Universidade Federal de Minas Gerais
Escola de Engenharia
Departamento de Engenharia Elétrica

Notas de Aula de Otimização

Jaime A. Ramírez
Felipe Campelo
Frederico G. Guimarães
Lucas S. Batista
Ricardo H. C. Takahashi

Sumário

Lista de Figuras

Capítulo 3

Métodos Numéricos para Otimização Irrestrita

3.1 Introdução

No capítulo anterior, vimos a caracterização da função objetivo, funções de restrição, e as condições de otimalidade – condições de Karush-Kuhn-Tucker – que servem de base para encontrar a solução de problemas de otimização utilizando técnicas numéricas.

O objetivo deste capítulo é o estudo de métodos numéricos para otimização irrestrita, em particular Métodos de Direções de Busca. Esses métodos são a base de vários pacotes comerciais de otimização e foram desenvolvidos a partir da ideia básica de fazer o algoritmo evoluir encontrando novos pontos situados em direções para as quais a função objetivo decresça, em relação ao ponto corrente.

A versão mais primitiva dessa família de métodos é o "Algoritmo do Gradiente": dado um ponto inicial no espaço de busca, obtém-se um novo ponto situado sobre a reta definida por esse ponto e pelo gradiente da função objetivo. Essa é a direção para a qual, localmente, a função mais rapidamente decresce, no sentido contrário ao do vetor gradiente. Determina-se o novo ponto como aquele em que a função objetivo atinge o mínimo sobre essa reta. A partir desse novo ponto, repete-se o processo, até que seja satisfeito um critério de convergência.

Ao longo das décadas de 50 e 60 do século XX, tal método básico foi aperfeiçoado para permitir que a direção de busca, na qual era feita a busca unidimensional, sofresse uma "correção", dando origem a uma família de métodos chamados "Métodos Quase-Newton". Tal correção leva em conta mais informações a respeito da função objetivo; além do valor de seu gradiente no ponto corrente, procura-se também levar em consideração a curvatura da função. Aproximações de segunda ordem, por exemplo, levando em consideração estimativas da Hessiana da função objetivo, permitem significativa aceleração de convergência dos métodos.

Os métodos agrupados neste capítulo, sob a denominação de "Métodos de Direção de Busca", têm essa raiz, e possuem em comum as seguintes características:

1. Cada novo ponto é obtido a partir de um processo de otimização unidimensional, que tem como ponto de partida o ponto anterior.

2. A direção na qual é feita a busca unidimensional é uma função das avaliações anteriores da função objetivo.

O capítulo é dividido em sete seções. Inicialmente, é definida a estrutura básica dos métodos de direção de busca. Posteriomente, é apresentado o problema exemplo que será utilizado para ilustrar as características dos métodos que serão estudados no capítulo. Então, é descrito o método de busca em direções aleatória. Este é seguido pelo método do gradiente, que inclui a descrição do método da seção áurea para minimização de funções unidimensionais. Em seguida, é apresentado o método de aproximações quadráticas, i.e., o método de Newton e a família de métodos quase-Newton, incluindo os métodos de Davidon-Fletcher-Powell (DFP) e Broyden-Fletcher-Goldfarb-Shanno (BFGS). Finalmente, a última seção dedica-se a métodos sem derivada: método de Hooke-Jeeves e método de Nelder-Mead. Ao final do capítulo é apresentada uma lista de exercícios. Leitura complementar sobre os métodos discutidos neste capítulo pode ser encontrada nas referências [1]– [2].

3.2 Estrutura Básica

Seja o problema de otimização mono-objetivo irrestrito:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3.1}$$

sendo que $\boldsymbol{x} \in \mathbb{R}^n$ e $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^1$. Dado um ponto inicial $\boldsymbol{x}_0 \neq \boldsymbol{x}^*$, obtém-se uma sequência $\boldsymbol{x}_k$ tal que $\boldsymbol{x}_k \to \boldsymbol{x}^*$ a partir do algoritmo de otimização. A família dos algoritmos de direção de busca possui a estrutura:

Algorithm 1: Algoritmo de Direção de Busca

```
k ← 0;
while (critério de parada não for satisfeito) do
    d_k ← h(x_1, ..., x_k, f(x_1), ..., f(x_k));
    α_k ← arg min_α f(x_k + α d_k);
    x_{k+1} ← x_k + α_k d_k;
    k ← k + 1;
end
```

Nessa estrutura, $\boldsymbol{h}(\cdot, \ldots, \cdot)$ é uma função que em geral será recursiva, isto é, não irá depender explicitamente dos pontos anteriores, mas irá armazenar sua influência em variáveis intermediárias. Um algoritmo irá diferir de outro essencialmente pela maneira como é calculada a direção de busca $\boldsymbol{d}_k$, ou na escolha dessa função. No caso do Método do Gradiente, tem-se simplesmente que:

$$\boldsymbol{d}_k = -\nabla f(\boldsymbol{x}_k) \tag{3.2}$$

No caso do Método de Newton, tem-se que:

$$H_k = \text{Hessiana}(f(\boldsymbol{x}_k)) \tag{3.3}$$

e

$$\boldsymbol{d}_k = -H_k^{-1} \nabla f(\boldsymbol{x}_k) \tag{3.4}$$

Tanto o gradiente quanto a Hessiana são determinados por meio de diversas avaliações da função $f(\cdot)$, tendo em vista a regra básica de que esta é o único tipo de informação disponível. A justificativa para a utilização dessas direções de busca será estudada neste capítulo. Os métodos chamados de *quase-Newton* substituem a avaliação da Hessiana da função objetivo pela construção de uma estimativa para essa Hessiana.

Os elementos para a construção de algoritmos de direções de busca são, portanto:

(i) um método de cálculo de direções de busca, possivelmente envolvendo o cálculo de estimativas para o gradiente e para a Hessiana da função objetivo;

(ii) um método de minimização de funções de uma única variável;

(iii) um critério de decisão que permita afirmar que o algoritmo convergiu para uma solução satisfatória, podendo ser terminada sua execução.

Esses elementos serão examinados a seguir. Antes porém, é apresentado o problema exemplo que será utilizado para ilustrar as características dos métodos discutidos neste capítulo. A natureza do processo de convergência, intrínseco aos métodos de direção de busca, é estudada através do exame de convergência de um algoritmo de interesse apenas conceitual: o algoritmo do método de busca em direções aleatórias.

3.3 Problema Exemplo

Considere o problema:

$$\begin{aligned} \boldsymbol{x}^* = \ & \arg \min_{\boldsymbol{x}} \ f(\boldsymbol{x}) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3 \\ & \text{Sujeito a: } \{ \ -6 \leq x_1 \leq 6; -6 \leq x_2 \leq 6 \end{aligned} \tag{3.5}$$

que representa a minimização de uma função de duas variáveis $f(x_1, x_2)$. Neste caso, a região factível é definida pelos limites inferiores e superiores das variáveis x_1 e x_2. Por ser uma função de apenas duas variáveis, $f(x_1, x_2)$ pode ser representada no plano $x_1 \times x_2$ através de curvas de nível, conforme indicado na Figura 3.1. Por inspeção, obtém-se que a solução é o ponto $x_1 = -1.5$; $x_2 = 2.5$. Este problema exemplo será utilizado nas seções seguintes para estudar as características dos métodos de direção de busca. Antes porém, analisaremos as condições de otimalidade para assegurar que a solução obtida por inspeção é de fato o ponto mínimo do problema.

As condições necessárias de primeira ordem requerem:

$$\frac{\partial f(\cdot)}{\partial x_1} = 4x_1 + 2x_2 + 1 = 0 \tag{3.6}$$

$$\frac{\partial f(\cdot)}{\partial x_2} = 2x_2 + 2x_1 - 2 = 0 \tag{3.7}$$

que resulta em um sistema de duas equações e duas incógnitas, cuja solução é $x_1^* = -1.5$ e $x_2^* = 2.5$.

As condições suficientes de segunda ordem requerem que a matriz Hessiana (3.8) seja positiva definida.

$$H = \begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} \tag{3.8}$$

A verificação se a matriz Hessiana é positiva definida será feita por meio de dois métodos: (i) cálculo dos autovalores de H, e (ii) cálculo dos determinantes de todas as submatrizes que envolvem a diagonal principal de H.

- Cálculo dos autovalores de H:

$$\begin{vmatrix} 4-\lambda_1 & 2 \\ 2 & 2-\lambda_2 \end{vmatrix} = (4-\lambda_1)(2-\lambda_2) - 4 = 0$$

  Os autovalores são $\lambda_1 = 5.24$ e $\lambda_2 = 0.76$; a matriz é positiva definida.

- Cálculo dos determinantes de todas as submatrizes que envolvem a diagonal principal de H:

  $|4| > 0$

  $$\begin{bmatrix} 4 & 2 \\ 2 & 2 \end{bmatrix} = 4 > 0$$

  A matriz é positiva definida.

Com isso, conclui-se que as condições de segunda ordem são satisfeitas e que $x_1^* = -1.5$ e $x_2^* = 2.5$ é de fato o ponto de mínimo da função.

3.4 Método de Busca em Direções Aleatórias

Considere-se o Algoritmo 2 apresentado a seguir.

A função $rand(n, 1)$ é definida tal que sua saída é um vetor de n componentes aleatórios independentes e identicamente distribuídos, segundo uma distribuição Gaussiana, com média 0 e variância 1. A convergência desse algoritmo para o ponto de mínimo de uma função unimodal é estabelecida no teorema a seguir.

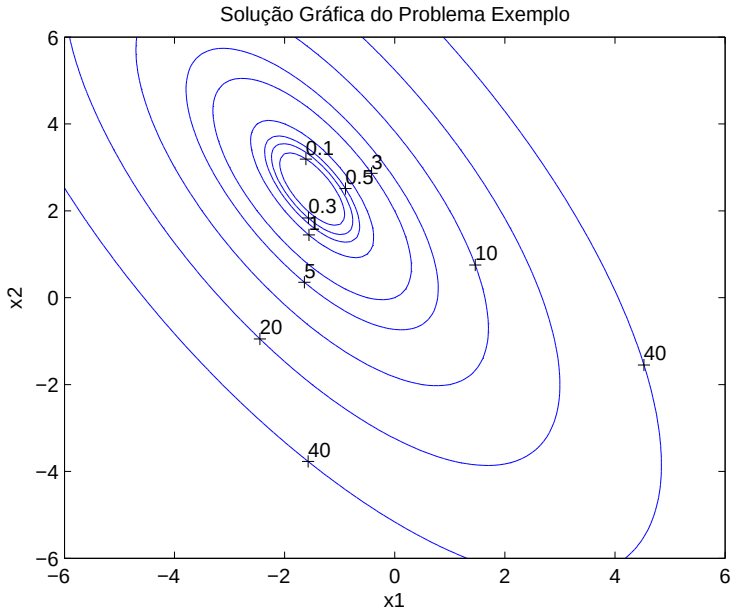

Figura 3.1: Solução Gráfica do Problema Exemplo

Algorithm 2: Algoritmo de Busca em Direções Aleatórias

$k \leftarrow 0$;
while *(critério de parada não for satisfeito)* **do**
  $\boldsymbol{d}_k \leftarrow rand(n, 1))$;
  $\alpha_k \leftarrow \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$;
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;
  $k \leftarrow k + 1$;
end

Teorema 3.1 *Seja $f(\boldsymbol{x}) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função estritamente unimodal, e seja $\boldsymbol{x}_0$ um ponto qualquer em seu domínio. A aplicação do algoritmo de busca em direções aleatórias a essa função, partindo desse ponto, produz uma sequência $[f(\boldsymbol{x}_k)]$ que se aproxima de forma monotônica do valor mínimo da função, $f(\boldsymbol{x}^*)$.* □

DEMONSTRAÇÃO: A subrotina de minimização unidimensional embutida no algoritmo implica que, qualquer que seja a direção $\boldsymbol{d}_k$ escolhida:

$$f(\boldsymbol{x}_k) \leq f(\boldsymbol{x}_{k-1})$$

o que demonstra a monotocidade da sequência. A unimodalidade estrita de $f(\boldsymbol{x})$ implica que para todo ponto $\boldsymbol{x}_k \neq \boldsymbol{x}^*$ haverá possíveis direções $\boldsymbol{d}_k$ para as quais ocorra:

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k)$$

para algum valor de α_k. Se uma dessas direções não for escolhida, ocorrerá:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$$

Do contrário:

$$\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$$

e

$$f(\boldsymbol{x}_{k+1}) < f(\boldsymbol{x}_k)$$

Pela construção da função aleatória geradora do vetor $\boldsymbol{d}_k$, há uma probabilidade não-nula de geração de direções em que ocorre a diminuição do valor da função, de forma que a aproximação fica demonstrada, ou seja:

$$\forall\, \boldsymbol{x}_k \neq \boldsymbol{x}^*,\ \exists\, N:\ f(\boldsymbol{x}_{k+N}) < f(\boldsymbol{x}_k)$$

■

Note-se que o Teorema mostra que ocorre a *aproximação*, mas não a *convergência* para o ponto de mínimo $\boldsymbol{x}^*$. De qualquer forma, este é um algoritmo que efetivamente funcionaria para a minimização de funções. A questão a ser observada é que uma escolha adequada da direção de busca $\boldsymbol{d}_k$, em substituição à aleatória, pode aumentar em muito a eficiência do algoritmo de minimização. Os diversos algoritmos de direções de busca surgem precisamente quando se propõem diferentes formas de se fazer tal escolha de uma direção.

3.4.1 Exemplo

A utilização do algoritmo de busca em direções aleatórias no problema exemplo definido pela equação (3.5) resulta na trajetória representada na Fig. 3.2 a seguir. Nesse exemplo, usou-se o ponto inicial $\boldsymbol{x}_0 = (4, -4)^T$ e o número máximo de iterações fixado em 20. O gráfico ilustrando a variação do valor da função objetivo $f(\boldsymbol{x})$ versus o número de iterações é mostrado na Fig. 3.3.

3.5 Método do Gradiente

A primeira escolha razoável para uma direção de busca eficaz, $\boldsymbol{d}_k$, é a da direção contrária à do gradiente da função no ponto corrente $\boldsymbol{x}_k$. Essa escolha se justifica com a observação de que, localmente, essa é a direção na qual a função $f(\cdot)$ decresce mais rapidamente. Isso define o *Algoritmo do Método do Gradiente*, esquematizado no Algoritmo 3.

