

Otimização ELE037

Métodos Numéricos para Otimização Irrestrita

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

Universidade Federal de Minas Gerais
Departamento de Engenharia Elétrica

Sumário

- 1 Introdução
- 2 Estrutura Básica
- 3 Método de Busca em Direções Aleatórias
- 4 Método do Gradiente
- 5 Otimização Unidimensional
- 6 Aproximações Quadráticas
- 7 Gradientes Conjugados
- 8 Métodos sem Derivadas

Nesta unidade são abordados Métodos de Direções de Busca.

Estes métodos 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.

Seja o problema de otimização irrestrito:

$$\mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x}) \quad (1)$$

sendo que $\mathbf{x} \in \mathbb{R}^n$ e $f(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}^1$.

Dado um ponto inicial $\mathbf{x}_0 \neq \mathbf{x}^*$, obtém-se uma sequência $\mathbf{x}_k$ tal que $\mathbf{x}_k \rightarrow \mathbf{x}^*$ a partir do algoritmo de otimização.

Estrutura Básica

A família dos algoritmos de direção de busca possui a estrutura:

Algorithm 1: Algoritmo de Direção de Busca

```
1  $k \leftarrow 0$ ;  
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{d}_k \leftarrow \mathbf{h}(\mathbf{x}_1, \dots, \mathbf{x}_k, f(\mathbf{x}_1), \dots, f(\mathbf{x}_k))$ ;  
4    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ ;  
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ ;  
6    $k \leftarrow k + 1$ ;  
7 end
```

Nessa estrutura, $\mathbf{h}(\cdot, \dots, \cdot)$ é uma função que em geral será recursiva;

Não dependerá explicitamente dos pontos anteriores, mas irá armazenar sua influência em variáveis intermediárias.

Estrutura Básica

Um algoritmo irá diferir de outro essencialmente pela maneira como é calculada a direção de busca $\mathbf{d}_k$.

Método do Gradiente:

$$\mathbf{d}_k = -\nabla f(\mathbf{x}_k) \quad (2)$$

Método de Newton:

$$\mathbf{d}_k = -H_k^{-1} \nabla f(\mathbf{x}_k) \quad (3)$$

Métodos quase-Newton:

$$\mathbf{d}_k = -\hat{H}_k^{-1} \nabla f(\mathbf{x}_k) \quad (4)$$

Elementos para a construção de Algoritmos de Direções de Busca:

- (i) **Método de cálculo de direções de busca**, possivelmente envolvendo o cálculo de estimativas para o gradiente e para a Hessiana da função objetivo;
- (ii) **Método de minimização de funções de uma única variável**;
- (iii) **Critério de decisão** que permita afirmar que o algoritmo convergiu para uma solução satisfatória.

Método de Busca em Direções Aleatórias

Introdução

Simple, efetivo, porém ineficiente.

Algorithm 2: Algoritmo de Busca em Direções Aleatórias

```
1  $k \leftarrow 0$ ;  
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{d}_k \leftarrow \text{randn}(n, 1)$ ;  
4    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ ;  
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ ;  
6    $k \leftarrow k + 1$ ;  
7 end
```

Método de Busca em Direções Aleatórias

Introdução

- A função $rand(n, 1)$ fornece um vetor $n \times 1$ de componentes aleatórios segundo uma distribuição Gaussiana, com média 0 e variância 1.
- A convergência desse algoritmo para o ponto de mínimo de uma **função unimodal** pode ser demonstrada se $f(\mathbf{x}_k) \leq f(\mathbf{x}_{k-1})$.
- O algoritmo produz uma sequência $[f(\mathbf{x}_k)]$ que se aproxima de forma monotônica do valor mínimo da função, $f(\mathbf{x}^*)$.

Método de Busca em Direções Aleatórias

Problema Exemplo

Consideremos o problema:

$$\begin{aligned} \mathbf{x}^* = \arg \min_{\mathbf{x}} f(\mathbf{x}) &= 2x_1^2 + x_2^2 + 2x_1x_2 + x_1 - 2x_2 + 3 \\ \text{sujeito a: } &\left\{ \begin{array}{l} -6 \leq x_1 \leq 6; \\ -6 \leq x_2 \leq 6 \end{array} \right. \end{aligned} \quad (5)$$

Método de Busca em Direções Aleatórias

Problema Exemplo

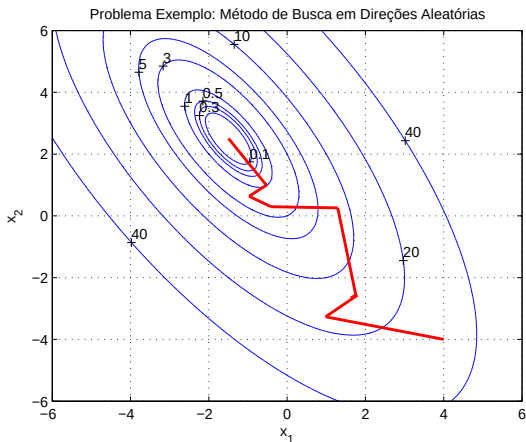


Figura: Solução usando o Método de Busca em Direções Aleatórias.

Método de Busca em Direções Aleatórias

Problema Exemplo

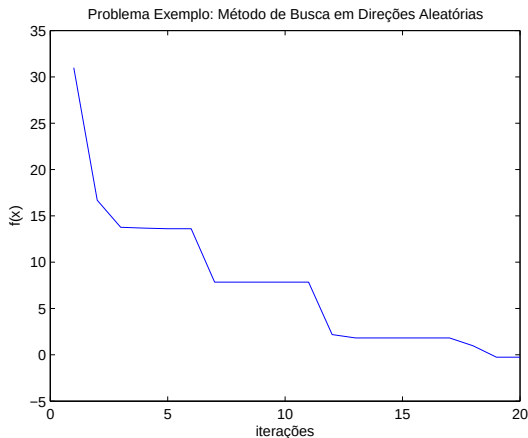


Figura: Variação da função objetivo versus o número de iterações.

Método do Gradiente

Introdução

Uma escolha razoável para uma direção de busca é $\mathbf{d}_k = -\nabla f(\mathbf{x}_k)$.

Localmente, essa é a direção na qual a função $f(\cdot)$ decresce mais rapidamente.

A função $f(\mathbf{x})$ deve ser diferenciável.

Método do Gradiente

Algoritmo

Algorithm 3: Algoritmo do Método do Gradiente

```
1  $k \leftarrow 0$ ;  
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{g}_k \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_k)$ ;  
4    $\mathbf{d}_k \leftarrow -\mathbf{g}_k$ ;  
5    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ ;  
6    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ ;  
7    $k \leftarrow k + 1$ ;  
8 end
```

Método do Gradiente

Considerações práticas

O algoritmo nos indica que temos que tratar quatro questões:

- Cálculo numérico do gradiente
- Critérios de parada e/ou convergência
- Otimização unidimensional

Cálculo Numérico do Gradiente

No caso geral, onde $f(\cdot)$ é do tipo caixa-preta, **como estimar o gradiente** em um ponto qualquer $\mathbf{x}$ do espaço de busca?

