Esta monografia foi julgada adequada como **TRABALHO DE CONCLUSÃO DE CURSO** no curso de Matemática - Habilitação Licenciatura e aprovada em sua forma final pela Banca Examinadora designada pela Portaria nº. 33/SCG/04.

---

Profª. Carmen Suzane Comitre Gimenez

Professora responsável pela disciplina

Banca examinadora:

---

Profº. Clóvis Caesar Gonzaga

Orientador

---

Profº. Lício Hernanes Bezerra

---

Profº. Mário César Zambaldi

**JANE BAUER**

# Implementação de Algoritmos de Programação Não Linear

Trabalho de Conclusão de Curso apresentado ao
Curso de Matemática - Habilitação Licenciatura
Departamento de Matemática
Centro de Ciências Físicas e Matemáticas
Universidade Federal de Santa Catarina

Orientador: Profº. Clóvis Caesar Gonzaga

Florianópolis
Julho 2004

# Agradecimentos

A Deus, pelo dom da vida;

À minha família, em especial aos meus pais Adolfo e Eni e ao meu irmão Adriano pelo apoio e compreensão em todos os momentos;

A meus professores, em especial ao meu orientador Clóvis Caesar Gonzaga, por ter deixado a minha disposição um pouco do seu vasto conhecimento, sua dedicação e sua paciência em todos os momentos;

À Márcia Regina Vanti pela valiosa colaboração e ajuda para a elaboração desde trabalho;

A meus colegas de graduação, em especial à minha grande amiga Jussara Brigo, por todos os momentos vividos juntos e por todo o apoio e incentivo nas horas de alegria e de tristeza;

A meus amigos de festa: Leandro, Marciano, Simone e Luciano, por sempre me alegrarem mesmo nas horas mais difícieis;

Enfim, a todos que de alguma maneira me ajudaram a dar mais este passo na minha vida...

Dedico este trabalho única e
exclusivamente a meus pais, Adolfo e
Eni, que são os grandes reponsáveis,
além de mim, pelo êxito em minha
vida profissional.

# Sumário

# Introdução

Neste trabalho, vamos mostrar de maneira simples alguns dos algoritmos de Programação Não Linear, de modo que, qualquer pessoa com um pouco de conhecimento em Álgebra Linear e Cálculo possa compreendê-los.

O objetivo desses algoritmos é minimizar uma função não linear, ou seja, encontrar um ponto $\bar{x}$ tal que $f(\bar{x}) < f(x)$ para todo $x \in \Re$. Estudaremos algoritmos de minimização em $\Re$, em $\Re^n$, e em $\Re^2$.

Os modelos usados na Programação Linear, são, como o nome diz, lineares. Em grande parte das aplicações, modelos lineares refletem apenas aproximações dos modelos reais. Fenônemos físicos ou econômicos são geralmente melhor representados por modelos não lineares. Por isso a importância da Programação Não Linear. O problema central da Programação Não Linear é minimizar uma função não linear, encontrando uma solução com uma certa precisão.

Descreveremos os algoritmos, explicando como e porque funcionam. Para todos os algoritmos faremos a implementação em Matlab.

A maneira como se desenvolve o trabalho é a seguinte:

No primeiro capítulo estudaremos métodos de busca na reta (Seção Áurea e Armijo).

No segundo, veremos o método de Cauchy que é um método de minimização em $R^n$. Faremos uma comparação entre o método de Cauchy usando Seção Áurea e o método de Cauchy usando Armijo.

No terceiro capítulo faremos um estudo do método de Newton, também um método de minimização em $\Re^n$, e faremos a mesma comparação do capítulo anterior.

No último capítulo, estudaremos um pouco de busca bidirecional. Este método usa as vantagens do método de Cauchy e Newton, fazendo a minimização sobre o subespaço gerado pelas direções de Cauchy e de Newton.

No primeiro capítulo, apresentamos exemplos de funções em $\Re$, mostrando as iterações de cada método. Nos outros capítulos, apresentamos exemplos de funções mostrando as suas curvas de nível e as iterações de cada método.

# Capítulo 1

# Métodos de Busca na Reta

O objetivo deste capítulo é estudar métodos para a minimização de uma função $\theta : \Re_+ \to \Re$ unimodal, ou seja,

$$\underset{x \in \Re_+}{\text{minimizar}}\, \theta(x) \tag{1.1}$$

Descreve-se dois métodos de busca unidirecional. O método da seção áurea, que trata da busca unidirecional exata e o método de armijo, que procura uma boa redução da função ao longo da direção.

## 1.1 O Método da Seção Áurea

O algoritmo de seção áurea inicia com uma busca que identifica um intervalo [a,b] no qual a função possui um mínimo. Depois, este intervalo inicial é reduzido até que o mínimo seja identificado com a precisão desejada. O método aplica-se a funções unimodais.

**Definição:** $\theta : \Re_+ \to \Re$ é uma função **unimodal** se e somente se existe um $\bar{x} \in (0, +\infty)$ tal que $\theta(x) \geqslant \theta(\bar{x})$ em $[0, \bar{x}]$ e decresce, e $\theta(x) \geqslant \theta(\bar{x})$ em $[\bar{x}, +\infty)$ e cresce. Em outras palavras, $\theta$ é estritamente decrescente à esquerda de $\bar{x}$, e estritamente crescente à direita de $\bar{x}$.

Nesta seção suporemos que é dada uma função $\theta : \Re_+ \to \Re$ unimodal.

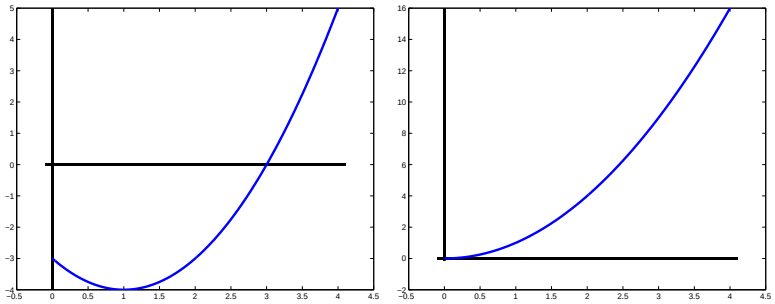

Figura 1.1: Exemplos de funções unimodais: $\theta = x^2 - 2x - 3$ e $\theta = x^2$

### 1.1.1 Determinação de um Intervalo Inicial

Por hipótese, $\theta$ é unimodal e portanto tem um único ponto de mínimo $\xi \in \Re_+$. Precisamos encontrar um intervalo $[a, b]$, que contém $\xi$.

Iniciamos com um palpite inicial $p > 0$ e com $a = 0$.

Se a função cresceu no ponto $p$, ou seja, $\theta(p) > \theta(0)$, o intervalo [0 , p] contém o ponto de mínimo. Fazemos $b = p$ e o algoritmo pára com $[0, b]$.

Caso contrário, expandimos o intervalo, fazendo $b = 2p$ e $a = p$, até que $\theta(b) > \theta(a)$.

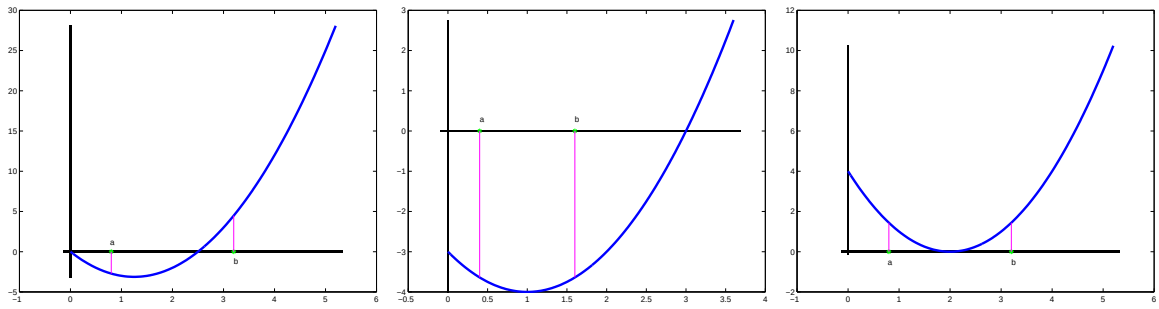


Figura 1.2: Determinação de um intervalo inicial

**Algoritmo:**

Se $\theta(p) > \theta(0), a = 0; b = p$; FIM

Senão

$k = p$

$b = 2p$

Enquanto $\theta(b) \leqslant \theta(k)$, fazemos:

$a = k$

$k = b$

$b = 2k$

Em todas as iterações, $\theta(k) < \theta(a)$, logo $a < \xi$. Quando o processo pára, $\theta(b) \geq \theta(k)$ e portanto $b \geq \xi$. Disso concluímos que $\xi \in [a, b]$.

Após identificar um intervalo que contém o mínimo, podemos passar à fase de refinamento deste intervalo.

### 1.1.2 O Algoritmo da Seção Áurea dado um Intervalo Inicial

Nesta fase temos $[a, b]$ (intervalo inicial), definimos $\epsilon > 0$ como o erro máximo tolerado e o tamanho do intervalo $l = b - a$.

O objetivo desta fase é encontrar $\bar{x}\ \epsilon\ [a, b]$ tal que $|\bar{x} - \xi| < \varepsilon$, ou seja, queremos encontrar $\xi$ aproximadamente com precisão de pelo menos $\varepsilon$.

Esse algoritmo necessita de dois pontos $m_1$ e $m_2$ interiores ao intervalo $[a, b]$ em cada iteração, dividindo assim $l$ em três partes.

$$m_1 = a + vl$$
$$m_2 = a + wl$$

Escolhemos $v$ e $w$ segundo a seção áurea do segmento $l = b - a$, ou seja,

$$v + w = 1 \text{ e } v = w^2$$

$$\text{Então } w^2 + w = 1$$

$$w = \frac{\sqrt{5} - 1}{2} \text{ (o caso negativo não é considerado)}$$

$$w \cong 0,62 \text{ e } v \cong 0,38.$$

$$\text{Logo: } m_1 = a + 0,38l \text{ e } m_2 = a + 0,62l$$

O algoritmo, a cada iteração, reduz o tamanho do intervalo $[a, b]$, sendo os pontos $v$ e $w$ definidos segundo a seção áurea do segmento $l = b - a$.

