

Systèmes linéaires avec Scilab - quelques commandes

Contents

1	Calcul matriciel avec Scilab	2
1.1	Matrices particulières	2
1.2	Structures	2
1.3	Produits	2
1.4	Normes, spectre et rayon spectral	2
1.5	Quelques fonctions	3
1.6	Réductions	3
2	Méthodes directes	3
3	Méthodes Itératives	4
3.1	Résidus	4
3.2	Matrice creuses	4
3.3	Méthodes classiques	5
4	Annexe	6
4.1	Disques de Gershgorin	6
4.2	Construction des matrices	7
4.3	Résolution et représentation des solutions	7

1 Calcul matriciel avec Scilab

1.1 Matrices particulières

- Matrice simple avec coefficients rentrés au clavier : `A=[1,2;3,4]`
- Matrice $n \times m$ à coefficients suivant une loi uniforme sur $[0, 1]$: `rand(n,m)`
- Matrice identité $n \times n$: `eye(n,n)` ou plus généralement `eye(n,m)`
- Matrice composée uniquement de 0 : `zeros(n,n)` ou plus généralement `zeros(n,m)`
- Matrice composée uniquement de 1 : `ones(n,n)` ou plus généralement `ones(n,m)`

1.2 Structures

- Partie triangulaire inférieure de A : `liu(A)`
- Partie triangulaire supérieure de A : `triu(A)`
- Partie diagonale de A : `diag(diag(A))`

1.3 Produits

Soient u et v deux vecteur colonnes de même taille ; u' est le vecteur (ligne) transposé de u .

- `u'*v` : produit scalaire de u par v
- `u*v'` : produit extérieur de u par v
- `cross(u,v)` : produit vectoriel de u par v (en dimension 3)

Soient A et B deux matrices carrées de même taille ; A' est la matrice transposé de A .

- `A*B` : produit de A par B
- `A.*B` : produit d'Hadamard de A par B , i.e., composante par composante

1.4 Normes, spectre et rayon spectral

- Norme 2 de A : `norm(A)` ou `norm(A,2)`
- Norme 1 de A : `norm(A,1)`
- Norme infinie de A : `norm(A,'inf')`
- Norme de Frobenius de A : `norm(A,'fro')`

1.5 Quelques fonctions

Soit A une matrice carrée

- Déterminant de A : `det(A)`
- Polynôme caractéristique de A : `p=poly(A, 's')`
- Racines de p : `roots(p)`
- Rang de A : `rank(A)`
- Noyau de A : `kernel(A)`
- Conditionnement de A : `cond(A)`
- Extraction d'une base orthonormée des colonnes de A : `orth(A)`
- Trace de A : `trace(A)`
- Inverse de A : `inv(A)`
- Valeur minimale des coeffs de A : `min(min(A))`
- Valeur maximale des coeffs de A : `max(max(A))`

1.6 Réductions

- Décomposition de Schur de A : `[T,U]=schur(A)` Test la matrice triangulaire supérieure et U la matrice unitaire.
- Valeurs propres de A : `spec(A)`
- Vecteurs et valeurs propres de A : `[R,diagevals]=spec(A)`, R contient les vecteurs propres et `diagevals` la matrice diagonale associée
- Décomposition en valeurs singulières : `svd(A)`

2 Méthodes directes

- Partie triangulaire supérieure : `triu(A)`
- Partie triangulaire inférieure : `tril(A)`
- Factorisation de Cholesky : `[R]=chol(A)`
- Factorisation de LU : `[R]=lu(A)`
- Factorisation QR : `[Q,R]=qr(A)`
- Moindres carrés : `lsq(A)`

- Calcul du temps de calcul

```
tic
:
instructions
:
toc
```

3 Méthodes Itératives

3.1 Résidus

- Comparaison de l'historique des résidus

```
plot(it1,log(R1)/log(10))
plot(it2,log(R2)/log(10))
drawnow
```

