SME0305 - 2016

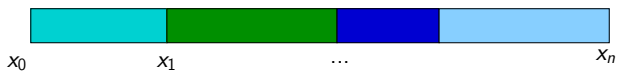
Gustavo C. Buscaglia / Roberto F. Ausas

ICMC - Ramal 738176, gustavo.buscaglia@gmail.com
ICMC - Ramal 736628, rfausas@gmail.com

Relaxações e gradientes

Nas aulas passadas vimos que:

- As matrizes simétricas e definidas ou semi-definidas positivas são frequentes na física e na engenharia, especialmente mecânica e civil.
- A configuração da figura,



é uma estrutura bastante simplificada que porém preserva muitos dos aspectos essenciais do cálculo de estruturas.

- Sua energía elástica pode se escrever, em termos dos deslocamentos $u(0, \dots, n)$ como

$$E(u) = \frac{1}{2}\, u^T K u = \sum_{i=1}^{n} \frac{k_i}{2}\, (u_i - u_{i-1})^2 \ . \tag{1}$$

- Quando a partir das posições relaxadas `[0;X(1:n)]` dos nós (uniões) se fixa o nó 0 e se desloca o nó `n` à posição `a`, os deslocamentos dos extremos ficam conhecidos:

$$u_0 = 0, \qquad u_n = a - X_n \ . \tag{2}$$

- As posições minimizantes da energia dos outros $n-1$ nós surgem de resolver um sistema linear

$$\tilde{K}\ \tilde{u}\ =\ b \tag{3}$$

cuja matriz $\tilde{K}$ consiste das linhas e colunas 1 a $n-1$ (começando em zero) da matriz K e o vetor $b = (0, 0, \dots, 0, k_n u_n)^T$.

- O mesmo sistema de equações pode ser escrito como o equilibrio de forças,

$$T_i - T_{i+1} = 0 \qquad (i = 1, \dots, n-1) \ , \tag{4}$$

ou, substituindo $T_i = k_i(u_i - u_{i-1})$,

$$-\ k_i\, u_{i-1}\ +\ (k_i + k_{i+1})\, u_i\ -\ k_{i+1}\, u_{i+1}\ =\ 0 \qquad (i = 1, \dots, n-1) \ . \tag{5}$$

- Deveria resulta claro que, se houver uma força externa aplicada no nó i, os lados direitos das últimas duas equações deveriam ser substituidos por f_i.

Relaxação

- De (5) resulta claro que, considerando os deslocamentos dos nós $i-1$ e $i+1$ fixos, o nó i não está em equilibrio. O deslocamento de equilibrio seria

$$u_i^{\text{eq}}(u_{i-1}, u_{i+1}) = \frac{k_i}{k_i + k_{i+1}}\, u_{i-1} + \frac{k_{i+1}}{k_i + k_{i+1}}\, u_{i+1} \tag{6}$$

$$= \alpha_i\, u_{i-1} + (1 - \alpha_i)\, u_{i+1} \tag{7}$$

  Notar que $0 < \alpha_i < 1$ se $0 < k_i < +\infty$ para todo i.

- Notar que, em equilíbrio,

$$\min(u_{i-1}, u_{i+1}) < u_i < \max(u_{i-1}, u_{i+1}) \tag{8}$$

  por ser u_i uma média (pesada) de u_{i-1} e u_{i+1}. Essa propriedade se conhece como **princípio do máximo**.

- Uma **boa ideia**:

Método de relaxação: Percorrer os nós sequencialmente colocando cada nó na posição que seria de equilibrio se os conectados com ele estivessem nas suas posições definitivas.

```
while (dif>tol)
 dif=0;
 for i=1:n-1
  ueq=a(i)*u(i-1)+(1-a(i))*u(i+1);
  corr=omega*(ueq-u(i));
  u(i)=u(i)+corr;
  dif=max(dif,abs(corr));
 end
end
```

onde `omega` é um parâmetro de sobre- ou sub-relaxação.

- O método está baseado na física, é simples de programar e se estende facilmente ao caso de $u_i^{\text{eq}}(u_{i-1}, u_{i+1})$ não linear. O método **não termina até todos os nós estarem em equilíbrio**, o que corresponde ao equilíbrio global do sistema.

Exo. 1: Provar que o método de relaxação da página anterior não é outra coisa que o **método de Gauss-Seidel** (quando `omega=1`). Como mudar o programa anterior para que seja uma implementação do método de Jacobi?