Aproximação baseada em diferenças finitas:

Algorithm 4: Algoritmo do Cálculo do Gradiente

```
1  $k \leftarrow 0$ ;  
2 for ( $i \leftarrow 1$  until  $n$ ) do  
3    $g_i \leftarrow [f(\mathbf{x} + \delta \mathbf{e}_i) - f(\mathbf{x})] / \delta$ ;  
4 end  
5  $\mathbf{g} \leftarrow [g_1, \dots, g_n]^T$ ;
```

Cálculo Numérico do Gradiente

Exemplo: Determinar analítica e numericamente o gradiente de $f(\cdot)$ no ponto $\mathbf{x}_0 = [0 \ 0]^T$.

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

Critérios de Parada

- Estabilização do valor da função-objetivo:

Algorithm 5: Critério de Parada: Função Objetivo

```
1  $\Delta_f \leftarrow f_{max} - f_{min};$   
2  $f_{5+} \leftarrow \max \{f(\mathbf{x}_k), f(\mathbf{x}_{k-1}), f(\mathbf{x}_{k-2}), f(\mathbf{x}_{k-3}), f(\mathbf{x}_{k-4}), f(\mathbf{x}_{k-5})\};$   
3  $f_{5-} \leftarrow \min \{f(\mathbf{x}_k), f(\mathbf{x}_{k-1}), f(\mathbf{x}_{k-2}), f(\mathbf{x}_{k-3}), f(\mathbf{x}_{k-4}), f(\mathbf{x}_{k-5})\};$   
4  $\delta_f \leftarrow f_{5+} - f_{5-};$   
5 if ( $\delta_f < 0.0001\Delta_f$ ) then  
6   |  $parada \leftarrow true;$   
7 else  
8   |  $parada \leftarrow false;$   
9 end
```

Critérios de Parada

- Estabilização do vetor de variáveis de otimização:

Algorithm 6: Critério de Parada: Vetor de Variáveis

```
1  $\Delta_{\mathbf{x}} \leftarrow \mathbf{x}_{max} - \mathbf{x}_{min};$   
2  $\mathbf{x}_{5+} \leftarrow \max \{ \mathbf{x}_k, \mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \mathbf{x}_{k-3}, \mathbf{x}_{k-4}, \mathbf{x}_{k-5} \};$   
3  $\mathbf{x}_{5-} \leftarrow \min \{ \mathbf{x}_k, \mathbf{x}_{k-1}, \mathbf{x}_{k-2}, \mathbf{x}_{k-3}, \mathbf{x}_{k-4}, \mathbf{x}_{k-5} \};$   
4  $\delta_{\mathbf{x}} \leftarrow \mathbf{x}_{5+} - \mathbf{x}_{5-};$   
5 if ( $\delta_{\mathbf{x}} < 0.0001\Delta_{\mathbf{x}}$ ) then  
6   |  $parada \leftarrow true;$   
7 else  
8   |  $parada \leftarrow false;$   
9 end
```

- Anulação do Vetor Gradiente:

Algorithm 7: Critério de Parada: Vetor Gradiente

```
1  $M_g = \max \{ \|\mathbf{g}(\mathbf{x}_k)\|, \|\mathbf{g}(\mathbf{x}_{k-1})\|, \|\mathbf{g}(\mathbf{x}_{k-2})\| \};$   
2 if ( $M_g < 0.0001M_{max}$ ) then  
3   |  $parada \leftarrow true;$   
4 else  
5   |  $parada \leftarrow false;$   
6 end
```

Outros:

- Tempo de execução; $\|\mathbf{g}(\mathbf{x}_k)\| \leq \epsilon$; número máximo de iterações, etc.

Caso o Algoritmo do Gradiente seja iniciado em um ponto $\mathbf{x}_0$ não situado na bacia de atração do mínimo global $\mathbf{x}^*$, 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 $\mathbf{x}_0$.
Tem-se uma convergência monotônica.
- 2 Caso o ponto inicial não esteja localizado em nenhuma bacia de atração, o Algoritmo do Gradiente não converge.

Problema de otimização unidimensional

Introdução

Definição

$$\alpha^* = \arg \min_{\alpha} \theta(\alpha) \in \mathbb{R}, \alpha \in [0, +\infty]$$

$$\theta(\alpha) = f(\mathbf{x}_k + \alpha \mathbf{d}), \quad \mathbf{x}_k \text{ e } \mathbf{d} \in \mathbb{R}^n$$

Exemplo

Determinar $\mathbf{x}_1$ que minimiza $f(\mathbf{x}) = 2\mathbf{x}_1^2 + \mathbf{x}_2^2$ partindo de $\mathbf{x}_0 = [1 \ 1]$ na direção $\mathbf{d} = -\nabla f(\mathbf{x}_0)$.

Problema de otimização unidimensional

Métodos de eliminação

- Busca irrestrita;
- Busca exaustiva;
- Busca dicotômica;
- Método de Fibonacci;
- Método da Seção Áurea.

Exigem funções unimodais, porém não exigem diferenciabilidade.

Problema de otimização unidimensional

Busca irrestrita

- Não exige que o espaço de busca seja conhecido.
- Versão elementar:
 - Move-se numa direção minimizante $\mathbf{d}$ usando passo fixo s ;
 - Determina-se uma sequência de pontos $u_{k+1} = u_k + s$;
 - O passo usado deve ser pequeno em relação à precisão desejada;
 - Assume-se unimodalidade da função ao longo de $\mathbf{d}$;
 - Limitação: pode exigir elevado número de avaliações de $\theta(\cdot)$ se u_0 estiver distante de u^* e s for pequeno.
- Versão melhorada:
 - Usar $s_{k+1} = \lambda s_k$, $\lambda > 1$, até “cercar” o intervalo que contém u^* ;
 - Feito isto, reduzir o intervalo até uma precisão desejada.

Problema de otimização unidimensional

Busca exaustiva

- Assume que o intervalo que contém u^* seja conhecido;
- Denota a e b os pontos que cercam o intervalo;
- Avalia $\theta(\cdot)$ num número pré-estabelecido de pontos igualmente espaçados em (a, b) ;
- Considerando unimodalidade de $\theta(\cdot)$, toma-se o novo menor intervalo (a, b) que contém u^* ;
- O intervalo é reduzido até uma precisão desejada;
- Limitação: caracteriza uma busca simultânea cujos testes subsequentes independem dos resultados já obtidos.

Problema de otimização unidimensional

Busca dicotômica

- Representa uma busca sequencial onde os testes realizados influenciam na escolha dos testes subsequentes.
- Assume que o intervalo (a, b) que cerca u^* seja conhecido.
- Escolhe dois pontos próximos ao centro do intervalo

$$u = \frac{L_0}{2} - \frac{\delta}{2}, \quad v = \frac{L_0}{2} + \frac{\delta}{2}, \quad \delta > 0$$

onde L_0 é o tamanho do intervalo inicial.

- Baseado na avaliação de $\theta(\cdot)$ nestes dois pontos, exclui-se quase metade do intervalo.
- O processo se repete até atingir a precisão desejada.