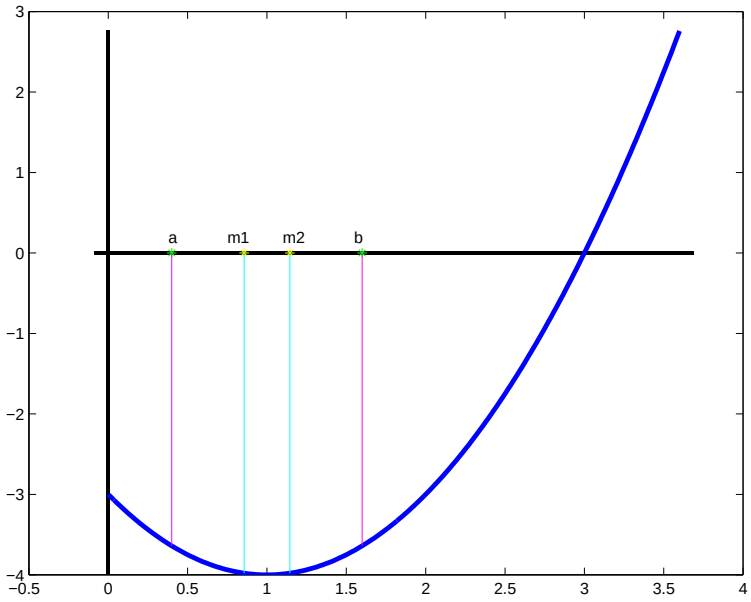


Figura 1.3: Intervalo inicial e os pontos intermediários m1 e m2 da primeira iteração

Suponhamos que em uma iteração temos os pontos a, $m_1$, $m_2$ e b, e na iteração seguinte, $\bar{a}, \bar{m_1}, \bar{m_2}$ *e* $\bar{b}$.

Então, $\bar{b} - \bar{a} = w(b - a)$.

Sempre teremos $\bar{m_2} = m_1$ ou $\bar{m_1} = m_2$, pois $v = w^2$. Desta forma será necessário apenas um cálculo de função a cada iteração. Daí, a razão pela qual usamos a seção áurea para definir $v$ e $w$.

O método da seção áurea fica assim:

$l = b - a$;

Enquanto $l > 2\varepsilon$, ou seja, enquanto não se alcançar a precisão desejada, fazemos:

$m_1 = a + 0,38l$

$m_2 = a + 0,62l$

E para reduzir o intervalo:

Se $\theta(m_1) \leq \theta(m_2)$

$b = m_2$

$m_2 = m_1$

$l = b - a$

$$m_1 = a + 0,38l$$

Senão

$$a = m_1$$

$$m_1 = m_2$$

$$l = b - a$$

$$m_2 = a + 0,62l$$

O algoritmo pára quando $l \leq 2\varepsilon$, e teremos $\bar{x} = \frac{(a+b)}{2}$.

$\bar{x}$ é o ponto de mínimo com precisão de pelo menos $\varepsilon$.

Na primeira iteração, sabemos pela fase anterior, que $\xi \in [a, b]$. A cada iteração, existem duas possibilidades:

Se $\theta(m_1) \leq \theta(m_2)$, como $b > m_2$ e a função é unimodal, $\theta(b) > \theta(m_2)$. Logo, $\xi \in [a, m_2]$. Assim, define-se o novo intervalo $[a, b]$ contendo $\xi$.

Se $\theta(m_1) > \theta(m_2)$, como $a < m_1$ e a função é unimodal, $\theta(a) > \theta(m_1)$. Logo, $\xi \in [m_1, b]$. Assim define-se o novo intervalo $[a, b]$ contendo $\xi$.

Ou seja, a cada iteração do método, o ponto de mínimo $\xi \in [a, b]$.

A Figura (1.4) ilustra a redução do intervalo $[a, b]$ em 4 iterações.

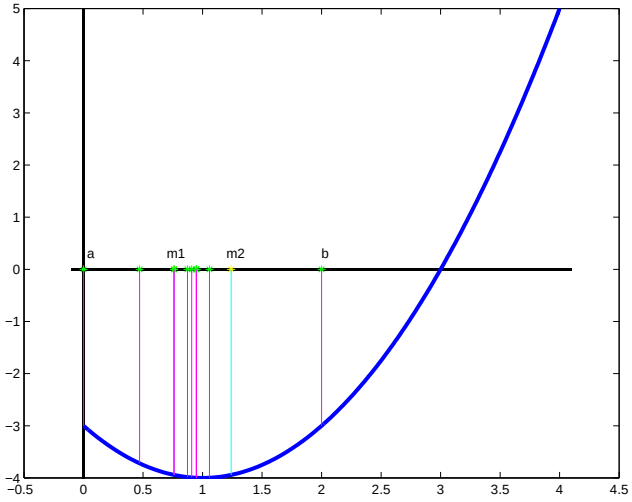


Figura 1.4: Diminuição do intervalo após algumas iterações

### 1.1.3 Implementação do Algoritmo de Seção Áurea em Matlab

Em Matlab o algoritmo fica assim:

%Sejam dados a função $\theta$ e a precisão $\epsilon$, queremos encontrar um intervalo $[a, b]$ contendo o mínimo.

p = 1; % palpite inicial.

a = 0;

k = p;

b = 2*p;

WHILE

$\theta(b) < \theta(k)$;

a = k;

k = b;

b = 2*k;

END

%Definido $[a, b]$, vamos refinar a busca por Seção Áurea.

$it = 0$; %contador de iterações.

epsilon = $\varepsilon$; %precisão desejada

$l = b - a$; % intervalo para a busca.

$m_1 = a + 0.38 * l$; $m_2 = a + 0.62 * l$;

$\theta 1 = \theta(m_1)$; $\theta 2 = \theta(m_2)$;

WHILE

$l > 2 * \epsilon$;

IF

$\theta 1 > \theta 2$;

$a = m_1$;

$m_1 = m_2$;

$\theta 1 = \theta 2;$

$l = b - a;$

$m_2 = a + 0.62 * l;$

$\theta 2 = \theta(m_2);$

ELSE

b $= m_2;$

$m_2 = m_1;$

$\theta 2 = \theta 1;$

$l = b - a;$

$m_1 = a + 0.38 * l;$

$\theta 1 = \theta(m_1);$

END

$l = b - a;$

$it = it + 1;$

END

$\bar{x} = \frac{(a+b)}{2}$ % $\bar{x}$ é o ponto de mínimo da função com precisão de pelo menos $\varepsilon$.

**A Seção Áurea**

A seção áurea é um dos mais famosos problemas da geometria, estando presente no corpo humano, nas flores, nos televisores, revistas, livros, etc.
O nome seção áurea foi dado por Leonardo da Vinci (1452 - 1519), genial artista florentino, autor de Gioconda e da Ceia.
Quando um segmento AB está dividido por um ponto M tal que:

$$\frac{AM}{AB} = \frac{MB}{AM}$$

diz-se que AB está dividido em média e extrema razão. Essa divisão chama-se seção áurea do segmento AB. A construção da seção áurea, equivale à resolução de uma equação quadrática:
Seja AM=x e AB=a.
Então pela propriedade da seção áurea, temos:
$\frac{x}{a} = \frac{a-x}{x}$
Logo:
$x^2 + ax - a^2 = 0$
$x = \frac{-a+a\sqrt{5}}{2}$ Como $a > 0$, o caso negativo não é considerado, e
$x = a(\frac{\sqrt{5}-1}{2})$
Assim, um segmento AB com comprimento $a$, tem o seu áureo de comprimento $a(\frac{\sqrt{5}-1}{2})$.

## 1.2 O Método de Armijo

Nesta seção vamos estudar o método de Armijo para encontrar um passo de descida de uma função, ou seja, queremos encontrar um ponto em que a função decresce "bastante" em relação ao ponto inicial. Esta abordagem é bastante utilizada e de simples implementação.

Seja $\theta : \Re_+ \to \Re$ uma função continuamente diferenciável com $\theta'(0) < 0$, não necessariamente unimodal.

O método de Armijo evita passos grandes com pouco decréscimo da função, porém

passos pequenos podem ocorrer. Para isso temos que encontrar $\bar{\lambda}$ tal que:

$$\theta(\bar{\lambda}) \leq \theta(0) + \alpha\bar{\lambda}\theta'(0).$$

onde $\alpha \in (0, 1)$, (em geral usamos $\alpha = 0.5$).

A condição exige que o decréscimo na função $\theta$ seja proporcional ao tamanho do passo.

Ressaltamos que o método de Armijo não minimiza a função, apenas tenta achar uma boa redução desta.

Este método avalia a derivada da função no ponto $\lambda = 0$ e define metade (se $\alpha = 0,5$) desta aproximação linear. Um ponto $\bar{\lambda}$ é "melhor" que o ponto inicial se satisfaz a condição de decréscimo suficiente $\theta(\lambda) \leq \theta(0) + \alpha\lambda\theta'(0)$.

A Figura (1.5) ilustra, para uma função $\theta$, a aproximação linear no ponto 0 e metade desta aproximação.

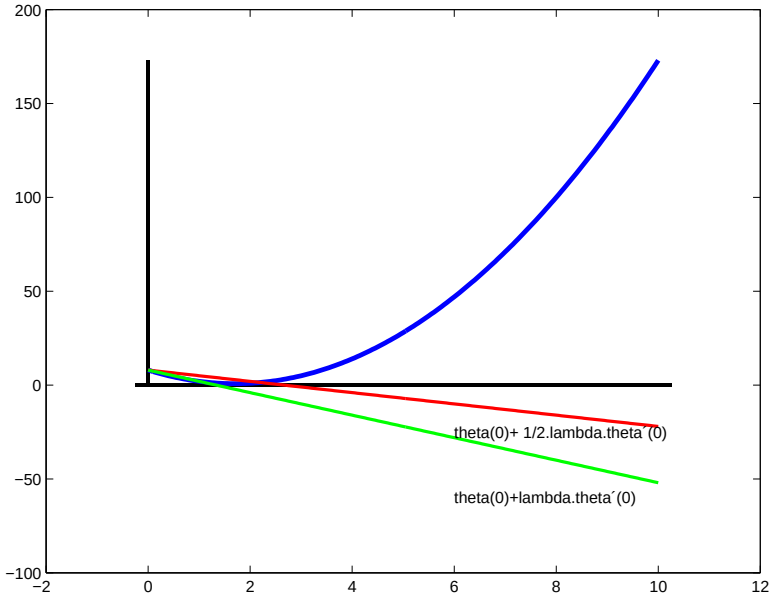


Figura 1.5: Aproximação linear da função em $x_0$ (em verde) e a metade da aproximação linear em $x_0$ (em vermelho)

### 1.2.1 O Algoritmo de Armijo

É dado um valor para $\alpha$ (em geral 0.5) e calculamos o derivada da função em 0, tal que $\theta'(0) < 0$.

