Université de Picardie Jules Verne
Master 2 AAM
Calcul Scientifique
2019-2020

Factorisations creuses et incomplètes

1 Commandes Scilab

1.1 Factorisation creuses - Cholesky[1]

- `spchol` : `[R,P]=spchol(X)`
  X : est la matrice SDP à factoriser
  P : est la matrice de permutation
  R : est le facteur de Cholesky
  Cette commande permet de calculer le facteur creux de Cholesky R et la matrice de permutation P. LE système $Ax = b$ se résout alors par : `[R,P] = spchol(A);`
  `x = P*(R'\(R\(P'*b)));`
  Cette commande est associée à :

  - `chsolve`: `x=chsolve(hand,b)`
    Cette commande permet de résoudre le système $R * R^T x = b$ avec la décomposition creuse construite par `chfact`

- Pour rappel `chol` effectue la factorisation sans permutation.

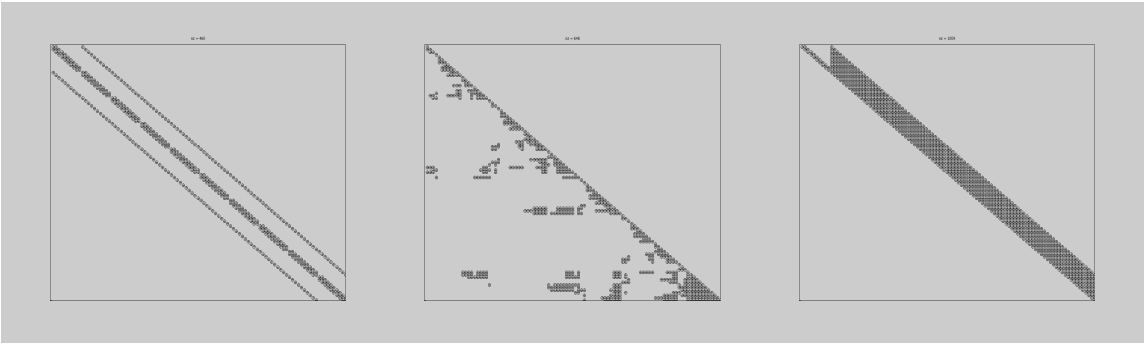

Figure 1: De gauche droite : la matrice A (`nnz=460`), le facteur de Cholesky avec permutation (`nnz=648`), le facteur de Cholesky sans permutation (`nnz=1009`). Utilisation de `spchol` et de `chsolve` cf programme en Annexe

[1]Pour une histoire de Cholesky, voir l'article de la revue d'histoire des mathématiques RHM de C. Brezinski 11, (2005), pp 205-238 `http://www.numdam.org/article/RHM_2005_11_2_205_0.pdf/`

1.2 Factorisation creuses et incomplètes - LU

- `umf_fact` : `umf_fact(A)` calcule les facteurs creux de la décomposition LU de A.
  Cette commande est associée à :

  - `umf_get` : `[L,U,p,q,R]=umf_get(A)`
    A : matrice creuse carrée
    L,U : les facteurs LU creux
    p,q : vecteurs de permutations
    Rd : vecteur des facteurs d'échelle

    Cette commande permet d'extraire les facteur creux L et U de la factorisation opérée par `umf_fact`

  - `lusolve`: `x=umf_lusolve(hand,b)`
    Cette commande permet de résoudre le système $LUx = b$ avec la décomposition creuse construite par `umf_ufact`

```
Lup = umf_fact(A);
x = umf_lusolve(Lup,b)
```

