$ é quadrática; e*
3. *a seqüência de normas do gradiente $\{\|\nabla f(x_k)\|\}$ converge quadraticamente para zero.*



(a) Curvas de nível da função



(b) Valores da função



(c) Norma do gradiente da função

Figura 3: O método de Newton.

Prova Seja $p_k^N = -[\nabla^2 f(x_k)]^{-1} \nabla f(x_k)$. Da condição de otimalidade $\nabla f(x^*) = 0$, nós temos que

$$\begin{aligned} x_k + p_k^N - x^* &= x_k - x^* - [\nabla^2 f(x_k)]^{-1} \nabla f(x_k) \\ &= [\nabla^2 f(x_k)]^{-1} [\nabla^2 f(x_k)(x_k - x^*) - (\nabla f(x_k) - \nabla f(x^*))]. \end{aligned} \quad (2.5)$$

Visto que

$$\nabla f(x_k) - \nabla f(x^*) = \int_0^1 \nabla^2 f(x_k + t(x^* - x_k))(x_k - x^*) dt,$$

nós temos

$$\begin{aligned} \|\nabla^2 f(x_k)(x_k - x^*) - (\nabla f(x_k) - \nabla f(x^*))\| &= \left\| \int_0^1 [\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))](x_k - x^*) dt \right\| \\ &\leq \int_0^1 \|\nabla^2 f(x_k) - \nabla^2 f(x_k + t(x^* - x_k))\| \|x_k - x^*\| dt \\ &\leq \|x_k - x^*\|^2 \int_0^1 L t dt = \frac{1}{2} L \|x_k - x^*\|^2, \end{aligned} \quad (2.6)$$

onde L é a constante Lipschitz para $\nabla^2 f(x)$ para x próximo de x^* . Visto que $\nabla^2 f(x^*)$ é não singular, e que $\nabla^2 f(x_k) \rightarrow \nabla^2 f(x^*)$, nós temos que $\|[\nabla^2 f(x_k)]^{-1}\| \leq 2\|[\nabla^2 f(x^*)]^{-1}\|$ para todo k

suficientemente grande. Substituindo em (2.5) e (2.6), nós obtemos

$$\|x_k + p_k^N - x^*\| \leq L \|\nabla^2 f(x^*)^{-1}\| \|x_k - x^*\|^2 = \tilde{L} \|x_k - x^*\|^2, \quad (2.7)$$

onde $\tilde{L} = L \|\nabla^2 f(x^*)^{-1}\|$. Usando esta desigualdade, por indução, nós deduzimos que se o ponto inicial estiver suficientemente próximo de x^* , então a seqüência converge para x^* , e a taxa de convergência é quadrática.

Usando as relações $x_{k+1} - x_k = p_k^N$ e $\nabla f(x_k) + \nabla^2 f(x_k)p_k^N = 0$, nós obtemos

$$\begin{aligned} \|\nabla f(x_{k+1})\| &= \|\nabla f(x_{k+1}) - \nabla f(x_k) - \nabla^2 f(x_k)p_k^N\| \\ &= \left\| \int_0^1 \nabla^2 f(x_k + tp_k^N)(x_{k+1} - x_k)dt - \nabla^2 f(x_k)p_k^N \right\| \\ &\leq \int_0^1 \|\nabla^2 f(x_k + tp_k^N) - \nabla^2 f(x_k)\| \|p_k^N\| dt \\ &\leq \frac{1}{2}L \|p_k^N\|^2 \\ &\leq \frac{1}{2}L \|\nabla^2 f(x_k)^{-1}\|^2 \|\nabla f(x_k)\|^2 \\ &\leq 2L \|\nabla^2 f(x^*)^{-1}\|^2 \|\nabla f(x_k)\|^2, \end{aligned}$$

provando que a norma do gradiente converge para zero quadraticamente. \square

Acabamos de ver, no teorema acima, que o método de Newton apresenta uma ótima convergência local. Porém, ele pode não ser adequado visto que nada garante que $\nabla^2 f(x_k)$ é definida positiva quando x_k está longe da solução. Além disso, mesmo que $\nabla^2 f(x_k)$ seja definida positiva a convergência pode não ocorrer, na verdade $\{f(x_k)\}$ pode não decrescer. Esta última possibilidade pode ser eliminada pelo método de Newton com busca unidirecional (veja [7]), visto que para $\nabla^2 f(x_k)$ definida positiva, a direção de Newton $p_k^N = -\nabla^2 f(x_k)^{-1}\nabla f(x_k)$ é uma direção de descida. De fato, $\nabla^2 f(x_k)^{-1}$ é definida positiva e

$$\nabla f(x_k)^T p_k^N = -\nabla f(x_k)^T \nabla^2 f(x_k)^{-1} \nabla f(x_k) < 0.$$

3 *Método de Região de Confiança*

3.1 Introdução

O objetivo principal do método de região de confiança é resolver o problema (1.1). É um método iterativo que gera uma seqüência $\{x_k\}_{k \in \mathbb{N}}$ convergindo para um minimizador local de f .

Em cada iteração é construído um modelo quadrático de f e minimiza-se este modelo em uma região, daí o nome *região de confiança*. Dependendo do resultado obtido com esta minimização, algumas decisões são tomadas:

- aceitar ou não o minimizador como um novo ponto (uma nova iterada);
- aumentar, reduzir, ou deixar inalterada a região de confiança.

Definição 1 *Região de confiança em torno de $x_k \in \mathbb{R}^n$ é o conjunto*

$$\mathcal{B}_k = \{x \in \mathbb{R}^n ; \|x - x_k\|_k \leq \Delta_k\},$$

onde $\Delta_k > 0$ é chamado *raio de região de confiança* e $\|\cdot\|_k$ é uma norma que depende de cada iteração.

Em nosso estudo, estaremos sempre trabalhando com a norma euclidiana, mas também é muito comum trabalhar com a norma ℓ_∞ .

Antes de iniciar o algoritmo, algumas informações precisam ser fornecidas:

- a função $f : \mathbb{R}^n \rightarrow \mathbb{R}$;
- o ponto inicial x_0 ;
- o raio da região de confiança Δ_0 ;
- o raio máximo $\bar{\Delta}$.

Em cada iterada x_k , construímos um modelo quadrático $m_k(p)$, esperando que este modelo seja uma boa representação da variação de f na vizinhança de x_k , e resolvemos o problema

$$\underset{p \in \mathbb{R}^n}{\text{minimizar}} \quad m_k(p) \quad \text{s.a.} \quad \|p\| \leq \Delta_k.$$

Definição 2 Sejam $f : \mathbb{R}^n \rightarrow \mathbb{R}$ uma função continuamente diferenciável e $x_k \in \mathbb{R}^n$. Chamamos de *modelo quadrático da variação de f em relação a x_k* , a função $m_k(p) : \mathbb{R}^n \rightarrow \mathbb{R}$ definida por

$$m_k(p) = p^T \nabla f(x_k) + \frac{1}{2} p^T B_k p$$

onde B_k é a hessiana $\nabla^2 f(x_k)$ ou uma aproximação dela.

Em um primeiro momento, vamos considerar que sabemos resolver tal problema e sua solução é p_k .

Observe que p_k minimiza o modelo m_k e não a função. Desta forma, precisamos verificar se a redução predita pelo modelo se compara com a redução real.

Definição 3 A *redução predita* pelo modelo é

$$\text{pred} = m_k(0) - m_k(p_k) = -m_k(p_k).$$

Definição 4 A *redução real* da função f é dada por

$$\text{ared} = f(x_k) - f(x_k + p_k).$$

Observe que o valor de pred será sempre maior ou igual a zero, pois p_k é minimizador de m_k na região de confiança.

Note, também, que $\text{pred} = 0$ se, e somente se, p_k é um ponto estacionário.

Se a redução real não for aceitável, ou seja, o valor de ared é muito pequeno em relação ao valor de pred , nós rejeitamos o passo e diminuimos o raio da região de confiança. Caso ela seja muito boa (reduz bastante o valor da função), aumentamos o raio. Em caso intermediário, nós deixamos a região de confiança como está.

Temos portanto três casos, dependendo da relação $r = \text{ared}/\text{pred}$:

- se $r < 1/4$, rejeitamos o passo, reduzimos o raio Δ e refazemos o processo;
- se $r \in [1/4, 3/4]$, aceitamos o passo e mantemos o raio;
- se $r > 3/4$, aceitamos o passo e aumentamos o raio para a próxima iteração.

3.2 Um Exemplo

Para deixar mais claro como funciona o método de região de confiança, apresentaremos um exemplo.

Suponha que queiramos resolver o problema (1.1) com $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ dada por

$$f(x) = -10x_1^2 + 10x_2^2 + 4 \sin(x_1 x_2) - 2x_1 + x_1^4,$$

em que x_1 e x_2 são as componentes do vetor x . As figuras 4(a) e 4(b) apresentam o gráfico da função objetivo e de suas curvas de nível, respectivamente.

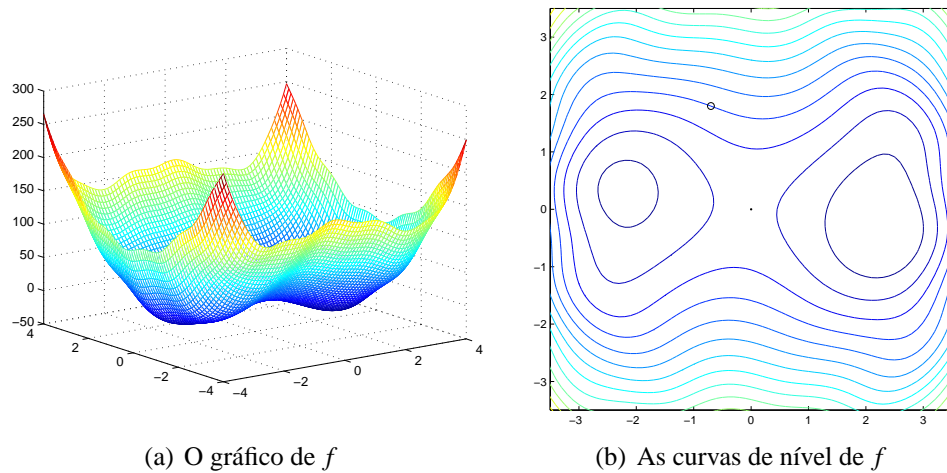


Figura 4: Função objetivo com dois mínimos locais.