Sabemos que, para $\lambda$ pequeno:

$$\theta(\lambda) \backsimeq \theta(0) + \lambda\theta'(0).$$

(Aproximação linear da função em 0).

Se $\theta'(0) < 0$, para $\lambda$ suficientemente pequeno:

$$\theta(\lambda) < \theta(0) + \frac{\lambda}{2}\theta'(0)$$

O método procura um valor para $\lambda$ que satisfaça a desigualdade acima.

Inicia-se o algoritmo com um palpite "grande" para o passo: $\lambda = \rho$

Se para esse valor de $\lambda$ obtermos

$$\theta(\lambda) > \theta(0) + \frac{\lambda}{2}\theta'(0)$$

reduzimos o valor de $\lambda$: $\lambda = \frac{\lambda}{2}$.

O procedimento se repete até que

$$\theta(\lambda) < \theta(0) + \frac{\lambda}{2}\theta'(0),$$

ou seja, enquanto o valor da funçao em $\lambda$ for maior que a metade da aproximação linear da função em $x_0$, reduzimos o valor de $\lambda$.

O algoritmo de Armijo em forma compacta fica assim:

$\lambda = \rho$ (palpite inicial), $\alpha = 0.5$

Enquanto $\theta(\lambda) > \theta(0) + \frac{\lambda}{2}\theta'(0)$

$\lambda = \frac{\lambda}{2}$.

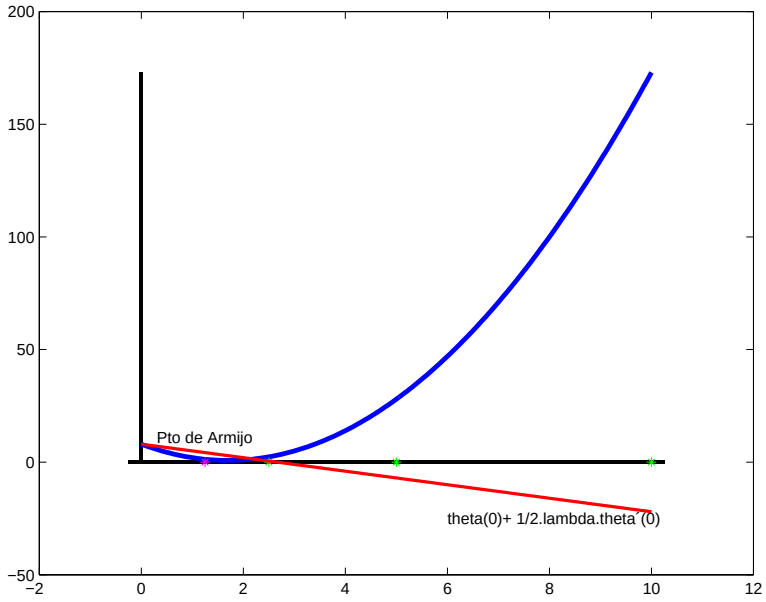


Figura 1.6: Iterações do Método: O ponto em rosa satisfaz a condição de Armijo

### 1.2.2 Implementação do Algoritmo de Armijo em Matlab

Em matlab o algoritmo fica assim:

% Definimos a função $\theta$ na qual queremos fazer a busca unidirecional , $\alpha \in (0,1)$, e um palpite $\rho$ (em geral usamos $\rho = 1$).

$it = 0$; % contador de iterações.

$\lambda = \rho$;

WHILE

$\theta(\lambda) > \theta(0) + \alpha\lambda\theta'(0)$;

$\lambda = \frac{\lambda}{2}$;

END

$it = it + 1$;

### 1.2.3 Buscas Unidirecionais em $\Re^n$

Seja dadas $f : \Re^n \to \Re$, um ponto inicial $x_0$ e uma direção $d \in \Re^n$, o problema de busca será:

$$\underset{\lambda \geq 0}{\text{minimizar}} \, f(x_0 + \lambda d)$$

Para este problema podem ser utilizados os métodos de busca unidirecional estudados nas seções anteriores, definindo

$\lambda \geq 0 \mapsto \theta(\lambda) = f(x_0 + \lambda d) \quad \text{e} \quad \theta'(0) = \nabla f(x_0)^T d.$

# Capítulo 2

# O Método de Cauchy

Neste capítulo iremos estudar o método de Cauchy, um dos mais antigos dos algoritmos de minimização, desenvolvido por Cauchy em 1847.

Este método, a cada iteração, usa a direção de máximo declive da função em um ponto dado.

O problema de minimização irrestrita é:

$$\underset{x\in\Re^n}{\text{minimizar}}\, f(x) \tag{2.1}$$

onde $f : \Re^n \to \Re$ é diferenciável com derivadas contínuas (função de classe $C^1$.)

Ou seja, queremos encontrar, se existir, $\bar{x} \in \Re^n$ tal que $f(\bar{x}) = min\{f(x), x \in \Re^n\}$.

## 2.1 Condições Necessárias de Otimalidade

Seja $f : \Re^n \to \Re$ de classe $C^1$.

Nosso objetivo é encontrar um ponto $\bar{x} \in \Re^n$ tal que $(\forall x \in \Re^n)\ f(x) \geqslant f(\bar{x})$.

O ponto $\bar{x}$ é um minimizador local se existe uma vizinhança $V$ de $\bar{x}$ tal que

$$(\forall x \in V)\ f(x) \geqslant f(\bar{x}).$$

**Teorema 1 (Condição de otimalidade de primeira ordem).** *Se $\bar{x}$ é um minimizador local de (2.1), então $\nabla f(x) = 0$*

**Demonstração:** Suponha que $\bar{x}$ é um minimizador local. Seja $d \in \Re^n$ uma direção arbitrária, $d \neq 0$. Como $f$ é diferenciável, temos:

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + \sigma(\bar{x}, \lambda)$$

(Aproximação de primeira ordem)

com $\lim_{\lambda \to 0} \frac{\sigma(\bar{x}, \lambda)}{\lambda} = 0$

Então, para $\lambda \neq 0$,

$$\nabla f(\bar{x})^T d = \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda} - \frac{\sigma(\bar{x}, \lambda)}{\lambda}$$

Por definição de minimizador local, para $\lambda$ suficientemente pequeno, temos:

$$f(\bar{x} + \lambda d) - f(\bar{x}) \geqslant 0$$

Tomando limites, obtemos:

$$\nabla f(\bar{x})^T d \geqslant 0$$

Repetindo para a direção $-d$, obtemos

$$\nabla f(\bar{x})^T d \leq 0,$$

e portanto

$$\nabla f(\bar{x})^T d = 0.$$

Logo, para qualquer direção $d \in \Re^n$,

$$\nabla f(\bar{x})^T d = 0$$

e portanto $\nabla f(\bar{x}) = 0$. ■

**Teorema 2 (Condição de otimalidade de segunda ordem).** *Se $f$ é de classe $C^2$, e $\bar{x}$ um minimizador local, então a hessiana $Hf(\bar{x})$ é semi definida positiva.*

**Demonstração:** Suponha que $\bar{x}$ é minimizador local e considere uma direção arbitrária $d \in \Re^n$.

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + \frac{\lambda^2}{2} d^T H f(\bar{x}) d + \sigma_2(\lambda)$$

(Aproximação de segunda ordem)

com $\lim_{\lambda \to 0} \frac{\sigma_2(\lambda)}{\lambda^2} = 0$

Pelo teorema anterior, $\nabla f(\bar{x}) = 0$

Para $\lambda \neq 0$, temos

$$\frac{d^T H f(\bar{x}) d}{2} = \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda^2} - \frac{\sigma_2(\lambda)}{\lambda^2}$$

Por hipótese, $f(\bar{x} + \lambda d) - f(\bar{x}) \geqslant 0$. Tomando limites ficamos com

$$(\forall d \in \Re^n) \quad \frac{d^T H f(\bar{x}) d}{2} \geqslant 0$$

O que é a definição de matriz semi definida positiva. ■

Nosso objetivo é encontrar $\bar{x}$ tal que $\nabla f(\bar{x}) = 0$, mas isso não garante que $\bar{x}$ seja minimizador local. $\bar{x}$ pode ser maximizador local ou ponto de inflexão. Por isso, precisamos estudar as condições suficientes de otimalidade.

## 2.2 Condições Suficientes de Otimalidade

**Teorema 3.** *Suponha que para $\bar{x} \in \Re^n$, $\nabla f(\bar{x}) = 0$ e $Hf(\bar{x})$ é definida positiva. Então $\bar{x}$ é minimizador local de (2.1).*

Demonstração: Seja $S = \{d \in \Re^n / \parallel d \parallel = 1\}$

Uma matriz é definida positiva se e só se $(\forall d \in S)\ d^T A d > 0$

A função $d \in S \mapsto d^T A d$ é contínua no conjunto compacto S, e portanto, tem minimizador $\bar{d} \in S$, ou seja,

$$(\forall d \in S) \quad d^T A d \geqslant \alpha = \bar{d}^T A \bar{d} > 0$$

($\alpha$ é o menor autovalor de A).