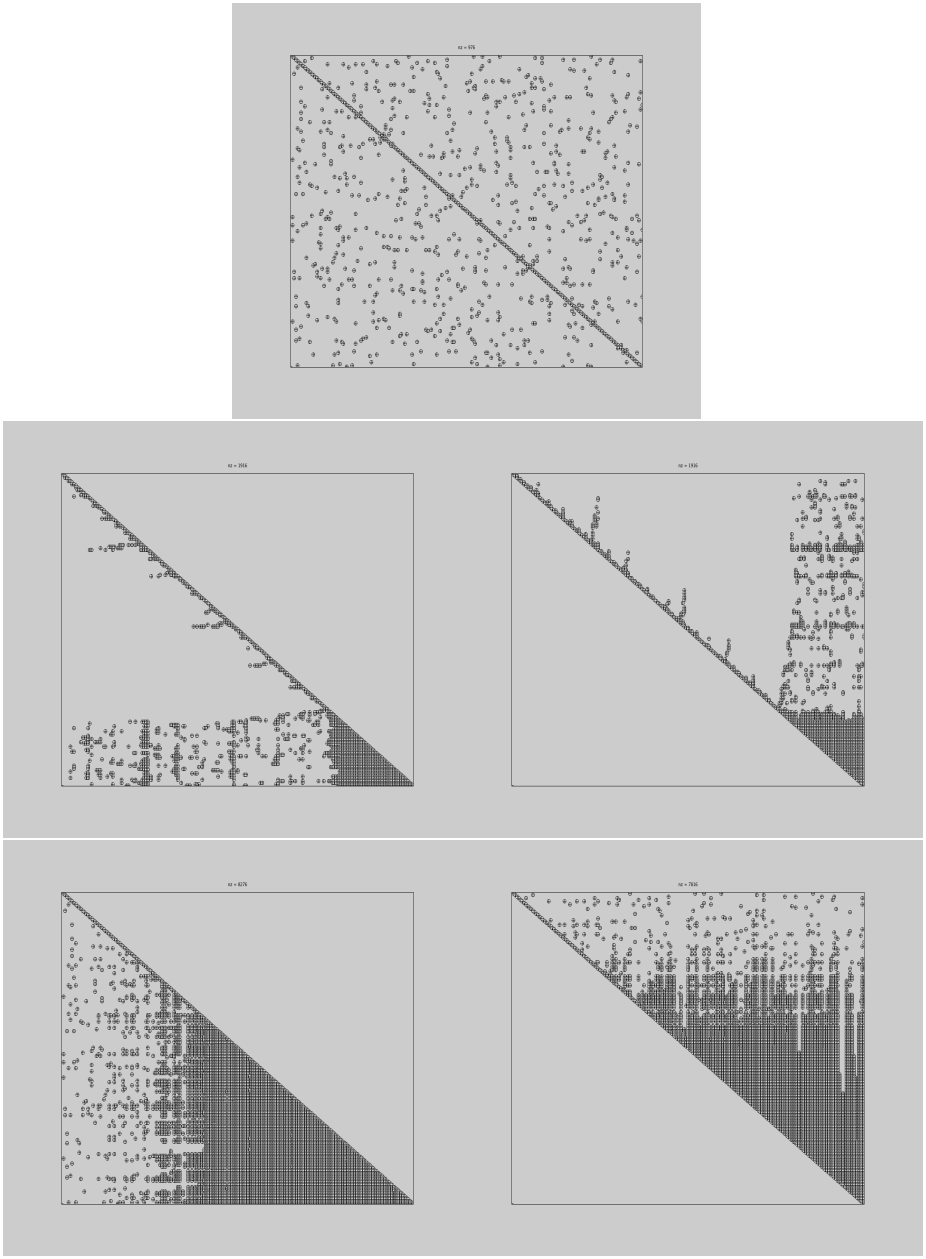

Figure 2: De gauche droite : la matrice A (`nnz=976`), les facteurs de LU avec permutation (`nnz=1916`), les facteur LU sans permutation (`nnz=8276`), cf programme en Annexe

2 Factorisations creuses et incomplète et premières applications

Exercice 1 (Cholesky) *Soit la matrice A définie par les commandes :*

```
A=sprand(N,N,0.1);
A=A+A';
A=1.1*norm(A,'inf')*Id+A;
A=sparse(A);
```