Exo. 2: Que método conhecido corresponde à seguinte implementação vetorizada da relaxação?

```
while (dif>tol)
 ueq=a.*[0;u(1:n-2)]+(1-a).*u(2:n);
 corr=omega*(ueq-u(1:n-1));
 u(1:n-1)=u(1:n-1)+corr;
 dif=norm(corr,Inf);
end
```

Os métodos **naturais**, **intuitivos**, são muitas vezes métodos da Algebra Linear Computacional, só que escritos sem matrizes. Sendo capazes de identificar a equivalência podemos predizer o comportamento dos códigos.

Nossa **boa ideia** acabou sendo equivalente a um método já estudado (Gauss-Seidel), com sobre-relaxação. Matricialmente se escreve, decompondo $\tilde{K} = A = L + D + U$ e chamando $x = \tilde{u}$,

$$(L + D)d^{(k)} = -(Ax^k - b), \qquad x^{(k+1)} = x^{(k)} + \omega\, d^{(k)} \tag{9}$$

que é um caso particular do **algoritmo geral** com $M = L + D$ e $\beta_k = \omega$. Vale para ele, portanto, tudo o já deduzido para o algoritmo geral.

Lembrete: Algoritmo geral

1. Determinar $d^{(k)}$ resolvendo: $M\, d^{(k)} = -r^{(k)} = b - A\, x^{(k)}$.
2. Determinar β_k.
3. Avançar: $x^{(k+1)} = x^{(k)} + \beta_k\, d^{(k)}$

Otimização

- Outra **boa ideia**: Sabendo que a solução de $Ax = b$ (sendo A sim. def. pos.) minimiza $F(x) = (1/2)x^T Ax - x^T b$, escolher a direção $d^{(k+1)}$ do algoritmo geral na direção de **máxima descida**:

$$d^{(k)} = -\nabla F(x^{(k)}) = b - A x^{(k)} \ . \tag{10}$$

Isto nos indica que $M = \mathbb{I}$ (identidade).

Exo. 3: (Como escolher o β_k?) Provar que, de todos os pontos da forma $y = x^{(k)} + \beta_k \, d^{(k)}$, qualquer que seja a escolha de $d^{(k)}$, o valor de β_k que minimiza $F(y)$ é

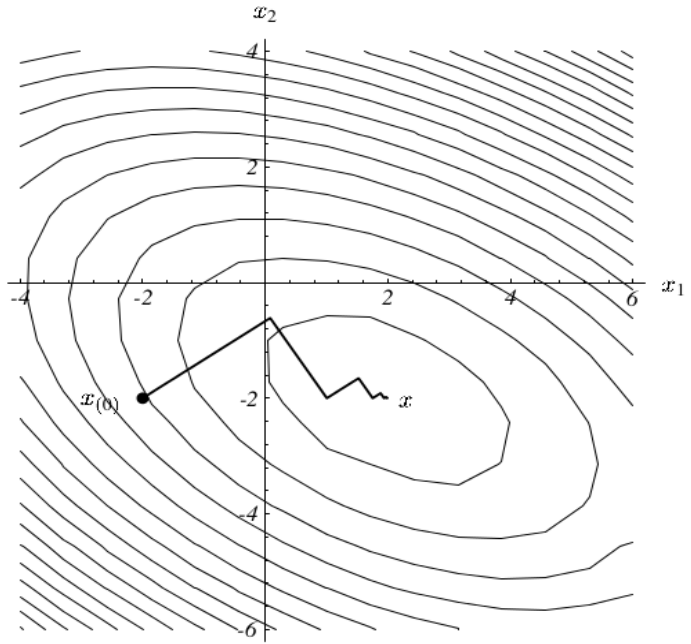
$$\beta_k = -\frac{{d^{(k)}}^T r^{(k)}}{{d^{(k)}}^T A \, d^{(k)}} \ . \tag{11}$$

Conferir que, com essa escolha, $\nabla F(x^{(k)} + \beta_k \, d^{(k)}) \perp d^{(k)}$.

O **método do gradiente** é um caso particular do nosso algoritmo geral, com $M = \mathbb{I}$ e β_k dado por (11).

Sua direção de avanço é, portanto, $d^{(k)} = -r^{(k)}$ e sua matriz de iteração (tal que $r^{(k+1)} = S^{(k)} r^{(k)}$) é

$$S^{(k)} = \mathbb{I} - \beta_k \, A \ . \tag{12}$$



Exo. 4: Determinar para que tipo de matriz A o método do gradiente converge sempre em uma iteração. É suficiente que A seja diagonal?

