Travaux dirigés Produit matriciel n°4

Mathématiques pour l'informatique

—IMAC 2—

▸ Exercice 1. *Gauss-Seidel*

1. *La méthode de Gauss-Seidel est une méthode itérative dédiée à la résolution de systèmes linéaires. La condition nécessaire et suffisante de sa convergence est que, en considérant le système linéaire* $\mathrm{A}\mathbf{x} = \mathbf{b}$*, la matrice* A *soit à diagonale dominante stricte :*

$$|a_{kk}| > \sum_{i=1, i\neq k}^{n} |a_{ki}|$$

Le système suivant satisfait-il les critères de convergence ?

$$\begin{bmatrix} 4 & 0 & -2 \\ 2 & 0 & 7 \\ -5 & 1 & -8 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -2 \\ 8 \\ 3 \end{pmatrix}$$

2. *Conditionner le système suivant afin que la matrice soit à diagonale dominante stricte.*

$$\begin{bmatrix} 3 & 1 & -1 \\ -1 & 0 & 4 \\ 1 & 4 & 2 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 12 \\ -1 \\ -1 \end{pmatrix}$$

3. *Pour nos tests à venir, nous souhaitons générer des matrices aléatoires à diagonale dominante stricte. Une possibilité consiste à générer une matrice* M *à coefficients aléatoires sur* $[-1, 1]$*, puis à générer une matrice* A *telle que :*

$$\mathrm{A} = \mathrm{M} + \alpha \mathrm{Id}$$

où α *est un nombre suffisamment grand. Pour une matrice* A *carrée d'ordre* n*, quelle valeur minimale de* α *peut-on choisir? Codez une fonction qui génère une matrice à diagonale dominante stricte de taille* $n \times n$*.*

4. *La méthode de Gauss-Seidel fonctionne en itérant sur le schéma suivant :*

$$\begin{cases} 5x - 3y + z = 10 \\ -x + 6y - 4z = 5 \\ 3x - 3y + 9z = -7 \end{cases} \rightarrow \begin{cases} x_{n+1} = \dfrac{10 + 3y_n - z_n}{5} \\ y_{n+1} = \dfrac{5 + x_{n+1} + 4z_n}{6} \\ z_{n+1} = \dfrac{-7 - 3x_{n+1} + 3y_{n+1}}{9} \end{cases}$$

où n *correspond au numéro d'itération. Codez la méthode de Gauss-Seidel avec la bibliothèque Eigen avec le prototype suivant :*

```
Eigen::VectorXd gaussSeidel(const Eigen::MatrixXd &A,
                            const Eigen::VectorXd &b,
                            const uint nbIter)
```

et testez la avec les matrices des questions précédentes. Combien d'itérations sont nécessaires pour obtenir un résultat satisfaisant?

5. *Donner la complexité de cette méthode.*

6. *On peut écrire la méthode de Gauss-Seidel matriciellement. Dans le système $A\boldsymbol{x} = \boldsymbol{b}$ à résoudre, on pose $A = D+E+F$ où D est la diagonale de A, E est la partie triangulaire supérieure de A et F sa partie triangulaire inférieure (E et F ont une diagonale nulle). On obtient :*

$$(D + E + F)\boldsymbol{x} = \boldsymbol{b}$$

$$D\boldsymbol{x} + E\boldsymbol{x} + F\boldsymbol{x} = \boldsymbol{b}$$

$$(D + E)\boldsymbol{x} = -F\boldsymbol{x} + \boldsymbol{b}$$

$$\boldsymbol{x} = -(D + E)^{-1}F\boldsymbol{x} + (D + E)^{-1}\boldsymbol{b}$$

Codez cette version matricielle et vérifiez que vous obtenez le même résultat qu'avec la version précédente.