Problema de otimização unidimensional

Busca da bisseção

- Exclui metade do intervalo de busca a cada iteração.
- Especifica três pontos, u , c e v , igualmente espaçados no intervalo inicial (a, b) ;
- Assumindo unimodalidade, tem-se:
 - Se $\theta_u < \theta_c < \theta_v$, deleta (c, b) , e faz-se $b = c$ e $c = u$;
 - Se $\theta_u > \theta_c > \theta_v$, deleta (a, c) , e faz-se $a = c$ e $c = v$;
 - Se $\theta_u > \theta_c$ e $\theta_v > \theta_c$, deleta (a, u) e (v, b) , e faz-se $a = u$ e $b = v$.
- Especifica novos pontos u e v , e continua o processo até $L \leq \epsilon$.

Problema de otimização unidimensional

Método de Fibonacci

- Assume unimodalidade de $\theta(\cdot)$ e o conhecimento do intervalo $[a, b]$ que contém o ótimo.
- Define dois pontos $u, v \in [a, b]$:
 - Se $\theta(u) < \theta(v)$, mínimo está em $[a, v]$;
 - Se $\theta(u) > \theta(v)$, mínimo está em $[u, b]$.
- Apenas um novo ponto precisará ser especificado nas iterações subsequentes.
- O número de avaliações de θ (ou a precisão desejada) deve ser especificado.

Problema de otimização unidimensional

Método de Fibonacci

- Os pontos u e v são definidos usando a série de Fibonacci:

$$F_0 = F_1 = 1, \quad F_i = F_{i-1} + F_{i-2}, \quad i = 2, 3, \dots, n$$

- Dado o intervalo inicial $[a_0, b_0]$, tem-se:

$$u_0 = b_0 - (F_{n-1}/F_n)(b_0 - a_0)$$

$$v_0 = a_0 + (F_{n-1}/F_n)(b_0 - a_0)$$

- Para uma iteração i qualquer ($i = 0, \dots, n-2$), tem-se:

$$u_i = b_i - (F_{n-i-1}/F_{n-i})(b_i - a_i)$$

$$v_i = a_i + (F_{n-i-1}/F_{n-i})(b_i - a_i)$$

- O comprimento do intervalo após k iterações é:

$$L_k = (F_{n-k}/F_n)(b_0 - a_0)$$

Problema de otimização unidimensional

Método da seção áurea

- Similar ao método de Fibonacci, porém não exige que o número de iterações seja especificado.
- O processo termina ao atingir a precisão desejada.
- Para uma iteração i qualquer ($i = 0, 1, \dots$), tem-se:

$$u_i = b_i - F(b_i - a_i) , \quad v_i = a_i + F(b_i - a_i)$$

onde $F = (\sqrt{5} - 1)/2 = 0.618$.

- O comprimento do intervalo após k iterações é:

$$L_k = (0.618)^k (b_0 - a_0)$$

- O tamanho do intervalo é multiplicado por 0.618 a cada iteração.
- Os métodos de Fibonacci e seção áurea são os mais eficientes, porém o segundo é mais prático.

Problema de otimização unidimensional

Algoritmo para determinação do intervalo

Algorithm 8: Algoritmo para determinação do intervalo

```
1  $a \leftarrow 0$  e  $b \leftarrow s$ ;  
2  $\theta(a) = \theta(0) = f(\mathbf{x}_k)$  e  $\theta(b)$ ;  
3  $nfe \leftarrow 2$ ;  
4 while  $\theta(b) < \theta(a)$  do  
5 |    $a \leftarrow b$  e  $\theta(a) \leftarrow \theta(b)$ ;  
6 |    $b \leftarrow 2b$  e  $\theta(b)$ ;  
7 |    $nfe \leftarrow nfe + 1$ ;  
8 end  
9 if  $nfe \leq 3$  then  
10 |   $a \leftarrow 0$ ;  
11 else  
12 |   $a \leftarrow a/2$ ;  
13 end  
14 return  $a, b$ ;
```

Problema de otimização unidimensional

Algoritmo da Seção Áurea

Algorithm 9: Algoritmo da Seção Áurea

```
1  $x_a \leftarrow b - 0.618(b - a)$  e  $x_b \leftarrow a + 0.618(b - a)$ ;  
2  $\theta_a \leftarrow \theta(x_a)$  e  $\theta_b \leftarrow \theta(x_b)$ ;  
3 while  $(b - a > \epsilon)$  do  
4   if  $(\theta_a > \theta_b)$  then  
5      $a \leftarrow x_a$ ;  
6      $x_a \leftarrow x_b$  e  $x_b \leftarrow a + 0.618(b - a)$ ;  
7      $\theta_a \leftarrow \theta_b$  e  $\theta_b \leftarrow \theta(x_b)$ ;  
8   else  
9      $b \leftarrow x_b$ ;  
10     $x_b \leftarrow x_a$  e  $x_a \leftarrow b - 0.618(b - a)$ ;  
11     $\theta_b \leftarrow \theta_a$  e  $\theta_a \leftarrow \theta(x_a)$ ;  
12  end  
13 end  
14  $\alpha \leftarrow (a + b)/2$ ;
```

Problema de otimização unidimensional

Métodos de interpolação

- Método de interpolação quadrática;
- Métodos de cálculo de raízes:
 - Método de Newton;
 - Método da Secante.

Exigem funções “bem comportadas” (convexas ou continuamente diferenciáveis de 1a ou 2a ordem).

Problema de otimização unidimensional

Método de interpolação quadrática

- A função $\theta(\alpha)$ é aproximada por uma quadrática $q(\alpha)$ e seu mínimo $\bar{\alpha}^*$ é determinado.
- Sendo $q(\alpha) = a + b\alpha + c\alpha^2$, a condição de 1a ordem fornece

$$\frac{dq}{d\alpha} = b + 2c\alpha = 0, \quad \text{ou seja,} \quad \bar{\alpha}^* = -\frac{b}{2c}$$

- Pela condição de 2a ordem $q''(\bar{\alpha}^*) > 0$, i.e., $c > 0$.
- Basta avaliar $q(\cdot)$ em três pontos distintos $A < B < C$, que satisfaçam $c > 0$, e calcular $\bar{\alpha}^*$. Para $c > 0$, $\theta_B < \max\{\theta_A, \theta_C\}$.
- Enquanto $\bar{\alpha}^*$ não for suficientemente próximo de α^* , estima-se uma nova quadrática:

$$\left| \frac{q(\bar{\alpha}^*) - \theta(\bar{\alpha}^*)}{\theta(\bar{\alpha}^*)} \right| \leq \epsilon$$

Problema de otimização unidimensional

Método de Newton

- Considera uma aproximação quadrática usando séries de Taylor:

$$\theta(\alpha) = \theta(\alpha_k) + \theta'(\alpha_k)(\alpha - \alpha_k) + \frac{1}{2}\theta''(\alpha_k)(\alpha - \alpha_k)^2$$

- Baseando-se na condição de 1a ordem:

$$\theta'(\alpha) = \theta'(\alpha_k) + \theta''(\alpha_k)(\alpha - \alpha_k) = 0$$

$$\alpha_{k+1} = \alpha_k - \frac{\theta'(\alpha_k)}{\theta''(\alpha_k)}$$

- A convergência do método pode ser verificada usando:

$$|\theta'(\alpha_{k+1})| \leq \epsilon$$

Problema de otimização unidimensional

Método de Newton

- As derivadas são aproximadas usando diferenças finitas:

$$\theta'(\alpha_k) = \frac{\theta(\alpha_k + \Delta\alpha) - \theta(\alpha_k - \Delta\alpha)}{2\Delta\alpha}$$

$$\theta''(\alpha_k) = \frac{\theta(\alpha_k + \Delta\alpha) - 2\theta(\alpha_k) + \theta(\alpha_k - \Delta\alpha)}{\Delta\alpha^2}$$

em que $\Delta\alpha$ representa uma pequena variação.

