Corrigé de la séance 5

Comme tous les corrigés des séances d'exercices le corrigé de la séance 5 est partiel. Vous trouverez ci-dessous quelques programmes qui répondent en partie aux questions posées dans l'énoncé de la séance et qui fournissent éventuellement des résultats supplémentaires. Ces programmes n'ont pas la prétention d'être optimaux et de répondre aux canons de l'informatique...

Toute information complémentaire peut-être obtenue soit auprès de la personne qui encadre les travaux pratiques du groupe auquel vous appartenez, soit en adressant vos demandes par courriel (au moyen de votre adresse ulb) à mtolley@ulb.ac.be, soit aux "intercours" ou lors des permanences organisées "à la carte" (cf. vos délégués d'année).

NB Ue est la solution analytique du problème

```
clc;clear
L=2; k=1; T=1;
f=@(x,k,T,L) -k*x.*(L-x)/T;
for N=5:5:25
h=L/N;
x=linspace(0,L,N+1);
A=-2*eye(N-1);
for i=1:N-2
    A(i,i+1)=1;
    A(i+1,i)=1;
end
b=h^2*f(x(2:N),k,T,L);
u=A\b';
U=[0 u' 0];
plot(x,U,'-o','LineWidth',1.5)
hold on
pause
end
z=linspace(0,L,501);
Ue=k*z.*(z.^3-2*L*z.^2+L^3)/(12*T);
plot(z,Ue,'r-','LineWidth',2.5)
xlabel('x','FontSize',14)
```

```
ylabel('u_x','FontSize',14)
title('Déformée d''une corde','FontSize',14)
% texte=['N = ' num2str(N)];
% gtext(texte,'FontSize',14)
pause
u=Gauss(A,b);
U=[0 u 0];
plot(x,U,'-ok','LineWidth',2.5)
pause
x0=b'; tol=1e-5;
for max= 200:200:1000
[u,Nb_iter] = jacobi_syst_alg(A,b,x0,tol,max);
U=[0 u' 0];
plot(x,U,'-om','LineWidth',2.5)
pause
end
plot(z,Ue,'r-','LineWidth',2.5)
```

Méthode directe - Elimination de Gauss - Nécéssite la "back-substitution"

```
function [x]=Gauss(A,b)
N=length(b);
if size(A)~=[N N]
   error('Dimensions non compatibles')
end
for k=1:N
    if A(k,k)==0
    error('Permuter cette ligne avec une autre qui suit...')
    end
    for i=k+1:N
        m(i,k)=A(i,k)/A(k,k);
        b(i)=b(i)-m(i,k)*b(k);
        for j=k+1:N
            A(i,j)=A(i,j)-m(i,k)*A(k,j);
        end
    end
end
x=trisup(A,b);
```

Back-substtution

```
function [x]=trisup(A,b)
N=length(b);
if size(A)~=[N N]
   error('Dimensions non compatibles')
```

```
end
if A(N,N)==0
   error('Déterminant nul')
end
x(N)=b(N)/A(N,N);
for k=N-1:-1:1
    if A(k,k)==0
        error('Déterminant nul')
    end
    sum=0;
    for j=k+1:N
        sum=sum+A(k,j)*x(j);
    end
    x(k)=(b(k)-sum)/A(k,k);
end
```

Méthode itérative de Jacobi

```
function [x,Nb_iter] = jacobi_syst_alg(A,b,x0,tol,max)
% Résolution approchée du système algébrique Ax = b
% Méthode itérative de Jacobi
% Entrées
% A : matrice du système (NxN)
% b : second membre du système (Nx1)
% x0 : vecteur initial (Nx1)
% tol : tolérance sur deux approximations successives --> STOP !
% max : nombre d'itérations maximum --> STOP !
% Sorties
% x : solution approchée (Nx1)
% Nb_iter : nombre d'itérations effectuées
[n m] = size(A);
if n ~= m
    display('La matrice n''est pas carrée')
    return
end
m = size(b);
if n ~= m
    display('Dimensions de A et b incompatibles')
    return
end
m = size(x0);
if n ~= m
    display('Dimension de A et x0 incompatibles')
    return
end
```

```
xold = x0;
C = -A;
for i=1:n
    C(i,i)=0;
end
for i=1:n
    C(i,:) = C(i,:)/A(i,i);
end
for i=1:n
    d(i,1) = b(i)/A(i,i);
end
Nb_iter = 1;
disp('           iter        x1          x2         x3       ...')
while Nb_iter <= max
    xnew = C*xold + d;
    if norm(xnew-xold) <= tol
        x = xnew;
        disp(['La méthode de Jacobi a convergé après ' num2str(Nb_iter) ' itérations
        return
    else
    xold = xnew;
    end
disp([Nb_iter xnew'])
Nb_iter = Nb_iter + 1;
end
disp('La méthode de Jacobi n''a pas convergé')
disp('Le nombre maximum d''itération est atteint')
x = xnew;
```