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

Différences finies en Scilab

1 Compléments Scilab

1.1 Construction des matrices

- Matrice de $-\dfrac{\partial^2}{\partial x^2}$ 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.

Exercice 1
Construire les matrices en Scilab pour des petites valeur de N.

1.2 Visualisation des matrices

La façon la plus simple de visualiser une matrice est de la représenter comme on le fait habituellement, c'est à dire sous forme tableau. Cela n'est cependant adapté qu'aux matrices de petites tailles. Les matrices obtenues par discrétisation d'opérateurs locaux, par exemple aux dérivées partielles, sont en général dites *creuse* (*sparse* en anglais). Dans ce cas, on représente graphiquement les coefficients non nuls. En Matlab, la fonction `spy` permet de le faire, elle est intégrée au logiciel. Ce n'est pas le cas en Scilab, il faut la construire, on utilisera alors la fonction `spy` donnée en annexe et inspirée de S. Steer.

Exercice 2
Visualiser les matrices précédemment construites. On utilisera la structure :

```
clear;
clf
//
exec('spy.sci');
// construction de la matrice A
spy(A);
```

1.3 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]);
```

2 Résolution numérique de systèmes linéaires

Comme indiqué précédemment, la résolution du système linéaire $Au = F$ s'effectue à l'aide de la commande : `U=A\ F`. La méthode utilisée est celle de l'élimination de Gauss pour les matrices carrées ; lorsque les systèmes sont rectangulaires, la solution est calculée au sens des moindres carrés. Dans le premier cas, on peut calculer la matrice invers A^{-1} par `inv(A)` mais cela est beaucoup plus coûteux en termes de calculs.

Exercice 3
Résoudre le problème de Poisson en dimension 1 et 2. On représentera graphiquement la solution à chaque fois.

Exercice 4 (Visualisation de la régularisation elliptique)
On part d'un second membre très irrégulier

- En dimension 1
  - Donnée aléatoire simple `f=5*(1-2*rand(N,1));`
  - Donnée brownienne
    ```
    dt = T / N;
    dW = sqrt(dt) * rand(N, 1, 'normal');
    f = [cumsum(dW)];
    ```
- En dimension 2 : donnée aléatoire simple `f=5*(1-2*rand(N*N,1));`

Résoudre la suite de problèmes suivants en représentant graphiquement chaque itérée.

$$\begin{array}{l} u^{(0)} = f \\ \text{Pour } k = 0... \\ Au^{k+1} = U^k \end{array}$$

Que remarquez-vous ?

3 Annexe

3.1 Visualisation des éléments non nuls d'une matrice

```
function spy(S)
[nargout,nargin] = argn(0)
//SPY Visualize sparsity pattern.
// SPY(S) plots the sparsity pattern of the matrix S.
// d'après S. STEER

if S==[] then S=0,end
[m,n] = size(S);
stx=max(1,10∧(int(log10(m))))

rect=[0 0 m n]

   [i,j] = find(S);i = i(:);j = j(:);

   plot2d(0,0,-1,'051',' ',rect,[1 m,1 n])

   plot2d(j,m-i,-3,'000');
xtitle('nz = '+string(nnz(S)),' ',' ');

   a=gca();
a.axes_visible =["off","off","off"];// Les axes sont effacés
endfunction
```