Problema de otimização unidimensional

Método da Secante

- Utiliza uma aproximação similar ao método de Newton:

$$\theta'(\alpha) = \theta'(\alpha_k) + s(\alpha - \alpha_k) = 0$$

em que s representa a inclinação entre os pontos $(A, \theta'(A))$ e $(B, \theta'(B))$:

$$s = \frac{\theta'(B) - \theta'(A)}{B - A}$$

em que A e B são estimativas de α^* .

- O processo iterativo utiliza

$$\alpha_{k+1} = \alpha_k - \frac{\theta'(\alpha_k)}{s}$$

- A convergência do método pode ser verificada usando:

$$|\theta'(\alpha_{k+1})| \leq \epsilon$$

Problema de otimização unidimensional

Considerações práticas

- Métodos de interpolação:
 - São mais baratos, porém dependem da estimação de derivadas;
 - Podem falhar caso a função não seja “bem comportada”.
- Métodos de eliminação:
 - São mais usuais e práticos;
 - Porém, precisam determinar o intervalo $[a, b]$ que cerca α^* :
 - Comumente emprega-se Busca Irrestrita.

Problema de otimização unidimensional

Exemplo

Uma busca é realizada sobre $f(\cdot)$ partindo de $\mathbf{x}_0 = [-1 \ 1]^T$ na direção $-\nabla f(\mathbf{x}_0)$. Determinar analiticamente a função unidimensional $\theta(\alpha)$ para essa situação. Qual o valor de α^* que minimiza $f(\cdot)$ nessa direção? Qual o novo vetor $\mathbf{x}$ obtido?

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

Resp.

$$\theta(\alpha) = 10\alpha^2 - 5\alpha + 1; \quad \alpha^* = 1/4; \quad \mathbf{x} = [-0.75 \ 1.5]^T.$$

Aproximações Quadráticas

Introdução

Seja uma aproximação quadrática de $f(\mathbf{x})$, ao redor de $\mathbf{x}_0$, dada por:

$$f(\mathbf{x}) \approx \mathbf{c}_0 + \mathbf{c}_1 \cdot (\mathbf{x} - \mathbf{x}_0) + (\mathbf{x} - \mathbf{x}_0)^T C_2 (\mathbf{x} - \mathbf{x}_0) \quad (6)$$

sendo $\mathbf{c}_0 \in \mathbb{R}^n$, $\mathbf{c}_1 \in \mathbb{R}^n$ e $C_2 \in \mathbb{R}^{n \times n}$.

Escrevendo $f(\mathbf{x})$ em termos de uma série de Taylor:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T H(\mathbf{x}_0) (\mathbf{x} - \mathbf{x}_0) + O(3) \quad (7)$$

Aproximações Quadráticas

Introdução

O gradiente da função $f(\mathbf{x})$ dada por (7) é:

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_0) + H(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \quad (8)$$

Das condições de 1a ordem, $\nabla f(\mathbf{x}^*) = 0$:

$$\nabla f(\mathbf{x}^*) = \nabla f(\mathbf{x}_0) + H(\mathbf{x}_0)(\mathbf{x}^* - \mathbf{x}_0) = 0 \quad (9)$$

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

$$\mathbf{x}^* = \mathbf{x}_0 - H(\mathbf{x}_0)^{-1} \nabla f(\mathbf{x}_0) \quad (10)$$

Aproximações Quadráticas

Introdução

- **Se a função for exatamente quadrática**, basta se conhecer o gradiente e a Hessiana em um ponto qualquer $\mathbf{x}_0$ para se determinar, em **uma única iteração**, o ponto de mínimo $\mathbf{x}^*$, através da equação (10).
- **Se a função for aproximadamente quadrática num certo domínio**, a equação (10) 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.

Aproximações Quadráticas

Método de Newton

O Método de Newton envolve a aplicação iterativa de (10):

Algorithm 10: Algoritmo do Método de Newton

```
1  $k \leftarrow 0$ ;  
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{g}_k \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_k)$ ;  
4    $H_k \leftarrow \text{Hessiana}(f(\cdot), \mathbf{x}_k)$ ;  
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k - H_k^{-1} \mathbf{g}_k$ ;  
6    $k \leftarrow k + 1$ ;  
7 end
```

Caso a função $f(\cdot)$ não seja quadrática:

- O Método de Newton garante convergência?
- A sequência de soluções obtidas produz valores monotônicos de $f(\cdot)$?

Aproximações Quadráticas

Método de Newton Modificado

- Para garantir que o algoritmo produza a diminuição monotônica do valor da função objetivo, introduz-se a **execução de uma minimização unidimensional** em cada direção:

Algorithm 11: Algoritmo do Método de Newton Modificado

```
1  $k \leftarrow 0$ ;  
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{g}_k \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_k)$ ;  
4    $H_k \leftarrow \text{Hessiana}(f(\cdot), \mathbf{x}_k)$ ;  
5    $\mathbf{d}_k \leftarrow -H_k^{-1} \mathbf{g}_k$ ;  
6    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ ;  
7    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ ;  
8    $k \leftarrow k + 1$ ;  
9 end
```

Aproximações Quadráticas

Convergência do Método de Newton Modificado

Caso $\mathbf{x}_0$ não pertença à bacia de atração do mínimo global $\mathbf{x}^*$, podem ocorrer três situações:

- (i) O algoritmo **converge para o mínimo local estrito** associado à bacia de atração em que estiver localizado seu ponto inicial $\mathbf{x}_0$.
- (ii) Caso o ponto inicial esteja localizado em uma bacia de atração de um mínimo local não estrito, **o algoritmo 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 não converge**, podendo ainda ficar indefinido.

Aproximações Quadráticas

Determinação Numérica da Hessiana

- Sendo $\mathbf{g}(\mathbf{x})$ o gradiente da função objetivo, avaliado numericamente por meio de diferenças finitas, tem-se:

Algorithm 12: Algoritmo do Cálculo da Hessiana por Diferenças Finitas

```
1  $k \leftarrow 0$ ;  
2 for ( $i \leftarrow 1$  until  $n$ ) do  
3    $F_i \leftarrow [\mathbf{g}(\mathbf{x} + \delta \mathbf{e}_i) - \mathbf{g}(\mathbf{x})] / \delta$ ;  
4 end  
5  $F \leftarrow [F_1 \ \cdots \ F_n]$ ;
```

- Cada estimação do gradiente envolve $n + 1$ avaliações de $f(\cdot)$;
- Para a aproximação da Hessiana tem-se $(n + 1)^2$.