Esse algoritmo baseia-se apenas na informação local a respeito da variação da função $f(\cdot)$ em todas as direções do espaço, sintetizada no gradiente da função $f(\cdot)$. A única suposição implícita na aplicação desse algoritmo é a de que a função $f(\boldsymbol{x})$ seja diferenciável.

Os elementos construtivos desse algoritmo são examinados a seguir.

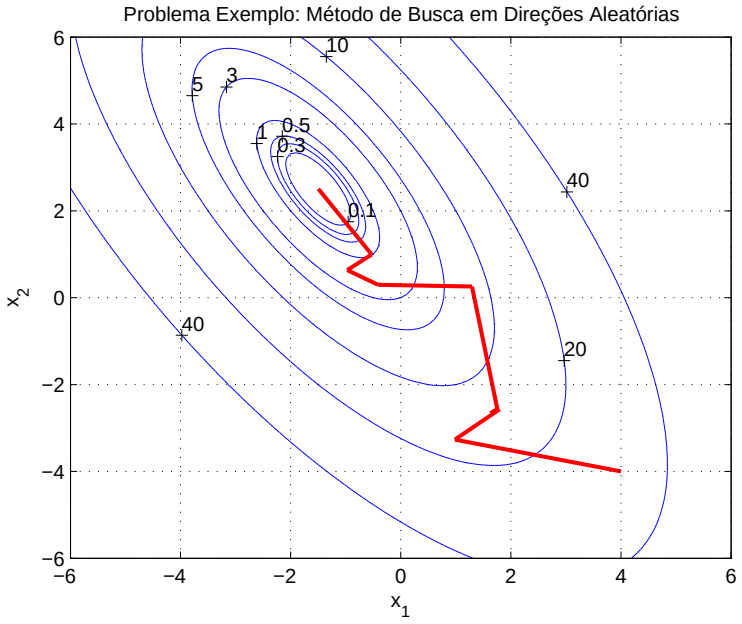

Figura 3.2: Problema exemplo – solução usando o Método de Busca em Direções Aleatórias.

Algorithm 3: Algoritmo do Método do Gradiente

$k \leftarrow 0$;
while *(critério de parada não for satisfeito)* **do**
  $\boldsymbol{g}_k \leftarrow \text{gradiente}(f(\cdot), \boldsymbol{x}_k)$;
  $\boldsymbol{d}_k \leftarrow -\boldsymbol{g}_k$;
  $\alpha_k \leftarrow \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$;
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;
  $k \leftarrow k + 1$;
end

3.5.1 Cálculo do Gradiente

No contexto da teoria de otimização, a suposição mais geral a respeito da informação sobre o sistema sendo otimizado é: dispõe-se *apenas* de um modelo que, recebendo como entrada o vetor de variáveis de otimização, fornece o valor da função-objetivo para tal vetor. Portanto, não se dispõe, em geral, de funções que explicitamente forneçam o gradiente da função objetivo para certa especificação do vetor de variáveis de otimização, o que torna necessária a construção de um algoritmo para calcular o gradiente de $f(\boldsymbol{x})$.

O algoritmo mais simples que se pode imaginar para o cálculo numérico aproximado do gradiente de uma função é decorrência imediata da definição de gradiente, sendo substituída a fórmula diferencial por diferenças finitas. Seja $\boldsymbol{x} \in \mathbb{R}^n$ o vetor de variáveis de otimização, e seja $\boldsymbol{e}_i$ o vetor definido por:

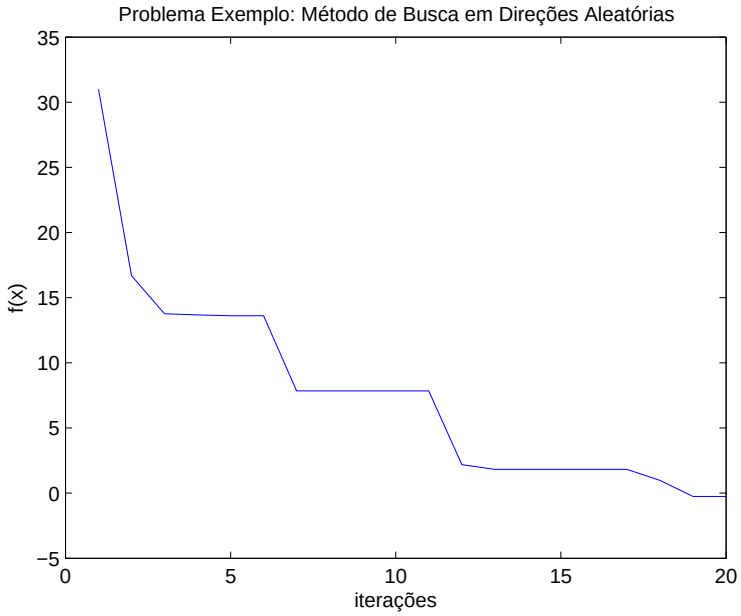

Figura 3.3: Problema exemplo – variação da função objetivo versus o número de iterações para o Método de Busca em Direções Aleatórias.

$$\boldsymbol{e}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \rightarrow \text{i-esima posição} \tag{3.9}$$

Considere-se um certo $\delta > 0$, tal que $\delta \approx 0$. O algoritmo de cálculo do vetor gradiente $\boldsymbol{g}$ no ponto $\boldsymbol{x}$ pode ser definido como:

Algorithm 4: Algoritmo do Cálculo do Gradiente

```
k ← 0;
for (i ← 1 until n) do
    g_i ← [f(x + δe_i) − f(x)] /δ;
end
g ← [g_1, ..., g_n]^T;
```

NOTA 3.1 *Deve-se observar que o Algoritmo de Cálculo do Gradiente é exato para funções lineares, ou seja, para funções cuja série de Taylor termina no termo de primeira ordem. Nesse caso, δ pode assumir qualquer valor: o cálculo será exato mesmo para δ grande.*

EXEMPLO 3.1 *Seja a função de duas variáveis $f(\boldsymbol{x})$, definida por*

$$f(\boldsymbol{x}) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3$$

Analiticamente, o gradiente dessa função é dado por:

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 4x_1 + 2x_2 + 1 \\ 2x_1 + 2x_2 - 2 \end{bmatrix}$$

Por essa fórmula analítica, sabe-se que no ponto $\boldsymbol{x}_0 = [0\ 0]^T$ *o gradiente é igual a:*

$$\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Utilizando-se o algoritmo de diferenças finitas, para $\delta = 0.0001$*, obtém-se a estimativa de gradiente igual a:*

$$\widehat{\nabla} f(\boldsymbol{x}_0) = \begin{bmatrix} 1.0002 \\ -1.9999 \end{bmatrix}$$

Deve-se notar que o Algoritmo do Gradiente não utiliza nenhuma informação analítica a respeito da função. A única informação utilizada é proveniente de avaliações da função em diferentes pontos.

3.5.2 Otimização Unidimensional

A seguinte linha do algoritmo do gradiente é agora examinada:

$$\alpha_k \leftarrow \arg \min_{\alpha} f(\boldsymbol{x}_k + \alpha d_k)$$

O cálculo de α_k é feito fixando-se o ponto atual $\boldsymbol{x}_k$ e uma direção de busca, $\boldsymbol{d}_k$. Isso faz com que a função objetivo, $f(\boldsymbol{x})$, que originalmente seria de n variáveis, ou seja, dependeria de um vetor $\boldsymbol{x}$ de dimensão n, torne-se agora uma função de uma única variável, α.

A otimização de funções de uma única variável, em uma única dimensão, portanto, é tarefa substancialmente mais simples que a otimização em diversas dimensões. Podem-se construir algoritmos diversos para resolver esse problema, baseados em premissas diversas a respeito da função a ser otimizada. Uma premissa comum, que necessariamente possui validade local em um ponto de ótimo estrito, é a de que a função-objetivo possua um único mínimo local no domínio em questão.

A estratégia que será adotada compõe-se de duas etapas:

1. Cercar o valor ótimo α_k^*, o qual determina o ponto de mínimo da busca unidimensional, $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k^* \boldsymbol{d}_k$. Para tal constroe-se um intervalo $[a, b]$ em que $\alpha_k^* \in [a, b]$;

2. Reduzir o intervalo $[a, b]$ até a precisão desejada, ou, até que $|\alpha - \alpha^*| < \xi$.

Determinação do Intervalo $[a, b]$

Determinar o intervalo $[a, b]$ resume-se a cercar α^* a partir de $\boldsymbol{x}_k$ e dar passos na direção $\boldsymbol{d}_k$ enquanto a função $\theta(\alpha) = f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$ decrescer. Assim que esta função volta a crescer, significa que passou-se por uma depressão ou por um ponto de mínimo α^* naquela direção (direção $\boldsymbol{d}_k$). Portanto, basta fechar o intervalo $[a, b]$ que contém esta depressão.

Esta ideia está organizada no Algoritmo 5. Admite-se que $\theta(\cdot)$ é unimodal, a e b são o início e fim, respectivamente, do intervalo $[a, b]$, s é o comprimento do passo inicial, e NFC representa o contador do número de avaliações da função objetivo.

Algorithm 5: Algoritmo para Determinação do Intervalo

```
a ← 0;
b ← s;
calcule θ(a) = θ(0) = f(x_k);
calcule θ(b);
NCF1 ← 2;
while θ(b) < θ(a) do
    a ← b;
    θ(a) ← θ(b);
    b ← 2b;
    calcule θ(b);
    NFC1 ← NFC1 + 1;
end
if NFC1 ≤ 3 then
    a ← 0;
else
    a ← a/2;
end
return a, b;
```

As seguintes observações são acrescentadas:

(i) Ao dar passos enquanto a função decresce está se fazendo $a = b$ e $b = 2b$. Essa escolha é arbitrária. Pode-se optar por $b = b+s$, ou $b = 3b$, etc. A consequência direta desta escolha será um menor ou maior número de iterações e intervalo final.

(ii) ...

Redução do Intervalo $[a, b]$

Há vários métodos, diretos e indiretos, que podem ser utilizados para reduzir o intervalo $[a, b]$; veja por exemplo [1] e [3]. Vamos concentrar a nossa atenção no método da seção áurea.

Teorema 3.2 *Seja uma função $\theta(\cdot) : \mathbb{R} \mapsto \mathbb{R}$. Seja um domínio $[a, b] \subset \mathbb{R}$, no qual $\theta(\cdot)$ possui um único mínimo local x^*. Sejam ainda dois pontos x_a e x_b tais que*

$$a < x_a < x_b < b \tag{3.10}$$

Se ocorrer

$$\theta(x_a) < \theta(x_b) \tag{3.11}$$

então a solução minimizante x^* não se encontra no intervalo $[x_b, b]$, e se ocorrer

$$\theta(x_a) > \theta(x_b) \tag{3.12}$$

então a solução minimizante x^* não se encontra no intervalo $[a, x_a]$.

□

DEMONSTRAÇÃO: Tome-se o interva-lo $[a, x_b]$. Nesse intervalo, há algum ponto x_0 para o qual $f(x_0) \leq f(x)\ \forall\ x \in [a, x_b]$ e $x_0 \neq x_b$, pela hipótese (3.11). Logo, x_0 é um mínimo local no segmento $[a, x_b]$. Como $x_0 \neq x_b$, tem-se que no intervalo $[a, b]$ x_0 permanece sendo mínimo local. Acrescentando-se agora a hipótese de que só há um mínimo local em $[a, b]$, obtém-se que $x^* = x_0$, que é o resultado pretendido. Para o outro lado do segmento, o argumento é análogo. ■

Com esse teorema, é possível construir um algoritmo que se fundamenta na lógica de excluir, a cada passo, um trecho do segmento considerado, de forma a fazê-lo contrair-se. Quando o segmento estiver suficientemente "pequeno", pode-se considerar que ocorreu a convergência para o ponto de mínimo da otimização unidimensional. A precisão dessa convergência, ou seja, o erro máximo cometido, será igual à metade do comprimento remanescente.

Existe, claramente, uma maneira de escolher os pontos x_a e x_b dentro do segmento, de forma a maximizar, em média, o comprimento do intervalo a ser excluído a cada passo, minimizando assim o número de iterações necessário para se atingir determinada precisão. Uma escolha frequentemente adotada é definida pela "seção áurea", em que escolhem-se x_a e x_b de forma que:

$$x_a = b - 0.618(b - a) \tag{3.13}$$

$$x_b = a + 0.618(b - a) \tag{3.14}$$

O fator 0.618 corresponde à "razão áurea", utilizada pelos antigos gregos para definir a razão dos lados adjacentes de um retângulo que seria "perfeita"sob o ponto de vista estético.

Com esta escolha, o método de minimização de uma função real no intervalo $[a, b]$ para se atingir uma precisão $\epsilon/2$ pode ser definido conforme apresentado no **Algoritmo** 6.

Claramente, a cada passo do algoritmo o comprimento do intervalo $[a, b]$ é multiplicado por um fator menor ou igual a 0.618, de forma que pode-se calcular o número esperado máximo de passos para se atingir a precisão desejada:

$$l_k \leq 0.618^{k-1} l_1 \tag{3.15}$$

Algorithm 6: Algoritmo da Seção Áurea

```
x_a ← b − 0.618(b − a);
x_b ← a + 0.618(b − a);
θ_a ← θ(x_a);
θ_b ← θ(x_b);
while (b − a > ϵ) do
    if (θ_a > θ_b) then
        a ← x_a;
        x_a ← x_b;
        x_b ← a + 0.618(b − a);
        θ_a ← θ_b;
        θ_b ← θ(x_b);
    else
        b ← x_b;
        x_b ← x_a;
        x_a ← b − 0.618(b − a);
        θ_b ← θ_a;
        θ_a ← θ(x_a);
    end
end
α ← (a + b)/2;
```

sendo l_k o comprimento do intervalo $[a, b]$ no passo k.

É importante salientar que é possível construir outros algoritmos, inclusive mais eficientes que o Método da Seção Áurea, para a otimização de funções de uma única variável. Para maiores detalhes, podem ser consultadas as referências [1] e [3].