Começaremos o nosso processo iterativo com o ponto $x_0 = (-0.7, 1.8)^T$, que está indicado na figura 4(b) por um pequeno círculo. Temos que $f(x_0) = 25.3317$. Utilizando os valores da função, do gradiente e da hessiana no ponto x_0 , construímos um *modelo* da variação da função objetivo em torno do ponto x_0 . Escolhemos, então, com certo grau de arbitrariedade, uma região, contendo x_0 , em que acreditamos que o modelo represente de forma razoável a variação da função objetivo. Esta região é a chamada *região de confiança*. As curvas de nível deste modelo são mostradas na figura 5, dentro de um círculo centrado em x_0 , que representa a região de confiança.

Examinando o modelo nesta região de confiança, podemos observar que no ponto $(-0.9, 0.9)$, representado pelo sinal +, há uma redução no valor do modelo, se comparado com o valor obtido no ponto x_0 . Calculamos, agora, o valor da função objetivo neste novo ponto, obtendo -0.4410 . A redução do modelo foi $\text{pred} = 26.0187$ e a redução do valor da função foi $\text{ared} = 25.7728$. Obtemos $\text{ared}/\text{pred} = 0.9905$. Decidimos então nos mover para o próximo ponto e chamá-lo de x_1 . Decidimos também aumentar o raio da nova região de confiança, uma vez que

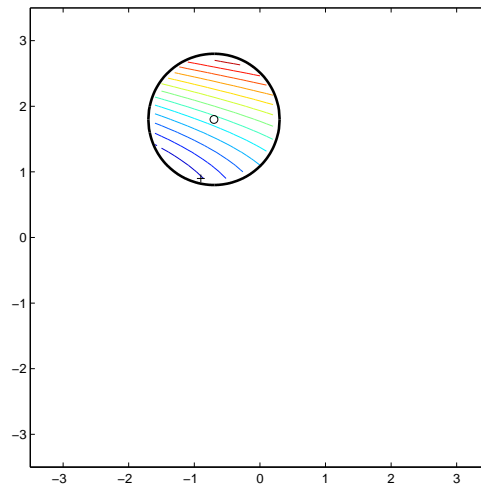


Figura 5: O modelo e a região de confiança ao redor de x_0 .

a redução $ared$ foi grande em relação a $pred$.

Construímos, neste momento, um novo modelo usando as informações da inclinação e da curvatura de f em x_1 . As curvas de nível do novo modelo na região de confiança centrada em x_1 é mostrada na figura 6.

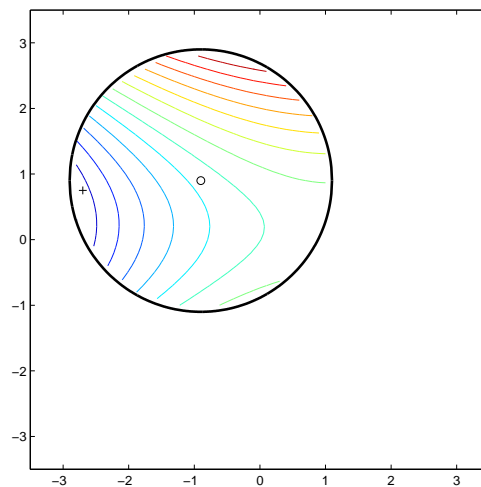
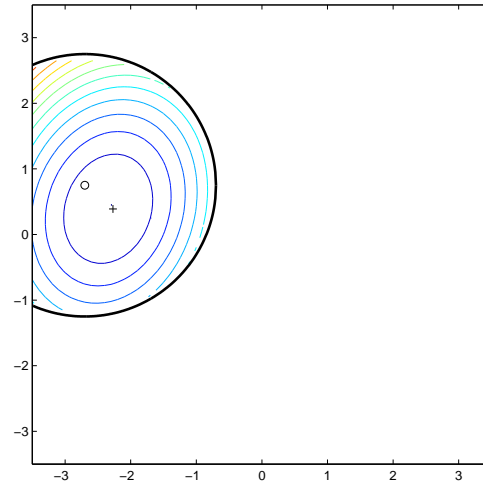


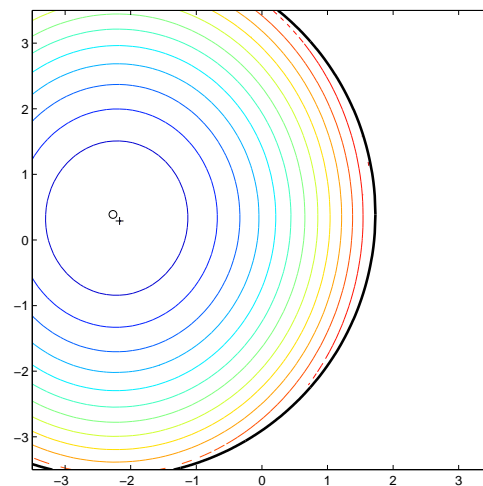
Figura 6: O modelo e a região de confiança ao redor de x_1 .

Como em x_0 , nós examinamos o modelo nesta região de confiança e, selecionamos um novo ponto $(-2.7, 0.75)^T$, indicado por um +, que diminui o valor do modelo em comparação ao valor obtido com x_1 . Como antes, calculamos o valor da função objetivo neste ponto (-12.3253) e verificamos que $ared = 11.8843$, $pred = 42.8363$ e portanto $r = ared/pred = 0.4611$. Houve uma redução significativa da função, o que justifica aceitar o passo, mas mantemos o raio da região inalterado para a iteração seguinte. Assim, aceitamos este ponto como um novo ponto e o chamamos de x_2 . Construímos um novo modelo da função objetivo ao redor dele usando o valor $f(x_2)$, e a inclinação e curvatura de f em x_2 . O resultado dessas operações é mostrado na

figura 7.

Figura 7: O modelo e a região de confiança ao redor de x_2 .

Repetimos o processo, e escolhemos o ponto $(-2.27, 0.39)^T$ pois ele é um mínimo claro neste modelo e o identificamos pelo sinal +. De novo, este movimento é satisfatório pois prediz bem a função objetivo (-22.0120) . Nós, então, nos movemos para o novo ponto, agora chamado de x_3 ; novamente, construímos um modelo ao redor dele, e aumentamos o raio da região da região de confiança. Como ilustrado pela figura 8, o mínimo do modelo agora está muito próximo de x_3 .

Figura 8: O modelo e a região de confiança ao redor de x_3 .

O processo é repetido e, desta maneira, achamos $x_4 = (-2.17, 0.29)^T$, que é mostrado na figura 9.

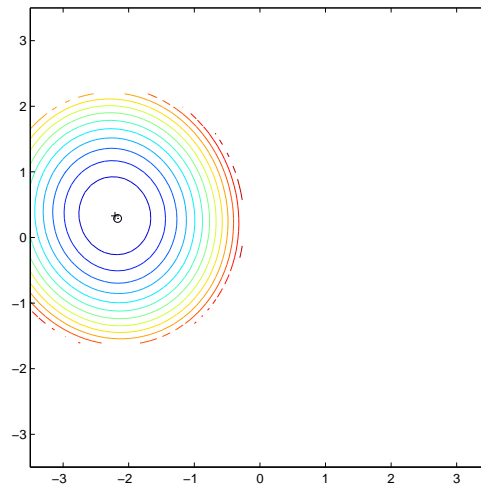


Figura 9: O modelo e a região de confiança ao redor de x_4 .

Como um último movimento, nos deslocamos para o minimizador do modelo, representado por +, e, assim, atingimos o valor $x^* = (-2.2102, 0.3297)^T$. Temos, agora, $f(x^*) = -22.1430$ que é um mínimo local de f . E, assim, encerramos este processo iterativo.

Observe, que este método localiza minimizadores locais da função e, portanto, poderíamos ter encontrado outras soluções. Para exemplificar, note que são nítidos dois mínimos locais na figura 4 e, se partirmos de outro ponto inicial, podemos chegar em outro minimizador da função. Isto está ilustrado na figura 10 e comentado em [3].

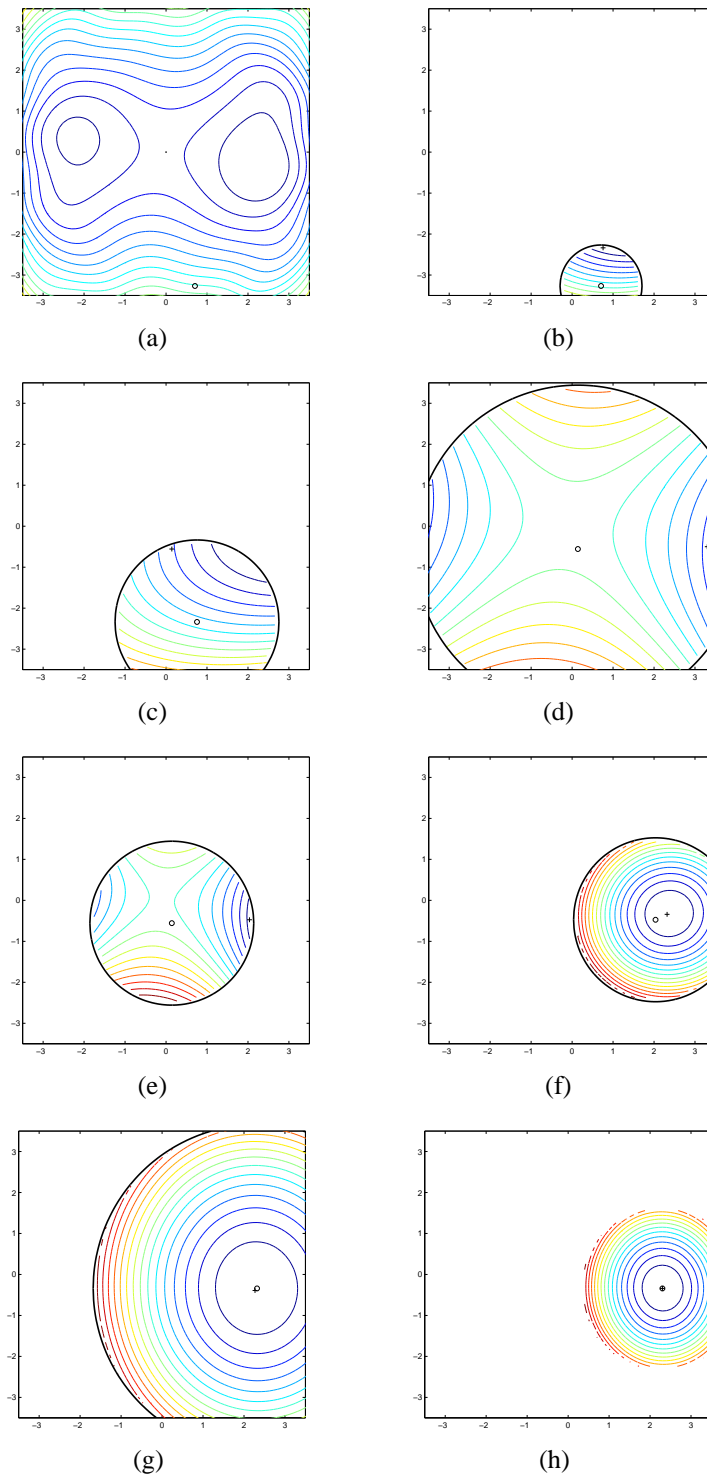


Figura 10: As iterações de uma execução alternativa do algoritmo de região de confiança.

