Analyse Numérique

Corrigés des Travaux Pratiques 2013 – 2014
Séance 3

1. Le raisonnement de la page 17 du Chapitre 1 montre qu'un algorithme qui a la stabilité directe doit satisfaire (pour une norme donnée, ici on considère la norme euclidienne $\|\cdot\|_2$)

$$\frac{||\hat{\mathbf{x}} - \mathbf{x}||_2}{||\mathbf{x}||_2} \;\leq\; C_1 C_2 \kappa(A) u \,, \tag{1}$$

avec $\mathbf{x}$ la solution exacte, $\hat{\mathbf{x}}$ la solution calculée, $u = 1.1 \cdot 10^{-16}$ et $C_1 C_2 \;\geq\; 1$ petit. Pour les systèmes linéaires le conditionnement est donné par

$$\kappa(A) \;=\; \|A^{-1}\|_2 \|A\|_2$$

et peut être estimé avec l'instruction `cond`. Le membre de droite dans (1) (sans facteur $C_1 C_2$ et avec conditionnement estimé via `cond`) pour les trois systèmes vaut :

système	$\kappa(A)$	$u\kappa(A)$
(1)	2.25	$2.5\,10^{-16}$
(2)	$3.3\,10^{8}$	$3.7\,10^{-8}$
(3)	1.002	$1.1\,10^{-16}$

2. factorisation LU :

```
function [L U] = an_lu(A)
% [L U] = an_lu(A) retourne les matrices triangulaire
% inférieure L et triangulaire supérieure U telles que
%                    A = LU,
% pour autant que A est carrée et qu'une telle
% factorisation existe;
m = size(A,1); n = size(A,2);
if(n~=m)
  error('A doit être carrée');
else
  for k = 1:n
    for j = k:n
      U(k,j) = A(k,j);
    end
    L(k,k) = 1;
    for i = k+1:n
      L(i,k) = A(i,k)/A(k,k);
    end
    for i = k+1:n
      for j = k+1:n
        A(i,j) = A(i,j)-L(i,k)*U(k,j);
      end
    end
  end
end % if
```

en complétant ce dernier avec une résolution du système triangulaire inférieur et supérieur (ici on présente le programme pour le système triangulaire inférieur ; celui pour le système triangulaire supérieur est en Annexe) :

```
function x = an_lts(L,b)
% x = an_lts(L,b) retourne la solution du système
%                            Lx=b
% avec L triangulaire inférieure et b un vecteur
% de dimensions compatibles
m = size(L,1); n = size(L,2);
if(n~=m)
  error('L doit être carrée');
elseif(size(b,2) > 1 || size(b,1)~=n)
  error(['b doit être un vecteur de dimensions ',...
         int2str(n),' x 1']);
else
  for k = 1:n
    x(k) = b(k);
    for j = 1:k-1
      x(k) =x(k) - L(k,j)*x(j);
    end
    x(k) = x(k)/L(k,k);
  end
end % if
```

le programme pour la résolution du système LU est comme suit :

```
function x = an_solve(A,b)
[L U] = an_lu(A); % A = LU
y = an_lts(L,b);  % y = L\b;
y = y';
x = an_uts(U,y);  % x = U\y;
x = x';
```

Pour les trois systèmes l'erreur relative sur la solution $\frac{||\hat{\mathbf{x}}-\mathbf{x}||}{||\mathbf{x}||}$ vaut :
(1) $1.4\,10^{-16}$
(2) $1.3\,10^{-11}$
(3) $1.3\,10^{-11}$
En comparant avec les résultats du premier exercice on constate que la précision sur la solution du troisième système n'est pas celle d'une méthode stable (perte de 5 décimales en précision).

3.

```
function tp3ex3
x = [-1; -1; -1; -1; -1];
A = [1 -1000 0 0 0; ...
     1 1 -1000 0 0; ...
     0 1 1 -1000 0; ...
     0 0 1 1 -1000; ...
     -1000 0 0 1 1];
```

```
b = [999; 998; 998; 998; 998];
   % factorisation à l'aide de l'instruction lu
   %  d'Octave, voir help lu
[L U P] = lu(A);
   % solution
b = P*b; % max n-1=4 permutations des éléments de b
y = an_lts(L,b);  % y = L\b;
y = y';
xc = an_uts(U,y);  % x = U\y;
xc = xc';
  % erreur relative
norm(x-xc)/norm(x) % alors?
```

Annexe

Le code pour la résolution d'un système triangulaire supérieur.

```
function x = an_uts(U,b)
% x = an_uts(U,b) retourne la solution du systeme
%                          Ux=b
% avec U triangulaire supérieure et b un vecteur
% de dimensions compatibles
m = size(U,1); n = size(U,2);
if(n~=m)
  error('U doit être carrée');
elseif(size(b,2) > 1 || size(b,1)~=n)
  error(['b doit être un vecteur de dimensions ',...
         int2str(n),' x 1']);
else
  for k = n:-1:1
    x(k) = b(k);
    for j = k+1:n
      x(k) =x(k) - U(k,j)*x(j);
    end
    x(k) = x(k)/U(k,k);
  end
end % if
```