EXEMPLO 3.2 *Tome-se a mesma função do Problema Exemplo.*

$$f(\boldsymbol{x}) = 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3$$

O gradiente da função é dado por:

$$\nabla f(\boldsymbol{x}) = \begin{bmatrix} 4x_1 + 2x_2 + 1 \\ 2x_1 + 2x_2 - 2 \end{bmatrix}$$

No ponto $\boldsymbol{x}_0 = [-1\ 1]^T$ *o gradiente é igual a:*

$$\nabla f(\boldsymbol{x}_0) = \begin{bmatrix} -1 \\ -2 \end{bmatrix}$$

A função $\theta(\cdot)$*, tomada a partir do ponto* $\boldsymbol{x}_0$ *na direção de* $-\nabla f(\boldsymbol{x}_0)$*, pode ser encontrada analiticamente por substituição da variável vetorial* $\boldsymbol{x}$ *pela variável escalar* α*. Baseando-se neste conceito, tem-se que:*

$$\boldsymbol{x} = \boldsymbol{x}_0 - \alpha \nabla f(\boldsymbol{x}_0)$$

ou:

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \end{bmatrix} - \alpha \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} \alpha - 1 \\ 2\alpha + 1 \end{bmatrix}$$

Realizando as substituições necessárias, obtém-se a função unidimensional:

$$\theta(\alpha) = 2(\alpha-1)^2+(2\alpha+1)^2+2(\alpha-1)(2\alpha+1)+(\alpha-1)-2(2\alpha+1)+3 = 10\alpha^2-5\alpha+1$$

Essa função possui mínimo para:

$$\frac{d\theta(\alpha)}{d\alpha} = 20\alpha - 5 = 0$$

ou seja, para $\alpha = 1/4$. *Para esse valor de* α, *o novo ponto* $\boldsymbol{x}$ *obtido no espaço vetorial é:*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ +1 \end{bmatrix} - \frac{1}{4} \begin{bmatrix} -1 \\ -2 \end{bmatrix} = \begin{bmatrix} -3/4 \\ +3/2 \end{bmatrix} = \begin{bmatrix} -0.75 \\ +1.5 \end{bmatrix}$$

Utilizando o algoritmo da seção áurea, obtém-se uma estimativa do ponto que minimiza $\theta(\cdot)$ *na direção considerada. Este ponto é igual a:*

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -0.7501 \\ +1.4998 \end{bmatrix}$$

para uma precisão estabelecida de $\epsilon = 0.001$. *Deve-se notar que o ponto determinado não é o ponto de mínimo global da função* $f(\boldsymbol{x})$, *nem corresponde a um mínimo local dessa função, pois o gradiente não se anula nesse ponto. Este vetor apenas minimiza a função* $f(\cdot)$ *sobre a reta definida pelo ponto* $\boldsymbol{x}_0$ *e pela direção de busca* $-\nabla f(\boldsymbol{x}_0)$.

3.5.3 Critérios de Parada

Após produzir uma sequência de estimativas da função objetivo, avaliada para uma sequência de pontos do espaço de variáveis de otimização, o algoritmo de otimização eventualmente deverá se aproximar de um ponto de ótimo local da função. Como a aproximação para o ótimo ocorre de forma assintótica, é necessário em algum momento tomar a decisão de interromper o algoritmo, sendo a aproximação obtida considerada o valor ótimo alcançado.

Alguns critérios possíveis, para a tomada dessa decisão, seriam:

Estabilização do Valor da Função-Objetivo

Caso o valor da função-objetivo, ao longo de um certo número de iterações, não varie mais que certo percentual da diferença entre seu valor máximo ocorrido em todo o processo de otimização e seu valor mínimo verificado também em todo o processo, é possível interromper o algoritmo supondo que dificilmente viriam a ocorrer melhorias significativas da função objetivo com essa continuidade.

A seguir é apresentado um trecho de algoritmo que exemplifica a construção desse critério, o qual considera como estabilizado um algoritmo que varia, nas ultimas cinco iterações, menos de 0.0001 da "amplitude" Δ_f da função objetivo, sendo f_{max} e f_{min}, respectivamente, o máximo e o mínimo valor ocorrido para a função objetivo durante toda a execução.

Algorithm 7: Critério de Parada: Função Objetivo

$\Delta_f \leftarrow f_{max} - f_{min}$;
$f_{5+} \leftarrow \max\ \{f(\boldsymbol{x}_k), f(\boldsymbol{x}_{k-1}), f(\boldsymbol{x}_{k-2}), f(\boldsymbol{x}_{k-3}), f(\boldsymbol{x}_{k-4}), f(\boldsymbol{x}_{k-5})\}$;
$f_{5-} \leftarrow \min\ \{f(\boldsymbol{x}_k), f(\boldsymbol{x}_{k-1}), f(\boldsymbol{x}_{k-2}), f(\boldsymbol{x}_{k-3}), f(\boldsymbol{x}_{k-4}), f(\boldsymbol{x}_{k-5})\}$;
$\delta_f \leftarrow f_{5+} - f_{5-}$;
if $(\delta_f < 0.0001\Delta_f)$ **then**
  $parada \leftarrow true$;
else
  $parada \leftarrow false$;
end

NOTA 3.2 *O leitor deve estar atento para o fato de que é necessário calcular o valor Δ_f, não sendo recomendável utilizar, em seu lugar, nem f_{min} nem f_{max}. Fica para o leitor o exercício de explicar que problemas poderiam ocorrer caso fossem feitas tais escolhas.*

NOTA 3.3 *Seria entretanto possível utilizar, para Δ_f, alguma definição um pouco mais sofisticada, que por exemplo excluísse alguns dos máximos valores ocorridos para a função objetivo antes do cálculo de f_{max}. Tal procedimento aumenta a complexidade do algoritmo, mas pode torná-lo mais estável.*

Estabilização do Vetor de Variáveis de Otimização

Outra alternativa para o problema de formulação de critérios de parada de algoritmos de otimização seria a constatação de que o vetor de variáveis se estabilizou em algum ponto do espaço.

A seguir é apresentado um trecho de algoritmo que exemplifica a construção desse critério, o qual considera como estabilizado um algoritmo cujo vetor de variáveis varia, nas últimas cinco iterações, menos de 0.0001 da "faixa de variação" verificada do vetor de variáveis ao longo de toda a execução. Sejam $\boldsymbol{x}_{max}$ e $\boldsymbol{x}_{min}$ os vetores cujas componentes são o máximo valor ocorrido para cada componente do vetor de variáveis durante toda a execução e o vetor cujas componentes são o mínimo valor ocorrido para cada componente do vetor de variáveis durante toda a execução do algoritmo, respectivamente. Aqui as operações com vetores são entendidas como operações realizadas sobre cada uma das componentes dos operandos. A comparação entre dois vetores será verdadeira se cada uma das comparações de componentes for verdadeira.

NOTA 3.4 *Novamente, observa-se que não é recomendável utilizar, para construir esse critério de parada, nem $\boldsymbol{x}_{max}$, nem $\boldsymbol{x}_{min}$, nem $\boldsymbol{x}_{k-1}$ (embora essa última alternativa seja frequentemente usada na literatura) em substituição a $\Delta_{\boldsymbol{x}}$. Fica para o leitor a tarefa de explicar que problemas ocorreriam nesses casos.*

Anulação do Vetor Gradiente

Por fim, é possível ainda determinar o final de um processo de otimização com uma informação a respeito do vetor gradiente da função objetivo. Sabendo-se *a priori*

Algorithm 8: Critério de Parada: Vetor de Variáveis

$\Delta_{\boldsymbol{x}} \leftarrow \boldsymbol{x}_{max} - \boldsymbol{x}_{min}$;
$\boldsymbol{x}_{5+} \leftarrow \max \{\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \boldsymbol{x}_{k-3}, \boldsymbol{x}_{k-4}, \boldsymbol{x}_{k-5}\}$;
$\boldsymbol{x}_{5-} \leftarrow \min \{\boldsymbol{x}_k, \boldsymbol{x}_{k-1}, \boldsymbol{x}_{k-2}, \boldsymbol{x}_{k-3}, \boldsymbol{x}_{k-4}, \boldsymbol{x}_{k-5}\}$;
$\delta_{\boldsymbol{x}} \leftarrow \boldsymbol{x}_{5+} - \boldsymbol{x}_{5-}$;
if $(\delta_{\boldsymbol{x}} < 0.0001\Delta_{\boldsymbol{x}})$ **then**
  $parada \leftarrow true$;
else
  $parada \leftarrow false$;
end

que a função-objetivo é diferenciável, seu gradiente será nulo em seus pontos de mínimos locais (condição necessária de primeira ordem). Pode-se, portanto, detectar a ocorrência desses mínimos pela monitoração do valor da norma do vetor gradiente.

No trecho do algoritmo a seguir, é mostrada uma implementação desse teste sobre o vetor $\boldsymbol{g}(\boldsymbol{x})$, que é o gradiente da função objetivo $f(\cdot)$ no ponto $\boldsymbol{x}$. A base da comparação adotada é o máximo valor da norma do gradiente ocorrido ao longo de toda a execução, denotado por M_{max}.

Algorithm 9: Critério de Parada: Vetor Gradiente

$M_g = max\{\|\boldsymbol{g}(\boldsymbol{x}_k)\|, \|\boldsymbol{g}(\boldsymbol{x}_{k-1})\|, \|\boldsymbol{g}(\boldsymbol{x}_{k-2})\|\}$;
if $(M_g < 0.0001 M_{max})$ **then**
  $parada \leftarrow true$;
else
  $parada \leftarrow false$;
end

3.5.4 Convergência

Pode-se mostrar, usando o *teorema da convergência global*, que o Algoritmo do Método do Gradiente converge para a solução dos problemas de otimização mediante as condições formuladas na proposição a seguir.

Proposição 3.1 *Seja o problema de otimização irrestrito definido por:*

$$\boldsymbol{x}^* = \arg \min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3.16}$$

sendo $\boldsymbol{x} \in \mathbb{R}^n$, *com* $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ *uma função contínua. Então o Algoritmo do Gradiente irá convergir para* $\boldsymbol{x}^*$ *para todo ponto inicial* $\boldsymbol{x}_0$ *situado na bacia de atração de* $\boldsymbol{x}^*$. □

DEMONSTRAÇÃO: As condições para a validade do teorema da convergência global se complementam quando se restringe o domínio da função à bacia de atração do ponto de mínimo. Nessa situação, os vetores obtidos ao longo das iterações do Algoritmo do

Gradiente apresentam valores de $f(\cdot)$ descendentes. As demais condições não dependem do domínio. ■

Corolário 3.1 *Caso o Algoritmo do Gradiente seja iniciado em um ponto $\boldsymbol{x}_0$ não situado na bacia de atração do mínimo global $\boldsymbol{x}^*$, podem ocorrer duas situações:*

1. *O Algoritmo do Gradiente converge para o mínimo local associado à bacia de atração em que estiver localizado seu ponto inicial $\boldsymbol{x}_0$.*
2. *Caso o ponto inicial não esteja localizado em nenhuma bacia de atração, o Algoritmo do Gradiente não converge.*

□

3.5.5 Exemplo

A utilização do algoritmo do método do gradiente no problema exemplo definido pela equação (3.5) resulta na trajetória ilustrada na Fig. 3.4 a seguir. Nesse exemplo, usou-se o ponto inicial $\boldsymbol{x}_0 = (4, -4)^T$ e, como critério de parada, o número máximo de iterações fixado em 20. O gráfico ilustrando a variação do valor da função objetivo $f(\cdot)$ versus o número de iterações é mostrado na Fig. 3.3.

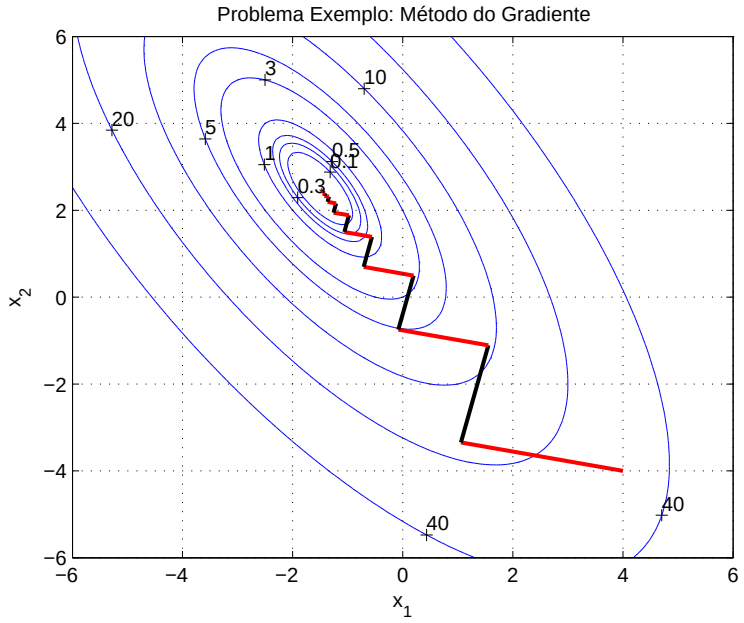

Figura 3.4: Problema exemplo – solução usando o Método do Gradiente.