▸ Exercice 2. *Pivot de Gauss : erreurs numériques*

Soit le système

$$\left(\begin{array}{cc|c} \epsilon & 1 & 1 \\ 1 & 1 & 2 \end{array}\right)$$

où $|\epsilon| \ll 1$

1. *Résoudre manuellement ce système avec le pivot de Gauss standard en utilisant :*
   - *un calcul algébrique exact (expression formelle du résultat).*
   - *le calcul numérique que ferait la machine, en y incluant les erreurs numériques successives.*
2. *Mêmes questions avec le pivot partiel.*

▸ Exercice 3. *Mises en situation*

Voici une liste de situations où l'on utilise des méthodes numériques pour résoudre des problèmes. Choisir pour chaque cas la méthode la plus adaptée en justifiant vos choix.

1. *Synthèse d'images et illumination globale : la radiosité. Il faut résoudre des systèmes mettant en jeu de très grosses matrices tri-diagonales.*
2. *La météo civile.*

3. *La météo militaire.*

4. *Un logiciel d'assistance au pilotage d'un avion.*

5. *Dans un dispositif temps réel (jeu vidéo par exemple) où l'on traite des matrices* 3×3.

6. *Dans un dispositif temps réel où l'on traite des matrices* 500×500.

7. *Dans le cas où l'on traite uniquement des matrices triangulaires.*

8. *Dans le cas où les données sont nombreuses, mais de mauvaise qualité.*

▸ **Exercice 4.** ***Comparatif***

Dans cet exercice, nous souhaitons comparer (en temps et en précision) différentes méthodes de résolutions de systèmes linéaires.

1. *Générez un système matriciel* $A\mathbf{x} = \mathbf{b}$ *aléatoire. Ce système n'a une solution que si la matrice* A *est inversible, c-à-d de rang plein (rang* $= n$*). La bibliothèque Eigen permet de calculer le rang d'une matrice en définissant par exemple la fonction suivante :*

```
unsigned int getRank(const Eigen::MatrixXd &A){
    Eigen::FullPivLU<Eigen::MatrixXd> lu_decomp(A);
    return lu_decomp.rank();
}
```

   *Créez un système linéaire aléatoire (*A *et* $\mathbf{b}$*) et vérifiez que sa matrice* A *est de rang plein.*

2. *Pour estimer la précision d'un solver, il suffit de tester la solution trouvée avec :*

$$\epsilon = \|\mathbf{b} - \mathrm{A}\mathbf{x}\|$$

   Codez un comparatif de précision et de rapidité pour les méthodes suivantes :

   - *Gauss-Seidel (codé par vos soins)*
   - *LU avec le code :*

```
Eigen::PartialPivLU<Eigen::MatrixXd> lu(A);
Eigen::VectorXd x_lu = lu.solve(b);
```

   - *QR avec le code :*

```
Eigen::ColPivHouseholderQR<Eigen::MatrixXd> qr(A);
Eigen::VectorXd x_qr = qr.solve(b);
```

   - *SVD avec le code :*

```
Eigen::JacobiSVD<Eigen::MatrixXd> svd(A,
                        Eigen::ComputeThinU | Eigen::ComputeThinV);
Eigen::VectorXd x_svd = svd.solve(b);
```

3. *Comparer le tout avec la méthode de réduction d'erreur de la méthode LU :*

$$
\begin{array}{lcl}
\mathtt{A}\mathbf{x} = \mathbf{b} & \xrightarrow{LU} & \mathbf{x}_0 \\
\mathtt{A}\mathbf{x}_0 = \mathbf{b}_0 \simeq \mathbf{b} & & \\
\mathtt{A}(\mathbf{x}_0 + \delta\mathbf{x}) = \mathbf{b} & \Leftrightarrow & \mathtt{A}\mathbf{x}_0 + \mathtt{A}\delta\mathbf{x} = \mathbf{b} \\
\mathtt{A}\delta\mathbf{x} = \mathbf{b} - \mathtt{A}\mathbf{x}_0 & &
\end{array}
$$

résoudre le système linéaire précédente en utilisant la décomposition LU de $\mathtt{A}$ *déjà calculée dans* $\mathtt{A}\mathbf{x}_0 = \mathbf{b}_0$. *La solution améliorée du système* $\mathtt{A}\mathbf{x} = \mathbf{b}$ *est alors :*

$$\mathbf{x} = \mathbf{x}_0 + \delta\mathbf{x}$$