

# Método de Newton truncado

Marina Andretta

ICMC-USP

25 de agosto de 2014

Baseado no livro Numerical Optimization, de J. Nocedal e S. J. Wright.

Lembre-se que o passo de Newton  $p_k^N$  é calculado resolvendo-se o sistema linear simétrico

$$\nabla^2 f_k p_k^N = -\nabla f_k. \quad (1)$$

Para garantir convergência global, pedimos que a direção  $p_k^N$  seja de descida, o que acontece quando  $\nabla^2 f_k$  é definida positiva.

Se a Hessiana  $\nabla^2 f_k$  não é definida positiva ou está perto de ser singular, a direção  $p_k^N$  pode ser de subida ou pode ser excessivamente longa.

Para contornar este problema usaremos duas estratégias:

- 1 **Modificar a matriz Hessiana  $\nabla^2 f_k$**  antes ou durante a resolução do sistema (1) para que ela se torne suficientemente definida positiva. Este método é chamado de **método de Newton modificado**.
- 2 Resolver o sistema (1) usando um **método de gradientes conjugados** que pára quando uma direção de curvatura negativa é encontrada. Este método é chamado de **método de Newton truncado**.

Outra preocupação ao desenvolver métodos práticos do tipo Newton é manter o custo computacional o menor possível.

No **método de Newton truncado** isto é feito **terminando a iteração de gradientes conjugados antes da solução exata de (1)**. Este método de Newton *inexato*, então, calcula uma aproximação  $p_k$  do passo de Newton  $p_k^N$ .

Ao usar um **método direto** para resolver o sistema linear (1), **podemos aproveitar a esparsidade da matriz Hessiana** usando, por exemplo, a eliminação de Gauss esparsa.

O problema do **método de Newton** é ter de **resolver o sistema (1) exatamente**. Técnicas baseadas em eliminação de Gauss e outras fatorações podem ser custosas quando o número de variáveis é grande.

Uma solução precisa de (1) pode não ser garantida no caso geral, já que o modelo quadrático usado para gerar o sistema newtoniano pode não fornecer uma boa aproximação do comportamento de  $f$ , principalmente quando o iterando  $x_k$  está longe da solução  $x^*$ .

Assim, é interessante **resolver o sistema (1) usando um método iterativo** que pare em uma **solução aproximada (inexata) do sistema**.

A maior parte das regras para terminar um método iterativo para resolver o sistema (1) são baseadas no resíduo

$$r_k = \nabla^2 f_k p_k + \nabla f_k, \quad (2)$$

com  $p_k$  passo de Newton inexato.

# Passos de Newton inexato

Como o tamanho do resíduo muda quando  $f$  é multiplicada por uma constante (ou seja,  $r_k$  não é invariante quanto ao escalamento de  $f$ ), consideramos seu tamanho relativo ao vetor do lado direito de (1) - ou seja,  $\nabla f_k$ .

Então, terminamos a execução do método iterativo quando

$$\|r_k\| \leq \eta_k \|\nabla f_k\|,$$

(3)

onde a sequência  $\{\eta_k\}$ , com  $0 < \eta_k < 1$  para todo  $k$ , é chamada de *sequência de forcing*.

A ordem de convergência de métodos de Newton inexato, baseados em (1)-(3), é afetada pela sequência de *forcing*.

O primeiro resultado diz que convergência local é obtida apenas garantindo que  $\eta_k$  esteja longe de 1.

**Teorema 1:** *Suponha que  $\nabla f(x)$  tenha derivada contínua em uma vizinhança da solução  $x^*$ . Suponha que  $\nabla^2 f(x^*)$  seja definida positiva. Considere a iteração da forma  $x_{k+1} = x_k + p_k$ , com  $p_k$  que satisfaz  $\|r_k\| \leq \eta_k \|\nabla f(x_k)\|$ . Suponha que  $\eta_k \leq \eta$  para alguma constante  $\eta \in [0, 1)$ .*

*Então, se o ponto inicial  $x_0$  está suficientemente próximo de  $x^*$ , a sequência  $\{x_k\}$  converge para  $x^*$  linearmente. Ou seja, para todo  $k$  suficientemente grande, temos que*

$$\|x_{k+1} - x^*\| \leq c \|x_k - x^*\|, \quad 0 < c < 1.$$

Note que a hipótese sobre a sequência  $\{\eta_k\}$  não é muito restritiva, no sentido que, se fosse relaxada, o algoritmo não convergiria.