3.6 Aproximações Quadráticas

Suponha-se agora que, conhecendo-se *a priori* a natureza da função objetivo, saiba-se que é razoável admitir que essa função corresponda, de maneira aproximada, a

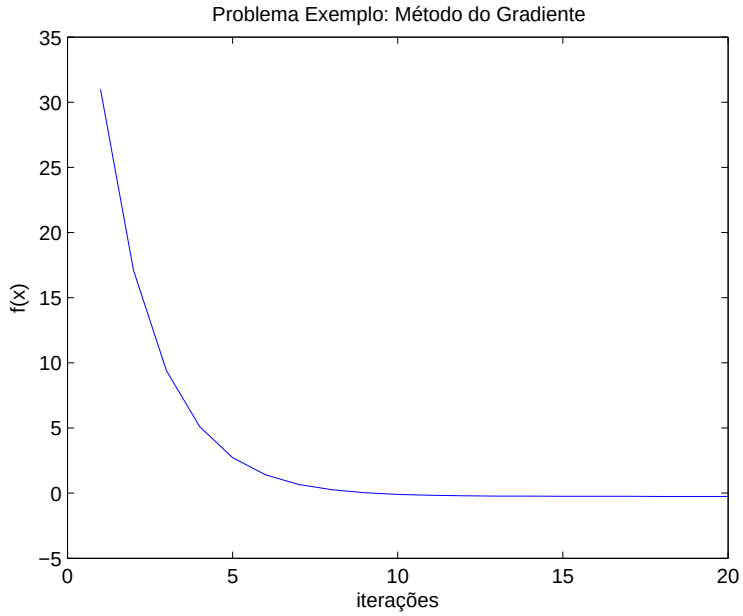

Figura 3.5: Problema exemplo – variação da função objetivo versus o número de iterações para o Método do Gradiente.

uma função quadrática, dentro de algum domínio que contenha o ponto de mínimo $\boldsymbol{x}^*$. A aproximação é feita ao redor de um ponto $\boldsymbol{x}_0$, também contido nesse domínio:

$$f(\boldsymbol{x}) \approx \boldsymbol{c}_0 + \boldsymbol{c}_1 \cdot (\boldsymbol{x} - \boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^T C_2 (\boldsymbol{x} - \boldsymbol{x}_0) \tag{3.17}$$

sendo $\boldsymbol{c}_0 \in \mathbb{R}^n$, $\boldsymbol{c}_1 \in \mathbb{R}^n$ e $C_2 \in \mathbb{R}^{n \times n}$. Essa hipótese, de fato, corresponde à suposição de que a função $f(\boldsymbol{x})$ seja de classe C^∞, pois toda função dessa classe pode ser escrita em termos de uma série de Taylor:

$$f(\boldsymbol{x}) = f(\boldsymbol{x}_0) + \nabla f(\boldsymbol{x}_0)^T (\boldsymbol{x} - \boldsymbol{x}_0) + \frac{1}{2} (\boldsymbol{x} - \boldsymbol{x}_0)^T H(\boldsymbol{x}_0) (\boldsymbol{x} - \boldsymbol{x}_0) + O(3) \tag{3.18}$$

onde o vetor $\nabla f(\boldsymbol{x}_0)$ é o gradiente da função no ponto $\boldsymbol{x}_0$, a matriz $H(\boldsymbol{x}_0)$ é a Hessiana da função em $\boldsymbol{x}_0$, e $O(3)$ é o conjunto das contribuições dos termos de ordem maior ou igual a três. O gradiente da função $f(\boldsymbol{x})$ dada por (3.18) é:

$$\nabla f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}_0) + H(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \tag{3.19}$$

Sabe-se, entretanto, a partir das condições de primeira ordem, que no ponto de mínimo local $\boldsymbol{x}^*$, o gradiente se anula, de maneira que:

$$\nabla f(\boldsymbol{x}^*) = \nabla f(\boldsymbol{x}_0) + H(\boldsymbol{x}_0)(\boldsymbol{x}^* - \boldsymbol{x}_0) = 0 \tag{3.20}$$

de onde se obtém a fórmula de determinação do ponto de mínimo:

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - H(\boldsymbol{x}_0)^{-1} \nabla f(\boldsymbol{x}_0) \tag{3.21}$$

Ou seja, se a função a ser otimizada for exatamente quadrática, basta se conhecer o gradiente e a Hessiana em um ponto qualquer $\boldsymbol{x}_0$ para se determinar, em uma

única iteração, o ponto de mínimo $\boldsymbol{x}^*$, através da equação (3.21). Se a função for aproximadamente quadrática num certo domínio, a equação (3.21) pode ainda ser empregada para produzir estimativas do ponto de mínimo que convergem muito mais rapidamente que aquelas produzidas pelo Algoritmo do Método do Gradiente.

3.6.1 Método de Newton

A partir da expressão definida pela equação (3.21), pode-se construir um algoritmo de minimização de funções que, em sua forma mais simples, envolve a aplicação iterativa de (3.21) para a busca do ótimo. O Método de Newton emprega esta abordagem (Algoritmo 10).

Algorithm 10: Algoritmo do Método de Newton

```
1 k ← 0;
2 while (critério de parada não for satisfeito) do
3 |   g_k ← gradiente(f(·), x_k);
4 |   H_k ← Hessiana(f(·), x_k);
5 |   x_{k+1} ← x_k − H_k^{-1} g_k;
6 |   k ← k + 1;
7 end
```

Convergência

No caso da otimização de funções com forma precisamente quadrática, o Algoritmo do Método de Newton não apenas converge para a solução exata do problema, como também o faz de maneira não iterativa, em um único passo. Essa não é, entretanto, a situação geral: as funções a serem otimizadas, embora frequentemente sejam duas vezes diferenciáveis, o que é necessário para a aplicabilidade desse método, na maioria das vezes não são quadráticas.

Nessa última situação, o Método de Newton, na formulação apresentada, pode até mesmo não convergir. Observando os requisitos arrolados entre as hipóteses do *teorema da convergência global*, verifica-se que o Algoritmo de Newton não satisfaz à exigência de que a iteração deva ser *descendente*, ou seja, de que o valor da função objetivo necessariamente decresça a cada iteração. De fato, nada garante que o cálculo analítico da solução que seria a exata para um problema quadrático, se aplicado a um problema que não é quadrático, não venha a levar até mesmo a um aumento no valor da função objetivo.

3.6.2 Método de Newton Modificado

Para garantir que o algoritmo produza a diminuição monotônica do valor da função objetivo, mesmo para funções não-lineares que tenham comportamento significativamente diferente da função quadrática, é empregada uma variação do Algoritmo de Newton que incorpora um aspecto crucial das características de convergência do

Algorithm 11: Algoritmo do Método de Newton Modificado

```
1 k ← 0;
2 while (critério de parada não for satisfeito) do
3 |   g_k ← gradiente(f(·), x_k);
4 |   H_k ← Hessiana(f(·), x_k);
5 |   d_k ← −H_k^{−1} g_k;
6 |   α_k ← arg min_α f(x_k + α d_k);
7 |   x_{k+1} ← x_k + α_k d_k;
8 |   k ← k + 1;
9 end
```

Algoritmo do Gradiente: a execução de uma minimização unidimensional em cada direção.

Com exceção da rotina de cálculo da Hessiana, todas as subrotinas envolvidas na construção desses algoritmos já foram apresentadas por ocasião da construção do Algoritmo do Gradiente, e são reaproveitadas aqui.

Convergência

O algoritmo modificado é exatamente equivalente ao Algoritmo de Newton original, no sentido de que ambos produzem a mesma sequência de pontos, caso a função a ser otimizada seja exatamente quadrática. Agora, no entanto, há a garantia de decrescimento monotônico da função objetivo a cada iteração, qualquer que seja a estrutura da função objetivo. Dessa forma, garante-se o atendimento de todos os requisitos do teorema da convergência global num sentido similar ao da convergência do Algoritmo do Gradiente. Agora, para estabelecer a completa equivalência da região de convergência do Algoritmo de Newton Modificado com a do Algoritmo do Gradiente, basta mostrar que o primeiro é bem definido na mesma região de convergência do último, ou seja, na bacia de atração. Isto é assegurado pela proposição a seguir.

Proposição 3.2 *Seja $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ uma função contínua infinitas vezes diferenciável. Seja $\boldsymbol{x}^*$ um mínimo local estrito dessa função. Sob tais condições, a Hessiana de $f(\cdot)$ é definida positiva na bacia de atração de $\boldsymbol{x}^*$.* □

Há a necessidade de diferenciabilidade infinita de $f(\cdot)$ neste caso, ao contrário das funções otimizadas com o algoritmo do gradiente, que precisam apenas ser diferenciáveis uma vez. Isso decorre da possibilidade que haveria, se não se colocasse tal exigência, de se concatenar trechos de hiperplanos por meio de funções suaves, que podem ser diferenciáveis até alguma ordem finita, formando bacias de atração suaves nas quais a Hessiana é nula em diversos trechos. O método de Newton simplesmente não seria definido para tais funções. O mínimo local, agora, ainda deve ser estrito, pois do contrário poderia ter posto incompleto, também invalidando a iteração de Newton.

Definidas essas exigências para a aplicabilidade do método, é possível estabelecer a região de convergência.

Proposição 3.3 *Seja o problema de otimização irrestrito definido por:*

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3.22}$$

sendo $\boldsymbol{x} \in \mathbb{R}^n$*, com* $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ *uma função contínua infinitas vezes diferenciável, e* $\boldsymbol{x}^*$ *um mínimo estrito. Então o Algoritmo de Newton Modificado irá convergir para* $\boldsymbol{x}^*$ *para todo ponto inicial* $\boldsymbol{x}_0$ *situado na bacia de atração de* $\boldsymbol{x}^*$*.* □

Corolário 3.2 *Garantidas as condições da proposição anterior, caso o Algoritmo de Newton Modificado seja iniciado em um ponto* $\boldsymbol{x}_0$ *não situado na bacia de atração do mínimo global* $\boldsymbol{x}^*$*, podem ocorrer três situações:*

(i) O Algoritmo de Newton Modificado converge para o mínimo local estrito associado à bacia de atração em que estiver localizado seu ponto inicial $\boldsymbol{x}_0$*.*

(ii) Caso o ponto inicial esteja localizado em uma bacia de atração de um mínimo local não estrito, o Algoritmo de Newton Modificado pode ficar indefinido, ou seja, a Hessiana pode não ser inversível. Caso contrário, ocorrerá convergência para o mínimo local.

(iii) Caso o ponto inicial não esteja localizado em nenhuma bacia de atração, o Algoritmo de Newton Modificado não converge, podendo ainda ficar indefinido.

□

NOTA 3.5 *O leitor deve estar ciente de que existem procedimentos ad-hoc para evitar que a "Hessiana"utilizada pelo algoritmo fique não inversível, ao custo da perda de fidelidade para representar a verdadeira Hessiana da função, porém garantindo as propriedades de convergência do algoritmo. Para maiores informações, ver [1].*

3.6.3 Determinação Numérica da Hessiana

Para a implementação do método de Newton é necessário o cálculo numérico da Hessiana. Por meio de um hipotético método de diferenças finitas, seria necessário avaliar o *gradiente* da função objetivo em $n+1$ pontos, no caso de uma função de n variáveis. Sendo $\boldsymbol{g}(\boldsymbol{x})$ o gradiente da função objetivo, avaliado numericamente por meio de diferenças finitas, como já visto, o método de cálculo da Hessiana por diferenças finitas pode ser formulado como:

Algorithm 12: Algoritmo do Cálculo da Hessiana por Diferenças Finitas

```
1 k ← 0;
2 for (i ← 1 until n) do
3 |   F_i ← [g(x + δe_i) − g(x)] /δ;
4 end
5 F ← [F_1  ···  F_n];
```

Cada uma das avaliações de gradiente por sua vez envolve, como já se viu, a avaliação da função objetivo em $n+1$ pontos, de forma que o número total de avaliações da função objetivo seria igual a $(n+1)^2$.

3.6.4 Construção da Hessiana

Examine-se novamente a equação (3.19), reproduzida a seguir por conveniência:

$$\nabla f(\boldsymbol{x}) = \nabla f(\boldsymbol{x}_0) + H(\boldsymbol{x}_0)(\boldsymbol{x} - x_0) \tag{3.23}$$

Essa equação foi o ponto de partida para a construção do método de Newton. Ela pode também ser usada para construir um método para estimar a própria Hessiana da função. Reescrevendo a equação para dois pontos distintos $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$, e supondo que a Hessiana seja constante em todo o espaço, tem-se:

$$H(\boldsymbol{x}_1 - \boldsymbol{x}_2) = \nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2) \tag{3.24}$$

Essa mesma fórmula pode ser repetida para a seguinte sequência de vetores:

$$\begin{aligned} H(\boldsymbol{x}_1 - \boldsymbol{x}_2) &= \nabla f(\boldsymbol{x}_1) - \nabla f(\boldsymbol{x}_2) \\ H(\boldsymbol{x}_2 - \boldsymbol{x}_3) &= \nabla f(\boldsymbol{x}_2) - \nabla f(\boldsymbol{x}_3) \\ &\vdots \\ H(\boldsymbol{x}_{n-1} - \boldsymbol{x}_n) &= \nabla f(\boldsymbol{x}_{n-1}) - \nabla f(\boldsymbol{x}_n) \\ H(\boldsymbol{x}_n - \boldsymbol{x}_{n+1}) &= \nabla f(\boldsymbol{x}_n) - \nabla f(\boldsymbol{x}_{n+1}) \end{aligned} \tag{3.25}$$

Definindo os vetores $\boldsymbol{v}_i$ e $\boldsymbol{r}_i$ como:

$$\begin{aligned} \boldsymbol{v}_i &= \boldsymbol{x}_i - \boldsymbol{x}_{i+1} \\ \boldsymbol{r}_i &= \nabla f(\boldsymbol{x}_i) - \nabla f(\boldsymbol{x}_{i+1}) \end{aligned} \tag{3.26}$$

tem-se que:

$$H\begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n\end{bmatrix} = \begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_n\end{bmatrix} \tag{3.27}$$

Definindo $V = \begin{bmatrix}\boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_n\end{bmatrix}$ e $R = \begin{bmatrix}\boldsymbol{r}_1 & \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_n\end{bmatrix}$, obtém-se:

$$HV = R \tag{3.28}$$

Observando agora que os vetores $\boldsymbol{v}_i$ tratam-se de *escolhas*, nota-se que é possível escolhê-los de tal forma que V seja inversível, o que permite fazer:

$$H = RV^{-1} \tag{3.29}$$

Isso significa que, avaliando o gradiente da função $f(\boldsymbol{x})$ em $n+1$ pontos adequadamente escolhidos no espaço, é possível determinar a Hessiana dessa função.

Nota 3.6 *Examinando-se o resultado obtido, verifica-se que a equação (3.28) é uma generalização do cálculo da Hessiana por diferenças finitas. De fato, fazendo-se $V = \delta I$ tem-se de (3.28) que $H = R/\delta$.*

Nota 3.7 *Da mesma forma como o cálculo do gradiente por diferenças finitas é exato para funções polinomiais de grau 1, o cálculo da Hessiana por (3.29) é exato para funções polinomiais de grau 2 (desde que se disponha, no entanto, de avaliações exatas do gradiente). Pelo mesmo motivo que no caso da avaliação exata do gradiente em funções lineares, caso a função seja quadrática, não é necessário que os*

pontos em que se avalia o gradiente estejam próximos entre si para que o cálculo da Hessiana permaneça exato.

