Université de Picardie Jules Verne
Préparation à l'Agrégation de Mathématiques
Option Calcul Scientifique et modélisation
2019-2020

Systèmes linéaires avec Scilab - TP

Conditionnement

Exercice 1 (Conditionnement)

1. On considère la matrice
$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1.0001 \end{pmatrix}$$
   (a) Résoudre les systmès $Au = b$ avec $b = (2, 2)^T$ puis $b = (2, 2.0001)^T$. Qu'observe-t-on ?
   (b) Calculer le conditionement de A

2. Soit la matrice
$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}.$$
   L'inverse de cette matrice est particulièrement simple :
$$A^{-1} = \begin{pmatrix} 25 & -41 & 10 & -6 \\ -41 & 68 & -17 & 10 \\ 10 & -17 & 5 & -3 \\ -6 & 10 & -3 & 2 \end{pmatrix}.$$
   (a) Calculer la solution de $Au = b$ pour $b = (32, 23, 33, 31)^t$, puis pour $Av = b + \delta b$ avec $\delta b = (0.1,\ -0.1,\ 0.1,\ -0.1)^t$. qu'observe-t-on ?
   (b) Calculer le conditionnement de A.

3. Matrice de Hilbert. Ce problème trouve son origine dans l'approximation polynomiale au sens des moindres carrés.

Soit f une fonction continue sur l'intervalle $[a, b]$. On désire déterminer le polynôme P de degré inferieur ou égal a n tel que la quantité

$$\int_a^b (f(x) - P(x))^2 dx$$

soit minimale. On démontre facilement qu'il existe un unique polynôme P ayant cette propriéte et que P est caracterisé par

$$\int_a^b q(x)(f(x) - p(x))dx = 0, \forall q \in P_n \ (R)$$

Maintenant cherchons à déterminer P. P peut s'écrire sous la forme

$$P(x) = \sum_{i=0}^{n} a_i x^i$$

Les monômes $ei(x) = x^i$ forment une base de l'espace vectoriel des polynômes de degrê inferieur ou égal à n, si bien que la relation (R) est équivalente à

$$\sum_{i=0}^{n} \left(\int_0^1 ei(x).ej(x)dx \right) a_i = \int_0^1 f(x).ej(x)dx = F_j \ \forall j = 0, \cdots, n.$$

Les nombres a_i sont donc solution du système linéaire

$$H.a = F$$

où $a = (a_0, \cdots, a_n)^t, , F = (F_0, \cdots, F_n)^t)$ et où H est la matrice (symétrique) de coefficients

$$H_{i,j} = \int_0^1 ei(x).ej(x)dx = \frac{1}{i+j-1}, i, j = 1, \cdots, n+1$$

H est la matrice de Hilbert. On montre que $\kappa(H) \simeq e^{7n/2}$ pour n grand !

(a) Soit $f(x) = e^x$. On fixe $n = 4$. Calculer les coefficients de F. Résoudre le système aec Scilab et représenter graphiquement f et P.

(b) Remplacer F par $F + 0.0001\ F$. Résoudre le nouveau système et représenter f et le (nouveau) polynôme P. Qu'observe-t-on ?

(c) calculer Cond(H).

Méthodes Directes

Exercice 2 (Cholesky et LU sur des exemples simples)
On construit la matrice SDP A comme suit : `N=5;`

```
Id=eye(N,N);
A=rand(N,N);
A=A'*A;
r=norm(A,'inf');
A=r*Id+A;
```

1. Que retourne la liste de commandes ci-dessus ?
2. Montrer que la matrice A ainsi construite est SDP. Calculer numériquement les dcomposition LU et de Cholesky de A

Exercice 3 (Remplissage des Matrices)
1. Pour la matrice pentadiagonale habituelle A, construire ses facteurs de Cholesky, les visualiser à l'aide de la fonction `spy` donnée en annexe. Que constatez-vous ?
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 4 (Factorisation incomplète à l'aide d'un point fixe)
Cette méthode a ét/'e proposée dans l'articlce Updating Incomplete Factorization Preconditioners for Model Order Reduction par Hartwig Anzt, Edmond Chow, Jens Saak et Jack Dongarra[1]. 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 $L^0 = (D-E)D^{-1}$, $U^0 = D - F$.

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

Programmer cette méthode. Que constatez-vous ?

[1] Article paru dans la revue Numerical Algorithm, November 2016, Volume 73, Issue 3, pp 611–630

Méthodes Itératives

Exercice 5 (Programmation générale)

1. Construire un seul programme principal Scilab pour les méthodes itératives classiques.

2. Soit la matrice
$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}.$$
Résoudre le système linéaire $Ax = b$ avec `b=ones(n,1)` avec les méthodes de Jacobi, Gauss-Seidel et relaxation ; on partira à chaque fois de la donnée initiale `x=zeros(n,1)`. On rangera dans un vecteur la suite des normes des résidus et on calculera le temps de calcul à chaque fois.