1. *Montrer que A est symétrique et définie positive.*
2. *Construire les facteurs de Cholesky S et R respectivement à l'aide des commandes* `chol` *, puis* `spchol`*, pour différentes valeurs de n. On comparera à chaque fois le rapport* $\frac{\texttt{nnz(R}}{\texttt{nnz(S)}}$.
3. *Tracer le rapport* $\tau = \frac{\texttt{nnz(R)}}{\texttt{nnz(S)}}$ *en fonction de n. Conclusion ?*

Exercice 2 (LU) *Soit la matrice A définie par les commandes :*

```
   A=sprand(N,N,0.1);
A=1.1*norm(A,'inf')*Id+A;
A=sparse(A);
```

1. *Montrer que A est définie positive.*
2. *Construire les facteurs LU plein et creux de A. On comparera à chaque fois le rapport* $\frac{\texttt{nnz(R}}{\texttt{nnz(S)}}$.
3. *Tracer le rapport* $\tau = \frac{\texttt{nnz(Lcreux)}}{\texttt{nnz(Lplein)}}$ *en fonction de n. Conclusion ?*

3 LU Incomplet

On considère 3 versions autoécrasantes de la factorisation LU

- Factorisation exacte
  $i, j, k : \ a_{i,j} = a_{i,j} - \frac{a_{i,k}a_{k,j}}{a_{k,k}}$
- Factorisation approchée
  $i, j, k : \text{Si}(i, j) \in \mathcal{S} \ a_{i,j} = a_{i,j} - \frac{a_{i,k}a_{k,j}}{a_{k,k}}$
- Réduire le stockage à travers la définition de $\mathcal{S}$ (via $\#\mathcal{S}$)
- Ne calculer le coefficient que si $\frac{a_{i,k}a_{k,j}}{a_{k,k}}$ est assez grand

ILU(0)

On définit la factorisation ilu(0) avec $\mathcal{S} = \{(i, j)A_{i,j} \neq 0\}$
On a

$$L = (D - E)D^{-1} \text{ et } U = (D - F)$$

Exercice 3 (Approximation de la matrice ou de son inverse par produit de facteurs) *Soit A la matrice pentadiagonale habituelle de discrétisation de $-\Delta$ sur le carré unité.*

1. *Calculer les facteurs L_0 et U_0 ilu(0) de A.*
2. *Construire $R = A - L_0 * U_0$ et calculer sa norme.*
3. *Comparer R^{-1} et A^{-1}.*

Exercice 4 (Application au préconditionnement de méthodes itératives Classiques) *Soit A la matrice pentadiagonale habituelle de discrétisation de $-\Delta$ sur le carré unité.*

1. *Calculer les facteurs L_0 et U_0 ilu(0) de A.*
2. *Programmer la méthode itérative classique avec $M = L_0 * U_0$. La comparer avec celles classiques (Jacobi, Gauss-Seidel).*
3. *Qu'observez-vous ?*

ILU(1)

On note $A_0 = L_0 * U_0$. La factorisation incomplète ilu(1) consiste à appliquer ILU(0) à A_0.

Plus généralement, si $A_0 = L_0 * U_0$, alors
$[L_k, U_K] = ilu(k, A)$, s'obtient en appliquant ilu(0) avec le profil de $A_{k-1} = L_{k-1}U_{k-1}$.

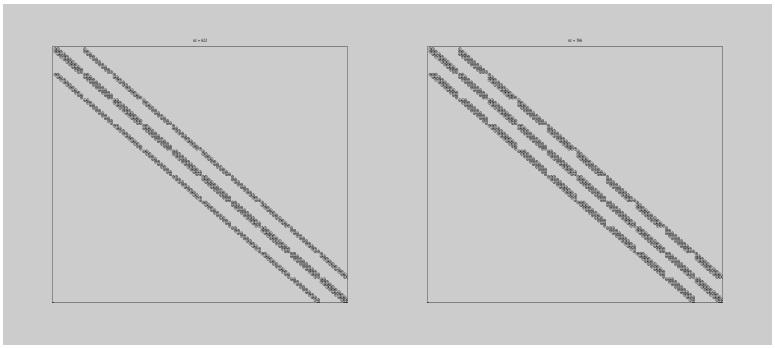

Figure 3: De gauche droite : les coefficients non nuls des matrices $A_0 = L_0 * U0$ (`nnz=460`) et $A_1 = L_1 * U_1$