Matrizes sem matrizes

- Multiplicar a matriz $A = \tilde{K}$ pelo vetor $d = d^{(k)}$ pode ser programado sem nunca montar a matriz A. Seja $z = A\,d$,

```
z(1)=(k(1)+k(2))*d(1)-k(2)*d(2);
for i=2:n-2
 z(i)=-k(i)*d(i-1)+(k(i)+k(i+1))*d(i)-k(i+1)*d(i+1);
end
z(n-1)=-k(n-1)*d(n-2)+(k(n-1)+k(n))*d(n-1);
```

  As equações (5) permitem programar o produto matriz-vetor de uma maneira **matrix-free**.

- Isto também pode ser vetorizado como

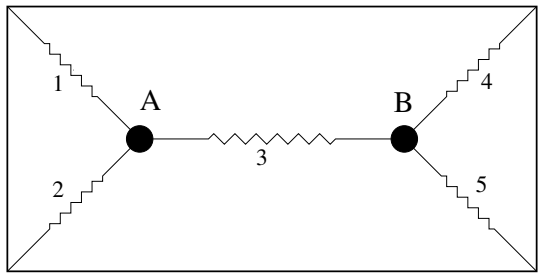
```
z=k(1:n-1).*(d-[0;d(1:n-2)])+k(2:n).*(d-[d(2:n-1);0]);
```

- O método do gradiente sem matrizes, então, é

```
while (dif>tol)
 ## Calcula d=b-Ktil*util
 b=[zeros(n-2,1);k(n)*u(n))];
 d=b-k(1:n-1).*(u-[0;u(1:n-2)])+k(2:n).*(u-[u(2:n-1);0]);
 ## Calcula Ktil*d
 Ad=k(1:n-1).*(d-[0;d(1:n-2)])+k(2:n).*(d-[d(2:n-1);0]);
 ## Calcula beta
 beta=d'*d/(d'*Ad);
 ## Avanca u
 u(1:n-1)=u(1:n-1)+beta*d;
 dif=norm(d,Inf);
end
```

Exo. 5: Entenda e explique cada linha do programa acima.

Exo. 6: Considere o sistema de massas e molas da figura na caixa $[0,4]\times[0,2]$. Todas as molas estão na sua configuração relaxada quando $x^A = (1,1)^T$ e $x^B = (3,1)^T$. As constantes das molas diagonais (1, 2, 4 e 5) é a mesma k_d, conhecida, e a constante da mola horizontal é k_h.



1. Escreva, para pequenos deslocamentos de A e B das posições relaxadas, a energia elástica do sistema como uma forma quadrática nas variáveis x_1^A, x_2^A, x_1^B, x_2^B. Identifique a matriz A tal que, sendo u o vetor de deslocamentos, se cumpra

$$E_{\text{elas}}(u) = \frac{1}{2}\, u^T A\, u\ . \tag{13}$$

Lembrete: A energia de uma mola é $k(\Delta\ell)^2/2$, onde $\Delta\ell$ é a variação de comprimento.

2. Nas mesmas hipótesis, escreva a energia gravitacional do sistema como uma forma linear em u, considerando a gravidade $g = (g_1, g_2)$ e as massas m_A e m_B conhecidas. Identifique um vetor $b \in \mathbb{R}^4$ (coluna) tal que

$$E_{\text{grav}}(u) = -u^T b \,. \tag{14}$$

3. Considere o sistema 4×4 linear

$$A\,u = b$$

no qual A e b surgem dos itens anteriores. Relacione com a teoria para concluir que a solução u^* dele minimiza a **energia total**,

$$E_{\text{tot}}(u) = E_{\text{elas}}(u) + E_{\text{grav}}(u) = \frac{1}{2}\, u^T A\, u \; - u^T b \,. \tag{15}$$

4. Escreva uma a uma as equações do sistema e verifique que as duas primeiras estabelecem o equilíbrio de forças agindo sobre a massa A, nas direções x_1 e x_2, respectivamente, e que as duas últimas fazem o mesmo para a massa B.

5. Programe em Octave as expressões antes deduzidas:

```
function Eelas=CalculaEelas(kd,kh,u)
```

```
function Egrav=CalculaEgrav(mA,mB,g1,g2,u)
function A=CalculaMatriz(kd,kh)
function b=CalculaVetor(mA,mB,g1,g2)
```

Os deslocamentos de equilíbrio são obtidos fazendo `ueq=A\b`.

6. Escreva, em Octave, um código que relaxe a massa A considerando a massa B fixa, e outra que relaxe a massa B considerando a A fixa. Compare com o seguinte código.

```
M=A(1:2,1:2);c=b(1:2)-A(1:2,3:4)*u(3:4);uA=M\c;u(1:2)=uA;
M=A(3:4,3:4);c=b(3:4)-A(3:4,1:2)*u(1:2);uB=M\c;u(3:4)=uB;
```

   A diferença das vistas antes, essa é uma relaxação **por blocos**, no sentido que cada passo são atualizadas blocos simultâneos de variáveis (duas no caso), o que requer resolver um sistema do tamanho de cada bloco.