3. Comparer l'historique des résidus pour ces méthodes.

4. Conclusion ?

Exercice 6 (Méthode de Jacobi pour les systèmes triangulaires)

1. Soit $A = D - F$ $n \times n$, triangulaire supérieure, inversible, on note $D = diag(A)$. Montrer que $J = D^{-1}F$ est nilpotente.

2. En déduire que la méthode de Jacobi converge en au plus n itérations.

3. Illustration sur un exemple de votre choix. On représentera graphiquement l'historique des résidus.

Exercice 7 (Méthode SSOR (Symmetric Successive Over Relaxation))
Soit $A = D + L + L^T$ une matrice symétrique. On considère la matrice M :
$$M = (D + L)D^{-1}(D + L)^T$$
et plus généralement
$$M(\omega) = \frac{\omega}{2 - \omega}\left(\frac{1}{\omega}D + L\right)(D)^{-1}\left(\frac{1}{\omega}D + L\right)^T$$

1. Programmer cette méthode pour différentes valeurs de $\omega \in [1, 2[$. On calculera le nombre d'ítérations nécessaires à la convergence.

2. Que constatez-vous ?

Exercice 8 (Directions alternées)

1. On rappelle la propriété $(A \bigotimes B)(C \bigotimes D) = AC \bigotimes BD$.
Soit A une admettant une décomposition LU. Montrer que $(Id \bigotimes A) = (Id \bigotimes L)(Id \bigotimes U)$.

2. Soit A une matrice SDP et $M = Id \bigotimes A + A \bigotimes Id = K_1 + K_2$. On considère la méthode itérative

Algorithm 1 Méthode des directions alternées

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 : $(Id + \frac{\alpha}{2}K_1)u^{(k+1/2)} = (Id - \frac{\alpha}{2}K_2)u^{(k)} + \frac{\alpha}{2}b$
5: Résoudre : $(Id + \frac{\alpha}{2}K_2)u^{(k+1)} = (Id - \frac{\alpha}{2}K_1)u^{(k+1/2)} + \frac{\alpha}{2}b$
6: Poser : $r^{(k)} = b - Mu^{(k+1)}$
7: Poser : $k = k + 1$
8: **end while**

Ici $\alpha > 0$. Exprimer $u^{(k+1)}$ en fonction de $u^{(k)}$ sous la forme $u^{(k+1)} = Gu^{(k)} + c$.

3. En déduire une CNS de convergence et montrer que si $u^{(k)}$ converge, c'est vers la solution de $Mu = b$.

4. Montrer que K_1 et K_2 commutent. Montrer que $\rho(G) < 1$ et conclure.

5. Programmation de la méthode en Scilab

   (a) Pour $\alpha > 0$ fixé, calculer les décompositions de Cholesky de $Id + \frac{\alpha}{2}K_1$ et de $Id + \frac{\alpha}{2}K_2$.

   (b) Programmer la méthode des directions alternée pour la matrice obtenue avec
   `A=2*eye(N,N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1)`

Exercice 9 (Variations autour de la méthode de Richardson)
Soit A la matrice de discrétisation de $-\Delta$ avec conditions aux limites de Dirichlet homogènes dur le carré unité $\Omega =]0,1[^2$.

1. Programmer la méthode de Richardson avec paramètre constant, puis avec le paramètre optimal ($\alpha_{opt} = \arg\min_\alpha \rho(Id - \alpha A)$)

2. Programmer la méthode de plus profonde descente ($\alpha_k = \dfrac{< r^{(k)}, r^{(k)}}{< Ar^{(k)}, r^{(k)} >}$ avec $r^{(k)} = b - Ax^{(k)}$).

3. Programmer variation suivante de cette dernière méthode

$$\begin{array}{ll} \text{Pour} & k \geq 0 \\ \text{Calculer} & \alpha_k = \dfrac{< r^{(k)}, r^{(k)}}{< Ar^{(k)}, r^{(k)} >}, \\ \textbf{Calculer} & \theta_{\mathbf{k}} = \xi_{\mathbf{k}}\alpha_{\mathbf{k}}, \text{avec } \xi \simeq U([0,2] \\ \text{Poser} & x^{(k+1)} = x^{(k)} + \theta_{\mathbf{k}} r^{(k)}, \\ \text{Poser} & r^{(k+1)} = r^{(k)} - \theta_{\mathbf{k}} A r^{(k)}, \end{array}$$