3.1 Factorisation incomplète à l'aide d'un point fixe[2]

On part de a relation

$$A_{i,j} = \sum_{k=1}^{min(i,j)} L_{i,k}U_{k,j}$$

qui donne

$$L_{i,j} = \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k}U_{k,j} \right) / U_{k,k}, i > j$$
$$U_{i,j} = \left(A_{i,j} - \sum_{k=1}^{i-1} L_{i,k}U_{k,j} \right), i \leq j$$

et l'on obtient la méthode de point fixe $L^0 = (D - E)D^{-1}$, $U^0 = D - F$.

$$L_{i,j}^{m+1} = \left(A_{i,j} - \sum_{k=1}^{j-1} L_{i,k}^m U_{k,j}^m \right) / U_{j,j}^m, i > j$$
$$U_{i,j}^{m+1} = \left(A_{i,j} - \sum_{k=1}^{i-1} L_{i,k}^m U_{k,j}^m \right), i \leq j$$

Exercice 5 *Programmer cette méthode. Que constatez-vous ?*

[2]Updating Incomplete Factorization Preconditioners for Model Order Reduction par Hartwig Anzt, Edmond Chow, Jens Saak et Jack Dongarra. Article paru dans Numerical Algorithm, November 2016, Volume 73, Issue 3, pp 611–630

4 Annexe

4.1 Factorisation Creuse de Cholesky en Scilab

```
clear all
clf()
exec("spy.sci");
//Construction de la matrice
N=10;
h=1/(N+1);//pas de discrétisation en espace
x=h:h:1-h;
y=h:h:1-h;
[X,Y]meshgrid(x,y);
Axx=2*eye(N,N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
//Axx=sparse(Axx);
Id=eye(N,N); //Matrice identit
//construction de la matrice en dimension 2
// A est de taille NxN par NxN
A=kron(Id,Axx)+kron(Axx,Id);// - d^2/dx^2-d^2/dy^2
h=1/(N+1);
A=A/h^22;
A=sparse(A);

[R,P] = spchol(A);
[S]=chol(full(A));
rhs =ones(N*N,1);
sol = P*(R'\(R\(P'*rhs)));

UM=matrix(sol,N,N);
figure(1)
surf(X,Y,UM)
figure(2)
spy(A)
figure(3)
spy(S)
figure(4)
spy(R)
```

4.2 Factorisation LU creuse en Scilab

```
clear all
clf()
exec("spy.sci");
//matrice
N=200;
Id=eye(N,N); //Matrice identit
//construction de la matrice
// A est de taille N par N
A=sprand(N,N,0.01); A=A+A';
A=1.1*norm(A,'inf')*Id+A;
A=sparse(A);
[spLU]=umf_lufact(A);//,prec);
rhs =ones(N,1);
sol=umf_lusolve(spLU,rhs);
[L,U,p,q,R] = umf_luget(spLU);
[Lf,Uf]=lu(full(A));
figure(1)
spy(A)
figure(2)
spy(L)
figure(3)
spy(U)
figure(4)
spy(Lf)
figure(5)
spy(Uf)
```

4.3 LU autoécrasant

```
function [L,U]=LUauto(A)
n=size(A,1);
for k=1:n-1
for i=k+1:n
A(i,k)=A(i,k)/A(k,k);
for j=k+1:n
A(i,j)=A(i,j)-(A(i,k))*A(k,j);
end
end
end
L=eye(n,n)+tril(A,-1);
U=triu(A);
endfunction
```