Aproximações Quadráticas

Construção da Hessiana

- Examinemos novamente a equação usada no método de Newton:

$$\nabla f(\mathbf{x}) = \nabla f(\mathbf{x}_0) + H(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \quad (11)$$

- Pode ser usada para construir um método para estimar a própria Hessiana da função.
- Reescrevendo a equação para dois pontos $\mathbf{x}_1$ e $\mathbf{x}_2$, e supondo que a **Hessiana seja constante** em todo o espaço:

$$H(\mathbf{x}_1 - \mathbf{x}_2) = \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2) \quad (12)$$

Aproximações Quadráticas

Construção da Hessiana

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

$$\begin{aligned}H(\mathbf{x}_1 - \mathbf{x}_2) &= \nabla f(\mathbf{x}_1) - \nabla f(\mathbf{x}_2) \\H(\mathbf{x}_2 - \mathbf{x}_3) &= \nabla f(\mathbf{x}_2) - \nabla f(\mathbf{x}_3) \\&\vdots \\H(\mathbf{x}_{n-1} - \mathbf{x}_n) &= \nabla f(\mathbf{x}_{n-1}) - \nabla f(\mathbf{x}_n) \\H(\mathbf{x}_n - \mathbf{x}_{n+1}) &= \nabla f(\mathbf{x}_n) - \nabla f(\mathbf{x}_{n+1})\end{aligned}\tag{13}$$

Aproximações Quadráticas

Construção da Hessiana

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

$$\mathbf{v}_i = \mathbf{x}_i - \mathbf{x}_{i+1} \quad (14)$$

$$\mathbf{r}_i = \nabla f(\mathbf{x}_i) - \nabla f(\mathbf{x}_{i+1})$$

tem-se que:

$$H[\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_n] = [\mathbf{r}_1 \ \mathbf{r}_2 \ \cdots \ \mathbf{r}_n] \quad (15)$$

Aproximações Quadráticas

Construção da Hessiana

- Definindo $V = [\mathbf{v}_1 \ \mathbf{v}_2 \ \cdots \ \mathbf{v}_n]$ e $R = [\mathbf{r}_1 \ \mathbf{r}_2 \ \cdots \ \mathbf{r}_n]$, obtém-se:

$$HV = R \quad (16)$$

Note que é possível escolher vetores $\mathbf{v}_i$ de tal forma que V seja inversível, o que permite fazer:

$$H = RV^{-1} \quad (17)$$

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

Aproximações Quadráticas

Construção da Hessiana

- A equação $HV = R$ é uma generalização do cálculo da Hessiana por diferenças finitas.
- De fato, fazendo-se $V = \delta I$ tem-se que $H = R/\delta$.
- Diversos métodos de otimização baseiam-se na equação $H = RV^{-1}$.
- Estes algoritmos diferem entre si em função da escolha dos pontos, o que implica na variação da escolha de V .

Aproximações Quadráticas

Correção de Posto 1

- Há certa arbitrariedade na escolha dos vetores $\mathbf{v}_i$.
- A única condição necessária é de que sejam n **vetores linearmente independentes**.
- A ideia é fazer a **construção recursiva da estimativa da Hessiana**, ou de sua inversa, durante o processo de otimização.
- Isso é particularmente útil na otimização de funções não-quadráticas, em que a Hessiana não é constante.

Aproximações Quadráticas

Correção de Posto 1

- Seja $\tilde{H}_k = H_k^{-1}$.
- A ideia é **construir recursivamente uma sequência de estimativas** $[\tilde{H}_k]$.
- Esta construção basea-se nos valores de $\mathbf{x}_k$ e $\nabla f(\mathbf{x}_k)$ em novos pontos.
- A recursão proposta é da forma:

$$\tilde{H}_{k+1} = \tilde{H}_k + \alpha_k \mathbf{z}_k \mathbf{z}_k^T \quad (18)$$

sendo $\mathbf{z}_k \in \mathbb{R}^n$ e $\alpha_k \in \mathbb{R}$.

Aproximações Quadráticas

Correção de Posto 1

- O termo $\alpha_k \mathbf{z}_k \mathbf{z}_k^T$ é uma matrix $n \times n$ com posto no máximo igual a 1.
- Supondo que a função objetivo seja exatamente quadrática, é preciso definir α_k e $\mathbf{z}_k$ em função dos valores conhecidos (os vetores $[\mathbf{x}_k]$ e $[\nabla f(\mathbf{x}_k)]$):

$$\tilde{H}_{k+1} \mathbf{r}_i = \mathbf{v}_i \quad \forall i = 1, \dots, k \quad (19)$$

Essa relação é quase a mesma que (16), mas exige a igualdade apenas para os pontos já avaliados, até o índice k .

Aproximações Quadráticas

Correção de Posto 1

- Desenvolvimento da fórmula para $i = k$:

Substituindo-se (18) em (19), obtém-se:

$$\alpha_k \mathbf{z}_k \mathbf{z}_k^T \mathbf{r}_k = \mathbf{v}_k - \tilde{H}_k \mathbf{r}_k$$

$$(\mathbf{v}_k - \tilde{H}_k \mathbf{r}_k)(\mathbf{v}_k - \tilde{H}_k \mathbf{r}_k)^T = (\alpha_k \mathbf{z}_k \mathbf{z}_k^T \mathbf{r}_k)(\alpha_k \mathbf{r}_k^T \mathbf{z}_k \mathbf{z}_k^T) \quad (20)$$

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

Note que o termo de correção $\alpha_k \mathbf{z}_k \mathbf{z}_k^T$ depende de dados conhecidos:
 $\tilde{H}_k$, $\mathbf{v}_k$ e $\mathbf{r}_k$, ...

A menos da quantidade escalar $\alpha_k (\mathbf{z}_k^T \mathbf{r}_k)^2$.

- Para se determinar essa constante, faz-se:

$$\mathbf{r}_k^T (\alpha_k \mathbf{z}_k \mathbf{z}_k^T \mathbf{r}_k) = \mathbf{r}_k^T (\mathbf{v}_k - \tilde{\mathbf{H}}_k \mathbf{r}_k) \quad (21)$$

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

Substituindo-se (21) em (20) obtém-se:

$$\alpha_k \mathbf{z}_k \mathbf{z}_k^T = \frac{1}{\mathbf{r}_k^T \mathbf{v}_k - \mathbf{r}_k^T \tilde{\mathbf{H}}_k \mathbf{r}_k} (\mathbf{v}_k - \tilde{\mathbf{H}}_k \mathbf{r}_k) (\mathbf{v}_k - \tilde{\mathbf{H}}_k \mathbf{r}_k)^T \quad (22)$$

Voltando à fórmula recursiva para cálculo de $\tilde{\mathbf{H}}_{k+1}$:

$$\tilde{\mathbf{H}}_{k+1} = \tilde{\mathbf{H}}_k + \frac{1}{\mathbf{r}_k^T \mathbf{v}_k - \mathbf{r}_k^T \tilde{\mathbf{H}}_k \mathbf{r}_k} (\mathbf{v}_k - \tilde{\mathbf{H}}_k \mathbf{r}_k) (\mathbf{v}_k - \tilde{\mathbf{H}}_k \mathbf{r}_k)^T \quad (23)$$