4. Comparer l'historique des résidus pour toutes ces méthodes ;, on partira de $x^{(0)} = 0$. Que constatez-vous ?

Méthodes de Kaczmarz et de Cimmino

Les méthodes suivantes reposent sur l'observation que la solution d'un système linéaire $m \times n$ $Ax = b$, si elle existe, est dans l'intersection des hyperplans affines :

$$\mathcal{H}_i : a_{i,1}x_1 + a_{i,2}x_2 + \cdots a_{i,n}x_n = b_i,\ i = 1, \cdots, m.$$

Bien sûr l'intersection est réduite à un point lorsque A est inversible. Une idée naturelle consiste donc à construire une suite de projetée orthogonales successives sur ces hyperplans affines. Il n'est pas difficile de voir que

$$Proj_{\mathcal{H}_i}(x) = P_i(x)y$$

avec $< a_i, y >= b_i$ et $x - y = \alpha a_i$. Du coup

$$< x, a_i >=< y, a_i > +\alpha\|a_i\|_2^2 = b_i + \alpha\|a_i\|_2^2.$$

Il en découle que $\alpha = \dfrac{< x, a_i > -b_i}{\|a_i\|_2^2}$ et que $P_i(x) = x + \dfrac{b_i - < x, a_i >}{\|a_i\|_2^2}a_i$.

Exercice 10 (Méthode de Kaczmarz)
Soit A une matrice $m \times n$. Soit $b \in \mathbb{R}^m$. On considère le système linéaire

$$Ax = b$$

On désigne par a_i la ième ligne de A.

Algorithm 2 Méthode de Kaczmarz

1: $x^{(0)}$ donné dans $\mathbb{R}^n$
2: **while** $k < Kmax$ & $\|r^{(k)}\| > \epsilon$ **do**
3: Poser : $x^{k+1} = x^k + \dfrac{b_i - \langle a_i, x^k \rangle}{\|a_i\|^2}a_i$
4: (avec $i = k \bmod m, i = 1, 2, ..., m$)
5: Poser : $k = k + 1$
6: **end while**

Cette méthode s'interprète de la manière suivante : x^{k+1} est la projection orthogonale de x^k dans l'hyperplan $< a_i, x >= b_i$, soit

$$x^{k+1} = P_i(x^k).$$

Le taux de convergence est $1 - \dfrac{1}{\kappa^2}$, pour $\kappa = \|A\|\|A^{-1}\|$ assez grand.

On peut généraliser cette méthode en introduisant un paramètre de relaxation λ^k comme suit :

$$x^{k+1} = x^k + \lambda^k \frac{b_i - \langle a_i, x^k \rangle}{\|a_i\|^2}a_i$$

1. Programmer la méthode pour A $m \times n$ de rang maximal ($rang(A) = m \leq n$).
2. Comparer la solution obtenue avec celle au sens des moindres carrés.

Exercice 11 (Méthode de Cimino)
La méthode de Karzmarz utilise les lignes de A séquentiellement, c'est à dire les unes après les autres. La méthode de Cimmino quant à elle utilise les lignes de A simultanément et calcule l'itération courante comme moyenne de toutes les projection de l'itérée précédente, soit

$$x^{k+1} = \frac{1}{m}\sum_{i=1}^{m} P_i(x^k) = x^k + \frac{1}{m}\sum_{i=1}^{m} P_i(x^k) = x^k + \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - \langle a_i, x^k\rangle}{\|a_i\|^2} a_i$$

1. Programmer la méthode pour A $m \times n$ de rang maximal ($rang(A) = m \leq n$).

2. Comparer la solution obtenue avec celle au sens des moindres carrés.

Remarque 1
La méthode de Cimino peut se réécrire comme suit :

$$\begin{aligned}
x^{k+1} &= x^k + \frac{1}{m}\sum_{i=1}^{m} \frac{b_i - \langle a_i, x^k\rangle}{\|a_i\|^2} a_i \\
&= x^k + \frac{1}{m}\left(\frac{a_1}{\|a_1\|_2^2}, \cdots, \frac{a_m}{\|a_m\|_2^2}\right)\begin{pmatrix} b_1 - < a_1, x^k > \\ \vdots \\ b_m - < a_m, x^k > \end{pmatrix} \\
&= x^k + \frac{1}{m}\begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix}^T \begin{pmatrix} \frac{1}{\|a_1\|_2^2} & & \\ & \ddots & \\ & & \frac{1}{\|a_m\|_2^2} \end{pmatrix}\left(b - \begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} x^k\right) \\
&= x^k + A^T M^{-1}\left(b - Ax^k\right)
\end{aligned}$$

1 Annexe

1.1 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
```