Diversos métodos de otimização baseiam-se na equação (3.29), variando-se, de método para método, a escolha dos pontos, o que implica na variação da escolha de V.

3.6.5 Correção de Posto 1

Conforme foi visto, há certa arbitrariedade na escolha dos vetores $\boldsymbol{v}_i$. A única condição necessária é de que sejam n vetores linearmente independentes. Dessa forma, é possível acrescentar restrições ao problema de forma a obter fórmulas recursivas particularmente interessantes.

A ideia a ser explorada aqui é a de que deve ser possível fazer a construção recursiva da estimativa da Hessiana, ou de sua inversa, durante o decorrer de um processo de otimização. A estimativa parcial da Hessiana deve poder ser utilizada no decorrer desse processo. Isso é particularmente útil na otimização de funções não-quadráticas, em que a Hessiana não é constante: esse procedimento permite a adaptação contínua da estimativa da Hessiana ao seu valor localmente válido.

É mostrado inicialmente o algoritmo mais simples possível para realizar o procedimento pretendido, que será aqui denominado *Algoritmo de Correção de Posto 1*.

Seja $\widetilde{H}_k = H_k^{-1}$. A ideia é construir um método recursivo que produza uma sequência de estimativas $[\widetilde{H}_k]$, a partir de novas avaliações da função e de seu gradiente em novos pontos. Observa-se inicialmente que a Hessiana de toda função é simétrica, de forma que a recursão deve gerar uma matriz simétrica. A recursão proposta é da forma:

$$\widetilde{H}_{k+1} = \widetilde{H}_k + \alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T \tag{3.30}$$

sendo $\boldsymbol{z}_k \in \mathbb{R}^n$ e $\alpha_k \in \mathbb{R}$. Claramente, o termo $\alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T$ é uma matrix $n \times n$ com posto no máximo igual a 1, de onde vem o nome do algoritmo. Supondo, para fins de desenvolvimento da fórmula de recursão, que a função objetivo fosse exatamente quadrática, é preciso definir α_k e $\boldsymbol{z}_k$ em função dos valores conhecidos (os vetores $[\boldsymbol{x}_k]$ e $[\nabla f(\boldsymbol{x}_k)]$), de forma a garantir que seja satisfeita a relação:

$$\widetilde{H}_{k+1} \boldsymbol{r}_i = \boldsymbol{v}_i \quad \forall\ i = 1, \ldots, k \tag{3.31}$$

Essa relação é quase a mesma que (3.28), mas exige a igualdade apenas para os pontos já avaliados, até o índice k. Em primeiro lugar, desenvolve-se a fórmula para $i = k$. Substituindo-se (3.30) em (3.31), obtém-se:

$$\alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T \boldsymbol{r}_k = \boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k$$

$$(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)^T = (\alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T \boldsymbol{r}_k)(\alpha_k \boldsymbol{r}_k^T \boldsymbol{z}_k \boldsymbol{z}_k^T) \tag{3.32}$$

$$(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)^T = \alpha_k (\boldsymbol{z}_k^T \boldsymbol{r}_k)^2 \alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T$$

Com isso, quase se tem uma fórmula para o termo de correção $\alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T$ em função de $\widetilde{H}_k$, $\boldsymbol{v}_k$ e $\boldsymbol{r}_k$, a menos da quantidade escalar $\alpha_k(\boldsymbol{z}_k^T \boldsymbol{r}_k)^2$. Para se determinar essa constante, faz-se:

$$\begin{aligned} \boldsymbol{r}_k^T \alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T \boldsymbol{r}_k &= \boldsymbol{r}_k^T (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k) \\ \alpha_k (\boldsymbol{z}_k^T \boldsymbol{r}_k)^2 &= \boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k \end{aligned} \tag{3.33}$$

Substituindo-se (3.33) em (3.32) obtém-se:

$$\alpha_k \boldsymbol{z}_k \boldsymbol{z}_k^T = \frac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)^T \tag{3.34}$$

ou, voltando à fórmula recursiva para cálculo de $\widetilde{H}_{k+1}$:

$$\widetilde{H}_{k+1} = \widetilde{H}_k + \frac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)^T \tag{3.35}$$

Essa fórmula, por construção, vale para $i = k$. Resta provar que ela é válida para $i < k$.

Teorema 3.3 *Seja F uma matriz simétrica fixa, e suponha-se que $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_k$ sejam vetores dados. Definam-se os vetores $\boldsymbol{r}_i = H\boldsymbol{v}_i$, para $i = 0, 1, \ldots, k$. Seja ainda H_0 uma matriz simétrica qualquer. Se:*

$$\widetilde{H}_{i+1} = \widetilde{H}_i + \frac{1}{\boldsymbol{r}_i^T \boldsymbol{v}_i - \boldsymbol{r}_i^T \widetilde{H}_i \boldsymbol{r}_i} (\boldsymbol{v}_i - \widetilde{H}_i \boldsymbol{r}_i)(\boldsymbol{v}_i - \widetilde{H}_i \boldsymbol{r}_i)^T \tag{3.36}$$

então:

$$\boldsymbol{v}_i = \widetilde{H}_{k+1} \boldsymbol{r}_i \quad \forall\, i = 1, \ldots, k \tag{3.37}$$

□

DEMONSTRAÇÃO: Por construção, a relação é válida para $i = k$. Tome-se algum $\boldsymbol{r}_i$ para $i < k$, e aplique-se esse vetor em $\widetilde{H}_{k+1}$:

$$\widetilde{H}_{k+1} \boldsymbol{r}_i = \widetilde{H}_k \boldsymbol{r}_i + \tfrac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k^T \boldsymbol{r}_i - \boldsymbol{r}_k^T \widetilde{H}_k^T \boldsymbol{r}_i)$$

Note-se que $\widetilde{H}_k$ é simétrica, de forma que:

$$\widetilde{H}_{k+1} \boldsymbol{r}_i = \widetilde{H}_k \boldsymbol{r}_i + \tfrac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k^T \boldsymbol{r}_i - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_i)$$

Adota-se neste ponto, como hipótese de indução, que

$$\boldsymbol{v}_i = \widetilde{H}_k \boldsymbol{r}_i$$

seja verdade. Isso implica que:

$$\widetilde{H}_{k+1} \boldsymbol{r}_i = \boldsymbol{v}_i + \tfrac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k^T \boldsymbol{r}_i - \boldsymbol{r}_k^T \boldsymbol{v}_i)$$

Entretanto:

$$\boldsymbol{r}_k^T \boldsymbol{v}_i = \boldsymbol{v}_k^T H^T \boldsymbol{v}_i = \boldsymbol{v}_k^T H \boldsymbol{v}_i = \boldsymbol{v}_k^T \boldsymbol{r}_i$$

de forma que:

$$\boldsymbol{v}_k^T \boldsymbol{r}_i - \boldsymbol{r}_k^T \boldsymbol{v}_i = 0$$

ou:

$$\widetilde{H}_{k+1} \boldsymbol{r}_i = \boldsymbol{v}_i$$

Isso completa a prova. ■

Sabe-se então que, usando-se a fórmula (3.35), obtém-se o valor exato da inversa da Hessiana de uma função quadrática, a partir de $n+1$ valores de pontos do espaço com as respectivas avaliações de gradientes da função nesses pontos.

Algoritmo de Correção de Posto 1

Com esse resultado, é possível construir um algoritmo de otimização utilizando a estrutura básica da "direção de busca", tomando $\widetilde{H}_k$ como aproximação da inversa da Hessiana. O algoritmo se inicia em um ponto $\boldsymbol{x}_0$ qualquer:

Algorithm 13: Algoritmo de Correção de Posto 1

$k \leftarrow 0$;
$\widetilde{H}_k \leftarrow I$;
$\boldsymbol{g}_k \leftarrow gradiente(f(\cdot), \boldsymbol{x}_k)$;
while *(critério de parada não for satisfeito)* **do**
  $\boldsymbol{d}_k \leftarrow -\widetilde{H}_k \boldsymbol{g}_k$;
  $\alpha_k \leftarrow \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$;
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;
  $\boldsymbol{g}_{k+1} \leftarrow gradiente(f(\cdot), \boldsymbol{x}_{k+1})$;
  $\boldsymbol{v}_k \leftarrow \boldsymbol{x}_k - \boldsymbol{x}_{k+1}$;
  $\boldsymbol{r}_k \leftarrow \boldsymbol{g}_k - \boldsymbol{g}_{k+1}$;
  $\widetilde{H}_{k+1} = \widetilde{H}_k + \frac{1}{\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} (\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)(\boldsymbol{v}_k - \widetilde{H}_k \boldsymbol{r}_k)^T$;
  $k \leftarrow k + 1$;
end

Deve-se notar que, de maneira arbitrária, a estimativa $\widetilde{H}_0$ foi inicializada com a matriz identidade. Qualquer outra matriz simétrica poderia ter sido utilizada, de acordo com o Teorema 3.3. Esse teorema, juntamente com o resultado anteriormente conhecido a respeito de aproximações quadráticas em geral, afirma que se a função objetivo for quadrática, a convergência exata do algoritmo para o mínimo global da função necessariamente ocorrerá, e o número de passos para tal convergência será menor ou igual a n. Note-se que, ao invés de serem tomados pontos quaisquer

que gerem vetores $\boldsymbol{v}_i$ linearmente independentes, estão sendo tomados exatamente aqueles pontos gerados pelo processo de otimização. Caso a função seja exatamente quadrática, estes pontos *geram* necessariamente vetores $\boldsymbol{v}_i$ linearmente independentes.

Sob o ponto de vista da otimização de uma função *a priori* sabida ser quadrática, não há vantagem computacional em se utilizar o *Algoritmo de Correção de Posto 1* em lugar da fórmula exata (3.21) junto com (3.29). A aplicação destas envolveria exatamente $n+1$ avaliações de gradiente, enquanto a aplicação do algoritmo de correção envolveria um número menor ou igual a este de iterações, cada uma envolvendo uma avaliação de gradiente, mas envolvendo também uma otimização unidimensional. Esta última poderia tornar o algoritmo de correção mais caro sob o ponto de vista computacional.

No entanto, sabe-se que no caso geral da otimização de funções não-lineares não necessariamente quadráticas, a Hessiana da função objetivo não será em geral constante. Não ocorrerá, de qualquer forma, a convergência em n iterações. O Algoritmo de Correção de Posto 1 torna-se nesse caso vantajoso, pois a estimativa da Hessiana vai mudando dinamicamente, de forma a acompanhar a variação dessa Hessiana. A cada passo, uma nova estimativa da Hessiana está disponível, para ser utilizada no processo de otimização. Essas são características gerais da categoria de métodos conhecidos como quase-Newton, que será vista a seguir.

Nota 3.8 *Deve-se notar qua a primeira iteração do Algoritmo de Correção de Posto 1, no formato anteriormente definido, corresponde exatamente a uma iteração do Algorimto do Gradiente. Isso ocorre porque, com a matriz H_0 sendo inicializada igual à identidade, no primeiro passo a direção de busca fica sendo igual à do gradiente. A partir do segundo passo, a direção começa a mudar gradativamente, até que no n-ésimo passo a direção passa a coincidir com a do Algoritmo de Newton, caso a Hessiana seja constante. Caso a Hessiana não seja constante, a estimativa do Algoritmo de Correção de Posto 1 será sempre inexata, e este algoritmo não chegará a convergir para o comportamento hipotético do Algoritmo de Newton. No entanto, como a avaliação direta da Hessiana, exigida pelo Algoritmo de Newton, é inconveniente, na prática usualmente se opta pela utilização de algoritmos quase-Newton.*

Convergência do Algoritmo de Correção de Posto 1

O *Algoritmo de Correção de Posto 1* possui propriedades de convergência que são intermediárias entre as do Algoritmo do Gradiente e as do Algoritmo de Newton. Este último simplesmente não se aplica quando a Hessiana não é definida positiva. Já o Algoritmo do Gradiente exige apenas a existência de uma bacia de atração de uma função diferenciável. O Algoritmo de Correção de Posto 1 não pode ficar indefinido em nenhum ponto, uma vez que não envolve inversões de matrizes. No entanto, sua formulação permite que a matriz H_{k+1} venha eventualmente perder a propriedade de ser positiva definida, caso ocorra:

$$\boldsymbol{r}_k^T \boldsymbol{v}_k - \boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k < 0 \tag{3.38}$$

Não há nada que impeça essa condição de ocorrer. Isso significa que a fórmula de correção pode eventualmente vir a ficar comprometida. Isso pode fazer com que o algoritmo fique estacionado em pontos que não correspondem à solução do problema. Pode-se evitar tal situação incluindo-se uma verificação dos autovalores de H_{k+1} a cada passo, fazendo-se a substituição dessa matriz pela identidade sempre que for detectado um autovalor negativo. Isso iria restaurar as condições do *teorema da convergência global*, e o algoritmo passaria a convergir exatamente na mesma região que o Algoritmo do Gradiente.

3.6.6 Métodos Quase-Newton

Os métodos de otimização conhecidos como *quase-Newton* são desenvolvidos de acordo com a mesma lógica que foi usada na elaboração do *Algoritmo de Correção de Posto 1*; de fato, este algoritmo é o exemplo mais simples de um algoritmo quase-Newton. Usa-se uma regra recursiva que permite a construção iterativa de uma matriz $\widetilde{H}_k$ que corresponde a uma estimativa da inversa da Hessiana da função objetivo. Como deve ter sido observado na seção anterior, diversas escolhas arbitrárias de regras foram realizadas, de forma que outras escolhas teriam sido possíveis para garantir as propriedades desejadas de $\widetilde{H}_k$. Com os graus de liberdade ainda remanescentes, é possível produzir métodos que evitem as dificuldades de convergência do *Algoritmo de Correção de Posto 1*: essencialmente, deve-se garantir que a matriz $\widetilde{H}_k$ permaneça sempre definida positiva, e, preferencialmente, bem condicionada, ou seja, com autovalores não muito distanciados entre si.

