Analyse Numérique
Travaux Pratiques 2013 – 2014
Séance 6

Le programme `tp6pb` en annexe utilise la fonction `spl_interpol` pour construire un polynôme cubique par morceaux dont les dérivées d'ordre ≤ 2 sont continues et qui passe par les points $(x(i), y(i))$ avec les abscisses $x(i) = 1, 2, 3, 4, 5, 6$ et les ordonnées $y(i)$ générées aléatoirement. L'instruction `spl_interpol` détermine la fonction interpolante en résolvant un système linéaire tridiagonal : la résolution correspond à l'instruction `x = A\d` et se situe 6 lignes avant la fin.

1. Ecrivez une fonction Octave qui résout un système tridiagonal avec la méthode de Jacobi. Choisissez comme critère d'arrêt la réduction de la norme du résidu relativement à celle du second membre. Utilisez cette fonction pour résoudre le système susmentionné avec 10^{-3} comme valeur du critère d'arrêt.

2. Répétez l'exercice 1 avec la méthode de Gauss-Seidel.

3. Pour une matrice de la forme

$$A = \begin{pmatrix} d_1 + c & c & & & \\ c & d_2 + 2c & c & & \\ & c & \ddots & \ddots & \\ & & \ddots & d_{n-1} + 2c & c \\ & & & c & d_n + c \end{pmatrix}$$

avec d_i , $c > 0$, $i = 1, \ldots, n$, on peut montrer que

$$\kappa(A) \leq \frac{\max_i d_i + 4c}{\min_i d_i} . \tag{1}$$

Utilisez cette inégalité pour estimer le conditionnement de la matrice des exercices 1-2. Quelle erreur sur la solution garantit le critère d'arrêt utilisé ? Vérifiez en comparant avec la solution produite par `x = A\d`.

4. **(facultatif)** Montrez la relation (1) en montrant que :

a) pour la matrice A définie dans l'exercice précédent et pour tout vecteur $\mathbf{v} = (v_i)$ on a

$$\mathbf{v}^T A \mathbf{v} = c \sum_{i=1}^{n-1} (v_i + v_{i+1})^2 + \sum_{i=1}^{n} d_i v_i^2 ;$$

b) sachant que $0 \leq (v_i + v_{i+1})^2 \leq 2(v_i^2 + v_{i+1}^2)$ pour tout v_i et v_{i+1} , on a

$$\min_i d_i \leq \frac{\mathbf{v}^T A \mathbf{v}}{\mathbf{v}^T \mathbf{v}} \leq 4c + \max_i d_i \; ; \tag{2}$$

Annexe

Ces fonctions sont également disponibles sur le site web du cours :

