Analyse numérique matricielle

Élimination de Gauss, factorisation LU et applications

L3 Mathématiques - Université d'Évry

L'objet de ce TD est d'utiliser les méthodes élémentaires de l'analyse numérique matricielle pour résoudre des systèmes linéaires simples. L'implémentation est faite à l'aide du logiciel Scilab.

1 Rappels de cours

Dans la suite, on considère une matrice carrée $\mathbf{A} = (a_{ij})_{i,j=1,\dots,n}$ de $\mathcal{M}_{nn}(\mathbb{R})$ supposée inversible. On cherche à résoudre le système linéaire

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{1}$$

pour $\mathbf{b} = (b_1, b_2, \dots, b_n)^\intercal \in \mathbb{R}^n$ donné. Résoudre (1) signifie déterminer $\mathbf{x} = (x_1, x_2, \dots, x_n)^\intercal \in \mathbb{R}^n$.

1.1 Résolution des systèmes triangulaires

Un cas facile à traiter est celui des systèmes triangulaires. Lorsque $\mathbf{A}$ est une matrice triangulaire inférieure, c'est-à-dire que $a_{ij} = 0$ pour tout $j > i$, il est très facile de voir que

$$\left\{ \begin{array}{rcl} x_1 & = & \dfrac{b_1}{a_{11}}, \\ x_i & = & \dfrac{1}{a_{ii}} \left(b_i - \displaystyle\sum_{j=1}^{i-1} a_{ij} x_j \right), \qquad i = 2, \dots, n. \end{array} \right. \tag{2}$$

Lorsque l'on a un système triangulaire supérieur, c'est-à-dire lorsque $a_{ij} = 0$ pour tout $i > j$, alors l'algorithme de résolution fonctionne par parcours inverse et devient

$$\left\{ \begin{array}{rcl} x_n & = & \dfrac{b_n}{a_{nn}}, \\ x_i & = & \dfrac{1}{a_{ii}} \left(b_i - \displaystyle\sum_{j=i+1}^{n} a_{ij} x_j \right), \qquad i = n-1, \dots, 1. \end{array} \right. \tag{3}$$

1.2 Triangulation par élimination de Gauss

La résolution des systèmes triangulaires est facile et peu coûteuse numériquement. L'idée est donc de proposer des algorithmes qui transforment un système linéaire général (1) en un système équivalent

$$\widetilde{\mathbf{A}}\mathbf{x} = \widetilde{\mathbf{b}}, \tag{4}$$

où $\widetilde{\mathbf{A}}$ est triangulaire supérieure, issue de $\mathbf{A}$.

L'algorithme d'élimination de Gauss permet de trianguler la matrice $\mathbf{A}$. Il comporte n étapes de transformation. On note $\mathbf{A}^{(k)}$ l'état de la matrice transformée à la k^{e} étape. La matrice $\widetilde{\mathbf{A}}$ recherchée correspond à $\mathbf{A}^{(n)}$. On initialise l'algorithme avec $\mathbf{A}^{(1)} = \mathbf{A}$, puis on calcule les étapes $k = 2, \ldots, n$ à l'aide de la relation de récurrence définie pour $i = k+1, \ldots, n$ par

$$\left\{ \begin{array}{rcll} a_{ij}^{(k+1)} & = & a_{ij}^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} a_{kj}^{(k)}, & j = k, \ldots, n, \\ b_i^{(k+1)} & = & b_i^{(k)} - \dfrac{a_{ik}^{(k)}}{a_{kk}^{(k)}} b_k^{(k)}. & \end{array} \right. \tag{5}$$

D'où l'algorithme 1.

Algorithme 1 : Algorithme d'élimination de Gauss

Entrées : $\mathbf{A}, \mathbf{b}$
pour $k = 1, \ldots, n-1$ **faire**
  `// On teste si le pivot est nul`
  si $|a_{kk}| < \varepsilon$ **alors**
    Afficher un message d'erreur
  fin
  sinon
    `//Calcul de` $\mathbf{A}^{(k)}$
    pour $i = k+1, \ldots, n$ **faire**
      $c \leftarrow \frac{a_{ik}}{a_{kk}}$
      $b_i \leftarrow b_i - c \times b_k$
      $a_{ik} \leftarrow 0$
      pour $j = k+1, \ldots, n$ **faire**
        $a_{ij} \leftarrow a_{ij} - c \times a_{kj}$
      fin
    fin
  fin
fin
$\widetilde{\mathbf{A}} \leftarrow \mathbf{A}$
$\widetilde{\mathbf{b}} \leftarrow \mathbf{b}$
Sorties : $\widetilde{\mathbf{A}}, \widetilde{\mathbf{b}}$