Com A $= Hf(\bar{x})$, temos: $(\forall d \in S)$,

$$f(\bar{x} + \lambda d) = f(\bar{x}) + \lambda \nabla f(\bar{x})^T d + \frac{\lambda^2}{2} d^T H f(\bar{x}) d + \sigma(\lambda^2) \geqslant \theta(\bar{x}) + \frac{\lambda^2}{2} \alpha + \sigma(\lambda^2)$$

dividindo por $\lambda^2$, para $\lambda > 0$:

$$(\forall \lambda \in S) \quad \frac{f(\bar{x} + \lambda d) - f(\bar{x})}{\lambda^2} = \frac{\alpha}{2} + \frac{\sigma(\lambda^2)}{\lambda^2}.$$

Para $\lambda \in [0, \bar{\lambda}]$ suficientemente pequeno, temos:

$$f(\bar{x} + \lambda d) - f(\bar{x}) > 0$$

Portanto $f(\bar{x}) \leqslant f(x), \forall x$, tal que $\| x - \bar{x} \| \leqslant \bar{\lambda}$

Logo, $\bar{x}$ é minimizador local. ■

## 2.3 Direção de Máximo Declive

A cada iteração deve-se encontrar uma direção $d$, em que a função "cai bastante", e minimizar esta função ao longo desta direção. Para esta minimização usamos as buscas unidirecionais (Seção Áurea e Armijo). A direção que usamos para resolver esse problema é a direção de máximo declive.

A direção de máximo declive é a direção que resolve

$$\underset{\|d\|=1}{\text{minimizar}} \nabla f(x)^T d$$

Usando a desigualdade de Cauchy-Schwarz, temos:

$$| \nabla f(x)^T d | \leqslant \| \nabla f(x) \| . \| d \|$$

sendo que, quando $\nabla f(x)$ e $d$ são colineares, ocorre a igualdade.

Logo, o mínimo será atingido por

$$d = \frac{-\nabla f(x)}{\| \nabla f(x) \|}$$

Por conveniência, trabalharemos com a direção de máximo declive $d = -\nabla f(x)$, sem normalizá-la.

Ou seja, a direção de máximo declive é a direção oposta ao gradiente da função no ponto $x$, já que o gradiente aponta para a direção de maior crescimento da função.

## 2.4 O Algoritmo de Cauchy

Definiremos ponto desejável, um ponto x tal que $\nabla f(x) = 0$.

São dados $f : \Re^n \to \Re$ de classe $C^1$ e um ponto inicial $x_0 \in \Re^n$.

Enquanto $x^k$ não for desejável (ou seja, enquanto $\nabla f(x^k) \neq 0$), calculamos a direção oposta ao gradiente, ou seja, $d = -\nabla f(x^k)$

Fazemos a busca unidirecional, encontrando, assim, $\bar{\lambda}$ (se existir).

Se não existir, o problema não tem solução.

Agora, fazemos $x^{k+1} := x^k + \bar{\lambda}d$

Em resumo, enquanto não acharmos um ponto desejável, "procuramos" a direção de máximo declive, ou seja, $d = -\nabla f(x^k)$, fazemos uma busca unidirecional ao longo dessa direção, encontrando o tamanho do passo. Repetimos o procedimento enquanto $\nabla f(\bar{x}) \neq 0$.

## 2.5 Implementação do Algoritmo de Cauchy em Matlab

Em matlab o algoritmo de Cauchy fica assim:

% Definimos a função $f$ que queremos minimizar, um ponto inicial $x_0 \in \Re^n$ e o erro máximo $\epsilon$.

WHILE

$\|\nabla f(x^k)\| < \epsilon$;

$d = -\nabla f(x^k)$;

% Agora, fazemos uma busca unidirecional ao longo de $d$, usando Seção Áurea ou Armijo, encontrando assim, $\lambda$.

$x^{k+1} = x^k + \lambda * d$;

END

## 2.6 Comparação do Método de Cauchy usando Seção Áurea com o Método de Cauchy usando Armijo

Agora, faremos uma comparação entre o método de Cauchy usando busca unidirecional de Seção Áurea e o método de Cauchy usando busca unidirecional de Armijo, através de dois exemplos. Um usa como função critério a função penalidade e o outro usa como função critério a função "banana".

**Exemplo 1.** *Comparação do método de Cauchy usando a busca unidirecional de Armijo com o método de Cauchy usando a busca unidirecional de Seção Áurea, tendo como função critério a função penalidade* $-\sum \log s$, *onde* $s = b - A'x$.

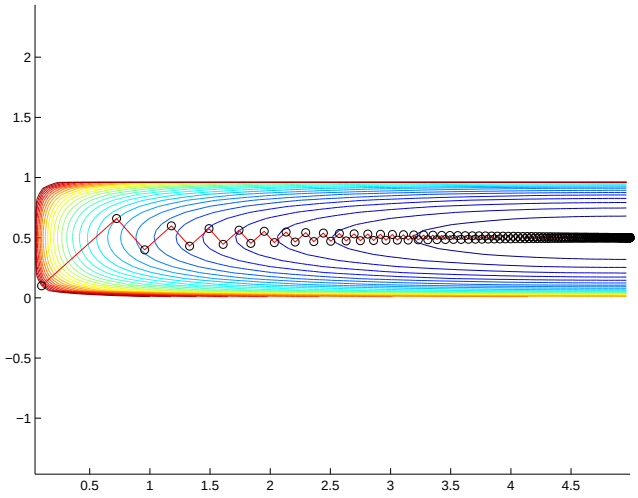


Figura 2.1: Iterações do método de Cauchy com busca unidirecional de Seção Áurea usando a função penalidade.

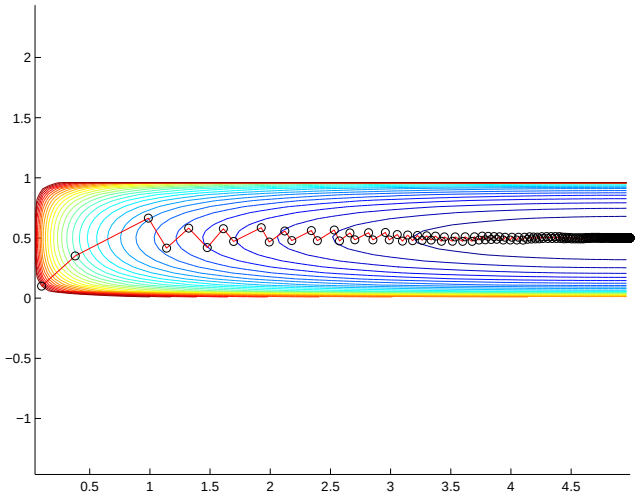


Figura 2.2: Iterações do método de Cauchy com busca de Armijo usando a função penalidade.

Para uma precisão de $\epsilon = 10^{-6}$, o método de Cauchy com busca unidirecional de Seção Áurea, faz 2336 iterações até atingir a precisão desejada, precisando, para isso, fazer 49895 cálculos de funções e 2336 cálculos de gradientes. Já o método de Cauchy com busca unidirecional de Armijo, precisa de 1857 iterações para atingir a precisão, precisando para isso fazer 38243 cálculos de funções e 1857 cálculos de gradientes.

Neste caso, usando a função penalidade, vemos que o número de iterações e de cálculos de funções e gradientes é muito grande. O método de Cauchy com Armijo se mostra mais eficaz, fazendo menos iterações e menos cálculos para alcançar a precisão.

Agora veremos, no caso da função penalidade, como se "comporta" a norma do gradiente nos dois métodos.

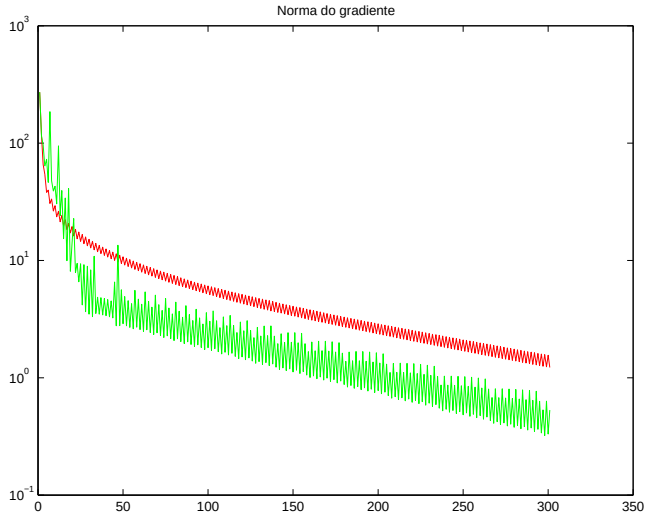


Figura 2.3: Comparação entre as normas dos gradientes do método de Cauchy usando busca unidirecional de Seção Áurea (vermelho) e usando busca unidirecional de Armijo (verde).

A Figura (2.3) mostra o comportamento da norma do gradiente nos dois casos. Em vermelho, mostra-se o "decrescimento" da norma do gradiente para o método de Cauchy usando Seção Áurea. A norma do gradiente do método de Cauchy com Armijo é mostrada em verde. Percebemos claramente que o gradiente da função no método de Cauchy com Armijo "cai" mais depressa do que com Seção Áurea.

**Exemplo 2.** *Comparação do método de Cauchy usando a busca unidirecional de Armijo com o método de Cauchy usando a busca unidirecional de Seção Áurea, tendo como função critério a função "banana"* $f(x_1, x_2) = 2(x_2 - x_1^2)^2 + (1 - x_1)^2$.

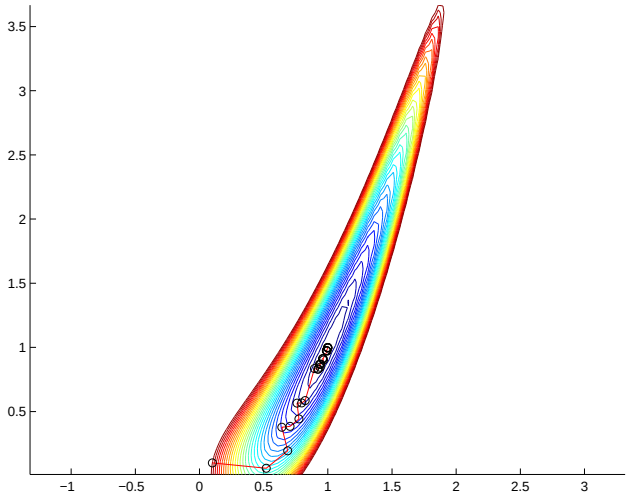


Figura 2.4: Iterações do método de Cauchy com busca unidirecional de Seção Áurea usando a função "banana".

