École Internationale des Sciences du Traitement de l'Information

Analyse Numérique
TP 1
Résolution de systèmes linéaires

Groupe D :

LANZERAY Alexandre
GIBERT Aurélien

Cergy, le 12 avril 2012

SOMMAIRE

Partie A : Construction des Matrices avec Scilab

L'objectif de la première partie est de suivre étape par étape la construction de matrice à l'aide du logiciel Scilab.

Dans un premier temps on commence par construire une matrice diagonale pour différentes valeurs de k et de n, que l'ont peut changer dans la phase de test du code Scilab.

```
-->d
 d  =


       column 1 to 7

   1.    0.                        0.                        0.                        0.                        0.                        0.
   0.    0.7742636826811271077986  0.                        0.                        0.                        0.                        0.
   0.    0.                        0.7498942093324558744172  0.                        0.                        0.                        0.
   0.    0.                        0.                        0.7196856730011520486556  0.                        0.                        0.
   0.    0.                        0.                        0.                        0.6812920690579612470472  0.                        0.
   0.    0.                        0.                        0.                        0.                        0.6309573444801932495807  0.
   0.    0.                        0.                        0.                        0.                        0.                        0.5623413251903490728267
   0.    0.                        0.                        0.                        0.                        0.                        0.
   0.    0.                        0.                        0.                        0.                        0.                        0.
   0.    0.                        0.                        0.                        0.                        0.                        0.

       column  8 to 10

   0.                        0.                        0.
   0.                        0.                        0.
   0.                        0.                        0.
   0.                        0.                        0.
   0.                        0.                        0.
   0.                        0.                        0.
   0.                        0.                        0.
   0.4641588833612779185778  0.                        0.
   0.                        0.3162277660168379411765  0.
   0.                        0.                        0.1000000000000000055511
```

Dans cet exemple nous avons tester la matrice d avec comme valeurs :

$$k = 10 \text{ et } n = 10$$

On fait de même avec k = 1000, 10^8 et 10^16, et n = 50, 100 et 150. Ainsi on peut construire la matrice que j'ai appelé T dans mon Scilab. On peut modifié la dimension de la matrice en changeant la valeurs de m, exemple avec m = 10 :

```
-->t
 t  =

    2.  - 1.    0.    0.    0.    0.    0.    0.    0.    0.
  - 1.    2.  - 1.    0.    0.    0.    0.    0.    0.    0.
    0.  - 1.    2.  - 1.    0.    0.    0.    0.    0.    0.
    0.    0.  - 1.    2.  - 1.    0.    0.    0.    0.    0.
    0.    0.    0.  - 1.    2.  - 1.    0.    0.    0.    0.
    0.    0.    0.    0.  - 1.    2.  - 1.    0.    0.    0.
    0.    0.    0.    0.    0.  - 1.    2.  - 1.    0.    0.
    0.    0.    0.    0.    0.    0.  - 1.    2.  - 1.    0.
    0.    0.    0.    0.    0.    0.    0.  - 1.    2.  - 1.
    0.    0.    0.    0.    0.    0.    0.    0.  - 1.    2.
```

On cherche à présent à calculer A, on tape donc sur Scilab la commande :

[U,S,V]=svd(t)

qui va faire une décomposition en valeurs singulière de la matrice.

Enfin, on calcul x en utilisant la fonction "resolv" en remplaçant les valeurs J par la matrice t et la dimension n par la même dimension que t soit dans notre exemple : 10.

x=resolv(t,10)

Partie B : Calculs à effectuer avec Scilab

L'Objectif de cette partie est de calculer l'erreur de deux méthode différente :

=> Méthode de Jacobi

=> Méthode de "linsolve"

Dans un premier temps on construit la fonction Jacobi afin d'utiliser la méthode itérative de Jacobi :

$$function[fctJacobi] = Jacobi(k, A, n)$$

avec A, la matrice de départ et n sa dimention.

Auparavant on aura calculer l'erreur gràce à la méthode de "linsolve", et par la suite on calcul l'erreur par la méthode de Jacobi avec la fonction :

$$function[ErJ] = ErreurJacobi(i, A, N)$$

Enfin on calcul de l'erreur relative en faisant la division de l'erreur par la méthode jacobi sur l'erreur par la méthode de "linsolve" :

$$ErreurRelative = ErJacobi/Erreur$$

Pour conclure, on souhaite que l'erreur résiduelle se rapproche de la valeurs de "seuil", c'est pourquoi on utilise la fonction :

$$function[ibis] = iterationJacobiBis(A, n, k)$$

qui va génerer un plus grand nombre d'itération.

Partie C : Erreur et nombre d'itérations

On cherche dans cette troisième partie à calculer tout d'abord le rayon spectrale de la matrice A :

```
function [Lambda]=Spectre(A)
D=diag(diag(A)) ;
L=tril(A)-D ;
U=triu(A)-D ;
Rj=-inv(D)*(L+U) ;
Lambda=abs(spec(Rj)) ;
endfunction
```

Ensuite de la même maniére que dans la partie précédente on trie les valeurs propres dans l'ordre décroissant.

Pour finir on cherche le nombre d'itération m nécessaire pour faire diminuer l'erreur, pour cela on utilise la fonction :

```
function [m]=mLambda(LambMax)
m=round(abs(8/log(LambMax)))
endfunction
```

Partie D : Relaxation

Pour finir sur ce TP, nous allons voir en quoi peut influencer la relaxation. Pour cela nous allons construire une fonction qui va nous permettre d'obtenir les valeurs des valeurs propres de Rj (en valeur absolu) ou Rj est la matrice d'itération de Jacobi.

$$function[Jacw] = Jacobiw(k, A, N, w)$$

On va par la suite à partir des fonction "LambdaMaxN et "LambdaMin1" pouvoir comparer notre valeur de relaxation choisit au départ avec les valeurs calculés via ces lambdas.
En prenant w comme relaxation, on va pouvoir utiliser la fonction :

```
function [w]=Validite(LambdaMin,LambdaMax)
w=2/(2-LambdaMin-LambdaMax)
endfunction
```