Note que, na figura 10(d), temos uma situação diferente. O modelo quadrático não representa bem a variação da função no ponto representado por +. Enquanto o modelo decresce quando vamos de \circ para +, ele aumenta em f (veja a figura 10(a)). Por causa disso, este novo ponto não é aceito e, decidimos, então, reduzir o raio da região de confiança. (figura 10(e)).

3.3 O Algoritmo

Nesta seção, enunciaremos o algoritmo do método de região de confiança e, em seguida, faremos os comentários passo a passo. É importante salientar que a função f para este algoritmo precisa ser de classe C^2 .

Algoritmo 1: Método de Região de Confiança

Entrada: função $f : \mathbb{R}^n \rightarrow \mathbb{R}$, raio da bola Δ_0 , raio máximo $\bar{\Delta}$, $\varepsilon, \eta \in [0, \frac{1}{4})$, ponto inicial x_0

Saída: minimizador x^*

```

1  $k \leftarrow 0$ 
2 Calcular  $\nabla f(x_k)$  e  $B_k = \nabla^2 f(x_k)$ 
3 enquanto  $\|\nabla f(x_k)\| > \varepsilon$  faça
4    $p_k \leftarrow$  solução de minimizar  $m_k(p) = p^T \nabla f(x_k) + \frac{1}{2} p^T B_k p$    sujeito a  $\|p\| \leq \Delta_k$ 
5    $\rho_k \leftarrow \frac{\text{ared}}{\text{pred}} = \frac{f(x_k) - f(x_k + p_k)}{-m_k(p_k)}$ 
6   se  $\rho_k < \frac{1}{4}$  então
7      $\Delta_{k+1} \leftarrow \frac{1}{4} \|p_k\|$ 
8   senão
9     se  $\rho_k > \frac{3}{4}$  e  $\|p_k\| = \Delta_k$  então
10        $\Delta_{k+1} \leftarrow \min(2\Delta_k, \bar{\Delta})$ 
11     senão
12        $\Delta_{k+1} \leftarrow \Delta_k$ 
13     fim
14   fim
15   se  $\rho_k > \eta$  então
16      $x_{k+1} \leftarrow x_k + p_k$ 
17     Calcular  $\nabla f(x_{k+1})$  e  $B_{k+1} = \nabla^2 f(x_{k+1})$ 
18   senão
19      $x_{k+1} \leftarrow x_k$ 
20   fim
21    $k \leftarrow k + 1$ 
22 fim
23  $x^* \leftarrow x_k$ 

```

3.3.1 Análise do Algoritmo

Começamos o algoritmo na iteração $k = 0$ (linha 1). Quando formos apresentar a programação em MATLAB, iniciaremos com o valor $k = 1$ pois o MATLAB não suporta o índice 0.

Em seguida, é calculado $\nabla f(x_0)$ e $B_0 = \nabla^2 f(x_0)$.

Na linha 3, verificamos se o ponto atual é um ponto estacionário de f . Observe que a regra de parada do algoritmo é a verificação da proximidade da $\|\nabla f(x_k)\|$ de zero. Tal regra é utilizada por causa do teorema 3.

Calculamos, então, p_k . Na verdade, não é necessário achar uma solução exata mas uma aproximação de

$$\underset{p \in \mathbb{R}^n}{\text{minimizar}} \quad m_k(p) = p^T \nabla f(x_k) + \frac{1}{2} p^T B_k p \quad \text{sujeito a } \|p\| \leq \Delta_k \quad (3.1)$$

Apresentaremos, neste trabalho, uma solução aproximada para (3.1). Formas de obter a solução exata podem ser encontradas em [3, p. 102–200].

Obtemos ρ_k como sendo a razão entre ared e pred . Se este valor for muito pequeno, significa que o ponto p_k não reduz a função tão bem em relação à redução obtida pelo modelo. Desta forma, diminuímos a região de confiança. O valor $\frac{1}{4} \|p_k\|$ utilizado pode ser substituído por outra fração de $\|p_k\|$.

Agora, se a redução da função for comparável a redução do modelo e, mais ainda, se p_k estiver na fronteira da região de confiança, significa que o nosso modelo representa muito bem a função permitindo um aumento da região de confiança. É importante frisar que a comparação de igualdade $\|p_k\| = \Delta_k$ serve para fins teóricos mas não é interessante do ponto de vista computacional, pois estamos lidando com números reais. Uma substituição plausível seria $\|p_k\| > 0.99\Delta_k$.

Uma última possibilidade é apresentada na linhas 11–13 em que temos um caso intermediário e optamos por manter o raio atual.

O passo seguinte, apresentado nas linhas 15–20 é verificar se vale a pena substituir o ponto atual. É comum utilizar valores pequenos para η (por exemplo, $\eta = 0.1$). Se ocorrer a substituição, é necessário calcular ∇f e B no novo ponto.

Agora, incrementamos k e retornamos para a linha 3.

Quando $\|\nabla f(x_k)\|$ for menor ou igual a ε espera-se ter encontrado uma boa aproximação para um minimizador de f .

4 O Método Dogleg: um Subproblema de Região de Confiança

Para implementarmos o algoritmo de região de confiança, falta resolvermos o problema (3.1). Uma forma de resolvermos este problema é através do método dogleg, que não nos dará uma solução exata, mas uma aproximação dela, o que é suficiente para o método de região de confiança.

4.1 O Ponto de Cauchy

Suponha que estejamos na k -ésima iteração do método de região de confiança para um problema do tipo (1.1).

Definição 5 O *arco de Cauchy* é definido por

$$x_k^C(t) = \{x; x = x_k - t\nabla f(x_k), t \geq 0 \text{ e } \|t\nabla f(x_k)\| \leq \Delta_k\}.$$

Observe, pela definição, que o arco de Cauchy nada mais é do que o segmento de reta que une o ponto x_k a fronteira da região de confiança na direção de máximo declive (figura 11).

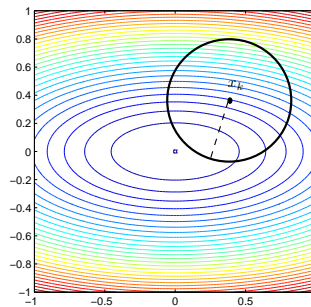


Figura 11: O arco de Cauchy representado pela linha tracejada.

Uma vez que temos o modelo quadrático de f no ponto x_k , $m_k(p)$, calculemos o ponto que minimiza m_k no arco de Cauchy, ou seja,

$$t_k^* = \arg \min_{\substack{t \geq 0 \\ \|\nabla f(x_k) - t \nabla f(x_k)\| \leq \Delta_k}} m_k(-t \nabla f(x_k)) \quad (4.1)$$

Definição 6 O *ponto de Cauchy*, que denotaremos p^C , é o ponto dado por

$$p^C = x_k - t_k^* \nabla f(x_k).$$

Em outras palavras, o ponto de Cauchy é o minimizador do modelo $m_k(\cdot)$ ao longo da direção de máximo declive, restrito à bola de raio Δ_k em torno de x_k .

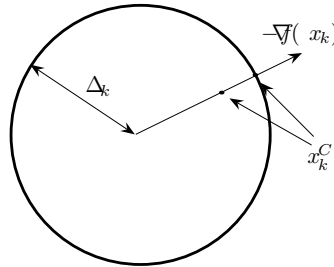


Figura 12: O ponto de Cauchy.

Três situações podem ocorrer na determinação do ponto de Cauchy:

- (i) O modelo ser convexo ao longo da direção $-\nabla f(x_k)$ e o seu minimizador irrestrito estar dentro da região de confiança (figura 13(a)).
- (ii) O modelo ser convexo ao longo da direção $-\nabla f(x_k)$ e o seu minimizador irrestrito estar fora da região de confiança (figura 13(b)).
- (iii) O modelo não ser convexo ao longo da direção $-\nabla f(x_k)$ (figura 13(c)).

No caso (i), o resultado para (4.1) pode ser obtido analiticamente através do cálculo abaixo.

$$\begin{aligned} 0 &= \frac{dm_k}{dt}(-t_k^* \nabla f(x_k)) \\ &= -\nabla f(x_k)^T \nabla f(x_k) + t_k^* \nabla f(x_k)^T \nabla^2 f(x_k) \nabla f(x_k) \end{aligned}$$

E, portanto,

$$t_k^* = \frac{\nabla f(x_k)^T \nabla f(x_k)}{\nabla f(x_k)^T \nabla^2 f(x_k) \nabla f(x_k)}.$$