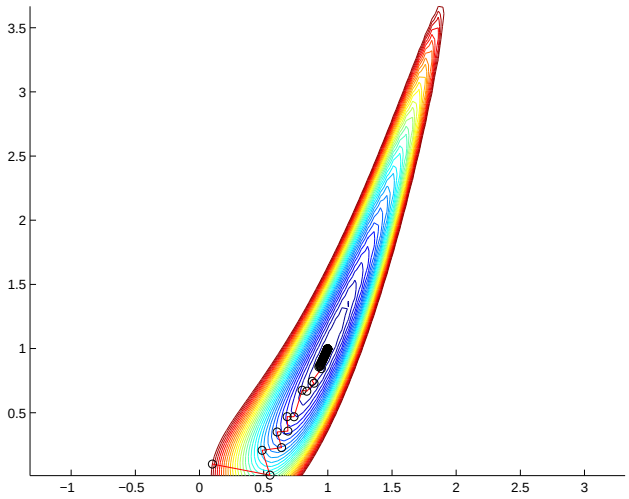


Figura 2.5: Iterações do método de Cauchy com busca unidirecional de Armijo usando a função "banana".

Para uma precisão de $\epsilon = 10^{-3}$, o método de Cauchy com busca unidirecional de Seção Áurea, faz 150 iterações até atingir a precisão desejada, precisando fazer 3904 cálculos de funções e 150 cálculos de gradientes. Já o método de Cauchy com busca

unidirecional de Armijo, precisa de 106 iterações até atingir a precisão, fazendo para isto, 867 cálculos de funções e 106 cálculos de gradientes.

Neste caso, usando a função banana, o método de Cauchy usando Armijo é mais eficaz, faz menos iterações, menos cálculos de funções e menos cálculos de gradientes.

Agora veremos, no caso da função "banana", como se "comporta" a norma do gradiente nos dois métodos.

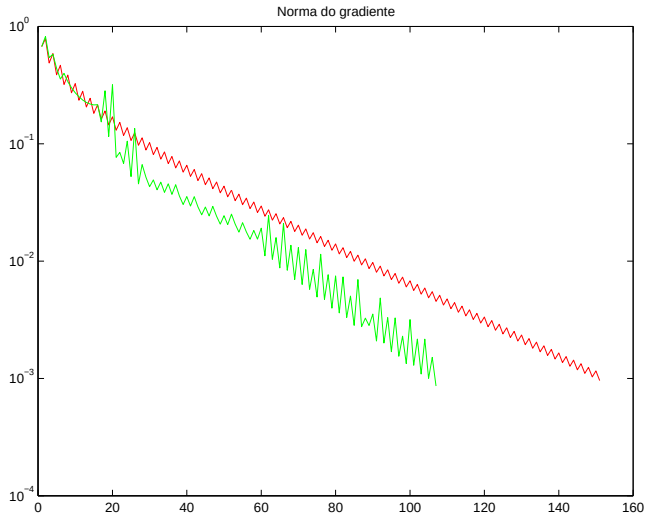


Figura 2.6: Comparação entre a norma do gradiente do método de Cauchy usando busca unidirecional de Seção Áurea (vermelho) e usando busca unidirecional de Armijo (verde).

A Figura (2.6) mostra o "decrescimento" da norma do gradiente nos dois métodos. Em vermelho mostra-se o comportamento da norma do gradiente no método de Cauchy com Seção Áurea, e em verde, o comportamento do gradiente no método de Cauchy com Armijo que, como vemos, "cai" mais depressa do que no outro método.

É preciso lembrar que essa comparação depende muito do ponto inicial que é tomado, sendo que essa comparação somente é válida para estes exemplos, podendo ser bem diferente para outras funções e outros pontos iniciais.

Portanto, não há vantagens em usar uma busca unidirecional exata, que, como vimos, é menos eficiente que a busca de Armijo, em geral faz mais iterações, além de precisar fazer mais cálculos de funções e gradientes que o método de Armijo.

**Cauchy**

Matemático e físico-matemático, nasceu no dia 21 de agosto de 1789 e faleceu no dia 23 de maio de 1857. Numerosos termos matemáticos têm o seu nome, por exemplo, o teorema integral de Cauchy, o método de cauchy, que estamos estudando, entre outros. Cauchy foi o primeiro a fazer um estudo das condições para a convergência de séries infinitas. Escreveu mais de setenta memórias, abordando quase todos os ramos da matemática.

# Capítulo 3

# O Método de Newton

O método que estudaremos neste capítulo, deriva do método de Newton para resolver sistemas de equações.

Este método tem estrutura idêntica ao método de Cauchy. O problema de minimização irrestrita continua sendo:

$$\underset{x\in\Re^n}{\text{minimizar}}\, f(x) \tag{3.1}$$

onde $f : \Re^n \to \Re$ é diferenciável de classe $C^2$.

O método usa a direção obtida pela minimização da aproximação quadrática em todo o espaço, se essa direção existir. Se não existir, o método falha.

Aproximações quadráticas não são somente melhores que aproximações lineares, mas ganham importância à medida que se aproximam de um minimizador local de $f(.)$. Também neste método, um ponto $\bar{x}$ é considerado desejável, se $\nabla f(\bar{x}) = 0$.

## 3.1 Minimização de uma Quadrática em $\Re^n$

Dada uma função $f : \Re^n \to \Re$, de classe $C^2$ e uma direção $d \in \Re^n$, perto de um ponto $x^k$ podemos aproximar $f$ pela expansão do polinômio de Taylor:

$$f_q(x^k) = f(x^k) + \nabla f(x^k)'d + \frac{1}{2}d'H(x^k)d.$$

Tomando a derivada dessa expressão e igualando a zero, temos:

$$\nabla f(x^k) + H(x^k)d = 0.$$

Se $H(x^k)$ for não singular, $H^{-1}(x^k)$ existe e então,

$$d = -H^{-1}(x^k)\nabla f(x^k)$$

Se $H(x^k)$ é definida positiva, então esta direção aponta de $x^k$ na direção de $\bar{x}$ que minimiza a aproximação quadrática de $f$.

Geralmente $f$ não é quadrática. Porém aproximações quadráticas são razoáveis sobretudo perto de soluções do problema (3.1). Quando $f$ for quadrática, o método convergirá em uma só iteração.

Quando a hessiana $H(x^k)$ é definida positiva, o método de Newton costuma ser excelente. Quando $H(x^k)$ não é definida positiva, a direção pode não ser de descida, e a busca unidirecional ao longo de $d$ resulta em $\lambda = 0$: o método falha. Se $x^k$ estiver próximo de um minimizador local $\bar{x}$ de (3.1), o método é muito eficiente convergindo rapidamente.

Sua implementação é dificultada pelo fato de nem sempre a hessiana ser definida positiva e pelo fato de ser necessário calcular a matriz hessiana, o que geralmente é muito "caro" computacionalmente.

## 3.2 O Algoritmo de Newton

Definimos ponto desejável, um ponto $x$ tal que $\nabla f(x) = 0$.

São dados $f : \Re^n \to \Re$ de classe $C^2$ e um ponto inicial $x_0 \in \Re^n$.

Enquanto $x^k$ não for desejável (ou seja, enquanto $\nabla f(x^k) \neq 0$), calculamos a direção $d$ resolvendo o sistema $H(x^k)d = -\nabla f(x^k)$.

(Este passo é bem-definido somente se o sistema tem solução.)

Fazemos a busca unidirecional, encontrando, assim, $\bar{\lambda}$ (se existir).

Se $\bar{\lambda} = 0$, o método falha.

Agora, fazemos $x^{k+1} := x^k + \bar{\lambda}d$

Em resumo, enquanto não acharmos um ponto desejável, calculamos a direção de Newton, fazemos uma busca unidirecional ao longo dessa direção, encontrando o tamanho do passo. Repetimos o procedimento até que $\nabla f(\bar{x}) = 0$, ou até que o método falhe.

## 3.3 Implementação do Algoritmo de Newton em Matlab

Em matlab o algoritmo de Newton fica assim:

% Definimos a função $f$ que queremos minimizar, um ponto inicial $x_0 \in \Re^n$ e o erro máximo $\epsilon$.

WHILE

$\|\nabla f(x^k)\| < \epsilon$;

$d = -H(x^k)\backslash\nabla f(x^k)$;

% Agora, fazemos uma busca unidirecional ao longo de $d$, usando Seção Áurea ou Armijo, encontrando assim, $\lambda$.

$x^{k+1} = x^k + \lambda * d$;

END

## 3.4 Comparação do Método de Newton usando Seção Áurea com o Método de Newton usando Armijo

Agora, faremos uma comparação entre o método de Newton usando busca unidirecional de Seção Áurea e o método de Newton usando busca unidirecional de Armijo, através de dois exemplos. O primeiro exemplo usa como função critério a função penalidade e o outro usa como função critério a função "banana".

**Exemplo 3.** *Comparação do método de Newton usando a busca unidirecional de*

*Armijo com o método de Newton usando a busca unidirecional de Seção Áurea, tendo como função critério a função penalidade* $-\sum \log s$, *onde* $s = b - A'x$.

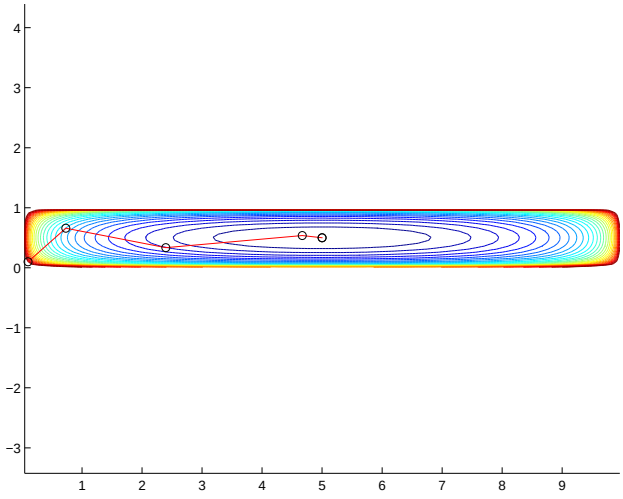


Figura 3.1: Iterações do método de Newton com busca unidirecional de Seção Áurea usando a função penalidade.