1.3 Factorisation LU

Supposons que l'on dispose d'une factorisation telle que $\mathbf{A} = LU$ ou L est une matrice triangulaire inférieure et U supérieure. Alors le système général (1) s'écrit

$$LUx = \mathbf{b}, \tag{6}$$

que l'on résout en traitant successivement les systèmes triangulaires

$$\left\{ \begin{array}{lcl} L\mathbf{y} & = & \mathbf{b}, \\ U\mathbf{x} & = & \mathbf{y}. \end{array} \right. \tag{7}$$

L'algorithme suivant, appelé algorithme de Doolittle, permet d'obtenir cette factorisation

Algorithme 2 : Algorithme de Doolittle

Entrées : $\mathbf{A}$

$L \leftarrow I_{nn}$

$U \leftarrow 0_{nn}$

pour $i = 1, \ldots, n-1$ **faire**

  pour $j = i, \ldots, n$ **faire**

    $u_{ij} \leftarrow a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}$

  fin

  pour $j = i+1, \ldots, n$ **faire**

    $l_{ji} \leftarrow \frac{1}{u_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} u_{ki} \right)$

  fin

fin

$u_{nn} \leftarrow a_{nn} - \sum_{k=1}^{n-1} l_{nk} u_{kn}$

Sorties : L, U

2 Applications numériques

Questions

1. Systèmes triangulaires

   a) Écrire une fonction `[x]=solinf(L,b)` qui résout un système triangulaire inférieur $L\mathbf{x} = \mathbf{b}$. Tester sur le système

$$\begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 1 & 4 & -1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 8 \\ 10 \end{bmatrix}.$$

b) Écrire une fonction `[x]=solsup(U,b)` qui résout un système triangulaire supérieur $U\mathbf{x} = \mathbf{b}$. Tester sur le système

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & 4 & 8 \\ 0 & 0 & 5 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 6 \\ 16 \\ 15 \end{bmatrix}.$$

2. Élimination de Gauss

a) Écrire une fonction `[At,bt]=trigGauss(A,b)` qui renvoie une matrice triangulaire supérieure $\widetilde{\mathbf{A}}$ et un vecteur $\widetilde{\mathbf{b}}$ tels que $\mathbf{A}x = \mathbf{b} \Leftrightarrow \widetilde{\mathbf{A}}x = \widetilde{\mathbf{b}}$. Tester sur le système

$$\begin{bmatrix} 3 & 1 & 2 \\ 3 & 2 & 6 \\ 6 & 1 & -1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 2 \\ 1 \\ 4 \end{bmatrix}.$$

b) Écrire une fonction `[x]=ResolutionGauss(A,b)` qui utilise `trigGauss` et `solsup` pour résoudre le système $\mathbf{A}\mathbf{x} = \mathbf{b}$ dans le cas général. Tester sur le système

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 1 \\ 3 & -1 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 5 \\ 5 \\ 6 \end{bmatrix}.$$

3. Factorisation LU

a) Écrire une fonction `[L,U]=FactLU(A)` qui implémente la méthode de Doolittle pour la factorisation LU. Tester sur la matrice

$$\begin{bmatrix} 3 & 1 & 2 \\ 3 & 2 & 6 \\ 6 & 1 & -1 \end{bmatrix}.$$

b) Écrire une fonction `[x]=ResolutionLU(A,b)` qui utilise `FactLU`, `solsup` et `solinf` pour résoudre $\mathbf{A}\mathbf{x} = \mathbf{b}$. Tester sur le système

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 1 \\ 3 & -1 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 5 \\ 5 \\ 6 \end{bmatrix}.$$

4. Application à l'inversion d'une matrice

Écrire une fonction `[B]=inverse(A)` qui calcule l'inverse d'une matrice $\mathbf{A}$ de $n \times n$ en résolvant n systèmes linéaires. Tester sur la matrice

$$\begin{bmatrix} 1 & 2 & 3 \\ 5 & 2 & 1 \\ 3 & -1 & 1 \end{bmatrix}.$$