Aproximações Quadráticas

Correção de Posto 1

Algorithm 13: Algoritmo de Correção de Posto 1

```
1  $k \leftarrow 0$ ;  
2  $\tilde{H}_k \leftarrow I$ ;  
3  $\mathbf{g}_k \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_k)$ ;  
4 while (critério de parada não for satisfeito) do  
5      $\mathbf{d}_k \leftarrow -\tilde{H}_k \mathbf{g}_k$ ;  
6      $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$ ;  
7      $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$ ;  
8      $\mathbf{g}_{k+1} \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_{k+1})$ ;  
9      $\mathbf{v}_k \leftarrow \mathbf{x}_k - \mathbf{x}_{k+1}$ ;  
0      $\mathbf{r}_k \leftarrow \mathbf{g}_k - \mathbf{g}_{k+1}$ ;  
1      $\tilde{H}_{k+1} = \tilde{H}_k + \frac{1}{\mathbf{r}_k^T \mathbf{v}_k - \mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k} (\mathbf{v}_k - \tilde{H}_k \mathbf{r}_k)(\mathbf{v}_k - \tilde{H}_k \mathbf{r}_k)^T$ ;  
2      $k \leftarrow k + 1$ ;  
3 end
```

Aproximações Quadráticas

Correção de Posto 1

- Se a função objetivo for quadrática, a convergência exata do algoritmo para o mínimo global ocorrerá em no máximo n passos;
- Note que **não se garante** que os pontos tomados ao longo da otimização gerem vetores $\mathbf{v}_i$ linearmente independentes.
- Caso a função seja exatamente quadrática, estes pontos *geram* necessariamente vetores $\mathbf{v}_i$ linearmente independentes.

Aproximações Quadráticas

Correção de Posto 1

- Não há vantagem computacional em se utilizar o *Algoritmo de Correção de Posto 1* em lugar da fórmula exata se $f(\cdot)$ for quadrática;
- No caso geral da otimização de funções não-lineares não quadráticas, **a Hessiana da função objetivo não será em geral constante**, e não ocorrerá a convergência em n iterações.
- O Algoritmo de Correção de Posto 1 torna-se então vantajoso, pois **a estimativa da Hessiana vai mudando dinamicamente**, de forma a acompanhar a variação dessa Hessiana.

Aproximações Quadráticas

Convergência do Algoritmo de Correção de Posto 1

- O Algoritmo de Correção de Posto 1 não pode ficar indefinido em nenhum ponto, uma vez que **não envolve inversões de matrizes**.
- A formulação do ACP 1 permite que a matriz H_{k+1} eventualmente **perca a propriedade de ser positiva definida**, caso ocorra:

$$\mathbf{r}_k^T \mathbf{v}_k - \mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k < 0 \quad (24)$$

- Devido a isso, **o algoritmo pode ficar 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.
- Quando for detectado um autovalor negativo, faz-se a substituição dessa matriz pela identidade.

Aproximações Quadráticas

Métodos Quase-Newton

- O *Algoritmo de Correção de Posto 1* é o exemplo mais simples de um algoritmo quase-Newton.
- Existem outros métodos que evitam as dificuldades de convergência do *Algoritmo de Correção de Posto 1*:
 - A matriz $\tilde{H}_k$ deve permanecer definida positiva, e,
 - Preferencialmente, bem condicionada.
- Dois métodos particularmente eficientes foram desenvolvidos:
 - DFP (Davidon-Fletcher-Powell);
 - BFGS (Broyden-Fletcher-Goldfarb-Shanno).
- Posteriormente, estes métodos foram agrupados em uma estrutura mais geral, a **família de Broyden**.

- A correção proposta pelo método DFP é dada por:

$$C_k^{DFP} = \frac{\mathbf{v}_k \mathbf{v}_k^T}{\mathbf{v}_k^T \mathbf{r}_k} - \frac{\tilde{H}_k \mathbf{r}_k \mathbf{r}_k^T \tilde{H}_k}{\mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k} \quad (25)$$

- A correção proposta pelo método BFGS é dada por:

$$C_k^{BFGS} = \left(1 + \frac{\mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{v}_k} \right) \frac{\mathbf{v}_k \mathbf{v}_k^T}{\mathbf{v}_k^T \mathbf{r}_k} - \frac{\mathbf{v}_k \mathbf{r}_k^T \tilde{H}_k + \tilde{H}_k \mathbf{r}_k \mathbf{v}_k^T}{\mathbf{r}_k^T \mathbf{v}_k} \quad (26)$$

- A correção genérica utilizada pelos métodos da *família de Broyden* é dada por:

$$C_k(\xi) = (1 - \xi) C_k^{DFP} + \xi C_k^{BFGS} \quad (27)$$

- Em todos os casos da família de Broyden, tem-se:

$$\tilde{H}_{k+1} = \tilde{H}_k + C_k(\xi) \quad (28)$$

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

Aproximações Quadráticas

Métodos Quase-Newton

- Com relação à correção usada pela 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 $\tilde{H}_k$ preservará sua propriedade de ser definida positiva.
 - Dados i e j tais que $0 \leq i < j \leq k$, então $\mathbf{v}_i^T H \mathbf{v}_j = 0$, ou seja, $\mathbf{v}_i$ e $\mathbf{v}_j$ são H -ortogonais.
 - Dado i tal que $0 \leq i \leq k$, então $\tilde{H}_{k+1} H \mathbf{v}_i = \mathbf{v}_i$.

Aproximações Quadráticas

Métodos Quase-Newton

Algorithm 14: Algoritmos Quase-Newton

```
1  $k \leftarrow 0; \tilde{H}_k \leftarrow I; \mathbf{g}_k \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_k);$   
2 while (critério de parada não for satisfeito) do  
3    $\mathbf{d}_k \leftarrow -\tilde{H}_k \mathbf{g}_k;$   
4    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k);$   
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k;$   
6    $\mathbf{g}_{k+1} \leftarrow \text{gradiente}(f(\cdot), \mathbf{x}_{k+1});$   
7    $\mathbf{v}_k \leftarrow \mathbf{x}_k - \mathbf{x}_{k+1};$   
8    $\mathbf{r}_k \leftarrow \mathbf{g}_k - \mathbf{g}_{k+1};$   
9    $C_k^{DFP} = \frac{\mathbf{v}_k \mathbf{v}_k^T}{\mathbf{v}_k^T \mathbf{r}_k} - \frac{\tilde{H}_k \mathbf{r}_k \mathbf{r}_k^T \tilde{H}_k}{\mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k};$   
10   $C_k^{BFGS} = \left(1 + \frac{\mathbf{r}_k^T \tilde{H}_k \mathbf{r}_k}{\mathbf{r}_k^T \mathbf{v}_k}\right) \frac{\mathbf{v}_k \mathbf{v}_k^T}{\mathbf{v}_k^T \mathbf{r}_k} - \frac{\mathbf{v}_k \mathbf{r}_k^T \tilde{H}_k + \tilde{H}_k \mathbf{r}_k \mathbf{v}_k^T}{\mathbf{r}_k^T \mathbf{v}_k};$   
11   $C_k(\xi) = (1 - \xi) C_k^{DFP} + \xi C_k^{BFGS};$   
12   $\tilde{H}_{k+1} = \tilde{H}_k + C_k(\xi);$   
13   $k \leftarrow k + 1;$   
14 end
```

Aproximações Quadráticas

Métodos Quase-Newton

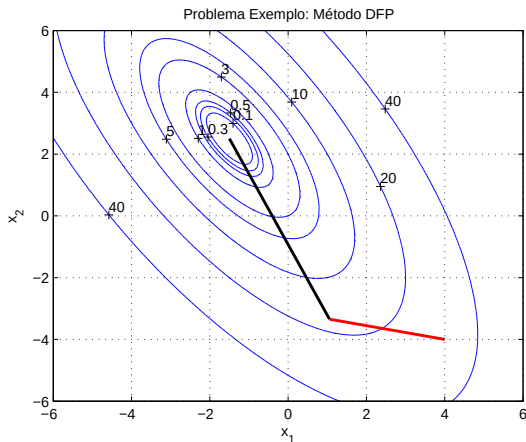


Figura: Solução usando o Método DFP.