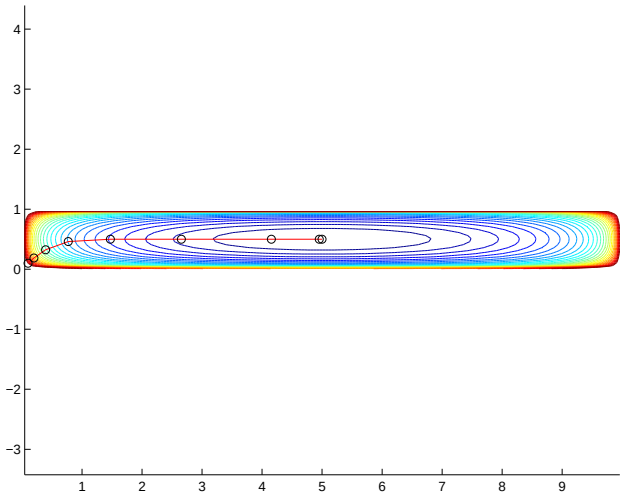


Figura 3.2: Iterações do método de Newton com busca unidirecional de Armijo usando a função penalidade.

Para uma precisão de $\epsilon = 10^{-3}$, o método de Newton com busca unidirecional de Seção Áurea, faz 7 iterações até atingir a precisão desejada, precisando, para isso, fazer 182 cálculos de funções, 7 cálculos de gradientes e 7 cálculos de hessianas. Já o método

de Newton com busca unidirecional de Armijo, precisa de 8 iterações para alcançar a precisão, fazendo 14 cálculos de funções, 8 cálculos de gradientes e 8 cálculos de hessianas.

Neste caso, usando a função penalidade, o método de Newton com Seção Áurea faz menos iterações, mas o método de Newton com Armijo faz bem menos cálculos de funções.

Agora veremos, no caso da função penalidade, como se "comporta" a norma do gradiente nos dois métodos.

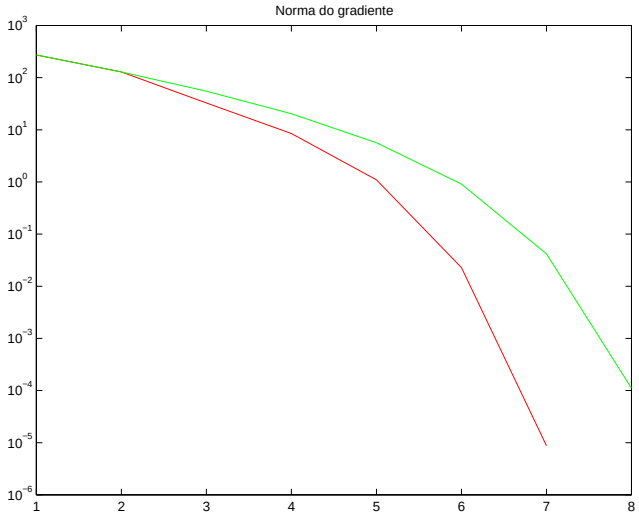


Figura 3.3: Comparação entre as normas dos gradientes do método de Newton usando busca unidirecional de Seção Áurea e usando busca unidirecional Armijo.

A Figura (3.3) mostra o comportamento da norma do gradiente nos dois casos. Em vermelho, mostra-se o "decrescimento" da norma do gradiente para o método de Newton usando Seção Áurea. A norma do gradiente do método de Newton com Armijo é mostrada em verde. Percebemos que o gradiente no método de Newton com Seção Áurea "cai" mais depressa que no outro método.

**Exemplo 4.** *Comparação do método de Newton usando a busca unidirecional de Armijo com o método de Newton usando a busca unidirecional de Seção Áurea, tendo como função critério a função "banana"* $f(x_1, x_2) = 2(x_2 - x_1^2)^2 + (1 - x_1)^2$.

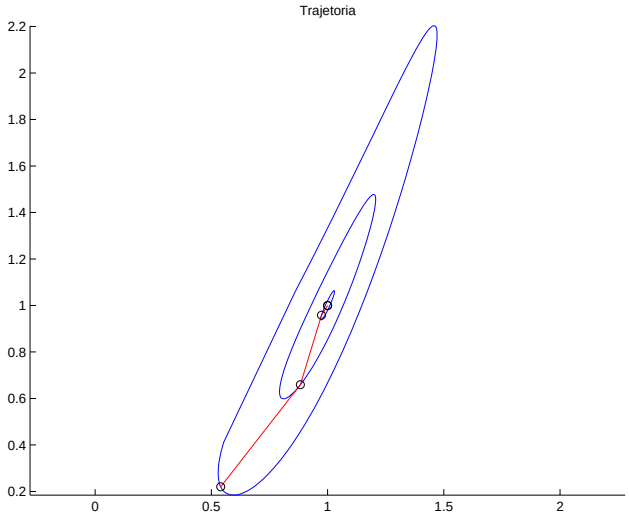


Figura 3.4: Iterações do método de Newton com busca unidirecional de Seção Áurea usando a função "banana".

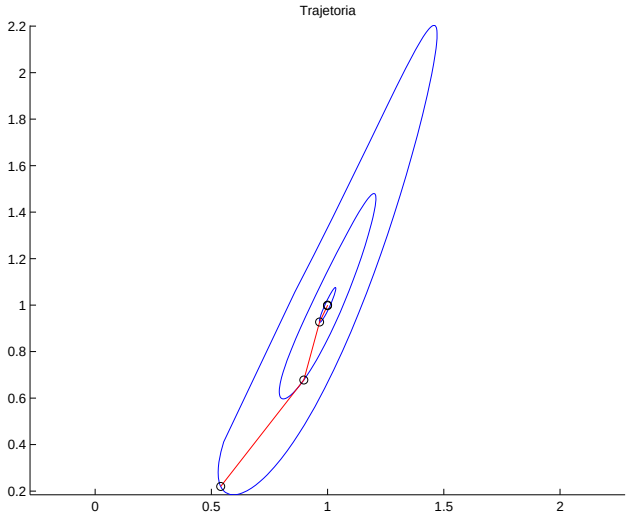


Figura 3.5: Iterações do método de Newton com busca unidirecional de Armijo usando a função "banana".

Para uma precisão de $\epsilon = 10^{-3}$, o método de Newton com busca unidirecional de Seção Áurea, faz 5 iterações até atingir a precisão desejada, precisando fazer 120 cálculos de funções, 5 cálculos de gradientes e 5 cálculos de hessianas. Já o método de Newton com busca unidirecional de Armijo, precisa também de 5 iterações até atingir a precisão, fazendo para isto, 8 cálculos de funções, 5 cálculos de gradientes e 5 cálculos

de hessianas.

Neste caso, usando a função "banana", os dois métodos fazem o mesmo número de iterações, mas o método de Newton com Armijo precisa fazer menos cálculos de funções.

Agora veremos, no caso da função "banana", como se "comporta" a norma do gradiente nos dois métodos.

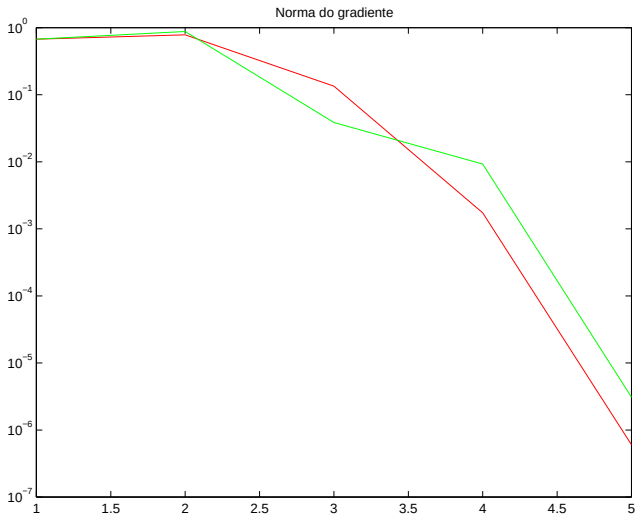


Figura 3.6: Comparação entre a norma dos gradientes do método de Newton usando busca unidirecional de Seção Áurea e usando busca unidirecional Armijo.

A Figura (3.6) mostra o "decrescimento" da norma do gradiente nos dois métodos. Em vermelho mostra-se o comportamento da norma do gradiente no método de Newton com Seção Áurea, e em verde, o comportamento do gradiente no método de Newton com Armijo. Neste caso, os comportamentos são semelhantes.

Lembramos novamente, que essa comparação depende muito do ponto inicial que é tomado, sendo que essa comparação somente é válida para estes exemplos. Para outras funções e outros pontos iniciais, a comparação muda.

Como vimos, através dos dois exemplos, o método de Newton com Armijo é mais eficaz do que o método de Newton com Seção Áurea. Vejamos porque:

Na reta, se $\theta$ é quadrática, e sendo $\rho$ um minimizador de $\theta$, sabemos que:

$$\theta(\rho) = \theta(0) + 0.5\theta'(0),$$

Em $\Re^n$, se $d$ é a direção de Newton a partir de $x^k$, para uma função quadrática $f$ com hessiana positiva, teremos:

$$f(x^k + d) = f(x^k) + 0.5f'(x^k, d)$$

Se $f$ não é quadrática mas $Hf(x^k)$ é definida positiva, e se o modelo quadrático é uma boa aproximação para $f$, então:

$$f(x^k + d) \cong f(x^k) + 0.5f'(x^k, d)$$

e para $\alpha < 0.5$ (digamos $\alpha = 0.3$)

$$f(x^k + d) < f(x^k) + \alpha f'(x^k, d)$$

Com isso cocluímos que Newton com Armijo é mais eficaz do que Newton com Seção Áurea.

Como podemos perceber, o método de Newton é muito mais eficaz que o método de Cauchy, faz bem menos iterações e muito menos cálculos de funções e gradientes. O incoveniente deste método é que, como já falamos, a direção $d$ nem sempre é uma direção de descida, por isso, nem sempre ele funciona, mas quando a direção é de descida, ele funciona muito bem. E também é necessário calcular a hessiana e a sua inversa, cálculos que não são necessários no método de Cauchy.

**Newton**

Matemático, físico e astrônomo, nasceu em 25 de dezembro de 1642 na cidade de Woolsthope, Lincolnshire, Inglaterra. Newton fez uma verdadeira revolução na matemática, óptica, física e astronomia. Lançou a base do cálculo integral e diferencial. Chegou a importantes conclusões sobre a lei da gravitação, a decomposição da luz solar no espectro, etc.