7. Dividindo o sistema linear em blocos de 2 incógnitas se obtem

$$A\,u \;=\; \left(\begin{array}{c|c} D_A & E_A \\ \hline E_B & D_B \end{array}\right) \left(\frac{u_A}{u_b}\right) \;=\; \left(\frac{b_A}{b_B}\right) \;=\; b \tag{16}$$

   o que faz que

$$u_A \;=\; D_A^{-1}\,(b_A - E_A\,u_B) \tag{17}$$

$$u_B \;=\; D_B^{-1}\,(b_B - E_B\,u_A) \tag{18}$$

   Comparar com o código do item anterior.

Resolução por blocos:

- Seja N o número de incógnitas de um sistema, cuja resolução requer $O(N^3)$ operações.
- Seja $N = n\,k$ onde n é o número de incógnitas por bloco e k o número de blocos. Se cada iteração de um método iterativo requer resolver k sistemas de tamanho n, então o custo de uma iteração é $O(k\,n^3)$.
- A razão entre resolver o sistema diretamente e fazer uma iteração é

$$O(N^3/(k\,n^3)) = O(k^2)\ .$$

  Por isto, se o método iterativo convergir em menos de k^2 iterações dá para ganhar bastante.
- As contas mudam dependendo dos algoritmos de resolução, obviamente. O custo de memória também pode influenciar na decisão.

Exo. 7: Quando a matriz do sistema é esparsa, do tipo que aparece em problemas de corpos elásticos deformáveis 2D/3D, o custo de resolução observado é $O(N^{2.5})$. Qual seria o número de iterações máximo nesse caso para que uma relaxação por blocos possa valer a pena?

Como usar métodos iterativos?

- Métodos iterativos são usados diretamente de **bibliotecas** que contém implementações dos métodos bastante parecidas com as que dispomos em Octave.
- Para **matrizes simétricas definidas positivas**, uma variação do método do gradiente, chamado de **método de gradientes conjugados**, é considerado o mais eficiente.

  No Octave ele é implementado pela função

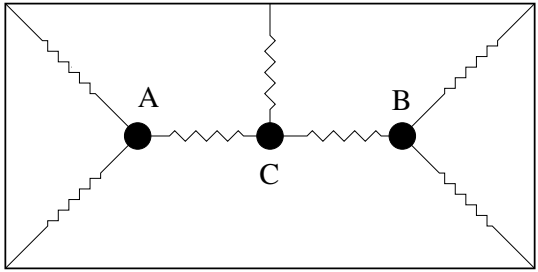
  ```
  x = pcg(A,b,tol,maxit,...)
  ```

  onde `tol` é a tolerância relativa e `maxit` é o máximo de iterações.
- Para matrizes gerais existem métodos eficientes também, um dos mais populares é o Generalized Minimum RESidual (`gmres`).
- **É possível utilizar esses métodos na versão matrix-free**, já que a matriz `A` e o lado direito `b` podem ser handles de função.

Mensagens para levar daqui:

- **Problemas elásticos em pequenas deformações** levam a **energias** que são **formas quadráticas** e equações lineares de equilíbrio cujas matrizes são **simétricas e definidas positivas**.
- Minimizar a energia é equivalente a equilibrar as forças.
- **Métodos de relaxação** incógnita a incógnita ou por blocos de incógnitas **correspondem a algoritmos iterativos da álgebra linear**.
- Esses algoritmos são de fácil implementação em Octave, por exemplo para experimentar antes de intervir num código industrial.
- Muitos algoritmos estão já implementados em Octave, e podem ser utilizados em qualquer outra linguagem **instalando bibliotecas**.
- Uma vantagem dos métodos iterativos é que **não é necessário construir em memória a matriz do sistema**. Os produtos matriz-vetor necessários podem ser calculados realizando cálculos locais com cada incógnita ou grupo delas.

Exo. 8: Repeter o **Exo. 4** para o sistema da Figura. A massa C está, na posição relaxada, no ponto médio entre A e B. As constantes das molas são k_d se diagonais, k_h se horizontais e k_v se verticais.



Verifique que a matriz A é singular quando $k_v = 0$, calcule seu núcleo e interprete físicamente.
É importante resolver os exos. 6 e 8 porque as matrizes também nos permitirão, mais tarde, calcular os modos de vibração dessas estruturas.