Em (ii) e (iii), o ponto de Cauchy está na fronteira da região de confiança, isto é,

$$t_k^* = \frac{\Delta}{\|\nabla f(x_k)\|}.$$

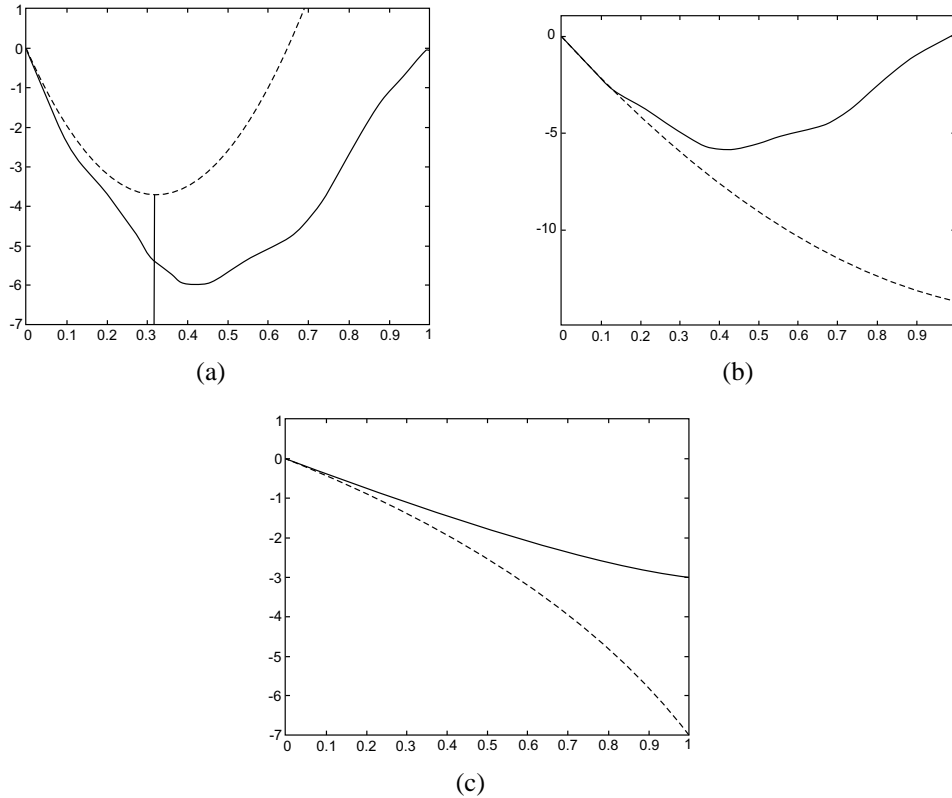


Figura 13: As três situações possíveis para o ponto de Cauchy. As linhas sólidas e tracejadas são, respectivamente, os gráficos de $f(x_k + td) - f(x_k)$ e $m_k(td)$ (com $d = -\Delta_k \nabla f(x_k) / \|\nabla f(x_k)\|$) em função de t .

4.2 O Método

A solução p_k do problema (3.1) depende do raio da região de confiança. Quando B_k é definida positiva, o minimizador irrestrito de m_k é dado por $p^N = -B_k^{-1} \nabla f(x_k)$, ou seja, a direção de Newton. Se este for um ponto viável para (3.1), ele será uma solução. Assim,

$$p_k(\Delta) = p^N, \quad \text{quando } \Delta \geq \|p^N\|. \quad (4.2)$$

Quando Δ é muito pequeno, escolhamos o ponto de Cauchy como solução de (3.1), isto é,

$$p_k(\Delta) \approx -\Delta \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}, \quad \text{quando } \Delta \text{ é pequeno.} \quad (4.3)$$

Para valores intermediários de Δ , a solução exata do problema de região de confiança segue

uma trajetória curva como a que está apresentada na figura 14.

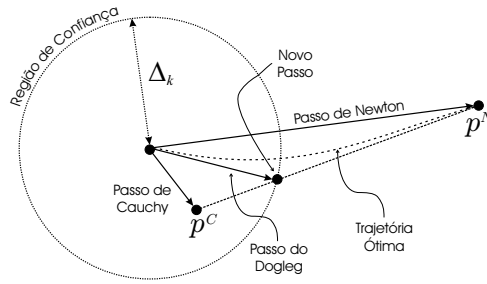


Figura 14: O passo do método dogleg.

O método dogleg acha uma solução aproximada trocando a trajetória ótima por $p_k(\Delta)$, um caminho com dois segmentos de reta. O primeiro segmento vai da origem a p^C enquanto o segundo vai de p^C a p^N . Formalmente, nós denotamos esta trajetória por $\tilde{p}(\tau)$ com $\tau \in [0, 2]$, onde

$$\tilde{p}(\tau) = \begin{cases} \tau p^C, & 0 \leq \tau \leq 1, \\ p^C + (\tau - 1)(p^N - p^C), & 1 \leq \tau \leq 2. \end{cases} \quad (4.4)$$

O método dogleg escolhe p para minimizar o modelo m ao longo deste caminho, sujeito a região de confiança. De fato, não é mesmo necessário realizar uma busca, basta tomar o ponto em que o caminho dogleg intersecta o limite da região de confiança. O ponto de intersecção pode ser calculado analiticamente. Estas afirmações ficam provadas com o lema a seguir.

Lema 1 *Seja B definida positiva, então:*

- (i) $\|\tilde{p}(\tau)\|$ é uma função crescente de τ , e
- (ii) $m(\tilde{p}(\tau))$ é uma função decrescente de τ .

Prova É fácil mostrar que (i) e (ii) valem para $\tau \in [0, 1]$, assim nós voltamos nossa atenção para o caso de $\tau \in [1, 2]$.

Para (i), defina $h(\alpha)$ por:

$$\begin{aligned} h(\alpha) &= \frac{1}{2} \|\tilde{p}(1 + \alpha)\|^2 \\ &= \frac{1}{2} \|p^C + \alpha(p^N - p^C)\|^2 \\ &= \frac{1}{2} \|p^C\|^2 + \alpha p^{CT} (p^N - p^C) + \frac{1}{2} \alpha^2 \|p^N - p^C\|^2. \end{aligned}$$

O lema é provado se mostrarmos que $h'(\alpha) \geq 0$ para $\alpha \in (0, 1)$. Agora,

$$\begin{aligned}
 h'(\alpha) &= -p^{CT}(p^C - p^N) + \alpha \|p^C - p^N\|^2 \\
 &\geq -p^{CT}(p^C - p^N) \\
 &= \frac{g^T g}{g^T B g} g^T \left(-\frac{g^T g}{g^T B g} g + B^{-1} g \right) \\
 &= g^T g \frac{g^T B^{-1} g}{g^T B g} \left(1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} \right) \\
 &\geq 0
 \end{aligned}$$

onde $g = \nabla f(x_k)$ e a última desigualdade vem do lema 2.

Para (ii), nós definimos $\hat{h}(\alpha) = m(\tilde{p}(1 + \alpha))$ e mostramos que $\hat{h}'(\alpha) \leq 0$ para $\alpha \in (0, 1)$.

Substituindo (4.4) em (3.1) e derivando em relação ao argumento conduz a

$$\begin{aligned}
 \hat{h}'(\alpha) &= (p^N - p^C)^T (g + B p^C) + \alpha (p^N - p^C)^T B (p^N - p^C) \\
 &\leq (p^N - p^C)^T (g + B p^C + B (p^N - p^C)) \\
 &= (p^N - p^C)^T (g + B p^N) = 0
 \end{aligned}$$

concluindo a prova. □

Lema 2 *Seja $g \in \mathbb{R}^n$ e seja $B \in \mathbb{R}^{n \times n}$ uma matriz definida positiva. Então*

$$g^T g \frac{g^T B^{-1} g}{g^T B g} \left(1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} \right) \geq 0.$$

Prova Sabemos que $g^T g = \|g\|^2 \geq 0, \forall g \in \mathbb{R}^n$.

Como B é definida positiva, temos que $g^T B^{-1} g$ e $g^T B g$ são valores positivos.

Sejam $u = B^{1/2} g$ e $v = B^{-1/2} g$, assim

$$\begin{aligned}
 \|(B^{1/2} g)^T B^{-1/2} g\|^2 &\leq [(B^{1/2} g)^T (B^{1/2} g)][(B^{-1/2} g)^T (B^{-1/2} g)] \\
 \|g^T B^{1/2} B^{-1/2} g\|^2 &\leq (g^T B g)(g^T B^{-1} g) \\
 (g^T g)^2 &\leq (g^T B g)(g^T B^{-1} g) \\
 \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} &\leq 1 \\
 1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} &\geq 0
 \end{aligned}$$

Portanto,

$$g^T g \frac{g^T B^{-1} g}{g^T B g} \left(1 - \frac{(g^T g)^2}{(g^T B g)(g^T B^{-1} g)} \right) \geq 0,$$

concluindo a demonstração do lema. □

Segue do lema 1 que o caminho $\tilde{p}(\tau)$ intersecta o limite da região de confiança $\|p\| = \Delta$ em exatamente um ponto se $\|p^N\| \geq \Delta$, e em nenhum caso mais. Visto que m é decrescente ao longo do caminho, o valor escolhido de p será em p^N se $\|p^N\| \leq \Delta$.

Se $\|p^N\| > \Delta$, escolhe-se o ponto de intersecção do dogleg com o limite da região de confiança. Para este caso, calcula-se o valor de τ resolvendo a equação:

$$\|p^C + \lambda d\|^2 = \Delta^2$$

em que $\lambda = \tau - 1$ e $d = p^N - p^C$.

Observe que resolver a equação acima é o mesmo que resolver

$$\|p^C\|^2 + 2\lambda p^{CT} d + \lambda^2 \|d\|^2 = \Delta^2$$

ou, ainda,

$$(\|d\|^2) \lambda^2 + (2p^{CT} d) \lambda + (\|p^C\|^2 - \Delta^2) = 0. \quad (4.5)$$

As soluções para a equação de segundo grau (4.5) são dadas por

$$\lambda = \frac{-2p^{CT} d \pm \sqrt{(2p^{CT} d)^2 - 4\|d\|^2 (\|p^C\|^2 - \Delta^2)}}{2\|d\|^2}.$$

Como $\|p^C\| < \Delta$, é fácil ver que há sempre uma raiz positiva e uma negativa. A solução positiva é a que corresponde com a intersecção do dogleg com o limite da região de confiança. Assim, a intersecção do dogleg com o limite da região de confiança acontece quando

$$\tau = \frac{-2p^{CT} (p^N - p^C) + \sqrt{(2p^{CT} (p^N - p^C))^2 - 4\|p^N - p^C\|^2 (\|p^C\|^2 - \Delta^2)}}{2\|p^N - p^C\|^2} + 1.$$