# Capítulo 4

# Busca Bidirecional

Nos capítulos 2 e 3 vimos, respectivamente, o método de Cauchy e Newton. O método de Cauchy geralmente é eficiente longe de uma solução ótima do problema, mas tem baixa velocidade de convergência, ou seja, faz muitas iterações para atingir o ponto de mínimo. Já o método de Newton é eficiente perto de uma solução ótima, e, em geral, tem velocidade de convergência muito boa, mas pode ser ineficiente longe da solução.

Para aproveitar as vantagens dos dois métodos, vamos descrever um algoritmo que, a cada iteração, minimiza (ou reduz) a função critério ao longo do subespaço gerado pelas direções de Cauchy e de Newton. Considere o problema:

$$\underset{x\in\Re^n}{\text{minimizar}}\, f(x) \tag{4.1}$$

onde $f(.)$ é de classe $C^2$ com hessiana definida positiva.

O algoritmo de busca bidirecional é o seguinte:

É dado $x^0 \in \Re^n$.

Enquanto $\nabla f(x^k) \neq 0$, calculamos a direção de Cauchy $d_c = -\nabla f(x^k)$ e a direção de Newton $d_n = -H(x^k)^{-1}\nabla f(x^k)$.

Define-se o subespaço S gerado por $d_c$ e $d_n$.

Agora resolvemos

$$x^{k+1} = \underset{d\in S}{\text{argmin}}\, f(x^k + d) \tag{4.2}$$

A seguir, vamos estudar um algoritmo de busca bidirecional, que será utilizado para resolver (4.2).

## 4.1 Minimização em $R^2$

Nesta seção vamos descrever um método para a minimização de uma função $f : \Re^2 \to \Re$. Na seção seguinte, esse método será aplicado ao problema acima. O método consiste em, a cada iteração, fazer duas iterações do método de Cauchy a partir de um ponto $x^0 \in \Re^2$ dado, e passar uma reta pelo ponto inicial e pelo ponto que minimiza a função ao longo da direção de cauchy na segunda iteração. Este método é chamado de PARTAN (método de tangentes paralelas).

### 4.1.1 O Algoritmo de Partan

Novamente, definiremos ponto desejável, um ponto $x$ tal que $\nabla f(x) = 0$.

São dados $f : \Re^2 \to \Re$ e um ponto inicial $x^1 \in \Re^2$.

Enquanto $x^k$ não for desejável (ou seja, enquanto $\nabla f(x^k) \neq 0$), calculamos a direção $d^1 = -\nabla f(x^k)$ (direção de Cauchy).

Fazemos uma busca unidirecional ao longo de $d^1$ a partir de $x^k$, encontrando assim, $\lambda^1$.

Calculamos o ponto $y = x^1 + \lambda^1 d^1$.

Agora, calculamos novamente a direção de Cauchy a partir do ponto $y$, ou seja, $d^2 = -\nabla f(y)$.

Fazemos uma busca unidirecional ao londo de $d^2$ a partir de $y$, encontrando assim, $\lambda^2$.

Calculamos o ponto $z = y + \lambda^2 d^2$.

Agora, calculamos a direção definida pelo ponto inicial $x^k$ e pelo ponto $z$ resultado da busca uidirecional ao longo de $d^2$, ou seja, $d = z - x^k$.

Fazemos uma busca ao longo de $d$ a partir de $x^k$ (ou equivalentemente a partir de $z$), encontrando $\lambda$.

E define-se o novo ponto $x^{k+1} = x^k + \lambda d$.

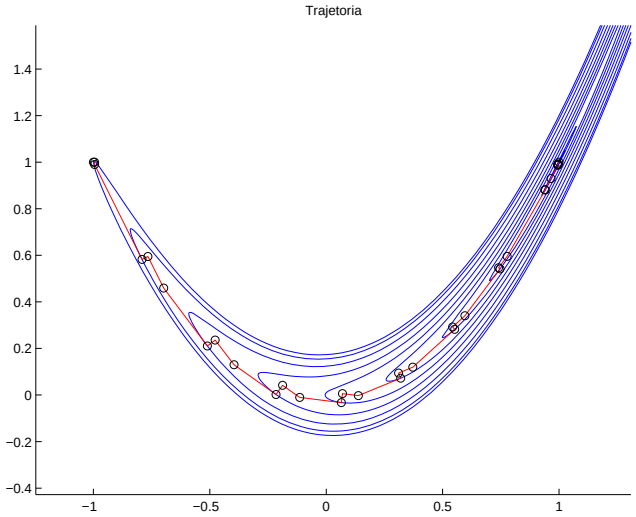


Figura 4.1: Iterações do método de Partan.

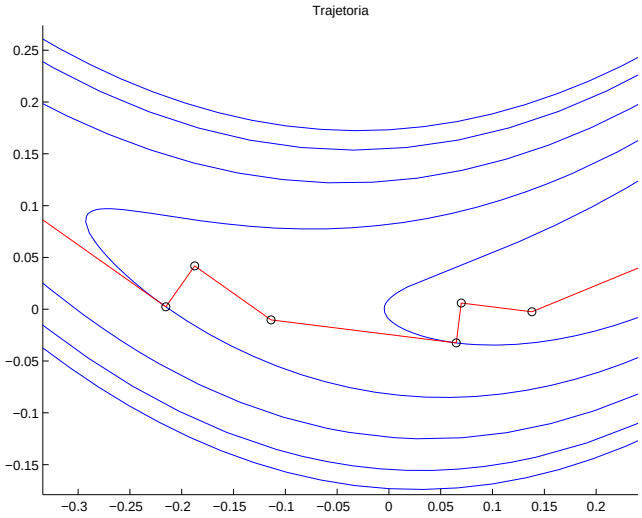


Figura 4.2: Iterações do método de Partan.

As Figuras (4.1) e (4.2), mostram algumas iterações do método de Partan.

A seguir, mostraremos que este método é um método de direções conjugadas.

### 4.1.2 Algoritmos de Direções Conjugadas

**Definição:** dada uma matriz simétrica $n \times n$, $H$. As direções $d^1$ e $d^2 \in \Re^n$ são H-conjugadas se e somente se ${d^1}^T H d^2 = 0$.

Se a matriz $H$ é definida positiva, então é possível encontrar $n$ direções $d^1, d^2, ..., d^n \neq 0$, mutuamente H-conjugadas. Em nosso caso, a seguinte propriedade de direções H-conjugadas é importante:

Considere o problema de minimizar uma função quadrática

$$x \in \Re^n \mapsto f(x) = c^T x + \frac{1}{2} x^T H x$$

com H definida positiva e simétrica.

Suponha que são conhecidas as direções $d^1, d^2, ..., d^n \neq 0$, H-conjugadas. Dado um ponto $x^0 \in \Re^n$ arbitrário, vamos estudar o seguinte procedimento:

$$\begin{gathered} k = 0 \\ \text{PARA } i = 0, 1, ..., n-1 \\ \lambda = \underset{\lambda \in \Re}{\operatorname{argmin}} f(x^k + \lambda d^k) \\ x^{k+1} = x^k + \lambda d^k \end{gathered} \tag{4.3}$$

O método faz $n$ minimizações em sequência, ao longo das $n$ direções H-conjugadas. Os minimizadores estão bem definidos, uma vez que $H$ é definida positiva. O resultado é que $x^n$ é o minimizador global de $f(.)$. Este resultado está demonstrado em [7].

Portanto, algoritmos de direções conjugadas são muito eficientes para a minimização de funções quadráticas. A dificuldade está em encontrar as direções H-conjugadas.

Mostraremos que o método Partan, acima descrito, gera direções H-conjugadas. Quando aplicado a uma função quadrática em $\Re^2$, resolve o problema em uma iteração. Quando aplicado a funções não quadráticas, tem convergência muito rápida.

**Teorema 4.** *Seja dada uma função quadrática em $R^n : f(x) = c^T x + \frac{1}{2} x^T H x$, com $H$ definida positiva. Sejam dados uma direção $d$, duas retas paralelas, $\{x^1 + \lambda d, \lambda \in \Re\}$ e $\{x^2 + \lambda d, \lambda \in \Re\}$, e os minimizadores de $f(.)$ ao longo dessas duas direções, digamos, $y^1$ e $y^2$. Então a direção $h = y^1 - y^2$ é $H$ - conjugada a $d$.*

**Demonstração:** Como $y^1$ é minimizador de $f(.)$ ao longo de $d$, temos $f'(y^1, d) = \nabla f(y^1)^T d = 0$. O mesmo vale para $y^2 : f'(y^2, d) = \nabla f(y^2)^T d = 0$. Para $y \in \Re^n$, temos

$\nabla f(y) = c + Hy$, $f'(y,d) = d^T \nabla f(y)$.

Portanto,

$$d^T c + d^T H y^1 = 0 \tag{4.4}$$

$$d^T c + d^T H y^2 = 0 \tag{4.5}$$

Subtraindo (4.4) de (4.5), temos $d^T H(y^1 - y^2) = 0$, que é a definição de direções H - conjugadas. ■

Vamos agora considerar o método Partan em $\Re^2$, utilizando minimização de uma função quadrática:

No primeiro passo, dado $x^1$, fazemos $d^1 = -\nabla f(x^1)$, $\lambda^1 = \text{argmin} f(x^1 + \lambda d^1)$(Cauchy a partir de $x^1$) e $y = x^1 + \lambda^1 d^1$.

No segundo passo, fazemos $d^2 = -\nabla f(y)$, $\lambda^2 = \text{argmin} f(y + \lambda^1 d^2)$ (Cauchy a partir de $y$) e $z = y + \lambda^2 d^2$

**Teorema 5.** *Com as condições acima, temos $d = z - x^1$ é H - conjugada a $d^2$.*

**Demonstração:** Temos, $\nabla f(y)^T d^1 = 0$, pois $y$ é minimizador de $f$ ao longo de $d^1$. Portanto $d^1$ é perpendicular a $\nabla f(y)$, ou seja, $d^1$ é perpendicular a $d^2$, e temos $d^1 = -\nabla f(x^1)$.

Consequentemente, $d^{2^T} \nabla f(x^1) = 0$ e $f'(x^1, d^2) = 0$, e portanto $x^1$ é minimizador de $f$ ao longo de $d^2$.