Dois métodos particularmente eficientes foram desenvolvidos para produzir estimativas recursivas para H_k com as propriedades requeridas: o método DFP (Davidon-Fletcher-Powell) e o método BFGS (Broyden-Fletcher-Goldfarb-Shanno), assim batizados em homenagem aos seus formuladores. Verificando-se, *a posteriori*, as conexões entre esses métodos, estes foram agrupados em uma estrutura mais geral, a *família de Broyden*. Esses métodos são apresentados a seguir.

Método DFP

A correção proposta pelo método DFP é dada por:

$$C_k^{DFP} = \frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{\boldsymbol{v}_k^T \boldsymbol{r}_k} - \frac{\widetilde{H}_k \boldsymbol{r}_k \boldsymbol{r}_k^T \widetilde{H}_k}{\boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k} \tag{3.39}$$

Método BFGS

A correção proposta pelo método BFGS é dada por:

$$C_k^{BFGS} = \left(1 + \frac{\boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k}{\boldsymbol{r}_k^T \boldsymbol{v}_k}\right) \frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{\boldsymbol{v}_k^T \boldsymbol{r}_k} - \frac{\boldsymbol{v}_k \boldsymbol{r}_k^T \widetilde{H}_k + \widetilde{H}_k \boldsymbol{r}_k \boldsymbol{v}_k^T}{\boldsymbol{r}_k^T \boldsymbol{v}_k} \tag{3.40}$$

Família de Broyden

A correção genérica utilizada pelos métodos conhecidos como *família de Broyden* é dada por:

$$C_k(\xi) = (1-\xi)\, C_k^{DFP} + \xi C_k^{BFGS} \tag{3.41}$$

Em todos os casos da família de Broyden, incluindo os casos extremos BFGS e DFP, a fórmula de atualização para a estimativa da inversa da Hessiana fica:

$$\widetilde{H}_{k+1} = \widetilde{H}_k + C_k\,(\xi) \tag{3.42}$$

Para $\xi = 0$, obtém-se o método DFP, e para $\xi = 1$ o método BFGS.

Alguns fatos devem ser citados a respeito dessa correção da família de Broyden:

- A correção realizada a cada passo é de posto possivelmente dois, o que é facilmente verificável por inspeção.
- A correção é sempre definida positiva, de forma que a matriz $\widetilde{H}_k$ preservará sua propriedade de ser definida positiva.
- Dados i e j tais que $0 \leq i < j \leq k$, então $\boldsymbol{v}_i^T H \boldsymbol{v}_j = 0$, ou seja, $\boldsymbol{v}_i$ e $\boldsymbol{v}_j$ são H-ortogonais.
- Dado i tal que $0 \leq i \leq k$, então $\widetilde{H}_{k+1} H \boldsymbol{v}_i = \boldsymbol{v}_i$.

As provas das afirmações anteriores podem ser encontradas em [1].

Algoritmos Quase-Newton

Os algoritmos obtidos a partir da família de Broyden, aqui denominados *Algoritmos Quase-Newton*, são estruturados no Algoritmo 14.

Evidentemente, para a implementação pura do Algoritmo DFP ou do Algoritmo BFGS, não haveria necessidade do cálculo intermediário de $C_k(\xi)$, sendo possível simplificar o programa, para o cálculo direto de $\widetilde{H}_k$ com a correção correspondente.

Convergência da Família Broyden

A maneira mais fácil de provar a convergência dos algoritmos da família de Broyden seria introduzindo uma modificação nos mesmos: se se faz com que a matriz $\widetilde{H}_k$ seja periodicamente reinicializada, sendo igualada à identidade, torna-se possível a aplicação direta do teorema da convergência global. Os algoritmos passam a convergir exatamente como o Algoritmo do Gradiente.

É possível, mesmo sem introduzir tal modificação, provar a convergência dos algoritmos, sendo necessárias entretanto algumas premissas adicionais sobre a função a ser otimizada.

Exemplo

A utilização do algoritmo do método DFP no problema exemplo definido pela equação (3.5) resulta na trajetória ilustrada na Fig. 3.6 a seguir.

Algorithm 14: Algoritmos Quase-Newton

$k \leftarrow 0$;
$\widetilde{H}_k \leftarrow I$;
$\boldsymbol{g}_k \leftarrow gradiente(f(\cdot), \boldsymbol{x}_k)$;
while *(critério de parada não for satisfeito)* **do**
  $\boldsymbol{d}_k \leftarrow -\widetilde{H}_k \boldsymbol{g}_k$;
  $\alpha_k \leftarrow \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$;
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;
  $\boldsymbol{g}_{k+1} \leftarrow gradiente(f(\cdot), \boldsymbol{x}_{k+1})$;
  $\boldsymbol{v}_k \leftarrow \boldsymbol{x}_k - \boldsymbol{x}_{k+1}$;
  $\boldsymbol{r}_k \leftarrow \boldsymbol{g}_k - \boldsymbol{g}_{k+1}$;
  $C_k^{DFP} = \frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{\boldsymbol{v}_k^T \boldsymbol{r}_k} - \frac{\widetilde{H}_k \boldsymbol{r}_k \boldsymbol{r}_k^T \widetilde{H}_k}{\boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k}$;
  $C_k^{BFGS} = \left(1 + \frac{\boldsymbol{r}_k^T \widetilde{H}_k \boldsymbol{r}_k}{\boldsymbol{r}_k^T \boldsymbol{v}_k}\right) \frac{\boldsymbol{v}_k \boldsymbol{v}_k^T}{\boldsymbol{v}_k^T \boldsymbol{r}_k} - \frac{\boldsymbol{v}_k \boldsymbol{r}_k^T \widetilde{H}_k + \widetilde{H}_k \boldsymbol{r}_k \boldsymbol{v}_k^T}{\boldsymbol{r}_k^T \boldsymbol{v}_k}$;
  $C_k(\xi) = (1 - \xi) C_k^{DFP} + \xi C_k^{BFGS}$;
  $\widetilde{H}_{k+1} = \widetilde{H}_k + C_k(\xi)$;
  $k \leftarrow k + 1$;
end

3.6.7 Método do Gradiente Conjugado

O método de otimização conhecido como Gradiente Conjugado foi desenvolvido inicialmente na década de 1950 para a solução de sistemas de equações lineares, e ainda é um dos métodos mais utilizados para a solução de sistemas com matrizes esparsas. Em 1964, Fletcher e Reeves generalizaram o método para resolver problemas de otimização não linear irrestrita com funções não quadráticas.

Considere um sistema de equações lineares da forma:

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{3.43}$$

sendo $\boldsymbol{A}$ uma matriz simétrica definida positiva.

A solução desse sistema de equações por meio do cálculo da inversa de $\boldsymbol{A}$ é impraticável para sistemas grandes, por demandar muito esforço computacional. Por essa razão, é interessante utilizar um método iterativo para a solução desse sistema.

Para isso, vamos considerar um problema de minimização da função quadrática a seguir:

$$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}'\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\boldsymbol{x} + c \tag{3.44}$$

O mínimo global dessa função é obtido a partir da condição de otimalidade de primeira ordem:

$$\nabla f(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b} = 0 \tag{3.45}$$

Portanto, o mínimo de $f(\boldsymbol{x})$ é também a solução do sistema linear (3.43), isto é, podemos resolver o sistema linear (3.43) minimizando a função quadrática associada

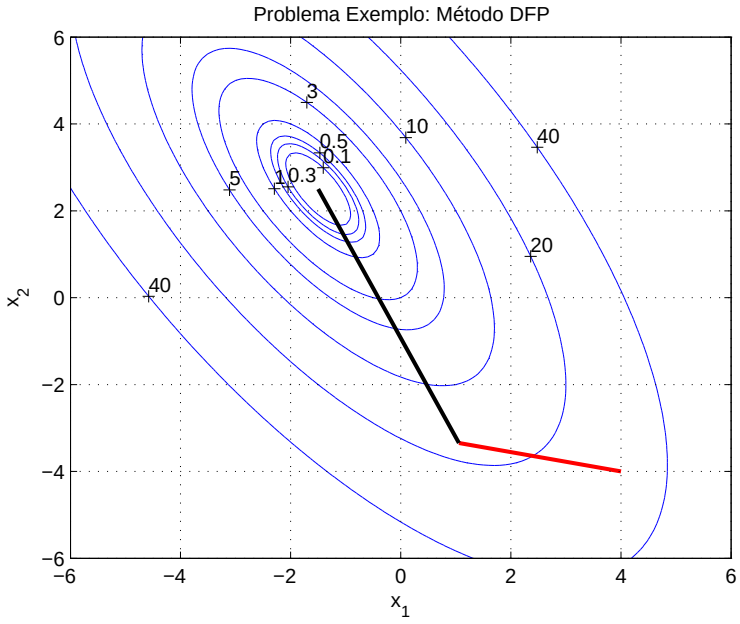

Figura 3.6: Problema exemplo – solução usando o Método DFP.

(3.44). Devido às particularidades do problema, podemos desenvolver um método do gradiente com cálculo exato do tamanho de passo.

A direção oposta ao gradiente equivale ao resíduo na solução do sistema (3.43):

$$-\nabla f(\boldsymbol{x}) = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} = \boldsymbol{r} \tag{3.46}$$

Assim, usaremos a seguinte fórmula iterativa baseado no método do gradiente:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{r}_k \tag{3.47}$$

O tamanho do passo pode ser determinado analiticamente:

$$\begin{aligned}
\frac{d}{d\alpha} f(\boldsymbol{x}_{k+1}) &= \nabla f(\boldsymbol{x}_{k+1})' \frac{d}{d\alpha}(\boldsymbol{x}_{k+1}) \\
&= \nabla f(\boldsymbol{x}_{k+1})' \frac{d}{d\alpha}(\boldsymbol{x}_k + \alpha_k \boldsymbol{r}_k) \\
&= -\boldsymbol{r}'_{k+1} \boldsymbol{r}_k
\end{aligned}$$

Fazendo

$$\frac{d}{d\alpha} f(\boldsymbol{x}_{k+1}) = -\boldsymbol{r}'_{k+1} \boldsymbol{r}_k = 0 \tag{3.48}$$

implica que os resíduos são ortogonais, ou seja:

$$\begin{aligned}
\boldsymbol{r}'_{k+1} \boldsymbol{r}_k &= 0 \\
(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k+1})' \boldsymbol{r}_k &= 0 \\
(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k - \alpha_k \boldsymbol{A}\boldsymbol{r}_k)' \boldsymbol{r}_k &= 0 \\
(\boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{r}_k)' \boldsymbol{r}_k &= 0
\end{aligned}$$

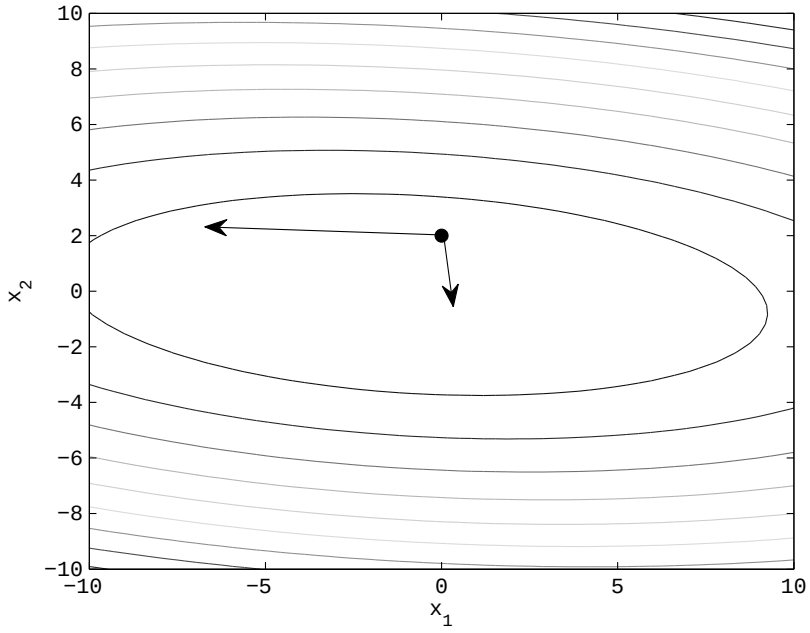

Figura 3.7: Ilustração de dois vetores conjugados em relação à matriz Hessiana da função quadrática cujas curvas de nível são mostradas.

que resulta em:

$$\alpha_k = \frac{\boldsymbol{r}_k' \boldsymbol{r}_k}{\boldsymbol{r}_k' \boldsymbol{A} \boldsymbol{r}_k} \tag{3.49}$$

Usando o passo ótimo determinado em (3.49) na fórmula iterativa (3.47), estamos minimizando a função quadrática (3.44) pelo método do gradiente. A minimização dessa função quadrática nos leva à solução do sistema linear original $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Contudo, essa abordagem herda os problemas do método do gradiente já discutidos anteriormente. Por essa razão, desenvolveu-se o método do gradiente conjugado, em que, além de se forçar a ortogonalidade dos resíduos em iterações sucessivas, utiliza-se uma fórmula iterativa da forma:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{3.50}$$

em que as direções $\boldsymbol{d}_k$ são conjugadas entre si.

Dois vetores $\boldsymbol{v}_i$ e $\boldsymbol{v}_j$ são ditos conjugados em relação à matriz $\boldsymbol{A}$, ou simplesmente $\boldsymbol{A}$-conjugados, se vale a relação:

$$\boldsymbol{v}_i' \boldsymbol{A} \boldsymbol{v}_j = \boldsymbol{v}_j' \boldsymbol{A} \boldsymbol{v}_i = 0 \tag{3.51}$$

Essa ideia é ilustrada na Figura 3.7. Se usarmos a transformação de variáveis $\boldsymbol{z} = \boldsymbol{B}\boldsymbol{x}$ de tal forma que $\boldsymbol{A} = \boldsymbol{B}'\boldsymbol{B} = \boldsymbol{B}^2$, então dois vetores $\boldsymbol{x}_i$ e $\boldsymbol{x}_j$ $\boldsymbol{A}$-conjugados serão ortogonais no espaço de variáveis transformado. Portanto, é intuitivo perceber que a noção de conjugação de dois vetores em relação à matriz $\boldsymbol{A}$ traz consigo informação sobre a curvatura do espaço, isto é, a Hessiana da função quadrática (3.44). Ao forçar que a próxima direção de busca $\boldsymbol{d}_{k+1}$ seja conjugada à direção

$\boldsymbol{d}_k$ estamos calculando uma nova direção de busca que implicitamente considera informação sobre a curvatura da função quadrática.