Mais especificamente, se fosse permitido que  $\eta_k \geq 1$ , o passo  $p_k = 0$  satisfaria (3) em toda iteração. Neste caso, o método faria  $x_k = x_0$  para todo  $k$  e não convergiria para a solução.

**Teorema 2:** *Suponha que as condições do Teorema 1 valham e suponha que os iterandos  $\{x_k\}$  gerados pelo método de Newton inexato convergem para  $x^*$ . Então a ordem de convergência é superlinear se  $\eta_k \rightarrow 0$  e quadrática se  $\eta_k = O(\|\nabla f(x_k)\|)$ .*

# Passos de Newton inexato

Para obter **convergência superlinear** podemos usar, por exemplo,  $\eta_k = \min\{0.5, \sqrt{\|\nabla f_k\|}\}$ . Usando  $\eta_k = \min\{0.5, \|\nabla f_k\|\}$  obteríamos **convergência quadrática**.

Todos os resultados apresentados aqui têm natureza local: supõe-se que a sequência  $\{x_k\}$  atinge uma vizinhança de  $x^*$ . Eles também supõem que o tamanho de passo unitário  $\alpha_k = 1$  é tomado e, então, estratégias de globalização (busca linear e regiões de confiança) não atrapalham a rápida convergência.

Veremos agora que **estratégias de Newton inexato** podem ser incorporadas em **implementações de métodos de Newton com busca linear**, gerando algoritmos com **boas propriedades de convergência local e global**.

O **método de Newton truncado** consiste em utilizar o **método de gradientes conjugados** na resolução do sistema newtoniano (1), a fim de satisfazer uma condição do tipo (3)

$$\|r_k\| \leq \eta_k \|\nabla f_k\|.$$

# Método de gradientes conjugados

O método de gradientes conjugados é um método iterativo que foi desenvolvido para resolver sistemas lineares  $Ax = b$ , com  $A$  simétrica e definida positiva.

Basicamente, este método parte de um ponto inicial  $x_0$  e, a cada iteração  $k$ , calcula uma direção conjugada a todas as anteriores (ou seja,  $(p^{(k)})^T A p^{(i)} = 0$ , para todo  $i < k$ ).

Calcula-se  $x^{(k+1)} = x^{(k)} + \alpha^{(k)} p^{(k)}$ , com  $\alpha^{(k)} = -\frac{r_k^T p^{(k)}}{(p^{(k)})^T A p^{(k)}}$ .

O método de gradientes conjugados encontra a solução do sistema linear em, no máximo,  $n$  iterações.

# Método de gradientes conjugados

**Método de gradientes conjugados:** Dados um ponto inicial  $x^{(0)}$ , uma matriz  $A$ , um vetor  $b$  e um escalar  $\epsilon > 0$ .

**Passo 1:** Faça  $r_0 \leftarrow Ax^{(0)} - b$ ,  $p^{(0)} \leftarrow -r_0$ ,  $k \leftarrow 0$ .

**Passo 2:** Se  $\|r_k\| \leq \epsilon$ , pare com  $x^{(k)}$  como solução.

**Passo 3:** Faça  $\alpha^{(k)} \leftarrow \frac{r_k^T r_k}{(p^{(k)})^T A p^{(k)}}$ .

**Passo 4:** Faça  $x^{(k+1)} \leftarrow x^{(k)} + \alpha^{(k)} p^{(k)}$ .

**Passo 5:** Faça  $r_{k+1} \leftarrow r_k + \alpha^{(k)} A p^{(k)}$ .

**Passo 6:** Faça  $\beta^{(k+1)} \leftarrow \frac{r_{k+1}^T r_{k+1}}{r_k^T r_k}$ .

**Passo 7:** Faça  $p^{(k+1)} \leftarrow -r_{k+1} + \beta^{(k+1)} p^{(k)}$ .

**Passo 8:** Faça  $k \leftarrow k + 1$  e volte para o Passo 2.

# Método de Newton truncado

Note que o **método de gradientes conjugados** foi desenvolvido para a resolução de **sistemas lineares definidos positivos**. No entanto, a Hessiana poderia possuir autovalores negativos longe da solução.

