Computação Numérica

Ano letivo 2011/12

Orientações de resposta ao e-fólio B

Aqui temos uma possibilidade para a rotina de decomposição de Cholesky. A rotina abaixo implementa o algoritmo de uma maneira muito simples; vejamos. À direita temos os comentários. A cor do texto corresponde ao seguinte: a verde temos definições, a vermelho verificação de condições, a azul início/fecho de ciclos 'for' e a preto cálculos.

```
Function [L,pd]=cholesky(A);
```
Define a função 'Cholesky', que tem A como variável de entrada e L como variável de saída. Define também uma variável auxiliar pd, cuja função é indicar se a matriz A é positiva-definida (pd=1) ou não (pd=0).

```
  Lin=rows(A); col=columns(A);
```
Duas definições para simplificar o léxico.

```
  L=zeros(lin,col);
```
Define a matriz L, com valores iniciais nulos.

```
  If lin!=col puts "Erro: matriz deve ser quadrada. "; pd=0; return endif;
```
Verifica se A é quadrada. Se não for, termina a rotina devolvendo um erro e carregando pd com o valor 0.

```
For i=1:lin
```
Inicia ciclo for #1.

```
  L(i,i)=sqrt(A(i,i)-L(i,1:i-1)*L'(1:i-1,i));
```
Calcula o elemento diagonal de L. Aqui substituímos a soma $\sum_{k=1}^{i-1} L_{ik}^2$ por uma multiplicação matricial. Para compreender este passo note que $\sum_{k=1}^{i-1} L_{ik}^2 = \sum_{k=1}^{i-1} L_{ik}L_{ik} = \sum_{k=1}^{i-1} L_{ik}L_{ki}^T$.

```
  If L(i,i)==0 puts "Erro: elemento diagonal L nulo; matriz não e' positiva-definida."; pd=0; return endif;
```
Verifica se o elemento diagonal é nulo (antes da possível divisão por zero abaixo). Se isso acontecer, a matriz não é positiva-definida e a rotina termina e devolve um erro e pd=0.

```
    For j=i+1:lin
```
Inicia ciclo for #2.

```
      L(j,i)=(A(j,i)-L(i,1:j-1)*L'(1:j-1,j))/L(i,i);
```
Calcula os elementos não-diagonais de L. Novamente substituímos a soma por uma multiplicação matricial, desta feita usando $\sum_{k=1}^{i-1} L_{ik}L_{jk} = \sum_{k=1}^{i-1} L_{ik}L_{kj}^T$.

```
    Endfor
```
Fecho de ciclo for #2.

```
Endfor
```
Fecho de ciclo for #1. A matriz L está completa. Falta agora verificar se está correta.

```
  For i=1:lin; for j=1:lin
```
Inicia ciclos for #3 e #4 para reconstruir A e assim verificar se A é positiva-definida.

```
    If abs((L*L'-A)(i,j))>10^-12 puts "Erro: reconstrução errada; matriz não e' positiva-definida. "; pd=0; return endif;
```
Se a matriz A não for positiva-definida, a decomposição não a reconstrói (dentro da precisão 10^{-12}) e a rotina devolve um erro e pd=0.

```
  Endfor; endfor;
```
Fecho de ciclos for #3 e #4.

```
pd=1;
```
Se a rotina chegar aqui, a matriz A é positiva-definida e tem-se pd=1.

```
endfunction
```

Note-se que para L_{11} a multiplicação `L(i,1:i-1)*L'(1:i-1,i)` não está definida. O Octave assume o resultado desta operação como sendo o conjunto vazio, não a considerando para efeitos de cálculo de L_{11}. Isto permite-nos implementar o algoritmo de uma forma extremamente simples, apenas com dois ciclos 'for'.

Vejamos agora uma possível rotina para a resolução de $Ax = b$, recorrendo à decomposição de Cholesky.

`function [x]= rsys(A,b)`	Define a função 'rsys' (resolve sistema). Esta rotina vai entrar com uma matriz A e um vetor-coluna b e devolve um vetor-coluna, x.
`if columns(b)>1 puts "Erro: b deve ser matriz-coluna. "; return endif;`	Verifica se b é de facto um vetor-coluna.
`if columns(A)!=rows(b) puts "Erro: A e b incompatíveis. "; return endif;`	Verifica se a multiplicação matricial Ax faz sentido.
`[L,pd]=cholesky(A);`	Recorre à rotina 'cholesky' para achar L.
`if pd==0 puts "Erro: matriz A não e' positiva-definida."; return endif;`	Verifica se a rotina 'cholesky' devolveu uma matriz positiva-definida. Se sim, pode prosseguir. Se não, interrompe aqui devolvendo um erro.
`lin=rows(A); col=columns(A);`	Definições para simplificar a escrita da rotina.
`x=zeros(col,1);`	Define vetor de saída x, com a dimensão certa e zeros de início.
`y=zeros(col,1);`	Define vetor auxiliar y da mesma forma.
`for i=1:lin`	Ciclo for para cálculo de y por substituição direta.
`y(i)=(b(i)-L(i,1:i-1)*y(1:i-1))/L(i,i)`	Cálculo dos elementos de y.
`endfor;`	Fecho do ciclo for de substituição direta.
`for j=lin:-1:1`	Ciclo for para cálculo de x por substituição inversa. Note-se a forma de obter a sequência n, n-1, n-2, ..., 2, 1. Isto é necessário para que o cálculo dos elementos de x seja feito pela ordem certa.
`x(j)=(y(j)-L'(j,j+1:lin)*x(j+1:lin))/L'(j,j)`	Cálculo dos elementos de x.
`endfor;`	Fecho do ciclo for de substituição inversa.
`Endfunction`	

Apliquemos agora as duas rotinas ao sistema de equações lineares

$$\begin{pmatrix} 116 & 44 & 14 & 7 \\ 44 & 56 & 24 & 14 \\ 14 & 24 & 68 & 56 \\ 7 & 14 & 56 & 49 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \\ 4 \\ 6 \end{pmatrix}$$

Introduzindo as matrizes A e b na linha de comandos do Octave e correndo

```
> [x]=rsys(A,b)
```

Obteremos

```
x =
  -0.017343
   0.208897
  -1.106065
   1.329316
```

Para verificar que o resultado obtido está certo podemos fazer

```
> A*x
```

Ao que o Octave devolverá b:

```
ans =
   1.00000
   3.00000
   4.00000
   6.00000
```

confirmando assim que o vetor x foi calculado corretamente.