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

Méthodes Itératives 1 - Méthodes classiques

1 Commandes Scilab

- Partie triangulaire supérieure : `triu(A)`
- Partie triangulaire inférieure : `tril(A)`
- Calcul du temps de calcul
  ```
  tic
  ⋮
  instructions
  ⋮
  toc
  ```
- Comparaison de l'historique des résidus
  ```
  plot(it1,log(R1)/log(10))
  plot(it2,log(R2)/log(10))
  drawnow
  ```
- 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 aléatroire
  - `speye` : matrice identité creuse
  - `spones` : matrice de 1 creuse
  - `spzeros` : matrice nulle
  - `nnz` : nombre déléments 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

Exercice 1
Effectuer et chronométrer le produit $A * B$ de deux matrices creuses, déclarées comme pleines, puis comme creuses. Que constatez-vous ?

2 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**

Exercice 2 (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 donné initiae `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 3 (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 4 (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'itérations nécessaires à la convergence.

2. Que constatez-vous ?

Exercice 5 (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 2 Méthode des directions alternées

$x^{(0)}$ donné dans $\mathbb{R}^n$
$k = 0$, $r^{(0)} = b - Ax^{(0)}$
while $k < Kmax$ & $\|r^{(k)}\| > \epsilon$ **do**
    Résoudre : $(Id + \frac{\alpha}{2}K_1)u^{(k+1/2)} = (Id - \frac{\alpha}{2}K_2)u^{(k)} + \frac{\alpha}{2}b$
    Résoudre : $(Id + \frac{\alpha}{2}K_2)u^{(k+1)} = (Id - \frac{\alpha}{2}K_1)u^{(k+1/2)} + \frac{\alpha}{2}b$
    Poser : $r^{(k)} = b - Mu^{(k+1)}$
    Poser : $k = k + 1$
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 6 (Méthode de Multi-décompositions)

Soit A une matrice inversible. On considère m couples de matrices (M_i, N_i) avec M_i inversible telles que

$$A = M_i - N_i, \ i = 1, \cdots, m.$$

Soit alors m matrices diagonales à coefficients dans $[0, 1]$ E_i telles que

$$\sum_{i=1}^{m} E_i = Id$$

où Id désigne la matrice identité.

On introduit maintenant la méthode itérative

$$u^{(k+1)} = \sum_{i=1}^{m} E_i M_i^{-1}(N_i u^{(k)} + b) = Su^{(k)} + Tb$$

1. On se place dans le cas $m = 1$. Expliciter la méthode et donner une CNS de convergence.

2. Maintenant $m > 1$.

   (a) Expliciter S et T.

   (b) Montrer que
   $$Id - TA = S.$$

   (c) En déduire une CNS de convergence.

3. On se place dans le cas $m = 2$ et pour simplifier on impose
   $$E_1 = \alpha Id, \quad E_2 = (1-\alpha)Id$$

   (a) Montrer que l'algorithme peut s'écrire sous la forme
   $$u^{(k+1)} = \alpha v^{(k+1)} + (1-\alpha)w^{(k+)}$$
   où $v^{(k+1)}$ et $w^{(k+1)}$ sont à préciser.

   (b) Exprimer le résidu $r^{(k+1)} = b - Au^{(k+1)}$ en fonction de $r_v^{(k+1)} = b - Av^{(k+1)}$ et de $r_w^{(k+1)} b - Aw^{(k+1)}$. Quelle valeur de α permettrait d'obtenir une convergence plus rapide ?

   (c) On modifie maintenant l'algorithme précédent en changeant $\alpha = \alpha_k$ à chaque itération. Calculer α_k de sorte à minimiser $\|r^{(k+1)}\|_2$.

   (d) Montrer qu'avec ce choix
   $$\|r^{(k+1)}\|_2 \leq \|r_v^{(k+1)}\|_2 \text{ et } \|r^{(k+1)}\|_2 \leq \|r_w^{(k+1)}\|_2.$$

   (e) Conclure.

Remarque : l'intérêt de cette méthode provient de ce que les calculs $E_i M_i^{-1}(N_i u^{(k)} + b)$ peuvent être exécutés indépendamment les uns des autres. Cela se réalise avantageusement sur des ordinateurs "parallèles", c'est à dire avec plusieurs processeurs, chacun ayant un calcul à executer "en même temps" indépendemment des autres. Le résultat final est assemblé une fois tous les calculs terminés.

3 Méthodes particulières

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 7 (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 3 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 8 (Méthode de Cimino)
La méthode de Karzmarz utilise les lignes de A séquentiellement, c'est à dire les unes après les autres. L a méthde 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 :

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