O método do gradiente conjugado utiliza as fórmulas recursivas a seguir:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{3.52}$$
$$\boldsymbol{r}_{k+1} = \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{d}_k \tag{3.53}$$
$$\boldsymbol{d}_{k+1} = \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{d}_k \tag{3.54}$$

com a imposição de que as direções $\boldsymbol{d}_k$ são $\boldsymbol{A}$-conjugadas e os resíduos são ortogonais entre si. Aplicando essas condições de ortogonalidade e conjugação, podemos derivar as fórmulas para α_k e β_k.

Inicialmente, vamos verificar a equação de atualização dos resíduos:

$$\begin{aligned}
\boldsymbol{r}_{k+1} &= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{k+1} \\
&= \boldsymbol{b} - \boldsymbol{A}\left(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k\right) \\
&= \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_k - \alpha_k \boldsymbol{A}\boldsymbol{d}_k \\
&= \boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{d}_k
\end{aligned}$$

A partir da condição de ortogonalidade dos resíduos, temos:

$$\begin{aligned}
\boldsymbol{r}_{k+1}'\boldsymbol{r}_k &= 0 \\
\left(\boldsymbol{r}_k - \alpha_k \boldsymbol{A}\boldsymbol{d}_k\right)'\boldsymbol{r}_k &= 0 \\
\boldsymbol{r}_k'\boldsymbol{r}_k - \alpha_k \boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{r}_k &= 0
\end{aligned}$$

que fornece

$$\alpha_k = \frac{\boldsymbol{r}_k'\boldsymbol{r}_k}{\boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{r}_k} \tag{3.55}$$

mas $\boldsymbol{r}_k = \boldsymbol{d}_k - \beta_{k-1}\boldsymbol{d}_{k-1}$, ver (3.54), assim:

$$\alpha_k = \frac{\boldsymbol{r}_k'\boldsymbol{r}_k}{\boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{d}_k - \beta_{k-1}\boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{d}_{k-1}} \tag{3.56}$$

Como os vetores $\boldsymbol{d}_k$ e $\boldsymbol{d}_{k-1}$ são conjugados, então:

$$\alpha_k = \frac{\boldsymbol{r}_k'\boldsymbol{r}_k}{\boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{d}_k} \tag{3.57}$$

A partir da condição de conjugação dos vetores de direção, temos:

$$\begin{aligned}
\boldsymbol{d}_{k+1}'\boldsymbol{A}\boldsymbol{d}_k &= 0 \\
\left(\boldsymbol{r}_{k+1} + \beta_k \boldsymbol{d}_k\right)'\boldsymbol{A}\boldsymbol{d}_k &= 0 \\
\boldsymbol{r}_{k+1}'\boldsymbol{A}\boldsymbol{d}_k + \beta_k \boldsymbol{d}_k'\boldsymbol{A}\boldsymbol{d}_k &= 0
\end{aligned}$$

que fornece

$$\beta_k = -\frac{\boldsymbol{r}'_{k+1}\boldsymbol{A}\boldsymbol{d}_k}{\boldsymbol{d}'_k\boldsymbol{A}\boldsymbol{d}_k} \tag{3.58}$$

Isolando o termo $\boldsymbol{A}\boldsymbol{d}_k$ em (3.53) e substituindo no numerador de (3.58):

$$\beta_k = -\frac{\boldsymbol{r}'_{k+1}(\boldsymbol{r}_k - \boldsymbol{r}_{k+1})}{\alpha_k \boldsymbol{d}'_k\boldsymbol{A}\boldsymbol{d}_k} \tag{3.59}$$

Lembrando que $\boldsymbol{r}_k$ e $\boldsymbol{r}_{k+1}$ são ortogonais, temos

$$\beta_k = \frac{\boldsymbol{r}'_{k+1}\boldsymbol{r}_{k+1}}{\boldsymbol{r}'_k\boldsymbol{r}_k} \tag{3.60}$$

Com base nessas equações, temos o algoritmo do gradiente conjugado para minimização de funções quadráticas da forma (3.44) ou, de maneira equivalente, para a solução de um sistema de equações lineares com matriz de coeficientes simétrica definida positiva $\boldsymbol{A}$.

Algorithm 15: Algoritmo dos Gradientes Conjugados

```
1  k ← 0;
2  r_k ← b − Ax_k;
3  d_k ← r_k;
4  while (critério de parada não for satisfeito) do
5  |  α_k ← (r'_k r_k)/(d'_k A d_k) ;
6  |  x_{k+1} ← x_k + α_k d_k ;
7  |  r_{k+1} ← r_k − α_k A d_k ;
8  |  β_k ← (r'_{k+1} r_{k+1})/(r'_k r_k) ;
9  |  d_{k+1} ← r_{k+1} + β_k d_k ;
10 |  k ← k + 1;
11 end
```

O método converge para o ponto de mínimo de uma função quadrática com n variáveis em n iterações. De fato, o método aproxima a solução do problema por meio da seguinte expansão:

$$\boldsymbol{x}^* = \boldsymbol{x}_0 + \sum_{k=1}^{n} \alpha_k \boldsymbol{d}_k \tag{3.61}$$

Método dos Gradientes Conjugados para otimização não linear

Como vimos o método dos gradientes conjugados foi desenvolvido a partir da observação de que o problema de resolver um sistema linear com matriz de coeficientes simétrica definida positiva é equivalente à minimização de uma função quadrática convexa. Mais tarde, observou-se que é possível adaptar o método para resolver problemas de otimização não linear mais gerais, não apenas aqueles envolvendo funções quadráticas. A versão do algoritmo para otimização não linear de funções não quadráticas apresenta três diferenças básicas:

1. A noção de resíduo perde o significado, uma vez que a minimização da função $f(\boldsymbol{x})$ não está mais ligada à solução de um sistema linear do tipo $\boldsymbol{Ax} = \boldsymbol{b}$. Por essa razão não é possível utilizar a fórmula recursiva para atualizar o resíduo. O vetor $\boldsymbol{r}_k$ é utilizado em sua noção mais geral, isto é, a direção oposta ao vetor gradiente.

2. Não é possível determinar o tamanho do passo α_k analiticamente, devendo-se empregar algum método de busca unidirecional para determinar seu valor.

3. Não é possível determinar β_k analiticamente, havendo diferentes escolhas possíveis para esse parâmetro.

Com essas modificações, temos o algoritmo apresentado a seguir.

Algorithm 16: Método dos Gradientes Conjugados

$k \leftarrow 0$;
$\boldsymbol{r}_0 \leftarrow -\nabla f(\boldsymbol{x}_0)$;
$\boldsymbol{d}_0 \leftarrow \boldsymbol{r}_0$;
while *(critério de parada não for satisfeito)* **do**
  $\alpha_k \leftarrow \arg\min_{\alpha} f(\boldsymbol{x}_k + \alpha \boldsymbol{d}_k)$;
  $\boldsymbol{x}_{k+1} \leftarrow \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$;
  $\boldsymbol{r}_{k+1} \leftarrow -\nabla f(\boldsymbol{x}_{k+1})$;
  Calcular β_k;
  $\boldsymbol{d}_{k+1} \leftarrow \boldsymbol{r}_{k+1} + \beta_k \boldsymbol{d}_k$;
  $k \leftarrow k + 1$;
end

Duas fórmulas bem conhecidas para o cálculo do parâmetro β_k são:

$$\text{Fletcher-Reeves: } \beta_k^{FR} = \frac{\boldsymbol{r}'_{k+1}\boldsymbol{r}_{k+1}}{\boldsymbol{r}'_k\boldsymbol{r}_k}$$

$$\text{Polak-Ribière: } \beta_k^{PR} = \frac{\boldsymbol{r}'_{k+1}(\boldsymbol{r}_{k+1} - \boldsymbol{r}_k)}{\boldsymbol{r}'_k\boldsymbol{r}_k}$$

Como agora a função a ser otimizada não é necessariamente uma função quadrática, o método em geral converge em mais do que n iterações. Portanto, como o método produz n direções conjugadas no espaço n-dimensional, deve-se reiniciar o método a cada n iterações, do contrário, as direções de busca deixarão de ser conjugadas. Esse reinício periódico é feito simplesmente fazendo $\boldsymbol{d}_k = \boldsymbol{r}_k$ a cada n iterações. De fato, o método sem reinício periódico converge mais lentamente do que o método adotando reinício periódico. Se a função for quadrática, a convergência continua garantida em n iterações.

Em geral, para funções não quadráticas, os métodos quase-Newton convergem em menos iterações, entretanto consomem mais operações e mais memória por iteração, uma vez que uma aproximação da inversa da matriz Hessiana deve ser armazenada. Além disso, as equações de atualização dessa aproximação demandam

mais operações. O método dos gradientes conjugados requer o armazenamento dos dois últimos vetores gradiente e o vetor $\boldsymbol{d}_k$. Por essa razão, o método de otimização dos gradientes conjugados é o mais indicado e o mais usado em problemas de otimização não linear de elevada dimensão, especificamente problemas com mais de 200 variáveis. Em problemas de baixa e média dimensão, os métodos quase-Newton são mais recomendados.

3.7 Métodos Sem Derivadas

Nas seções anteriores, discutimos métodos baseados em direções de busca que se sustentam no cálculo de derivadas da função objetivo. Contudo, dentro da estrutura básica dos métodos de direções de busca, apresentada no início do capítulo, é possível incluir métodos que definem uma direção de busca $\boldsymbol{d}_k$ que não depende de estimativas do gradiente da função no ponto. Estes métodos são discutidos aqui.

Em geral, métodos baseados em derivadas convergem mais rapidamente, mas só podem ser usados em problemas caracterizados por funções continuamente diferenciáveis, o que nem sempre se verifica em algumas aplicações práticas. Além disso, em problemas com muitas variáveis, os erros numéricos introduzidos por aproximações no cálculo do gradiente podem se tornar significativos, prejudicando a convergência dos métodos baseados em derivadas. Por essas razões, faz-se necessário apresentar alguns métodos numéricos de otimização sem derivadas.

3.7.1 Método de Hooke-Jeeves

O Método Hooke-Jeeves foi proposto na década de 1960 para otimizar funções sem a necessidade de que estas sejam contínuas ou diferenciáveis. O método testa pontos padrões a partir do ponto atual, por essa razão é também conhecido como *Pattern Search* na literatura. O método alterna direções de pesquisa na direção dos eixos coordenados e direções construídas a partir do ponto da iteração anterior, isto é, direções na forma $\boldsymbol{x}_{k+1} - \boldsymbol{x}_k$.

O funcionamento do método é bastante simples. Seja $\boldsymbol{x}_k$ o ponto atual, $\boldsymbol{y}_0 = \boldsymbol{x}_k$, e o vetor $\boldsymbol{e}_i$ associado à i-ésima coluna da matriz identidade. O método testa perturbações na direção de cada eixo coordenado, de forma que um novo ponto $\boldsymbol{y}_{i+1}$ é gerado de acordo com alguma das seguinte situações:

1. $\boldsymbol{y}_i = \boldsymbol{y}_{i-1} + \lambda \boldsymbol{e}_i$ se uma perturbação de magnitude λ na direção positiva da coordenada x_i causa uma melhora no valor da função objetivo;

2. $\boldsymbol{y}_i = \boldsymbol{y}_{i-1} - \lambda \boldsymbol{e}_i$ se uma perturbação de magnitude λ na direção negativa da coordenada x_i causa uma melhora no valor da função objetivo;

3. $\boldsymbol{y}_i = \boldsymbol{y}_{i-1}$ caso contrário.

Note que as perturbações nas direções $\boldsymbol{e}_i$ são acumulativas, isto é, nesta primeira fase, a sequência de pontos $\{\boldsymbol{y}_0, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ representa passos de tamanho λ na direção daqueles eixos coordenados que diminuem o valor da função objetivo. Após serem feitas as perturbações em todas as coordenadas, obtém-se o ponto $\boldsymbol{x}_{k+1} = \boldsymbol{y}_n$.

Observe que o ponto $\boldsymbol{x}_{k+1}$ foi obtido a partir de $\boldsymbol{x}_k$ com perturbações nas direções de melhora de cada variável. Assim, a direção

$$\boldsymbol{d}_k = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k \tag{3.62}$$

sugere uma boa direção de busca em que a função objetivo pode decrescer mais rapidamente. Em geral, essa direção não está alinhada com os eixos coordenados e representa mais uma direção padrão de pesquisa no método. Portanto, pode-se fazer

$$\boldsymbol{y}_0 = \boldsymbol{x}_{k+1} + \alpha\,(\boldsymbol{x}_{k+1} - \boldsymbol{x}_k)$$

começando a próxima iteração a partir deste ponto. O algoritmo do método Hooke-Jeeves é apresentado a seguir.

Algorithm 17: Método Hooke-Jeeves

```
1  k ← 0;
2  y_0 ← x_k;
3  while λ > ξ do
4  |  foreach i = 0 ..., n − 1 do
5  |  |  if f(y_i + λe_{i+1}) < f(y_i) then y_{i+1} ← y_i + λe_{i+1} ;
6  |  |  else if f(y_i − λe_{i+1}) < f(y_i) then y_{i+1} ← y_i − λe_{i+1} ;
7  |  |  else y_{i+1} ← y_i;
8  |  end
9  |  if f(y_n) < f(x_k) then
10 |  |  x_{k+1} ← y_n;
11 |  |  y_0 ← x_{k+1} + α (x_{k+1} − x_k);
12 |  else
13 |  |  λ ← λ/2;
14 |  |  x_{k+1} ← x_k;
15 |  |  y_0 ← x_k;
16 |  end
17 |  k ← k + 1;
18 end
```

As linhas 4–8 executam a fase de busca nos eixos coordenados. Se $\boldsymbol{y}_n$ for igual a $\boldsymbol{x}_k$, significa que não foi possível minimizar a função em nenhuma direção coordenada usando o tamanho atual de λ. Assim, deve-se reduzir o valor de λ, em geral pela metade, o que é feito na linha 13.