Método dos Gradientes Conjugados

Introdução

Histórico

- Apresentado pela primeira vez em 1908 por Schmidt, reinventado de forma independente em 1948 e aprimorado nos anos 1950;
- Desenvolvido inicialmente para a solução de sistemas lineares, ainda usado em sistemas com matrizes esparsas;
- Em 1964, Fletcher e Reeves generalizaram o método para resolver problemas de otimização não linear irrestrita.

Método dos Gradientes Conjugados

Introdução

Solução de sistemas lineares

O método dos gradientes conjugados foi desenvolvido para resolver iterativamente grandes sistemas lineares da forma

$$\mathbf{Ax} = \mathbf{b}$$

com $\mathbf{A}$ simétrica e definida positiva.

Método dos Gradientes Conjugados

Introdução

Solução de sistemas lineares

Considere a forma quadrática:

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

O mínimo global dessa função pode ser obtido a partir da condição de otimalidade de 1a ordem:

$$\nabla f(\mathbf{x}) = \mathbf{A} \mathbf{x} - \mathbf{b} = 0$$

O mínimo de f é também a solução do sistema linear $\mathbf{A} \mathbf{x} = \mathbf{b}$.

Método dos Gradientes Conjugados

Introdução

Solução de sistemas lineares

O método atualiza a solução dando um passo α_k na direção oposta ao gradiente. A direção oposta ao gradiente é dada por:

$$-\nabla f(\mathbf{x}) = \mathbf{b} - \mathbf{Ax} = \mathbf{r} \text{ (resíduo)}$$

Assim:

$$\text{dado } \mathbf{x}_k \Rightarrow \mathbf{r}_k = \mathbf{b} - \mathbf{Ax}_k$$

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{r}_k$$

Método dos Gradientes Conjugados

Introdução

Solução de sistemas lineares

O tamanho do passo pode ser determinado analiticamente:

$$\frac{d}{d\alpha} f(\mathbf{x}_{k+1}) = \nabla f(\mathbf{x}_{k+1})' \frac{d}{d\alpha} \mathbf{x}_{k+1} = \nabla f(\mathbf{x}_{k+1})' \frac{d}{d\alpha} (\mathbf{x}_k + \alpha_k \mathbf{r}_k) = -\mathbf{r}'_{k+1} \mathbf{r}_k$$

o que implica resíduos ortogonais:

$$\mathbf{r}'_{k+1} \mathbf{r}_k = 0$$

$$(\mathbf{b} - \mathbf{A}\mathbf{x}_{k+1})' \mathbf{r}_k = 0$$

$$(\mathbf{b} - \mathbf{A}\mathbf{x}_k - \alpha_k \mathbf{A}\mathbf{r}_k)' \mathbf{r}_k = 0$$

que resulta em:

$$\alpha_k = \frac{\mathbf{r}'_k \mathbf{r}_k}{\mathbf{r}'_k \mathbf{A} \mathbf{r}_k}$$

Método dos Gradientes Conjugados

Algoritmo para Otimização Linear

Algorithm 15: Método dos Gradientes Conjugados

Input: $\mathbf{x}_0$, matriz de coeficientes $\mathbf{A}$

```
1  $k \leftarrow 0$ ;  
2  $\mathbf{r}_k \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_k$ ;  $\mathbf{d}_k \leftarrow \mathbf{r}_k$ ;  
3 while  $\neg$  critério de parada do  
4    $\alpha_k \leftarrow \frac{\mathbf{r}_k' \mathbf{r}_k}{\mathbf{d}_k' \mathbf{A} \mathbf{d}_k}$  ;  
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$  ;  
6    $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{d}_k$  ;  
7    $\beta_k \leftarrow \frac{\mathbf{r}_{k+1}' \mathbf{r}_{k+1}}{\mathbf{r}_k' \mathbf{r}_k}$  ;  
8    $\mathbf{d}_{k+1} \leftarrow \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$  ;  
9    $k \leftarrow k + 1$ ;  
0 end
```

Método dos Gradientes Conjugados

Introdução

Gradientes conjugados para otimização não linear

A versão do método para otimização não linear apresenta três diferenças básicas:

- O resíduo não pode ser calculado recursivamente;
- O tamanho do passo não pode ser determinado analiticamente - deve-se usar busca unidirecional;
- Há diferentes escolhas para β .

Método dos Gradientes Conjugados

Algoritmo para Otimização Não Linear

Algorithm 16: Método dos Gradientes Conjugados

Input: $\mathbf{x}_0 \in \mathcal{X}$, função-objetivo $f(\cdot)$

```
1  $k \leftarrow 0$ ;  
2  $\mathbf{r}_0 \leftarrow -\nabla f(\mathbf{x}_0)$ ;  $\mathbf{d}_0 \leftarrow \mathbf{r}_0$ ;  
3 while  $\neg$  critério de parada do  
4    $\alpha_k \leftarrow \arg \min_{\alpha} f(\mathbf{x}_k + \alpha \mathbf{d}_k)$  ;  
5    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \mathbf{d}_k$  ;  
6    $\mathbf{r}_{k+1} \leftarrow -\nabla f(\mathbf{x}_{k+1})$  ;  
7   Calcular  $\beta_k$ ;  
8    $\mathbf{d}_{k+1} \leftarrow \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$  ;  
9    $k \leftarrow k + 1$ ;  
0 end
```

Método dos Gradientes Conjugados

Introdução

Gradientes conjugados para otimização não linear

Duas fórmulas bem conhecidas para β_k são:

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

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

Método dos Gradientes Conjugados

Introdução

- Como o método se baseia na geração de n direções conjugadas no espaço n -dimensional, deve-se reiniciar o método a cada n iterações em problemas não quadráticos;
- Converge em n iterações em funções quadráticas. Em funções não quadráticas, as direções deixam de ser conjugadas após algumas iterações, sendo preciso reinício periódico;
- Em geral, métodos quasi-Newton convergem em menos iterações, porém requerem mais computação e mais memória por iteração. Portanto, gradientes conjugados é mais indicado em problemas de elevada dimensão.

- Método Nelder-Mead Simplex;
- Método Hooke-Jeeves.

Métodos sem Derivadas

Motivação

- Métodos baseados em derivadas convergem mais rapidamente, mas só podem ser usados em problemas caracterizados por funções continuamente diferenciáveis;
- Em problemas com muitas variáveis, os erros numéricos introduzidos por aproximações no cálculo do gradiente podem se tornar significativos.

