ALGORITHMIQUE NUMERIQUE
TP N°1 : Résolution des systèmes linéaires

AN – L3
Jan 2011
D. Achvar

Position d'un problème :

Afin d'illustrer les méthodes de résolution des systèmes et cerner leurs limites dans des situations concrètes de calculs scientifiques, nous nous plaçons dans un environnement multi-sensoriel où le problème consiste à distinguer les informations issues de n sources observables au moyen de n capteurs.

Nous montrons sur les figures ci-dessous deux manières de disposer les capteurs relativement aux sources. Dans tous les cas nous admettons que les signaux issus des capteurs s'expriment vectoriellement par une équation de la forme : Y=A.X où X est le vecteur sources, Y le vecteur capteur, et A est une matrice $n_x n$…

Représentation d'un environnement multi-sensoriel pour n=5

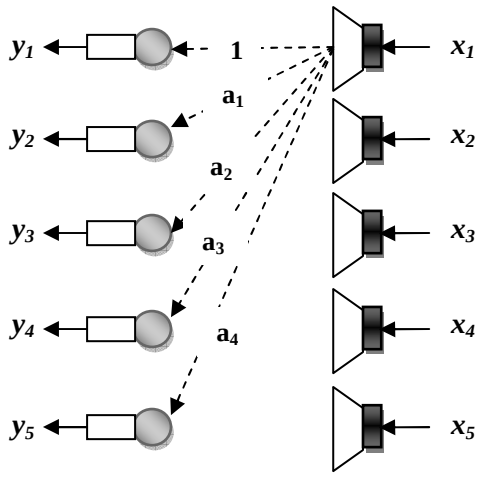

Figure 1

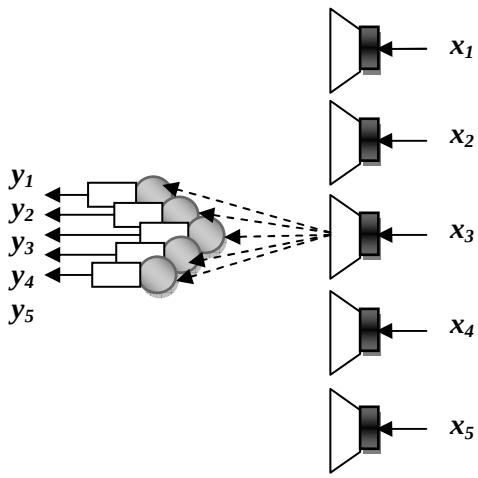

Figure 2

Et pour toutes les applications numériques :

```
A1=[1.0000   0.4472   0.2425   0.1644   0.1240;
    0.4472   1.0000   0.4472   0.2425   0.1644;
    0.2425   0.4472   1.0000   0.4472   0.2425;
    0.1644   0.2425   0.4472   1.0000   0.4472;
    0.1240   0.1644   0.2425   0.4472   1.0000]

Y1=[0.8;
    0.9;
    1.0;
    0.9;
    0.8] ;
```

```
A2=[0.2474   0.4562   0.9800   0.4383   0.2377;
    0.2450   0.4517   0.9900   0.4427   0.2401;
    0.2425   0.4472   1.0000   0.4472   0.2425;
    0.2401   0.4427   0.9900   0.4517   0.2450;
    0.2377   0.4383   0.9800   0.4562   0.2474]

Y2=[1;
    1;
    1;
    1;
    1] ;
```

Objectifs du TP

Ce simple (et très vieux) problème revient à l'inversion de la matrice A dans le but de remonter au vecteur X et va nous permettre de faire un point expérimental sur :

- ***Le conditionnement d'un système linéaire et stabilité.***
- ***La résolution par la méthode du pivot de Gauss.***
- ***La résolution par les méthodes itératives.***

Modalités du TP

1. ***Remettre un compte-rendu à la fin de la séance.***
2. ***Ne rédiger que les questions affectées d'un barème***

I. CONDITIONNEMENT ET STABILITE D'UN SYSTEME

Nous étudions en premier lieu quelques outils programmés de MATLAB dans le but de mettre en évidence l'importance du conditionnement d'un problème. L'équation $A_1.X_1=Y_1$ *sera donc tout simplement résolue par inversion de la matrice* A_1 *dans MATLAB puisque le vecteur source inconnu s'écrit :* $X_1=A_1^{-1}.Y_1$.

INVERSION D'UNE MATRICE

Q1. Le petit programme ci-contre (StabA1 : disponible en ligne) simule et résout le système linéaire $A_1.X_1=Y_1$ de la figure 1 par inversion de la matrice A_1.