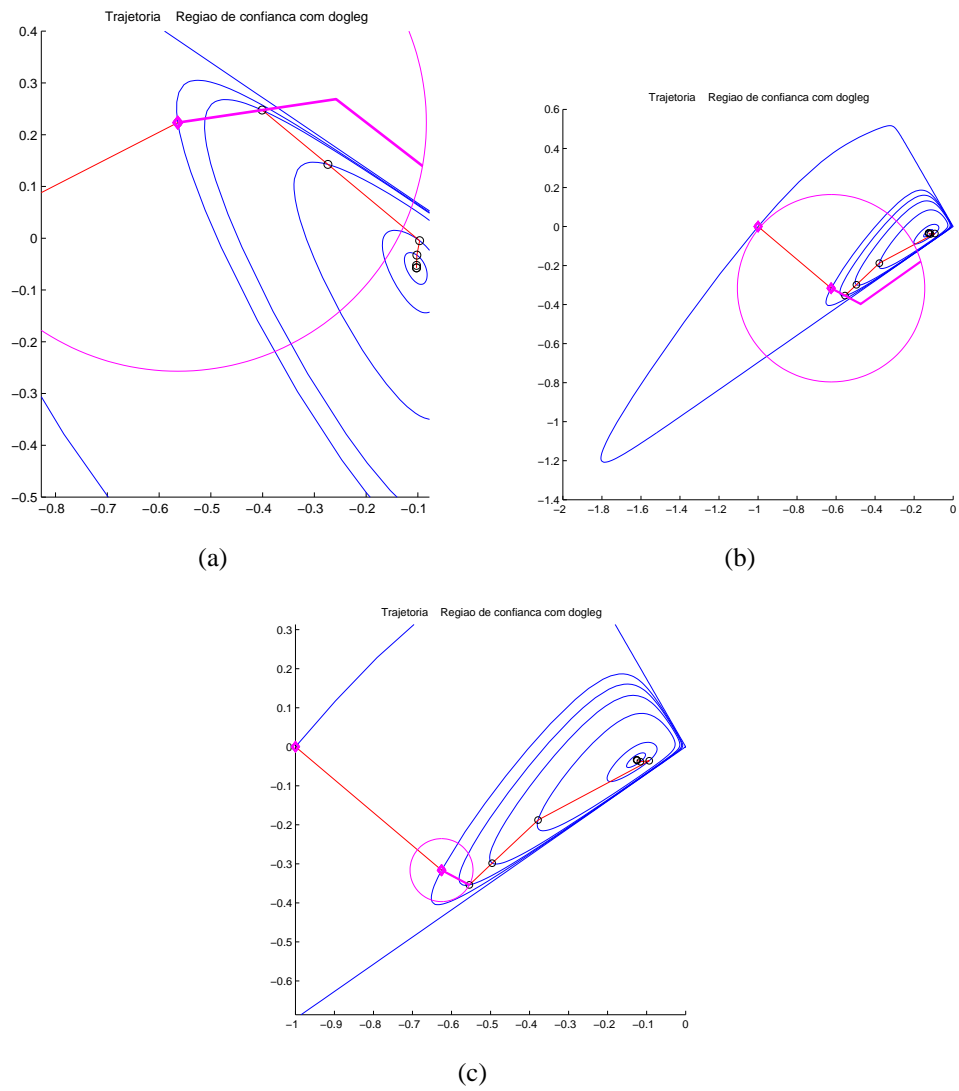


Figura 15: Ilustrações do método dogleg.

4.3 O Algoritmo

Com base na seção anterior, apresentamos, agora, um algoritmo para o método dogleg.

Algoritmo 2: Método Dogleg

Entrada: matriz B ($n \times n$) positiva definida, vetor $\nabla f(x_k)$ ($n \times 1$), raio da bola Δ

Saída: passo do dogleg pd ($n \times 1$)

1 $curv \leftarrow \nabla f(x_k)^T B \nabla f(x_k)$

2 **se** $curv \leq 0$ **então**

3 $pd \leftarrow -\Delta \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$

4 **senão**

5 $p^C \leftarrow -\frac{\nabla f(x_k)^T \nabla f(x_k)}{curv} \nabla f(x_k)$

6 **se** $\|p^C\| \geq \Delta$ **então**

7 $pd \leftarrow -\Delta \frac{\nabla f(x_k)}{\|\nabla f(x_k)\|}$

8 **senão**

9 **se** $B \neq 0$ **então**

10 Informar que B não é definida positiva.

11 $pd \leftarrow p^C$

12 **senão**

13 $p^N \leftarrow -B^{-1} \nabla f(x_k)$

14 **se** $\|p^N\| \leq \Delta$ **então**

15 $pd \leftarrow p^N$

16 **senão**

17 $\lambda \leftarrow \frac{-2p^{CT}(p^N - p^C) + \sqrt{(2p^{CT}(p^N - p^C))^2 - 4\|p^N - p^C\|^2(\|p^C\|^2 - \Delta^2)}}{2\|p^N - p^C\|^2}$

18 $pd \leftarrow p^C + \lambda(p^N - p^C)$

19 **fim**

20 **fim**

21 **fim**

22 **fim**

5 Testes Práticos

Este capítulo traz uma série de problemas que exemplificam o uso do método de região de confiança, utilizando o passo de dogleg para resolver o subproblema 3.1. Os resultados apresentados a seguir foram obtidos através do software MATLAB versão 7.0 e os arquivos fonte encontram-se no apêndice A.

Para os testes a seguir, utilizamos o raio da bola iniciando com o valor $\Delta_0 = 1$ e o valor para o raio máximo foi fixado em $\bar{\Delta} = 10$. Definimos, também, $\varepsilon = 10^{-5}$ e $\eta = 1/8$.

5.1 Método de Barreiras

Como um primeiro teste, apresentaremos o método de barreiras. A idéia principal do método de barreiras é transformar um problema com restrições não lineares de desigualdade em um outro problema que não tenha restrições.

Consideremos o problema de programação não linear restrito

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} \quad f(x) \quad \text{s.a.} \quad g(x) \geq 0. \quad (5.1)$$

Definição 7 A região viável é definida por

$$\Omega = \{x \in \mathbb{R}^n ; g(x) \geq 0\}.$$

Para eliminarmos as restrições, acrescentamos um termo na função objetivo. Tal termo fará com que a nova função objetivo tenda ao infinito nos pontos próximos a fronteira da região viável, criando assim uma barreira que não permite que os métodos de otimização irrestritos, que começam no interior da região, se aproximem destes pontos.

Definição 8 A região estritamente viável é definida por

$$\Omega^\circ = \{x \in \mathbb{R}^n ; g(x) > 0\}.$$

Assumiremos que Ω° é um conjunto não vazio. Funções barreiras para o problema (5.1) têm as seguintes propriedades:

- elas são infinitas em todos os pontos, exceto em Ω° ;
- elas são suaves em Ω° ;
- seus valores tendem a infinito quando x se aproxima da fronteira de Ω° .

Um exemplo de função barreira é a função barreira logarítmica

$$-\sum_{i=1}^m \log(g_i(x)).$$

Assim, a função que irá substituir a função objetivo do problema (5.1) será dada por:

$$P(x; r) = f(x) - r \sum_{i=1}^m \log(g_i(x)), \quad (5.2)$$

onde r é chamado de *parâmetro de barreira*.

Pode-se mostrar que a solução $x^*(r)$ de

$$\underset{x \in \mathbb{R}^n}{\text{minimizar}} P(x, r) = f(x) - r \sum_{i=1}^m \log(g_i(x)) \quad (5.3)$$

satisfaz

$$\lim_{r \rightarrow 0} x^*(r) = x^*,$$

onde x^* é solução do problema (5.1).

5.1.1 Exemplos com a Função Barreira Logarítmica

Em nossa primeira implementação, resolvemos o problema (5.1) com

$$f(x) = c^T x \quad \text{e} \quad g(x) = b - A^T x$$

em que

$$A = \begin{bmatrix} 1.3978 & 0.7645 & -0.1615 & 0.3728 \\ 0 & -1.6656 & 0.1253 & 26.4890 \end{bmatrix}, \quad b = \begin{bmatrix} 0 \\ 0 \\ 1.2130 \\ 0.9754 \end{bmatrix}, \quad c = \begin{bmatrix} -3.0017 \\ 0.7900 \end{bmatrix}.$$

Foi dado como ponto inicial $x_0 = (-1, 0)^T$.

Para este problema, foram necessárias 18 iterações. Nestas 18 iterações tivemos:

- 18 cálculos de função;
- 13 cálculos do gradiente da função;
- 13 cálculos da hessiana da função;
- 11 cálculos de $-\nabla^2 f(x_k)^{-1} \nabla f(x_k)$;
- 12 passos aceitos.

O valor encontrado foi $(-0.069004, -0.001679)^T$.

Abaixo, na tabela 1, apresentamos algumas informações obtidas em cada iteração:

- o gradiente de f no ponto x_k (coluna 2);
- o valor de f no ponto x_k (coluna 3);
- o raio da região de confiança (coluna 4);
- se o passo foi aceito (coluna 5)¹;
- se o passo foi na fronteira da região de confiança (coluna 6)².

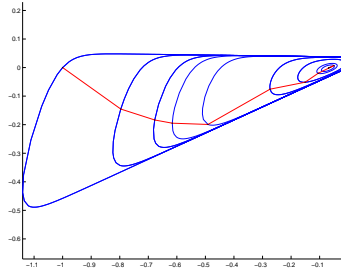
k	$\ \nabla f(x_k)\ $	$f(x_k)$	Δ_k	aceitou	fronteira
1	37.785139	29.601455	1.000000	0	0
2	37.785139	29.601455	0.250000	1	1
3	28.117550	21.924689	0.500000	0	0
4	28.117550	21.924689	0.125000	1	1
5	25.422356	18.565069	0.250000	0	0
6	25.422356	18.565069	0.062500	1	1
7	23.278287	17.042047	0.125000	1	1
8	27.958341	14.662737	0.250000	1	1
9	17.529071	9.775083	0.500000	0	0
10	17.529071	9.775083	0.125000	1	1
11	36.629147	8.142141	0.250000	0	0
12	36.629147	8.142141	0.055537	1	1
13	10.992944	7.356282	0.111074	1	0
14	9.277164	7.242155	0.111074	1	0
15	1.811304	7.169047	0.111074	1	0
16	0.111485	7.164605	0.111074	1	0
17	0.000491	7.164586	0.111074	1	0
18	0.000000	7.164586	0.111074	0	0

Tabela 1: Resultados da primeira implementação com a função barreira logarítmica.

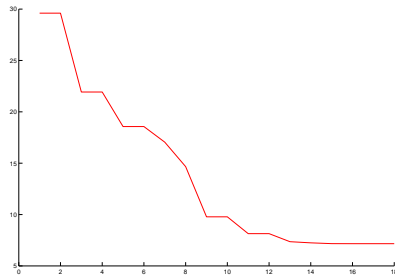
¹Passos aceitos foram marcados com 1, os demais ficaram com 0.

²Passos na fronteira foram marcados com 1, os demais ficaram com 0.