3.2 Matrice creuses

- Matrice creuse

Convertir une matrice pleine en matrice creuse

```
A=sparse(X)
```

Convertir une matrice creuse en matrice pleine

```
A=full(X)
```

- Opérations sur les matrices creuses

- `sprand` : matrice creuse aléatoire
- `speye` : matrice identité creuse
- `spones` : matrice de 1 creuse
- `spzeros` : matrice nulle
- `nnz` : nombre d'éléments non nuls
- `lufact` : factorisation LU d'une matrice creuse
- `lusolve` : solveur LU de système linéaire creux
- `spchol` : factorisation de Cholesky d'une matrice creuse
- `chsolve` : solveur de Cholesky de système linéaire creux
- `spget` : entrées non nulles d'une matrice creuse

3.3 Méthodes classiques

On rappelle que les méthodes itératives classiques peuvent se programmer comme suit :

Algorithm 1 Méthode itérative classique

```
1:  $x^{(0)}$  donné dans  $\mathbb{R}^n$ 
2:  $k = 0$ ,  $r^{(0)} = b - Ax^{(0)}$ 
3: while  $k < Kmax$  &  $\|r^{(k)}\| > \epsilon$  do
4:   Résoudre :  $Mz^{(k)} = r^{(k)}$ 
5:   Poser :  $x^{(k+1)} = x^{(k)} + z^{(k)}$ 
6:   Poser :  $r^{(k+1)} = r^{(k)} - Az^{(k)}$ 
7:   Poser :  $k = k + 1$ 
8: end while
```

4 Annexe

4.1 Disques de Gershgorin

```
// -----  
//JPC - Prepa Agreg  
// Disques de Gershgorin  
// -----  
clear all;  
clf;  
n=5;  
un=ones(n,1);  
A=2*(1-2*rand(n,n));  
SP=spec(A);  
// ----- representation des valeurs propres dans le plan complexe  
plot(real(SP),imag(SP),'*b')  
for i=1:n  
Xa=real(A(i,i));  
Ya=imag(A(i,i));  
R(i)=abs(A(i,:))*un-abs(A(i,i));  
// ----- representation des disques de Gershgorin dans le plan complexe  
X100=Xa+R(i)*cos (2*%pi /100* [ 0 : 100] )';  
Y100=Ya+R(i)*sin (2* %pi /100* [ 0 : 100] )';  
plot(X100,Y100)  
drawnow  
end
```

4.2 Construction des matrices

- Matrice de $-\frac{\partial^2}{\partial x^2}$ en dimension 1
N=20; //nombre de points de discrétisation
h=1/(N+1); //pas de discrétisation en espace
x=h:h:1-h;
//matrice
A=2*eye(N,N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
A=A/h^2;
- Matrice de $-\Delta$ en dimension 2
N=100; //nb de points de discrétisation
h=1/(N+1); //pas de discrétisation en espace
x=h:h:1-h;
y=h:h:1-h;
[X,Y]=meshgrid(x,y); // maillage
//*****
//matrice
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 identité
//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
A/h^2;
- Matrice de $-\Delta$ en dimension 3
N=4; //nb de points de discrétisation
h=1/(N+1); //pas de discrétisation en espace
x=h:h:1-h;
y=h:h:1-h;
z=h:h:1-h;
[X,Y,Z]=meshgrid(x,y,z);

Axx=2*eye(N,N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
Axx=sparse(Axx);
Id=speye(N,N); //Matrice identité
A3=kron(Id,kron(Id,Axx))+kron(Id,kron(Axx,Id))+kron(Axx,kron(Id,Id));
- La commande `nnz(A)` retournera le nombre d'éléments de A non nuls.

4.3 Résolution et représentation des solutions

- En dimension 1
F=ones(N,1);
//Résolution du systeme Au=F
//

```

U=A\ F;
//representation graphique de la solution
plot(x,U)

```

- En dimension 2

```

F=ones(N*N,1);//vecteur second membre
//
//Résolution du systeme Au=F
//
U=A\ F;
//representation graphique de la solution - le données sont rangées sous forme de matrice
//
UM=matrix(U,N,N);
//
// Représentation sous forme de surface
//
surf(X,Y,UM)
//
// sous forme de courbes de niveau
//
contour(x,y,UM,10)
//
// en mêlant surface et courbes de niveau
//
plot3d(x,y,UM)
contour(x,y,UM,20,flag=[0 2 4]);

```