- Exécuter ce programme et relever sa sortie X1.
- Relever la valeur des variables N1, M1, C1 et D1.
- A quoi correspondent ces grandeurs ?
- Examiner et mesurer l'influence d'une variation de 1% du coefficient $A_1(2,2)$ ($A_1(2,2) = A_1(2,2) * 0.99$) sur le calcul de x. Commentaires.
  (Cette variation de 1% pourrait simuler par exemple le vieillissement d'un capteur, ses dérives en température …)

```
 9 -  clc; clf;
10 -  x=zeros(5,1);
11
12 -  A1=[1 a1 a2 a3 a4;
13        a1 1 a1 a2 a3;
14        a2 a1 1 a1 a2;
15        a3 a2 a1 1 a1;
16        a4 a3 a2 a1 1]  ;
17
18 -  Y1=[0.8;0.9;1;0.9;0.8];
19
20 -  N1=norm(A1)
21 -  M1=norm(inv(A1))
22 -  C1=cond(A1)
23 -  D1=det(A1)
24 -  x1=inv(A1)*Y1
```

StabA1 :
Résolution par inversion d'une matrice

CAPTEURS DISPOSES AUTREMENT

3 pts

Q2. Reprendre cette étude dans le cas de la figure 2 avec le système linéaire $A_2.X_2=Y_2$ programmé dans Stab2 (disponible en ligne).

- Justifier qualitativement les avertissements (*Warning*) générés par la fenêtre de commande de MATLAB :

```
Command Window
Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate. RCOND = 3.372829e-018.
```

- Quel est le système le mieux conditionné ?

II. RESOLUTION PAR LA METHODE DU PIVOT DE GAUSS

PHASE DE TRIANGULATION

Q3. Nous rappelons avec l'encadré ci-contre l'algorithme de triangulation d'un système de dimension n.

- Exécuter ce programme et relever le nouveau système $\mathbf{A_t.x=b_t}$ à la fin de la transformation de triangulation.
- Résoudre en premier lieu le système par simple inversion de $\mathbf{A_t}$ ($x=A_t^{-1}.b_t$). Comparer le résultat à celui obtenu en Q1.

```
26    % ---------- triangulation -----------
27 -  n=5; n1=0;n2=0;n3=0;
28 -  for k=1:n-1
29 -      n1=n1+1;
30 -      for i=k+1:n
31 -          Lambda=A(i,k)/A(k,k); n2=n2+1; i;
32 -          for j=k:n
33 -              A(i,j)=A(i,j)-Lambda*A(k,j);
34 -              n3=n3+1; j;
35 -              At=A
36 -          end
37 -          Y(i)=Y(i)-Lambda*Y(k); Yt=Y
38 -      end
39 -  end
40    % ------------------------------------
```

3 pts

Q4. Nous avons établit en cours la complexité théorique de l'algorithme de triangulation.

- Relever la valeur finale des variables n1, n2 et n3.
- Combien d'itérations ont été nécessaires pour évaluer la matrice $\mathbf{A_t}$ et le vecteur colonne $\mathbf{Y_t}$?
- Comparer ces résultats à l'ordre de complexité de l'algorithme de triangulation (en $n^3/3$; cf. cours).

PHASE DE REMONTEE

Q5. La résolution de la dernière ligne du système $\mathbf{A_t.X=Y_t}$ est triviale. Il s'agit maintenant de résoudre le système complet en le remontant ligne par ligne au moyen de l'algorithme ci-contre.

- Exécuter cet algorithme et relever sa sortie dans le vecteur X.
- Relever et commenter la valeur finale de n4.
- Combien d'itérations ont été finalement nécessaires à la résolution du système par l'algorithme du pivot?

3 pts

```
51    % ------------ Résolution------
52 -  X=Y; n4=0;
53 -  for k=n:-1:1
54 -      for j=k+1:n
55 -          X(k)=X(k)-A(k,j)*X(j);
56 -          n4=n4+1;
57 -      end
58 -      X(k)=X(k)/A(k,k);
59 -  end
```

PROBEMES ET SOLUTIONS

Q6. Considérons maintenant le système **A'.X=Y'** dans lequel **A'** est une matrice ne différant de **A** que d'un coefficient.