Métodos sem Derivadas

Método Nelder-Mead Simplex

- O método Nelder-Mead Simplex foi desenvolvido para otimização não linear (não confundir com o método Simplex para programação linear);
- O método trabalha com $n + 1$ pontos a cada iteração, e elimina o “pior” ponto;
- Um novo ponto é criado com base no ponto eliminado.

Métodos sem Derivadas

Método Nelder-Mead Simplex

Fecho convexo

O fecho convexo (ou invólucro convexo) de um conjunto $\mathcal{A}$, denotado por $\bar{\mathcal{A}}$, é definido como a interseção de todos os conjuntos convexos que contêm $\mathcal{A}$.

Politopo

O fecho convexo de um conjunto finito de pontos $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k \in \mathbb{R}^n$ é chamado politopo.

Métodos sem Derivadas

Método Nelder-Mead Simplex

Simplex

Se $\mathbf{x}_2 - \mathbf{x}_1, \mathbf{x}_3 - \mathbf{x}_1, \dots, \mathbf{x}_k - \mathbf{x}_1$ são vetores linearmente independentes, então o fecho convexo desse conjunto de pontos é chamado simplex.

- O número máximo de vetores linearmente independentes em $\mathbb{R}^n$ é n , portanto um simplex em $\mathbb{R}^n$ possui $n + 1$ vértices.
- O simplex é assim chamado por ser o polítopo mais simples de sua dimensão.

Métodos sem Derivadas

Método Nelder-Mead Simplex

Usaremos a seguinte notação:

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

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

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

Métodos sem Derivadas

Método Nelder-Mead Simplex

Reflexão: Visa rejeitar a pior solução e avançar o simplex na direção de melhora. Essa operação reflete o pior vértice do simplex sobre a face oposta:

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

Expansão: Expande o simplex na direção de melhora:

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

Contração externa: Contrai o simplex na direção de melhora:

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

Contração interna: Contrai o simplex internamente:

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

Métodos sem Derivadas

Método Nelder-Mead Simplex

Algorithm 17: Método Nelder-Mead Simplex

Input: $\mathbf{x}_0 \in \mathcal{X}$, função-objetivo $f(\cdot)$

```
1  $k \leftarrow 0$ ;  
2 while  $\neg$  critério de parada do  
3    $\mathbf{x}_r = \hat{\mathbf{x}} + \alpha (\hat{\mathbf{x}} - \mathbf{x}_w)$  ;  
4   if  $f(\mathbf{x}_r) < f(\mathbf{x}_b)$  then Expansão  
5     calcule e avalie  $\mathbf{x}_e$ ;  
6     if  $f(\mathbf{x}_e) < f(\mathbf{x}_r)$  then  $\mathbf{x}_{new} = \mathbf{x}_e$ ;  
7     else  $\mathbf{x}_{new} = \mathbf{x}_r$ ;  
8   else if  $f(\mathbf{x}_r) < f(\mathbf{x}_s)$  then  $\mathbf{x}_{new} = \mathbf{x}_r$ ;  
9   else if  $f(\mathbf{x}_r) < f(\mathbf{x}_w)$  then Contração externa  
10     calcule e avalie  $\mathbf{x}_{c+}$ ;  
11     if  $f(\mathbf{x}_{c+}) \leq f(\mathbf{x}_w)$  then  $\mathbf{x}_{new} = \mathbf{x}_{c+}$ ;  
12   else if  $f(\mathbf{x}_r) \geq f(\mathbf{x}_w)$  then Contração interna  
13     calcule e avalie  $\mathbf{x}_{c-}$ ;  
14     if  $f(\mathbf{x}_{c-}) \leq f(\mathbf{x}_w)$  then  $\mathbf{x}_{new} = \mathbf{x}_{c-}$ ;  
15   else  
16     Encolhe o simplex  
17   end
```

Métodos sem Derivadas

Método Nelder-Mead Simplex

- Critérios de parada do método são baseados no tamanho (volume) do simplex;
- Convergência para funções convexas provada apenas recentemente;
- Inicialização do simplex pode ser obtida com perturbações ortogonais a $\mathbf{x}_0$;
- Método usado na função `fminsearch` do MatlabTM.

Métodos sem Derivadas

Método Hooke-Jeeves

- O método Hooke-Jeeves testa pontos padrões a partir do ponto atual;
- Ele alterna direções de pesquisa na direção dos eixos coordenados com direções de melhora do tipo $\mathbf{x}_{k+1} - \mathbf{x}_k$.

Métodos sem Derivadas

Método Hooke-Jeeves

- 1 Seja $\mathbf{x}_0$ o ponto inicial, $\mathbf{e}_1, \dots, \mathbf{e}_n$ as direções coordenadas, e $\mathbf{y}_0 = \mathbf{x}_0$. O algoritmo testa os pontos $\mathbf{y}_i \pm \lambda \mathbf{e}_{i+1}$, fazendo um movimento na direção de melhora ou ficando no ponto atual;
- 2 Após pesquisar todas as coordenadas, terminamos no ponto $\mathbf{x}_{k+1}$;
- 3 Neste ponto, efetuamos uma pesquisa na direção $\mathbf{x}_{k+1} - \mathbf{x}_k$:

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

- 4 A partir desse ponto, reinicia-se a pesquisa nas direções coordenadas. Se a função não decrescer, então fazemos $\lambda \leftarrow \lambda/2$ até que a precisão desejada seja atingida.

Métodos sem Derivadas

Método Hooke-Jeeves

Algorithm 18: Método Hooke-Jeeves

Input: $\mathbf{x}_0 \in \mathcal{X}$, função-objetivo $f(\cdot)$, λ , α

```
1  $k \leftarrow 0$ ,  $\mathbf{y}_0 \leftarrow \mathbf{x}_k$ ;  
2 while  $\lambda > \xi$  do  
3   foreach  $i = 0 \dots, n - 1$  do  
4     if  $f(\mathbf{y}_i + \lambda \mathbf{e}_{i+1}) < f(\mathbf{y}_i)$  then  $\mathbf{y}_{i+1} \leftarrow \mathbf{y}_i + \lambda \mathbf{e}_{i+1}$ ;  
5     else if  $f(\mathbf{y}_i - \lambda \mathbf{e}_{i+1}) < f(\mathbf{y}_i)$  then  $\mathbf{y}_{i+1} \leftarrow \mathbf{y}_i - \lambda \mathbf{e}_{i+1}$ ;  
6     else  $\mathbf{y}_{i+1} \leftarrow \mathbf{y}_i$ ;  
7   end  
8   if  $f(\mathbf{y}_n) < f(\mathbf{x}_k)$  then  
9      $\mathbf{x}_{k+1} \leftarrow \mathbf{y}_n$ ;  
10     $\mathbf{y}_0 \leftarrow \mathbf{x}_{k+1} + \alpha (\mathbf{x}_{k+1} - \mathbf{x}_k)$ ;  
11  else  
12     $\lambda \leftarrow \lambda/2$ ;  
13     $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k$ ;  
14     $\mathbf{y}_0 \leftarrow \mathbf{x}_k$ ;  
15  end  
16   $k \leftarrow k + 1$ ;  
17 end
```

Métodos sem Derivadas

Método Hooke-Jeeves

- 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.