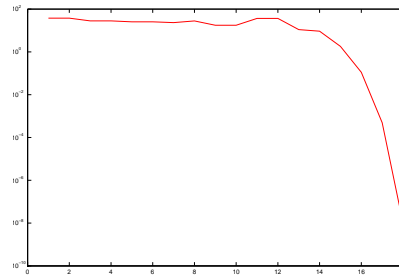
Os gráficos com as curvas de nível da função e o “caminho” do método são apresentados na figura 16(a). Outros gráficos interessantes são os da função objetivo em função das iteradas, e o do gradiente em função das iteradas, que estão nas figuras 16(b) e 16(c), respectivamente.



(a) Curvas de nível da função



(b) Valores da função



(c) Norma do gradiente da função

Figura 16: Primeira implementação com a função barreira logarítmica.

Em uma segunda implementação, continuaremos com o problema (5.1) mas, agora, tendo A como uma matriz 200×400 , $b \in \mathbb{R}^{400}$ e $c \in \mathbb{R}^{200}$.

Para este problema foram necessárias 46 iterações. Nestas 46 iterações tivemos:

- 46 cálculos de função;
- 32 cálculos do gradiente da função;
- 32 cálculos da hessiana da função;
- 16 cálculos de $-\left[\nabla^2 f(x_k)\right]^{-1} \nabla f(x_k)$;
- 31 passos aceitos.

Os resultados para este problema são mostrados na tabela 2.

Veja, na figura 17(a), a redução obtida com o método e, na figura 17(b), como o gradiente da função se aproximou de 0.

k	$\ \nabla f(x_k)\ $	$f(x_k)$	Δ_k	aceitou	fronteira
1	2326.812062	2090.986375	1.000000	0	0
2	2326.812062	2090.986375	0.250000	0	0
3	2326.812062	2090.986375	0.062500	1	1
4	2171.695337	1952.243337	0.125000	0	0
5	2171.695337	1952.243337	0.031250	1	1
6	2135.037055	1884.996203	0.062500	1	1
7	2068.831989	1753.837040	0.125000	1	1
8	2054.720360	1503.412459	0.250000	1	1
9	1835.183817	1213.945076	0.500000	0	0
10	1835.183817	1213.945076	0.125000	0	0
11	1835.183817	1213.945076	0.031250	1	1
12	1772.011362	1157.593842	0.062500	1	1
13	1638.606883	1051.785569	0.125000	0	0
14	1638.606883	1051.785569	0.031250	1	1
15	1526.072440	1003.022932	0.062500	0	0
16	1526.072440	1003.022932	0.015625	1	1
17	1460.025154	979.700715	0.031250	1	1
18	1320.538947	936.653393	0.062500	0	0
19	1320.538947	936.653393	0.015625	1	1
20	1250.206556	916.643691	0.031250	1	1
21	1133.374925	879.920604	0.062500	0	0
22	1133.374925	879.920604	0.015625	1	1
23	1061.125839	863.010680	0.031250	1	1
24	951.725511	833.237588	0.062500	1	1
25	893.126303	790.464984	0.125000	0	0
26	893.126303	790.464984	0.031250	0	0
27	893.126303	790.464984	0.007812	1	1
28	786.740999	783.962127	0.015625	1	1
29	776.430384	773.240459	0.031250	1	1
30	667.357915	757.841857	0.062500	0	0
31	667.357915	757.841857	0.015625	1	1
32	701.838405	748.521557	0.031250	1	1
33	565.843097	736.266737	0.062500	0	0
34	565.843097	736.266737	0.015625	1	1
35	554.884251	728.188572	0.031250	1	1
36	481.956545	717.120297	0.062500	0	0
37	481.956545	717.120297	0.015625	1	1
38	583.743573	711.201247	0.031250	1	1
39	369.206660	703.968249	0.062500	1	1
40	904.997392	691.908529	0.125000	1	1
41	354.661008	679.719275	0.250000	1	0
42	107.100715	675.706204	0.250000	1	0
43	16.781689	675.386233	0.250000	1	0
44	1.079610	675.377234	0.250000	1	0
45	0.007717	675.377196	0.250000	1	0
46	0.000000	675.377196	0.250000	0	0

Tabela 2: Resultados da segunda implementação com a função barreira logarítmica.

5.2 Função de Rosenbrock

A função de Rosenbrock é a função $f : \mathbb{R}^2 \rightarrow \mathbb{R}$ dada por

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \quad (5.4)$$

Também conhecida como função banana, a função de Rosenbrock, é muito utilizada em exemplos de otimização por causa da baixa convergência que muitos métodos apresentam quando tentam resolver este problema.

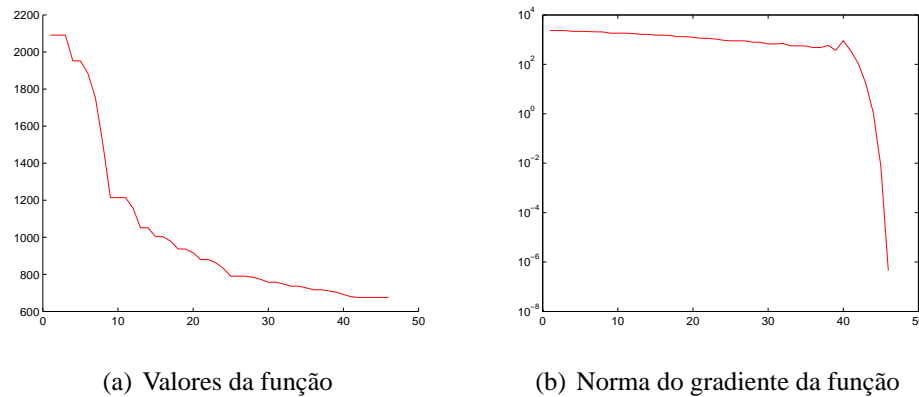


Figura 17: Segunda implementação com a função barreira logarítmica.

Esta função tem um único mínimo no ponto $x^* = (1, 1)^T$, onde $f(x) = 0$.

5.2.1 Comparação com Outros Métodos

O MATLAB traz consigo uma demonstração de convergência de alguns métodos utilizando a função banana. Para isto, ele parte do ponto inicial $(-1.9, 2)^T$ e informa quantas iterações e quantos cálculos de função são necessários para completar o processo. Estes resultados são apresentados na tabela 3.

Método	Nº de iterações	Nº de cálculos da função
Broyden-Fletcher-Goldfarb-Shanno (BFGS)	34	50
Davidon-Fletcher-Powell (DFP)	45	64
Máximo Declive	56	250
Simplex (Nelder-Mead)	109	201
Gauss-Newton	11	48
Levenberg-Marquardt	18	82

Tabela 3: Comparação da convergência de alguns métodos, utilizando a função banana.

Utilizando esta mesma função e mesmo ponto inicial, o algoritmo apresentado neste trabalho resolve o problema em 29 iterações (tendo 24 passos aceitos), com 29 cálculos de função. As informações relativas a cada iteração são mostradas na tabela 4.

Os gráficos relativos a este problema são mostrados na figura 18.

5.3 Outro Exemplo

O exemplo de região de confiança apresentado na página 17 é agora mostrado aqui. O ponto inicial escolhido é o ponto $(-0.7, 1.8)^T$.

k	$\ \nabla f(x_k)\ $	$f(x_k)$	Δ_k	aceitou	fronteira
1	1270.869136	267.620000	1.000000	1	1
2	93.218143	9.074450	2.000000	1	0
3	13.626585	7.053383	2.000000	0	0
4	13.626585	7.053383	0.500000	1	1
5	15.181018	6.292423	1.000000	1	1
6	55.313182	5.757279	1.000000	1	0
7	9.077338	4.328102	1.000000	0	0
8	9.077338	4.328102	0.250000	1	1
9	6.102582	3.866287	0.500000	1	1
10	20.021374	3.227292	1.000000	1	0
11	10.565481	2.528438	1.000000	1	0
12	16.304848	2.124937	1.000000	1	0
13	3.815624	1.498859	1.000000	0	0
14	3.815624	1.498859	0.112875	1	1
15	2.440385	1.236493	0.225751	1	1
16	6.906411	0.907624	0.451501	1	0
17	2.559568	0.617520	0.451501	1	0
18	12.243209	0.543025	0.451501	1	0
19	0.716679	0.254326	0.451501	0	0
20	0.716679	0.254326	0.112875	1	1
21	1.813501	0.185785	0.225751	1	1
22	8.506109	0.142163	0.225751	1	0
23	0.449517	0.054371	0.225751	1	1
24	5.980214	0.034922	0.225751	1	0
25	0.184879	0.006879	0.225751	1	0
26	2.270830	0.002783	0.225751	1	0
27	0.005582	0.000030	0.225751	1	0
28	0.013108	0.000000	0.225751	1	0
29	0.000000	0.000000	0.225751	0	0

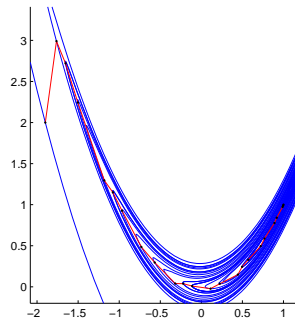
Tabela 4: Resultados da função banana.

O problema foi resolvido em 8 iterações, encontrando como solução $(-2.210220, 0.329748)^T$. Os detalhes são mostrados na tabela 5 e na figura 19.

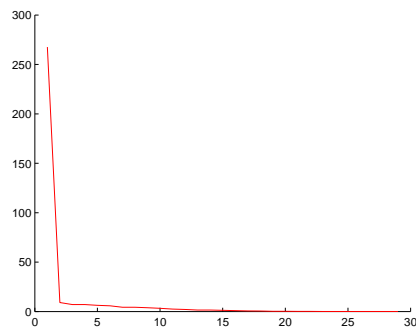
k	$\ \nabla f(x_k)\ $	$f(x_k)$	Δ_k	aceitou	fronteira
1	37.412383	25.331739	1.000000	1	1
2	22.015435	-3.328389	2.000000	1	1
3	21.804908	-11.946020	2.000000	1	0
4	10.810258	-19.502076	2.000000	1	0
5	7.079480	-21.504334	2.000000	1	0
6	0.656196	-22.136762	2.000000	1	0
7	0.008697	-22.142960	2.000000	1	0
8	0.000002	-22.142961	2.000000	0	0

Tabela 5: Resultados.