Portanto, **terminamos a iteração de gradientes conjugados assim que uma direção de curvatura negativa é gerada**. Esta adaptação do método dos gradientes conjugados garante que a direção  $p_k$  é de descida e que a convergência rápida do método de Newton puro é conservada.

# Método de Newton truncado

O sistema linear a ser resolvido é dados por

$$\nabla^2 f_k p_k = -\nabla f_k.$$

Usando a notação do algoritmo de gradientes conjugados, temos que  $A = \nabla^2 f_k$  e  $b = -\nabla f_k$ .

Denotaremos por  $\{x^{(i)}\}$  e  $\{p^{(i)}\}$  os iterandos e direções de busca gerados pelo método de gradientes conjugados.

# Método de Newton truncado

São impostas três condições ao **método de gradientes conjugados** para a resolução de sistema (1):

- 1 O ponto inicial para o método de gradientes conjugados é  $x^{(0)} = 0$ .
- 2 **Teste de curvatura negativa**: se uma direção  $p^{(i)}$  gerada pelo método de gradientes conjugados é tal que  $(p^{(i)})^T A p^{(i)} \leq 0$ , verificamos se  $i$  é a primeira iteração. Em caso positivo, **terminamos a iteração de gradientes conjugados**, definimos  $x^{(1)}$  como  $p^{(0)}$  e paramos. Caso contrário, paramos o método de gradientes conjugados imediatamente e usamos a solução  $x^{(i)}$ .
- 3 A **direção de Newton**  $p_k$  é definida como o iterando final do método de gradientes conjugados  $x^{(f)}$ .

# Método de Newton truncado

Se a condição  $(p^{(i)})^T A p^{(i)} \leq 0$  não é satisfeita, o método dos gradientes é o mesmo apresentado anteriormente.

Uma direção  $p^{(i)}$  que satisfaz essa condição com desigualdade estrita é chamada de **direção de curvatura negativa** para  $A$ .

Note que, **se uma direção de curvatura negativa é encontrada** na primeira iteração do **método de gradientes conjugados**, a direção  $-\nabla f_k$  **será usada no método de Newton**. Daí a razão para usar o ponto inicial  $x^{(0)} = 0$ .

É possível mostrar que, se uma direção de curvatura negativa é encontrada em uma iteração  $i > 1$  do método de gradientes conjugados, a aproximação  $x^{(i-1)}$  é uma direção de descida.

# Método de Newton truncado

Note que o método de Newton truncado não usa a Hessiana  $\nabla^2 f_k$  de maneira explícita, apenas em produtos matriz-vetor. Assim, o usuário pode fornecer uma rotina que calcula o produto  $\nabla^2 f_k v$ , para um dado vetor  $v$ , em vez de fornecer uma rotina que calcule  $\nabla^2 f_k$ .

Para definir o método de Newton truncado com busca linear usaremos  $\eta_k = \min\{0.5, \sqrt{\|\nabla f_k\|}\}$  para garantir convergência superlinear.

**Método de Newton truncado com busca linear:** Dado um ponto inicial  $x_0$  e um escalar  $\epsilon > 0$ .

**Passo 1:** Faça  $k \leftarrow 0$ .

**Passo 2:** Se  $\|\nabla f(x_k)\| \leq \epsilon$ , pare com  $x_k$  como solução.

**Passo 3:** Calcule uma direção de busca  $p_k$  aplicando o método de gradientes conjugados para resolver o sistema  $\nabla^2 f_k p_k = -\nabla f_k$ , com ponto inicial  $x^{(0)} = 0$ .

Pare quando  $\|r_k\| \leq \min\{0.5, \sqrt{\|\nabla f_k\|}\} \|\nabla f_k\|$  ou uma direção de curvatura negativa for encontrada.

**Passo 4:** Faça  $x_{k+1} \leftarrow x_k + \alpha_k p_k$ ,  $\alpha_k$  satisfaz condições de Wolfe ou é calculado usando *backtracking* com Armijo.

**Passo 5:** Faça  $k \leftarrow k + 1$  e volte para o Passo 2.