http://mntek3.ulb.ac.be/pub/MATH-H-202/index.html

```
function tp6pb
   % générer les données & les représenter via plot
x = [1,2,3,4,5,6];
f = rand(length(x),1);
plot(x,f,'*r'); hold on;
   % construire la fonction d'interpolation
s = spl_interpol(x,f);
   % représenter la fonction d'interpolation
xr = x(1)+[0:0.01:1]*(x(end)-x(1));
for i = 1:length(xr)
    fr(i) = spl_eval(xr(i), x, s);
end
plot(xr,fr)
hold off
```

```
function s = spl_interpol(x,y)
% S = spl_interpol(X,Y)
%    la fonction determine le polynôme cubique par morceaux
%    qui passe par N points (X(1), Y(1)),...,(X(N), Y(N));
%    le polynôme entre les deux abscisses X(i) et X(i+1)
%    est defini via les parametres par S(i,1),..., S(i,4)
%
% arguments:
%    X - le vecteur de taille N x 1 qui contient
%        les points d'interpolation; on demande à ce que
%        N > 2 et X(1) < X(2) < ... < X(N)
%    Y - le vecteur de taille N x 1 qui contient
%        les valeurs aux points d'interpolation;
%
% sortie:
%    S - S(i,1), S(i,2), S(i,3) et S(i,4), i=1,...,N - 1
%        sont les parametres qui specifient le polynôme
%        d'interpolation entre les abscisses X(i) et X(i+1)
%
    n = length(x);
    if (n < 3)
        error(['Le nombre d"éléments de X doit être > 2']);
    elseif (min(size(x))~=1 || min(size(y))~=1)
        error(['les arguments doivent être des vecteurs']);
    elseif (length(y) ~= n)
        error(['Les vecteurs X & Y doivent avoir la même taille']);
    elseif (min( x(2:n)-x(1:n-1) ) <= 0)
        error(['X(i) doit être un vecteur strictement croissant']);
    else
        % on manipule vecteurs-colonnes
        if(size(x,2)~=1) x = x'; end;
```

```
        if(size(y,2)~=1) y = y'; end;
           % h(i) vaut x(i+1)-x(i)
        h = x(2:n) - x(1:n-1);
           % mu, lb
        hm = h(1:n-2);
        hp = h(2:n-1);
        mu = hm./(hm + hp); mu(n-1) = 1/2;
        lb(2:n-1) = hp./(hm + hp); lb(1) = 1/2;
        d(2:n-1) = 6./(hm + hp).* ...
            ((y(3:end)-y(2:end-1))./hp-(y(2:end-1)-y(1:end-2))./hm);
        d(1) = d(2)/2; d(n) = d(n-1)/2;
        d = d';
           % détermination des parametres
        A = 2*diag(ones(n,1)) + diag(mu,-1) + diag(lb,1);
        u = A\d; % <=
        s(:,1) = u(1:n-1)./h./6;
        s(:,2) = u(2:n)./h./6;
        s(:,3) = (y(2:n)-y(1:n-1))./h-(u(2:n)-u(1:n-1)).*h./6;
        s(:,4) = y(1:n-1)-u(1:n-1).*h.*h./6;
    end
```

```
function fo = spl_eval(xo,x,s)
% FO = spl_eval(XO,X,S)
%    la fonction renvoie la valeur FO d'un polynôme cubique
%    par morceaux qui est défini par les N abscisses
%    X(1), ..., X(N) et les parametres S(i,:) du polynome
%    entre les deux abscisses X(i) et X(i+1) successives
%
% arguments:
%    XO- l'abscisse où on veut évaluer le polynôme cubique
%        par morceaux; XO doit être entre X(1) et X(N)
%    X - le vecteur de taille N x 1 qui contient
%        les points d'interpolation; on demande à ce que
%        N > 1 et X(1) < X(2) < ... < X(N)
%    S - S(i,1), S(i,2), S(i,3) et S(i,4), i=1,...,N - 1
%        sont les parametres qui definissent le polynôme
%        d'interpolation entre les abscisses X(i) et X(i+1)
%
% sortie:
%    FO- valeur d'un polynôme cubique par morceaux
%        au point XO
%
    n = length(x);
    if (n < 3)
        error(['Le nombre d"éléments de X doit être > 2']);
    elseif (min(size(x))~=1)
        error(['X doit être un vecteur']);
    elseif (size(s,1)~=n-1)
        error(['La matrice S doit avoir ',int2str(n-1),' lignes']);
    elseif (size(s,2)~=4)
        error(['La matrice S doit avoir 4 colonnes']);
```

```
elseif (xo < x(1) || xo > x(n))
    error(['XO doit être entre X(1) et X(N)']);
elseif (min( x(2:n)-x(1:n-1) ) <= 0)
    error(['X(i) doit être un vecteur strictement croissant']);
else
    i = min(max( find(x <= xo) ), n - 1);
    fo = s(i,1)*(x(i+1)-xo)^3 + s(i,2)*(xo-x(i))^3 + ...
         s(i,3)*(xo-x(i)) + s(i,4);
end
```