3 pts

- Résoudre le système par inversion de la matrice **A'**.
- Examiner la stabilité du système en notant la valeur de det(**A'**) et de cond(**A'**).
- Résoudre le système en employant l'algorithme du pivot. Quelle est l'origine des erreurs générées ici ?
- Corriger l'algorithme du pivot de Gauss.

```
Ap=[0 a1 a2 a3 a4;
    a1 1 a1 a2 a3;
    a2 a1 1 a1 a2;
    a3 a2 a1 1 a1;
    a4 a3 a2 a1 1]  ;

Y=[0.8;0.9;1;0.9;0.8];
```

II. RESOLUTION PAR LES METHODES ITERATIVES

DECOMPOSITION DE LA MATRICE A

Q7. Les algorithmes étudiés sont basés sur une répartition particulière de la matrice **A** de sorte que : **A= D – (U+L)**.

- Vérifier cette décomposition de **A** en relevant les matrices générées par le petit programme ci-contre.

La résolution du système sera alors assurée par la convergence d'une suite de vecteurs x_k vers une limite définie par une récurrence de la forme :

$$x_k=M^{-1}.(N.x_{k-1}+b)$$

M et N étant deux matrices fonction de A, de L et de U.

```
% ------ Décomposition de A -------
n=5; U=zeros(n,n); L=zeros(n,n);
     D=zeros(n,n); UL=zeros(n,n);

for i=1:n
    D(i,i)=A(i,i); iD(i,i)=1/D(i,i);
    for j=1:n
    if (j>i) U(i,j)=-A(i,j); end
    end
end
L=D-U-A; UL=U+L;    %----- UL=D-A;
```

ALGORITHME DE JACOBI

Q8. L'algorithme de résolution du système **A.X=Y** par la méthode de Jacobi est illustré ci-contre.

3 pts

- Rappeler l'expression des matrices **M** et **N** dans ce cas.
- Exécuter ce programme et noter la convergence (ou la non-convergence) de l'algorithme de Jacobi dans le cas particulier de notre étude.
- Justifier ce résultat après avoir calculé les valeurs propres de la matrice $\mathbf{M.N^{-1}}$ au moyen de la fonction **eig**($\mathbf{M.N^{-1}}$) .

```
% ------------ Jacobi -------------
n=5;
e=zeros(n,1); XO=zeros(n,1);
erreur=zeros(1,45); nj=0; eps=1;
while (eps>0.01)
    nj=nj+1; XO=X;
    X=iD*(UL*X+Y);
    eps=((norm(X-XO))/norm(X));
    erreur(nj)=eps;
end
Xja=X
% ---- Représentions graphiques ----
figure(1);subplot(211); plot((erreur));
xlabel('itérations'),ylabel('erreur');
grid; title('Algo de Jacobi: Convergence');
```

ALGORITHME DE GAUSS-SEIDEL

Q9. Nous présentons au moyen de l'encadré ci-contre une premier approche de la méthode de Gauss-Seidel dans le but de marquer sa différence avec celle de Jacobi.

3 pts

- Quelle est la nouvelle expression des matrices **M** et **N** ?
- Exécuter ce programme et relever sa solution. Justifier sa convergence après avoir calculé les valeurs propres de la matrice ($\mathbf{M.N^{-1}}$).
- Combien d'itérations ont été nécessaires à ces calculs ?
- Relever la solution du système. Quelle est la précision atteinte ?
- Comparaison avec la méthode algébrique de Gauss.

```
% ------- Gauss Seidel ppe théorique -------
n=5; X=zeros(n,1);
e=zeros(n,1); XO=zeros(n,1);
erreur=zeros(1,45); ngs=0; eps=1;
while (eps>0.01)
    ngs=ngs+1;
    XO=X; X=inv(D-L)*(U*X+Y);
    eps=((norm(X-XO))/norm(X));
    erreur(ngs)=eps;
end
% ------- Représentations graphiques ------
subplot(212); plot(erreur);grid;
xlabel('itérations'),ylabel('erreur');
title('Gauss-Seidel: Convergence théorique');
```

2 pts

Q10. Montrer que nous pouvons éviter l'inversion de la matrice (D-L) en remplaçant la boucle *while* du programme précédant par celle qui est représentée ci contre.

- Exécuter ce programme et comparer son résultat à celui obtenu en Q_9.

```
while (eps>0.01)
    ngs=ngs+1; XO=X;
    for i=1:n
        somme=0;
        for j=1:n
            somme=somme-UL(i,j)*X(j)
        end
        X(i)=(Y(i)-somme)/A(i,i);
    eps=((norm(X-XO))/norm(X));
    erreur(ngs)=eps;
    end
end
Xgs=X
```