O método Hooke-Jeeves é de fácil programação e é competitivo computacionalmente com outros métodos. Modificações podem ser incluídas, tais como um λ para cada variável, ou acoplar métodos de busca unidirecional. No método básico, os parâmetros α e λ são fixos e fornecidos pelo usuário. Contudo, estes parâmetros podem ser determinados usando-se algum método de busca unidirecional.

3.7.2 Método de Nelder-Mead

O método Nelder-Mead Simplex[1] foi desenvolvido também na década de 1960 para otimização não linear. O método não exige que a função objetivo seja diferenciável, mas requer que a função seja contínua. O método trabalha com $n+1$ pontos a cada iteração, eliminando o pior ponto. Um novo ponto é criado com base em regras específicas que serão discutidas a seguir. Esses $n+1$ pontos formam os vértices de um politopo especial denominado simplex. Dessa forma, o comportamento do método pode ser visto como a expansão, contração e movimentação desse simplex no espaço de busca do problema.

No que se segue, usaremos a seguinte notação:

- $b \in \{1, \ldots, n+1\}$ representa o índice do vértice com o melhor valor de função objetivo;
- $w \in \{1, \ldots, n+1\}$ representa o índice do vértice com o pior valor de função objetivo;
- $s \in \{1, \ldots, n+1\}$ representa o índice do vértice com o segundo pior valor de função objetivo;

O centróide da face oposta a $\boldsymbol{x}_w$ é dado por:

$$\hat{\boldsymbol{x}} = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq w}}^{n+1} \boldsymbol{x}_i$$

Este ponto é usado como base para definir as operações do método Nelder-Mead Simplex. Essas operações modificam a forma do simplex adaptando-o às características da função. Cada operação visa gerar o novo vértice do simplex, que substituirá o pior vértice. Essas operações são descritas a seguir e ilustradas na Figura 3.8.

Reflexão: A operação de reflexão tem por objetivo rejeitar a pior solução e avançar o simplex na direção de melhora. Essa operação reflete o pior vértice do simplex sobre a face oposta:

$$\boldsymbol{x}_r = \hat{\boldsymbol{x}} + \alpha \left(\hat{\boldsymbol{x}} - \boldsymbol{x}_w \right), \quad \alpha = 1$$

Expansão: Essa operação expande o simplex na direção de melhora, gerando um ponto além do ponto de reflexão:

$$\boldsymbol{x}_e = \hat{\boldsymbol{x}} + \gamma \left(\hat{\boldsymbol{x}} - \boldsymbol{x}_w \right), \quad \gamma = 2$$

Contração externa: Contrai o simplex na direção de melhora:

$$\boldsymbol{x}_{c+} = \hat{\boldsymbol{x}} + \beta \left(\hat{\boldsymbol{x}} - \boldsymbol{x}_w \right), \quad \beta = 0.5$$

Contração interna: Contrai o simplex internamente:

$$\boldsymbol{x}_{c-} = \hat{\boldsymbol{x}} - \beta \left(\hat{\boldsymbol{x}} - \boldsymbol{x}_w \right), \quad \beta = 0.5$$

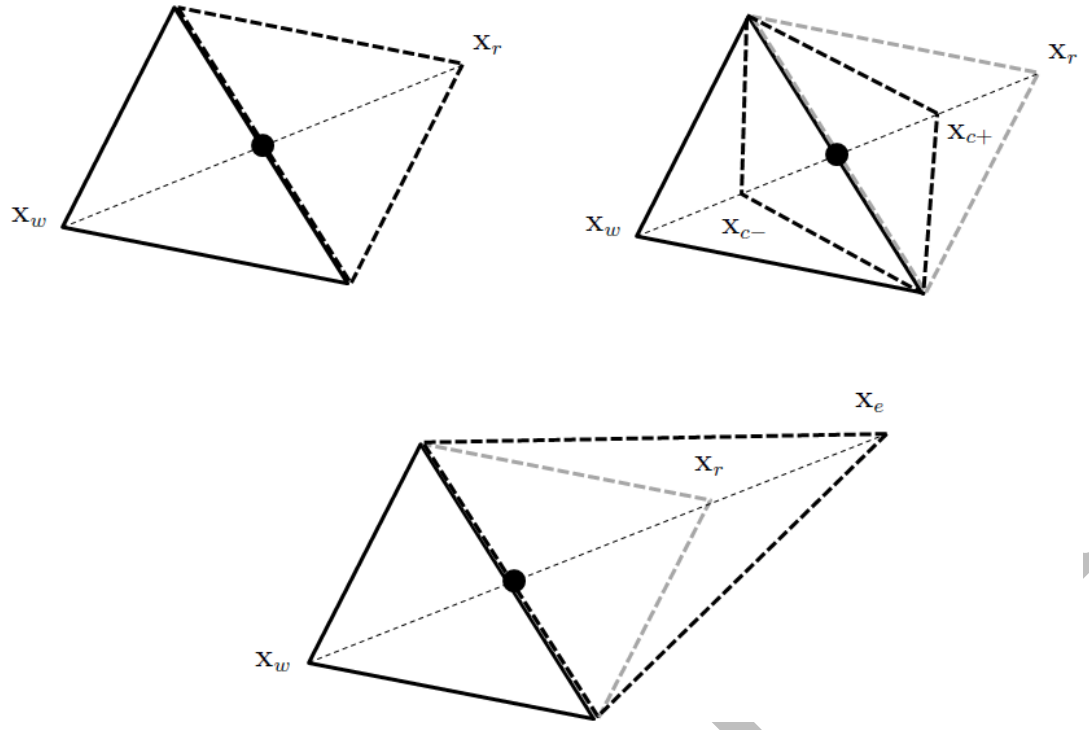

Figura 3.8: Ilustração das operações de reflexão, contração e expansão do simplex no método Nelder-Mead.

Com base nestas operações, temos o Algoritmo 18 apresentado a seguir.

Observe que se o ponto refletido é melhor do que o melhor vértice, então o simplex foi refletido numa direção que minimiza bastante a função objetivo, portanto vale a pena expandir o simplex nessa direção testando o ponto $\boldsymbol{x}_e$. Caso contrário, se $\boldsymbol{x}_r$ não possui um valor tão bom de função objetivo, então é provável que o simplex esteja próximo do ponto de mínimo, por isso são testados os pontos de contração interna e externa, ver linhas 9–15.

Se nenhuma das operações resultou num novo vértice com valor de função objetivo pelo menos melhor do que aquele correspondente ao vértice a ser rejeitado, então deve-se encolher o simplex, pois o ponto de mínimo está em seu interior. A operação de encolhimento do simplex é feita preservando o vértice $\boldsymbol{x}_b$ e aproximando os demais vértices na direção de $\boldsymbol{x}_b$:

$$\boldsymbol{x}_i \leftarrow \boldsymbol{x}_b + \sigma\left(\boldsymbol{x}_i - \boldsymbol{x}_b\right), \quad i = 1, \ldots, n+1, \; i \neq b \tag{3.63}$$

com $\sigma = 0.5$.

Finalmente, os critérios de parada são em geral baseados no volume do simplex. Por exemplo, pode-se monitorar os tamanhos das arestas $\|\boldsymbol{x}_i - \boldsymbol{x}_b\|$ e caso estas estejam abaixo de um valor de tolerância, considera-se que o método convergiu.

[1]Não confundir com o método Simplex desenvolvido para otimização linear.

Algorithm 18: Método Nelder-Mead Simplex

$k \leftarrow 0$;
while *(critério de parada não for satisfeito)* **do**
  Calcule $\boldsymbol{x}_r = \hat{\boldsymbol{x}} + \alpha\,(\hat{\boldsymbol{x}} - \boldsymbol{x}_w)$;
  if $f(\boldsymbol{x}_r) < f(\boldsymbol{x}_b)$ **then** Expansão
    calcule e avalie $\boldsymbol{x}_e$;
    if $f(\boldsymbol{x}_e) < f(\boldsymbol{x}_r)$ **then** $\boldsymbol{x}_{new} = \boldsymbol{x}_e$;
    else $\boldsymbol{x}_{new} = \boldsymbol{x}_r$;
  else if $f(\boldsymbol{x}_b) < f(\boldsymbol{x}_r) < f(\boldsymbol{x}_s)$ **then** $\boldsymbol{x}_{new} = \boldsymbol{x}_r$;
  else if $f(\boldsymbol{x}_s) < f(\boldsymbol{x}_r) < f(\boldsymbol{x}_w)$ **then** Contração externa
    calcule e avalie $\boldsymbol{x}_{c+}$;
    if $f(\boldsymbol{x}_{c+}) \leq f(\boldsymbol{x}_w)$ **then** $\boldsymbol{x}_{new} = \boldsymbol{x}_{c+}$;
  else if $f(\boldsymbol{x}_r) \geq f(\boldsymbol{x}_w)$ **then** Contração interna
    calcule e avalie $\boldsymbol{x}_{c-}$;
    if $f(\boldsymbol{x}_{c-}) \leq f(\boldsymbol{x}_w)$ **then** $\boldsymbol{x}_{new} = \boldsymbol{x}_{c-}$;
  else
    Encolhe o simplex
  end
end

3.8 Exercícios

1. Dado $f : \mathbb{R}^2 \mapsto \mathbb{R}$, $f(\boldsymbol{x}) = x_1^2 + x_2$ e $\boldsymbol{x}_0 = (2, 2)^T$, efetuar uma busca direcional pelo Método da Seção Áurea na direção $-\nabla f(\cdot)$ a partir de $\boldsymbol{x}_0$. Considere $\delta = 0.0001$ para a estimação do gradiente; $s = 0.1$ para determinar o intervalo $[a, b]$; e $\epsilon = 0.1$ como critério de parada do Algoritmo da Seção Áurea [4].

2. Considere a função f definida por $f(\boldsymbol{x}) = (x_1 + x_2^3)^2 + 2(x_1 - x_2 - 4)^4$. Dado um ponto $\boldsymbol{x}_0$ e um vetor direção não nulo $\boldsymbol{d}$, seja $\theta(\alpha) = f(\boldsymbol{x}_0 + \alpha \boldsymbol{d})$. Pede-se:

   (i) Obter a expressão explícita para $\theta(\alpha)$.

   (ii) Calcular o valor de α^* que resolve o problema de minimização de $\theta(\alpha)$, sujeito a $\alpha \in \mathbb{R}$, para $\boldsymbol{x}_0 = (5, 4)^T$ e $\boldsymbol{d} = (-2, 1)^T$.

3. Seja o problema de minimizar $f(\boldsymbol{x}) = x_1^3 + x_1 x_2 - x_1^2 x_2^2$ usando o Método de Newton a partir do ponto $\boldsymbol{x}_0 = (1, 1)^T$. Um programa computacional cuidadosamente implementado para executar este método não foi bem sucedido. Discutir as prováveis razões para o não sucesso [4].

4. Dada a função $f(\boldsymbol{x}) = x_1^2 + 4x_2^2 - 4x_1 - 8x_2$, pergunta-se [4]:

   (i) Qual a direção de máximo declive no ponto $\boldsymbol{x}_0 = (1, 1)^T$?

   (ii) Qual a direção inicial de busca $\boldsymbol{d}$ determinada pelo Método de Newton a partir de $\boldsymbol{x}_0 = (1, 1)^T$?

   (iii) Qual o comprimento de α, para o item (ii), tal que $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \alpha \boldsymbol{d}$?

   (iv) Usando o Método de Newton, quantos passos são necessários para minimizar $f(\boldsymbol{x})$ partindo de $\boldsymbol{x}_0 = (1, 1)^T$? Por que?

5. Seja o problema definido por: minimize $f(\boldsymbol{x}) = x_1^4 - 2x_1^2 x_2 + x_1^2 + x_2^2 - 2x_1 + 4$, com $-4 \leq x_1,\ x_2 \leq 4$. A partir do ponto $\boldsymbol{x}_0 = (3,\ 4)^T$, encontrar o mínimo de $f(\boldsymbol{x})$ usando: (i) o Método do Gradiente, (ii) o Método DFP, e (iii) o Método BFGS. O mesmo critério de parada deve ser empregado para todos os algoritmos. Plotar o gráfico de $f(\boldsymbol{x})$ versus o número de iterações para os três métodos e comparar o processo de convergência. Os métodos convergem para a mesma solução? Por que?

6. Considere o seguinte problema:

$$\text{minimize} \ \sum_{i=2}^{n} [100(x_i - x_{i-1}^2)^2 + (1 - x_{i-1})^2]$$

   Assumindo os valores de dimensões $n = 5$, 10, e 50, e ponto inicial $\mathbf{x}_0 = (-1.2, 1.0, -1.2, 1.0, \ldots)$, encontrar o mínimo de $f(\mathbf{x})$ usando: (i) o Método do Gradiente, (ii) o Método DFP, e (iii) o Método BFGS. O mesmo critério de parada deve ser empregado para todos os algoritmos. Plotar o gráfico de $f(\boldsymbol{x})$ versus o número de iterações para os três métodos e comparar o processo de convergência. Os métodos convergem para a mesma solução? Por que?

7. Considere o sistema de equações simultâneas

$$h_i(\boldsymbol{x}) = 0 \quad \text{para } i = 1, \ldots, l$$

(i) Mostre como resolver esse sistema de equações usando otimização irrestrita. (Dica: considere o problema de minimizar $\sum_{i=1}^{l} |h_i(\boldsymbol{x})|^p$, onde p é um inteiro positivo.)

(ii) Usando essa estratégia, resolva o seguinte sistema:

$$2(x_1 - 2)^4 + (2x_1 - x_2)^2 - 4 = 0$$

$$x_1^2 - 2x_2 + 1 = 0$$

Referências Bibliográficas

[1] D. G. Luenberger. *Linear and Nonlinear Programming*. Addison-Wesley, 2 edition, 1989.

[2] P. Venkataraman. *Applied Optimization with Matlab Programming*. John Wiley, 1 edition, 2002.

[3] M.S. Bazaraa, H.D. Sherali, and C.M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley, 3 edition, 2006.

[4] G.R. Mateus e H.P.L. Luna. *Programação Não-Linear*. V Escola de Computação, 1 edition, 1986.