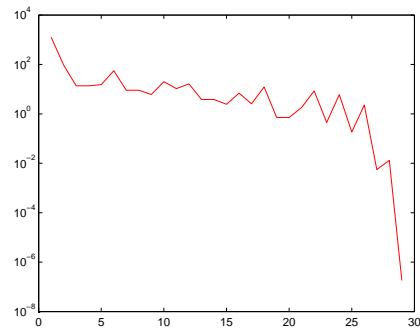
Esta mesma função é apresentada em [3] com ponto inicial $(0.7067, -3.2672)^T$. Em nossa implementação, esse mesmo problema é resolvido em 8 iterações, tendo 6 passos aceitos. A tabela 6 traz os resultados numéricos e os gráficos podem ser analisados na figura 20.



(a) Curvas de nível da função



(b) Valores da função

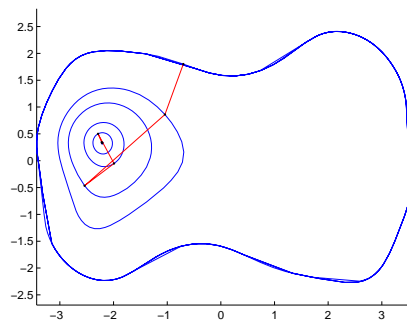


(c) Norma do gradiente da função

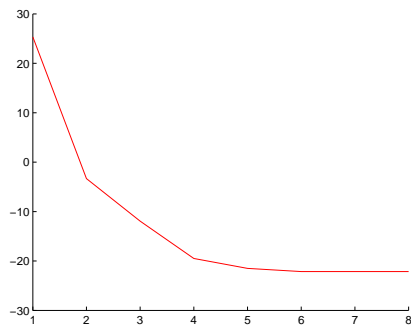
Figura 18: Função banana.

k	$\ \nabla f(x_k)\ $	$f(x_k)$	Δ_k	aceitou	fronteira
1	67.506970	97.628833	1.000000	1	1
2	48.170384	40.182833	2.000000	1	1
3	20.502459	-18.529253	4.000000	0	0
4	20.502459	-18.529253	1.000000	1	1
5	3.975764	-30.978600	1.000000	1	0
6	0.162488	-31.180400	1.000000	1	0
7	0.000459	-31.180733	1.000000	1	0
8	0.000000	-31.180733	1.000000	0	0

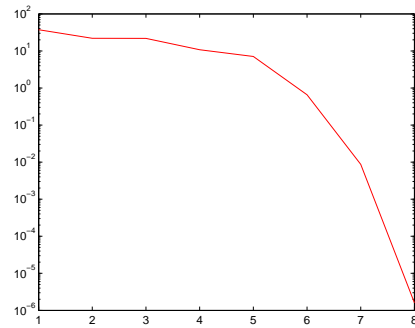
Tabela 6: Resultados.



(a) Curvas de nível da função

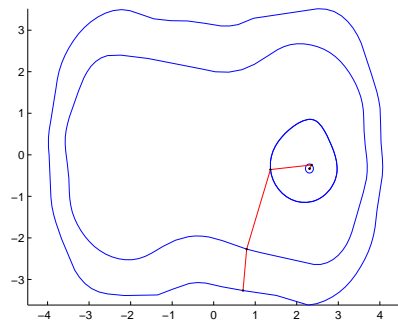


(b) Valores da função

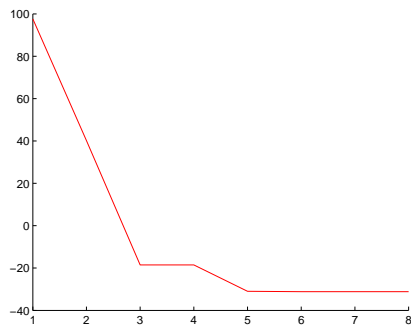


(c) Norma do gradiente da função

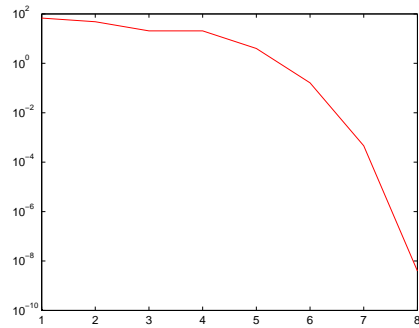
Figura 19: Função de exemplo.



(a) Curvas de nível da função



(b) Valores da função



(c) Norma do gradiente da função

Figura 20: Função de exemplo.

Conclusão

O principal objetivo do presente trabalho dizia respeito ao estudo do método dogleg e sua implementação em MATLAB. Conseguimos atingir tal objetivo e, buscar resultados significativos com este método, que resolveu, de forma consideravelmente rápida e segura, problemas clássicos na área de otimização.

A realização dos testes práticos com os métodos citados neste trabalho, possibilitou ao aluno um aprimoramento em sua forma de programar algoritmos no software MATLAB, tomando sempre os devidos cuidados relativos aos custos computacionais.

Foi uma experiência que uniu os conceitos apresentados durante o curso de Bacharelado em Matemática e Computação Científica, principalmente de disciplinas como Programação Não Linear e Álgebra Linear o que, acreditamos tornar esta uma experiência válida como um Trabalho de Conclusão de Curso.

Referências

- [1] NOCEDAL, J.; WRIGHT, S. J. *Numerical Optimization*. New York: Springer-Verlag New York, Inc., 1999. (Springer Series in Operations Research).
- [2] GONZAGA, C. C. *Algoritmos de Pontos Interiores Para Programação Linear*. Rio de Janeiro: Instituto de Matemática Pura e Aplicada, 1989. (17º Colóquio Brasileiro de Matemática).
- [3] CONN, A. R.; GOULD, N. I. M.; TOINT, P. L. *Trust-Region Methods*. Philadelphia: Society for Industrial and Applied Mathematics, 2000. (MPS-SIAM Series on Optimization).
- [4] KELLEY, C. T. *Iterative Methods for Optimization*. Philadelphia: Society for Industrial and Applied Mathematics, 1999. (Frontiers in Applied Mathematics).
- [5] POLIAK, B. T. *Introduction to Optimization*. New York: Optimization Software, Inc., Publications Division, 1987. (Translations Series in Mathematics and Engineering).
- [6] ZANARDINI, R. A. D. *Método de Newton*. Disponível em <http://www.cesec.ufpr.br/zanardin/Arquivos/ProgramacaoNaoLinear/MetodoDeNewton.pdf>. Acesso em : 26 maio 2005.: UFPR, 2003.
- [7] DENNIS, J. E.; SCHNABEL, R. B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Philadelphia: Society for Industrial and Applied Mathematics, 1996. (Classics in Applied Mathematics).

APÊNDICE A - Programação em MATLAB

Os testes apresentados no capítulo 5 foram implementados no software MATLAB. Apresentamos, aqui, os arquivos que foram utilizados para gerar os resultados obtidos.

A.1 Arquivo go.m

```

1 % TRUST - Regiao de confianca
2 %
3 % Arquivo de inicialização para o algoritmo de região de confiança
4 % utilizando o método dogleg para o calcular o subproblema de
5 % região de confiança
6
7 % limpar todas as variáveis
8 clear all
9
10 % atribuir variáveis globais para plotar gráficos
11 % e gerar contadores
12 global X G F D contadores
13
14 % contadores:
15 % (1) iterações
16 % (2) cálculos de função
17 % (3) cálculos de gradientes
18 % (4) cálculos de hessianas
19 % (5) cálculos de  $-B \setminus g$ 
20 contadores=zeros(1,5);
21
22 % buscar dados
23 dados;
24
25 % inicializar arquivo trust.m
26 trust;
```

A.2 Arquivo dados.m (exemplo)

```

1 % Arquivo dados.m
2
3 fun = 'exemplo'; % determinar função
4 n = 2; % ordem do vetor x
5 x = [0.7067;-3.2672]; % ponto inicial
6 deltamax=10; % raio máximo para região de confiança
7 epsilon= 1e-5; % valor de epsilon
8 delta = 1; % raio inicial para região de confiança
9 eta = 1/8; % valor de eta
10 kmax=100; % número máximo de iterações
```

A.3 Arquivo trust.m

```

1 % TRUST - Regiao de confianca
2 %
3
4 % vetores F, G, D e X para guardar resultados para plotar
5 F=zeros(1,kmax);
6 G=F;
7 D=F;
8 if n<=10, X=zeros(n,kmax);end;
9
10 % inicializa variável k
11 k = 1;
12
13 % guarda o ponto x (pra plotar)
14 if n<=10, X(:,k)=x; end;
15
16 % valor da funcao no ponto x
17 % contadores(2) calcula o número de cálculos da função no algoritmo
18 F(1) = feval(fun,x,0); contadores(2)=contadores(2)+1;
19
20 % delta inicial (pra plotar)
21 D(1)=delta;
22
23 % hessiana de f ou aproximacao no ponto x
24 % contadores(4) calcula o número de cálculos da hessiana no algoritmo
25 h = feval(fun,x,2); contadores(4)=contadores(4)+1;
26
27 % gradiente de f no ponto x
28 % contadores(3) calcula o número de cálculos do gradiente no algoritmo
29 g = feval(fun,x,1); contadores(3)=contadores(3)+1;
30
31 % norma do gradiente de f no ponto x (pra plotar)
32 G(k) = norm(g);
33
34 % cabeçalho dos resultados que serão apresentados na tela
35 disp(' k |grad| f raio aceitou fronteira')
36
37
38 while (G(k) > epsilon) % condição de parada
39     if k>kmax
40         % encerra o algoritmo e informa que ultrapassou k máximo
41         disp('Número de iterações alto!');
42         break;
43     end;
44
45     % cálculo do subproblema de região de confiança utilizando
46     % o método dogleg
47     pk = dogleg(h,g,delta);
48     normpk = norm(pk);
49
50     % valor da função no ponto (x+pk) : candidato a novo passo
51     % incremento de contadores(2)
52     F(k+1) = feval(fun,x+pk,0); contadores(2)=contadores(2)+1;
53
54
55     ared = F(k)-F(k+1); % redução real
56     pred = -g'*pk -(1/2)*pk'*h*pk; % redução predita pelo modelo
57     rho = ared/pred;
58
59     if rho < 1/4
60         % candidato a novo passo não é bom,
61         % redução da região de confiança
62         delta = (1/4) * normpk;
63     else
64         if rho > (3/4) & normpk>0.99* delta
65             % candidato a novo passo está na fronteira da região de
66             % confiança e é um bom passo
67             % dobramos o tamanho da região de confiança, cuidando pra
68             % que não ultrapasse o raio máximo estipulado
69             delta = min(2*delta,deltamax);