Temos então, que as retas $r^1 = \{x^1 + \lambda d^2, \lambda \in \Re\}$ e $r^2 = \{y + \lambda d^2, \lambda \in \Re\}$ são paralelas.

$x^1$ é minimizador de $f$ em $r^1$ e $z$ é minimizador de $f$ em $r^2$.

Pelo teorema 4, as direções $d^2$ e $d = (z - x^1)$ são H-conjugadas. ■

### 4.1.3 Implementação do Algoritmo de Partan em Matlab

Em Matlab, o algoritmo de Partan fica assim:

%Definimos a função $f : \Re^2 \to \Re$ que queremos minimizar, um ponto $x^1 \in \Re^2$ e uma precisão $\epsilon$.

k = 1

WHILE $\|\nabla f(x^k)\| > \epsilon$;

$d^1 = -\nabla f(x^1)$;

%Agora fazemos uma busca unidirecional ao longo de $d^1$ a partir de $x^1$, usando Seção Áurea, encontrando assim , $\lambda^1$.

$y = x^1 + \lambda^1 * d^1$;

$d^2 = -\nabla f(y)$;

%Agora fazemos uma busca unidirecional ao longo de $d^2$ a partir de $y$, usando Seção Áurea, encontrando assim, $\lambda^2$.

$z = y + \lambda^2 * d^2$;

%Agora calculamos a direção definida pelo ponto inicial e pelo resultado da minimização unidirecional ao longo de $d^2$.

$d = z - x^k$;

% Novamente fazemos uma busca unidirecional ao longo de $d$ a partir de $x^1$ (ou equivalentemente a partir de $z$), encontrando $\lambda$.

$x^{k+1} = x^k + \lambda * d$;

k=k+1;

END

## 4.2 Minimização Bidirecional

Vamos agora, aplicar o método de Partan à resolução do problema de busca bidirecional (4.2).

Considere uma ponto $x^k \in \Re^n$ e as direções $d_c$ e $d_n$ geradas pelo algoritmo de busca bidirecional, descrito anteriormente. Se $d_c$ e $d_n$ são colineares, então o problema se reduz a uma busca unidirecional. Suponhamos então, que $d_c$ e $d_n$ são linearmente independentes.

Definimos $V = [d_c\ d_n]$ e $S = \mathcal{I}(V)$, onde $\mathcal{I}(V)$ é o conjunto imagem de V.

O problema é:

$$\underset{d \in S}{\text{minimizar}}\ f(x^k + d) \tag{4.6}$$

Para utilizar o método Partan, temos que calcular em cada passo o gradiente da função restrita à variedade $x^k + S$. O gradiente dessa função em um ponto $x$ é a projeção de $\nabla f(x)$ sobre o núcleo de $V^T$. Vamos chamar essa projeção de $g_p$. Então temos:

$$\nabla f(x) = g_p + \tilde{g}_p$$

onde $g_p \in \mathcal{N}(V^T)$ e $\tilde{g}_p \in \mathcal{I}(V)$, isto é, $V^T g_p = 0$ e $\tilde{g}_p = Vy$ com $y \in \Re^2$. Daí, temos:

$$\nabla f(x) = g_p + Vy$$

Multiplicando por $V^T$, ambos os lados da expressão acima, obtem-se:

$$V^T V y = V^T \nabla f(x)$$

Como $d_c$ e $d_n$ são linearmente independentes, é fácil ver que $V^T V$ é não singular, e então

$$y = (V^T V)^{-1} V^T \nabla f(x)$$

e portanto,

$$g_p = \nabla f(x) - Vy$$

Substituindo y, temos,

$$g_p = \nabla f(x) - V(V^T V)^{-1} V^T \nabla f(x)$$

ou seja,

$$g_p = (I - V(V^T V)^{-1} V^T) \nabla f(x)$$

A matriz $P = (I - V(V^T V)^{-1} V^T)$ é a matriz de projeção sobre o núcleo de $V^T$.

Com essa expressão, podemos adaptar o algoritmo PARTAN para a resolução do problema (4.2).

### 4.2.1 O Algoritmo de Busca Bidirecional

Agora, vamos descrever o algoritmo completo de busca bidirecional:

Sejam dados $x^k \in \Re^n$, e uma precisão $\epsilon > 0$.

Enquanto $\|\nabla f(x^k)\| > \epsilon$,

Calculamos a direção de Cauchy $d_c = -\nabla f(x^k)$ e a direção de Newton $d_n = -H(x^k)^{-1}\nabla f(x^k)$.

Agora, definimos $V = [d_c\ d_n]$ e a matriz de projeção $P = (I - V(V^TV)^{-1}V^T)$.

Inicia-se, agora, a busca bidirecional:

Fazemos $w^1 = x^k$ e $j = 1$.

Enquanto $\|P\nabla f(w^j)\| > \epsilon$,

Calculamos $d^1 = -P\nabla f(w^j)$. Fazemos uma busca unidirecional ao longo de $d^1$ a partir de $w^j$, usando seção áurea, encontrando $\lambda$.

O próximo ponto será $y = w^j + \lambda d^1$.

Agora calculamos uma nova direção $d^2 = -P\nabla f(y)$. Fazemos nova busca unidirecional ao longo de $d^2$ a partir de $y$, usando seção áurea, encontrando $\lambda$.

Calculamos o novo ponto $z = y + \lambda d^2$.

Agora, definimos a nova direção $d = z - w^j$. Faz-se uma nova busca unidirecional ao longo de $d$ a partir de $w^j$, encontrando $\lambda$. Finalmente, fazemos $w^{j+1} = w^j + \lambda d$ e $j = j + 1$.

E assim termina a busca bidirecional, e fazemos $x^{k+1} = w^j$.

Salientamos que, quando $j = 1$, $d^1 = -P\nabla f(w^j) = -\nabla f(x^k)$, e portanto, a projeção é desnecessária.

Em resumo, calculamos as direções de Cauchy e de Newton, definimos o subspaço gerado por essas duas direções, calculamos a matriz projeção sobre o nucleo de $V^T$ e fazemos a minimização nesse subespaço, usando o método de Partan. O resultado dessa minimização será o ponto $x^{k+1}$.

Não há interesse por uma busca bidirecional muito precisa, pela mesma razão de que não havia necessidade de muita precisão em buscas unidirecionais.

### 4.2.2 Implementação do Algoritmo de Busca Bidirecional em Matlab

Em matlab o algoritmo de Busca Bidirecional fica assim:

%Definimos a função que queremos minimizar, um ponto $x^k \in \Re^n$ e uma precisão $\epsilon$, o algoritmo gera um ponto $x^{k+1}$.

$k = 1;$

$WHILE\ \|\nabla f(x^k)\| > \epsilon,$

$d_c = -\nabla f(x^k);$

$d_n = -H(x^k)^{-1}\nabla f(x^k);$

$V = [d_c\ d_n];$

$P = (I - V(V^TV)^{-1}V^T);$

$w^1 = x^k;$

%Agora, usamos o método de Partan a partir de $w^1$ para obter $\bar{w}$, solução de

$$\underset{w\ \in\ x^k+S}{\text{minimizar}}\ f(w) \tag{4.7}$$

$x^{k+1} = \bar{w};$

$k = k + 1;$

END

%Agora descreveremos o método de Partan para resolver o problema interno.

$j = 1;$

WHILE $\|P\nabla f(w^j)\| > \epsilon;$

$d^1 = -P\nabla f(w^j);$

%Agora fazemos uma busca unidirecional, usando seção áurea ao longo de $d^1$ a partir de $w^j$, encontrando assim, $\lambda$.

$y = w^j + \lambda d^1;$

$d^2 = -P\nabla f(y);$

% Fazemos nova busca unidirecional ao longo de $d^2$, a partir de $y$, usando seção áurea, encontrando assim, $\lambda$.

$z = y + \lambda d^2$;

$d = z - w^j$;

%Novamente fazemos uma busca unidirecional usando seção áurea, ao longo de $d$, a partir de $w^j$, ou de $z$, encontrando $\lambda$.

$w^{j+1} = w^j + \lambda d$;

$j = j + 1$;

END

# Conclusão

O presente trabalho descreve Algoritmos de Programação Não Linear, de forma simples e útil. O problema da Programação Não Linear é encontrar uma solução para o problema de minimização de uma função não linear. Este trabalho me possibilitou não só conhecer mais de Programação Não Linear, como também aprender Matlab e LaTeX.

A sequência em que se desenvolve o trabalho é descrever como cada método funciona e mostrar como é sua implementação em Matlab.

Podemos perceber que, em geral, o método de Armijo é mais eficiente que o método de Seção Áurea para a busca unidirecional e que o método de Newton, quando a direção é de descida, é muito mais eficiente do que o método de Cauchy, fazendo, em geral, menos iterações e menos cálculos de funções.

Finalizando, este trabalho foi uma oportunidade de colocar em prática a teoria aprendida durante o curso de matemática. Neste sentido, acredito que este trabalho de conclusão de curso, atingiu os objetivos propostos no programa de Licenciatura em Matemática da Universidade Federal de Santa Catarina.

# Referências Bibliográficas


[1] GONZAGA, Clóvis C.; *Algoritmos de Pontos Interiores Para Programação Linear*. 17º Colóquio Brasileiro de Matemática, IMPA, RJ.

[2] FRIEDLANDER, Ana; *Elementos de Programação Não Linear*.Editora da UNICAMP, SP, 1994.

[3] HANSELMAN, Duane; LITTLEFIELD, Bruce., *MATLAB 5 - Guia do Usuário*. Makron Books, 1999.

[4] FLETCHER, R.; LEYFFER, S., *Nonlinear programming without a penalty function*. Mathematical Programming, 91 (2002), pp. 239 - 269.

[5] GONZAGA, Clóvis C.; KARAS, E; VANTI, M., *A Globally convergent filter method for nonlinear programming*. SIAM Journal on Optimization, 2003. to appear.

[6] FRITZSCHE, Helmut, *Programação Não-Linear: Análise e Métodos*. Editora da Universidade de São Paulo, São Paulo, 1978.

[7] J. NOCEDAL AND S. J. WRIGHT, Numerical Optimization, Springer Series in Operations Research, Springer-Verlag, 1999.