```

```

70     end
71 end
72
73 k = k+1;
74 D(k)=delta;    % informando o raio (pra plotar)
75
76 if rho > eta
77     % passo aceito
78     aceitou(k-1)=1; % valor guardado pra imprimir na tela
79     x = x + pk;    % atualizamos o ponto
80
81     % atualiza gradiente, hessiana e norma do gradiente no
82     % novo ponto x
83     % incremento nos contadores
84     h=feval(fun,x,2); contadores(4)=contadores(4)+1;
85     g=feval(fun,x,1); contadores(3)=contadores(3)+1;
86     G(k) = norm(g);
87     if n<=10, X(:,k)=x; end;
88 else
89     % passo não foi aceito
90     aceitou(k)=0; % valor guardado pra imprimir na tela
91     F(k) = F(k-1); % manter o valor da função pra próxima iteração
92     G(k) = G(k-1); % manter o valor do gradiente pra próxima iteração
93     if n<=10, X(:,k)=X(:,k-1); end;
94 end
95
96 % se o passo foi aceito, verifica se o novo ponto está na fronteira
97 % e guarda a informação pra apresentar na tela
98 if (normpk>0.99* D(k-1)) & (aceitou(k-1)==1)
99     fronteira = 1;
100 else
101     fronteira = 0;
102 end
103
104 % imprime dados da iteração atual
105 fprintf('%3.0f %12f %12f %12f %3.0f %3.0f\n', ...
106     k-1,G(k-1),F(k-1),D(k-1),aceitou(k-1),fronteira);
107 end
108
109 % imprime dados da última iteração
110 fprintf('%3.0f %12f %12f %12f %3.0f %3.0f\n', ...
111     k,G(k),F(k),D(k),0,0);
112
113 % imprime valor achado
114 fprintf('\nValor achado:')
115 fprintf('%12f\n',x);
116
117 % imprime a quantidade de cálculos de função, de gradiente,
118 % de hessiana, de -B\g e de passos aceitos
119 fprintf('\nCalculos de funcao: %3.0f\n',contadores(2))
120 fprintf('Calculos de gradiente: %3.0f\n',contadores(3))
121 fprintf('Calculos de hessiana: %3.0f\n',contadores(4))
122 fprintf('Calculos de -B\g: %3.0f\n',contadores(5))
123 fprintf('Passos aceitos: %3.0f\n', sum(aceitou))
124
125 % informações pra fazer figuras
126 F = F(1:k);
127 G = G(1:k);
128 if n<=10,X=X(:,1:k);end;
129 metodo='Regiao de confianca com dogleg';
130
131 % geração de figuras
132 figuras

```

A.4 Arquivo dogleg.m

```

1 function pd=dogleg(B,g,delta)
2 % DOGLEG - Minimiza uma quadrática na bola
3 %
4 %   min g^T d + 1/2 d^T B d
5 %   s.a ||d|| < delta
6 %
7 %   pd = DOGLEG(B,g,delta)
8 %
9 %   Parâmetros de ENTRADA
10 %   B       - matriz n x n (definida positiva)
11 %   g       - vetor n x 1
12 %   delta   - raio da região de confiança
13 %
14 %   Parâmetro de SAÍDA
15 %   pd      - solução aproximada n x 1
16 %
17
18 % definição de variáveis globais
19 % X, G e F são variáveis pra plotar os gráficos
20 % contadores guarda as quantidades de cálculos de
21 % função, gradiente, hessiana, -B\g
22 global X G F contadores
23
24 curv=g'*B*g;           % curvatura
25 normg=norm(g);
26
27 % se a curvatura for negativa, pd estará na direção de máximo
28 % declive com norma delta
29 if curv<=0
30     pd = -delta*g / normg;
31 else
32     pu = -g'*g/curv *g;   % cálculo do ponto de Cauchy
33     normpu = norm(pu);
34
35     if normpu >= delta
36         % ponto de Cauchy fora da região de confiança
37         % pd estará na direção de máximo declive com norma delta
38         pd = -delta*g / normg;
39     else
40         % ponto de Cauchy dentro da região de confiança
41
42         % calculamos os autovalores de B
43         % se todos eles forem positivos significa que
44         % B é definida positiva
45         eigB = eig(B);
46         mineigB = min(eigB);
47         if mineigB<=0
48             % matriz B não é definida positiva
49             % pd é o ponto de Cauchy
50             disp('A matriz B não é definida positiva!');
51             pd=pu;
52         else
53             % cálculo do passo de Newton
54             % incremento de contadores(5)
55             pb = -B\g; contadores(5)=contadores(5)+1;
56             normpb = norm(pb);
57
58             if normpb <= delta
59                 % ponto de Newton dentro da região de confiança
60                 % pd é o ponto de Newton
61                 pd = pb;
62             else
63                 % ponto de Newton fora da região de confiança
64                 % pd é o ponto de intersecção entre a fronteira
65                 % da região de confiança e o segmento de reta
66                 % que une pu a pb
67                 normpbpu=norm(pb -pu);
68                 a = normpbpu^2;
69                 b = 2*pu'*(pb-pu);

```



```

70         c = norm(pu)^2 - delta^2;
71         lambda = (-b+sqrt(b^2-4*a*c))/(2*a);
72         pd = pu + lambda*(pb -pu);
73     end
74 end
75 end
76 end

```

A.5 Arquivo figuras.m

```

1  % Figuras.m
2  % gera figuras usando os vetores F,G e X
3  % para gerar figuras sobrepostas, verifica se existe 'sobreposta'
4
5  cores=['r-'; 'g-'; 'm-'; 'c-'; 'y-'; 'k-'; 'b-'];
6  if ~exist('sobreposta'), sobreposta=0; end;
7  figure(1);
8  if sobreposta==0, clf; end;
9  hold on;
10 figure(2);
11 if sobreposta==0, clf; end;
12 if n==2,
13     figure(3);
14     if sobreposta==0, clf; end;
15     hold on;
16 end;
17 sobreposta=sobreposta+1;
18 cor=cores(sobreposta,:);
19 npontos=size(F,2);
20 indices=1:npontos;
21
22 figure(1);
23 plot(indices,F,cor);
24 title(['Funcao objetivo ' metodo]);
25 figure(2);
26 G=G+1e-10;
27 semilogy(indices,G,cor);
28 hold on
29 title(['Norma do gradiente ' metodo]);
30 if n==2
31     vmin=F(end);
32     vmax=F(1);
33     del=(vmax-vmin)/30;
34     U=vmin:del:vmax;
35
36     figure(3);
37     otimo=X(:,npontos);
38     for i=1:npontos
39         erro=norm(X(:,i)-otimo);
40         if erro>0.01,
41             result=level(fun,X(:,i),100,'b',0.3);
42         end;
43     end
44     plot(X(1,:),X(2,:),cor,X(1,:),X(2,:),'.k')
45     title(['Trajetoria ' metodo])
46     axis equal
47     axis manual
48     for i=1:npontos
49         [lixo,lixo,botao]=ginput(1);
50         if botao>1, break; end;
51         xx=X(:,i); temp=xx(1); xx(1)=xx(2); xx(2)=temp;
52         [h1,h2]=ellipse(xx,[1 0;0 1],D(i));
53         plot(h2,h1,'-m');
54     end
55 end;

```

A.6 Arquivos de funções

A.6.1 Arquivo pen.m

```

1 function z=pen(x,ordem)
2 %function y=pen(x)
3 %função penalidade logarítmica para o problema em ld.mat
4 % Problema: min c'x s. a A'x<=b.
5 global A b c alfapen
6 if nargin==1, ordem=0; end
7 alfapen=10;
8 if isempty(A), load ld; disp('carreguei'); end;
9 s=b-A'*x;
10
11 switch ordem
12 case 0 % cálculo da função
13     if s>0, z=alfapen*c'*x-sum(log(s));
14     else
15         z=+inf;
16     end;
17
18 case 1 % cálculo do gradiente
19     if s>0,
20         sinv=1./s;
21         z=alfapen*c+ A*sinv;
22     else
23         z=[];
24     end;
25
26 case 2 % cálculo da hessiana
27     if s>0,
28         Sinv2=diag(1./(s.^2));
29         z= A*Sinv2*A';
30     else
31         z=[];
32     end;
33
34 otherwise error('Ordem da derivada inexistente');
35 end
36
37

```

A.6.2 Arquivo banana.m

```

1 function z=pen(x,ordem)
2 %function y=pen(x)
3 %função penalidade logarítmica para o problema em ld.mat
4 % Problema: min c'x s. a A'x<=b.
5 global A b c alfapen
6 if nargin==1, ordem=0; end
7 alfapen=10;
8 if isempty(A), load ld; disp('carreguei'); end;
9 s=b-A'*x;
10
11 switch ordem
12 case 0 % cálculo da função
13     if s>0, z=alfapen*c'*x-sum(log(s));
14     else
15         z=+inf;
16     end;
17
18 case 1 % cálculo do gradiente
19     if s>0,
20         sinv=1./s;
21         z=alfapen*c+ A*sinv;

```

```

22     else
23         z=[];
24     end;
25
26 case 2 % cálculo da hessiana
27     if s>0,
28         Sinv2=diag(1./(s.^2));
29         z= A*Sinv2*A';
30     else
31         z=[];
32     end;
33
34     otherwise error('Ordem da derivada inexistente');
35 end
36
37

```

A.6.3 Arquivo exemplo.m

```

1 function y=exemplo(x,ordem)
2 % function y=exemplo(x,ordem)
3 %     f(x1,x2) = -10 * x1^2 + 10 * x2^2 + 4 * sin(x1* x2) - 2 * x1 + x1^4;
4
5 if nargin==1, ordem=0; end
6 x1=x(1);
7 x2=x(2);
8 n=length(x);
9 if n~=2, error('O comprimento de x deve ser igual a 2 para esta função!'); end
10
11 switch ordem
12 case 0 % cálculo da função
13     y = -10 * x1^2 + 10 * x2^2 + 4 * sin(x1* x2) - 2 * x1 + x1^4;
14
15 case 1 % cálculo do gradiente
16     y(1,1) = -20*x1+4*cos(x1*x2)*x2-2+4*x1^3;
17     y(2,1) =20*x2+4*cos(x1*x2)*x1;
18
19 case 2 % cálculo da hessiana
20     y(1,1) = -20-4*x2^2*sin(x1*x2)+12*x1^2;
21     y(1,2) = 4*cos(x1*x2) - 4*x1*x2*sin(x1*x2);
22     y(2,1) = y(1,2);
23     y(2,2) = 20 - 4*(x1^2)*sin(x1*x2);
24
25     otherwise error('Ordem da derivada inexistente');
26 end

```
